4-24-2015

On Lowness for Isomorphism as Restricted to Classes of Structures

Jacob D. Suggs
University of Connecticut - Storrs, jacob.suggs@gmail.com

On Lowness for Isomorphism as Restricted to Classes of Structures

Jacob Suggs, Ph.D.
University of Connecticut, 2015

ABSTRACT

We explore the notion of lowness for isomorphism as restricted to various classes and subclasses of structures. We present results for equivalence structures, scattered linear orders, the non-scattered linear orders known as shuffle sums, and for various restrictions of these classes, with an emphasis on examining which properties, in general terms, of these structures play a role in which results and how changes to the various constructions affect the resulting structures. Finally, we present partial information on the relationships between the various types of results and their associated subclasses of structures, with notes on how we might expect these results to generalize.

On Lowness for Isomorphism as Restricted to Classes of Structures

Jacob Suggs

M.S. Mathematics, University of Connecticut, 2012
B.S. Mathematics, Physics, University of Tennessee 2005

A Dissertation
Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy
at the
University of Connecticut

2015

APPROVAL PAGE

Doctor of Philosophy Dissertation

On Lowness for Isomorphism as Restricted to Classes of Structures

Presented by
Jacob Suggs, M.S. Math, B.S. Math, Physics

Major Advisor	______________________ David Reed Solomon
Associate Advisor	______________________ Damir Dzhafarov
Associate Advisor	______________________ Stephen Flood

University of Connecticut
2015

ACKNOWLEDGEMENTS

I would like to express my thanks to my advisor, Dr. Reed Solomon, for his guidance throughout the process of writing my dissertation and my time at the University of Conneciticut in general, and for introducing me to computability theory. I would also like to thank the other members of my thesis committee, Dr. Damir Dzhafarov and Dr. Stephen Flood, for their time, effort and support.

And finally, I would like to express my appreciation to all the faculty in the math department for their contributions to my academic growth, the staff for always being helpful and knowing more about how things work than should be humanly possible, and my friends and family for their encouragement along the way.

TABLE OF CONTENTS

NOTATION

Classes of Linear Orders

- $\mathcal{C}_{\mathcal{L}}$, the class of all linear orders.
- $\mathcal{C}_{\omega}$, the class of linear orders of type omega.
- $\mathcal{C}_{Sc}$, the class of scattered linear orders.
- $\mathcal{C}_{\mathcal{S}}$, the class of all shuffle sums.
- $\mathcal{C}_{\mathcal{S}1}$, the class of all shuffle sums with $\mathcal{N} \subseteq \omega$.
- $\mathcal{C}_{\mathcal{S}0}$, the class of all shuffle sums of $\mathcal{N} = \{2\}$.

Classes of Equivalence Structures

- $\mathcal{C}_{\mathcal{E}}$, the class of all equivalence structures.
- $\mathcal{C}_{\mathcal{E}1}$, the subclass of $\mathcal{C}_{\mathcal{E}}$ with no equivalence classes of infinite size.
- $\mathcal{C}_{\mathcal{E}0}$, the subclass of $\mathcal{C}_{\mathcal{E}1}$ with exactly one equivalence class of each finite size.
- $\mathcal{C}_{\mathcal{E}0'}$, the subclass of $\mathcal{C}_{\mathcal{E}}$ with exactly one equivalence class of each finite size and one equivalence class of infinite size.

Introduction

Motivation and Preliminary Results

One of the many lowness notions used to explore the Turing degrees is the notion of lowness for isomorphism. Essentially, a degree is low for isomorphism if every time it can compute an isomorphism between two computable structures, then there is a computable isomorphism between these structures. In this way, to say that a degree **a** is low for isomorphism is to say that while **a** may compute many non-computable isomorphisms between pairs of structures, it can only do so when the computable degree can also compute isomorphisms, so that **a** is no better at matching up isomorphic structures than the computable sets.

We will focus on the behavior of the notion of lowness for isomorphism with the structures under consideration restricted to one of various classes of structures. This is motivated by a particular property of general lowness for isomorphism, as well as by the relationships between various structural properties and the degrees that will become apparent as we work within these classes. To make this more precise, we introduce the following definitions, as well as present some known results for context.

0.1 Definitions and Basic Results

Definition 0.1.1. *A structure $\mathcal{A}$ in a finite language $\mathscr{L}$ is computable iff it has computable domain and every relation and function symbol in $\mathscr{L}$ corresponds to a computable relation or function on this domain.*

We can assume the domain to be $\mathbb{N}$. Also note that only the relations and function symbols in $\mathscr{L}$ need correspond to computable relations and functions. This will be important because structures often have non-primitive relations that do not correspond to relation symbols in $\mathscr{L}$, but still need to be preserved by isomorphism: for instance, the successor relation on linear orders.

