Computability Theory and Some Applications

by

Michael Deveau

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Pure Mathematics

Waterloo, Ontario, Canada, 2019

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Valentina Harizanov Professor, Dept. of Mathematics, George Washington University
Supervisor:	Barbara Csima Professor, Dept. of Pure Mathematics, University of Waterloo
Internal Members:	Ross Willard Professor, Dept. of Pure Mathematics, University of Waterloo Jason Bell Professor, Dept. of Pure Mathematics, University of Waterloo
Internal-External Member:	Jonathan Buss Associate Professor, Dept. of Computer Science, University of Waterloo

Statement of Contributions

I am the sole author of chapters 1, 4 and 5. Chapter 2 is joint work with Barbara Csima, Matthew Harrison-Trainor and Mohammad Assem Mahmoud. Chapter 3 is joint work with Barbara Csima and Jonny Stephenson.

Abstract

We explore various areas of computability theory, ranging from applications in computable structure theory primarily focused on problems about computing isomorphisms, to a number of new results regarding the degree-theoretic notion of the bounded Turing hierarchy.

In Chapter 2 (joint with Csima, Harrison-Trainor, Mahmoud), the set of degrees that are computably enumerable in and above $\mathbf{0}^{(\alpha)}$ are shown to be degrees of categoricity of a structure, where α is a computable limit ordinal. We construct such structures in a particularly useful way: by restricting the construction to a particular case (the limit ordinal ω) and proving some additional facts about the widgets that make up the structure, we are able to produce a computable prime model with a degree of categoricity as high as is possible. This then shows that a particular upper bound on such degrees is exact. This joint work appears in [9].

In Chapter 3 (joint with Csima and Stephenson), a common trick in computable structure theory as it relates to degrees of categoricity is explored. In this trick, the degree of an isomorphism between computable copies of a rigid structure is often able to be witnessed by the clever choice of a computable set whose image or preimage through the isomorphism actually attains the degree of the isomorphism itself. We construct a pair of computable copies of $(\omega, <)$ where this trick will not work, examine some problems with decidability of the structures and work with $(\omega^2, <)$ to resolve them by proving a similar result.

In Chapter 4, the effectivization of Walker's Cancellation Theorem in group theory is discussed in the context of uniformity. That is, if we have an indexed collection of instances of sums of finitely generated abelian groups $A_i \oplus G_i \cong A_i \oplus H_i$ and the code for the isomorphism between them, then we wish to know to what extent we can give a *single* procedure that, given an index i, produces an isomorphism between G_i and H_i.

Finally, in Chapter 5, several results pertaining to the bounded Turing degrees (also known as the weak truth-table degrees) and the bounded jump are investigated, with an eye toward jump inversion. We first resolve a potential ambiguity in the definition of sets used to characterize degrees in the bounded Turing hierarchy. Then we investigate some open problems related to lowness and highness as it appears in this realm, and then generalize a characterization about reductions to iterated bounded jumps of arbitrary sets. We use this result to prove the non-triviality of the hierarchy of successive applications of the bounded jump above any set, showing that the problem of jump inversion must be non-trivial if it is true in any relativized generality.

Acknowledgements

I would like to thank Dr. Barbara Csima, without whom this thesis would not have been remotely possible. Your knowledge, guidance, and patience were all instrumental, and I believe that without it, I could not have completed this degree. Giving me the opportunity to see as much of computability theory as I did was excellent, and encouraging me to step outside of my comfort zone with the learning seminars, conferences and talks could not have prepared me more for this. And of course, who could forget your always-relevant life advice? I cannot begin to thank you enough and I can only hope every supervisor is as supportive and helpful as you.

I would also like to thank my examining committee, Dr. Valentina Harizanov, Dr. Jonathan Buss, Dr. Ross Willard, and Dr. Jason Bell. Your hard work and piercing insight makes this thesis better than I could have on my own, and your attention to detail in your feedback and questions made them invaluable. I am forever grateful. I would also like to thank Dr. Russell Miller for his role in my oral comprehensive, preparing me for the latter half of this degree.

To my colleagues, Dr. Matthew Harrison-Trainor, Dr. Jonny Stephenson, and the newly minted Dr. Mohammad Assem Mahmoud: Working with you has been one of the best environments a graduate student could hope for. Your advice, ideas, and intelligence made for an amazing experience in this field of study and made collaboration so fruitful.

To the faculty, staff and others I relied on in the Pure Mathematics department, the Faculty of Mathematics, and the University of Waterloo as a whole: Without your dedication and tireless effort, I would not be here. There is no other place like this in the world. I could not have chosen a better institution to study mathematics.

I would like to thank my friends for supporting me (and keeping me sane). Thanks in particular to Robie Hennigar, whose conversations and advice helped me immeasurably.

Finally, special thanks to my family. To my parents, for instilling a passion for this subject in me at a young age, encouraging me to learn and practice, and helping me become who I am today academically and, more importantly, as a person. To my brother, for our discussions and time spent together when I was home. To my grandparents, aunts and uncles, and the rest of my extended family, it has been through your kind words, thoughts and prayers that I find motivation. I only wish I could have visited more often – and maybe done a better job explaining exactly what my research is! Thanks to all of you for your love and support.

For my father

Table of Contents

List of Figures

Chapter 1

Introduction

Although the subjects we investigate in each chapter of this thesis are relatively disparate, it is worth beginning with an overview of the notation that we will use throughout. We mostly follow the standard reference text Soare [28] in the notations that we use and conventions that we follow. We write $\varphi_0, \varphi_1, \ldots$ as some enumeration of the partial computable functions. The exact method for producing such a list are unimportant to us, however. More generally, for a given set X, we write Φ_e^X to refer to the eth Turing functional with oracle X. If the index e of the functional is not important, we may sometimes represent the functional without it, such as Γ^X or Ψ^X. The amount of X that is used by a certain computation is called the *use* of the computation, and we write the use of the computation $\Phi_e^X(y)$ as $u_e^X(y)$ or $\mathrm{use}_e^X(y)$. If we have a functional that lacks an index such as Γ^X, then we may remove ambiguity by writing its use as u_Γ^X should the need arise. In all cases, we may add a subscripted stage number (usually s or t) to indicate that the computation should only be allowed to run for at most that many steps. For instance, $\varphi_{e,s}(x)$ refers to the computation $\varphi_e(x)$ run for only s steps.

When partial functions and functionals converge on an input x and yield an output y, we write it as $\varphi_e(x)\downarrow = y$ and $\Phi_e^X(x)\downarrow = y$, respectively. In the case that they do not converge, we write $\varphi_e(x)\uparrow$ and $\Phi_e^X(x)\uparrow$ instead. In some cases, we may wish to enforce that a function does not converge to a particular value y, but nevertheless converges, which we write as $\varphi_e(x)\downarrow \neq y$. On the other hand, if we simply desire that it does not converge to a particular value and include the case where it diverges, then we write $\varphi_e(x) \neq y$. Similar notation holds for functionals as well. In the case where we restrict a computation to run for only s steps, $\varphi_{e,s}(x)\uparrow$ indicates that the function has not converged by stage s.

In some cases, we have several portions of a computation expression that depend on some other value (typically a stage number). For instance, we may have a set A we are building by stages, so that at every stage s we have an approximation A_s to A, which we then use in some computation like $\Phi_{e,s}^{A_s}(x)$. For brevity, it is customary to write such an expression as $\Phi_e^A(x)[s]$, where we interpret the $[s]$ as a directive to restrict everything to its stage s approximation where sensible.

When we have a set A, we will write $A(x)$ as shorthand for $\chi_A(x)$, where χ_A is the characteristic function corresponding to the set A. In general, we may pass back and forth between viewing sets as usual, as infinite bitstrings, or as characteristic functions without much comment. If we wish to take the first n bits of A as a finite bitstring, we write $A \upharpoonright n$, and $A \upharpoonright\!\upharpoonright n$ if we wish to include $A(n)$ in addition (i.e. the first $n+1$ bits of A; the bits of A up to and including n). However, finite bitstrings will generally be called σ, τ and so on if they are defined independently of some set so that they are not mistaken for sets themselves.

We write $A \oplus B$ to mean the *join* of the sets A and B, defined as $\{2n \mid n \in A\} \cup \{2n+1 \mid n \in B\}$. Since the bits of A are placed at bits 0, 2, 4, etc., we say that A forms the *even half* of $A \oplus B$ and B forms the *odd half.* When discussing groups, we will use $\oplus$ for the direct sum, as the context makes it clear that we are working group-theoretically and so it will not be confused for the join.

A set that is the range of a computable function is said to be *computably enumerable*, which we routinely abbreviate as c.e. We write W_e to mean the set $\{x \in \omega \mid \varphi_e(x)\downarrow\}$; by a basic result such sets are exactly the c.e. sets, and so we often use this notation to refer to them. Given an oracle A, a set that is c.e. relative to A and also can compute A is said to be *c.e. in and above A*. This terminology can be confusing at first glance, but it is more sensible when the "and above" is parsed as a parenthetical; the set is c.e. in A and also above A. It is common to abbreviate being c.e. in and above as being c.e.a.

There are several generalizations of sets being c.e., and we shall expand upon those more later as the need arises. One such generalization that is particularly important is a set being *ω-c.e.* A set A is such if there is some computable approximation function $f : \omega \times \omega \to \{0, 1\}$ and a computable function $g : \omega \to \omega$ such that

- $\lim_{s\to\infty} f(x, s) = A(x)$ for all x, and
- $|\{s \mid f(x, s) \neq f(x, s+1)\}| \leq g(x)$ for all x.

Here, rather than an element of A being enumerated and never after allowed to leave, as it is for a c.e. set, we are permitted to change our mind about membership in A finitely often, but we have a bound g that limits how often this may occur. In this vein, we sometimes say that the approximation for A is allowed to change at most computably often to highlight that not just any finite bound will do.

We use $\langle a, b\rangle$ to refer to the *standard pairing function* that provides a one-to-one correspondence between $\omega \times \omega$ and ω. We use the same notation for functions obtained by iterating this operation, which allows us to represent any n-tuple as a value in ω and vice versa. A common use for this function is to partition ω into subsets, one for each index $i \in \omega$. We call such subsets *slices* and write the ith slice as $\omega^{[i]}$, defined as $\omega^{[i]} \coloneqq \{\langle x, i\rangle \mid x \in \omega\}$.

As is usual, we write $A \leq_T B$ to mean that A is computable from B, i.e. there is a functional Γ such that $\Gamma^B = A$. In this case, we say that A is *Turing reducible* to B. We say that A and B are *Turing equivalent*, written $A \equiv_T B$, if $A \leq_T B$ and $B \leq_T A$. The equivalence classes of $\equiv_T$ are called *Turing degrees* and the inherited partial order on degrees is simply written as $\leq$. We write $\mathbf{a} = \deg(A)$ to refer to the degree of the set A. In order to work with ordinals, we implicitly use the method of coding as in the book Ash and Knight [4], which gives rise to the *computable ordinals*. In general, infinite computable ordinals have non-unique representations, but we may fix a unique choice of representations in a given context by specifying a largest computable ordinal of interest.

The most important operator on sets in computability theory is the *jump operator*, which relativizes the Halting Problem to any set A, producing a set A' that is of strictly higher degree. In general, we write $A^{(n)}$ to refer to the nth iterate of the jump operator. More generally, for any computable ordinal α, we can define the αth iterate of the jump of A, written $A^{(\alpha)}$, by induction on (the codes for) the computable ordinals that are at most α. We define $A^{(1)} \coloneqq A'$ and $A^{(\beta+1)} \coloneqq (A^{(\beta)})'$ and for a limit ordinal γ, we have a sequence $\beta_1, \beta_2, \ldots < \gamma$ whose limit is γ (a *fundamental sequence* for γ) and let $A^{(\gamma)} \coloneqq \bigoplus_i A^{(\beta_i)}$. As explained in Ash and Knight [4], different fundamental sequences for γ give rise to different sets, but they are all Turing equivalent. If we have chosen a unique code for each ordinal as mentioned above, then the fundamental sequence is determined by the code and thus the jump is well-defined.

In both Chapter 2 and Chapter 3, we study mathematical structures under the lens of computability theory. We primarily work with computable copies of structures and computable isomorphisms between them. Interestingly, however, it is easy to find computable structures where the isomorphism between them is not computable and, in fact, is highly non-computable. The standard example is the linear order $(\omega, <)$; we can exhibit computable copies of this structure so that isomorphisms between these copies codes $\varnothing'$.

A closer look at this structure shows that $\varnothing'$ is able to compute the isomorphism between any two computable copies, and so the copies mentioned above that actually realize this are somewhat special, and $\mathbf{0}'$, the degree of $\varnothing'$, is *the* degree needed for general isomorphisms between computable copies of $(\omega, <)$: no lesser degree will always work, and any higher degree is more than sufficient. Such degrees are known as *degrees of categoricity* and were introduced by Fokina, Kalimullin and Miller [16]:

Definition 1.1 (Folklore)**.**

Let $\mathcal{A}$ be a computable structure, and suppose that $\mathbf{d}$ is a Turing degree which can compute an isomorphism between any two computable copies of $\mathcal{A}$. Then we say that $\mathcal{A}$ is $\mathbf{d}$-*computably categorical*.

Definition 1.2 (Fokina, Kalimullin and Miller [16])**.**

If $\{\mathbf{c} \mid \mathcal{A} \text{ is } \mathbf{c}\text{-computably categorical }\} = \{\mathbf{c} \mid \mathbf{c} \geq \mathbf{d}\}$, then we say that $\mathbf{d}$ is *the degree of categoricity* of $\mathcal{A}$.

If $\mathcal{A}$ has degree of categoricity $\mathbf{d}$ and there exist computable copies $\mathcal{A}_1$ and $\mathcal{A}_2$ of $\mathcal{A}$ such that every isomorphism $f : \mathcal{A}_1 \cong \mathcal{A}_2$ computes $\mathbf{d}$, we say $\mathcal{A}$ has *strong* degree of categoricity $\mathbf{d}$.

Finally, we say $\mathbf{d}$ is a (strong) degree of categoricity if there exists a computable structure with (strong) degree of categoricity $\mathbf{d}$.

In Chapter 2, in joint work with Barbara Csima, Matthew Harrison-Trainor and Mohammad Assem Mahmoud, (which appears in [9]) we investigate a particular class of degrees and find structures that witness these degrees as strong degrees of categoricity. These degrees are those c.e. in and above $\mathbf{0}^{(\alpha)}$, where α is a computable limit ordinal. Although the result is known for successor ordinals due to Csima, Franklin, and Shore [12], the method that they used relied heavily on existence of the predecessor ordinal. The fact that there is no predecessor ordinal means that more work needs to be done as we see better and better approximations to α from below.

A modification to this result gives some amount of progress to a question of Bazhenov and Marchuk [6] about degrees of categoricity of computable prime models. Prime models of a structure are those that are particularly "simple"; they can be elementarily embedded into any other model of their theory. Bazhenov and Marchuk [6] prove that the degree of categoricity of such models can be at most $\mathbf{0}^{(\omega+1)}$ but the highest degree they were able to realize was $\mathbf{0}^{(\omega)}$. By modifying the result above using what was gained about working with limit ordinals, we are able to construct, for any degree $\mathbf{d}$ c.e. in and above $\mathbf{0}^{(\omega)}$, a computable prime model with strong degree of categoricity $\mathbf{d}$. This cements the lower bound on degrees $\mathbf{d}$ where computable prime models are always $\mathbf{d}$-computably categorical to be precisely $\mathbf{0}^{(\omega+1)}$.

In Chapter 3, in joint work with Barbara Csima and Jonny Stephenson, we take a different tack entirely, where we use degrees of categoricity for motivation. Many of the proofs of constructions of degrees of categoricity use a particular trick for computing degrees of categoricity that they have constructed. The structures are built so that they are rigid and the isomorphism f between them carries a computable relation U on one copy to the set $f(U)$, which is shown to have the desired degree. Thus since f can compute $f(U)$, and the degree of categoricity must compute f, this degree must be at least $f(U)$. Paired with an argument that this degree will suffice to compute the isomorphism between any computable copies of the structure, it provides a convenient method to design a construction to demonstrate a degree is the degree of categoricity. But is this always possible? Could we produce two computable copies of the structure so pathological such that the isomorphism is of high degree – perhaps even a degree of categoricity for the structure – but that same degree cannot be found as the degree of an image of any computable set?

In fact, unary relations are the only relations of interest, as binary relations and higher can encode enough structure within themselves to completely recover the isomorphism. We prove that there are computable copies of $(\omega, <)$ so that the isomorphism between them has degree $\mathbf{0}'$ and so that no computable U exists with $f(U) \equiv_T f$ or $f^{-1}(U) \equiv_T f$. As noted above, $\mathbf{0}'$ is the degree of categoricity for this structure, so even though f has the highest possible degree, it is still not enough to force that some image or preimage of a computable set is enough to compute f.

By modifying the proof of this result to forgo the requirement that $f^{-1}(U)$ cannot compute f, it is possible to produce a computable copy of $(\omega, <)$ so that the isomorphism from the standard copy of $(\omega, <)$ into this constructed copy has degree $\mathbf{0}'$ but no computable U exists such that $f(U) \equiv_T f$. This is not a particularly surprising revelation, but it is somewhat unexpected that we are able to prove that this is impossible if we look for an isomorphism *into* the standard copy from the constructed copy rather than *out of* the standard copy.

That isomorphisms into the standard copy are always coded by a computable U is, however, something peculiar to $(\omega, <)$, which has relatively limited freedom in how it can be built. Indeed, we are able to prove in the case of $(\omega^2, <)$ that there is a computable copy whose isomorphism into the standard copy does not have the same degree as the image of any computable sets through it, which we then attempt to improve to code $\mathbf{0}'''$, the degree of categoricity for this structure. More investigation is needed, though, since we were only able to code $\mathbf{0}''$, so perhaps a more powerful (or clever) technique is needed. Nonetheless, this line of inquiry shows that the method of artfully choosing the set U so that $f(U)$ helps determine the degree of the isomorphism is not always guaranteed to work, provided the copy of the structure is particularly mutilated.

In Chapter 4, we investigate the problem of effectivization of a result on abelian groups proved independently by Cohn [8] and Walker [29], now known as Walker's Cancellation Theorem, which states the following:

Theorem 1.1 (Cohn [8] and Walker [29])**.**
Let A be a finitely generated abelian group. Let G and H be abelian groups such that $A \oplus G \cong A \oplus H$. Then $G \cong H$.

In order to simplify this theorem somewhat, Cohn represented both $A \oplus G$ and $A \oplus H$ inside a single isomorphic copy, E. That is, by identifying A, G and H with their images inside E, we can think of E as being equal to $A \oplus G$ and $A \oplus H$ at the same time. However, the images of the two copies of A need not be the same, so we instead have E containing finitely generated abelian groups A and B which are isomorphic. Thus we can write this as $E = A \oplus G = B \oplus H$ with $A \cong B$, and desire an isomorphism between G and H.

By examining the proof of Cohn, we find that the theorem can be effectivized in all cases. In other words, given a computable group E with computable relations for A, B, G and H and a computable isomorphism between A and B, we can produce a computable isomorphism between G and H. Of course, this is not the end of the story. In order to do so, it seems in Cohn's proof that one must have some finite amount of non-computable information. Thus, the interest in this question from a computability theoretic standpoint is about effective uniformity. That is, can one find a *single* computable procedure that, given indices for a computable group E and the relations for A, B, G and H and a computable isomorphism between A and B, produces an isomorphism between G and H? The key difference is that now the single procedure cannot use any information specific to a given instance of the problem, since it must work for all instances. Unsurprisingly given the non-computable information, the answer is no.

A closer examination of the proof reveals exactly what portion of it fails to be uniform: determining generators for cyclic abelian groups. The surprise is that even offering the uniform procedure a large amount of information about the constituent groups and isomorphisms is not enough to rectify this issue.

Finally, in Chapter 5 we turn our attention away from applications of computability theory and begin looking at a more pure topic within computability theory itself. We explore a modified form of Turing reducibility, called *bounded Turing reducibility.* In many natural contexts, where structures in other areas of mathematics are related via computability theoretic methods, reductions tend to have a particularly nice property where the use of the reduction can be specified ahead of time via a computable function, so that when A is reducible to B via the functional Γ, we have a computable function f such that $\Gamma^{B \upharpoonright f(x)}(x) = A(x)$ for all x. Some authors refer to this type of reduction as a *weak truth table reduction* and write $\leq_{wtt}$. We shall explain in more detail where this notation comes from and how the two definitions are equivalent, but we will only use the notation $\leq_{bT}$ as we consider it to be preferable in this context. Such a reduction is more restrictive because of the need to abide by this bound, and a number of questions arise about which classical results – that is, results about unrestricted Turing reducibility – remain true when replaced by this bounded version.

A collection of such results are the theorems about jump inversion, where given a set A we find conditions on when there is a set X whose jump is of the same degree as A, i.e. $A \equiv_T X'$. However, Csima, Downey and Ng [11] showed that some of these theorems do not hold when we replace Turing reductions with bounded Turing reductions. The classical theorems are:

Theorem (Shoenfield Jump Inversion ([27]))**.**
For every A that is c.e. in and above $\varnothing'$, there is some $X \leq_T \varnothing'$ such that $X' \equiv_T A$.

Theorem (Sacks Jump Inversion ([26]))**.**
For every A that is c.e. in and above $\varnothing'$, there is a non-computable c.e. set X such that $X' \equiv_T A$.

However, by replacing the jump operator with a new jump-like operator designed specifically for the bounded Turing degrees, Csima and Anderson [2] showed that the Shoenfield Jump Inversion Theorem now held. The definition of this *bounded jump* for a set A is a set A^b defined as follows:

$$A^b \coloneqq \{x \mid (\exists i \leq x)[\varphi_i(x)\downarrow \wedge \Phi_x^{A \upharpoonright \varphi_i(x)}(x)\downarrow]\}.$$

To avoid confusion with the classical case, the nth iterate of the bounded jump is written as A^{nb} rather than $A^{(n)}$.

Expanding on this work, Anderson, Csima and Lange [3] endeavored to understand the relationship between the bounded jump and the classical jump by exploring low and high sets in each setting and how they interact. A set A is *low* if $A' \leq_T \varnothing'$ and *high* if $A' \geq_T \varnothing''$, and analogously, A is *bounded low* if $A^b \leq_{bT} \varnothing^b$ and *bounded high* if $A^b \geq_{bT} \varnothing^{2b}$. Additionally, we have the notions of *superhigh* and *superlow* where the Turing reducibility is replaced with a much stronger reducibility known as *truth table* reducibility, of which bounded Turing reducibility is a weakened form, as alluded to above. The details of how this is defined are not relevant at the moment, but it seems natural that lowness, bounded lowness and superlowness should be closely related to one another, as should highness, bounded highness and superhighness. However, Anderson, Csima and Lange showed that this was not the case: there are sets that are low and also bounded high, or high and also bounded low.

We answer some of the open problems from that work by first constructing a set that is low, bounded low, but not superlow and then constructing a set that superhigh but not bounded high. Wu and Wu [30] answered these questions independently using slightly different methods, and did additional work on some of the other problems. They also proved an analogue of pseudo-jump inversion, which has been used classically to help prove the Sacks Jump Inversion Theorem, as in [28]. We will remark more on this and other partial results in this direction after we have investigated a useful characterization of bounded Turing reductions.

Such a characterization was initially provided by Anderson and Csima [2], where they related reductions to $\varnothing^{nb}$ to the Ershov hierarchy, which consists of the sets obtained by generalizing the notion of ω-c.e. sets to arbitrary computable ordinals: the α-c.e. sets. We produce a relativized version of this characterization, allowing us to replace $A \leq_{bT} B^{nb}$ by the equivalent condition that A is ω^n-b.c.e. in B, where we modify the definition of α-c.e. in a precise way to bound parts of it in a computable fashion. This is needed so that it is agreeable with the computable bounds used in the bounded Turing reduction and the bounded jump. The definition is somewhat unwieldy, but not without merit: we then use it to prove that the bounded hierarchy is non-trivial at any level, i.e. that B and B^b have a set X such that $B <_{bT} X <_{bT} B^b$, so that bounded jump inversion is non-trivial.

We also prove that a certain result of Epstein, Haas and Kramer [15] is not detrimental to the use of the Ershov hierarchy in Anderson and Csima's [2] work. That result showed that which sets are ω^2-c.e. is ultimately up to how the computable ordinals are coded in the system of notation used to represent them, which would mean that any effort to, for instance, produce a set that is not ω^2-c.e. would be highly dependent on how ordinals are coded and so results that use such sets would not be well-defined. However, we mitigate this by showing that we can limit this ambiguity by enforcing additional information to be coded about the ordinals we use, and such coding is naturally performed with the ordinals of the form ω^n, which are the ones we are interested in using.

Chapter 2

Degrees of Categoricity Above Limit Ordinals

The material in this chapter is joint work with Barbara Csima, Matthew Harrison-Trainor and Mohammad Assem Mahmoud. We worked on this by having regular working sessions with the whole group; everyone contributed to every section and result. My focus was primarily Theorem 2.3, several of the claims in Theorem 2.6, much of Theorem 2.7 and verification of results of Hirschfeldt and White [19] that were needed in the main constructions. The content of this chapter appears in [9].

2.1 Background

Recall Definition 1.2:

Definition 1.2 (Fokina, Kalimullin and Miller [16]).
If $\{\mathbf{c} \mid \mathcal{A} \text{ is } \mathbf{c}\text{-computably categorical }\} = \{\mathbf{c} \mid \mathbf{c} \geq \mathbf{d}\}$, then we say that $\mathbf{d}$ is *the degree of categoricity* of $\mathcal{A}$.

If $\mathcal{A}$ has degree of categoricity $\mathbf{d}$ and there exist computable copies $\mathcal{A}_1$ and $\mathcal{A}_2$ of $\mathcal{A}$ such that every isomorphism $f : \mathcal{A}_1 \cong \mathcal{A}_2$ computes $\mathbf{d}$, we say $\mathcal{A}$ has *strong* degree of categoricity $\mathbf{d}$.

Finally, we say $\mathbf{d}$ is a (strong) degree of categoricity if there exists a computable structure with (strong) degree of categoricity $\mathbf{d}$.

Fokina, Kalimullin, and Miller showed that every degree that can be realized as a difference of computably enumerable (d.c.e.) sets in and above $\mathbf{0}^{(n)}$, for any $n < \omega$, and also the degree $\mathbf{0}^{(\omega)}$, are degrees of categoricity. Later, Csima, Franklin, and Shore [12] showed that every degree $\mathbf{0}^{(\alpha)}$ for any computable ordinal α, and every degree d.c.e. in and above $\mathbf{0}^{(\alpha)}$ for any successor ordinal α, is a degree of categoricity. Csima and Ng have announced a proof that every Δ_2^0 degree is a degree of categoricity. Recently, Bazhenov, Kalimullin, and Yamaleev [5] have shown that there is a c.e. degree $\mathbf{d}$ and a structure $\mathcal{A}$ with degree of categoricity $\mathbf{d}$, but $\mathbf{d}$ is not a strong degree of categoricity for $\mathcal{A}$. Csima and Stephenson [14] have shown that there is a structure of finite computable dimension that has a degree of categoricity but no strong degree of categoricity.

It is often the case when trying to prove some property $P(\alpha)$ for ordinals α that things get tricky at limit ordinals. Roughly speaking, if α is a successor ordinal and we know something must happen before α, we can safely say it has happened by $\alpha - 1$. For α a limit ordinal, in such a situation there is no canonical choice of earlier ordinal to look at. This is why the methods of [12] did not work above limit ordinals.

Our main result is:

Theorem 2.7.
Let α be a computable limit ordinal and $\mathbf{d}$ a degree c.e. in and above $\mathbf{0}^{(\alpha)}$. There is a computable structure with (strong) degree of categoricity $\mathbf{d}$.

This fills in a gap that was missing from [12] above limit ordinals, making further progress towards Question 5.1 of that paper.

Recall that the *theory* of a structure is the set of first order sentences true in the structure, and that models of the same theory need not be isomorphic. The *type* of a tuple in a structure is the set of formulas (with the appropriate number of free variables) that the tuple satisfies in the structure. The types of a theory are the types that are realized by models of the theory. A type is called *principal* if there is one formula from which the rest follow. A model of a theory is *prime* if it elementarily embeds into all other models of the theory, and when everything is countable, this is the same as saying that the model only realizes principal types. In a sense, prime structures are the most basic or natural structures. Our second result gives progress towards a question of Bazhenov and Marchuk.

Question 2.1 (Bazhenov and Marchuk [6]).
What can be the degrees of categoricity of computable prime models?

A computable prime model – in fact, as [6] shows, a computable homogeneous model – is always $\mathbf{0}^{(\omega+1)}$-categorical, as we can ask $\mathbf{0}^{(\omega+1)}$ if two tuples satisfy the same type.

Bazhenov and Marchuk construct a computable homogeneous model with degree of categoricity $\mathbf{0}^{(\omega+1)}$. The complexity here is in the structure itself, rather than in the theory. To build a prime model with degree of categoricity $\mathbf{0}^{(\omega+1)}$, the complexity must be in the theory: If $\mathcal{A}$ is a computable prime model of a theory T, then $\mathcal{A}$ is $T' \oplus \mathbf{0}^{(\omega)}$-categorical as T' can decide whether a formula is complete, and $\mathbf{0}^{(\omega)}$ can decide whether a formula holds of a tuple in $\mathcal{A}$. We build a computable prime model with degree of categoricity $\mathbf{0}^{(\omega+1)}$ (or any other degree c.e. in and above $\mathbf{0}^{(\omega)}$), showing that the bound cannot be lowered.

Theorem 2.9.
Let $\mathbf{d}$ be a degree c.e. in and above $\mathbf{0}^{(\omega)}$. There is a computable prime model $\mathcal{A}$ with strong degree of categoricity $\mathbf{d}$.

Bazhenov and Marchuk stated in [6] that a careful examination of the structures constructed in [16] shows they are prime models, so that all degrees d.c.e. in and above $\mathbf{0}^{(n)}$ for a finite n, as well as $\mathbf{0}^{(\omega)}$, are strong degrees of categoricity of prime models. Along the way to proving Theorem 2.9 we verify in Theorem 2.4 that the building blocks used by Csima, Franklin and Shore for their examples in [12] are prime. This is enough to see that their structures realizing degrees of categoricity less than or equal to $\mathbf{0}^{(\omega)}$ are prime. However, the structure in [12] with degree of categoricity $\mathbf{0}^{(\omega+1)}$ is not prime. With Theorem 2.9, we see that all known degrees of categoricity less than or equal to the $\mathbf{0}^{(\omega+1)}$ bound can be realized by a prime model.

2.2 Categoricity Relative to Decidable Models

As a warm-up to illustrate the methods used to prove these two theorems, we give a simple proof of a result of Goncharov [17] that for every c.e. degree $\mathbf{d}$, there is a decidable prime model with degree of categoricity $\mathbf{d}$ with respect to decidable copies. Recall that a structure is said to be decidable if its full elementary diagram is computable. In [17], Goncharov made the following definitions:

Definition 2.1 (Goncharov [17])**.**
Let $\mathbf{d}$ be a Turing degree and $\mathcal{A}$ a decidable structure. Then $\mathcal{A}$ is $\mathbf{d}$-*categorical with respect to decidable copies* if for every decidable copy $\mathcal{B}$ of $\mathcal{A}$, $\mathbf{d}$ computes an isomorphism between $\mathcal{A}$ and $\mathcal{B}$.

Definition 2.2 (Goncharov [17])**.**
Let $\mathbf{d}$ be a Turing degree and $\mathcal{A}$ a decidable structure. Then $\mathbf{d}$ is the *degree of categoricity of $\mathcal{A}$ with respect to decidable copies* if:

- $\mathcal{A}$ is $\mathbf{d}$-categorical with respect to decidable copies, and
- whenever $\mathcal{A}$ is $\mathbf{c}$-categorical with respect to decidable copies, $\mathbf{c} \geq \mathbf{d}$.

It is not hard to see that between any two decidable copies of a prime model, there is a $\mathbf{0}'$-computable isomorphism. Goncharov showed that any c.e. degree can be the degree of categoricity with respect to decidable copies of a prime model. We give a different proof, which we think is simpler, and which demonstrates some of the techniques that we will use later.

Theorem 2.2 (Goncharov [17, Theorem 3])**.**
Let $\mathbf{d}$ be a c.e. degree. Then there is a decidable prime model $\mathcal{M}$ which has strong degree of categoricity $\mathbf{d}$ with respect to decidable models.

Proof:
Let $D \in \mathbf{d}$ be a c.e. set. We will construct the structures $\mathcal{M}$ and $\mathcal{N}$. They are the disjoint union of infinitely many structures $\mathcal{M}_n$ and $\mathcal{N}_n$, with $\mathcal{M}_n$ and $\mathcal{N}_n$ picked out by unary relations R_n. The nth sort will code whether $n \in D$. Fix n. The structure $\mathcal{M}_n$ will have infinitely many elements $(a_i)_{i\in\omega}$. There will be infinitely many unary relations $(U_\ell)_{\ell\in\omega}$ defined on $\mathcal{M}_n$ so that:

$$a_0 \in U_s \iff n \in D_{\text{at } s}$$

where $n \in D_{\text{at } s}$ means that n enters D at exactly stage s, and

$$a_i \notin U_s \text{ for } i > 0 \text{ and all } s.$$

Similarly, $\mathcal{N}_n$ will have infinitely many elements $(b_i)_{i\in\omega}$ with the unary relations defined so that:

$$b_i \in U_s \iff i = s \text{ and } n \in D_{\text{at } s}.$$

It is easy to see that we can build computable copies of $\mathcal{M}$ and $\mathcal{N}$. These copies are in fact decidable.

<u>Claim 2.2.1:</u> $\mathcal{M}$ and $\mathcal{N}$ are decidable.

<u>Proof of Claim:</u> Given a formula $\varphi(x_1,\ldots,x_n)$ with k quantifiers and $a_{i_1},\ldots,a_{i_n} \in \mathcal{M}$, it is not hard to see that $\mathcal{M} \models \varphi(a_{i_1},\ldots,a_{i_n})$ if and only if the finite substructure $\mathcal{M}'$ of $\mathcal{M}$ whose domain consists of $a_0,\ldots,a_{k+n}$ and $a_{i_1},\ldots,a_{i_n}$ also has $\mathcal{M}' \models \varphi(a_{i_1},\ldots,a_{i_n})$. Thus $\mathcal{M}$ is decidable.

For $\mathcal{N}$, suppose we have a formula $\varphi(x_1,\ldots,x_n)$ with at most k quantifiers and which uses only some subset of the relations $U_0,\ldots,U_k$. Let $b_{i_1},\ldots,b_{i_n}$ be elements of $\mathcal{N}$.

Then $\mathcal{N} \models \varphi(b_{i_1}, \ldots, b_{i_n})$ if and only if the finite substructure $\mathcal{N}'$ of $\mathcal{M}$ whose domain consists of $b_0, \ldots, b_{k+n}$ and $b_{i_1}, \ldots, b_{i_n}$ also has $\mathcal{N}' \models \varphi(b_{i_1}, \ldots, b_{i_n})$. Thus $\mathcal{N}$ is decidable. ■

Claim 2.2.2: $\mathcal{M}$ and $\mathcal{N}$ are isomorphic.

Proof of Claim: It suffices to show that for each n, $\mathcal{M}_n$ and $\mathcal{N}_n$ are isomorphic. If $n \notin D$, then $a_i \mapsto b_i$ induces an isomorphism between $\mathcal{M}_n$ and $\mathcal{N}_n$. If $n \in D_{\text{at } s}$, then the map

$$\begin{aligned} a_0 &\mapsto b_s && \\ a_i &\mapsto b_{i-1} && \text{when } 0 < i \leq s \\ a_i &\mapsto b_i && \text{when } i > s \end{aligned}$$

is an isomorphism between $\mathcal{M}_n$ and $\mathcal{N}_n$. ■

Claim 2.2.3: $\mathcal{M}$ and $\mathcal{N}$ are prime.

Proof of Claim: It suffices to show that each $\mathcal{M}_n$ and $\mathcal{N}_n$ are prime, since these structures are determined inside $\mathcal{M}$ and $\mathcal{N}$ uniquely by the relation R_n. It is not hard to see that $\mathcal{M}_n$ and $\mathcal{N}_n$ are models of an $\aleph_0$-categorical theory, and hence are prime. ■

Claim 2.2.4: Any isomorphism between $\mathcal{M}$ and $\mathcal{N}$ can compute D.

Proof of Claim: Let g be an isomorphism between $\mathcal{M}$ and $\mathcal{N}$. For each n, let $(a_i)_{i \in \omega}$ and $(b_i)_{i \in \omega}$ be the elements in the definition of $\mathcal{M}_n$ and $\mathcal{N}_n$. Let s be such that $g(a_0) = b_s$. Then $n \in D$ if and only if $n \in D_s$. ■

Claim 2.2.5: Given a computable copy $\widetilde{\mathcal{M}}$ of $\mathcal{M}$, D can compute an isomorphism between $\mathcal{M}$ and $\widetilde{\mathcal{M}}$.

Proof of Claim: For each n, let $\widetilde{\mathcal{M}}_n$ be the structure with domain R_n in $\widetilde{\mathcal{M}}$. It suffices to compute an isomorphism g between $\mathcal{M}_n$ and $\widetilde{\mathcal{M}}_n$ for each n. Let $(c_i)_{i \in \omega}$ be the elements of $\widetilde{\mathcal{M}}_n$. If $n \notin D$, no relation U_j holds of any of the the elements $(a_i)_{i \in \omega}$ or $(c_i)_{i \in \omega}$. So $a_i \mapsto c_i$ is an isomorphism. On the other hand, if $n \in D$, then for some unique s, $a_0 \in U_s$. We can look for c_k such that $c_k \in U_s$. Map a_0 to c_k; map each other a_i to some other c_i. ■

These claims complete the proof of the theorem. □

2.3 Back-and-forth Trees

Fix a path through the computable ordinals, as in Ash and Knight [4]. We will identify computable ordinals with their notation on this path. We will always first fix a limit ordinal α and work below it. Recall that one can decide effectively whether a given computable ordinal is a limit ordinal or a successor ordinal. For each limit ordinal $\beta < \alpha$, fix a fundamental sequence for β, that is, an increasing sequence of successor ordinals whose limit is β.