Definition 0.1.2. *Computable structures $\mathcal{A}$ and $\mathcal{B}$ are* **d***-computably isomorphic ($\mathcal{A} \cong_{\mathbf{d}} \mathcal{B}$) iff there exists an isomorphism $\alpha : \mathcal{A} \to \mathcal{B}$ s.t. $\alpha \leq_T \mathbf{d}$.*

We will often say **d**-isomorphic rather than **d**-computably isomorphic. We will write $\mathcal{A} \cong_{\Delta^0_1} \mathcal{B}$ if $\mathcal{A}$ and $\mathcal{B}$ are (**0**-)computably isomorphic.

Definition 0.1.3. *A degree* **a** *is low for isomorphism if and only if for any two computable structures $\mathcal{A}$ and $\mathcal{B}$ in $\mathcal{C}$, whenever $\mathcal{A} \cong_{\mathbf{a}} \mathcal{B}$, we also have $\mathcal{A} \cong_{\Delta^0_1} \mathcal{B}$.*

This notion first appeared in print in [3]. As mentioned, we can restrict the structures under consideration to arrive at a related notion.

Definition 0.1.4. *Let $\mathcal{C}$ be a class of structures closed under isomorphism. A degree* **a** *is low for $\mathcal{C}$-isomorphism if and only if for any two computable structures $\mathcal{A}$ and $\mathcal{B}$ in $\mathcal{C}$, whenever $\mathcal{A} \cong_{\mathbf{a}} \mathcal{B}$, we also have $\mathcal{A} \cong_{\Delta^0_1} \mathcal{B}$.*

Note that if we let $\mathcal{C}$ be the class of all structures, then a degree is low for isomorphism if it is low for $\mathcal{C}$-isomorphism. From these definitions and properties of the degrees, we get the following basic results:

Proposition 0.1.5. *If* $\mathbf{d}$ *is low for* $\mathcal{C}$*-isomorphism and* $\mathbf{d}$ *computes* $\mathbf{a}$*, then* $\mathbf{a}$ *is low for* $\mathcal{C}$*-isomorphism.*

Proof: Suppose $\mathbf{a}$ is not low for $\mathcal{C}$-isomorphism, and fix computable structures $\mathcal{A}$ and $\mathcal{B}$ in $\mathcal{C}$ such that $\mathcal{A} \cong_{\mathbf{a}} \mathcal{B}$ and $\mathcal{A} \ncong_{\Delta_1^0} \mathcal{B}$. Since $\mathbf{d}$ computes $\mathbf{a}$, and hence any isomorphism that $\mathbf{a}$ computes, we have $\mathcal{A} \cong_{\mathbf{d}} \mathcal{B}$ and $\mathcal{A} \ncong_{\Delta_1^0} \mathcal{B}$. Thus $\mathbf{d}$ is not low for $\mathcal{C}$-isomorphism, giving a contradiction.

$\square$

We obtain the following proposition in much the same way.

Proposition 0.1.6. *If* $\mathbf{a}$ *is not low for* $\mathcal{C}$*-isomorphism, and* $\mathbf{d}$ *computes* $\mathbf{a}$*, then* $\mathbf{d}$ *is not low for* $\mathcal{C}$*-isomorphism.*

Proposition 0.1.6 will be used to help characterize low for $\mathcal{C}$-isomorphism for some classes. We will also be interested in what happens as we restrict or expand the class of structures $\mathcal{C}$, and so will make use of the following propositions.

Proposition 0.1.7. *If* $\mathcal{C}_0 \subseteq \mathcal{C}_1$ *and* $\mathbf{a}$ *is not low for* $\mathcal{C}_0$*-isomorphism, then* $\mathbf{a}$ *is not low for* $\mathcal{C}_1$*-isomorphism.*

Proof: Any pair of structures in $\mathcal{C}_0$ that witness that **a** is not low for $\mathcal{C}_0$-isomorphism are also in $\mathcal{C}_1$.

$\square$

Proposition 0.1.8. *If $\mathcal{C}_0 \subseteq \mathcal{C}_1$ and* **a** *is low for $\mathcal{C}_1$-isomorphism, then* **a** *is low for $\mathcal{C}_0$-isomorphism.*

Proof: If there are no structures in $\mathcal{C}_1$ witnessing that **a** is not low for isomorphism, then there cannot be any such structures in $\mathcal{C}_0 \subseteq \mathcal{C}_1$.

$\square$

Note that any class of structures is contained in the class of all structures, so every set that is low for isomorphism is low for $\mathcal{C}$-isomorphism for every $\mathcal{C}$. Likewise if there is a class of structures $\mathcal{C}$ such that some set is not low for $\mathcal{C}$-isomorphism, then that set is not low for isomorphism.