Hirschfeldt and White [19] defined, for each successor ordinal β, a pair of trees $\mathcal{A}_\beta$ and $\mathcal{E}_\beta$ which can be differentiated exactly by β jumps. These trees are called *back-and-forth trees.*

Definition 2.3 (Hirschfeldt and White [19, Definition 3.1])**.**
Back-and-forth trees are defined recursively in β. We view these as structures in the language of graphs with the root node distinguished.

We take $\mathcal{A}_1$ to be the tree with just a root node and no children, and we take $\mathcal{E}_1$ to be the tree where the root node has infinitely many children, none of which have children. See Fig. 2.1. We say that these trees have *back-and-forth rank* 1.

Suppose β is a successor ordinal. Define $\mathcal{A}_{\beta+1}$ as a root node with infinitely many children, each the root of a copy of $\mathcal{E}_\beta$, and define $\mathcal{E}_{\beta+1}$ as a root node with infinitely many children, each the root of a copy of $\mathcal{A}_\beta$, and also infinitely many other children, each the root of a copy of $\mathcal{E}_\beta$. See Fig. 2.2. These trees have back-and-forth rank $\beta + 1$.

Now suppose β is a non-zero limit ordinal, and let $\beta_0, \beta_1, \ldots$ be a fundamental sequence of successor ordinals for β, that is, a sequence of successor ordinals below β with limit β. We first define a family of helper trees $\mathcal{L}_{\beta,k}$ where $k \in \omega \cup \{\infty\}$. Define $\mathcal{L}_{\beta,\infty}$ to consist of a root node whose children are root nodes of copies of $\mathcal{A}_{\beta_i}$, and such that each copy appears exactly once as a child. For $k \in \omega$, $\mathcal{L}_{\beta,k}$ has a root node whose children are root nodes of copies of $\mathcal{A}_{\beta_0}, \ldots, \mathcal{A}_{\beta_k}, \mathcal{E}_{\beta_{k+1}}, \mathcal{E}_{\beta_{k+2}}, \ldots$ where again each copy appears exactly once as a child. Such trees are shown in Fig. 2.3. We say these trees have back-and-forth rank β.

We can now define $\mathcal{A}_{\beta+1}$ and $\mathcal{E}_{\beta+1}$ for the non-zero limit ordinal β. For $\mathcal{A}_{\beta+1}$, we have a root node with infinitely many children, each the root node of a copy of $\mathcal{L}_{\beta,k}$ such that for each $k \in \omega$, $\mathcal{L}_{\beta,k}$ appears infinitely many times. The definition of $\mathcal{E}_{\beta+1}$ is similar, except k is drawn from $\omega \cup \{\infty\}$. See Fig. 2.4. These trees have back-and-forth rank $\beta + 1$.

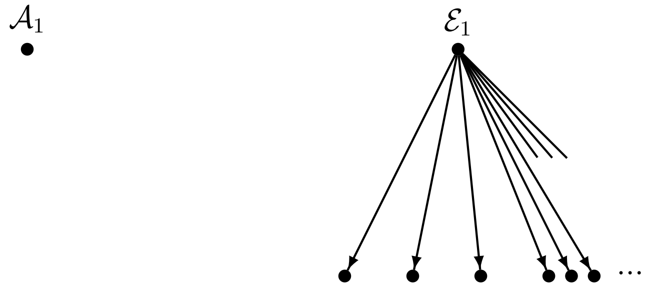

Figure 2.1: $\mathcal{A}_1$ and $\mathcal{E}_1$

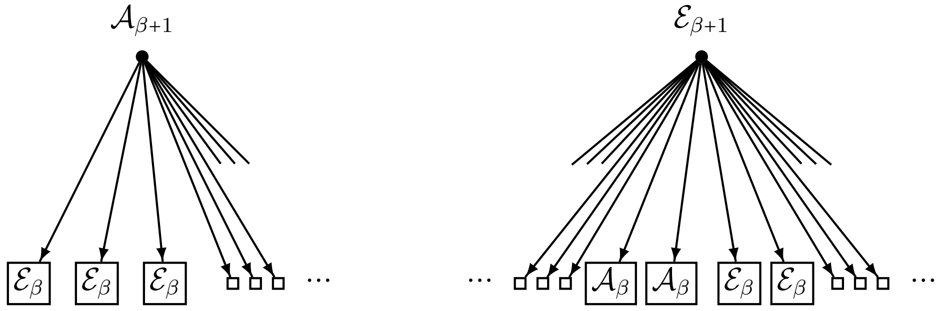

Figure 2.2: $\mathcal{A}_{\beta+1}$ and $\mathcal{E}_{\beta+1}$ when β is a successor ordinal.

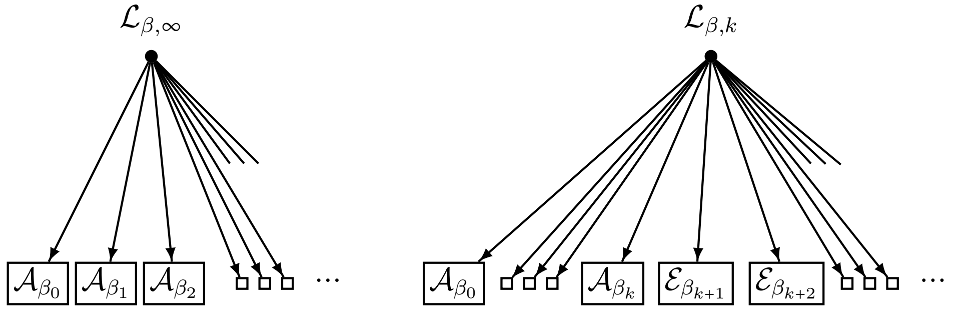

Figure 2.3: Helper trees $\mathcal{L}_{\beta,\infty}$ and $\mathcal{L}_{\beta,k}$ for $k \in \omega$ for the non-zero limit ordinal β.

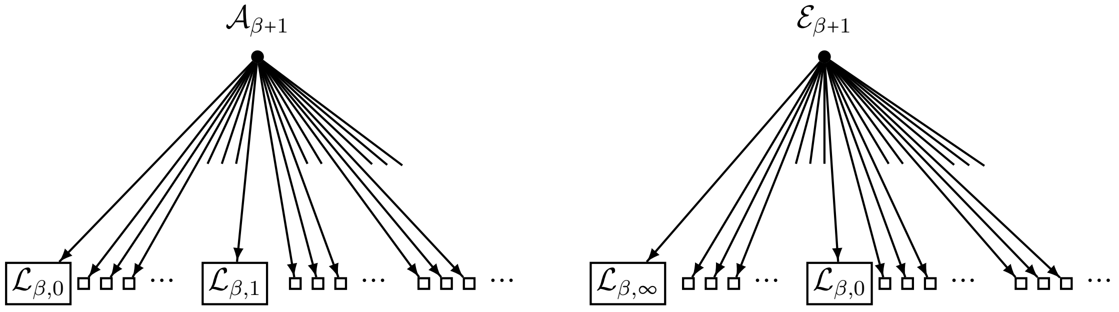

Figure 2.4: $\mathcal{A}_{\beta+1}$ and $\mathcal{E}_{\beta+1}$ for the non-zero limit ordinal β.

The next two lemmas piece together the facts that we will need about the back-and-forth trees, first for arbitrary β, and second some additional properties for finite β in particular. These facts come from Hirschfeldt and White [19] and Csima, Franklin and Shore [12].

Lemma 2.3.
Let α be a computable ordinal. For a successor ordinal $\beta < \alpha$, the structures $\mathcal{A}_\beta$ and $\mathcal{E}_\beta$ satisfy the following properties:

1. Uniformly in β and an index for a Σ^0_β set S, there is a computable sequence of structures $\mathcal{C}_x$ such that

$$x \in S \iff \mathcal{C}_x \cong \mathcal{E}_\beta \quad \text{and} \quad x \notin S \iff \mathcal{C}_x \cong \mathcal{A}_\beta.$$

2. Uniformly in β, there is a Σ^0_β sentence φ such that $\mathcal{E}_\beta \models \varphi$ and $\mathcal{A}_\beta \not\models \varphi$.

3. $\mathcal{A}_\beta$ and $\mathcal{E}_\beta$ are uniformly $\mathbf{0}^{(\beta)}$-categorical.

Proof:
For (1), take $(\mathcal{C}_x)_x$ to be the computable sequence of trees given by Proposition 3.2 in Hirschfeldt and White [19].

For (2), take φ to be the sentence given by evaluating the formula guaranteed by Lemma 3.5 in Hirschfeldt and White [19] for $\mathcal{B} = \mathcal{E}_\beta$ at its own root node. The complexity of φ is the natural complexity of $\mathcal{E}_\beta$, which is Σ_β. This lemma says that for any tree $\mathcal{T}$, $\mathcal{T} \models \varphi$ if and only if $\mathcal{T} \cong \mathcal{E}_\beta$.

Finally, for (3), we use a result from Csima, Franklin, and Shore [12] about back-and-forth trees. We will consider $\mathcal{A}_\beta$; the case for $\mathcal{E}_\beta$ is identical. We have that $\mathcal{A}_\beta$ is a back-and-forth tree, and hence if $\mathcal{C}$ is a computable structure isomorphic to $\mathcal{A}_\beta$, then it is also a computable back-and-forth tree. Corollary 2.6 in [12] allows $\varnothing^{(\gamma)}$ to uniformly compute an isomorphism between these two trees when the back-and-forth rank of the trees is at most γ. Since the rank of $\mathcal{A}_\beta$ is β by construction, the isomorphism is uniformly computable in $\mathbf{0}^{(\beta)}$. □

Now for finite ordinals β (and writing n for β), we have some additional properties. We will state the lemma in full, including properties that were covered by the previous lemma. We think that these facts are well-known, but we do not know of a reference in print.

Lemma 2.4.
For $0 < n < \omega$, the structures $\mathcal{A}_n$ and $\mathcal{E}_n$ satisfy the properties:

1. Uniformly in n and an index for a Σ^0_n set S, there is a computable sequence of structures $\mathcal{C}_x$ such that

$$x \in S \Longleftrightarrow \mathcal{C}_x \cong \mathcal{E}_n \quad \text{and} \quad x \notin S \Longleftrightarrow \mathcal{C}_x \cong \mathcal{A}_n.$$

2. For each n, there is an elementary first-order $\exists_n$ sentence φ_n, computable uniformly in n, such that $\mathcal{E}_n \vDash \varphi$ and $\mathcal{A}_n \nvDash \varphi$.

3. $\mathcal{A}_n$ and $\mathcal{E}_n$ are prime.

4. $\mathcal{A}_n$ and $\mathcal{E}_n$ are $\mathbf{0}^{(n)}$-categorical uniformly in n.

Proof:
(1) and (4) are the same as in the previous lemma. We show using induction on n that these sequences satisfy (2) and (3) as well. It is easy to see that $\mathcal{A}_1$ and $\mathcal{E}_1$ are prime models of their theories and that they are distinguishable (in the sense of (2) in the statement of the lemma) by the existential sentence $\varphi_1 := \exists x \exists y(x \neq y)$. Assume now that $\mathcal{A}_n$ and $\mathcal{E}_n$ are prime and distinguishable by a first-order $\exists_n$ sentence φ_n (in the sense that $\mathcal{E}_n \vDash \varphi_n$ but $\mathcal{A}_n \nvDash \varphi_n$). We show that $\mathcal{A}_{n+1}$ and $\mathcal{E}_{n+1}$ are prime and distinguishable by a first-order $\exists_{n+1}$ sentence φ_{n+1}.

It is not hard to see that we can take φ_{n+1} to be the sentence

$$(\exists x)[\neg\varphi_n[\leq x] \wedge (x \text{ is a child of the root node})]$$

where x is a new variable not appearing in φ_n and $\varphi_n[\leq x]$ is the formula obtained from φ_n by bounding every quantifier to the subtree below x. (Note that in a tree of rank n, if z is a descendant of x, i.e. there is a path from x to z, the length of the path is at most n, and so this is first-order definable and does not change the quantifier rank.) $\mathcal{E}_{n+1} \models \varphi_{n+1}$ but $\mathcal{A}_{n+1} \not\models \varphi_{n+1}$.

It remains to show that $\mathcal{A}_{n+1}$ and $\mathcal{E}_{n+1}$ are prime. The same method will work for both structures. Let $\bar{a}$ be an arbitrary tuple in $\mathcal{E}_{n+1}$. We describe a formula that isolates the type of the tuple $\bar{a}$. Let $r_1, \ldots, r_k$ be the children of the root which are the roots of subtrees containing elements of $\bar{a}$; say that $\bar{a} = (\bar{a}_1, \ldots, \bar{a}_k)$ where $\bar{a}_i$ is in the subtree below r_i. (Note that we can re-order the tuples as we like, as if the type of some permutation of $\bar{a}$ is isolated, so is $\bar{a}$.) By the induction hypothesis, we know that the subtree with root r_i is prime for every i. Hence for each $i = 1, \ldots, k$ there is a formula which isolates the type of the tuple $\bar{a}_i$ in the subtree with root r_i. There is also, for each r_i, a formula (either φ_n or $\neg\varphi_n$) which distinguishes between whether the subtree below r_i is isomorphic to $\mathcal{A}_n$ or $\mathcal{E}_n$. So we can isolate the type of $\bar{a}$ by saying that there are children $r_1, \ldots, r_k$ of the root such that $\bar{a}_i$ satisfies the formula, in the subtree below r_i, which isolates it, and by saying whether the subtree below each r_i is isomorphic to $\mathcal{A}_n$ or $\mathcal{E}_n$. □

Fokina, Kalimullin, and Miller [16] showed that there is a structure $\mathcal{A}$ with strong degree of categoricity $\mathbf{0}^{(\omega)}$. We note the well-known fact that one can also have $\mathcal{A}$ be a prime model. Our proof follows that of Csima, Franklin and Shore [12].

Theorem 2.5.
There is a computable structure $\mathcal{A}$ with strong degree of categoricity $\mathbf{0}^{(\omega)}$ such that $\mathcal{A}$ is a prime model of its theory.

Proof sketch:
The structure is just the disjoint union of infinitely many copies of each $\mathcal{E}_n$ for $n < \omega$. Theorem 3.1 of Csima, Franklin and Shore [12] shows that this has strong degree of categoricity $\mathbf{0}^{(\omega)}$, and it is not hard to see using Theorem 2.4 that this structure is prime. □

2.4 C.E. In And Above a Limit Ordinal

We begin this section by a short discussion of how we code a c.e. set into a structure. Consider a c.e. set C. If one knows, for each n, at what point the approximation to $C(n)$ has settled, then one can compute C. Moreover, one does not need to know exactly when C settles, but just a point after which $C(n)$ has settled. In particular, any sufficiently large function can compute C. Moreover, C itself can compute such a function. Following the terminology of Groszek and Slaman [18], we say that C has a self-modulus.

Definition 2.4 (Groszek and Slaman [18])**.**
Let $F{:}\,\omega \to \omega$ and $X \subseteq \omega$. Then:

- F is a *modulus* (of computation) for X if every $G{:}\,\omega \to \omega$ that dominates F pointwise computes X.
- X has a *self-modulus* if X computes a modulus for itself.

The self-modulus of a c.e. set C is the function $f(n) = \mu s(C_s(n) = C(n))$. Groszek and Slaman showed that every Δ^0_2 or α-CEA set has a self-modulus. In fact, the self-modulus of a c.e. set has a nice form: it has a non-decreasing computable approximation.

Definition 2.5.
A function $F{:}\,\omega \to \omega$ is *limitwise monotonic* if there is a computable approximation function $f{:}\,\omega \times \omega \to \omega$ such that, for all n,

- $F(n) = \lim_{s\to\infty} f(n,s)$.
- For all s, $f(n,s) \le f(n,s+1)$.

In fact, it is well-known and an easy exercise to show that the sets of c.e. degree are exactly those with limitwise monotonic self-moduli. These remarks also relativize.

The next lemma encodes a limitwise monotonic function into the isomorphisms of copies of a computable structure. Any isomorphism dominates the limitwise monotonic function; but it does not seem to be the case that dominating the limitwise monotonic function is sufficient to compute isomorphisms.

Lemma 2.6.
Let α be a computable limit ordinal. Let $f{:}\,\omega \to \omega$ be limitwise monotonic relative to $\mathbf{0}^{(\alpha)}$. There is a structure with computable copies $\mathcal{M}$ and $\mathcal{N}$ such that:

1. Every isomorphism between $\mathcal{M}$ and $\mathcal{N}$ computes a function which dominates f.
2. $f \oplus \mathbf{0}^{(\alpha)}$ computes an isomorphism between any two computable copies of $\mathcal{M}$ and $\mathcal{N}$.

Proof:
Let Φ be a computable operator such that $f(n) = \lim_{s\to\infty} \Phi^{\varnothing^{(\alpha)}}(n,s)$ and this is monotonic in s. Write $\varnothing^{(\alpha)} = \bigoplus_{\gamma<\alpha} \varnothing^{(\gamma)}$ for successor ordinals $\gamma < \alpha$. By convention, for $\beta < \alpha$, we say that $\Phi^{\varnothing^{(\beta)}}(n,s)$ converges if the computation $\Phi^{\varnothing^{(\alpha)}}(n,s)$ halts, but the

only part of the oracle $\varnothing^{(\alpha)} = \bigoplus_{\gamma<\alpha} \varnothing^{(\gamma)}$ that is read during the computation is that part with $\gamma \le \beta$. So if $\Phi^{\varnothing^{(\beta)}}(n,s) = m$ then $\Phi^{\varnothing^{(\alpha)}}(n,s) = m$, and because α is a limit ordinal, if $\Phi^{\varnothing^{(\alpha)}}(n,s) = m$ then $\Phi^{\varnothing^{(\beta)}}(n,s) = m$ for some successor ordinal $\beta < \alpha$.

Let $(\mathcal{A}_\beta)_{\beta<\alpha}$ and $(\mathcal{E}_\beta)_{\beta<\alpha}$ be as in Theorem 2.3. We will construct the structures $\mathcal{M}$ and $\mathcal{N}$. They are the disjoint union of infinitely many structures $\mathcal{M}_n$ and $\mathcal{N}_n$, with $\mathcal{M}_n$ and $\mathcal{N}_n$ picked out by unary relations R_n. The nth sort will code the value of $f(n)$.

Fix n. The structure $\mathcal{M}_n$ will have infinitely many elements $(a_i)_{i\in\omega}$ satisfying a unary relation S. Each of these elements will be attached to, for each successor ordinal $\beta < \alpha$, a "box" $\mathcal{M}_{i,\beta}$ which contains within it a copy of either $\mathcal{A}_\beta$ or $\mathcal{E}_\beta$; each of the boxes are disjoint. By this we mean that there are binary relations T_β such that $T_\beta(a_i, x)$ holds for exactly those $x \in \mathcal{M}_{i,\beta}$. We define the structure $\mathcal{M}_{i,\beta}$ in the language of Theorem 2.3 so that:

1. $\mathcal{M}_{0,\beta} \cong \mathcal{A}_\beta$ for all β.
2. $\mathcal{M}_{i,\beta} \cong \mathcal{E}_\beta$, $i \ge 1$, if there is s such that $\Phi^{\varnothing^{(\beta)}}(n,s) \ge i$.
3. $\mathcal{M}_{i,\beta} \cong \mathcal{A}_\beta$, $i \ge 1$, otherwise.

Note that the condition in (2) is Σ^0_β and so we can build such a structure $\mathcal{M}_n$ computably.

Similarly, $\mathcal{N}_n$ will have infinitely many elements $(b_i)_{i\in\omega}$, each of which is attached to, for each $\beta < \alpha$, a box $\mathcal{N}_{i,\beta}$ which contains within it:

1. $\mathcal{N}_{i,\beta} \cong \mathcal{E}_\beta$ if there is s such that $\Phi^{\varnothing^{(\beta)}}(n,s) > i$.
2. $\mathcal{N}_{i,\beta} \cong \mathcal{A}_\beta$ otherwise.

Again, the condition in (1) is Σ^0_β and so we can build such a structure $\mathcal{N}_n$ computably.

<u>Claim 2.6.1:</u> Fix n.

1. For each $j < f(n)$, there is $\beta < \alpha$ such that:
   - for $\gamma < \beta$, $\mathcal{M}_{j+1,\gamma} \cong \mathcal{N}_{j,\gamma} \cong \mathcal{A}_\gamma$,
   - for $\gamma \ge \beta$, $\mathcal{M}_{j+1,\gamma} \cong \mathcal{N}_{j,\gamma} \cong \mathcal{E}_\gamma$,
2. For each $j \ge f(n)$ and $\beta < \alpha$, $\mathcal{M}_{j+1,\beta} \cong \mathcal{N}_{j,\beta} \cong \mathcal{M}_{0,\beta} \cong \mathcal{A}_\beta$.

Proof of Claim: For (1), it is clear from the definitions of $\mathcal{M}_{j+1,\beta}$ and $\mathcal{N}_{j,\beta}$ that for all $\beta < \alpha$, $\mathcal{M}_{j+1,\beta} \cong \mathcal{N}_{j,\beta}$. Since $j < f(n)$, there is s such that $\Phi^{\varnothing^{(\alpha)}}(n,s) = f(n) > j$. In particular, there must be some $\beta < \alpha$ such that there is s with $\Phi^{\varnothing^{(\beta)}}(n,s) > j$. Let β be the least such ordinal. Then for all $\gamma \geq \beta$, there is s such that $\Phi^{\varnothing^{(\gamma)}}(n,s) > j$, and so $\mathcal{M}_{j+1,\gamma} \cong \mathcal{N}_{j,\gamma} \cong \mathcal{E}_\gamma$. By choice of β, for $\gamma < \beta$, there is no s such that $\Phi^{\varnothing^{(\gamma)}}(n,s) > j$, and so $\mathcal{M}_{j+1,\gamma} \cong \mathcal{N}_{j,\gamma} \cong \mathcal{A}_\gamma$.

For (2), it is clear that $\mathcal{M}_{j+1,\beta} \cong \mathcal{N}_{j,\beta}$ for each $j \geq f(n)$ and each $\beta < \alpha$, and it is also clear that $\mathcal{M}_{0,\beta} \cong \mathcal{A}_\beta$ for each $\beta < \alpha$. If $j \geq f(n)$, then since Φ is limitwise monotonic approximation to f, $\Phi^{\varnothing^{(\beta)}}(n,s) \leq f(n) \leq j$ for all s and β. Thus $\mathcal{N}_{j,\beta} \cong \mathcal{A}_\beta$ for all β. ■

Claim 2.6.2: $\mathcal{M}$ and $\mathcal{N}$ are isomorphic.

Proof of Claim: It suffices to show that for each n, $\mathcal{M}_n$ and $\mathcal{N}_n$ are isomorphic. Fix n. Using Claim 2.6.1, we see that the map

$$\begin{aligned} &a_0 \mapsto b_{f(n)} && \\ &a_i \mapsto b_{i-1} && \text{when } 0 < i \leq f(n) \\ &a_i \mapsto b_i && \text{when } i > f(n) \end{aligned}$$

extends to an isomorphism between $\mathcal{M}_n$ and $\mathcal{N}_n$. ■

Claim 2.6.3: Any isomorphism between $\mathcal{M}$ and $\mathcal{N}$ can compute a function which dominates f.

Proof of Claim: Let g be an isomorphism between $\mathcal{M}$ and $\mathcal{N}$. We will compute, using g, a function $\hat{g}$ which dominates f. For each n, define $\hat{g}(n)$ as follows. Let $(a_i)_{i\in\omega}$ and $(b_i)_{i\in\omega}$ be the elements in the definition of $\mathcal{M}_n$ and $\mathcal{N}_n$. Then $\hat{g}(n)$ is the number satisfying $g(a_0) = b_{\hat{g}(n)}$.

To see that $\hat{g}(n) \geq f(n)$, we use Claim 2.6.1. For each $\beta < \alpha$, $\mathcal{M}_{0,\beta} \cong \mathcal{A}_\beta$, but if $j < f(n)$, there is $\beta < \alpha$ such that $\mathcal{N}_{j,\beta} \cong \mathcal{E}_\beta$. Thus no isomorphism can map a_0 to b_j for $j < f(n)$, and so $\hat{g}(n) \geq f(n)$. ■

Claim 2.6.4: Given a computable copy $\widetilde{\mathcal{N}}$ of $\mathcal{N}$, $f \oplus \mathbf{0}^{(\alpha)}$ can compute an isomorphism between $\mathcal{N}$ and $\widetilde{\mathcal{N}}$.

It is more convenient for the proof to consider $\mathcal{N}$ rather than $\mathcal{M}$ in this claim, but as they are isomorphic it does not matter which we choose.

Proof of Claim: For each n, let $\widetilde{\mathcal{N}}_n$ be the structure with domain R_n in $\widetilde{\mathcal{N}}$. It suffices to compute an isomorphism g between $\mathcal{N}_n$ and $\widetilde{\mathcal{N}}_n$ for each n. Inside of $\widetilde{\mathcal{N}}_n$, let $(c_i)_{i\in\omega}$ list the elements x satisfying $S(x)$. For each c_i, let $\widetilde{\mathcal{N}}_{i,\beta}$ be the tree whose domain consists of the elements y satisfying $T_\beta(c_i, y)$. To begin, we will define g on $(b_i)_{i\in\omega} \subseteq \mathcal{N}_n$. Compute $f(n)$. Using $\mathbf{0}^{(\alpha)}$, look for $f(n)$ elements c_i such that, for some $\beta < \alpha$, $\widetilde{\mathcal{N}}_{i,\beta} \cong \mathcal{E}_\beta$. This search is computable relative to $\mathbf{0}^{(\alpha)}$ by Theorem 2.3 (2), and by Claim 2.6.1 we know that there are exactly $f(n)$ such elements and so the search will terminate after finding every such element. Rearranging $(c_i)_{i\in\omega}$, we may assume that these elements are $c_0, \dots, c_{f(n)-1}$.

Now, for each $k < f(n)$, find the least β_k such that $\mathcal{N}_{k,\beta_k} \cong \mathcal{E}_{\beta_k}$, and the least γ_k such that $\widetilde{\mathcal{N}}_{k,\gamma_k} \cong \mathcal{E}_{\gamma_k}$. Again, this is computable in $\mathbf{0}^{(\alpha)}$ by Theorem 2.3 (2). Note that we must ask $\mathbf{0}^{(\alpha)}$ to determine what β_k and γ_k are least. The sets $\{\beta_0, \dots, \beta_{f(n)-1}\}$ and $\{\gamma_0, \dots, \gamma_{f(n)-1}\}$ must be identical including multiplicity (but possibly in a different order) as $\widetilde{\mathcal{N}}_n$ and $\mathcal{N}_n$ are isomorphic. So by rearranging $(c_i)_{i\in\omega}$ once again we may assume that $\beta_k = \gamma_k$ for each $k < f(n)$.

We have now rearranged the list $(c_i)_{i\in\omega}$ so that for each i and $\beta < \alpha$, $\mathcal{N}_{i,\beta} \cong \widetilde{\mathcal{N}}_{i,\beta}$. Define g so that $g(b_i) = c_i$. For each i and $\beta < \alpha$, $\mathcal{N}_{i,\beta} \cong \widetilde{\mathcal{N}}_{i,\beta}$ are isomorphic to either $\mathcal{A}_\beta$ or $\mathcal{E}_\beta$, which are uniformly $\mathbf{0}^{(\beta)}$-categorical (Theorem 2.3 (3)), and we can compute using $\mathbf{0}^{(\alpha)}$ which case we are in. So we can define g on $\mathcal{N}_{i,\beta}$ to be an isomorphism to $\widetilde{\mathcal{N}}_{i,\beta}$. Thus g is an isomorphism from $\mathcal{N}_n$ to $\widetilde{\mathcal{N}}_n$. ■

These claims complete the proof of the theorem. □

Using this lemma, and taking the limitwise monotonic function to be the self-modulus of a c.e. set, it is not hard to prove our main theorem.

Theorem 2.7.
Let α be a computable limit ordinal and $\mathbf{d}$ a degree c.e. in and above $\mathbf{0}^{(\alpha)}$. There is a computable structure with (strong) degree of categoricity $\mathbf{d}$.

Proof:
Fix α and let $D \in \mathbf{d}$ be a set c.e. in and above $\mathbf{0}^{(\alpha)}$. Since D is c.e. in and above $\mathbf{0}^{(\alpha)}$, it has a self-modulus f that is limitwise monotonic relative to $\mathbf{0}^{(\alpha)}$. Consider the structure $\mathcal{M}$ constructed in Theorem 2.6 for this f. We will enrich this structure slightly to produce a new structure $\mathcal{S}$. Let $\mathcal{S}_\alpha$ be the computable structure with strong degree of categoricity $\mathbf{0}^{(\alpha)}$ constructed in Theorem 3.1 of Csima, Franklin and Shore [12]. The new structure $\mathcal{S}$ consists of $\mathcal{M}$ and a disjoint copy of $\mathcal{S}_\alpha$, and a new unary relation R such that $R(x)$ holds exactly when x belongs to the copy of $\mathcal{S}_\alpha$. We claim that $\mathcal{S}$ has strong degree of categoricity $\mathbf{d}$.

First, suppose that $\mathcal{T}$ is some other computable copy of $\mathcal{S}$. We will show that there is a $\mathbf{d}$-computable isomorphism between $\mathcal{S}$ and $\mathcal{T}$. Using the relation R, we may identify the component of $\mathcal{T}$ isomorphic to $\mathcal{S}_\alpha$. Since $\mathcal{S}_a$ has (strong) degree of categoricity $\mathbf{0}^{(\alpha)} \leq \mathbf{d}$, we can $\mathbf{d}$-computably find an isomorphism between the copies of $\mathcal{S}_\alpha$ in $\mathcal{S}$ and $\mathcal{T}$. We can also identify the component isomorphic to $\mathcal{M}$ in each structure. By choice of $\mathcal{M}$, any two such copies have an isomorphism between them computable in $f \oplus \mathbf{0}^{(\alpha)}$, and D can compute this self-modulus f. Hence $\mathbf{d}$ can computably produce such an isomorphism, since it can compute $f \oplus \mathbf{0}^{(\alpha)}$. Gluing these two isomorphisms together gives us the result.

Since $\mathcal{S}_\alpha$ has strong degree of categoricity $\mathbf{0}^{(\alpha)}$, there is a computable copy $\hat{\mathcal{S}}_\alpha$ of $\mathcal{S}_\alpha$ such that every isomorphism between the two computes $\mathbf{0}^{(\alpha)}$. Let $\widetilde{\mathcal{S}}$ be a computable copy of $\mathcal{S}$ built in the following way. Rather than using the "standard" copy $\mathcal{S}_\alpha$, use the "hard" copy $\hat{\mathcal{S}}_\alpha$ of $\mathcal{S}_\alpha$. Additionally, rather than using $\mathcal{M}$, instead use $\mathcal{N}$ as built in Theorem 2.6. Any isomorphism between $\mathcal{S}_\alpha$ and $\hat{\mathcal{S}}_\alpha$ computes $\mathbf{0}^{(\alpha)}$, and any isomorphism between $\mathcal{M}$ and $\mathcal{N}$ must compute a function that dominates f. Let g be any isomorphism between $\mathcal{S}$ and $\widetilde{\mathcal{S}}$. Then by using R, we can restrict g to an isomorphism between $\mathcal{S}_\alpha$ and $\hat{\mathcal{S}}_\alpha$ and hence g can compute $\mathbf{0}^{(\alpha)}$. Since g can also be restricted to an isomorphism between $\mathcal{M}$ and $\mathcal{N}$, it must compute a function dominating f. But f is a modulus for D computable in $\mathbf{0}^{(\alpha)}$, and hence g must be able to compute D since it can compute $\mathbf{0}^{(\alpha)}$ and a function dominating f. Hence g can compute $\mathbf{d}$. $\square$

We now turn to prime models, working above $\mathbf{0}^{(\omega)}$. Essentially, our work here is to check that in taking $\alpha = \omega$ in the previous theorem and lemma, the construction results in a prime model.

Lemma 2.8.
Let $f : \omega \to \omega$ be limitwise monotonic relative to $\mathbf{0}^{(\omega)}$. There is a prime model with two computable copies $\mathcal{M}$ and $\mathcal{N}$ such that:

1. Every isomorphism between $\mathcal{M}$ and $\mathcal{N}$ computes a function which dominates f.
2. $f \oplus \mathbf{0}^{(\omega)}$ computes an isomorphism between any two computable copies of $\mathcal{M}$ and $\mathcal{N}$.

Proof:
The construction is exactly the same as that of Theorem 2.6 with $\alpha = \omega$. We refer to the structures $\mathcal{A}_\beta$ and $\mathcal{E}_\beta$ of Theorem 2.3 as $\mathcal{A}_n$ and $\mathcal{E}_n$, $n < \omega$, but of course these are the same. It remains to argue, using the properties from Theorem 2.4 which hold only for the structures $\mathcal{A}_n$ and $\mathcal{E}_n$ with n finite, that the resulting structure $\mathcal{N}$ is prime.

Recall that $\mathcal{N}$ is the disjoint union of structures $\mathcal{N}_n$, each of which satisfies the relation R_n. So it suffices to show that the structures $\mathcal{N}_n$ are prime. Also recall that $\mathcal{N}_n$ was defined as follows: there were infinitely many elements $(b_i)_{i\in\omega}$ (satisfying the unary relation S), each of which is attached to (by binary relations T_m), for each $m<\omega$, a box $\mathcal{N}_{i,m}$ which contains within it:

1. $\mathcal{N}_{i,m} \cong \mathcal{E}_m$ if there is s such that $\Phi^{\varnothing^{(m)}}(n,s) > i$.
2. $\mathcal{N}_{i,m} \cong \mathcal{A}_m$ otherwise.

By Claim 2.6.1 of Theorem 2.6, for each i, either $i < f(n)$ and there is some $m_i < \omega$ such that:

- for $\ell < m_i$, $\mathcal{N}_{i,\ell} \cong \mathcal{A}_\ell$,
- for $\ell \geq m_i$, $\mathcal{N}_{i,\ell} \cong \mathcal{E}_\ell$,

or $i \geq f(n)$ and for all $m < \omega$, $\mathcal{N}_{i,m} \cong \mathcal{A}_m$. Note that the sequence $\{m_i\}_{i<f(n)}$ is non-decreasing.

By Theorem 2.4 (2), for $i < f(n)$, the automorphism orbit of b_i is determined by the first-order formula with free variable x which expresses that S holds of x, that the structure with domain $T_{m_i}(x,\cdot)$ satisfies φ_{m_i} (and so is isomorphic to $\mathcal{E}_{m_i}$), and that the structure with domain $T_{m_i-1}(x,\cdot)$ satisfies $\neg\varphi_{m_i-1}$ (and so is isomorphic to $\mathcal{A}_{m_i-1}$). For $i \geq f(n)$, the automorphism orbit of b_i is determined by the first-order sentence with free variable x which expresses that S holds of x, and that the structure with domain $T_{m_{f(n)-1}}(x,\cdot)$ satisfies $\neg\varphi_{m_{f(n)-1}}$ (and so is isomorphic to $\mathcal{A}_{m_{f(n)-1}}$).

Fix a tuple $\bar{c}$ from $\mathcal{N}_n$. We will give a first-order formula defining the orbit of $\bar{c}$. We may assume that whenever $\bar{c}$ contains an element of $\mathcal{N}_{i,m}$, $\bar{c}$ contains b_i as well. We can break the tuple $\bar{c}$ up into finitely many elements $b_{i_1},\dots,b_{i_k}$ and finitely many tuples $\bar{c}_{i,m}$ from $\mathcal{N}_{i,m}$. The orbit of $\bar{c}$ is determined by the orbits of $b_{i_1},\dots,b_{i_k}$ (each of which is determined by a first-order formula as described in the previous paragraph), the fact that $T_m(b_i,y)$ holds for any $y \in \bar{c}_{i,m}$, and the orbits of each of the tuples $\bar{c}_{i,m}$ within $\mathcal{N}_{i,m}$. The latter orbits are first-order definable by Theorem 2.4 (3). $\square$

Theorem 2.9.
Let $\mathbf{d}$ be a degree c.e. in and above $\mathbf{0}^{(\omega)}$. There is a computable prime model $\mathcal{A}$ with strong degree of categoricity $\mathbf{d}$.

Proof:

The construction of such a model is similar to Theorem 2.7, except we replace $\mathcal{M}$ and $\mathcal{N}$ from Theorem 2.6 with those $\mathcal{M}$ and $\mathcal{N}$ from Theorem 2.8 (which are actually the same structures, if $\alpha = \omega$), and we also replace the "easy" and "hard" copies of $\mathcal{S}_\alpha$ with copies of the structure from Theorem 2.5 such that any isomorphism between them computes $\mathbf{0}^{(\omega)}$. The same argument from Theorem 2.7 shows that this new structure has strong degree of categoricity $\mathbf{d}$. It remains to show that such models are prime; they are the disjoint union of prime structures, distinguishable by the relation R, and hence must be prime themselves. □

Chapter 3

Coding Isomorphisms With Relations

The material in this chapter is joint work with Barbara Csima and Jonny Stephenson. The main details and the general course of the results were produced by equal discussion among all three co-authors. I was responsible for the details and write-up of each proof in Sections 3.2 to 3.4. In particular, I authored the entirety of Theorem 3.7 based on a short proof sketch produced by personal discussion with Stephenson. Much of the content excluding Section 3.4 appears in the joint work [10].