0.2 Motivating Results

A coding method by Hirschfeldt, Khoussainov, Shore and Slinko in [1] provides the following useful result.

Proposition 0.2.1. *Let $\mathcal{G}$ be the class of directed graphs. A degree* **d** *is low for isomorphism iff* **d** *is low for $\mathcal{G}$-isomorphism.*

Proposition 0.2.1 is very useful for studying lowness for isomorphism: Franklin and Solomon[3] used this result to extend their forcing proof demonstrating that every

Cohen and Mathias 2-generic is low for $\mathcal{G}$-isomorphism to show that such sets are low for isomorphism, among other results.

Proposition 0.2.1 also raises questions about what happens when we restrict to other classes of structures. How does changing the class of structures $\mathcal{C}$ change which degrees are and are not low for $\mathcal{C}$-isomorphism, and when does placing or lifting restrictions on a collection of degrees for which we've proven a result allow us to prove the same result with a more or less restricted class of structures? Which properties of classes of structures are necessary for a given result about lowness for $\mathcal{C}$-isomorphism, and which can be weakened?

We will find the following two results useful:

Theorem 0.2.2 (Franklin, Solomon[3]). *Let $\mathcal{C}_{\mathcal{L}}$ be the class of all linear orders. No set that computes a separating set for computably inseparable c.e. sets is low for $\mathcal{C}_{\mathcal{L}}$-isomorphism.*

The construction Franklin and Solomon use in Theorem 0.2.2 can be used with little modification to prove the same theorem for computably inseparable Σ_2^0 sets, as we will see in Section 2.4. We will remove the padding copies of $\mathbb{Q}$ from their construction so as to highlight a distinction between how these proofs can work for scattered and non-scattered linear orders. We also use the same general type of coding for other structures.

Theorem 0.2.3 (Anderson, Csima[4])**.** *There is a Σ^0_2 degree that is low for isomorphism.*

It should be noted that in [4], Anderson and Csima's stated theorem is that there is a Σ^0_2 degree that is not a degree of categoricity (a related notion). However, the requirements in their construction make the Σ^0_2 set they built directly satisfy the definition of being low for isomorphism as well. This theorem will be useful in Chapter 2.

Chapter 1

Diagonalization Constructions

In this chapter, we give several results proved through a diagonalization technique. Of the two broad types of techniques we will use, the diagonalization arguments are more complicated, but they are broadly applicable to questions of whether Δ_2^0 and c.e. sets are low for $\mathcal{C}$-isomorphism for varying classes $\mathcal{C}$. We will use this diagonalization technique to obtain a result for the class of linear orders of type ω first, then move on to subclasses of shuffle sums and finally to equivalence structures. We will see in Chapter 2 that each of these structures can be approached with a coding argument as well, and in the case of shuffle sums that coding argument is much simpler. However, the simplicity comes at the cost of less control over the structure.

1.1 Linear Orders of Type ω and Δ_2^0 Degrees: Diagonalization Method

Definition 1.1.1. *Let $\mathcal{C}_\omega$ denote the class of linear orders with classical order type $(\omega, \leq)$. We say that* **a** *is low for ω-isomorphism if* **a** *is low for $\mathcal{C}_\omega$ isomorphism.*

Linear orders of type ω (the order type of the natural numbers $\mathbb{N}$) are extremely common objects of study. This alone suggests using them as an example, however ω is particularly well suited to illustrate these techniques due to a simplicity that arises from the following useful lemma. Because of this simplicity, we will prove the main theorem of this section again using a different technique in Section 2.1.

Lemma 1.1.2. *If $\leq_0$ and $\leq_1$ are computable orders of the classical order type $(\omega, \leq)$, then there is exactly one isomorphism $f : \leq_0 \to \leq_1$, and that isomorphism is computable from $\mathbf{0}'$.*

Proof: Uniqueness follows from the fact that any isomorphism must send the first element of $\leq_0$ to the first element of $\leq_1$, and respect the successor.

The isomorphism is computable from $\mathbf{0}'$ because "y is the successor of x" can be written as a single quantifier statement: $\forall z\, [(z \leq x \vee z \geq y) \wedge y > x]$.

$\square$

In addition to simplifying the construction in Theorem 1.1.3, Lemma 1.1.2 will help lead to a corollary categorizing all of the sets that are low for ω-isomorphism.

Theorem 1.1.3. *No non-computable Δ^0_2 set D is low for ω-isomorphism.*

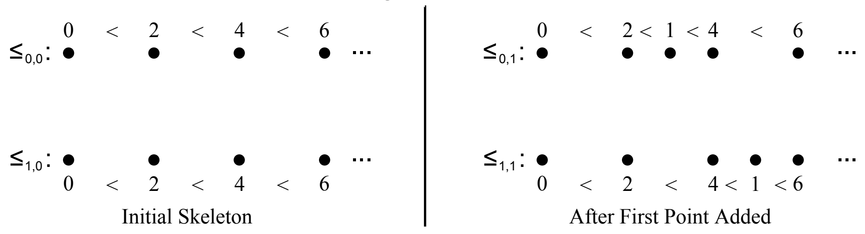
Proof: We will prove Theorem 1.1.3 using a diagonalization technique. It is possible to use a coding technique instead, however the simplicity of ω allows for a demonstration of this method without many of the technical details that arise in other classes of structures (such as the shuffle sums of $\{2\}$ we discuss later) where we do not have a proof using a coding technique.

Let D be an arbitrary non-computable Δ_2^0 set such that $D = \lim_s D_s$ for some fixed computable approximation $\{D_s\}$. We build two ω-type linear orders, $\leq_0$ and $\leq_1$, and a non-computable, D-computable isomorphism $\Gamma^D : \leq_0 \to \leq_1$. Since Lemma 1.1.2 tells us that there is only one isomorphism, building Γ^D to be non-computable shows us that there is no computable isomorphism between them. This will mean $\leq_0$ and $\leq_1$ are D-isomorphic but not computably isomorphic, which will show that D is not low for ω-isomorphism.

Our general strategy will be to lay out two identical copies of ω to serve as initial skeletons for $\leq_0$ and $\leq_1$, then to add points in such a way that the unique isomorphism has the properties we require. While doing this, we need to ensure both that $\leq_0$ and $\leq_1$ are computable and that they maintain order type ω.

Making $\leq_0$ and $\leq_1$ computable requires both that each have computable domain, and that the order relation itself is computable. We will ensure that the domains are computable by creating the initial skeletons using only even numbers, and then always adding the smallest odd number that is not yet in the domain each time the construction needs to add a point. (Each number is said to code its associated point.)

Figure 1.1.1:



In this way, so long as we add infinitely many points, the domains of both structures will be $\mathbb{N}$, hence computable. Further, once a point n is set to be less than (or greater than) a point m, that will never change - we will add points, but will not otherwise move points that we have already added. In this way, the order relation will also be computable: to compute whether $n \leq m$, watch the construction until both points have entered and check. Note that the successor relation will change, but as the successor relation is not associated with a symbol in our language, it is not required to be computable in order for the structure to be computable.

In addition to making $\leq_0$ and $\leq_1$ computable, we must ensure that the unique isomorphism between them is non-computable and computable from D, which we accomplish by constructing the Turing functional Γ in tandem with the orders. For the purposes of diagonalization, we will make the use of $\Gamma^D(n)$ oracle independent and increase to infinity with n for all n in our initial skeleton. We do this to control the uses that will affect Γ^{D_s} at our witnesses; this provides some helpful stability that we will take advantage of later. To ensure Γ^D is not computable, we meet the following requirements.

Requirements:

$$R_e\colon \Phi_e \neq \Gamma^D.$$

Meeting all R_e will complete the proof. No R_e will injure any other.

Notation: We will use the following notaton throughout the construction.

- $\leq_{0,s}$ and $\leq_{1,s}$ are the construction stages of $\leq_0$ and $\leq_1$ respectively.
- $x_{e,i}$ is the $(i+1)^{th}$ witness chosen by R_e. ($x_{e,0}$ is the first.)
- y is near x iff $y = x$ or y was added as part of meeting R_e at witness x.
- The neighborhood of x the set of points near x.
- n_e is the largest i index of the witnesses $x_{e,i}$ chosen by R_e so far.
- $t_{e,i}$ is the stage at which R_e chose $x_{e,i}$ as a witness.
- $\gamma_{e,i}$ is the use of $\Gamma^D(x_{e,i})$, and is oracle independent.
- If $x \in \leq_{0,0}$, $\hat{x}$ refers to the element with the same code in $\leq_{1,0}$.

Construction: Let D be a non-computable Δ^0_2 set with fixed approximation stages D_s as in the Limit Lemma.

Stage 0: Initialize $\leq_{0,0}$ and $\leq_{1,0}$ as identical copies of ω using the even numbers as described above. All requirements will choose all witnesses from these initial skeletons.

Stage $s+1$: Give attention to R_0 up to R_s as required (see strategy below). For all x for which $\Gamma^{D_{s-1}}(x)$ is defined, and for which no R_e defined $\Gamma^{D_s}(x)$ this stage, define

$\Gamma^{D_s}(x) = \Gamma^{D_{s-1}}(x)$ with the same use. For all $x < s$ such that $x \in \leq_{0,0}$ and $\Gamma^{D_s}(x)$ is undefined, set $\Gamma^{D_s}(x) = \hat{x}$ with use s.