3.1 Background

In computable structure theory, we are interested in studying mathematical structures from a computable point of view, so it is natural to regard two computable copies of the same structure are being equivalent if they are isomorphic via some computable isomorphism. However, there are many examples of very standard structures for which there are computable copies which are not computably isomorphic. For example, consider $(\omega, <)$, the linear order with order type ω. We let $\mathcal{N}$ denote the usual decidable copy. One can readily construct another computable copy $\mathcal{A}$ of $(\omega, <)$ such that the unique isomorphism between $\mathcal{N}$ and $\mathcal{A}$ is of Turing degree $\mathbf{0}'$. In fact, in the case of $(\omega, <)$, it is easy to show that $\mathbf{0}'$ can compute the isomorphism between any two computable copies. So there is a sense in which $\mathbf{0}'$ is the degree of difficulty of computing isomorphisms between copies of $(\omega, <)$. This gives rise to Definition 1.2, but we repeat it here since it is particularly important:

Definition 1.2 (Fokina, Kalimullin and Miller [16]).
If $\{\mathbf{c} \mid \mathcal{A} \text{ is } \mathbf{c}\text{-computably categorical }\} = \{\mathbf{c} \mid \mathbf{c} \geq \mathbf{d}\}$, then we say that $\mathbf{d}$ is *the degree of categoricity* of $\mathcal{A}$.

If $\mathcal{A}$ has degree of categoricity $\mathbf{d}$ and there exist computable copies $\mathcal{A}_1$ and $\mathcal{A}_2$ of $\mathcal{A}$ such that every isomorphism $f : \mathcal{A}_1 \cong \mathcal{A}_2$ computes $\mathbf{d}$, we say $\mathcal{A}$ has *strong* degree of categoricity $\mathbf{d}$.

Finally, we say $\mathbf{d}$ is a (strong) degree of categoricity if there exists a computable structure with (strong) degree of categoricity $\mathbf{d}$.

In addition to introducing this definition, Fokina, Kalimullin and Miller [16] showed that every degree $\mathbf{d}$ which is 2-c.e. in and above $\mathbf{0}^{(n)}$ for some $n \in \omega$ is a strong degree of categoricity. This was extended by Csima, Franklin and Shore [12] to $\mathbf{0}^{(\alpha)}$ for any computable ordinal α, and degrees 2-c.e. in and above $\mathbf{0}^{(\alpha)}$ for computable successor ordinals α. Here 2-c.e. sets are the difference of c.e. sets, which can also be seen as a restricted version of ω-c.e. where we bound the number of changes by the constant 2 function.

Bazhenov, Kalimullin, and Yamaleev [5] as well as Csima and Stephenson [14] constructed structures that have a degree of categoricity, but no strong degree of categoricity.

Many of the above papers use the same approach for computing the degree of categoricity of the structures constructed. The structures are built in such a way that there is a computable unary relation U on one of the copies of the structure, so that if f is the isomorphism between the copies, then the isomorphism has degree $f(U)$.

This raises the natural question: To what extent is this possible?

The reader may note that we are restricting attention to computable subsets of the domain A of a structure $\mathcal{A}$, rather than, for instance, working within A^n for arbitrary n, or even in $A^{<\omega}$. Apart from our original motivating observations, there is a very straightforward reason for this restriction: if we work with tuples instead, then we can trivially recover the degree of an isomorphism f from $f(R)$, where R is chosen entirely independently of the structure we are working with:

Observation 3.1.
If $\mathcal{A}$ is a computable structure and $f : \mathcal{A} \to \mathcal{B}$ is an isomorphism to another computable copy $\mathcal{B}$, let $R \coloneqq \{(n, n+1)\}_{n \in \omega}$. Then $f(R) = \{(f(n), f(n+1))\}$ computes f.

Let us first examine the situation for unary relations on the linear order $(\omega, <)$. We begin with the following proposition.

Proposition 3.2.

Let $\mathcal{A}$ be any computable copy of $(\omega, <)$, and let $\mathcal{N}$ denote the standard decidable copy. Let $f : \mathcal{A} \to \mathcal{N}$ be the isomorphism between the two copies. Let $U \coloneqq \{m \mid (\exists n)[n <^{\mathcal{N}} m \wedge m <^{\mathcal{A}} n]\}$. Then $f(U) \equiv_T f$.

Proof Sketch:

Suppose we are given $\overline{f(U)}$. We show how to build the isomorphism f. First, find the least member of $\overline{f(U)}$, say n_0. Then we know there are exactly n_0-many numbers that are $<^{\mathcal{A}}$-below 0. We reveal the order $<^{\mathcal{A}}$ until we find $a_0 <^{\mathcal{A}} a_1 <^{\mathcal{A}} \cdots <^{\mathcal{A}} a_{n_0-1} <^{\mathcal{A}} a_{n_0} = 0$ and define $f(a_i) = i$. We have now defined f on an initial segment, and proceed inductively. □

However, there is an asymmetry with $(\omega, <)$, as we will see that there exists a computable copy $\mathcal{A}$ of $(\omega, <)$ such that for $f : \mathcal{N} \cong \mathcal{A}$ there is no computable U with $f(U) \equiv_T f$. Indeed, this will follow from an easy modification to the proof of the following Theorem.

Theorem 3.4.

There are two computable copies $\mathcal{A}$ and $\mathcal{B}$ of $(\omega, <)$ such that if $f : \mathcal{A} \to \mathcal{B}$ is the isomorphism between them, then f is of Turing degree $\mathbf{0}'$, and there is no computable set U such that $f(U) \equiv_T f$ or $f^{-1}(U) \equiv_T f$.

Note that Theorem 3.2 implies that for any computable copy $\mathcal{A}$ of $(\omega, <)$, if $f : \mathcal{A} \to \mathcal{N}$ is the isomorphism between $\mathcal{A}$ and the usual decidable copy of the order, there is a computable set U such that $f(U) \equiv_T f$; this exposes a fundamental distinction between effectiveness of isomorphisms mapping *into* and *out of* the standard copy $\mathcal{N}$ of $(\omega, <)$.

One might wonder whether this phenomenon is somehow more about mapping into and out of decidable structures, rather than about the particular choice of the structure $(\omega, <)$. This raises the question:

Question 3.3.

Suppose $\mathcal{A}$ is a computable structure and that $\mathcal{B}$ is a decidable copy isomorphic to $\mathcal{A}$. Suppose that the structures are rigid and that $f : \mathcal{A} \to \mathcal{B}$ is the isomorphism between them. Must there be a computable U such that $f(U) \equiv_T f$?

This conjecture turns out to be rather easy to dismiss; indeed, all we need to do is to look at $(\omega^2, <)$ rather than $(\omega, <)$.

Theorem 3.6.

Let $\mathcal{N}^2$ be a decidable copy of $(\omega^2, <)$. There is a computable copy $\mathcal{A}$ of $(\omega^2, <)$ such that if $f : \mathcal{A} \to \mathcal{N}^2$, then for no computable unary relation U on $\mathcal{A}$ do we have $f(U) \equiv_T f$.

Note that in Theorem 3.4, the isomorphism constructed has degree $\mathbf{0}'$, which is the degree of categoricity of the structure in question, $(\omega,<)$. However, the isomorphism f we construct above is clearly limit computable, so it does not have degree $\mathbf{0}'''$, the degree of categoricity of $(\omega^2,<)$. So in this sense, Theorem 3.6 is a weak analogue of Theorem 3.4. However, it is possible to modify the construction above to combine it with the method of the theorem for the $(\omega,<)$ case to code at least $\varnothing''$ into the isomorphism.

Theorem 3.7.
Let $\mathcal{N}^2$ be a decidable copy of $(\omega^2,<)$. There is a computable copy $\mathcal{A}$ of $(\omega^2,<)$ such that if $f:\mathcal{A}\to\mathcal{N}^2$, then for no computable unary relation U on $\mathcal{A}$ do we have $f(U)\equiv_T f$, and furthermore, $f \geq_T \varnothing''$.

Although this is not enough to show that $f \equiv_T \varnothing'''$, it is a good indicator that such a proof may be quite difficult, since although Theorem 3.6 is only finite injury, the additional constraint in Theorem 3.7 naturally pushes one to use an infinite injury argument. (See Soare [28] for a detailed introduction to injury proofs, both finite and infinite.) We present both Theorem 3.6 and Theorem 3.7 since it is easier to understand the requirements in the simpler setting of the former, and allows us to highlight only the differences in the proof of Theorem 3.7.

Section 3.2 is devoted to the proof of Theorem 3.4, Section 3.3 to proof of Theorem 3.6 and Section 3.4 to the proof of Theorem 3.7.

3.2 Isomorphisms on copies of $(\omega,<)$

This section is devoted to the proof of Theorem 3.4, which we restate here.

Theorem 3.4.
There are two computable copies $\mathcal{A}$ and $\mathcal{B}$ of $(\omega,<)$ such that if $f:\mathcal{A}\to\mathcal{B}$ is the isomorphism between them, then f is of Turing degree $\mathbf{0}'$, and there is no computable set U such that $f(U)\equiv_T f$ or $f^{-1}(U)\equiv_T f$.

Proof:
We aim to meet, for all $e,j\in\omega$, the following requirements:

$\mathbf{R}_{\langle e,j\rangle}$: If $\varphi_e = U$ for some set U, then $(\exists x)[\Phi_j^{f(U)}(x)\neq f(x)]$, and

$\mathbf{S}_{\langle e,j\rangle}$: If $\varphi_e = U$ for some set U, then $(\exists x)[\Phi_j^{f^{-1}(U)}(x)\neq f^{-1}(x)]$.

To do this, we will build $\mathcal{A}$ and $\mathcal{B}$ by stages, enumerating the least unused element into the domains of $\mathcal{A}$ and $\mathcal{B}$, and perhaps more, at each stage. We will enforce that there are only finitely many enumerations at any given position, so that $\mathcal{A}$ and $\mathcal{B}$ are isomorphic to $(\omega, <)$. At each stage s, we let $\mathcal{A}_s$ and $\mathcal{B}_s$ denote the partially constructed portions of their respective structures at that stage, and f_s the partial isomorphism between $\mathcal{A}_s$ and $\mathcal{B}_s$. Note that by enumerating into $\mathcal{A}$ and $\mathcal{B}$ at different positions, we can force that $f_s(x) \neq f_{s+1}(x)$ or $f_s^{-1}(y) \neq f_{s+1}^{-1}(y)$, so $f \coloneqq \lim_{s\to\infty} f_s$ need not extend any f_s. Nevertheless, since any given position will change at most finitely often, f will be computably approximable by this sequence of partial isomorphisms. That is, f will extend longer and longer initial segments of the partial isomorphisms, so that for a fixed initial segment of f, there is eventually some stage after which all partial isomorphisms extend that initial segment.

An important note: although we aim to explicitly meet all requirements through the construction, we will only implicitly meet some of them. If φ_e is the characteristic function of some set U which is finite or cofinite, then we do not need to meet $\mathbf{R}_{\langle e,j\rangle}$ or $\mathbf{S}_{\langle e,j\rangle}$ explicitly for any j — that is, declaring a witness x or y during the construction that eventually shows the requirement is met. Instead, since $f(U)$ and $f^{-1}(U)$ will be finite or cofinite (hence computable), we will automatically have that $f(U) \not\geq_T f$ or $f^{-1}(U) \not\geq_T f$, provided we make f non-computable. We will make $f \equiv_T \varnothing'$, so in particular, f will be non-computable. However, we cannot know which indices e correspond to the finite or cofinite sets, so we must still ensure that they do not stall the construction, even if the actions that they take do not explicitly ever meet their requirements.

Because we are working with partial approximations to computable sets, we set the following notation. At stage s for index e, we let $\sigma_{e,s}$ be the string defined by the longest segment of $\varphi_{e,s}$ that looks like a characteristic function. Thus $|\sigma_{e,s}| \leq s$ by conventions on convergence. Hence if $\varphi_e = U$, then $\lim_{s\to\infty} \sigma_{e,s} = U$.

The plan for meeting a single $\mathbf{R}_{\langle e,j\rangle}$ is to choose a witness x and wait for $\Phi_{j,s}^{f(\sigma_{e,s})}(x)\downarrow = f(x)$. If this happens, we place an immediate $<^{\mathcal{B}}$-predecessor to $f(x)$. However, this may have the unfortunate side-effect of also causing the use of the computation $\Phi_{j,s}^{f(\sigma_{e,s})}(x)$ to be damaged, since it may be the case that $f_s(\sigma_{e,s})$ no longer agrees with $f_{s+1}(\sigma_{e,s+1})$. This is because the enumeration of this predecessor value will change the alignment of all values above it in $\mathcal{B}_{s+1}$. Thus, after such an enumeration – which attempts to diagonalize against some computation – we may wish to restore that computation if it was damaged by aligning the values used in the computation so that they are again either in or out of $f(U)$ in the limit.

Notice that we do not have to re-align these values with the exact same values as the ones they were aligned with when the computation first existed. For example, if at stage s the use of a computation contained 5 with $f_s(3) = 5$ and $\varphi_{e,s}(3)\downarrow = 1$, (that is,

$5 \in f_s(\sigma_{e,s})$) then if we wish to restore this computation, we only need to find a value n such that $n \in U$ and arrange for $f(n) = 5$, since then $5 \in f(U)$. So, as long as U is both infinite and co-infinite, we can always wait for a sequence of values in the correct order to appear and then arrange for them to be aligned with the use that we wish to restore. We cannot stall the computation waiting for these values, however, since we do not know if φ_e determines an infinite, co-infinite set, so we shall instead wait for such a configuration to appear. If one does not, then we shall show that φ_e does not determine an infinite, co-infinite set and therefore we do not have to satisfy the requirement directly, as noted above.

Additionally, we shall have markers $\{\gamma_i\}_{i\in\omega}$ that will code $\varnothing'$. We shall arrange these markers so that if γ_i eventually comes to rest on x, then $i \in \varnothing'$ if and only if $i \in K_{f(x)}$, where here $\{K_s\}_{s\in\omega}$ is a standard enumeration of $\varnothing'$. To show that $f \geq_T \varnothing'$, we will show that f allows us to additionally compute these final resting places of each γ_i. Since any isomorphism between two copies of $(\omega, <)$ is $\varnothing'$-computable, this is all we need to show that $f \equiv_T \varnothing'$. For each marker, we have the following requirement, which we aim to meet:

$\mathbf{\Gamma}_i$: Each γ_i eventually comes to rest on some value z_i such that $i \in \varnothing'$ if and only if $i \in K_{f(z_i)}$.

To enable f to compute the final resting places of each marker, we shall take certain actions to leave a trace of when each $\mathbf{\Gamma}_i$ is injured. Since a given $\mathbf{\Gamma}_i$ can only be injured by higher priority requirements, we will have these higher priority requirements leave the trace when they act, which allows us to use f to determine a stage where all requirements of priority higher than a given $\mathbf{\Gamma}_i$ have finished acting. By simulating the construction to this stage and then waiting – if we need to – for a stage where $\mathbf{\Gamma}_i$ is (re-)initialized, we can then determine the final position of γ_i, since it will never be injured after this stage.

We arrange the requirements according to the priority order $\mathbf{R}_0 > \mathbf{S}_0 > \mathbf{\Gamma}_0 > \mathbf{R}_1 > \mathbf{S}_1 > \mathbf{\Gamma}_1 > \cdots$ and proceed via a finite injury construction.

At any stage $s+1$, each requirement of index less than s will have a witness value. For the requirement $\mathbf{R}_{\langle e,j\rangle}$, we denote this witness by $x_{\langle e,j\rangle}$, and similar for $\mathbf{S}_{\langle e,j\rangle}$, we write $y_{\langle e,j\rangle}$.

Additionally, during the course of the construction, we might associate (the graph of) a finite characteristic partial function, say g, with a requirement. When we do this, we say that the requirement has *assigned function* g. At certain stages, we may also declare a requirement to be *satisfied*. This means that at such stages the requirement believes that it has no more action to take and will be met. This is reset by injury, however, since the requirement may be forced to abandon its witness by higher priority requirements.

We will enumerate the least value not in the domain of $\mathcal{A}_s$ at the end of stage s, to ensure that $\mathcal{A}$ eventually has domain ω. This also serves a second purpose: any value enumerated during stage s must be at least s. This value will be enumerated at the end of $\mathcal{A}_s$, i.e. we will enumerate it and declare it to be $<^{\mathcal{A}_s}$-larger than any other value in the domain of $\mathcal{A}_s$. This entire procedure is repeated for $\mathcal{B}$.

At each stage s, each requirement may require attention and then possibly receive it. We say that a requirement that is not satisfied requires attention under the following conditions. We only list the $\mathbf{R}_i$ requirements and the $\mathbf{\Gamma}_i$ requirements, as the conditions and actions for the $\mathbf{S}_i$ requirements are similar to the $\mathbf{R}_i$ requirements once the obvious changes have been made.

- $\mathbf{R}_{\langle e,j\rangle}$ *requires attention for diagonalization* if $\Phi_{j,s}^{f_s(\sigma_{e,s})}(x_{\langle e,j\rangle})\downarrow = f_s(x_{\langle e,j\rangle})$ and it has no assigned function.

  If this requirement receives attention for this reason, then enumerate the least value not in the domain of $\mathcal{B}_s$ so that it is just below $f_s(x_{\langle e,j\rangle})$, so that we have that $f_{s+1}(x_{\langle e,j\rangle}) \neq f_s(x_{\langle e,j\rangle})$. Let g denote the characteristic function of the use segment of the computation $\Phi_{j,s}^{f_s(\sigma_{e,s})}(x_{\langle e,j\rangle})$. That is, the domain of g is the set of values in the domain of $\mathcal{B}_s$ used in this computation. Assign g to this requirement.

- $\mathbf{R}_{\langle e,j\rangle}$ *requires attention for restoration* if it has assigned function g and for the elements of the domain of g exceeding $f_s(x_{\langle e,j\rangle})$, say $f_s(x_{\langle e,j\rangle}) <^{\mathcal{B}_s} d_1 <^{\mathcal{B}_s} d_2 <^{\mathcal{B}_s} \cdots <^{\mathcal{B}_s} d_k$, there are elements in the domain of $\mathcal{A}_s$ say, $a_1 <^{\mathcal{A}_s} a_2 <^{\mathcal{A}_s} \cdots <^{\mathcal{A}_s} a_k$ such that $\sigma_{e,s}(a_i)\downarrow = g(d_i)$ and such that $d_1 <^{\mathcal{B}_s} f_s(a_1)$. (This last condition ensures that the d_is and a_is are not already aligned, which is important since we must make an enumeration no matter what for coding a trace of injury, and this enumeration would mis-align them if they were already so.)

  If this requirement receives attention for this reason, then enumerate the least value not in the domain of $\mathcal{B}_s$ so that it is just below $f_s(x_{\langle e,j\rangle})$. Next enumerate unused values under each d_i in $<^{\mathcal{B}_s}$-increasing order so that $f_{s+1}(a_i) = d_i$. We may assume that this is always possible by noting that the values $d_1, \ldots, d_k$ were originally $<^{\mathcal{B}_s}$-consecutive, as they were the tail of the use segment for a computation, and we will enforce that enumerations between these values (by a higher priority requirement) would re-initialize this requirement, so the values must still be $<^{\mathcal{B}_s}$-consecutive. Hence with the correct pattern of enumerations, we can align each d_i with its corresponding a_i. Furthermore, the first enumeration just below

$f_s(x_{\langle e,j\rangle})$ will not inhibit our ability to do this, because $d_1 <^{\mathcal{B}_s} f_s(a_1)$. Declare that $\mathbf{R}_{\langle e,j\rangle}$ is satisfied.

- $\mathbf{\Gamma}_i$ *requires attention* if $i \in K_s$.

  If this requirement receives attention for this reason, then enumerate the least value not in the domain of $\mathcal{B}_s$ so that it is just below $f_s(z_i)$, where z_i is the value currently marked by γ_i. Declare that $\mathbf{\Gamma}_i$ is satisfied.

Note that if $\mathbf{R}_i$ receives attention at stage s and has witness x, then $f_{s+1}(x) \geq s$ since we enumerate a fresh value in this spot in both cases. Similarly, if $\mathbf{S}_i$ receives attention at stage s and has witness y, then $f^{-1}_{s+1}(y) \geq s$, and if $\mathbf{\Gamma}_i$ receives attention at stage s and γ_i marking z_i, then $f_{s+1}(z_i) \geq s$. In this way, the witness / marked values of each requirement codes the stage where it has caused injury most recently (if at all).

Construction:

> **Stage** s: For the highest priority active requirement that requires attention at stage s, perform the action indicated above and injure all lower priority requirements, de-activating them. For the highest priority requirement that is not active, (re-)initialize it, assigning it a fresh large witness / marked value as needed. These values are chosen larger than the use of any computation seen so far and larger than any values ever used as witnesses or marked values.
>
> Enumerate the least value not in the domain of $\mathcal{A}_s$ as the final, largest element. Proceed similarly for $\mathcal{B}_s$.

Let $\mathcal{A} := \bigcup_s \mathcal{A}_s$, $\mathcal{B} := \bigcup_s \mathcal{B}_s$ and $f := \lim_{s\to\infty} f_s$. This completes the construction.

Note that since witnesses are always chosen larger than any existing witness, and enumerations always occur (for that witness) at most just below it, any given position can only be enumerated into finitely often, provided we show that each requirement receives attention at most finitely often.

Claim 3.4.1: Each requirement receives attention at most finitely often.

Proof of Claim: We proceed by induction on the priority order. Suppose that we have requirement $\mathbf{R}_{\langle e,j\rangle}$ (the case for $\mathbf{S}_{\langle e,j\rangle}$ is similar) and that all higher priority requirements receive attention at most finitely often. So there exists a stage s after which all higher priority requirements never again receive attention, and hence after stage s, $\mathbf{R}_{\langle e,j\rangle}$ will permanently choose a witness, x. We may assume that this stage is also s. If $\mathbf{R}_{\langle e,j\rangle}$ never receives attention after stage s, then we are done. Since $\mathbf{R}_{\langle e,j\rangle}$ has no assigned function when it is re-initialized, the first stage $t \geq s$ where $\mathbf{R}_{\langle e,j\rangle}$ requires attention (and receives it, since it is of highest priority by choice of s) must be for diagonalization, and $\mathbf{R}_{\langle e,j\rangle}$ will be assigned some function g.

As $\mathbf{R}_{\langle e,j\rangle}$ will never be re-initialized by choice of s, if $\mathbf{R}_{\langle e,j\rangle}$ requires attention after stage t, it must be for restoration. Receiving attention for restoration causes $\mathbf{R}_{\langle e,j\rangle}$ to be declared satisfied, and this will never change. So $\mathbf{R}_{\langle e,j\rangle}$ cannot receive attention again.

Now suppose that we have requirement $\mathbf{\Gamma}_i$, and that again all higher priority requirements receive attention at most finitely often, finishing by stage s. If $\mathbf{\Gamma}_i$ never receives attention after stage s, then we are done, and if it does require attention, it must receive it (as it is of highest priority) and will be declared satisfied, and this will never change. So $\mathbf{\Gamma}_i$ cannot receive attention again. ■

Claim 3.4.2: For each infinite, coinfinite, computable set U and index j, there is some x such that $\Phi_j^{f(U)}(x) \neq f(x)$. That is, if φ_e is the characteristic function for U, then requirement $\mathbf{R}_{\langle e,j\rangle}$ is met.

Proof of Claim: Fix a computable set U with $\varphi_e = U$ and index j. Suppose otherwise, so that $\Phi_j^{f(U)} = f$. By the previous claim, there is some stage s after which $\mathbf{R}_{\langle e,j\rangle}$ never receives attention and has permanent witness x. Choose $t \geq s$ large enough so that $\Phi_{j,t}^{f(U)}(x) = f(x)$, and also large enough so that after stage t, any partial isomorphism $f_{t'}$ extends the initial segment of f given by the use of this computation. That is, if n is in the use of this computation, and $f^{-1}(n) = m$, then m is in the domain of $\sigma_{e,t}$ and $f_{t'}(m) = f(m)$ for all $t' \geq t$. Such a stage exists eventually since φ_e really is a characteristic function so $\sigma_{e,t}$ can be chosen to be arbitrarily long, and f is the limit of the sequence $\{f_s\}_s$.

Since $\mathbf{R}_{\langle e,j\rangle}$ is not re-initialized after stage s, all requirements that are not of lower priority must have stopped enumerating values, and all lower priority requirements only enumerate values above the witness x of $\mathbf{R}_{\langle e,j\rangle}$. Hence at stage t, we have that $f_t(x) = f(x)$, and also by choice of t we have that $\Phi_{j,t}^{f_t(U)}(x)\downarrow = \Phi_j^{f(U)}(x)\downarrow$.

So, at stage t, we have that $\Phi_{j,t}^{f_t(\sigma_{e,t})}(x) = \Phi_j^{f(U)}(x)\downarrow = f(x) = f_t(x)$. Then $\mathbf{R}_{\langle e,j\rangle}$ should receive attention for diagonalization, but this is impossible since we are beyond stage s. So it must be the case that $\mathbf{R}_{\langle e,j\rangle}$ has an assigned function g.

We claim that $\mathbf{R}_{\langle e,j\rangle}$ cannot permanently have a function assigned without eventually requiring attention for restoration. Let the elements of the domain of g exceeding $f(x) = f_t(x)$ be $f(x) <^{\mathcal{B}} d_1 <^{\mathcal{B}} d_2 <^{\mathcal{B}} \cdots <^{\mathcal{B}} d_k$. Since U is infinite and coinfinite, there must be values $a_1 < a_2 < \cdots < a_k$ such that $U(a_i) = g(d_i)$ for all $i \leq k$. So wait for a stage

$t' \geq t$ where $\sigma_{e,t'}(a_i)\downarrow$. Then at stage t', $\mathbf{R}_{\langle e,j\rangle}$ would receive attention for restoration at stage t'. But this is impossible, since $t' \geq t \geq s$.

So it must have been that at stage t, $\mathbf{R}_{\langle e,j\rangle}$ is already declared satisfied. This implies that there was a stage t_1 where $\mathbf{R}_{\langle e,j\rangle}$ received attention for diagonalization and was assigned some function g, then at some later stage t_2 received attention for restoration, and was marked as satisfied, and then was never re-initialized.

At stage t_1, it must be that $\Phi_{j,t_1}^{f_{t_1}(\sigma_{e,t_1})}(x)\downarrow = f_{t_1}(x)$. A value was enumerated into $\mathcal{B}$ so that $f_{t_1+1}(x) \neq f_{t_1}(x)$, and $\mathbf{R}_{\langle e,j\rangle}$ was assigned the function g whose domain is the set of values in $\mathcal{B}_{t_1}$ used in this computation, with $g(a_i) = \sigma_{e,t_1}(a_i) = \varphi_e(a_i) = U(a_i)$. Since no higher priority requirements receive attention and enumerate values, all the values in $\mathcal{B}_{t_1}$ that are $<^{\mathcal{B}}$-below $f_{t_1}(x)$ are never enumerated below, and all the values in $\mathcal{A}_{t_1}$ that are $<^{\mathcal{A}}$-below x are never enumerated below. So at all future stages $\hat{t} > t_1$, we have that these values are in $\sigma_{e,\hat{t}}$ if and only if they are in σ_{e,t_1}. Hence for these d_i, we have that $d_i \in f_{\hat{t}}(\sigma_{e,\hat{t}})$ if and only if $d_i \in f_{t_1}(\sigma_{e,t_1})$.

At stage t_2, values are enumerated into $\mathcal{B}$ so that $f_{t_2+1}(a_i) = d_i$ where d_i are the values in the domain of g not considered above, and a_i is chosen so that $U(a_i) = \sigma_{e,t_2}(a_i) = g(d_i)$. As no values are then ever enumerated into $\mathcal{A}$ or $\mathcal{B}$ to destroy this, we have that for any stage $\hat{t} > t_2$, these values d_i have that $d_i \in f_{\hat{t}}(\sigma_{e,\hat{t}})$ if and only if $\sigma_{e,\hat{t}}(a_i) = g(d_i) = 1$, and $g(d_i) = 1$ exactly when $d_i \in f_{t_1}(\sigma_{e,t_1})$.

Hence we must have that for all such stages $\hat{t} > t_2 > t_1$ we have that $\Phi_{j,\hat{t}}^{f_{\hat{t}}(\sigma_{e,\hat{t}})}(x)\downarrow = \Phi_{j,t_1}^{f_{t_1}(\sigma_{e,t_1})}(x)\downarrow = f_{t_1}(x) \neq f_{\hat{t}}(x)$. Therefore this holds in the limit, so $\Phi_j^{f(U)}(x)\downarrow \neq f(x)$, and so the requirement is met. ∎

An extremely similar argument shows that the following claim holds:

Claim 3.4.3: For each infinite, coinfinite, computable set U and index j, there is some y such that $\Phi_j^{f^{-1}(U)}(y) \neq f^{-1}(y)$. That is, if φ_e is the characteristic function for U, then requirement $\mathbf{S}_{\langle e,j\rangle}$ is met.

Therefore, as noted above, *all* $\mathbf{R}_i$ requirements and $\mathbf{S}_i$ requirements are met, since the remaining requirements are automatically met once f is non-computable, and this must be the case, for the previous claims could not be true if f were computable.

Claim 3.4.4: Given f, we can compute the final resting position of each γ_i. Furthermore, f can compute $\varnothing'$.

Proof of Claim: To determine the final marked value of some γ_i, notice that it suffices to determine a stage after which no requirement of higher priority than $\mathbf{\Gamma}_i$ ever receives attention. Once we know such a stage, we can run the construction to that stage and then wait for $\mathbf{\Gamma}_i$ to be initialized and mark some value z. Since no higher priority requirement will ever receive attention after this initialization, $\mathbf{\Gamma}_i$ cannot be injured, and must mark z forever after. That is, z is the final resting position of γ_i.

This then allows us to decide if $i \in \varnothing'$: Note that if i ever enters $\varnothing'$, $\mathbf{\Gamma}_i$ will require attention and receive it at some point once it has marked z, since no higher priority requirement ever receives attention after z has been marked. Say this occurs at stage t. In this case, we will enumerate a fresh large value into the domain of $\mathcal{B}$ just below $f_t(z)$, so that $f_t(z) \geq t$ – since values enumerated at stage t must be at least t. Since no enumeration can take place below this point, we must have $f(z) \geq t$. So if i enters $\varnothing'$ by stage t, then $f(z) \geq t$. Hence to decide if $i \in \varnothing'$, compute $s \coloneqq f(z)$ and then determine if $i \in K_s$.

It remains to show that, given f, we can determine a stage after which no higher priority requirement ever receives attention. Proceed by induction on the priority order. Note: We need to include *all* types of requirements in this induction, not just $\mathbf{\Gamma}_j$s. Suppose we have a requirement $\mathbf{Q}$ of some type and using f we can determine a stage s after which no requirement of priority higher than $\mathbf{Q}$ receives attention. Run the construction to stage s and then wait for $\mathbf{Q}$ to be (re-)initialized. We need to determine a stage t after which $\mathbf{Q}$ never receives attention.

If $\mathbf{Q} = \mathbf{R}_j$ for some j, then notice that if such a requirement receives attention at stage t, then it enumerates a value into the domain of $\mathcal{B}$ such that $f_t(x_j) \geq t$. Since no higher priority requirement can disrupt this, we would have $f(x_j) \geq t$. Hence, we use f to compute $t \coloneqq f(x_j)$. We need to wait for $\mathbf{R}_j$ to be initialized above so that we can determine what its witness x_j is.

Similarly, if $\mathbf{Q} = \mathbf{S}_j$ for some j, then we can compute $t \coloneqq f^{-1}(y_j)$, where y_j is the witness chosen for $\mathbf{S}_j$ when it is (re-)initialized for the final time after stage s.

Finally, if $\mathbf{Q} = \mathbf{\Gamma}_j$ for some j, then again notice that if such a requirement receives attention at stage t, it enumerates a value into the domain of $\mathcal{B}$ such that $f_t(z_j) \geq t$, where z_j is the value marked by $\mathbf{\Gamma}_j$. So again, we use f to compute $t \coloneqq f(z_j)$.

This concludes the induction. We can use f to determine up to what stage to run the construction for the highest priority requirement to stop receiving attention, run the construction until the next requirement is (re-)initialized and then repeat this for each successive requirement under the priority ordering until we can determine when a given $\mathbf{\Gamma}_i$ marks its final value, which, as noted above, allows us to decide if $i \in \varnothing'$ using f once again.

So $f \geq_T \varnothing'$, and hence $f \equiv_T \varnothing'$. ■

Since $\mathcal{A}$, $\mathcal{B}$ and f are as claimed, this completes the proof. □

Note that if we remove the $\mathbf{S}_i$ requirements, then no enumerations occur into $\mathcal{A}$ except for the ones that occur at the end of each stage, which always occur at the end of the current segment $\mathcal{A}_s$, and so $\mathcal{A}$ will be the standard copy of $(\omega, <)$. Hence we also have the following:

Corollary 3.5.
There is a computable copy $\mathcal{B}$ of $(\omega, <)$ such that if $f : \mathcal{N} \to \mathcal{B}$ is the isomorphism between the standard copy of $(\omega, <)$ and $\mathcal{B}$, then $f \equiv_T \varnothing'$ and no computable set U exists such that $f(U) \equiv_T f$.

3.3 Isomorphisms on copies of $(\omega^2, <)$ – Finite Injury

This section is devoted to the proof of Theorem 3.6, which we restate here.

Theorem 3.6.
Let $\mathcal{N}^2$ be a decidable copy of $(\omega^2, <)$. There is a computable copy $\mathcal{A}$ of $(\omega^2, <)$ such that if $f : \mathcal{A} \to \mathcal{N}^2$, then for no computable unary relation U on $\mathcal{A}$ do we have $f(U) \equiv_T f$.

Proof:
To build $\mathcal{A}$, we will meet the following requirements:

$\mathbf{R}_{\langle e,j \rangle}$: If $\varphi_e = U$ for some set U, then $(\exists x)[\Phi_j^{f(U)}(x) \neq f(x)]$.

We build $\mathcal{A}$ by stages, enumerating finitely many values into the domain of $\mathcal{A}$. Since at any stage s, $\mathcal{A}_s$ will be isomorphic to some $n < \omega$, we shall guarantee $\mathcal{A} := \cup_s \mathcal{A}_s$ is isomorphic to ω^2 by having an infinite sequence of markers that are promised to be the limit points. Although a marker may occasionally change its value, we will ensure that this occurs at most finitely often (in fact, at most twice) so that eventually each marker settles. We will also ensure that all limit points in $\mathcal{A}$ arise in this way, by ensuring that only marked values have new values enumerated into the domain of $\mathcal{A}$ directly below them infinitely often. This allows us to make strong claims about uses of computations that occur in the limit.

At the end of every stage s, we enumerate new values into the domain of $\mathcal{A}_s$, A_s in a way to eventually forced the marked values to become limit points, provided the markers do not change. We refer to this action as "upkeep", since it maintains the guarantee that the marked values become limit points. For a marked value x, we define the *tail* of x to be the set $\{y \geq^{\mathcal{A}} x \mid (\forall z)[x <^{\mathcal{A}} z \leq^{\mathcal{A}} y \Rightarrow z \text{ is not marked}]\}$. That is, the tail of

x is the smallest set containing x and closed under unmarked successors. The upkeep action consists of enumerating a single new value into A_s at the end of every marker's tail.

For example, if at the end of stage s we have $\mathcal{A}_s$ as

$$\begin{array}{cccccccc} \boxed{\bullet} & \bullet & \bullet & \boxed{\bullet} & \bullet & \bullet & \boxed{\bullet} & \boxed{\bullet} \\ 0 & 1 & 3 & 2 & 5 & 4 & 6 & 7 \end{array}$$

where the marked values are boxed, then when we perform this step, we enumerate four new values into the domain of $\mathcal{A}$ – 8, 9, 10 and 11 – so that $\mathcal{A}_s$ becomes

$$\begin{array}{cccccccccccc} \boxed{\bullet} & \bullet & \bullet & \bullet & \boxed{\bullet} & \bullet & \bullet & \bullet & \boxed{\bullet} & \bullet & \boxed{\bullet} & \bullet \\ 0 & 1 & 3 & 8 & 2 & 5 & 4 & 9 & 6 & 10 & 7 & 11 \end{array}$$

It is clear that if the markers eventually settle, then $\mathcal{A}$ will be isomorphic to ω^2 through this procedure, provided that there are infinitely many markers, and provided we do not disrupt this as mentioned above, by building non-marked limit points, or by a more obviously destructive action, such as building an ω^* somewhere, for example.

In light of this, we will think of each marker as corresponding to a potential ω-chain in $\mathcal{A}$ consisting of the marked value and its eventual infinite tail.

To meet a single requirement $\mathbf{R}_{\langle e,j\rangle}$ in isolation, we employ the following strategy. Choose some witness value, say x_0, and mark it, so that it is associated with some limit point, and choose some other value ℓ to the left of x_0 and mark it as well. For simplicity, we shall suppose for this single requirement that ℓ is associated with the limit point 0 and x_0 is associated with the limit point ω. (We also mark infinitely many values to the right of x_0 so that we build a structure isomorphic to ω^2, but those values are unimportant for now.) If no other action is taken, then the upkeep action detailed above will build $\mathcal{A}$ isomorphic to ω^2 via f, such that $f(\ell) = 0$ and $f(x_0) = \omega$. If the requirement is not met, then $\varphi_e = U$ and $(\forall x)[\Phi_j^{f(U)}(x) = f(x)]$, and so in particular, we would have $\Phi_j^{f(U)}(x_0) = \omega = f(x_0)$. So, if we see a computation of this form at some stage s, we seek to diagonalize against it, by introducing a new value x_1 to the immediate right of x_1, and moving the marker from x_0 to x_1. This has the effect of making x_1 the value associated with ω, and pushes x_0 into the tail of ℓ, so that x_0 is now associated with some $m \in \omega$.