The stage $s+1$ step of our construction does three things: First, whatever the relevant requirements tell it to do, second extend any definitions of $\Gamma^{D_{s-1}}$ from last stage to also hold for this stage (unless a requirement said to do otherwise), and third, to define $\Gamma^{D_s}(x) = x$ for the next x in the initial skeleton. This third part is simply the construction (barring interference from a requirement) being sure to map the n^{th} element of the initial skeleton of $\leq_0$ to the n^{th} element of the initial skeleton of $\leq_1$ at stage n. We do this as we go rather than in the initial stage so that we can always choose witnesses from our initial skeletons where Γ^{D_s} has never been defined.

Strategy for R_e at stage s: (Recall, R_e is $\Phi_e \neq \Gamma^D$.) R_e can need attention for three reasons, and possibly for several at once. We will give a brief description of how this strategy plays out for a particular R_e first, then give the specifics.

The short explanation for the process for a particular R_e, without going into many of the details, is as follows: First, R_e chooses some large witness from $\leq_{0,0}$, and sets Γ^{D_s} to map that witness to the corresponding element in $\leq_{1,0}$. R_e then waits to see if $\Phi_{e,s}$ agrees with that definition of Γ^{D_s}. If this never happens, then we win. If it does happen, then R_e picks a new large witness and starts trying to diagonalize on the first witness (and undiagonalizing when it must). If R_e ever sees a stage where it doesn't think it is winning (that is, $\Phi_{e,s}$ agrees with Γ^{D_s} on all of R_e's witnesses), then we pick a new witness, and try to diagonalize on all but the most recent.

Here we will also adopt the practice of referring to elements of $\leq_{1,0}$ that have the same code as $x \in \leq_{0,0}$ by $\hat{x}$. This is not technically necessary, but is convenient for bookkeeping purposes.

In detail, R_e needs attention at stage s if:

0. If R_e does not have any witnesses, choose a large witness $x_{e,0}$ from $\leq_{0,0}$ and set $t_{e,0} = s$, $\Gamma^{D_s}(x_{e,0}) = \hat{x}_{e,0}$ and $n_e = 0$.

1. If $n_e > 0$, for each $0 \leq i < n_e$, check if the answer to "does $D_s \upharpoonright \gamma_{e,i} = D_t \upharpoonright \gamma_{e,i}$ for some t with $t_{e,i} \leq t \leq t_{e,i+1}$" is different from the answer to "does $D_{s-1} \upharpoonright \gamma_{e,i} = D_t \upharpoonright \gamma_{e,i}$ for some t with $t_{e,i} \leq t \leq t_{e,i+1}$" (Check and see if we either gained or lost permission to diagonalize). For each i, if there is such a change, then follow the diagonalization procedure below.

2. If $\Phi_{e,s}(x_{e,n_e}) \downarrow= \hat{x}_{e,n_e}$ and either $n_e = 0$ or $\Gamma^{D_s}(x_{e,i}) = \hat{x}_{e,i}$ for all $0 \leq i < n_e$ (meaning R_e does not have permission to diagonalize on any previous witness), then increment n_e and, using the newly incremented n_e, remember the current stage number s as t_{e,n_e}, pick a new large witness x_{e,n_e} from $\leq_{0,0}$, and define $\Gamma^{D_s}(x_{e,n_e}) = \hat{x}_{e,n_e}$.

(Note that for any fixed e, R_e will always need attention for reason 2 once before it ever can for reason 1, but after the first time that reason 2 applies, it is necessary to check and satisfy reason 1 at each stage before checking reason 2.)

To see that the condition in 1 accurately describes when we may and may not diagonalize, consider the action of R_e on witness $x_{e,i}$. We choose this witness at stage $t_{e,i}$ and sets $\Gamma^{D_{t_{e,i}}}(x_{e,i}) = \hat{x}_{e,i}$ with use $\gamma_{e,i}$. We then begin waiting to see if $\Phi_e(x_{e,i}) = \hat{x}_{e,i}$ (at a stage where no earlier witness is diagonalized). While waiting, we continue to define $\Gamma^{D_s}(x_{e,i}) = \hat{x}_{e,i}$ with use $\gamma_{e,i}$, for all of the (potentially) different configurations of $D_s \upharpoonright \gamma_{e,i}$ that we see.

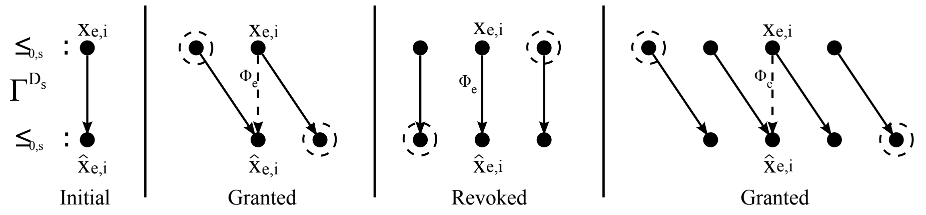
If we see $\Phi_e(x_{e,i}) = \hat{x}_{e,i}$ (and have no diagonalizations on earlier witnesses) at stage s, we set $t_{e,i+1} = s$ and choose are new witness $x_{e,i+1}$. At this point, we are committed to $\Gamma^{D_t}(x_{e,i}) = \hat{x}_{e,i}$ for all $t_{e,i} \leq t \leq t_{e,i+1}$, but have not defined $\Gamma^{D_s}(x_{e,i})$ for any s such that $D_s \upharpoonright \gamma_{e,i} \neq D_t \upharpoonright \gamma_{e,i}$ for all such t.

If we find a stage s_0 such that $D_{s_0} \upharpoonright \gamma_{e,i} \neq D_t \upharpoonright \gamma_{e,i}$ for all $t_{e,i} \leq t \leq t_{e,i+1}$, we are free to define $\Gamma^{D_{s_0}}(x_{e,i})$ differently - that is, we have permission to diagonalize - and will modify our orders and define $\Gamma^{D_{s_0}}(x_{e,i})$ according to the procedure below (always maintaining the same use $\gamma_{e,i}$). However, since D is Δ^0_2, it is possible that we later see a stage $s_1 > s_0$ at which $D_{s_1} \upharpoonright \gamma_{e,i} = D_t \upharpoonright \gamma_{e,i}$ for some $t_{e,i} \leq t \leq t_{e,i+1}$. If this happens, our computation reverts to $\Gamma^{D_{s_1}}(x_{e,i}) = \Gamma^{D_t}(x_{e,i}) = \hat{x}_{e,i}$, and we will have to make adjustments to our orders to accommodate this undiagonalization (also described below).

Of course, since D is Δ^0_2, we could gain permission to diagonalize again, so that we could cycle between being diagonalized and not being diagonalized at $x_{e,i}$ an arbitrarily large (but finite) number of times. Note though that we will always have permission to diagonalize exactly when $D_s \upharpoonright \gamma_{e,i} \neq D_t \upharpoonright \gamma_{e,i}$ for all $t_{e,i} \leq t \leq t_{e,i+1}$: Diagonalizing does not require that we define $\Gamma^{D_s}(x_{e,i})$ to be some value that it has never taken before, only that it be defined to not be $\hat{x}_{e,i}$ (and in fact each point will only ever have at most one diagonalized image and one undiagonalized image). This means that if $D_s \upharpoonright \gamma_{e,i}$ reverts to (say) $D_{s_0} \upharpoonright \gamma_{e,i}$, we will still be diagonalized, so long as the diagonalization procedure succeeds in modifying orders so that Γ^{D_s} is always an isomorphism and our orders are always copies of ω.

Diagonalization procedure for 1.: Since D is Δ^0_2, and hence D_s can revert to earlier configurations, we need to diagonalize in a way that can be undone and redone arbitrarily many times while still ensuring that Γ^D will be an isomorphism. We do this by adding points and changing Γ^{D_s} as shown below. The circled points are added each step.

Figure 1.1.2:



Where every time we add points while diagonalizing or undiagonalizing at $x_{e,i}$, we define the (oracle independent) use of Γ^D at those points to be $\gamma_{e,i}$.

To describe this process, we introduce the words near and neighborhood. Once a point $x_{e,i}$ is chosen as a witness, we say that any point our diagonalization adds to some $\leq_{0,s}$ while diagonalizing at $x_{e,i}$ is near $x_{e,i}$ (in the neighborhood of $x_{e,i}$), and likewise that any point added to some $\leq_{1,s}$ during the diagonalization is near (in the neighborhood of) $\hat{x}_{e,i}$. This allows us to refer to the portions of the orders that are being used for specific diagonalization attempts.

If we gain permission to diagonalize at stage s, we do two things. First, we add a point on the left side of the neighborhood $x_{e,i}$, and set it to be the preimage under Γ^{D_s} of the leftmost point near $\hat{x}_{e,i}$ (possibly $\hat{x}_{e,i}$ itself). Similarly, we place a point to the right of all points near $\hat{x}_{e,i}$, and set it to be image of the right most point in the neighborhood of $x_{e,i}$ (again, possibly $x_{e,i}$ itself). All other points in in the neighborhood $x_{e,i}$ have their images shifted one point to the right, so that Γ^{D_s} remains an isomorphism.