Unfortunately, this action may destroy the use of the original computation that we were diagonalizing against. Notice, however, that the only affected value is ω. So, if $\varphi_e(x_0) = \varphi_e(x_1)$, then $x_0 \in U$ if and only if $x_1 \in U$. Hence the computation will be restored, as then $\omega \in f(U) \Leftrightarrow x_1 \in U \Leftrightarrow x_0 \in U \Leftrightarrow \omega \in f(U)[s]$. Then we will have $\Phi_j^{f(U)}(x_0) = \omega$, but $f(x_0) = m \neq \omega$, a win.

In the case where $\varphi_e(x_0) \neq \varphi_e(x_1)$, we perform the same trick, but this time using x_1 in place of x_0. So, we wait for a stage where $\Phi_j^{f(U)}(x_1) = \omega = f(x_1)$ and then introduce a new value x_2 to the immediate right of x_1 and move the marker from x_1 to x_2. This pushes x_1 also into the tail of ℓ, and we now win automatically: If $\varphi_e(x_1) = \varphi_e(x_2)$, then the argument above works with x_0 and x_1 replaced by x_1 and x_2, respectively. On the other hand, if $\varphi_e(x_1) \neq \varphi_e(x_2)$, then $\varphi_e(x_0) = \varphi_e(x_2)$, since φ_e would be $\{0,1\}$-valued, and so this restores the original computation we diagonalized against when introducing x_1, so $\Phi_j^{f(U)}(x_0) = \omega$, but $f(x_0) = m \neq \omega$.

In this way, through at most two actions, we can always diagonalize against a computation, or wait forever for such a computation, which is also a win. Notice that in meeting this single requirement, we only needed to consider two marked values, ℓ and one of x_0, x_1 or x_2. So, we can satisfy a single requirement in a $\omega + \omega$ inside ω^2.

So we group the limit points of ω^2 into consecutive pairs, as $(0,\omega), (\omega\cdot 2, \omega\cdot 3), \ldots$ and use these pairs as locations to satisfy each requirement. When a requirement is injured, it abandons its associated pair and is assigned a fresh large pair. Of course, this may require enumerating new values at the end of $\mathcal{A}_s$ and marking them, so as to "create" a new pair of potential limit points. Since each pair of marked values and their tails will eventually correspond to an $\omega\cdot 2$ inside $\mathcal{A}$, we refer to the pair of marked values itself as an $\omega\cdot 2$, and so we may speak of "creating a fresh large $\omega\cdot 2$ at stage s", for instance, even though this only truly refers to enumerating two values at the end of $\mathcal{A}_s$ and marking them.

As we can see above, each requirement will also have a state: it is either waiting for a computation involving x_0, waiting for a computation involving x_1 or has won. When computations are (re-)initialized, they will always be waiting for x_0.

For a given requirement $\mathbf{R}_{\langle e,j\rangle}$, we will say that $\mathbf{R}_{\langle e,j\rangle}$ *requires attention at stage s* under the following conditions:

- If $\mathbf{R}_{\langle e,j\rangle}$ is waiting for x_0, then its $\omega\cdot 2$ has the form

  $$\underset{\ell}{\boxed{\bullet}}\cdots\underset{x_0}{\boxed{\bullet}}\cdots \quad .$$

  The requirement requires attention if $\Phi_j^{f(U)}(x_0)[s]\downarrow = f_s(x_0)$, and if it receives attention, then enumerate a new value, x_1, directly to the right of x_0, and move the marker from x_0 to x_1. The $\omega\cdot 2$ for the requirement will then have the form

$$\begin{array}{ccccc} \boxed{\bullet} & \cdots & \bullet & \boxed{\bullet} & \cdots \\ \ell & & x_0 & x_1 & , \end{array}$$

and it will be waiting for x_1.

- If $\mathbf{R}_{\langle e,j\rangle}$ is waiting for x_1, then its $\omega \cdot 2$ has the form

  $$\begin{array}{ccccccc} \boxed{\bullet} & \cdots & \bullet & \cdots & \boxed{\bullet} & \cdots\cdots \\ \ell & & x_0 & & x_1 & . \end{array}$$

  The requirement requires attention if either $\varphi_{e,s}(x_0)\downarrow = \varphi_{e,s}(x_1)\downarrow$, or if $\varphi_{e,s}(x_0)\downarrow \neq \varphi_{e,s}(x_1)\downarrow$ and $\Phi_j^{f(U)}(x_1)[s]\downarrow = f_s(x_1)$. In the former case, no further action is needed, and the requirement is met (unless it is later injured).

  In the latter case, if the requirement receives attention then enumerate a new value, x_2, directly to the right of x_1, and move the marker from x_1 to x_2. The $\omega \cdot 2$ for the requirement will then have the form

  $$\begin{array}{cccccccc} \boxed{\bullet} & \cdots & \bullet & \cdots & \bullet & \boxed{\bullet} & \cdots\cdots \\ \ell & & x_0 & & x_1 & x_2 & \end{array}$$

  and the requirement will be met (again, unless it is later injured).

Here, when we write something like $\Phi_j^{f(U)}(x_0)[s]$, we think of this computation as converging if each value y in the use is verifiably in $f_s(U)$ or verifiably not in $f_s(U)$. That is, if x is such that $f_s(x) = y$, then $\varphi_{e,s}(x)\downarrow \in \{0,1\}$.

We arrange the requirements according to the priority ordering $\mathbf{R}_0 > \mathbf{R}_1 > \cdots$, and proceed by a finite injury argument.

Construction:

> **Stage** s: For the highest priority active $\mathbf{R}_i$ that requires attention at stage s, perform the action indicated above and injure all lower priority requirements, de-activating them. Perform the upkeep as mentioned above, where new values are enumerated into A_s at the end of the tail associated to each marker. Finally, for the highest priority requirement that is not active, assign to it a fresh large $\omega \cdot 2$, beyond the use of any computation seen so far in the construction.

Verification:

Claim 3.6.1: Every requirement receives attention at most finitely often and is met.

Proof of Claim: It is clear from the construction that each requirement can only be injured by higher priority requirements, and each requirement can receive attention at most twice if it is never injured. Because of this, it is clear that each requirement receives attention at most finitely often. Indeed, $\mathbf{R}_i$ receives attention at most $2^{i+2} - 2$ times.

Consider the requirement $\mathbf{R}_{\langle e,j\rangle}$. By induction on the priority ordering, we may assume that there is a stage after which no requirement of higher priority than $\mathbf{R}_{\langle e,j\rangle}$ receives attention, and a least stage s after that where $\mathbf{R}_{\langle e,j\rangle}$ was (re-)initialized for the final time, and permanently associated with an $\omega \cdot 2$, say with limit points $\omega \cdot n$ and $\omega \cdot (n+1)$.

Suppose for a contradiction that $\varphi_e = U$ for some set U (i.e. φ_e is total and $\{0,1\}$-valued) and for all x we have $\Phi_j^{f(U)}(x)\downarrow = f(x)$.

First, $\mathbf{R}_{\langle e,j\rangle}$ cannot permanently be waiting for x_0. To see why, note that if no action takes place, then $f_t(x_0) = \omega \cdot (n+1)$ for all $t > s$, and so $f(x_0) = \omega \cdot (n+1)$. Since $\Phi_j^{f(U)}(x_0)\downarrow = f(x_0)$ by assumption, there must be some stage $s' > s$ where $\Phi_j^{f(U)}(x_0)[s']\downarrow$ $= \omega \cdot (n+1) = f_{s'}(x_0)$. But then at stage s', $\mathbf{R}_{\langle e,j\rangle}$ would require attention and receive it, since it is of highest priority. So at some stage s', $\mathbf{R}_{\langle e,j\rangle}$ must receive attention and wait for x_1.

Recall that while waiting for x_1, requirements can require attention for two reasons, which we refer to as condition (1) and condition (2). We will first show that if a requirement that is waiting for x_1 never requires attention via condition (2), then it must require attention via condition (1). Since $\mathbf{R}_{\langle e,j\rangle}$ will be of highest priority, if it requires attention for either of these reasons, it will receive it. We will therefore show that in either case, the action taken contradicts the supposition above, which will complete the proof of the claim.

So, suppose the requirement never requires attention via condition (2). By similar reasoning as in the previous case, we can find some stage $s'' > s' > s$ where $\Phi_j^{f(U)}(x_1)[s'']\downarrow$ $= \omega \cdot (n+1) = f_{s''}(x_1)$. So at no stage $t \geq s''$ can we have $\varphi_{e,t}(x_0)\downarrow \neq \varphi_{e,t}(x_1)\downarrow$. Since φ_e is total by assumption, there must be some stage $t' \geq s''$ where $\varphi_{e,t'}(x_0)\downarrow = \varphi_{e,t'}(x_1)\downarrow$. But this is exactly condition (1), so $\mathbf{R}_{\langle e,j\rangle}$ would require attention via condition (1) at stage t'.

Suppose first that $\mathbf{R}_{\langle e,j\rangle}$ receives attention via condition (1). Note that at stage s', it must have been the case that $\Phi_j^{f(U)}(x_0)[s']\downarrow = \omega \cdot (n+1)$. Since all higher priority requirements do not act after $s < s''$ and lower priority requirements were injured at stage s' and then later re-initialized beyond the use of this computation, we know that $f(U)[s'] = f(U)[s'']$ on the use of this computation, except for possibly at x_0 since $f_{s'}(x_0) = \omega \cdot (n+1)$ and $f_{s'+1}(x_0) = \omega \cdot n + m$ for some $m \in \omega$.

Hence, at stage t', we have $f_{t'}(x_1) = \omega\cdot(n+1)$ and $\varphi_{e,t'}(x_0)\downarrow = \varphi_{e,t'}(x_1)\downarrow$, so $x_0 \in f(U)[s']$ if and only if $x_1 \in f(U)[t']$, and so $\Phi_j^{f(U)}(x_0)[t']\downarrow = \omega\cdot(n+1)$. Now note that x_0 is never again marked after we stop waiting for x_0, so $f(x_0) = \omega\cdot n + m \neq \omega\cdot(n+1)$, and so $\Phi_j^{f(U)}(x_0) \neq f(x_0)$, a contradiction. Hence $\mathbf{R}_{\langle e,j\rangle}$ is met in this case.

Suppose second that $\mathbf{R}_{\langle e,j\rangle}$ receives attention via condition (2). Hence $\varphi_e(x_0)\downarrow \neq \varphi_e(x_1)\downarrow$, and so $x_0 \in U$ if and only if $x_1 \notin U$. Wait for some stage where $\varphi_e(x_2)$, which exists since φ_e is total. If $\varphi_e(x_2)\downarrow = \varphi_e(x_1)\downarrow$, then similar reasoning as before gives that $\Phi_j^{f(U)}(x_1)\downarrow = \omega\cdot(n+1) \neq \omega\cdot n + m' = f(x_1)$, where $m' \in \omega$. If $\varphi_e(x_2)\downarrow \neq \varphi_e(x_1)\downarrow$, then since φ_e is $\{0,1\}$-valued by assumption, it must be the case that $\varphi_e(x_2)\downarrow = \varphi_e(x_0)\downarrow$, and so $\Phi_j^{f(U)}(x_0)\downarrow = \omega\cdot(n+1) \neq \omega\cdot n + m = f(x_0)$. So again, we have a contradiction and $\mathbf{R}_{\langle e,j\rangle}$ is met.

Thus $\mathbf{R}_{\langle e,j\rangle}$ is met in all cases, as desired. ■

Note that in any given $\omega\cdot 2$, enumerations only take place twice when the associated requirement (if there is any) receives attention and also during the upkeep at the end of each stage. The latter clearly preserves the ordinal structure, since it occurs at the end of the two tails, and the former occurs only twice. The marker moves at most twice. Hence it is clear that each pair of marked values really does build an $\omega\cdot 2$ and so $\mathcal{A} \cong \omega^2$, since infinitely many $\omega\cdot 2$ will be created during the construction, as a new $\omega\cdot 2$ is created each time a requirement is newly initialized. □

3.4 Isomorphisms on copies of $(\omega^2, <)$ – Infinite Injury

This section is devoted to the proof of Theorem 3.7, which we restate here. Because of the similarity of the requirements to those in Theorem 3.6, we will mostly highlight the differences as we change to the infinite injury setting used for this result.

Theorem 3.7.
Let $\mathcal{N}^2$ be a decidable copy of $(\omega^2, <)$. There is a computable copy $\mathcal{A}$ of $(\omega^2, <)$ such that if $f : \mathcal{A} \to \mathcal{N}^2$, then for no computable unary relation U on $\mathcal{A}$ do we have $f(U) \equiv_T f$, and furthermore, $f \geq_T \varnothing''$.

Proof:
The broad strategy is to combine the $\mathbf{R}_i$ requirements from the proof of the weaker version of the theorem with the idea of the $\mathbf{\Gamma}_i$ requirements from the proof where we were working with $(\omega, <)$ (rather than $(\omega^2, <)$) and coding $\varnothing'$ into an isomorphism.

As with the weaker version of this theorem, we will be working with the $\omega \cdot 2$ blocks we get after partitioning ω^2 as such. To begin with, we describe the mechanism of action that a $\mathbf{\Gamma}$-type requirement takes to code whether $e \in \varnothing''$ in the absence of other requirements. This coding will take place in a single $\omega \cdot 2$ block – since we are ignoring other requirements for now – and so for sake of example, we will assume that we are working in the $[0, \omega \cdot 2)$ block. We code $e \in \mathrm{Inf} \equiv_T \varnothing''$ into this block. Recall that Inf is the set of indices of computable functions whose domains are infinite. That is, $\mathrm{Inf} \coloneqq \{e \mid |W_e| = \infty\}$.

The method of coding is as follows: when creating this block, we (as before) enumerate values that we mark, and then allow the upkeep step to build $\omega \cdot 2$ by the end of time. However, the action that we take to change the marked values is now more complicated than it was for the $\mathbf{R}$-type requirements. We instead enumerate three values $\ell < p < m$ and mark ℓ and m. The value ℓ is permanently marked and nothing will ever be marked below it, so we will have $f(\ell) = 0$. The value p is the *infinitary location* and m is the current *finitary location.* The value p will always be the infinitary location, but the finitary location may change from stage to stage, moving to $<^{\mathcal{A}}$-greater values. If $e \in \mathrm{Inf}$, then we want $f(p) = \omega$, while if $e \notin \mathrm{Inf}$, then we want $f(p) \neq \omega$ and $f(n) = \omega$ where n is some value that permanently becomes the finitary location at some stage. Thus, knowing f and the infinitary location p will allow us to decide if e is in Inf or not.

To arrange for these conditions to occur, we take the following action at each stage s. Looking forward, we shall also indicate what is to be done if this requirement has been initialized but is not able to receive attention. If W_e has grown since the last stage where $\mathbf{\Gamma}$ received attention and $\mathbf{\Gamma}$ is permitted to receive attention at this stage, then we unmark the current finitary location and mark the infinitary location. Otherwise, if the infinitary location is marked, designate the right-most value in this $\omega \cdot 2$ – that is, the value at the end of the tail – as the the current finitary location and mark it, unmarking the infinitary location. At the end of the stage, perform upkeep on this $\omega \cdot 2$ in all cases.

Now, if W_e is infinite, then there will be infinitely many stages where we marked p and perform upkeep, so p will have an infinitely long tail. Also, each value that is ever a finitary location only stays as such for finitely many stages, and hence is not marked for infinitely many stages, so they cannot correspond to a limit point. Then $f(p) = \omega$ when $e \in \mathrm{Inf}$. On the other hand, if W_e is finite, then there is some finite stage where p is marked for the last time, and the next stage a finitary location n is chosen and marked, and p is permanently unmarked. Hence p is absorbed into the tail of ℓ and n

is marked at infinitely many stages, so n is a limit point. Thus $f(n) = \omega$ and $f(p) \neq \omega$ when $e \notin \mathrm{Inf}$.

In fact, this strategy gives more, which we will make use of later. Suppose we have finished building the computable copy $\mathcal{A}$, and we are given f. Again, for the sake of example, we will assume that $\mathbf{\Gamma}$ is coding in the block $[0, \omega \cdot 2)$, but later on we will need to determine which block the coding is taking place in, which we ignore for now. Use f to determine $f^{-1}(\omega)$. If $f^{-1}(\omega) = p$, then $e \in \mathrm{Inf}$, and if $f^{-1}(\omega) \neq p$, then $e \notin \mathrm{Inf}$, as we have already determined. If $f^{-1}(\omega) = n \neq p$ for some n, then in this case we can run the construction until a stage where n is designated as the finitary location. We then know that after this stage, $\mathbf{\Gamma}$ never reverts to marking the infinitary location p, since we cannot abandon a finitary location and then come back to it, thanks to the upkeep step. This additional information will prove useful, since the marking of an infinitary location will cause injury. So using f, we can not only decide if e is in Inf or not, but also at what stage $\mathbf{\Gamma}$ stops causing injury (if such a stage exists).

Notice however that this method of action for $\mathbf{\Gamma}$-type requirements causes problems for $\mathbf{R}$-type requirements of lower priority. Since $\mathbf{R}$-type requirements must diagonalize against a computation whose use depends crucially on f, $\mathbf{R}$-type requirements should be injured whenever a $\mathbf{\Gamma}$-type requirement changes between the infinitary case and the finitary case, as this changes between $f(p) = \omega$ and $f(n) = \omega$, respectively. But this type of action may occur infinitely often if W_e is infinite, and so lower priority $\mathbf{R}$-type requirements would be re-initialized forever and therefore fail to be met.

Because of this, we will use an infinite injury priority strategy on a tree, with requirements guessing about the outcome of higher priority $\mathbf{\Gamma}$-type requirements. As per usual, the infinitary guess is denoted 0 and the finitary guess is denoted 1 with 0 to the left of 1.

Again, as usual, we define the notation of a *σ-stage* recursively for all $\sigma \in 2^{<\omega}$ as follows:

- Every stage is a $\varnothing$-stage.
- If s is a σ-stage and if $\left|W_{|\sigma|,s}\right| > \left|W_{|\sigma|,t}\right|$ where $t < s$ is the previous σ-stage (where $W_{|\sigma|,-1} \coloneqq \varnothing$ and -1 is considered to be a σ-stage), then s is a $\sigma^\frown 0$ stage, and a $\sigma^\frown 1$ stage if not.

For each $\sigma \in 2^{<\omega}$, we have requirements $\mathbf{R}_\sigma$ and $\mathbf{\Gamma}_\sigma$ attempting to meet $\mathbf{R}_{|\sigma|}$ and $\mathbf{\Gamma}_{|\sigma|}$ respectively. Each requirement will take actions in a particular $\omega \cdot 2$ block, but may also injure other requirements and cause them to be reset and re-initialized in a fresh large block. When this occurs, the requirement that causes the injury can, in some cases, seize control of the abandoned block and change which elements are marked so as to change which elements become limit points at the end of the construction.

In light of now needing infinite injury, let us re-examine an $\mathbf{R}$-type requirement and describe its action. As in the proof of the weaker theorem, $\mathbf{R}_{\langle e,i\rangle}$ is attempting to find a computation to diagonalize against, so that if $\varphi_e = U$ for some set U, then $\Phi_i^{f(U)} \neq f$.

Let us recall the action of this requirement in the weaker theorem. We had the $\mathbf{R}$-type requirements respect requirements of higher priority and wait to see a computation that it desired to diagonalize against whose use is some initial segment of $f(U)[s]$ that was *verifiable.* We said that $f(x)$ was *verifiably* in (or not in) $f(U)[s]$ if $\varphi_{e,s}(x)\downarrow \in \{0,1\}$. If this occurred, the requirement would injure lower priority requirements, causing them to abandon their blocks and allow $\mathbf{R}_{\langle e,i\rangle}$ to preserve the use of the computation as long it was never injured by a higher priority requirement. Requirement $\mathbf{R}_{\langle e,i\rangle}$ would take this action at most twice, provided it was not later injured.

However, we must now worry about requirements not on the true path acting infinitely often and spoiling preserved computations. We must also worry about computations that come into existence at the end of the construction, but were never seen at any finite stage during the construction, since $\mathbf{R}$-type requirements would not be able to diagonalize against these computations.

To fix this, we allow $\mathbf{R}_\sigma$ to simulate computations that do not exist at the current stage, but which could exist if σ is on the true path. If such a computation is found, then when $\mathbf{R}_\sigma$ injures lower priority requirements, it may need to seize control of their abandoned blocks in order to change which values are marked. It can then arrange for this simulated computation to actually become a reality, so the computation actually exists at a finite stage.

More formally, the action of $\mathbf{R}_\sigma$ at the σ-stage s is as follows: assume that $\mathbf{R}_\sigma$ is currently in the $\omega\cdot 2$ block $[\omega\cdot n, \omega\cdot(n+2))$ with ℓ permanently marked, so $f(\ell) = \omega\cdot n$. When $\mathbf{R}_\sigma$ is newly initialized in this block, it marks x_0 as the value corresponding to $\omega\cdot(n+1)$. At stage s, if x_0 is marked, search for a possible computation to diagonalize against. This process is not easy to formally describe, so the below steps are quite verbose, although the actual idea is not particularly complicated.

First, find the longest initial segment of $\mathcal{A}_s$ that is verifiably in or out of U by computing $\varphi_{e,s}(x)\downarrow \in \{0,1\}$ for each x that has been enumerated into the domain of $\mathcal{A}$. Next, determine all possible limit point outcomes in $\mathcal{A}$ as of stage s if σ is on the true path. To do this, assume that each $\mathbf{\Gamma}$-type requirements of higher priority than $\mathbf{R}_\sigma$ acts according to the guess encoded in σ. For example, if $\sigma > 0$, then $\mathbf{\Gamma}_\varnothing$ is assumed to have the value in its infinitary location correspond to the second limit point for its block, while if $\sigma > 1$, then $\mathbf{\Gamma}_\varnothing$ is assumed to have the value in its current finitary location correspond to the this limit point. Also assume that all $\mathbf{R}$-type requirements will never act after this stage, and so their current marked values correspond to their associated limit point. Finally, for all lower priority $\mathbf{\Gamma}$-type requirements, include *both* the infinitary location and the current finitary location as possible corresponding values for that requirement's block's second limit point.

Second, once this list of possible list of limit point outcomes has been built, generate the associated list of partial isomorphisms, with each partial isomorphism corresponding to a certain limit point outcome.

Third, search for a computation to diagonalize against using all these partial isomorphisms. Rather than simply waiting for a computation of the form $\Phi_i^{f(U)}(x_0)[s]\downarrow = \omega \cdot (n+1)$, we also include all computations $\Phi_i^{g(U)}(x_0)[s]\downarrow = \omega \cdot (n+1)$, where g ranges over this list of partial isomorphisms that we have just built for $\mathbf{R}_\sigma$ at stage s.

If such a computation exists, take action by injuring lower priority requirements and seizing control of their blocks, and (un)mark values in these blocks to create the limit point outcome of the particular g that was used. Enumerate x_1 into $\mathcal{A}$ and mark it as before. The rest of the action is very similar to the proof of the weaker theorem, where x_2 is possibly enumerated at some later stage, except again when searching for a computation to diagonalize against, we use this method of simulating desirable computations rather than passively waiting.

Construction:

> **Stage** s: Determine the σ of length s such that s is a σ-stage. For each initialized $\mathbf{\Gamma}$-type requirement, take the prescribed action for that requirement. Note that only $\mathbf{\Gamma}_\tau$ with $\tau \leq \sigma$ are permitted to receive attention. If $\mathbf{\Gamma}_\tau$ receives attention and marks its infinitary location, then all requirements with priority to the right of $\tau^\frown 0$ are injured and abandon their blocks. Now find the highest priority $\mathbf{R}_\tau$ with $\tau \leq \sigma$ that requires attention. If there is one, it receives attention, injuring requirements of lower priority and seizing control of their blocks (and any unowned blocks that are needed). Finally, initialize the highest priority requirement that is not initialized (assigning it a fresh large block) and perform upkeep on all blocks.

Verification:

Let TP be the true path, that is, the left-most path visited infinitely often.

Claim 3.7.1: $TP(i) = 0 \Leftrightarrow i \in \text{Inf}$.

<u>Proof of Claim:</u> Suppose $TP(i) = 0$. Then for infinitely many stages s, we must have that s is a $(TP \upharpoonright i)^\frown 0$-stage. Then $|W_{i,s}| > |W_{i,t}|$, where t is the previous $(TP \upharpoonright i)$-stage. Hence $|W_i|$ must be infinite, and so $i \in \mathrm{Inf}$.

Conversely, suppose $TP(i) = 1$. Then $(TP \upharpoonright i)^\frown 0$ is only visited finitely often (since any extension of it is to the left of TP), so there must be some stage $\hat{s}$ after which nothing to the left of $(TP \upharpoonright i)^\frown 1 = (TP \upharpoonright\upharpoonright (i+1))$ is ever visited. So $|W_{i,s}| = |W_{i,t}|$ for all $(TP \upharpoonright i)$-stages s with prior such stage t which both exceed $\hat{s}$, and so $|W_{i,s}| = |W_{i,s+1}|$ for all stages $s > \hat{s}$. Thus $|W_i| = |W_{i,\hat{s}}| < \infty$ and therefore $i \notin \mathrm{Inf}$. ■

<u>Claim 3.7.2:</u> For each n, consider $\sigma := (TP \upharpoonright n)$, the initial segment of the true path of length n. Then the requirement $\mathbf{R}_\sigma$ receives attention at most finitely often.

<u>Proof of Claim:</u> We proceed inductively on n. So suppose that there is some stage s by which all $\mathbf{R}_\tau$ with $\tau \prec \sigma$ have finished receiving attention. Additionally, since σ is on the true path, we may assume s is large enough so that no path to the left of σ is visited after stage s. Thus, after stage s, no requirement $\mathbf{R}_\tau$ or $\boldsymbol{\Gamma}_\tau$ receives attention where τ is to the left of σ.

We claim that $\mathbf{R}_\sigma$ cannot be injured after stage s. If $\mathbf{R}_\tau$ were to injure $\mathbf{R}_\sigma$, then σ must have lower priority than τ. But after stage s, we know that for $\mathbf{R}_\tau$ to receive attention, τ cannot be to the left of σ and it cannot be an initial segment of σ. Hence it is to the right of σ or extends σ, neither of which are of higher priority than σ.

Since $\mathbf{R}_\sigma$ is never injured after stage s, it is permanently assigned to an $\omega \cdot 2$ block at or after stage s. As in the case of the weaker theorem, it can thereafter receive attention at most twice: once to mark a new value x_1 and once more to mark a new value x_2. Hence $\mathbf{R}_\sigma$ can receive attention at most finitely often, as desired. ■

<u>Claim 3.7.3:</u> For each n, consider $\sigma := (TP \upharpoonright n)$ as before. Then the requirement $\boldsymbol{\Gamma}_\sigma$ eventually is permanently assigned an $\omega \cdot 2$ block. Moreover, in this block, if $|\sigma| \in \mathrm{Inf}$, then the limit points are the permanently marked location ℓ and the infinitary location, and if $|\sigma| \notin \mathrm{Inf}$, then the limit points are the permanently marked location ℓ and a finitary location.

<u>Proof of Claim:</u> As in the previous claim, there must eventually be some stage after which no higher priority requirement receives attention and hence eventually $\boldsymbol{\Gamma}_\sigma$ is (re)-initialized and assigned an $\omega \cdot 2$ block and then never after injured.

Since σ is on the true path, there are infinitely many σ-stages, and hence infinitely many stages to see that $\left|W_{|\sigma|}\right|$ has grown. So if $|\sigma|$ is in Inf, then there are infinitely many stages where the infinitary location will be marked. Conversely, if $|\sigma|$ is not in Inf, then there are only finitely many stages where the infinitary location will be marked, and hence at some stage a finitary location will be permanently marked. The exact details are essentially as we informally outlined when introducing the $\mathbf{\Gamma}$-type requirements above, so they will not be repeated. ■

Claim 3.7.4: For each i and e, the requirement $\mathbf{R}_{\langle e,i\rangle}$ is eventually met. That is, if $\varphi_e = U$ for a computable set u, there is some x such that $\Phi_i^{f(U)}(x) \neq f(x)$.

Proof of Claim: Fix i and e and suppose otherwise. That is, for all x we must have $\Phi_i^{f(U)}(x) = f(x)$. Consider $\sigma \coloneqq TP \upharpoonright (\langle e,i\rangle)$. By Claim 3.7.1, there must be some stage s after which no node to the left of σ on the tree is ever visited. Moreover, since $\mathbf{R}$-type requirements along the true path only receive attention at most finitely often, we may take s large enough so that after stage s, $\mathbf{R}_\sigma$ will never after be injured and if it requires attention, then it will receive it. Suppose the $\omega \cdot 2$ block that $\mathbf{R}_\sigma$ is assigned to by or after this stage is $[\omega \cdot n, \omega \cdot (n+2))$. Without loss of generality, we may assume $\mathbf{R}_\sigma$ is (re-)initialized in this block at stage s. Finally, take s large enough so that if $\sigma(j) = 1$ (i.e. $j \notin \mathrm{Inf}$), then $W_{j,s} = W_j$.

First, $\mathbf{R}_\sigma$ cannot be permanently waiting for x_0. If no action by $\mathbf{R}_\sigma$ ever takes place after stage s, then $f_t(x_0) = \omega\cdot(n+1)$ for all $t > s$, and so $f(x_0) = \omega\cdot(n+1)$. (Requirements that are assigned blocks to the left of $\mathbf{R}_\sigma$'s block cannot change the number of limit points in these blocks, which is why $f_t(x_0)$ and $f(x_0)$ can be explicitly computed as $\omega \cdot (n+1)$.)

Since $\Phi_j^{f(U)}(x_0)\downarrow = f(x_0)$ by assumption, we must be able to find a σ-stage $s' > s$ where $\Phi_{j,s'}^{f(U)}(x_0)\downarrow = \omega \cdot (n+1) = f_{s'}(x_0)$. Unlike in the proof of the weaker theorem, we cannot guarantee that $\Phi_{j,s'}^{f(U)}(x_0)\downarrow = \Phi_j^{f(U)}(x_0)[s']\downarrow$ for some large enough s', since higher priority $\mathbf{\Gamma}$-type requirements may continue to act infinitely often.

Consider the set of blocks used in this computation. Some of them are owned by $\mathbf{R}$-type requirements. If the $\mathbf{R}$-type requirement is of higher priority, then by choice of s, it cannot ever receive attention since this would injure $\mathbf{R}_\sigma$. If it is of lower priority, then $\mathbf{R}_\sigma$ could seize control of that block should it need to. Additionally, some of the blocks are owned by $\mathbf{\Gamma}$-type requirements. At any σ-stage $s' > s$, we must have that the higher priority $\mathbf{\Gamma}$-type requirements act according to the guess σ, and so at this stage s', they will mark the infinitary or finitary location in their block as appropriate. In particular, since by assumption s is large enough so that $\sigma(j) = 1$ implies that $W_{j,s} = W_j$, if one of these $\mathbf{\Gamma}$-type requirements would eventually mark a permanent finitary location, it has already done so by stage s and then never changes this. That is, these higher-priority $\mathbf{\Gamma}$-type requirements correctly mark the limit points in their blocks at any σ-stage after stage s. The lower priority $\mathbf{\Gamma}$-type requirements mark either their infinitary location or a finitary location. Take s' large enough so that if a finitary location is marked by one of these requirements in the limit, then it has actually been chosen as the (permanent) finitary location by stage s'. It may not be marked at stage s', but since we simulate computations for $\mathbf{R}_\sigma$ to see if it should act, we will simulate computations where either the infinitary location and the finitary location are eventually marked.

Finally, take s' large enough so that enough of the tails of every eventual limit point has already been correctly enumerated into $\mathcal{A}$ by stage s'. Hence, the computation $\Phi_{j,s'}^{f(U)}(x_0)\downarrow$ would actually be simulated at the σ-stage s' and so $\mathbf{R}_\sigma$ should act and diagonalize against it by enumerating x_1 into this block and marking it.

The remainder of the proof is similar to the proof of the same claim in the weaker theorem. However, one important note is that $\mathbf{R}_\sigma$ takes control of any blocks owned by lower priority requirements should it see a computation that it needs to diagonalize against. It does this so that it can make the desired limit point outcome definitely occur for that diagonalization. Otherwise, if it was now waiting for x_1, then it needs to guarantee that the computation it diagonalized against for x_0 cannot go away. This allows us to make the same claim that $f(U)[s'] = f(U)[s'']$ (except possibly at x_0) for a future stage of interest s'', which we could not do if we diagonalized against a possible future computation but never actually made this computation occur. ■

Claim 3.7.5: Given f, and the eventual final resting location of some $\mathbf{R}_\sigma$ or $\mathbf{\Gamma}_\sigma$ where $\sigma \prec TP$, then we can determine if that requirement receives attention finitely often or infinitely often in this location, and if the former, find a stage s after which that requirement never again receives attention, if such a stage exists.

Proof of Claim: Suppose we have been told that the requirement has final resting position in the $\omega \cdot 2$ block $[\omega \cdot n, \omega \cdot (n+2))$. Run the construction until the requirement is (re-)initialized in this block, say at stage t.

First suppose the requirement is an $\mathbf{R}$-type requirement. Then by Claim 3.7.2, we know that $\mathbf{R}_\sigma$ receives attention at most finitely often. To find a stage s after which $\mathbf{R}_\sigma$ never receives attention, compute $f^{-1}(\omega \cdot (n+1))$, and wait for a stage s where $f_s^{-1}(\omega \cdot (n+1)) = f^{-1}(\omega \cdot (n+1))$. Such a stage must exist since the action of $\mathbf{R}_\sigma$ permits it to only change this value at most twice, and it never returns to a previous value.

Now suppose the requirement is an $\mathbf{\Gamma}$-type requirement. Compute $f^{-1}(\omega \cdot (n+1))$. If this is the infinitary location that was chosen when $\mathbf{\Gamma}_\sigma$ was (re-)initialized in this block, then $\mathbf{\Gamma}_\sigma$ receives attention infinitely often, and hence no such stage s exists. On the other hand, if it is not the infinitary location, then it must be some finitary location. Run the construction until a stage s where it is chosen as the finitary location. Then after stage s, $\mathbf{\Gamma}_\sigma$ cannot receive attention, since this location would be forever abandoned as a marked location and hence could not be $f^{-1}(\omega \cdot (n+1))$. ∎

Claim 3.7.6: Given f, for each n let $\sigma_n \coloneqq TP \upharpoonright n$. Then we can compute the final resting position of $\mathbf{R}_{\sigma_n}$ and find a stage s after which it never receives attention, and compute the final resting position of $\mathbf{\Gamma}_{\sigma_n}$ and determine if there is a stage t after which it never receives attention and find this t if so. Additionally, we can determine $TP(n)$ and hence compute $TP \upharpoonright\!\upharpoonright n$.

Then since $TP(i) = 0 \Leftrightarrow i \in \mathrm{Inf}$ by Claim 3.7.1, this allows us to compute Inf from f, and thus $f \geq_T \mathrm{Inf} \equiv_T \varnothing''$.

Proof of Claim: We proceed by induction on n, so suppose the claim holds for $m < n$. Then we can compute $\sigma_n \coloneqq TP \upharpoonright n$ and also find stages after which all requirements $\mathbf{R}_\sigma$ and $\mathbf{\Gamma}_\sigma$ for $\sigma \preceq \sigma_n$ stop receiving attention if such a stage exists for that requirement. Take the maximum M of all these stages and run the construction up to stage M. (Take $M = 0$ if no requirements have such a stage.)

From stage M, run the construction until $\mathbf{R}_{\sigma_n}$ is initialized (if it is not already initialized at stage M). Then the block that $\mathbf{R}_{\sigma_n}$ currently in is its final resting position, since no requirement of higher priority can receive attention after stage M and cause injury, since if requirements to the left of the true path received attention, then when the true path was next visited, a $\mathbf{\Gamma}$-type requirement that only receives attention would do so after stage M, contradicting the choice of M.

By the previous claim, we can then use f to find a stage s after which $\mathbf{R}_{\sigma_n}$ never receives attention. Run the construction to this stage and then run the construction until $\mathbf{\Gamma}_{\sigma_n}$ is initialized (if it is not already initialized at stage s). Then similarly to the previous, we know that the block that $\mathbf{\Gamma}_{\sigma_n}$ is currently in is its final resting position. By the previous claim, we can determine if this requirement receives attention infinitely or finitely often. If the latter, by the claim we can also determine a stage t after which it never receives attention.

Now, if $\mathbf{\Gamma}_{\sigma_n}$ receives attention infinitely often, then $TP(n) = 0$. Otherwise, $TP(n) = 1$, and so we can determine $TP \upharpoonright\!\upharpoonright n$.

This completes the induction, and hence $f \geq_T TP \equiv_T \mathrm{Inf} \equiv_T \varnothing''$. ∎ □

3.5 Future Work

In light of the above proof, a natural question is:

Question 3.8.
Can Theorem 3.7 be improved so that we code $\varnothing'''$ into f instead of just $\varnothing''$?

Of course, since we are working with the ordinal $(\omega, <)$ and $(\omega^2, <)$, a broader question is

Question 3.9.
For any $k \geq 1$, it is possible to produce a computable copy $\mathcal{A}$ of ω^k such that if $f : \mathcal{A} \to \mathcal{N}^k$, then for no computable unary relation U on $\mathcal{A}$ do we have $f(U) \equiv_T f$, and furthermore, $f \equiv_T \varnothing^{(2k-1)}$?