If our permission is revoked at stage s, we must revert the computation of $\Gamma^{D_s}(x_{e,i})$ back to $x_{e,i}$. All points near $x_{e,i}$ except the most recently added (the leftmost, call it z) already have an image defined for some other $D_s \upharpoonright \gamma_{e,i}$ where we did not have permission to diagonalize, so we simply revert their images to their previous values and add a point to on the left side of the neighborhood of $\hat{x}_{e,i}$ to be the image of z. This will always leave the rightmost point in the neighborhood of $\hat{x}_{e,i}$ without a preimage, so we then add a new point on the right side of the neighborhood of $x_{e,i}$) to be that preimage.

Since we always add points to the edges of a neighborhood, the only changes in images under Γ^{D_s} near $x_{e,i}$ are right shifts or left shifts, and since the left shifts and right shifts will alternate, this also guarantees that all points will have the same image under $\Gamma^{D_{s_0}}$ and $\Gamma^{D_{s_1}}$ should $D_{s_0} = D_{s_1}$ up to the appropriate use.

It is worth noting that because D is Δ^0_2, $D \upharpoonright \gamma_{e,i}$ can only change a finite number of times, and so R_e only adds a finite number of points around each witness. This guarantees that each $\leq_j$ will remain a linear order of type ω, and also that Γ^{D_s} will eventually settle on a fixed computation, so that Γ^D will be defined. We should also note that R_e must continue to check condition 2 forever, even if it eventually learns that Φ_e halts and differs from the identity at some later witness (and hence that R_e is actually won forever) - while the values of $\Gamma^{D_s}(x_{e,i})$ will eventually stabilize, we have no way to tell when they have and so must continually run this procedure in order to keep Γ^D well defined.

Further, R_e always thinks it has won because either $\Phi_e(x_{e,n_e})$ has not halted to $\hat{x}_{e,n_e}$, or R_e has managed to diagonalize using at least one of its earlier witnesses. It remains to show only that R_e will eventually think it has won for a reason that's actually true.

Verification of strategy for R_e: First consider the case where R_e chooses only finitely many witnesses, and hence n_e reaches a finite limit. Fix a stage u_0 such that n_e does not change after u_0, and let n_e denote the final value. If $\Phi_e(x_{e,n_e}$ does not converge to $\hat{x}_{e,n_e}$, then we win R_e because Φ_e is not an isomorphism. Therefore, assume $\Phi_{e,u_1}(x_{e,n_e}) =$

$\hat{x}_{e,n_e}$ for some stage $u_1 > u_0$.

Under the assumption that $\Phi_{e,u_1}(x_{e,n_e}) = \hat{x}_{e,n_e}$, we cannot have that $n_e = 0$ because if so we would immediately choose another witness and increment n_e, violating our assumption that n_e is fixed after u_0. Therefore, $n_e > 0$. Because n_e is not incremented after stage u_1, we must have that at every stage $s > u_1$, $\Gamma^{D_s}(x_{e,i}) \neq \hat{x}_{e,i}$ for some $0 \leq i < n_e$ (otherwise, R_e would need attention for reason 2 and choose another witness). Because the values of $\Gamma^{D_s}(x_{e,i})$ all eventually stabilize, this must mean that there must be at least one index i with $0 \leq i < n_e$ such that $\Gamma^D(x_{e,i}) \neq \hat{x}_{e,i}$. Since $\Phi_e(x_{e,i}) = \hat{x}_{e,i}$ for all i with $0 \leq i < n_e$ (and Γ^D is the only isomorphism), we have won R_e.

Second, consider the case when R_e chooses infinitely many witnesses, and hence the value of n_e increases forever. Then the values of x_{e,n_e} and γ_{e,n_e} increase unboundedly with n_e, as does t_{e,n_e}. Further Γ cannot retain permission to diagonalize at any $x_{e,i}$, or R_e would stop picking new witnesses and n_e would stabilize. We show that D is computable in this case, deriving a contradiction.

To compute $D(y)$: Run the construction until we find an i such that $y < \gamma_{e,i}$ and either $D_t(y) = 1$ or $D_t(y) = 0$ for all t between the corresponding $t_{e,i}$ and $t_{e,i+1}$. This is guaranteed to occur by the Limit Lemma (D is Δ^0_2) together with (from our supposition) that γ_{e,n_e} and t_{e,n_e} approach infinity (hence there are i such that $\gamma_{e,i}$ and $t_{e,i}$ are arbitrarily large).

If it were not the case that $D(y) = D_t(y)$ for all such t, then we know that $D \upharpoonright \gamma_{e,i} \neq D_t \upharpoonright \gamma_{e,i}$ for any such t and hence we must eventually get (and retain) permission to diagonalize at $x_{e,i}$ (via reason to diagonalize 1.). But by our supposition, this does not happen. So we must have $D(y) = D_t(y)$ for t in the interval.