A potential starting point for such a proof is to use the metatheorem about η-systems developed by Montalbán [22], which is inspired by Ash's similar metatheorem as in [4]. Montalbán's version of the metatheorem was later improved by Csima and Harrison-Trainor [13] to make it more sensible for limit ordinals. If such a proof could be developed, one might wonder about extracting the "intermediate" structure (if any) that has degree of categoricity $\mathbf{0}^{(2k)}$ for which the metatheorem is able to prove a similar result.

Even more generally, one might wonder if there are any natural examples of this phenomenon of the ordinals, and if so, is there a characterization of structures where such a result holds? Further, even in the examples we give, the copies we construct are, in some sense pathological, since there exist copies of $(\omega, <)$ and $(\omega^2, <)$ where the isomorphisms are relatively benign in this regard, so one may wish to know the following:

Question 3.10.
Fix $\mathcal{A}$, a rigid and computable structure with strong degree of categoricity $\mathbf{d}$. Under what conditions are there computable copies $\mathcal{B}, \mathcal{C}$ of $\mathcal{A}$ with isomorphism $f : \mathcal{B} \to \mathcal{C}$ and a computable U such that $f(U)$ is of Turing degree $\mathbf{d}$? What about when $\mathcal{A}$ need not be rigid?

Additionally, the results we have proven above are weak in the sense that $f(U)$ is unable to compute f, but could potentially be *any* degree below f. Is it possible to construct two copies of some rigid structure where $f(U) \equiv_T \varnothing$ for all computable unary relations U, where f is the isomorphism between copies?

Chapter 4

Direct Sums of Abelian Groups

This section was co-supervised by Matthew Harrison-Trainor, based on a question that he posed after discussion with Noah Schweber. Jason Bell helpfully pointed us toward the relevant references in the literature.

4.1 Background

Suppose that A is a finitely generated abelian group, and G and H are abelian groups such that $A \oplus G \cong A \oplus H$. Must it be the case that G and H are isomorphic? Cohn [8] and Walker [29] independently proved this result, now know as Walker's Cancellation Theorem, using essentially the same techniques. We aim to determine to what extent can such an isomorphism be constructed computably given knowledge of the constituent groups. We mainly follow Cohn's notation and refer to his method of proof.

We note first the following simplification: Rather than considering two separate but isomorphic groups $A \oplus G$ and $A \oplus H$, we instead let E be any isomorphic copy and identify A and G with their images as subgroups of E under such an isomorphism. For instance, if $\varphi : A \oplus G \to E$ is such an isomorphism, then we can think of A as $\varphi(A) \subseteq E$, and similarly think of G as $\varphi(G) \subseteq E$. In the same way, we can identify the $A \oplus H$ inside E. However, since the identification of the copy of A from $A \oplus G$ and the copy of A from $A \oplus H$ may not co-incide inside E, we call the latter B. So, we now have a group E which can be written as the (internal) direct sum $A \oplus G$ and also as $B \oplus H$, with $A \cong B$. By making these identifications, we can see how the two representations interact with each other directly rather than passing through isomorphisms every time we wish to do so.

First, we begin by examining to what extent the theorem is true in a computable setting and what portion of the question is ultimately of interest to us. We shall see that if we are only interested in a particular E, then the theorem is always true computably. However, the natural question of uniformity arises and it is in this setting that we find important limitations in Cohn and Walker's work. When we speak of "uniformity" here, we mean that, given an index for E and computable relations that decide A, B, G and H, and an index for the isomorphism between A and B, then can we uniformly compute an index for an isomorphism between G and H? We will show that if we are attempting to work in this way, then the theorem cannot be true, even if we also give the procedure access to a large amount of information about the groups and subgroups.

4.2 Computably Splitting Direct Sums

We begin with the most amount of information about A and B, namely what groups they are and what their generators are. That is, since A and B are finitely generated abelian groups, we know that they must be isomorphic to some $F_1 \oplus F_2 \oplus \cdots \oplus F_k$ where each F_i is cyclic of either prime power or infinite order. We suppose that this expansion is known ahead of time, both the order of each F_i and the corresponding $a_i \in A$ and $b_i \in B$ such that the copy of F_i in A is generated by a_i (and similarly b_i in B). We say that we *know the groups* and *know the generators*, respectively.

First we make a small remark about uniformity, which is ultimately what concerns us. By examining the proof of Cohn [8] closely, we see that determining a computable isomorphism between G and H must always be possible, provided (finite) additional information can be hardcoded into the program that produces such an isomorphism. In fact, if we know both the groups and generators for $A \cong B$ and A and B are known to be finite, then we can discover this information as we proceed in the construction of the isomorphism. That is:

Theorem 4.1 (Based on Cohn [8])**.**
Suppose we know both the groups and generators of $A \cong B$ and $A \oplus G = B \oplus H$. Then G and H are computably isomorphic, and moreover, if A and B are known ahead of time to be finite, then such an isomorphism can be constructed uniformly in the indices for A, B, G, H and the isomorphism between A and B, as mentioned above.

Proof:
We will essentially follow Cohn's proof, commenting on what can be performed computably and what needs to be hard-coded.

We proceed by induction on k, the number of cyclic summands F_i that comprise A and B. Since we know the groups and generators, we may assume that the representation

$F_1 \oplus F_2 \oplus \cdots \oplus F_k$ is in some canonical order. We will show that we can solve the problem for A and B cyclic, with known order and known generators, which we call a and b, respectively. The details of the induction step are easy and hence omitted.

Suppose A and B are infinite. Let $D \coloneqq G \cap H$, then $G/D \cong G/(G \cap H) \cong (G+H)/H$. Since $G+H$ is a subgroup of E, G/D is thus isomorphic to a subgroup of E/H, and hence isomorphic to a subgroup of B. As B is infinite and cyclic, G/D is therefore either infinite cyclic or trivial. A similar argument shows that H/D is either infinite cyclic or trivial. Suppose G/D is infinite and choose $u \in G$ so that $u + D \in G/D$ is a generator for G/D. Let $U \subseteq G$ be the subgroup generated by u. Then it is easily seen that $G = U \oplus D$. Then $E = A \oplus U \oplus D = B \oplus H$. We have that D is a subset of H, so taking a quotient by D results in $A \oplus U = B \oplus (H/D)$. Hence H/D must also be infinite. The converse is identical, so G/D and H/D must be either both infinite or both trivial. In either case, let $u + D$ be a hardcoded generator for G/D and $v + D$ be a hardcoded generator for H/D. Let $U \coloneqq \langle u \rangle$ and $V \coloneqq \langle v \rangle$. Then $G \cong U \oplus D \cong V \oplus D \cong H$. This isomorphism is computable: given an element $g \in G$, search for the unique representation of g as $ku+d$, where $k \in \omega$ and $d \in G \cap H$. To do this, compute $g - ku$ for more and more $k \in \omega$ until $g - ku \in G \cap H$ is found. Then map g to the element $h \coloneqq kv + d$.

Now suppose A and B are finite cyclic, say of order p^n for some prime p. First, there is an element $u \in E$ such that no multiple of u is in $G \cup H$ unless it is 0, and u has order p^n. In fact, one of a, b or $a + b$ always works, as can be easily shown. For each of these three candidate values x, we compute $x, 2x, \ldots (p^n - 1)x$ and ensure that if it is not 0, then it is not in $G \cup H$. There are only finitely many values to compute, and hence u can be found. Let U be the subgroup generated by u. Then $U \cap G$ must be $\{0\}$. Also, $(U + G)/G \cong U/(U \cap G)$ by the isomorphism theorem, and hence $(U + G)/G \cong U$. So $[U + G : G] = \mathrm{ord}(u) = p^n$. But $[E : G] = \mathrm{ord}(a) = p^n$ and $U + G \subseteq E$, so we must have $U + G = E$, and hence $E = U \oplus G$ since $U \cap G = \{0\}$. Similarly, $E = U \oplus H$. Thus any element of E can be written uniquely as $su + g$ for some $0 \le s < p^n$ and $g \in G$ and also as $tu + h$ for some $0 \le t < p^n$ and $h \in H$, and so $G \cong E/U \cong H$. To build the isomorphism between G and H, given g we search for and eventually find such the unique t and h such that $g = tu + h$, and map g to h. □

Note that in the finite case, we did not need to have any hardcoded information about the groups; if our procedure is given the order p^n of A and B and their generators a and b as we pre-supposed, then it can construct an isomorphism between G and H using only indices for the subgroups and the isomorphism between A and B. On the other hand, however, in the infinite case, Cohn's proof demands the generator of an infinite cyclic subgroup, and this cannot be determined from the given information. Such a generator needs to be supplied from outside the procedure.

Of course, perhaps we are simply not being clever enough. Cohn's proof is not designed to take computability-theoretic questions into mind. Could a different proof avoid the need for non-uniform information in the infinite case? The answer is no, as the next result shows: we can defeat uniformity in the infinite case, even if we know both the groups and generators. We shall construct an example of such groups, so that A and B are ultimately isomorphic to $\mathbb{Z}$ and G and H are isomorphic to $\mathbb{Z} \times \mathbb{Z}$. A key leverage point of the proof is that we do not need to specify in advance any information about the generators of G or H.

Theorem 4.2.

There is no partial computable function F which, given the index for (the presentation of) the computable group $E \coloneqq A \oplus G = B \oplus H$ and indices for computable relations determining A, B, G and H and the generators of A and B, respectively a and b, outputs an index for a computable isomorphism between G and H.

Proof:

For each partial computable function F, we construct a counterexample such that A and B are isomorphic to $\mathbb{Z}$ and G and H are isomorphic to $\mathbb{Z} \times \mathbb{Z}$. We use the Recursion Theorem (see Soare [28]) to be given the indices for our counterexample. In exchange, we must ensure that if F fails to halt at any stage, we nevertheless build groups of the required form. We shall explicitly note where we wait for F to halt and show that no problems arise should F fail to halt. The same holds true for the function that F may produce: if it fails to halt on some value in G, then we win provided we ultimately end up building the desired type of groups in this case.

We must specify the generators of A and B in advance so that the uniform procedure may have access to them. We arrange the following conventions: we view elements of $E \coloneqq A \oplus G = B \oplus H$ as elements of $\mathbb{Z} \times \mathbb{Q} \times \mathbb{Q}$ – although E will not be this full group, but rather a subgroup of it. We define A to be the subgroup generated by $a \coloneqq (1, 0, 0)$. Importantly to the proof, we do not define the generators of G ahead of time. Instead, we declare the elements $g_1 \coloneqq (0, 1, 0)$ and $g_2 \coloneqq (0, 0, 1)$ to be elements of G. Note that g_1 and g_2 will never be multiples of each other. If we do not explicitly declare additional elements to be in G, then it is not hard to see that $G \cong \mathbb{Z} \times \mathbb{Z}$, with g_1 and g_2 as generators. However, we could, for example, declare that $(0, 1/2, 0)$ is also in G, in which case g_1 is clearly not a generator. In this way, we may reveal new information that changes which elements of G appear to be generators, and it is this technique that allows us to defeat the uniform procedure F.

We have not yet supplied the generator of B. We define B to be generated by $b \coloneqq (-2, -1, -2) = -2a - g_1 - 2g_2$. As with G, we define $h_1 \coloneqq (5, 2, 5) = 5a + 2g_1 + 5g_2$ and $h_2 \coloneqq (0, 0, 1) = g_2$, which we declare as elements of H.

Note that if we never reveal new generators of G or H, then $A \oplus G$ will be generated by $\{a, g_1, g_2\}$ and $B \oplus H$ will be generated by $\{b, h_1, h_2\}$, and these two groups are equal to each other (and to $\mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}$) since $a = 2b + h_1 - h_2$, $g_1 = -5b - 2h_1$ and $g_2 = h_2$.

Thus the uniform procedure F we are attempting to defeat must provide an isomorphism between G and H, say $f : G \to H$. If it never does, then we do not enumerate new elements into our group that are not sums of existing elements. In this way, as noted above, $E = A \oplus G = B \oplus H$, with A and B generated by a and b, respectively, and G generated by $\{g_1, g_2\}$ and H generated by $\{h_1, h_2\}$ and both G and H are isomorphic to $\mathbb{Z} \times \mathbb{Z}$, since we will never introduce torsion.

So now suppose that F has halted and given us the index for a function, which it purports to be an isomorphism $f : G \to H$. We first claim that $f(g_2)$ must be $\pm h_2$. Wait for $f(g_2)$ to be declared. Again, if $f(g_2)$ never halts, then we continue building the groups as before and win, since we have not taken any action that would disturb this yet.

We know that f must send g_2 to $mh_1 + nh_2$ for some $m, n \in \mathbb{Z}$. Suppose first that $m \neq 0$. Take N to be larger than $|m|$, and reveal a new element of G and H, $\hat{g} = (0, 0, 1/N)$ and note that $N\hat{g} = g_2 = h_2$. Now $A \oplus G$ is generated by $\{a, g_1, \hat{g}\}$, and $B \oplus H$ is generated by $\{b, h_1, \hat{g}\}$, and the relationship between elements in $A \oplus G$ and in $B \oplus H$ remain the same. However, $g_2 = N\hat{g}$, so there must be some element $\hat{h}$ of H such that $N\hat{h} = f(g_2)$ or else the potential isomorphism f fails. The generators for H are now h_1 and $\hat{g}$, so $\hat{h}$ must be some linear combination of h_1 and $\hat{g}$, say $m'h_1 + n'\hat{g}$. Then $mh_1 + nh_2 = f(g_2) = N\hat{h} = Nm'h_1 + Nn'\hat{g}$. As $N\hat{g} = h_2$, this gives $m = Nm'$ and $n = n'$. But $m \neq 0$ and N is larger than $|m|$, so this is impossible and we defeat f, no matter what it is. Hence $m = 0$.

If $m = 0$, then we have $nh_1 = f(g_2)$ and so h_1 non-trivially divides $f(g_2)$ unless $n = \pm 1$, as we have claimed. Since g_2 appears at this moment to be a generator of G, there are no elements in $A \oplus G$ that non-trivially divide g_2 and so there cannot be any elements in $B \oplus H$ that non-trivially divide $f(g_2)$.

Hence if f is truly a potential isomorphism between G and H, it must declare that $f(g_2) = \pm h_2$. Since $\{g_1, g_2\}$ generates G unless we take further action, f must be such that $\{f(g_1), f(g_2) = \pm h_2\}$ generates H. Then $f(g_1)$ must be of the form $k_1h_1 + nh_2$ for some $k_1 \in \{1, -1\}$ and $n \in \mathbb{Z}$. Wait for $f(g_2) = k_2h_2$ to be declared, where $k_2 =\in \{1, -1\}$ and also wait for $f(g_1)$ to be declared. Note that we are in the $m = 0$ case, and thus we have not taken any action like we did in the $m \neq 0$ case that would change the generators of G and H, so if f does not halt on both of these values, we can continue building G and H as before and win.

So suppose $f(g_1)$ and $f(g_2)$ have both halted and are $k_1h_1+nh_2$ and k_2h_2, respectively. We will show that we can reveal new elements of G and H such that f cannot be an isomorphism, thus we will win.

If $n \not\equiv 0 \pmod 5$, then let $d \coloneqq 0$. Otherwise, $n \equiv 0 \pmod 5$, and let $d \coloneqq 1$. We proceed as follows: Reveal a new element u of G such that $5u = g_1 + dg_2$ and a new element v of H such that $5v = h_1 + 2dh_2$. That is, $u \coloneqq (0, 1/5, d/5)$ and $v \coloneqq (1, 2/5, (5+2d)/5)$. Now $A \oplus G$ is generated by $\{a, u, g_2\}$ and $B \oplus H$ is generated by $\{b, v, h_2\}$, and as before the relationships between elements in $A \oplus G$ and $B \oplus H$ remains the same. Also, $A \oplus G$ is still equal to $B \oplus H$, since we have

$$\begin{bmatrix} -2 & -5 & -2+d \\ 1 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ u \\ g_2 \end{bmatrix} = \begin{bmatrix} b \\ v \\ h_2 \end{bmatrix}$$

as can be checked easily, and this matrix is invertible over $\mathbb{Z}$. Then the element $g_1+dg_2 = 5u$, so $f(g_1+dg_2)$ must also be divisible by 5. Such a divisor must be of the form $pv+qh_2$ for some $p, q \in \mathbb{Z}$ since $f(g_1+dg_2) \in H$, and H is generated by $\{v, h_2\}$. Then we have

$$\begin{aligned} f(g_1+dg_2) &= 5(pv+qh_2) \\ f(g_1)+df(g_2) &= p(5v)+5qh_2 \\ k_1h_1+nh_2+dk_2h_2 &= p(h_1+2dh_2)+5qh_2 \\ k_1h_1+(n+dk_2)h_2 &= ph_1+(2pd+5q)h_2 \end{aligned}$$

and so $p = k_1$ and $n + dk_2 = 2pd + 5q = 2k_1d + 5q$. Hence $n = 5q + (2k_1 - k_2)d$. If $n \not\equiv 0 \pmod 5$, then $d = 0$, and we have $n = 5q$, a contradiction. On the other hand, if $n \equiv 0 \pmod 5$, then $d = 1$ and we have $0 \equiv n \equiv (2k_1 - k_2) \pmod 5$, but $k_1, k_2 \in \{1, -1\}$, so this is impossible. In either case, f fails to be an isomorphism, since it maps the element $g_1 + dg_2$ which is divisible by 5 to an element of H not divisible by 5. □

4.3 Further Work

As we have seen above, when we must supply the groups and generators in advance, we know only that the problem is not computable. Thus we have the following question:

Question 4.3.
What is the complexity of uniformity in Walker's Cancellation Theorem when we know both the groups and generators? Is it a natural complexity like $\varnothing'$, or something else?

Furthermore, what happens when we weaken the information we give to the uniform procedure? If we give only the groups and allow the generators to change, does the complexity increase? If so, to what? Similarly, what happens if we give only the generators, but allow the size of the groups to change?

Additionally, notice that although we are permitted to use any finitely generated abelian groups as A and B, the troublesome groups were, essentially, direct sums of $\mathbb{Z}$, which can be represented as lattices. Is there some analogous question that we could ask about lattices that would provide more insight, or conversely, show that some problem of lattices is also of interest?

Finally, as noted in Lubarsky and Richman [20], Walker's Cancellation Theorem is, in some sense, directly related to the projectivity of subgroups of $\mathbb{Z}$ via the following classical result, which is easily seen to be able to be effectivized:

Theorem.
Let A be an abelian group and f and g surjective homomorphisms from A onto $\mathbb{Z}$. Then $f(\ker(g)) = g(\ker(f))$.

It is possible that this property on kernels is equivalent to Walker's Cancellation Theorem. Similarly, it could be the case that the non-effective portion of the proof is all one needs, i.e. that subgroups of $\mathbb{Z}$ are projective. Here, when we say "equivalent", we mean in the sense of Weihrauch reducibility (see, for instance Brattka and Gherardi [7] for an overview) which allows us to relate computational complexity of theorems.

Chapter 5

Bounded Turing Reducibility and the Bounded Jump

5.1 Background

Recall that when A is Turing reducible to B (i.e. $A \leq_T B$), we have a Turing functional Γ such that $\Gamma^B = A$. However, many natural reductions also have the property that we can, ahead of time, produce a computable function f such that the use of the functional is bounded by f, so that $\Gamma^{B \upharpoonright f(x)}(x) = A(x)$ for all x. Such reductions are called *bounded* Turing reductions, and we refer to the function f as the *use bound* for the functional Γ. We write $A \leq_{bT} B$ in this case.

Such reductions can also be seen as a weakened form of a truth-table reduction, where A is truth-table reducible to B if there exists a computable function $k : \omega \to \omega$, a uniformly computable collection of truth tables $T_x : \{0,1\}^{k(x)} \to \{0,1\}$ and a computable *use location* function $g : \omega \to \omega^{k(x)}$ so that $T(B(g(x))) = A(x)$ for all x. In other words, g determines a $k(x)$-tuple of locations within ω, which are then viewed as a row in the truth table T_x by checking which elements of the tuple belong to B. We write this as $A \leq_{tt} B$. The crucial difference between a truth-table reduction and a bounded Turing reduction is that in the former, the truth table (which corresponds to $\Gamma(x)$) is computable and thus must be able to produce output no matter which $k(x)$-tuple it receives, whereas in the latter, $\Gamma(x)$ may diverge with an oracle that is not B. Because of this, it is traditional to refer to bounded Turing reductions as *weak truth-table* reductions, written $A \leq_{wtt} B$, but we do not follow this convention, since in most cases it is more natural for us to think of the reduction as a Turing reduction whose use is bounded rather than a truth-table reduction whose truth table is not total.

Because the jump is a strictly increasing operator, we can wonder if it is surjective (on degrees). This type of property is known as *jump inversion*, and there a few important variants that we mention:

Theorem 5.1 (Friedberg Jump Inversion)**.**
For every $A \geq_T \varnothing'$, there is some X such that $X' \equiv_T A \equiv_T X \oplus \varnothing'$.

Theorem 5.2 (Shoenfield Jump Inversion ([27]))**.**
For every A that is c.e. in and above $\varnothing'$, there is some $X \leq_T \varnothing'$ such that $X' \equiv_T A$.

Theorem 5.3 (Sacks Jump Inversion ([26]))**.**
For every A that is c.e. in and above $\varnothing'$, there is a non-computable c.e. set X such that $X' \equiv_T A$.

Note that Sacks Jump Inversion is a strict improvement over Shoenfield Jump Inversion, because it improves X from Δ^0_2 to non-computable c.e., i.e. strictly Σ^0_1.

A natural question is to ask if analogues of these results hold for the bounded Turing degrees. Friedberg Jump Inversion holds even for truth-table degrees, as Anderson [1] showed, while Csima, Downey and Ng [11] showed that Shoenfield (and hence Sacks) Jump Inversion fails for both the bounded Turing and truth-table degrees. In response to this failure, Csima and Anderson [2] produced a definition of the jump more suitable to the bounded Turing degrees.

The bounded jump of a set A, denoted A^b, is defined as

$$A^b \coloneqq \{x \mid (\exists i \leq x)[\varphi_i(x)\downarrow \wedge \Phi_x^{A \upharpoonright \varphi_i(x)}(x)\downarrow]\}.$$

Using this definition, which they showed satisfied the expected properties of a jump operator – for instance, it is strictly increasing – they were able to prove the analogue of Shoenfield Jump Inversion holds. They also proved a useful characterization between bounded Turing reductions and the Ershov hierarchy, which we will make use of and expand upon.

Because of how often this characterization is used in the following sections, it is worth mentioning it here. Recall that a set A is said to be *ω-c.e.* if there is some computable approximation function $f : \omega \times \omega \to \{0,1\}$ and a computable function $g : \omega \to \omega$ such that

- $\lim_{s\to\infty} f(x,s) = A(x)$ for all x, and
- $|\{s \mid f(x,s) \neq f(x,s+1)\}| \leq g(x)$ for all x.

Then a well-known result says that $A \leq_{bT} \varnothing'$ exactly when A is ω-c.e. The characterization expands upon this using a more general definition. We give this in a slightly different format than it is usually presented, but it is essentially equivalent to those seen elsewhere.

Definition 5.1 (α-c.e.)**.**
Let α be a computable ordinal. We say that a set A is *α-c.e.* if there is a computable function $f : \omega \times \omega \to \{0,1\} \times \alpha$ such that for all x:

- $\lim_{s\to\infty} f_0(x,s) = A(x)$ and $f_0(x,0) = 0$,
- if $f_0(x,s+1) \neq f_0(x,s)$, then $f_1(x,s+1) < f_1(x,s)$, and
- $f_1(x,s+1) \leq f_1(x,s)$.

(Here f_0 and f_1 are the projections of f onto the first and second coordinate, respectively.)

The full characterization states that $A \leq_{bT} \varnothing^{nb}$ exactly when A is ω^n-c.e. The reader familiar with results about such sets may be aware that the coding of computable ordinals has an effect on which sets are ω^2-c.e. and which are not, which is not suitable for our purposes. In Section 5.2, we resolve the problem by showing that not any coding of computable ordinals will do for our purposes, but rather a particularly nice one that computably admits a Cantor normal form for the ordinals we care about. Such codings agree on which sets are ω^2-c.e. and thus do not pose a problem.

Once we have resolved this issue, we prove two results that answer questions of Anderson, Csima and Lange [3]. Recall the following definitions:

Definition 5.2.
Let A be a set. Then:

- A is *low* if $A' \leq_T \varnothing'$,
- A is *high* if $A' \geq_T \varnothing''$,
- A is *bounded low* if $A^b \leq_{bT} \varnothing^b$,
- A is *bounded high* if $A^b \geq_{bT} \varnothing^{2b}$,
- A is *superlow* if $A' \leq_{tt} \varnothing'$,
- A is *superhigh* if $A' \geq_{tt} \varnothing''$.

Section 5.3 is devoted to the proofs of these results, which are the following:

Theorem 5.5.
There is a c.e. set A that is low and bounded low and is also not superlow. Since any superlow set is easily seen to be both low and bounded low, this shows that the low, bounded low sets properly contain the superlow sets.

Theorem 5.6.
There is a c.e. set A that is bounded low and Turing complete, hence a set that is superhigh but not bounded high.

Next, in Section 5.4 we extend the characterization mentioned above so that it works relative to an arbitrary oracle. That is, we give equivalent conditions for when a set A is bT-reducible to B^{kb} for some set B and $k > 0$. Although the characterization is not particularly beautiful – and hence we do not reproduce it here for brevity – we are able to use it to prove the following result, to which Section 5.5 is dedicated:

Theorem 5.13.
For any set B, there is a set A such that $B <_{bT} A <_{bT} B^b$. Furthermore, $B^b <_{bT} A^b <_{bT} B^{2b}$.

This theorem shows that (relativized) bounded jump inversion is non-trivial relative to any oracle B and we expand more upon this question in Section 5.6 and detail further work and open problems related to the bounded jump.

5.2 Resolving A Foundational Problem

In various proofs relating the bounded jump to the Ershov hierarchy – for example, Anderson and Csima [2] – we use a computable coding of some initial segment of the computable ordinals. The details are omitted, but this seemingly causes a problem: A result of Epstein, Haas and Kramer [15] shows that given any $f \leq_T \varnothing'$, there is some system of notation S in which f is ω^2-c.e., which suggests that the hierarchy is highly dependent on the exact choice of how ordinals are coded. To resolve this, we will first briefly remind the reader of the definition of a system of notation, then reproduce this proof, comment on why the proof proceeds as it does, and then finally show that this result does not actually pose a problem for our purposes.

We begin with the definition of the system of notations, due to Rogers.

Definition 5.3 (System of Notation, Rogers [25, pg. 205])**.**
A *system of notation* S below the ordinal α is a map ν from $D \subseteq \omega$ onto the set of ordinals below α with the following additional functions:

- A computable function k_S so that $k_S(x)$ determines the type of ordinal that $\nu(x)$ is. (Among zero, successor, non-zero limit).

- A partial computable function p_S so that for all x with $\nu(x)$ a successor ordinal, $p_S(x)\downarrow$ and $\nu(p_S(x)) + 1 = \nu(x)$. That is, p_S determines a predecessor ordinal, if possible.
- A partial computable function q_S so that for all x with $\nu(x)$ a non-zero limit ordinal, $e \coloneqq q_S(x)\downarrow$, φ_e is total and $\{\nu(\varphi_e(0)), \nu(\varphi_e(1)), \ldots\}$ is a strictly increasing set of ordinals with limit $\nu(x)$. Such a sequence is called a *fundamental sequence for* $\nu(x)$.

The stated form of the definition does not quite match the original, where k_S was permitted to be only partial computable on ω but computable on D. We are not interested in what occurs for x not in D, so restricting k_S in this way is not an issue.

We will informally say that x is "a name" for the ordinal $\nu(x)$. In the result of [15], each ordinal has at most one name, but this need not be true of a generic system of notation.

Notice that this definition is restrictive because it requires that we can determine predecessors and fundamental sequences effectively, but is also rather weak since there is no guaranteed computable way of performing ordinal operations like addition or multiplication.

We now turn to the proof we are interested in examining.

Theorem 5.4 (Epstein, Haas and Kramer [15, 8b])**.**
For each $g \leq \varnothing'$, there is a system of notation S such that g is ω^2-c.e. under this coding.

Proof:
Since $g \leq \varnothing'$, we can by the Limit Lemma (see Soare [28]) find some computable function $f(x,s)$ with $g(x) = \lim_{s\to\infty} f(x,s)$ for all x and $f(x,0) = 0$ for all x. Let $X \coloneqq \{\langle x,s\rangle \mid s = 0 \vee f(x,s) \neq f(x,s-1)\}$ and define $\langle x,s\rangle \leq_R \langle y,t\rangle$ if either $x \leq y$, or $x = y$ and $s \geq t$. Notice that $(X, \leq_R)$ is a well-order of type ω, and also both X and $\leq_R$ are computable since $f(x,s)$ is.

Pad X into a system of notation as follows: The domain D is $\omega^{[X]} \coloneqq \{\langle u,m\rangle \mid u \in X \wedge m \in \omega\}$, and we define $\langle u,m\rangle \leq_S \langle v,n\rangle$ if $v \leq_R u$ should $v \neq u$ or if $m \leq n$ should $u = v$. Then $(D, \leq_S)$ is a well-order of type ω^2, with the ordinal $\omega \cdot i + j$ having name $\langle u,j\rangle$, where the ith element of $(X, \leq_R)$ is u. We will give more detail on why S satisfies the full requirements to be a system of notation in a later remark.

Define the computable function $d(x,s) \coloneqq \langle f(x,s), \langle\langle x,t\rangle, 0\rangle\rangle$, where $t \leq s$ is largest so that $\langle x,t\rangle \in X$. That is, if we treat the second coordinate as an ordinal in the system of notation S, then $d(x,s) \coloneqq \langle f(x,s), \omega \cdot i\rangle$, where i is chosen so that it has name $\langle x,t\rangle$

with $t \leq s$ as large as possible. Then it is not hard to see that g is ω^2-c.e. via d in the system of notation S.

Note: The original proof used a different (but equivalent) definition of ω^2-c.e., which we have adapted to fit our definition of α-c.e. □

In Fig. 5.1, we reproduce Diagram 3 from [15], which is very instructive in understanding the definition of X and also why d will work.

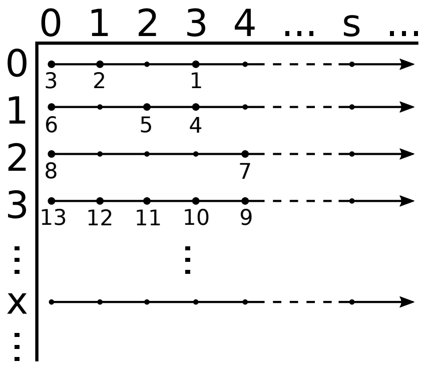

Figure 5.1: Enumeration order for $(X, \leq_R)$ as determined from $f(x,s)$

The attentive reader may perhaps wonder why we must pad $(X, \leq_R)$ to $(D, \leq_S)$. After all, can we not simply set

$$\hat{d}(x,s) \coloneqq \langle f(x,s), \langle x,t \rangle \rangle$$

where $t \leq s$ is largest so that $\langle x,t \rangle \in X$ as before, and hence g would be ω-c.e. via $\hat{d}$? The answer is that we cannot, because X cannot be made into a system of notation. Recall from the definition that not only do we require some set D that acts as the "names" of ordinals, but we also require that we have three functions that give some information about those ordinals. In this case, it is easy to see that k_X and q_X could be defined, but p_X cannot be partial computable unless g is computable – the diagram is again instructive as to why.

By padding out X into S, we obviate this problem. It is easy to see that in S, k_S and p_S are easily defined and now it is q_S that is the sticking point. We repeat here the proof of Rogers [25][11.8.XX] that produces a q_S that will work. We know that all names for non-zero limit ordinals have the form $\langle u, 0 \rangle$. Let m be the minimal element for X. Fix $u \in X$.

We will define the function $h(x)$ to produce a fundemental sequence for the non-zero limit ordinal $\langle u, 0\rangle$. We do this inductively:

$$h(0) \coloneqq \langle m, 0\rangle$$
$$h(n+1) \coloneqq \begin{cases} \langle w, 0\rangle, & \text{if } w = (\mu x \leq n)[s <_R x <_R u] \\ \langle s, t+1\rangle, & \text{if no such } w \text{ exists.} \end{cases} \quad \text{where } h(n) \coloneqq \langle s, t\rangle.$$

Then h computes a fundamental sequence for $\langle u, 0\rangle$, and so we set $q_S(\langle u, 0\rangle) \coloneqq y$ where y is an index for the partial computable function h.

We now resolve the issue: not just any coding of ordinals will work for our purpose of defining ω^k-c.e. We also require that the system of notation has some way of computing a Cantor normal form from the names of the ordinals. That is, if x is the name of the ordinal $\omega^k \cdot a_k + \cdots + \omega \cdot a_1 + a_0$, then there should be some computable function g that maps x to $(a_k, \ldots, a_0)$. We only work with ordinals less than some pre-determined ω^{k+1}, so we can assume that our Cantor normal form can be encoded in such a way; we call such a g a *Cantor normal form function* or *CNF function*. Recall from the remark above that the existence of a CNF function is not intrinsically prescribed by a system of notation and must be justified separately. A result of Ash and Knight [4] shows that once we choose our largest ordinal we care about, we can always find some particular system of notation where a Cantor normal form representation exists. So this is not a tall order in the cases we are interested in. Indeed, if we choose that largest ordinal to be ω^{k+1}, then it can be easily determined that the representation that is constructed by that proof is effectively the same as the $(k+1)$-tuple produced by a CNF function.

Consider the system of notation S that we constructed above – that was designed so that $g \leq_T \varnothing'$ was ω^2-c.e. Suppose S has a CNF function C. Since S represents ordinals less than ω^2, C maps elements of D to ordered pairs, where $C(x) = (i, j)$ if x is the name of the ordinal $\omega \cdot i + j$. Notice that by our definition of D, $\langle u, m\rangle$ names the ordinal $\omega \cdot i + m$ where $u \in X$ is associated to $i \in \omega$ as we view $(X, \leq_R)$ as ω. So we can define $\hat{C}(u) \coloneqq g(\langle u, 0\rangle)$ for all $u \in X$, so that $\hat{C}$ provides an order-preserving isomorphism from $(X, \leq_R)$ to ω.

Fix x and define $i \coloneqq \hat{C}(\langle x-1, 0\rangle)$ should $x > 0$ and -1 should $x = 0$. Enumerate X until we find $\langle x, s\rangle$ such that $\hat{C}(\langle x, s\rangle) = i + 1$. Then $g(x) = f(x, s)$, which we can compute. So $g \leq_T \hat{C} \leq_T C$. If we demand that the CNF function C be *computable*, then the the function g that S was designed to make ω^2-c.e. was already computable and so trivially ω^2-c.e.

Hence, if the results of Anderson and Csima are modified so that they require a computable CNF function (as noted in the follow-up paper Anderson, Csima and Lange [3]), then this resolves this issue. In fact, given those results, this is enough to make the notion of ω^n-c.e.

completely well-defined, since then the proofs of Anderson and Csima provide an ordinal-free characterization of ω^n-c.e., namely being bT-below $\varnothing^{nb}$, and so the choice of notations for ω^n cannot have any impact on which sets are ω^n-c.e.

5.3 Bounded Low and Bounded High

The aim of this section is to answer some questions posed by Anderson, Csima and Lange [3]. Recall from Definition 5.2 that we say that a set A is *low* if $A' \leq_T \varnothing'$ and *high* if $A' \geq_T \varnothing''$, and analogously, A is *bounded low* if $A^b \leq_{bT} \varnothing^b$ and *bounded high* if $A^b \geq_{bT} \varnothing^{2b}$. Finally, following the definition of Mohrherr [21], a set A is said to be *superlow* if $A' \leq_{tt} \varnothing'$, and *superhigh* if $A' \geq_{tt} \varnothing''$.

In Anderson, Csima and Lange [3], the existence of a c.e. set that was bounded low and yet high was demonstrated, as was the existence of a set bT-below $\varnothing'$ that is bounded high and yet low. These two results show the disconnect between classical low/highness and bounded low/highness. In that vein, they posed some questions about refining the overlap between these classes. We reproduce the questions of interest here:

Question (Anderson, Csima and Lange [3, 4.2a])**.**
Does there exist a bounded low set that is low, but not superlow?

Question ([Anderson, Csima and Lange [3, 4.3])**.**
Does there exists a set that is superhigh but not bounded high?

We answer both of these positively. In fact, for the latter, we improve both of the conditions by constructing a set that is complete (hence superhigh) but is bounded low (hence not bounded high). These results were proved independently and simultaneously by Wu and Wu [30], who also answered additional open problems from [3]. Their method for these other open problems is illuminating for the question of jump inversion, which we remark more on at the conclusion to this chapter.

5.3.1 Low, Bounded Low, Not Superlow

We construct a c.e. low, bounded low but not superlow set A via a finite injury construction.

Theorem 5.5.
There is a c.e. set A that is low and bounded low and is also not superlow. Since any superlow set is easily seen to be both low and bounded low, this shows that the low, bounded low sets properly contain the superlow sets.

Proof:

We aim to meet the following requirements.

To ensure that A is low, we meet the standard lowness requirements (see Soare [28]):

$\mathbf{L}_e$: $(\exists^\infty s)[\Phi_e^A(e)[s]\downarrow] \Rightarrow \Phi_e^A(e)\downarrow$.

As in the proof of the bounded low set that is high from [3], we have the following requirements that ensure A^b is ω-c.e.:

$\mathbf{P}_x$: If $\varphi_{n,s}(x)\downarrow$ with $n \le x$, then all requirements $\mathbf{U}_e$ with $e \ge x$ are restrained from enumerating any $y \le \varphi_n(x)$ into A after stage s.

Meeting these requirements will then show that $A^b \le_{bT} \varnothing' \equiv_1 \varnothing^b$, and so A will be bounded low. A requirement $\mathbf{P}_x$ imposes a restraint on all requirements $\mathbf{U}_e$ with $e \ge x$ so that we can ensure that A^b is ω-c.e. In fact, we will show that the computable function $g(x) \coloneqq 2x(x+1)$ bounds the number of changes in the natural approximation of $A^b(x)$, as in [3].

Finally, we must ensure that A is not superlow. To do so, we note that A is superlow if and only if A' is ω-c.e., so we have the following requirements, which follow the requirements used in the construction of a low but not superlow set in Nies [24].

$\mathbf{U}_{\langle i,j\rangle}$: There is an x such that $\varphi_i(x,t)$ cannot approximate $A'(x)$ with at most $\varphi_j(x)$-many changes. That is, if $\varphi_j(x)\downarrow$ and $\varphi_i(y,s)$ is a total $\{0,1\}$-valued function with $\lim_{s\to\infty}\varphi_i(y,s) = A'(y)$ for all y, then there are at least $(\varphi_j(x)+1)$-many stages s where $\varphi_i(x,s) \ne \varphi_i(x,s+1)$.

We build a Turing functional $\Gamma^Y(x,q)$. For a fixed q, we use the following procedure to define $\Gamma^Y(x,q)$ and also a set A_q at the same time. For brevity in the construction, however, we write A for A_q.

Let $\Psi^Z(y,z) \coloneqq \Phi_q^Z(y)$ for all z. By the relativized s-m-n Theorem (see Soare [28]), there is a computable, injective function p such that for all sets Z and for all z, $\Phi_{p(y)}^Z(z) = \Psi^Z(y,z) = \Phi_q^Z(y)$. In particular, take $z = p(y)$ to see that $\Phi_{p(y)}^Z(p(y)) = \Phi_q^Z(y)$ for all sets Z.

We define a restraing function $R(s)$ for all the requirements $\mathbf{P}_x$ at stage s. Let $R(s) \coloneqq \max\{k(n,x,s) \mid n \le x \le s\}$ where $k(n,x,s) \coloneqq \begin{cases} \varphi_n(x), & \text{if } \varphi_{n,s}(x)\downarrow \\ -1, & \text{otherwise} \end{cases}$.

Note that R is finite and computable.

To each requirement $\mathbf{U}_e$, we will associate a counter c_e, a witness x_e and markers m_e and t_e. We will always choose x_e and m_e from $\omega^{[e]}$.

To build the c.e. set A, we proceed by stages:

Construction:

Stage 0: Set $A_0 \coloneqq \varnothing$. Set $x_e \coloneqq \langle e, 0\rangle$, $t_e \coloneqq 0$, $c_e \coloneqq 0$ for all e, and leave all m_e undefined.

Stage $s+1$: If $\varphi_n(x)$ converges for the first time at stage $s+1$ and $n \leq x \leq s$, then we say that $\mathbf{P}_x$ *acts*. For all $e \geq x$, run the injury procedure for $\mathbf{U}_e$, which we describe at the end of the construction.

For each $e \leq s$, if $\Phi_e^A(e)[s]\downarrow$ but $u_e^A(e)[s-1] \neq u_e^A(e)[s]$ in case $s \geq 1$, then we say that $\mathbf{L}_e$ *acts*. For all $i \geq e$, run the injury procedure for $\mathbf{U}_i$. (If $\Phi_e^A(e)[s-1]\uparrow$, we say that $u_e^A(e)[s-1] = -1$.)

In any case, choose the least $\langle i, j\rangle \leq s$ such that $\varphi_{j,s}(z)\downarrow$, $c_{\langle i,j\rangle} \leq \varphi_j(z) + 1$, $\mathbf{U}_{\langle i,j\rangle}$ has no requests issued and either

$$c_{\langle i,j\rangle} \text{ is even } \wedge (\exists t \leq s)[t > t_{\langle i,j\rangle} \wedge \varphi_{i,s}(z,t) = 0]$$

or

$$c_{\langle i,j\rangle} \text{ is odd} \wedge (\exists t \leq s)[t > t_{\langle i,j\rangle} \wedge \varphi_{i,s}(z,t) = 1],$$

where $z \coloneqq p(x_{\langle i,j\rangle})$.

In the first case, define $\Gamma^A(x_{\langle i,j\rangle}, q)\downarrow$ with use $u > m_{\langle i,j\rangle}$, where we define $m_{\langle i,j\rangle}$ to be some value from $\omega^{[\langle i,j\rangle]}$ not yet in A and exceeding $R(s)$ and the stage number $s+1$. For all $e > \langle i,j\rangle$ run the injury procedure for $\mathbf{U}_e$. Issue a *request for convergence* for $\mathbf{U}_{\langle i,j\rangle}$. We insist that the convergence of $\Gamma^Y(x_{\langle i,j\rangle}, q)$ depends only on the single value $m_{\langle i,j\rangle}$, so that $\Gamma^Y(x_{\langle i,j\rangle}, q)\downarrow$ exactly when $m_{\langle i,j\rangle} \notin Y$. This will allow us to enumerate $m_{\langle i,j\rangle}$ into A at some later stage if we desire, and so destroy the convergence of $\Gamma^A(x_{\langle i,j\rangle}, q)$.

In the second case, enumerate $m_{\langle i,j\rangle}$ into A. Issue a *request for divergence* for $\mathbf{U}_{\langle i,j\rangle}$.

In either case, we say that requirement $\mathbf{U}_{\langle i,j\rangle}$ *acts* at stage $s+1$ and increment $c_{\langle i,j\rangle}$ by 1.

Finally, for all $\mathbf{U}_e$ that have requests issued, we attempt to *fulfill their request* at end of stage $s+1$ in the following way. Let $z \coloneqq p(x_e)$. If $\mathbf{U}_e$ has a request for convergence (resp. divergence) issued and $\Phi_z^A(z)[s+1]$ converges (resp. diverges) then set $t_e \coloneqq s+1$ and consider the request fulfilled.

Injury Procedure: Whenever a requirement $\mathbf{U}_e$ is injured, we run the following procedure: set $t_e \coloneqq 0$, $c_e \coloneqq 0$, and set m_e to be undefined. If $x_e = \langle e, i\rangle$ for some i, set $x_e \coloneqq \langle e, i+1\rangle$. Cancel all requests that have been issued for $\mathbf{U}_e$. We say that the requirement $\mathbf{U}_e$ has been *reinitialized*.

This completes the construction of A_q and the definition of the partial computable Turing function $\Gamma^Y(x,q)$.

By the relativized s-m-n Theorem, there is some computable function $f(q)$ such that $\Phi^Y_{f(q)}(x) = \Gamma^Y(x,q)$ for all sets Y. By the relativized Recursion Theorem (see [28]), there is a fixed point for f, say q', so that $\Phi^Y_{q'}(x) = \Phi^Y_{f(q')}(x) = \Gamma^Y(x,q')$ for all Y. Notice in the construction, we choose p – depending on q – so that $\Phi^Y_{p(y)}(p(y)) = \Phi^Y_q(y)$, so this gives $\Phi^Y_{p(x)}(p(x)) = \Gamma^Y(x,q')$ for the function p associated to q'. That is, $p(x) \in Y' \Leftrightarrow \Phi^Y_{p(x)}(p(x))\downarrow \Leftrightarrow \Gamma^Y(x,q')\downarrow$. We claim that the set $A_{q'}$ satisfies all the requirements and is our desired set. Again, for brevity we write A, but the reader should not forget that our choice of q' allows us to relate Y' to $\Gamma^Y(x,q')$ via p.

Claim 5.5.1: For every e, requirement $\mathbf{L}_e$ acts at most finitely often and is met. For every x, requirement $\mathbf{P}_x$ acts at most finitely often. For every e, requirement $\mathbf{U}_e$ acts at most finitely many times.

Proof of Claim: By induction, we may assume that all higher priority requirements act at most finitely often, and hence there is some stage s_0 such that no requirement of higher priority acts at any stage $t \geq s_0$.

Suppose the requirement under consideration is $\mathbf{L}_e$ for some e. If $\Phi^A_e(e)[t]\uparrow$ for all $t \geq s_0$, then $\mathbf{L}_e$ never acts after stage s_0, and so acts at most finitely often. Further, $\mathbf{L}_e$ is met vacuously, since its hypothesis fails. So suppose some stage $t \geq s_0$ exists with $\Phi^A_e(e)[t]\downarrow$. Let $t' \leq t$ be the smallest stage where $u^A_e(e)[s] = u^A_e(e)[t]$ for all $t' \leq s \leq t$ – note that it need not be the case that $t' \geq s_0$. Then $\mathbf{L}_e$ must have acted at stage t' and reinitialized all lower priority requirements. So after stage t', all lower priority requirements would only be able to enumerate values exceeding t' into A. Since all higher priority requirements stop acting at stage s_0, no values below t' can be enumerated into A beyond stage t. By assumption, $u^A_e(e)[t] = u^A_e(e)[t'] \leq t'$ and so the computation $\Phi^A_e(e)[t]$ will persist forever as no value below its use will be enumerated into A after stage t. Thus $\mathbf{L}_e$ will never act after stage t, and $\mathbf{L}_e$ is met, since $\Phi^A_e(e) = \Phi^A_e(e)[t]$.

Now suppose the requirement under consideration is $\mathbf{P}_x$ for some x. It is easy to see that $\mathbf{P}_x$ acts at most finitely often; indeed, it can act at most $x+1$ times, once for each $n \leq x$, since $\varphi_n(x)$ can converge for the first time at most once.

Finally, suppose the requirement under consideration is $\mathbf{U}_{\langle i,j\rangle}$ for some i, j. Notice that the condition $c_{\langle i,j\rangle} \leq \varphi_j(z) + 1$ forces $\mathbf{U}_{\langle i,j\rangle}$ to act at most finitely often provided it is never reinitialized, since $c_{\langle i,j\rangle}$ is only ever incremented, or reset by reinitialization of $\mathbf{U}_{\langle i,j\rangle}$. It is plain to see from the construction that $\mathbf{U}_{\langle i,j\rangle}$ can only be reinitialized by higher priority requirements. Hence after stage s_0, $\mathbf{U}_{\langle i,j\rangle}$ will never be reinitialized, and so will act at most $(\varphi_j(z) + 2)$-many times after stage s_0. Thus $\mathbf{U}_{\langle i,j\rangle}$ acts at most finitely often. ■

Since A satisfies all the lowness requirements, it is low. It remains to show that A^b is ω-c.e., while A' is not ω-c.e., i.e. A is bounded low but not superlow.

Claim 5.5.2: A^b is ω-c.e.

Proof of Claim: To show that A^b is ω-c.e., we use the same approximation function as in [3]. That is, let

$$f(x,s) \coloneqq \begin{cases} 1, & \text{if } (\exists n \leq x)[\varphi_{n,s}(x)\downarrow \wedge \Phi_x^{A\upharpoonright\varphi_n(x)}(x)[s]\downarrow] \\ 0, & \text{otherwise} \end{cases}$$

Suppose that $f(x,s) = 1$, i.e. there is some $n \leq x$ such that $\varphi_{n,s}(x)\downarrow$ and $\Phi_x^{A\upharpoonright\varphi_n(x)}(x)[s]\downarrow$. Since $\varphi_{n,s}(x)\downarrow$, there was a stage $s' \leq s$ where $\varphi_n(x)$ first converged. So at stage s', $\mathbf{P}_x$ must have acted and reinitialized all lower priority requirements. If those requirements enumerate a value into A after stage s', they must have chosen it to exceed $R(s')$ since $R(t) \geq R(s')$ for all stages $t \geq s'$. We know $R(s') \geq \varphi_n(x)$ since $\varphi_{n,s'}(x)\downarrow$, so if any requirement of lower priority than $\mathbf{P}_x$ enumerated a value after stage s', it would not injure the computation $\Phi_x^{A\upharpoonright\varphi_n(x)}(x)[s]$.

So it remains to consider the higher priority requirements, since they are not reinitialized at stage s' and still may enumerate a value below $\varphi_n(x)$. We show that each of these requirements can only injure the computation $\Phi_x^{A\upharpoonright\varphi_n(x)}(x)[s]$ at most once, for each $n \leq x$. Indeed, if the requirement $\mathbf{U}_e$ enumerates a value m_e into A, it may choose a new value for m_e at a later stage. But at any later stage t, *all* requirements obey the restraint function $R(t)$ and hence m_e will be chosen to exceed $R(t)$ – and hence exceed $\varphi_n(x)$ – even though $\mathbf{U}_e$ is of higher priority than $\mathbf{P}_x$.

Hence for a given x, each requirement of higher priority than $\mathbf{P}_x$ can enumerate below $\varphi_n(x)$ for each $n \leq x$ at most once. Therefore the number of changes to $f(x,s)$ for a given x is bounded by $g(x) \coloneqq 2x(x+1)$, by the same proof as in [3]. So A^b is ω-c.e., and thus A is bounded low. ■

Claim 5.5.3: Each requirement $\mathbf{U}_e$ is met for all e, which makes A' not ω-c.e.

Proof of Claim: Consider the requirement $\mathbf{U}_{\langle i,j\rangle}$.

We know by Claim 5.5.1 that all requirements of higher priority act at most finitely often, and hence there is a least stage s_0 after which $\mathbf{U}_{\langle i,j\rangle}$ is never reinitialized. Hence $x_{\langle i,j\rangle}$ would never be increased after stage s_0. For brevity, let x be this eventual value of $x_{\langle i,j\rangle}$. We claim that $z \coloneqq p(x)$ is a witness for $\mathbf{U}_{\langle i,j\rangle}$.

Suppose that $\varphi_j(z)\downarrow$ and $\lim_{s\to\infty}\varphi_i(z,s) = A'(z)$ where $\varphi_i(z,s)$ is in $\{0,1\}$ for all z and s. Since $\mathbf{U}_{\langle i,j\rangle}$ acts only finitely often and is never reinitialized after stage s_0, there must be some limiting value for $c_{\langle i,j\rangle}$, which we denote by c. We claim that $c > \varphi_j(z)+1$. Let t' be the limiting value of $t_{\langle i,j\rangle}$. Let $s' \geq s_0$ be the first stage after which $c_{\langle i,j\rangle}$ never changes. Since whenever $\mathbf{U}_{\langle i,j\rangle}$ acts we increment $c_{\langle i,j\rangle}$, we know that $\mathbf{U}_{\langle i,j\rangle}$ never acts after stage s'.

We first claim that if $\mathbf{U}_{\langle i,j\rangle}$ issues a request at stage $t \geq s_0$, it will be fulfilled at some future stage. Suppose otherwise.

First, if it is a request for convergence, we would have defined $\Gamma^A(x,q')\downarrow$ at stage t. All higher priority requirements have finished acting by stage s_0. Only $\mathbf{U}_{\langle i,j\rangle}$ can enumerate $m_{\langle i,j\rangle}$, as $m_{\langle i,j\rangle} \in \omega^{[\langle i,j\rangle]}$. Thus if the request for convergence is never fulfilled, then $\mathbf{U}_{\langle i,j\rangle}$ will never act again and this computation persists forever, i.e. $\Gamma^A(x,q')\downarrow$ for the final set A. By our choice of q', this means $z \in A'$. Then there is some $s \geq t$ such that $\Phi^{A_s}_{z,s}(z)$ by the use principle. But then at stage s we would fulfill the request for convergence.

Second, if it is a request for divergence, we would have $\Gamma^A(x,q')\uparrow$ at stage t. Since only $\mathbf{U}_{\langle i,j\rangle}$ can define convergence for x, if the request persists forever, then $\Gamma^A(x,q')\uparrow$ for the final set A. By our choice of q', this means $z \notin A'$. Since A satisfies the lowness requirements by construction, $\Phi^A_z(z)\uparrow$ means that all but finitely many stages s have $\Phi^{A_s}_z(z)\uparrow$. In particular, there is some $s \geq t$ such that $\Phi^{A_s}_z(z)\uparrow$ and so $\Phi^{A_s}_{z,s}(z)\uparrow$. But then at stage s we would fulfill the request for divergence.

Hence, $\mathbf{U}_{\langle i,j\rangle}$ never acting after some stage is not because a request was issued that was never fulfilled. Suppose $c \leq \varphi_j(z)+1$ for a contradiction. By assumption, the conditions $\varphi_j(z)\downarrow$ and $c_{\langle i,j\rangle} \leq \varphi_j(z)+1$ are met, so the only reason for $\mathbf{U}_{\langle i,j\rangle}$ not acting is because no value of $t > t_{\langle i,j\rangle} = t'$ could be found satisfying associated condition.

If c is even, it must be that $\varphi_i(z,t) = 1$ for all $t > t'$. By assumption, $\varphi_i(z,t)$ settles on $A'(z)$, so we get that $z \in A'$. Since $\mathbf{U}_{\langle i,j\rangle}$ never acts after stage s', we never define $\Gamma^A(x,q')\downarrow$ at any stage after s'. Hence $\Gamma^A(x,q')\uparrow$ for the final set A. But by our choice of q', this means $z \notin A'$, a contradiction.

If c is odd, it must be that $\varphi_i(z,t) = 0$ for all $t > t'$. So $z \notin A'$. Since A satisfies the lowness requirements by construction, all but finitely many stages s have $\Phi^{A_s}_z(z)\uparrow$. In particular, there is some stage $s > s'$ such that $z \notin A'_s$. So $\Gamma^{A_s}(z,q')\uparrow$. During the last stage where $\mathbf{U}_{\langle i,j\rangle}$ acts and incremented $c_{\langle i,j\rangle}$ to c, we would have defined $\Gamma^{A}(z,q')\downarrow$. Only $\mathbf{U}_{\langle i,j\rangle}$ can injure the computation, by enumerating $m_{\langle i,j\rangle}$. But by assumption this never happens. So $A_{s'} \upharpoonright u = A_s \upharpoonright u$ and yet $\Gamma^{A_s}(z,q')\uparrow$ but $\Gamma^{A_{s'}}(z,q')\downarrow$ with use u, a contradiction.

So in any case there is a contradiction, and hence $c > \varphi_j(z)+1$. Let $s' \leq s_1 < s_2 < \cdots < s_c$ be the stages where $\mathbf{U}_{\langle i,j\rangle}$ acts, so that $c_{\langle i,j\rangle}$ is incremented from $e-1$ to e at stage s_e. Then $\varphi_i(z,s_1-1)=0$, $\varphi_i(z,s_2-1)=1$, etc. Hence there must be at least $c-1$ stages s such that $\varphi_i(z,s-1)\neq\varphi_i(z,s)$. But $c-1>\varphi_j(z)$ and so the number of changes exceed $\varphi_j(z)$. Hence $\mathbf{U}_{\langle i,j\rangle}$ is met. ■

So A is low and bounded low, and since A' is not ω-c.e., A is not superlow. Therefore A is the desired set. □

5.3.2 Bounded Low, Complete

Theorem 5.6.
There is a c.e. set A that is bounded low and Turing complete, hence a set that is superhigh but not bounded high.

Proof:
In order to show that A is complete, we will meet, for all i, the following requirements:

$\mathbf{R}_i$: $\quad (\exists m)(\forall \ell > m)[\varphi_i(\ell)\downarrow \rightarrow p_{\bar{A}}(\ell) \geq \varphi_i(\ell)]$

where $p_{\bar{A}}(\ell)$ is the element of $\bar{A}$ in the ℓth position. That is, if $\bar{A}=\{a_0<a_1<\cdots\}$, then $p_{\bar{A}}(\ell) \coloneqq a_\ell$.

To show that A is bounded low, we use a modified version of the method used in the construction of the high, bounded low set in [3]. In that proof, we needed requirements to prevent "too much" action by $\mathbf{R}_i$ below certain restraints that we set, so that computations involved in the definition of A^b did not change too often. However, we can take aggressive pre-emptive action so that no explicit restraint functions are needed. Otherwise, the bulk of the argument will proceed in the same way. That is, to show that A is bounded low, we will equivalently show that A^b is ω-c.e., and to do so we will bound the number of changes in a natural approximation of A^b by a computable function.

Additionally, we will meet, for all e, the following restrictions to ensure that A is coinfinite:

$\mathbf{N}_e$: $|\bar{A}| \geq e$.

As per usual, at each stage $s+1$, we will be given an approximation A_s and enumerate finitely many values from $\bar{A}_s = \{a_{0,s} < a_{1,s} < a_{2,s} < \cdots\}$ into A to build A_{s+1}.

Construction:

Stage 0: Set $A_0 \coloneqq \varnothing$.

Stage $s+1$: Write $\bar{A}_s = \{a_{0,s} < a_{1,s} < a_{2,s} < \cdots\}$.

(Pre-emptive Action): If $\varphi_{n,s}(x)\downarrow$ newly (i.e. $\varphi_{n,s-1}(x)\uparrow$ but $\varphi_{n,s}(x)\downarrow$) with $n \leq x$, then enumerate all $a_{\ell,s}$ such that $\ell > x$ and $a_{\ell,s} \leq \varphi_{n,s}(x)$ into A.

(Normal Action): We ask:

$$(\exists i \leq s)(\exists \ell \leq s)[\ell > i \wedge \varphi_{i,s}(\ell)\downarrow > a_{\ell,s}].$$

If so, find the least such pair $\langle i, \ell\rangle$. Compute u least such that $a_{u,s} > \varphi_i(\ell)$ and enumerate $a_{\ell,s}, a_{\ell+1,s}, \ldots, a_{u-1,s}$ into A for the sake of $\mathbf{R}_i$. We say that these enumerations are *due to the pair* $\langle i, \ell\rangle$.

Set $A \coloneqq \cup_s A_s$. This completes the construction.

Claim 5.6.1: For all e, the limit $\lim_{s\to\infty} a_{e,s}$ exists. Hence all $\mathbf{N}_e$ hold, so A is coinfinite.

Proof of Claim: Fix e. Inductively, we may assume that there is some stage s_0 such that $\lim_{s\to\infty} a_{i,s} = a_{i,t}$ for all $t \geq s_0$ and for all $i < e$. So no elements below a_{e,s_0} are enumerated into A after stage s_0.

First notice that if $a_{\ell,s}$ is enumerated during pre-emptive action at some stage $s+1$, then $\ell > x$ and $\varphi_{n,s}(x)\downarrow$ newly for some $n \leq x$. Notice second that if $a_{\ell,s}$ is enumerated for the sake of $\mathbf{R}_i$ at stage $s+1$, then at any stage $t \geq s+1$, we have $a_{\ell,t} \geq a_{\ell,s+1} \geq a_{u,s} > \varphi_i(\ell)$, so $\mathbf{R}_i$ will not enumerate any elements into A at stage $t+1$.

It is not hard to see that we can find a stage $s_1 \geq s_0$ such that (a) for all $i, y \leq e$, $\varphi_i(y)\downarrow$ exactly if $\varphi_{i,s_1-1}(y)\downarrow$ – so that no $\varphi_{i,t}(y)\downarrow$ newly for $i, y \leq e$ at any stage $t \geq s_1$ – and (b) no $\mathbf{R}_i$ with $i < e$ enumerates into A due to a pair $\langle i, \ell\rangle$ with $\ell \leq e$ at any stage after s_1.

We show that $\lim_{s\to\infty} a_{e,s} = a_{e,s_1}$. Suppose otherwise. Then there is some stage $s'+1 \geq s_1$ where some $a_{\ell,s'}$ with $\ell \leq e$ is enumerated into A. By assumption, $s'+1 \geq s_1 \geq s_0$, so we must have $\ell = e$ by choice of s_0. For $a_{e,s'}$ to be enumerated during pre-emptive action at stage $s'+1$, we need $e > x$ and $\varphi_{n,s'}(x)\downarrow$ newly for some $n \leq x$. This contradicts the choice of s_1 by taking $y \coloneqq x < e$ and $i \coloneqq n \le x < e$ in (a). So $a_{e,s'}$ must have been enumerated for the sake of some $\mathbf{R}_i$. For this to be the case, we must have had the following condition hold:

$$(\exists \ell' \le s')[\ell' > i \wedge \varphi_{i,s'}(\ell')\downarrow > a_{\ell',s'}]$$

and $e \ge \ell'$ so that $a_{e,s'} \in \{a_{\ell',s'}, a_{\ell'+1,s'}, \ldots, a_{u-1,s'}\}$. Then $e \ge \ell' > i$ and $\mathbf{R}_i$ with $i < e$ enumerates into A due to a pair $\langle i, \ell' \rangle$ with $\ell' \le e$, contradicting (b). So no such stage $s' + 1$ exists.

Hence $a_{e,s} = a_{e,s_1}$ for all $s \ge s_1$ and so $\lim_{s\to\infty} a_{e,s}$ exists. So A is coinfinite, and we may write $p_{\bar{A}}(e) = \lim_{s\to\infty} a_{e,s}$ for all e. ■

Claim 5.6.2: For all i, $\mathbf{R}_i$ is satisfied. Hence A is complete.

Proof of Claim: Fix i and consider $m \coloneqq i$. Suppose $\ell > m$ and $\varphi_i(\ell)\downarrow$. Let s_0 be a stage such that $\lim_{s\to\infty} a_{\ell,s} = a_{\ell,t}$ for all $t \ge s_0$. Since a_{ℓ,s_0} never changes after stage s_0, $\mathbf{R}_i$ must never enumerate $a_{\ell,t} = a_{\ell,s_0}$ at any stage $t \ge s_0$. In particular, $\mathbf{R}_i$ never acts due to the pair $\langle i, \ell \rangle$ after stage s_0.

Since $\ell > m = i$, this should occur unless $\varphi_{i,t}(\ell)\downarrow > a_{\ell,t}$ fails for all $t \ge s_0$. Since $\varphi_i(\ell)\downarrow$ by assumption, there is some stage $t \ge s_0$ such that $\varphi_{i,t}(\ell)\downarrow$, so $\varphi_i(\ell)\downarrow = \varphi_{i,t}(\ell)\downarrow \le a_{\ell,t} = \lim_{s\to\infty} a_{\ell,s} = p_{\bar{A}}(\ell)$, as desired.

Thus $p_{\bar{A}}$ dominates every partial computable function on their domains, so A must be complete. ■

Claim 5.6.3: A^b is ω-c.e. That is, A is bounded low.

Proof of Claim: We construct a computable function $f : \omega \times \omega \to \{0, 1\}$ such that

- $\lim_{s\to\infty} f(x, s) = A^b(x)$ for all x and
- $f(x, s) \ne f(x, s + 1)$ for at most $g(x) \coloneqq 2(x + 3)(x + 1)$ many stages s.

Let

$$f(x,s) \coloneqq \begin{cases} 1, & \text{if } (\exists n \le x)[\varphi_{n,s}(x)\downarrow \wedge \Phi_x^{A\restriction\varphi_n(x)}(x)[s]\downarrow], \\ 0, & \text{otherwise.} \end{cases}$$

Suppose $f(x, s) = 1$, i.e. there is some $n \le x$ such that $\varphi_{n,s}(x)\downarrow$ and $\Phi_x^{A\restriction\varphi_n(x)}(x)[s]\downarrow$. Then there is some $s_0 \le s$ such that $\varphi_{n,s_0}(x)\downarrow$ newly. At stage $s_0 + 1$, we would take pre-emptive action and enumerate all a_{ℓ,s_0} such that $\ell > x$ and $a_{\ell,s_0} \le \varphi_n(x)$ into A. Then $a_{x+1,s_0+1} \ge \varphi_n(x)$, and hence the only elements of $\bar{A}_t$ below $\varphi_n(x)$ for $t \ge s_0 + 1$

are drawn from $\{a_{0,s_0+1}, \ldots, a_{x,s_0+1}\}$. So for the use of $\Phi_x^{A\restriction\varphi_n(x)}(x)$ to change at stage $s > s_0 + 1$, it must be because one or more of these $x+1$ elements were enumerated into A at stage s. Thus after stage s_0+1, there are at most $x+1$ stages that change A below the use of the the computation we are interested in. As $\varphi_{n,s_0}(x)\downarrow$ newly, such changes only occur at stage s_0, s_0+1 or these at most $(x+1)$-many stages.

So $f(x,s) \neq f(x,s+1)$ for at most $g(x) \coloneqq 2(x+3)(x+1)$-many stages, since for each of the $(x+1)$-many potential uses – given by the $n \leq x$ with $\varphi_n(x)\downarrow$ – there are $(x+3)$-many stages where enumerations below this potential use may occur and each such enumeration may lead to at most two changes (divergence, then reconvergence at some later stage).

Therefore f witnesses that A^b is ω-c.e. with computable bound g. ■

Thus A is the desired bounded low and complete c.e. set. □

As an aside, we have shown that we can construct a set that is bounded low and *Turing* complete, that is, $A \equiv_T \varnothing'$. We cannot improve this so that $A \equiv_{bT} \varnothing'$, since then $A^b \equiv_{bT} \varnothing^b \equiv_1 \varnothing' \equiv_{bT} A$ and this is impossible since $A^b \not\leq_{bT} A$.

The above proof also gives us some new information on the structure of the bounded jump:

Corollary 5.7.
There are c.e. sets A and B such that A is bounded low and B is bounded high, but A and B have the same Turing degree.

Proof:
Take A as above and $B \coloneqq \varnothing^b$. Then the result follows immediately from the completeness of A, the definition of bounded high and the fact that $\varnothing' \equiv_1 \varnothing^b$. □

5.4 Relativized Characterization of Bounded Turing Reductions

Recall that when we are working relative to some set B, it is extremely useful to be able to characterize sets A that are reducible to $B^{(n)}$. In the typical Turing reduction, such as characterization is well-known, provided by the relativized form of Post's Theorem, which relates such sets to the arithmetic hierarchy. That is, $A \leq_T B^{(n)}$ exactly when A is Δ_{n+1}^B.

A non-relativized version of such a characterization for the bounded Turing degrees was proven in Anderson and Csima [2], which relates reductions to the Ershov hierarchy, which consists of the ω^k-c.e. sets. Since we will be working with such sets and attempting to relativize this result, it is worth first recalling the definition of such sets.

In light of Section 5.2, we will dispense with the system of notation for the computable ordinals involved in such sets and work more informally. Recall the definition of α-c.e. we have from Section 5.1:

Definition 5.4 (α-c.e.)**.**
Let $\alpha \geq 1$ be a computable ordinal. We say that a set A is *α-c.e.* (*$\alpha - 1$-c.e.* when α is finite) if there is a computable function $f : \omega \times \omega \to \{0,1\} \times \alpha$ such that for all x:

- $\lim_{s\to\infty} f_0(x,s) = A(x)$ and $f_0(x,0) = 0$,
- if $f_0(x,s+1) \neq f_0(x,s)$, then $f_1(x,s+1) < f_1(x,s+1)$, and
- $f_1(x,s+1) \leq f_1(x,s)$.

(Recall that f_0 and f_1 are the projections of f onto the first and second coordinate, respectively.)

This definition reduces to a simpler one when $\alpha = \omega$, and that this definition is equivalent to the normal definition of ω-c.e. can be seen with not much effort. We will use this usual definition to motivate the first result, and return to the more general definition once we have explored relativizing this simpler case. The change in index when α is finite is so that 1-c.e. aligns with our usual notion of c.e., 2-c.e. with the difference of c.e. sets, and so on. However, for the most part, we will have $\alpha \geq \omega$, so this notational irregularity will not matter.

We now begin the results proper. The first generalizes the familiar result that a set A is bT-below $\varnothing'$ if and only if A is ω-c.e.

Theorem 5.8.
Let A and B be sets. The following are equivalent:

(i) $A \leq_{bT} B^b$

(ii) There exists a function $f \leq_T B$ via Γ such that $\lim_{s\to\infty} f(x,s) = A(x)$ and $f(x,0) = 0$ for all x, a computable function g such that $|\{s \mid f(x,s+1) \neq f(x,s)\}| \leq g(x)$ for all x, and a computable function $\hat{h}$ that is monotone increasing in the second coordinate such that $|\{s \mid \hat{h}(x,s) \neq \hat{h}(x,s+1)\}| \leq g(x)$ for all x and $u_\Gamma^B(x,s) \leq h(x) \coloneqq \lim_{t\to\infty} \hat{h}(x,t)$ for all x and all s.

Proof:

Suppose $A \leq_{bT} B^b$ via Γ with use bound p. For each s, define $(B^b)_s \coloneqq \{x \mid (\exists i < x)[\varphi_{i,s}(x)\downarrow \wedge \Phi_{x,s}^{B \upharpoonright \varphi_{i,s}(x)}(x)\downarrow]\}$. Note that $\{(B^b)_s\}_{s\in\omega}$ is uniformly computable from B.

Define $f(x,s) \coloneqq \begin{cases} \Gamma_s^{(B^b)_s}(x), & \text{if } \Gamma_s^{(B^b)_s}(x)\downarrow \text{ with use less than } p(x) \text{ and } s \neq 0, \\ 0, & \text{if } s = 0, \\ f(x,s-1), & \text{otherwise.} \end{cases}$.

Notice that f is computable from B since $\{(B^b)_s\}_{s\in\omega}$ is uniformly computable from B. Also, $\lim_{s\to\infty}(B^b)_s = B^b$, so for large enough s, $f(x,s) = \Gamma_s^{(B^b)_s}(x)\downarrow = \Gamma^{B^b}(x) = A(x)$.

To compute $f(x,s)$, we first compute $\{(B^b)_t \upharpoonright p(x) \mid t \leq s\}$ from B. Once we know these initial segments, we can compute $f(x,s)$ according to the definition. To compute $(B^b)_t$, we require at most $\max\{\varphi_{i,t}(y)\downarrow \mid i < y \leq p(x)\}$-many bits of B, and so to compute $f(x,s)$, we need $\hat{h}(x,s) \coloneqq \max\{\varphi_{i,t}(y)\downarrow \mid i < y \leq p(x) \wedge t \leq s\}$-many bits of B. It is clear that $\hat{h}$ is computable and monotone increasing in s and possibly changes only when some $\varphi_i(y)$ newly converges for i and y as above, i.e. at most $(p(x)^2 + p(x))/2$-many times.

Also, $f(x,0) = 0$ by definition. Finally, $|\{s \mid f(x,s+1) \neq f(x,s)\}| \leq p(x) + 1$ since the approximation can only change at most $p(x)$-many times for each use change, plus one more for the initial convergence (which does not require a use change). The use can only change at most $p(x)$-many times since once an element enters some $(B^b)_t$, it cannot leave at any later stage. Hence taking $g(x) \coloneqq (p(x)^2 + p(x))/2$ will work, as this exceeds $p(x) + 1$ and satisfies the requirements.

It now remains to show the converse statement, which is a more involved argument. So, suppose f, Γ, g and $\hat{h}$ are as given. We begin with a preliminary definition.

Define the injective computable function $j(x,n)$ as follows:

$$\varphi_{j(x,n)}(y) \coloneqq \begin{cases} \hat{h}(x,t), & \text{where } t \text{ is least such that } |\{\hat{h}(x,s) \mid s \leq t\}| = n+1, \\ \uparrow, & \text{if no such } t \text{ exists.} \end{cases}$$

Then for a fixed x, $\varphi_{j(x,n)}(y)\downarrow$ implies that $n \leq g(x)$, and if n is the largest such, then $\varphi_{j(x,n)}(y)\downarrow = h(x)$.

Let X be an unknown oracle. We define a function $v^X(x,n,s)$ that is computable in X, which will be defined by recursion in n and s as follows:

$$v^X(x,n,s) \coloneqq \begin{cases} v^X(x,n,s-1), & \text{if } v^X(x,n,s-1) \neq -1, \\ \Gamma_s^{X \upharpoonright \hat{h}(x,s)}(x,t), & \text{if } (v^X(x,n,s-1) = -1 \vee s = 0) \wedge \\ & \quad t \leq s \text{ is largest such that } z \coloneqq \Gamma_s^{X \upharpoonright \hat{h}(x,s)}(x,t)\downarrow \wedge \\ & \quad (v^X(x,n-1,s) \notin \{-1,z\} \vee n = 0), \\ -1, & \text{if no such } t \text{ exists.} \end{cases}$$

$v^X(x,n,s)$ records the value of the computation we are interested in after it has changed n times by stage s, or -1 if it changes fewer than n times. This information will be useful as we attempt to build a bounded reduction from A to B^b.

We can now define the injective computable function $\ell(x,n,c)$ where $x,n \in \omega$ and $c \in \{0,1\}$ as follows:

$$\Phi^X_{\ell(x,n,c)}(y) \coloneqq \begin{cases} 0, & \text{if } (\exists s)[v^X(x,n,s) \neq -1 \wedge ((c=0) \vee (c = 1 \wedge v^X(x,n,s) = 1))], \\ \uparrow, & \text{otherwise.} \end{cases}$$

and then use the Padding Lemma (see Soare [28]) to ensure that $\ell(x,n,c) \geq \max\{j(x,i) \mid i \leq g(x)\}$ for all x,n,c as above.

Notice that when $X = B$, $v^B(x,n,s)$ behaves as follows:

- For fixed x and s, a sequence like $v^B(x,0,s) \neq v^B(x,1,s) \neq \cdots \neq v^B(x,n_s,s)$ (none of which are -1) witnesses the existence of $n_s + 1$ values for t such that $\Gamma_s^{B \upharpoonright \hat{h}(x,s)}(x,t)$ converge and disagree in sequence. That is, $n_s + 1$ values $t_0, \ldots, t_{n_s}$ such that $\Gamma_s^{B \upharpoonright \hat{h}(x,s)}(x,t_i)\downarrow \neq \Gamma_s^{B \upharpoonright \hat{h}(x,s)}(x,t_{i+1})\downarrow$ for all $i < n_s$. Hence, by choice of Γ, $\Gamma^B = f$ and so this witnesses $n_s + 1$ values for t where $f(x,t)$ disagree in sequence, i.e. $f(x,t)$ changes at least n_s-many times.

  By assumption, $|\{t \mid f(x,t) \neq f(x,t+1)\}| \leq g(x)$, and so $n_s \leq g(x)$. Hence $v^B(x,n,s) \neq -1$ implies that $n \leq g(x)$.

- If $v^B(x,n,s) \neq -1$, then $v^B(x,n,t) = v^B(x,n,s)$ for all $t \geq s$. Hence, when we compute $\Phi^B_{\ell(x,n,c)}$, if we find some s such that $v^B(x,n,s) \neq -1$, then we know that for any $t \geq s$ we will have the same answer and can thus stop if and when we find such an s. That is, there is no need to continue the search if the first s we find with $v^B(x,n,s) \neq -1$ does not satisfy the remainder of the condition.

- Choose s_1 large enough so that $f(x,s_1) = f(x,t) = A(x)$ for all $t \geq s_1$ and $\hat{h}(x,t) = \hat{h}(x,s_1) = h(x)$ for all $t \geq s_1$. Next, choose $s \geq s_1$ such that $\Gamma^B_s(x,s_1)\downarrow$, which exists since $\Gamma^B = f$ is total. Then at stage s, the largest n such that $v^B(x,n,s) \neq -1$ will have $v^B(x,n,s) = f(x,s_1) = A(x)$, and since no larger t exists where $f(x,t)$ differs from this (by choice of s_1), we will have that $v^B(x,n+1,\hat{s}) = -1$ for all $\hat{s}$.
- Hence, by the above, to decide if $x \in A$, we need to determine the largest n such that there is some s where $v^B(x,n,s) \neq -1$ and then we will know that $x \in A$ if and only if $v^B(x,n,s) = 1$. By the first point, if such an n exists, it must be no larger than $g(x)$.

So, by the definition of $\Phi^B_{\ell(x,n,c)}$ and the previous remarks, we can compute if $x \in A$ by finding the largest $n \leq g(x)$ such that $\Phi^B_{\ell(x,n,c)}(y)\downarrow$ and then determining if $\Phi^B_{\ell(x,n,1)}\downarrow$ for that largest n. If so, $x \in A$, and if not, $x \notin A$. (Here y can be any value.)

To compute $\Phi^B_{\ell(x,n,c)}(y)$ we need to be able to compute $v^B(x,n,s)$ for all s, and a quick inspection of the definition of $v^B(x,n,s)$ shows that this can be computed using only $B \upharpoonright \hat{h}(x,s)$. By the assumptions on $\hat{h}$, it will suffice to use $B \upharpoonright h(x)$, since $\hat{h}(x,s)$ monotonically approaches $h(x)$ from below.

By the definition of $\varphi_{j(x,i)}$, we know that $\max\{\varphi_{j(x,i)}(y)\downarrow \mid i \leq g(x)\} = h(x)$ and $\ell(x,n,c) \geq \max\{j(x,i) \mid i \leq g(x)\}$ by assumption, so we have that

$$\begin{aligned}\Phi^B_{\ell(x,n,c)}(y)\downarrow &\Leftrightarrow \Phi^{B \upharpoonright\upharpoonright h(x)}_{\ell(x,n,c)}(\ell(x,n,c))\downarrow \\ &\Leftrightarrow (\exists j \leq \ell(x,n,c))[\varphi_j(\ell(x,n,c)\downarrow \wedge \Phi^{B\upharpoonright\varphi_j(\ell(x,n,c))}_{\ell(x,n,c)}(\ell(x,n,c))\downarrow] \\ &\Leftrightarrow \ell(x,n,c) \in B^b.\end{aligned}$$

So, to decide if x is in A, we find the largest $n \leq g(x)$ such that $\ell(x,n,0) \in B^b$ and then we will have $x \in A$ if and only if $\ell(x,n,1) \in B^b$ for that n. As we need only $B^b \upharpoonright\upharpoonright \max\{\ell(x,n,c) \mid n \leq g(x) \wedge c \in \{0,1\}\}$, we get that $A \leq_{bT} B^b$, as desired. $\square$

In fact, $A \leq_{tt} B^b$, since we know in advance that we only need to determine the membership of $\{\ell(x,n,c) \mid n \leq g(x) \wedge c \in \{0,1\}\}$ in B^b, and since ℓ is computable, we can determine these locations ahead of time. Then, we find the largest n such that $\ell(x,n,0) \in B^b$ and check if $\ell(x,n,1) \in B^b$, which works for any oracle (not just B^b) once we add a provision that if we cannot find such an n, we declare that $x \notin A$.

Note also that a careful examination of the previous theorem shows that we can effectively pass between the different indices for A. That is, if $A \leq_{bT} B^b$ via Φ_e and φ_i, and $f = \varphi_{j_1}$,

$g = \varphi_{j_2}$, $f \leq_T B$ via Φ_{j_3} and $\hat{h} = \varphi_{j_4}$, then we can pass effectively from $\langle e, i \rangle$ to $\langle j_1, j_2, j_3, j_4 \rangle$. This will prove useful later on, when we are given $C \leq_{bT} D^b$, say, and wish to (effectively) produce the respective f, g, Γ and $\hat{h}$.

Finally, it does not take much effort to see that when $B = \varnothing$, we can take $\hat{h}$ to be any computable function and recover that $A \leq_{bT} \varnothing^b \equiv_1 \varnothing'$ exactly when A is ω-c.e. Because of this, we can now view the second of the two equivalent statements as a *definition* of how to relativize the notion of ω-c.e. such that it characterizes when a set A is bT-below B^b. Since this definition is different from how we might expect to relativize the definition of ω-c.e. to make it suitable for a bounded setting, we refer to this as A being ω-b.c.e. in B instead.

This motivates the following definition, where we extend this notion to all infinite computable ordinals:

Definition 5.5 (α-b.c.e. in B)**.**
For any computable ordinal $\alpha \geq \omega$ and any set B, we say that A is *α-b.c.e. in B* if

- There exists $f \leq_T B$ via Γ such that $f : \omega \times \omega \to \{0, 1\} \times \alpha$ and
  - for all x, $\lim_{s \to \infty} f_0(x, s) = A(x)$ and $f_0(x, 0) = 0$,
  - for all x and all s, $f_0(x, s) \neq f_0(x, s+1)$ implies $f_1(x, s) > f_1(x, s+1)$, and
  - for all x and all s, $f_1(x, s) \geq f_1(x, s+1)$
- There exists a computable $g : \omega \times \omega \to \alpha$ such that for all x and all s, there is some n such that $f(x, s) + n = g(x, s)$. (Note that n need not be computable from x and s, since this would imply that f is computable.)
- There is a computable function $\hat{h} : \omega \times \omega \to D$, where D is the set of total computable functions from the set of limit ordinals below α to $(\omega \times \omega) \cup \{\text{None}\}$ such that the following hold:
  - For all x and all s, if there is some $n \in \omega$ such that $f_1(x, s) = \beta + n$ for some limit ordinal β, then $\hat{h}(x, s)(\beta) \neq \text{None}$ and $u^B_\Gamma(x, s) \leq h_\beta(x) \coloneqq \lim_{t \geq s} \hat{h}_0(x, t)(\beta)$.
  - For all x, all s, and all limit ordinals β, if $\hat{h}(x, s)(\beta) \neq \text{None}$, then $\hat{h}(x, s+1)(\beta) \neq \text{None}$. Furthermore, in this case we have that $\hat{h}_0(x, s)(\beta) \leq \hat{h}_0(x, s+1)(\beta)$ and $\hat{h}_1(x, s)(\beta) \geq \hat{h}_1(x, s+1)(\beta)$, and if the former inequality is strict, then so is the latter.

For brevity, we will sometimes say that $\hat{h}(x,s)(\beta)$ "exists" when it is not None, and "does not exist" if it is None. Note that since the underlying function is still computable, we can effectively determine if $\hat{h}(x,s)(\beta)$ exists or not; we do not need to rely on a computation converging or not.

Note that the definition above has the following properties:

- If $B \equiv_T \varnothing$, then A is α-c.e. in B if and only if A is α-c.e.
- When $\alpha = \omega$, this reduces to the previous definition, as expected.
- We can equivalently use g instead of f in the definition of $\hat{h}$, since by assumption we know that for all x and all s, there is some $n \in \omega$ such that $f(x,s) + n = g(x,s)$, so $f(x,s) = \beta + n_1$ for some $n_1 \in \omega$ if and only if $g(x,s) = \beta + n_2$ for some $n_2 \in \omega$.

We now establish that the definition above has the properties that we expect of it. These proofs are heavily based upon those found in Anderson and Csima [2] where the non-relativized versions are proven; i.e. when $B \equiv_T \varnothing$. In many of these proofs, we need to perform ordinal arithmetic using commutative ordinal addition, denoted $+_c$, defined in [2]. Informally, we write the ordinal in Cantor normal form and then group like terms, as though we were symbolically adding two polynomials with indeterminate ω.

Theorem 5.9.
Let $k > 0$ and A and C be sets with $A \leq_{bT} C$. If C is ω^k-b.c.e. in a set B, then so is A.

Proof:
Let Ψ and p witness that $A \leq_{bT} C$, and f, Γ, g and $\hat{h}$ witness that C is ω^k-b.c.e. in B. We will define $\tilde{f}$, $\tilde{g}$, and $\tilde{\hat{h}}$ to witness that A is ω^k-b.c.e. in B. (The analogue to Γ will be defined implicitly.)

Fix n and s. Let σ_s^n be the string of length $p(n)+1$ defined by $\sigma_s^n(i) \coloneqq f_n(i,s)$ for $i \leq p(n)$. It is clear that $\{\sigma_s^n\}_{n,s\in\omega}$ is uniformly computable from B.

Let $\delta_s \coloneqq f_1(0,s) +_c \cdots +_c f_1(p(n),s) +_c u(f_1(0,s) +_c \cdots +_c f_1(p(n),s))$ and $\delta_0 \coloneqq f_1(0,0) +_c \cdots +_c f_1(p(n),0) +_c u(f_1(0,0) +_c \cdots +_c f_1(p(n),0))$.

Let

$$\tilde{f}(n,s) \coloneqq \begin{cases} \langle \Psi_s^{\sigma_s^n}(n), \delta_s \rangle, & \text{if } \Psi_s^{\sigma_s^n}(n)\downarrow \wedge s \neq 0, \\ \langle 0, \delta_0 +_c 1 \rangle, & \text{if } s = 0, \\ \langle \tilde{f}_0(n,s-1), \delta_s +_c 1 \rangle, & \text{otherwise.} \end{cases}$$

It is clear that defining $\tilde{g}(n,s) \coloneqq g(0,s) +_c \cdots +_c g(p(n),s) +_c u(g(0,s) +_c \cdots +_c g(p(n),s)) +_c 1$ will work with this $\tilde{f}$.

Before we define $\tilde{h}$, we will first verify that $\tilde{f}$ has the properties that we require. The second and third properties are easy and use ordinal properties as in the proofs we are mimicking, while the fourth property is true by definition of $\tilde{f}$. So it remains to show the first property, that $\lim_{s\to\infty} \tilde{f}_0(x,s) = A(x)$ for all x.

Fix n and choose $s > 0$ so that for all $t \geq s$ we have $f_0(i,t) = C(i)$ for all $i \leq p(n)$ and so that $\Psi_s^{C \Uparrow p(n)}(n)\downarrow = A(n)$. It is possible to find an s since $\lim_{s\to\infty} f_0(i,t) = C(i)$ for all i by choice of f and $\Psi^C = A$ with use bound p. Then for $t \geq s$, we have $\sigma_t^n = C \Uparrow p(n)$ and hence $\tilde{f}_0(i,t)$ is given by $\Psi_t^{\sigma_t^n}(n) = \Psi_t^{C\Uparrow p(n)}(n) = A(n)$, since $\Psi_s^{C\Uparrow p(n)}(n)\downarrow$ by choice of s and $s \neq 0$.

Now we define $\tilde{h}$. Fix n, and let s be the first stage where the limit ordinal β is encountered in $\tilde{f}(n,\cdot)$, which can be determined from the computable function $\tilde{g}(n,\cdot)$ instead. We know that $\tilde{g}(n,t)$ is defined as $g(0,t) +_c \cdots +_c g(p(n),t) +_c u(g(0,t) +_c \cdots +_c g(p(n),t)) +_c 1$ for all t. Let the limit ordinal part of $g(i,t)$ be denoted by $\beta_{i,t}$ for $t \leq s$. Then the collection of $\{\beta_{i,t}\}_{i\leq p(n), t\leq s}$ is finite and since $\hat{h}(i,t)(\beta_{i,t})$ must exist, $\hat{h}(i,t')(\beta_{i,t})$ must exist for all $t' \geq s$.

So at all stages $t' \geq s$, we define

$$\tilde{\tilde{h}}(n,t')(\beta) := \left\langle \max_{i\leq p(n),t\leq s} \left\{\hat{h}_0(i,t')(\beta_{i,t})\right\}, \sum_{i\leq p(n),t\leq s} \hat{h}_1(i,t')(\beta_{i,t}) \right\rangle.$$

We verify that $\tilde{\tilde{h}}$ has the desired properties. First, $\tilde{\tilde{h}}_0(n,\cdot)(\beta)$ is non-decreasing (resp. $\tilde{\tilde{h}}_1(n,\cdot)(\beta)$ is non-increasing) because its constituents consist of $\hat{h}_0(i,t')(\beta_{i,t})$ (respectively $\hat{h}_1(i,t')(\beta_{i,t})$), and these are non-decreasing (resp. non-increasing) by choice of $\hat{h}$. Also, if $\tilde{\tilde{h}}_0(n,t'+1)(\beta) > \tilde{\tilde{h}}_0(n,t')$, then this must be because some $\hat{h}_0(i,t'+1)(\beta_{i,t}) > \hat{h}_1(i,t')(\beta_{i,t})$, so $\hat{h}_1(i,t'+1)(\beta_{i,t}) < \hat{h}_1(i,t')(\beta_{i,t})$ and thus $\tilde{\tilde{h}}_1(n,t'+1)(\beta) < \tilde{\tilde{h}}_1(n,t')(\beta)$.

We now just need to verify one final property of $\tilde{\tilde{h}}$, namely that $u_{\tilde{\Gamma}}^B(n,t') \leq \tilde{h}_\beta(n)$ whenever the limit ordinal part of $\tilde{f}(n,t')$ is β (equivalently, whenever the limit ordinal part of $\tilde{g}(n,t')$ is β). Note that $\tilde{h}_\beta(n) = \max_{i\leq p(n),t\leq s}\{h_{\beta_{i,t}}(i)\}$ and so with $h_\beta(n)$-many bits of B we can determine (recursively) $\tilde{f}(n,0),\ldots,\tilde{f}(n,s),\ldots,\tilde{f}(n,t')$.

As an aside, we need to consider $\beta_{i,t}$ for all $t \leq s$ since the definition of $\tilde{f}$ is recursive, and so we may need to compute $\tilde{f}(n,t'-1),\tilde{f}(n,t'-2),\ldots$ to obtain the value of $\tilde{f}(n,t')$. However, since we first see the limit ordinal β at stage s, we know that $t' \geq s$ and between these two stages we do not see any new limit ordinals. So it suffices to have enough of B to compute the $\beta_{i,t}$ that contribute to the limit ordinals we see before β in addition to β itself. $\square$

Theorem 5.10.
Let $k > 0$ and A be ω^k-b.c.e. in some set B. Then A^b is ω^{k+1}-b.c.e. in B.

Proof:
Let f, Γ, g and $\hat{h}$ witness that A is ω^k-b.c.e. in B. As before, we will define $\tilde{f}$, $\tilde{g}$, $\tilde{\hat{h}}$ (and implicitly $\tilde{\Gamma}$) to witness that A^b is ω^{k+1}-b.c.e. in B.

Fix n and s. Let σ_s^n be the string of length $m(n,s)$ (defined momentarily) such that $\sigma_s^n(i) \coloneqq f_s(i,s)$ for all $i < m(n,s)$.

We define $m(n,0) \coloneqq 0$ and $m(n,s+1) \coloneqq \max\{\varphi_{i,s+1}(n)\downarrow \mid i \leq n\}$. Note that $m(n,s)$ is non-decreasing in s for a fixed n. We will also have a function ℓ to keep track of when this changes. Define $\ell(n,0) \coloneqq n+1$, and if $s+1$ is a stage where $m(n,s+1) > m(n,s)$, then we set $\ell(n,s+1) \coloneqq \ell(n,s) - 1$. It is nor hard to see that ℓ is well-defined, since $m(n,\cdot)$ can increase at most $n+1$ times, one for each $i \leq n$. Since $m(n,s)$ is computable, $\{\sigma_s^n\}_{s,n\in\omega}$ is uniformly computable from B.

Let

$$r(n,s) \coloneqq \omega^k \cdot \ell(n,s) +_c f_1(0,2) +_c \cdots +_c f_1(m(n,s),s) +_c u(f_1(0,s) +_c \cdots +_c f_1(m(n,s),s)).$$

We can now define $\tilde{f}$. Let $\tilde{f}(n,0) \coloneqq \langle 0, \omega^k \cdot (n+1)\rangle$ and for $s > 0$, we set

$$\tilde{f}(n,s) \coloneqq \begin{cases} \langle 1, r(n,s)+1\rangle, & \text{if } \Phi_{n,s}^{\sigma_s^n}(n)\downarrow, \\ \langle 0, r(n,s)+2\rangle, & \text{otherwise.} \end{cases}$$

Setting

$$\tilde{g}(n,s) \coloneqq \omega^k \cdot \ell(n,s) +_c g(0,s) +_c \cdots +_c g(m(n,s),s) +_c u(g(0,s) +_c \cdots +_c g(m(n,s),s)) +_c 2$$

clearly works by properties of g. By how we have defined $\tilde{f}$, $\tilde{g}$ will be computable since ℓ, m and g are. Before we define $\tilde{\hat{h}}$, we will first verify that $\tilde{f}$ has the properties we require.

First, we must show that $\lim_{s\to\infty} \tilde{f}_0(n,s) = A^b(n)$ for all n. Fix n. If $n \in A^b$, then there is some $i \leq n$ such that $\varphi_i(n)\downarrow$ and $\Phi_n^{A\upharpoonright\varphi_i(n)}(n)\downarrow$. Let s be large enough so that $\varphi_{i,s}(n)\downarrow$. Then for all $t \geq s$, we know $m(n,t) \geq \varphi_i(n)$. Since $\lim_{t\to\infty} f_o(x,t) = A(x)$ for all x, take s even larger so that for all $j \leq \varphi_i(n)$ and all $t \geq s$ we also have $f_0(j,t) = A(t)$, so that $\sigma^n_t \upharpoonright\!\upharpoonright \varphi_i(n) = A \upharpoonright\!\upharpoonright \varphi_i(n)$. Now take s even larger so that $\Phi^{A \upharpoonright\!\upharpoonright \varphi_i(n)}_{n,s}(n)\downarrow$. Then for all $t \geq s$ we will have $\Phi^{\sigma^n_t}_{n,t}(n)\downarrow$ and so $\tilde{f}_0(n,t) = 1$. Thus $\lim_{t\to\infty} \tilde{f}_0(n,t) = 1 = A^b(n)$.

Conversely, suppose $n \notin A^b$. Let s be large enough so that all $\varphi_i(n)$ with $i \leq n$ have converged by stage s if they ever do. Then $m(n,t) = m(n,s)$ for all $t \geq s$; call this limit value m. As before, take s even larger so that for all $j \leq m$ and all $t \geq s$ we also have $f_0(j,t) = A(j)$ so that $\sigma^n_t = A \upharpoonright m$. Then for all $t \geq s$, $\tilde{f}_0(n,t) = 1$ means that $\Phi^{\sigma^n_t}_{n,t}(n)\downarrow$ and so $\Phi^{A\upharpoonright m}_n(n)\downarrow$. Then $m > 0$ and so we must have some $i \leq n$ such that $\varphi_i(n)\downarrow = m$. But then $n \in A^b$, since $\varphi_i(n)\downarrow$ and $\Phi^{A\upharpoonright \varphi_i(n)}_n(n)\downarrow$, a contradiction. So for $t \geq s$, we must have $\tilde{f}_0(n,t) = 0$ and so $\lim_{t\to\infty} \tilde{f}_0(n,t) = 0 = A^b(n)$.

Next, we must show that if $\tilde{f}_0(n,s+1) \neq \tilde{f}_0(n,s)$, then $\tilde{f}_1(n,s+1) < \tilde{f}_1(n,s)$. Since $r(n,s+1) \leq r(n,s)$ for all n and all s, this is clear when $\tilde{f}_1(n,s+1) = 1$ and $\tilde{f}_0(n,s) = 0$. For the other case, notice that if $\Phi^{\sigma^n_s}_{n,s}(n)\downarrow$ but $\Phi^{\sigma^n_{s+1}}_{n,s+1}(n)\uparrow$, then we must have that $\sigma^n_t \neq \sigma^n_{s+1}$. But this can only happen if we have some $i \leq |\sigma^n_s| = m(n,s)$ with $f_0(i,s+1) \neq f_0(i,s)$ and hence by the choice of f, we know that $f_1(i,s+1) < f_1(i,s)$ and thus $r(n,s+1) < r(n,s)$. Then by ordinal arithmetic, this means that $r(n,s+1)+2 < r(n,s)+1$. So we get the desired result.

Finally, we must show that $\tilde{f}_1(n,s+1) \leq \tilde{f}_1(n,s)$. Since $r(n,s+1) \leq r(n,s)$ for all n and all s, this is obvious from the definition unless $\tilde{f}_0(n,s) = 0$ and $\tilde{f}_0(n,s+1) = 1$. But then $\tilde{f}_0(n,s+1) \neq \tilde{f}_0(n,s)$ and so $\tilde{f}_1(n,s+1) < \tilde{f}_1(n,s)$ by the above.

Now we define $\tilde{\hat{h}}$. Fix n. Let s be the first stage where the limit ordinal β is encountered in $\tilde{f}(n,\cdot)$ (again, computed using $\tilde{g}$). We know that

$$\tilde{g}(n,s) \coloneqq \omega^k \cdot \ell(n,s) +_c g(0,s) +_c \cdots +_c g(m(n,s),s) +_c u(g(0,s) +_c \cdots +_c g(m(n,s),s)).$$

So let the limit ordinal part of $g(i,s)$ be denoted as β_i. Note that ℓ and m are computable and constant for a given β, as ℓ is clearly constant, and m changing implies ℓ changes. So at all stages $t \geq s$, we define

$$\tilde{\hat{h}}(n,t)(\beta) \coloneqq \left\langle \max_{i \leq m(n,s)} \{\hat{h}_0(i,t)(\beta_i)\}, \sum_{i \leq m(n,s)} \hat{h}_1(i,t)(\beta_i) \right\rangle.$$

As before, some thought shows that $\tilde{\hat{h}}_0(n,\cdot)(\beta)$ is non-decreasing and $\tilde{\hat{h}}_1(n,\cdot)$ is non-increasing.

So we just need to ensure that $u^B_\Gamma(n,t) \leq \tilde{h}_\beta(n)$ whenever the limit ordinal part of $\tilde{f}(n,t)$ is β. To compute this, we need to compute $f_0(i,t)$ for all $i \leq m(n,t)$. If s is as above, then we know $m(n,s) = m(n,t)$, since if m changes, then ℓ does and so we could not have the same β. We have $\tilde{h}_\beta(n) = \max_{i\le m(n,s)} \tilde{h}_{\beta_i}(n)$, so this will work. (Again, we can use $\tilde{h}_{\beta_i}$ because we do not see any new limit ordinals for $f_1(i,\cdot)$ between stage s – where we chose β_i – and stage t, since this would cause β to change. So this same β_i will work at stage t.) □

Theorem 5.11.
Let $k > 0$ and A be ω^k-b.c.e. in some set B. Then $A \le_{bT} B^{kb}$. Furthermore, given (the codes for) the functional Γ and the computable functions g and $\hat{h}$ that witness A being ω^k-b.c.e. in B, we can effectively pass to (codes for) the functional Ψ and the computable function p that witness $A \le_{bT} B^{kb}$.

Proof:
We proceed by induction on k. We have already proven this for $k = 1$ in Theorem 5.8. So assume this holds for k and we wish to show it holds true for $k+1$. Let f, Γ, g and $\hat{h}$ witness that A is ω^{k+1}-b.c.e. in B.

Uniformly in i and n perform the following sequence of steps: Search for the first s, if it exists, where $g(n,s) = \omega^k \cdot i + \alpha$ for some $\alpha < \omega^k$. If no such s exists, we will have $e_i(n)\uparrow$. If such an s exists, however, we call it s_0. Then we build $\tilde{f}$, $\tilde{g}$ and $\tilde{\hat{h}}$ as follows. Note that $\tilde{\Gamma}$ is built implicitly.

$$\tilde{f}(m,s) \coloneqq \begin{cases} \langle j, \alpha +_c 1\rangle, & \text{if } m = n \text{ and } f(m, s_0 + s) = \langle j, \omega^k \cdot i + \alpha\rangle \text{ for some } \alpha < \omega^k,\\ \langle 0,0\rangle, & \text{otherwise.}\end{cases}$$

$$\tilde{g}(m,s) \coloneqq \begin{cases} \alpha +_c 1, & \text{if } m = n \text{ and } g(m, s_0 + s) = \omega^k \cdot i + \alpha \text{ for some } \alpha < \omega^k,\\ 0, & \text{otherwise.}\end{cases}$$

$$\tilde{\hat{h}}(\beta) \coloneqq \begin{cases} \hat{h}(m, s_0+s)(\omega^k \cdot i + \beta), & \text{if } m = n \text{ and } \hat{h}(m, s_0+s)(\omega^k \cdot i + \beta) \text{ exists},\\ \langle 0,0\rangle, & \text{if } g(m, s_0+s) < \omega^k \cdot i \\ & \quad \text{and } \hat{h}(m, s_0+s)(\omega^k \cdot i) \text{ does not exist,}\\ \uparrow, & \text{otherwise.}\end{cases}$$

We claim that these will witness a set $A_{i,n}$ that is ω^k-b.c.e. in B, namely $\lim_{s\to\infty} \tilde{f}(x,s)$; it is not hard to see that this limit exists for all x. The properties of $\tilde{f}$, $\tilde{g}$ and $\tilde{\hat{h}}$ are easily checked, since all we are doing is ignoring all but $x = n$, as $A_{i,n}(m) = 0$ for $m \ne n$, and then accelerating our existing approximation so that it starts at stage s_0. We also abandon it and set $A_{i,n} = 0$ if the approximation for $A(n)$ ever starts to use ordinals below $\omega^k \cdot i$, hence we add 1 to α so that we can do this. We also must add a caveat to

$\tilde{\hat{h}}$ so that if it encounters the limit ordinal 0 for the reason of abandonment, it uses the value $\langle 0,0\rangle$. However, it is possible that $\tilde{\hat{h}}$ has encountered 0 before – if $\hat{h}$ encounters $\omega^k \cdot i + 0$ – so we only do this if this has not occurred, and hence cannot occur since the approximation will have gone below $\omega^k \cdot i$ when abandonment occurs.

Apply the induction hypotheses to $\tilde{f}$, $\tilde{\Gamma}$, $\tilde{g}$ and $\tilde{\hat{h}}$ to get that $A_{i,n} \leq_{bT} B^{kb}$ via Φ_e and p. Set $e_i(n)\downarrow = \langle e, p(n)\rangle$. We write $e_{i,0}(n)$ and $e_{i,1}(n)$ to refer to these two values respectively.

Define the computable function v by $\varphi_{v(i,n)}(y) \coloneqq e_{i,1}(n)$ and let $u(n) \coloneqq \max\{v(i,n)\}$. This is computable since we only need to consider i at most the one we can compute using $g(n,0) = \omega^k \cdot i_{\max} + \alpha$, since all i must then be below $i_{\max}$. We write i_n to be this $i_{\max}$. Note that $\{i_n\}_{n\in\omega}$ is computable in n.

Now note that if, for a given n, we could determine the least i such that there is an s with $g(n,s) = \omega^k \cdot i + \alpha$ for some $\alpha < \omega^k$, we could use $\Phi^{B^{kb}}_{e_{i,0}(n)}(n)$ with use bound $e_{i,1}(n)$ to compute $A_{i,n}(n)$, and since i is least we would know that this is $A(n)$.

We shall use the following:

Claim 5.11.1: Let A be any set. There is a computable function j such that $\varnothing' \leq_1 A^b$. Furthermore, this choice of j is uniform and does not depend on A.

Proof of Claim: We know $x \in \varnothing'$ if and only if $\varphi_x(x)\downarrow$. Define the functional $\Phi^C_{j(x)}$ as follows:

$$\Phi^C_{j(x)}(y) \coloneqq \begin{cases} 0, & \text{if } \varphi_x(x)\downarrow, \\ \uparrow, & \text{otherwise.} \end{cases}$$

and by the Padding Lemma (see Soare [28]), increase $j(x)$ so that it exceeds e, where φ_e is an index of some fixed total function, say, the constant 0 function. Then

$$j(x) \in A^b \Leftrightarrow (\exists i \leq j(x))[\varphi_i(x)\downarrow \wedge \Phi^{A\upharpoonright\varphi_i(x)}_{j(x)}(x)\downarrow] \tag{5.1}$$
$$\Leftrightarrow (\exists i \leq j(x))[\varphi_i(x)\downarrow \wedge \varphi_x(x)\downarrow] \tag{5.2}$$
$$\Leftrightarrow \varphi_x(x)\downarrow \Leftrightarrow x \in \varnothing'. \tag{5.3}$$

Here we when go from the third line to the second line, we take $i = e$, since $\varphi_e(x)\downarrow = 0$ and $e \leq j(x)$.

Thus $\varnothing' \leq_1 A^b$ via j. As j clearly has no dependence on A, we are done. ■

Let j be as in the lemma and let $x_{i,n}$ denote the bit of $\varnothing'$ that answers the Σ_1-question $(\exists s)(\exists \alpha)[g(n,s) = \omega^k \cdot i + \alpha]$ for $i \leq i_n$.

For all i and all n, we define the injective computable function $r(i,n)$ such that

$$\Phi^C_{r(i,n)}(y) \coloneqq \begin{cases} 0, & \text{if } i \le i_n \wedge e_i(n)\downarrow \wedge \Phi^C_{e_{i,0}(n)}(n)\downarrow = 1, \\ \uparrow, & \text{otherwise.} \end{cases}$$

By the Padding Lemma, we may enforce that $r(i,n) \ge u(n)$ for all $i \le i_n$. Now, for $i \le i_n$, we have that

$$\begin{aligned} r(i,n) \in B^{(k+1)b} &\Leftrightarrow (\exists u \le r(i,n))[\varphi_u(r(i,n))\downarrow \wedge \Phi^{B^{kb} \Uparrow \varphi_u(r(i,n))}_{r(i,n)}(r(i,n))\downarrow] & (5.4)\\ &\Leftrightarrow (\exists \le r(i,n)[\varphi_u(r(i,n))\downarrow \wedge e_i(n)\downarrow \wedge \Phi^{B^{kb} \Uparrow \varphi_u(r(i,n))}_{e_{i,0}(n)}(n)\downarrow = 1] & (5.5)\\ &\Leftrightarrow \Phi^{B^{kb} \Uparrow e_{i,1}(n)}_{e_{i,0}(n)}(n)\downarrow = 1 \Leftrightarrow A_{i,n}(n) = 1. & (5.6) \end{aligned}$$

Here when we go from the second line to the third, we take $u = v(i,n) \le u(n) \le r(i,n)$ so that $\varphi_u(r(i,n))\downarrow = \varphi_{v(i,n)}(r(i,n))\downarrow = e_{i,1}(n)$.

Thus to compute if $n \in A$, we find the least $i \le i_n$ such that $x_{i,n} \in \varnothing'$ by finding the least $i \le i_n$ such that $j(x_{i,n}) \in B^{(k+1)b}$. Then $n \in A$ exactly when $n \in A_{i,n}$, which occurs if and only if $r(i,n) \in B^{(k+1)b}$ by the above.

Formally, we define Ψ^C so that on input n, it finds the least $i \le i_n$ such that $j(x_{i,n}) \in C$ and then halts and outputs $C(r(i,n))$ for this least i. If no such least i exists – although it must when $C = B^{(k+1)b}$ – Ψ^C halts and outputs 0. We have that the use of $\Psi^C(n)$ is at most $\max\{j(x_{i,n}, r(i,n) \mid i \le i_n\}$ which is computable since i_n, r, and j are, and finding $x_{i,n}$ from i and n is. So taking $\hat{p}(n) \coloneqq \max\{j(x_{i,n}, r(i,n) \mid i \le i_n\}$ will work, proving the result. □

Finally, we combine the previous results to prove that this definition provides a characterization of $\le_{bT}$ with respect to the bounded jump.

Theorem 5.12.
Let A and B be sets. Then $A \le_{bT} B^{kb}$ if and only if A is ω^k-b.c.e. in B.

Proof:
We have just proved the reverse direction in Theorem 5.11. To prove the forward direction, we use Theorem 5.9 and Theorem 5.10. We will show that for all $k > 0$, B^{kb} is ω^k-b.c.e. in B. Then if $A \le_{bT} B^{kb}$, we will be able to conclude that A is also ω^k-b.c.e. in B, by Theorem 5.9.

Since we know that a set C^b is ω^{k+1}-b.c.e. in B whenever C is ω^k-b.c.e. in B by Theorem 5.10, it suffices to show the base case of this induction: B^b is ω-b.c.e. in B for any set B. But by our characterization of ω-b.c.e., this amounts to showing that $B^b \leq_{bT} B^b$, since a set C is ω-b.c.e. in B exactly when $C \leq_{bT} B^b$. Since $B^b \leq_{bT} B^b$ trivially, we are done. $\square$

5.5 Non-triviality of the Bounded Jump Hierarchy

We shall use the definition of α-b.c.e. obtained in the previous section to prove the following:

Theorem 5.13.
For any set B, there is a set A such that $B <_{bT} A <_{bT} B^b$. Furthermore, $B^b <_{bT} A^b <_{bT} B^{2b}$.

This result is of interest because it demonstrates that the bounded jump hierarchy is not trivial at any level. This is somewhat different from showing the hierarchy does not *collapse* at any level, which is far easier, since it is immediate once one knows that $B <_{bT} B^b$ for any set B. Instead, we are interested in generating, for an arbitrary B, a set that is strictly between B and B^b. This demonstrates, for example, that the question of bounded jump inversion is non-trivial relative to any B.

Proof:
It is enough to produce a set A such that $B \leq_{bT} A \leq_{bT} B^b$ with $B^b <_{bT} A^b <_{bT} B^{2b}$. To meet $B \leq_{bT} A$, we code B into A via a 1-reduction, and to meet $A \leq_{bT} B^b$, we also build A so that it is ω-b.c.e in B. To meet $B^b <_{bT} A^b <_{bT} B^{2b}$, we aim to satisfy the requirements:

$$\mathbf{R}_{\langle e,i\rangle}: \quad \Phi_e^{B^b \upharpoonright \varphi_i} \neq A^b$$

and

$$\mathbf{Q}_{\langle e,i\rangle}: \quad \Phi_e^{A^b \upharpoonright \varphi_i} \neq C$$

where C is some set that is $(\omega+1)$-b.c.e. in B which we build during the course of the construction.

We code B directly into the odd half of A, and so all coding (in A) happens in the even half. We will arrange it so that witnesses are kept in priority order. That is, since the requirements are arranged according to the priority order $\mathbf{R}_0 > \mathbf{Q}_0 > \mathbf{R}_1 > \mathbf{Q}_1 > \cdots$, at each stage, the witnesses we select will always be in the order $x_0 < y_0 < x_1 < y_1 < \cdots$, where x_i is the witness for $\mathbf{R}_i$ and y_i is the witness for $\mathbf{Q}_i$. (The actual witnesses will be more complicated; in particular, each $\mathbf{R}_i$ will have two witnesses at certain stages.

We will remark more on this as appropriate.) Additionally, each witness for $\mathbf{Q}_i$ will be drawn from $\omega^{[i]}$, and each witness for $\mathbf{R}_i$ will be drawn from $2\omega^{[i]}$. There is an exception in the latter case, due to the "double witnesses" already mentioned.

Using the (Relativized) Recursion Theorem, we fix computable injective functions p and q such that for all z, we control $\Phi_{p(z)}$ and $\varphi_{q(z)}$ and $q(z) < p(z)$. At certain stages, we might demand a new pair of such indices, i.e. $(p(z), q(z))$ for some z. When we do so, we will be interested in making $p(z)$ suitably large. Since p is injective, this is always possible. We shall also enforce that once this occurs, any pairs requested later will have $p(z') > p(z)$. Note that this is stronger than making p strictly increasing and then emitting $(p(0), q(0)), (p(1), q(1)), \ldots$ when queried, since each query can also demand that $p(z)$ is suitably large, and so we might have to skip some values. We refer to this procedure as *querying the witness machine* for a new pair $(p(z), q(z))$.

According to the definitions above, we must produce a pair of computable functions to witness that A is ω-b.c.e. in B, and another pair of computable functions to witness that C is $(\omega+1)$-b.c.e. in B. Since the construction of A itself will use B as an oracle – really, an arbitrary oracle Y so that we can use the Relativized Recursion Theorem (see Soare [28]) – we cannot build these functions simultaneously as we build A. Instead, we will arrange the construction of A so that we keep all of the non-computable portions separate, and hence we can computably follow the broad course of the construction, even though we might not be able to computably determine the exact approximations of A and C at any given stage.

To begin, we will first make some notes about the pairs of computable functions mentioned above, so that we may ease notation when discussing them.

Recall that we must produce a pair of computable functions g and $\hat{h}$ for both A and C. In fact, we must produce a functional Γ for each of A and C as well, but since are allowed to have B as an oracle, they can be defined implicitly during the course of the construction. The difficulty is in defining the *computable* functions, which as noted above, cannot use B and thus must be dealt with explicitly.

We refer to these computable functions are g_A, $\hat{h}_A$ and g_C, $\hat{h}_C$, respectively. Defining g_A and g_C is simple; we may do so now. We will arrange it so that for any x, the approximation of $A(x)$ changes at most x times, hence setting $g_A(x, s) = x$ will work for all s and x. Note that this is all we require for a ω-b.c.e. in B set (examine the definition of α-b.c.e. in B when $\alpha = \omega$ to see this.) Similarly, for g_C, we set

$$g_C(x, s) \coloneqq \begin{cases} \omega, & \text{if } \varphi_{i,s}(x)\uparrow \text{ where } x \in \omega^{[\langle e,i\rangle]} \text{ for some } e, \\ \varphi_{i,s}(x) + 2, & \text{if } \varphi_{i,s}(x)\downarrow \text{ where } i \text{ is as above.} \end{cases}$$

To agree with this, at each stage s, we will check if any $x \leq s$ has $\varphi_{i,s}(x)\downarrow$ for $i \leq s$. If we have not begun changing the approximation of x in C, i.e. we still have that $x \notin C$ with confidence ω, we will say that $x \notin C$ at stage s with confidence $\varphi_{i,s}(x) + 3$. This will not change any of the strategies we employ, since we will only be interested in changing the membership of x in C should $\varphi_i(x)$ converge, and so we may enforce that $x \notin C$ until $\varphi_{i,s}(x)\downarrow$.

The definitions of $\hat{h}_A$ and $\hat{h}_C$ are more complicated, and will be saved for after the construction. However, we first make some general remarks about our strategy for defining them. They need to approach a limit which gives a bound on the number of bits to decide if $x \in A_s$ (or if $y \in C_s$) for any stage s. To do this, we need enough of B to decide the following:

1. Is x (resp. y) ever selected as the witness by stage s by a requirement $\mathbf{R}_i$ (resp. $\mathbf{Q}_i$) that is permitted to use it a witness?
2. If so, is x (resp. y) ever abandoned as the witness for its requirement by stage s once it has been chosen?
3. How many times do we change the approximation of $x \in A$ (resp. $y \in C$) up to stage s while it is the witness?

It is important to keep these points in mind throughout the construction, since they restrict how and why we can take action. We shall return to them after the construction, when we have enough information in order to answer the three questions.

Before the construction proper, we describe the (re-)initialization module for each each requirement:

Initialization:

- $\mathbf{R}_{\langle e,i \rangle}$: Query the witness machine for a new pair of (x, j) such that x is larger than any *computable* convergence seen so far and larger than the current witness of the next highest priority requirement. We declare that x is a *non-coding* witness.
- $\mathbf{Q}_{\langle e,i \rangle}$: Pick some large $y \in \omega^{[\langle e,i \rangle]}$ larger than the current witness of the next highest priority requirement such that y is larger than any *computable* convergence seen so far.

Note that we demand that witnesses are larger than the current witness of the next highest priority requirement so that we maintain the priority ordering of witnesses.

We make one final note about terminology. When we initially request witnesses from the witness machine, the usage of these values is different depending on if they are used for **R**-type requirements or **Q**-type requirements. In the latter case, the witness is used as we expect from other constructions. However, in the former case, we sometimes need to generate a new value (not from the machine) to stand in as the witness for the purpose of bound calculations, etc. We still call this new value a "witness" so that we can simply refer to the collection of the values important for these calculations in a convenient way. This is why we mentioned above the need for *two* witnesses for **R**-type requirements: one is the true witness obtained from the machine and the other is a stand-in that will not actually be used to witness the truth of the requirement per se. We will call the true witness the "non-coding" witness and call the stand-in witness the "coding" witness, since the stand-in witness' only role is to be used to code a value into A^b. If we are working with an arbitrary requirement, the term "witness" refers to the active witness for that requirement, i.e. the coding witness if it exists and the non-coding witness otherwise. However, if we are examining an **R**-type requirement in particular, we shall be sure to distinguish between the two witnesses, should they exist.

Construction:

Stage 0: Let $A_0 \coloneqq \varnothing$ and $C_0 \coloneqq \varnothing$.

Stage $s+1$:

- (Initialize New Requirements): Initialize $\mathbf{R}_s$ and $\mathbf{Q}_s$.
- (Cause Injury):
  * We say that $\mathbf{R}_{\langle e,i\rangle}$ *requires attention for witness* if its current witness x is non-coding, and $\varphi_{i,s}(x)\downarrow$.
  * We say that $\mathbf{R}_{\langle e,i\rangle}$ *requires attention for coding* if its current witness m is coding (with non-coding witness x) and $\Phi_{e,s}^{(B^b)_s \upharpoonright \varphi_{i,s}(x)}(x)\downarrow = (A^b)_s(x)$.
  * We say that the requirement $\mathbf{Q}_{\langle e,i\rangle}$ *requires attention* if it has current witness y and $\varphi_{i,s}(y)\downarrow$ and $\varphi_{\ell,s}(z)\downarrow$ newly since y was chosen as the witness, for some $\ell < z \leq \varphi_i(y)$.

  We say that the highest priority requirement that requires attention *receives attention* at stage $s+1$ and we do the following:

  * If $\mathbf{R}_{\langle e,i\rangle}$ requires attention for witness, choose some fresh large $m \in 2\omega^{[\langle e,i\rangle]}$ such that $m > x$ and $m > \varphi_{i,s}(x) + 3$. We say that this requirement now has coding witness m with non-coding witness x. Since x was the first half of some pair produced by the witness machine, $x = p(z)$

for some z, there is some $j = q(z)$ with $j < x$ such that we control φ_j. Declare that $\varphi_j(x)\downarrow = m+1$. Also, since we control Φ_x, declare that $\Phi_x^Y(x)\downarrow$ if and only if $m \in Y$.

  * If $\mathbf{R}_{\langle e,i\rangle}$ requires attention for coding, toggle m in A, so that $A_s(m) \neq A_{s+1}(m)$.
  * If $\mathbf{Q}_{\langle e,i\rangle}$ requires attention, do nothing, except for what follows.

In all cases, re-initialize all lower priority requirements, forcing them to abandon their current witnesses and choose new witnesses. We re-initialize requirements in priority order, so that the witnesses they choose maintain this priority order when re-initialization is complete. In the first case, note that the witnesses will now be chosen larger than m (which is larger than the previous witness for this requirement, x). In the third case, witnesses will be chosen larger than the newly convergent $\varphi_{\ell,s}(z)$. (In the second case, toggling m in A simply causes injury to lower priority requirements' approximation of A, and so the witnesses of lower priority requirements need to be re-chosen for this reason alone.)

We say that all lower priority requirements are *injured* at stage $s+1$.

We have the following important restriction that will prove useful later on: When a requirement $\mathbf{R}_i$ is forced to abandon a coding witness m, we will enforce that it withdraws m from A if needed. We will need to be mindful of this when we count the number of changes the approximation of m in A has.

- (Tend to $\mathbf{Q}$ Requirements): For every $\mathbf{Q}_{\langle e,i\rangle}$ that has been initialized at or before stage s, we tend to it as follows: If the requirement has witness y, $\varphi_{i,s}(y)\downarrow$ and $\Phi_e^{A^b \upharpoonright \varphi_i(y)}(y)[s]\downarrow = C_s(y)$, toggle y in C, and set the new confidence to be 1 lower than before. (If the previous confidence was ω, then we set the new confidence to be $2\varphi_{i,s}(y)+1$. This can only happen if this is the first stage where $\varphi_{i,s}(y)$ converges; see the remark made while defining g_C above.)

Set $A \coloneqq \lim_{s\to\infty} A_s$ and $C \coloneqq \lim_{s\to\infty} C_s$. This concludes the construction.

To show that A and C are well-defined, we must show that every requirement receives attention or is tended to at most finitely often. This suffices, since we do not return to witnesses once we abandon them.

<u>Claim 5.13.1:</u> All requirements receive attention or is tended to at most finitely often and is eventually met. Thus $A^b \nleq_{bT} B^b$ and $C \nleq_{bT} A^b$.

Proof of Claim: We proceed inductively on the priority order. Suppose we have a requirement such that, at some stage s, all higher priority requirements have stopped receiving attention. We will show that in this case, the requirement under consideration receives attention or is tended to at most finitely often, and is eventually met.

$\mathbf{R}_{\langle e,i\rangle}$: If the witness for this requirement is non-coding, then we have two cases. If $\mathbf{R}_{\langle e,i\rangle}$ never requires attention for witness, then it can never require attention at all, since it will permanently have a non-coding witness. In this case, $\mathbf{R}_{\langle e,i\rangle}$ never receives attention after stage s, and must be met, since $\varphi_i(x)\uparrow$, where x is the non-coding witness. Hence φ_i is not total, and so A^b cannot be bT-computed from B^b using Φ_e and use bound φ_i. On the other hand, if $\mathbf{R}_{\langle e,i\rangle}$ does at some stage receive attention for witness, then we can wait until this stage, and continue as below.

If the witness is a coding witness, then it must be the permanent witness for $\mathbf{R}_{\langle e,i\rangle}$, since no higher priority requirement ever receives attention after this stage. So let m be this coding witness, with non-coding witness x. Let t be a future stage where $B^b \Upharpoonright \varphi_i(x)$ no longer changes. (Note that we know that $\varphi_i(x)\downarrow$ since m was created as the coding witness for the non-coding witness x.) In fact, the approximation to this initial segment of B^b can only change at most $(\varphi_i(x)+1)$-many times, since B is a fixed oracle, and hence elements can only enter B^b and then never leave.

If $\Phi_e^{B^b\Upharpoonright\varphi_i(x)}(x)\uparrow$, then this never occurs after stage t, since $B^b \Upharpoonright \varphi_i(x) = (B^b)_t \Upharpoonright \varphi_i(x)$ and elements cannot leave B^b once they have entered. In this case, $\mathbf{R}_{\langle e,i\rangle}$ is met, and never receives attention after stage t. So assume that $\Phi_e^{B^b\Upharpoonright\varphi_i(x)}(x)\downarrow$. By increasing t appropriately, we may assume that this happens by stage t. So $\Phi_{e,t}^{(B^b)_t\Upharpoonright\varphi_i(x)}(x)\downarrow$. Call this value w. If $\mathbf{R}_{\langle e,i\rangle}$ does not require attention (for coding), it is because $w \neq (A^b)_t(x)$. If it does require attention, then it will receive it – since all higher priority requirements no longer require attention – and so $w = (A^b)_t(x) \neq (A^b)_{t+1}(x)$. As long as $(A^b)_{t+1}(x)$ never changes after stage $t+1$, in either case we will never have that $\mathbf{R}_{\langle e,i\rangle}$ requires attention, and will be met.

We have

$$\begin{aligned} x \in (A^b)_t &\Leftrightarrow (\exists \ell < x)[\varphi_{\ell,t}(x)\downarrow \wedge \Phi_{x,t}^{A_t\Upharpoonright\varphi_\ell(x)}(x)\downarrow] \\ &\Leftrightarrow (\exists \ell < x)[\varphi_{\ell,t}(x)\downarrow \wedge m \in A_t \Upharpoonright \varphi_\ell(x)] \text{ by our definition of } \Phi_x \\ &\Leftrightarrow m \in A_t \text{ by taking } \ell = j \text{ with } \varphi_j(x)\downarrow = m+1 \end{aligned}$$

so this is true. That is, we can control $x \in A^b$ by changing $m \in A$ appropriately.

$\mathbf{Q}_{\langle e,i\rangle}$: Let y be the permanent witness for this requirement. Then this requirement receives attention only when $\varphi_{m,s}(z)\downarrow$ newly since the last initialization – the final initialization – of $\mathbf{Q}_{\langle e,i\rangle}$, and $m < z \leq \varphi_i(y)\downarrow$. If $\varphi_i(y)\uparrow$, then this requirement never receives attention and is met trivially. Otherwise, since new convergences can only occur at most once, $\mathbf{Q}_{\langle e,i\rangle}$ must receive attention at most finitely often.

Let s be a stage whereafter $\mathbf{Q}_{\langle e,i\rangle}$ never again receives attention and the witness y has been permanently chosen by stage s. Since y is permanent, no higher priority requirement receives attention after stage s. Then any x whose membership in A changes after stage s must be larger than $\max\{\varphi_m(z)\downarrow \mid m < z \leq \varphi_i(y)\}$, since all such convergences happen before stage s, and so all x must be lower priority witnesses that exist after stage s, and these x will have been chosen larger than this bound.

Then $A^b \upharpoonright \varphi_i(y) \supseteq (A^b)_s \upharpoonright \varphi_i(y)$, since elements can only enter A^b and then never leave (as A does not change below $\max\{\varphi_m(z)\downarrow \mid m < z \leq \varphi_i(y)\}$). Hence $\mathbf{Q}_{\langle e,i\rangle}$ will be tended to at most $\varphi_i(y)+2$ times, toggling the membership of y in C so that eventually $\Phi_e^{A^b \upharpoonright \varphi_i(y)}(y)\downarrow \neq C(y)$ forever. Hence $\mathbf{Q}_{\langle e,i\rangle}$ will be met, and receives attention and is tended to at most finitely often. ■

From this claim, it is clear that the approximation to C agrees with g_C. As mentioned above, we must be careful about coding witnesses being withdrawn from A if their requirement abandons them, so let us explicitly compute the number of times the approximation for such a witness may change. Since $B^b \upharpoonright \varphi_i(x)$ changes at most $\varphi_i(x)+1$ times, and the approximation to $A(m)$ changes only when $B^b \upharpoonright \varphi_i(x)$ does, we get a total number of changes as $\varphi_i(x)+3$, where the extra two changes are an initial convergence (which does not require a B^b change) and the final withdrawal upon abandonment by the requirement. Since $m > \varphi_i(x)+3$ by assumption, the number of changes is bounded by m, so g_A is correct.

We are now able to define $\hat{h}_A$ and $\hat{h}_C$. First, since 0 is the only limit ordinal below ω, we may think of $\hat{h}_A(x,t)(0)$ as an increasing computable function such that for each x and each s, $\lim_{t\to\infty} \hat{h}_A(x,t)(0)$ exists and bounds the use of B needed to determine if $x \in A_s$, and this approximation increases at most computably often. In fact, since we code B directly into the odd half of A, we can take $\hat{h}_A(2z+1,s)(0) \coloneqq (z,0)$, since at any given stage s, we know that we need only $B \upharpoonright z$ and also this approximation never needs to change. The even half is more complicated, so let us first discuss $\hat{h}_C$.

For $\hat{h}_C$, we note that there are only two limit ordinals below $\alpha = \omega + 1$: ω and 0. Since we know $y \notin C$ when the confidence of the approximation is ω, we can set $\hat{h}_C(y,s)(\omega) = (0,0)$, since we need no bits of B to determine this, and this approximation never needs to change. Since we only require $\hat{h}_C(y,s)(0)$ to exist if the membership of y in C has been declared with confidence less than ω, we only must worry about defining $\hat{h}_C(y,s)(0)$ once we see that this occurs. Since we have enforced that this happens if and only if $\varphi_i(y)\downarrow$ where $y \in \omega^{[\langle e,i\rangle]}$ for some e, we have that $\hat{h}_C(y,s)(0)$ may depend on the value of $\varphi_i(y)\downarrow$, even though this function is not a priori known to converge.

Let us now turn our attention to the three points we had to consider, which we said would help us determine $\hat{h}_A$ and $\hat{h}_C$.

1. Is x (resp. y) ever selected as the witness by stage s for the (single) requirement $\mathbf{R}_i$ (resp. $\mathbf{Q}_i$) that is permitted to use it a witness?
2. If so, is x (resp. y) ever abandoned as the witness for its requirement by stage s once it has been chosen?
3. How many times do we change the approximation of $x \in A$ (resp. $y \in C$) up to stage s while it is the witness?

Let us deal with the first and second points for $\hat{h}_A$ and $\hat{h}_C$ simultaneously, since witnesses for both types of requirements are chosen using the same procedure. In the construction, the conditions for picking a new witness are listed in the (Cause Injury) section. Importantly, tending to $\mathbf{Q}_i$ does *not* cause injury – so no new witnesses are picked and none are abandoned. Hence, given x, it is enough to find enough of B so that we can simulate this portion of the construction until the stage we desire. The restriction is that the amount of B cannot depend on s, although the simulation procedure itself certainly does.

So, consulting the construction, we note that none of B is required for when requirements $\mathbf{R}_i$ require attention for witness, nor when requirements $\mathbf{Q}_i$ require attention. The only amount of B we need is when we determine if a requirement $\mathbf{R}_{\langle e,i\rangle}$ requires attention for coding. For our fixed value, which we shall call w, we only need to simulate the action of requirements whose witnesses are below w, for if they are above w, they are of weak enough priority that any action they take cannot cause a change of witnesses below w. So we may assume that $\mathbf{R}_{\langle e,i\rangle}$ has coding witness $m < w$ and non-coding witness x.

Then for each stage $t \leq s$, we need enough of B to determine $(B^b)_t \upharpoonright \varphi_i(x)$. Now, $(B^b)_t \coloneqq \{z \mid (\exists j < z)[\varphi_{j,t}(z)\downarrow \wedge \Phi_{j,t}^{B\upharpoonright\varphi_{j,t}(z)}(z)\downarrow]\}$. Hence we need $B \upharpoonright \max\{\varphi_{j,t}(z)\downarrow \mid j < z \leq \varphi_i(x), t \leq s\}$. Our choice needs to work for all s, so we expand this to $B \upharpoonright \max\{\varphi_j(z)\downarrow \mid j < z \leq \varphi_i(x)\}$. By choice of m, we will have $\varphi_i(x) < m$, and we know $m < w$, so we need $B \upharpoonright \max\{\varphi_j(z)\downarrow \mid j < z < w\}$. This increases at most computably often (in w), and so we can define $\hat{h}_A$ and $\hat{h}_C$ using this to decide the first and second points.

In fact, to decide the number of times a value is toggled in A by stage s, (i.e. the third point for $\hat{h}_A$), this same strategy will work, as long as we additionally simulate the action of the requirement that selects w as its (coding) witness. To do so, we just use $B \upharpoonright \max\{\varphi_j(z)\downarrow \mid j < z \leq w\}$, by the reasoning above.

It remains to consider the third point for $\hat{h}_A$. This is substantially more complicated than what we have just done, since elements of C are not changed during the (Cause Injury) section, but rather when a requirement $\mathbf{Q}_{\langle e,i\rangle}$ has selected the given element as its witness and it is tended to. First, because we draw the witnesses for $\mathbf{Q}_{\langle e,i\rangle}$ from $\omega^{[\langle e,i\rangle]}$, we can determine ahead of time which requirement will be associated to which witness. Also, as mentioned in our prior discussion of $\hat{h}_C$, we need only define $\hat{h}_C(y,s)(0)$ at each stage s once $\varphi_i(y)\downarrow$ – recall that here i is such that $\mathbf{Q}_{\langle e,i\rangle}$ is the requirement associated to y – and our definition of this function is therefore permitted to use the value of $\varphi_i(y)$.

We toggle the membership of y in C only when $\mathbf{Q}_{\langle e,i\rangle}$ is tended to, i.e. only when $\Phi_e^{A^b\upharpoonright\varphi_i(y)}(y)[s]\downarrow = C_s(y)$. So, we need to determine $(A^b)_t \upharpoonright \varphi_i(y)$. Now, $(A^b)_t \coloneqq \{z \mid (\exists j < z)[\varphi_{j,t}(z)\downarrow \wedge \Phi_{j,t}^{A\upharpoonright\varphi_{j,t}(z)}(z)\downarrow]\}$, and so as before we need $A_t \upharpoonright \max\{\varphi_{j,t}(z)\downarrow \mid j < z \leq \varphi_i(y), t \leq s\}$. However, unlike before, this is not B, but A. We have just discussed $\hat{h}_A$, and so the natural idea is to use this to help us decide how much of B we need to compute this much of A_t. Unfortunately, each fixed value of the approximation $\hat{h}_A$ changes at most computably often, and without a fixed (computable) bound of the amount of A_t, we cannot find an approximation that changes at most computably often. (Instead, the number of changes would be computably bounded by ω^2 rather than ω).

Fortunately, we are rescued by the priority ordering of witnesses, and our insistence of withdrawing $\mathbf{R}_i$-witnesses as they are being abandoned. First, notice that $A \upharpoonright y$ cannot change, since this would cause injury to $\mathbf{Q}_{\langle e,i\rangle}$, and hence cause y to be abandoned as a witness. So, we just need to compute $A \upharpoonright y$ at the first stage where y is selected as a witness for $\mathbf{Q}_{\langle e,i\rangle}$. Now, let t be a stage after which y was selected as the witness for $\mathbf{Q}_{\langle e,i\rangle}$, but before it is abandoned (if ever). We know $\mathbf{Q}_{\langle e,i\rangle}$ must receive attention if it requires it, since no higher priority requirement can act without spoiling y. Hence, if $\max\{\varphi_{j,t}(z)\downarrow \mid j < z \leq \varphi_i(y)\}$ increased, we would injure all lower priority requirements, causing them to withdraw their witnesses from A (if they have a witness in A) and choose new witnesses above this range. Hence at any such stage t, we can assume that $A_t \upharpoonright \max\{\varphi_{j,t}(z)\downarrow \mid j < z \leq \varphi_i(y), t \leq s\}$ is of the form $\varnothing \oplus B$ above y.

So, to compute this initial segment of A_t, it suffices to compute $A_t \upharpoonright y$ and then determine $B \upharpoonright \max\{\varphi_{j,t}(z)\downarrow \mid j < z \leq \varphi_i(y), t \leq s\}$. We can remove the stage dependence of the latter by extending this to $B \upharpoonright \max\{\varphi_j(z)\downarrow \mid j < z \leq \varphi_i(y)\}$, which changes at most computably often. Since $A_t \upharpoonright\upharpoonright y$ is now a constant-length initial segment of A_t, we can use $\hat{h}_A$ to provide a bound on B required to compute this, and crucially, this bound will still only increase computably often.

This completes the verification: A is ω-b.c.e. in B and has $B \leq_1 A$, so $B \leq_{bT} A \leq_{bT} B^b$. Thus $B^b \leq_{bT} A^b \leq_{bT} B^{2b}$, and since all the requirements are met, we have that $B^b <_{bT} A^b <_{bT} B^{2b}$, as desired. □

Corollary 5.14.
Let $n > 0$. Then there is some set X such that $\varnothing^{nb} <_{bT} X <_{bT} \varnothing^{(n+1)b}$.

The attentive reader will note that we have proved a stronger statement than the one we set out to prove. Indeed, we produce a set A that is strictly bT-between B and B^b, but the reason this is so is because A^b is strictly bT-between B^b and B^{2b}. That is, A is bounded intermediate, i.e. not bounded low nor bounded high relative to B. An open question is thus if the proof can be adapted to generate sets that are bounded intermediate$_n$ for any $n \geq 1$ so that A^{nb} is strictly bT-between B^{nb} and $B^{(n+1)b}$. The classical proof of the existence of such intermediates (in the Turing degrees) is typically proved using pseudo-jump inversion (see Soare [28], for instance), but there is no known notion of pseudo-jump inversion for the bounded Turing degrees. Such a notion perhaps could be used, as in the classical case, to prove an equivalent of the Sacks Jump Inversion Theorem.

5.6 Jump Inversion and Further Work

The question of jump inversion for bounded Turing degrees has made an appearance in several of the preceding sections; it is now worth discussing it on its own.

As we have previously mentioned, the motivation for Anderson and Csima [2] to introduce the bounded jump was in an effort to rescue the Shoenfield Jump Inversion theorem in the bounded Turing degrees. However, the analogue of the Sacks Jump Inversion Theorem is not known. Recall that in the classical setting, the Theorem is as follows:

Theorem (Sacks Jump Inversion ([26]))**.**
If A is a set that is c.e. in and above $\varnothing'$, then there is a non-computable c.e. set X such that $X' \equiv_T A$.

In fact, Sacks proved more than this: he added the consequent that for any non-computable Δ^0_2 set C, A can be chosen so that it cannot compute C. However, this is typically omitted unless it is needed. Also, the proof of the Theorem is easily relativized, and it is this form that we are particularly interested in.

A result of Ng and Yu [23] shows that a non-relativized form of the Theorem is not true:

Theorem 5.15 (Ng and Yu [23, 2.1])**.**
There exists an $(\omega+1)$-c.e. set A such that for any c.e. set V, either $V^b \nleq_{bT} A \oplus \varnothing'$ or $A \nleq_{bT} V^b$.

Note that even this result need not quell any hopes of a version of Sacks for bounded reductions, since it replaces "c.e. in and above $\varnothing'$" with "$(\omega+1)$-c.e.". Indeed, one of the key differences between Sacks and Shoenfield is that Sacks is concerned with (relatively) *c.e.* sets rather than sets that are (relatively) Δ^0_2. Because of this, we may expect that if we wish to prove a version of the Sacks Jump Inversion Theorem for bounded Turing degrees, then we will need a notion of c.e. that works for bounded Turing degrees. At first blush, this may seem odd, but notice that relative to a set B, the definition of a c.e. set W_e^B is permitted to access an arbitrary amount of B, which is precisely what we wish to avoid when working with bounded Turing reductions. Thus even the existence of an $(\omega+1)$-c.e. set that fails to be invertible may not be an issue, if this set were outside what "ought" to be the equivalent of c.e. in and above $\varnothing'$.

In light of the above and the preceding work, the natural guess for a bounded analogue of a c.e. set is obtained by setting α to be 2 in the definition of α-b.c.e., since when $B \equiv_T \varnothing$, this would recover the definition of a c.e. set as usual. In this case, we get that a "bounded c.e." set in B is of the form W_e^B where the functional responsible has a use bounded by the function h and $h(x) = \lim_{s\to\infty} \hat{h}(x,s)$ changes computably often, and $\hat{h}$ is computable.

This definition is not what we would expect, however, since the more natural guess would be one that mimics the definition of A^b, namely that the use of the functional should be bounded by a computable function (i.e. h would be computable). If we are even more careful, we might mimic the definition even more closely and arrive at a definition where the use of the functional is not computable, but instead is the limit of a computable $\hat{h}$ as before, but $\hat{h}$ is permitted to change at most once, not computably often.

Perhaps then this is the reason why the analogue of the Sacks Jump Inversion Theorem (and pseudo-jump inversion, which is essentially equivalent) has resisted much effort thus far: the different equivalent forms of when a set A is computably enumerable relative to B seems to separate for bounded Turing reductions. Wu and Wu [30] have proved a version of pseudo-jump inversion for bounded Turing reductions, but that result is not a direct analogue as we would hope. Instead, it modifies the classical result to produce bounded high or bounded low sets, but does not actually provide any inversion related to the bounded jump per se, since it still deals with usual c.e. sets and (unbounded) Turing reductions.

These problems raise a question: is $\leq_{bT}$ really the best reducibility to be working with when it comes to the bounded jump? The question sounds absurd, but perhaps the bounded jump is instead a jump for a different reducibility and only happens to work for $\leq_{bT}$. If a better reducibility were chosen, perhaps some of the issues above would disappear, since then the two "types" of c.e. sets might be equivalent.

A member of the following family of reducibilities is the natural guess:

Definition 5.6.
Let A and B be sets such that $A = \Gamma^B$ for some Turing functional Γ. If there is a computable function $\hat{h}$ such that $u_\Gamma^B(x) \leq h(x)$ where $h(x) = \lim_{s\to\infty} \hat{h}(x,s)$ and $\hat{h}(x,\cdot)$ changes at most n times (resp. computably often), then we say that $A \leq_{n\text{-}bT} B$ (resp. $A \leq_{\omega\text{-}bT} B$).

Clearly $\leq_{0-bT}$ is just $\leq_{bT}$. Also note that $A \leq_{n-bT} B$ implies $A \leq_{m-bT} B$ when $n \leq m$, including when $m = \omega$. We would (perhaps) hope to resolve our problems above by finding some way of showing an equivalence between $\leq_{1-bT}$ and $\leq_{\omega-bT}$. Unfortunately, this is not possible, as the following propositions prove.

Proposition 5.16.
Fix $n \in \omega$. Then there are sets A and B such that $A \leq_{(n+1)-bT} B$ but $A \nleq_{n-bT} B$.

Proof:
In order to have that $A \nleq_{n-bT} B$, we require that if $p(x) = \lim_{s\to\infty} \varphi_i(x,s)$ is a use bound for the functional Φ_e, then this functional does not provide a reduction from A to B. So, we aim to meet the following requirements:

$\mathbf{R}_{\langle e,i\rangle}$: If φ_i is total and changes at most n times, with limit p, then there is some x such that $\Phi_e^{B\upharpoonright p(x)}(x) \neq A(x)$.

On the other hand, to show that $A \leq_{(n+1)-bT} B$, we build a function $\hat{h}(x,s)$ that changes at most $(n+1)$-many times for any fixed x, with limit h, and implicitly build a functional Ψ so that $\Psi_e^{B\upharpoonright h(x)} = A$.

We proceed using a finite injury construction. Since the proof is completely unsurprising, we describe the basic module and the actions taken during the construction; the interested reader can fill in the standard details. The basic module for $\mathbf{R}_{\langle e,i\rangle}$ is as follows:

Choose a witness $x \in \omega^{[\langle e,i\rangle]}$, and define $\hat{h}(x,0) = m$ where m is the least element of $\omega^{[x]}$. At stage $s+1$, compute $M \coloneqq \max_s\{\varphi_{i,s}(x,s)\downarrow\}$. If this exists and exceeds $\hat{h}(x,s)$, then set $\hat{h}(x, s+1)$ to be an element of $\omega^{[x]}$ that is larger than anything seen so far (including larger than M) and unused by the construction. Otherwise, set $\hat{h}(x, s+1) = \hat{h}(x, s)$. Ignore this if φ_i has increased n times. We do this for all $x \in \omega^{[\langle e,i \rangle]}$, even if x is not selected as the witness for the requirement at this stage. If x is the witness and $\Phi_{i,s}^{B \upharpoonright M}(x) \downarrow = A(x)$, then toggle the value of $A(x)$ to break this, and also toggle $B(\hat{h}(x, s+1))$ to match.

Since we start ignoring more than n changes in φ_i, $\hat{h}(x, \cdot)$ changes at most $n+1$ times, since it may need to increase an additional time after discovering the value of $\varphi_i(x, 0)$, which does not require a increase of φ_i. Since we destroy possible computations until φ_i has changed more than n times, we meet requirement $\mathbf{R}_{\langle e,i \rangle}$.

When a requirement receives attention, it injures requirements of lower priority, forcing them to abandon their witnesses, and they must choose a fresh large witness. In this way, witnesses like x can be toggled in and out of A and their coding location $\hat{h}(x, s)$ can be toggled in and out of B without disrupting computations that higher priority requirements would like to preserve.

Now note that we can compute A from B with use bound $h(x) = \lim_{s \to \infty} \hat{h}(x, s)$ since $x \in A$ if and only if $h(x) \in B$. □

Proposition 5.17.
There are sets A and B such that $A \nleq_{n-bT} B$ for any $n \in \omega$, but yet $A \leq_{\omega-bT} B$.

Proof Sketch:
Repeat the above, but requirements now also guess the maximum number of changes that their function is permitted to make and ignore them after they reach this number rather than the fixed value n as in the previous proof. Potential witnesses for $\mathbf{R}_{\langle e,i,n \rangle}$ – that is, elements in $\omega^{[\langle e,i,n \rangle]}$ – have $h(x)$ change at most $n+1$ times, which is computable from x, and $\hat{h}(x, s)$ is computable as before.

The rest of the basic modules, priority method and verification are essentially the same. □

(These results actually prove more, since the "use bound" is actually a coding location, so the reductions are more like a generalization of a 1-reduction.)

Thus, the multiple notions of c.e. cannot be reconciled under any of these reducibilities, because they are, respectively, the natural definitions for $0 - bT$, $1 - bT$ and $\omega - bT$ and by the above results, none of the reducibilities can coincide. Hence if we are to work with "the" definition of c.e. for the bounded Turing degrees, we will need to pick one of these and will be unable to pass between them as we like, as we do in the classical setting.

Hence we have the following broad questions, each building on the previous:

Question 5.18.
Is there a definition of b.c.e. that is more natural than others, and can it be used to prove analogues of results about relatively c.e. sets?

Question 5.19.
With such a definition, is the analogue of the Sacks Jump Inversion Theorem true? Does pseudo-jump inversion exist for the bounded Turing degrees?

Question 5.20.
Using such theorems (or perhaps without), can one show that the bounded high / bounded low hierarchies do not collapse? Are there bounded intermediate sets, i.e. sets that are not bounded low$_n$ or bounded high$_n$ for any n? If not, for which n are there sets that are bounded intermediate$_n$?

Question 5.21.
Are there natural characterizations of when a set is bounded low$_n$ or bounded high$_n$? How much about (classically) low and high sets can transferred to bounded low and bounded high sets?

References

[1] B. A. Anderson. Automorphisms of the truth-table degrees are fixed on a cone. *The Journal of Symbolic Logic*, 74(2):679–688, 2009.

[2] B. A. Anderson and B. F. Csima. A bounded jump for the bounded Turing degrees. *Notre Dame J. Form. Log.*, 55(2):245–264, 2014.

[3] B. A. Anderson, B. F. Csima, and K. M. Lange. Bounded low and high sets. *Archive for Mathematical Logic*, 56(5-6):507–521, 2017.

[4] C. J. Ash and J. Knight. *Computable Structures and the Hyperarithmetical Hierarchy.* Elsevier, 2000.

[5] N. A. Bazhenov, I. Sh. Kalimullin, and M. M. Yamaleev. Degrees of categoricity and spectral dimension. *J. Symb. Log.*, 83(1):103–116, 2018.

[6] N. A. Bazhenov and M. Marchuk. Degrees of categoricity for prime and homogeneous models. In *Conference on Computability in Europe*, pages 40–49. Springer, 2018.

[7] V. Brattka and G. Gherardi. Effective choice and boundedness principles in computable analysis. *Bulletin of Symbolic Logic*, 17(1):73–117, 2011.

[8] P. M. Cohn. The complement of a finitely generated direct summand of an abelian group. *Proceedings of the American Mathematical Society*, 7(3):520–521, 1956.

[9] B. F. Csima, M. Deveau, M. Harrison-Trainor, and M. Assem Mahmoud. Degrees of categoricity above limit ordinals. *Preprint*, page arXiv:1805.10249, May 2018. `https://arxiv.org/abs/1805.10249`.

[10] B. F. Csima, M. Deveau, and J. Stephenson. When does a relation code an isomorphism? *Preprint.*

[11] B. F. Csima, R. Downey, and K. M. Ng. Limits on jump inversion for strong reducibilities. *The Journal of Symbolic Logic*, 76(4):1287–1296, 2011.

[12] B. F. Csima, J. N. Y. Franklin, and R. A. Shore. Degrees of categoricity and the hyperarithmetic hierarchy. *Notre Dame J. Form. Log.*, 54(2):215–231, 2013.

[13] B. F. Csima and M. Harrison-Trainor. Degrees of categoricity on a cone via η-systems. *The Journal of Symbolic Logic*, 82(1):325–346, 2017.

[14] B. F. Csima and J. Stephenson. Finite computable dimension and degrees of categoricity. *Annals of Pure and Applied Logic*, 170(1):58–94, 2019.

[15] R. L. Epstein, R. Haas, and R. L. Kramer. Hierarchies of sets and degrees below $0'$. In *Logic Year 1979–80*, pages 32–48. Springer, 1981.

[16] E. B. Fokina, I. Sh. Kalimullin, and R. Miller. Degrees of categoricity of computable structures. *Arch. Math. Logic*, 49(1):51–67, 2010.

[17] S. S. Goncharov. Degrees of autostability relative to strong constructivizations. *Tr. Mat. Inst. Steklova*, 274(Algoritmicheskie Voprosy Algebry i Logiki):119–129, 2011.

[18] M. J. Groszek and T. A. Slaman. Moduli of computation. Talk presented at the Conference on Logic, Computability and Randomness, Buenos Aires, Argentina, 2007.

[19] D. R. Hirschfeldt and W. M. White. Realizing levels of the hyperarithmetic hierarchy as degree spectra of relations on computable structures. *Notre Dame J. Form. Log.*, 43(1):51–64 (2003), 2002.

[20] R. Lubarsky and F. Richman. Walker's cancellation theorem. *Communications in Algebra*, 42(4):1644–1649, 2014.

[21] J. Mohrherr. A refinement of low n and high n for the r.e. degrees. *Mathematical Logic Quarterly*, 32(1-5):5–12, 1986.

[22] A. Montalbán. Priority arguments via true stages. *The Journal of Symbolic Logic*, 79(4):1315–1335, 2014.

[23] K. M. Ng and H. Yu. Effective domination and the bounded jump. *Notre Dame J. Form. Log.*, to appear.

[24] A. Nies. *Computability and Randomness*, volume 51 of *Oxford Logic Guides*. Oxford University Press, Oxford, 2009.

[25] H. Rogers. *Theory of Recursive Functions and Effective Computability*, volume 5.

[26] G. E. Sacks. Recursive enumerability and the jump operator. *Transactions of the American Mathematical Society*, 108(2):223–239, 1963.

[27] J. R. Shoenfield. On degrees of unsolvability. *Annals of Mathematics*, pages 644–653, 1959.

[28] R. I. Soare. *Turing Computability: Theory and Applications.* Springer, Berlin, 2016.

[29] E. A. Walker. Cancellation in direct sums of groups. *Proceedings of the American Mathematical Society*, 7(5):898–902, 1956.

[30] G. Wu and H. Wu. Bounded jump and the high/low hierarchy. In *International Conference on Theory and Applications of Models of Computation*, pages 647–658. Springer, 2019.