Note that even in this case we do not know when $D_s(y)$ will become fixed - it need not be at the $t_{e,i}$ found earlier. Because $D_s(y)$ will eventually stop changing, there will certainly be stages $t_{e,i}$ and $t_{e,i+1}$ so that $D_t(y)$ is fixed between them - but there is no reason to assume that all later $D_s(y)$ are equal to such $D_t(y)$. We only know that if $D_s(y)$ does change after stage $t_{e,i+1}$, it will eventually change back to and remain fixed at the value it had for s in the interval - otherwise we would get and retain permission to diagonalize.

Therefore, if n_e approaches infinity for any e, then D is computable. But D is not computable. Therefore n_e must have a finite limit for all e, and so all R_e are met.

$\square$

This, together with Lemma 1.1.2 leads immediately to a full categorization of which Turing degrees are low for ω-isomorphism.

Corollary 1.1.4. *A degree* **a** *is not low for ω-isomorphism if and only if* **a** *computes a non-computable Δ^0_2 degree.*

Proof: ($\Leftarrow$): No Δ^0_2 degree is low for ω-isomorphism by Theorem 1.1.3. Together with Proposition 0.1.6, this implies that no degree **a** that computes a Δ^0_2 degree is low for ω-isomorphism.

($\Rightarrow$): If **a** is not low for ω-isomorphism, then there are two copies of ω, $\leq_0$ and $\leq_1$, such that $\leq_0 \cong_{\mathbf{a}} \leq_1$ but $\leq_0 \not\cong_{\Delta^0_1} \leq_1$.

From Lemma 1.1.2, there is exactly one isomorphism between $\leq_0$ and $\leq_1$, and it has Δ^0_2 degree. Since **a** computes an isomorphism, it computes this unique isomorphism with Δ^0_2 degree, and since $\leq_0 \not\cong_{\Delta^0_1} \leq_1$, this isomorphism is not computable. Therefore, **a** computes a non-computable Δ^0_2 degree.

□

This a powerful result, but it relies on the fact that for any two copies of ω, there is exactly one isomorphism between them, and most structures do not have this property. However, we can get a similar result with a weaker condition, as we will see in Section 1.3.

We also get the following corollary, through Proposition 0.1.7.

Corollary 1.1.5. *Let $\mathcal{C}_{\mathcal{L}}$ be the class of linear orders. No non-computable Δ^0_2 set D is low for $\mathcal{C}_{\mathcal{L}}$-isomorphism.*

1.2 Shuffle Sums and Δ^0_2 Degrees

From orders of type ω, we move to shuffle sums. Shuffle sums are modifications of the rationals $\mathbb{Q}$, as follows. (See Kach[2] for background and other results regarding shuffle sums.)

Definition 1.2.1. *The shuffle sum of a countable set $\mathcal{N} = \{\mathcal{L}_n\}_{n\in\omega}$ of linear orders is the (unique) linear order obtained by partitioning the rationals into dense sets $\{\mathcal{Q}_n\}_{n\in\omega}$ and replacing each rational of $\mathcal{Q}_n$ with the linear order $\mathcal{L}_n$.*

We will refer to each instance of some $\mathcal{L}_n$ in the shuffle sum as a suborder (or pair, for instances of $\mathcal{L}_n = 2$). This means that in general, the suborder containing a point and the neighborhood of a point ("neighborhood" is used in the same way as previously) can be different.

It is important to note that the choice of partition $\{\mathcal{Q}_n\}_{n\in\omega}$ has no effect on the isomorphism type of the shuffle sum. If $\mathbb{Q}$ and $\hat{\mathbb{Q}}$ are copies of the rationals, partitioned into dense $\{\mathcal{Q}_n\}$ and $\{\hat{\mathcal{Q}}_n\}$ respectively, then the density of each $\mathcal{Q}_n$ and $\hat{\mathcal{Q}}_n$ allows us to use a standard back and forth construction to build an order-preserving isomorphism $f : \mathbb{Q} \to \hat{\mathbb{Q}}$ such that $f(\mathcal{Q}_n) = \hat{\mathcal{Q}}_n$ for each n. The process of converting $\mathbb{Q}$ and $\hat{\mathbb{Q}}$ into shuffle sums $\mathcal{S}$ and $\hat{\mathcal{S}}$, so long as each uses the same $\mathcal{N}$, also naturally extends f to an isomorphism between shuffle sums by simply matching up each $\mathcal{L}_n$ as they replace each $x \in \mathcal{Q}_n$ and $f(x) \in \hat{\mathcal{Q}}_n$.

One consequence of this is that it really is the set $\mathcal{N}$ that determines the isomorphism type of the shuffle sum rather than the sequence of (not necessarily distinct) $\mathcal{L}_n$ and associated $\mathcal{Q}_n$. If $\mathcal{L}_m = \mathcal{L}_n$, we could simply remove $\mathcal{L}_m$ from the sequence and merge $\mathcal{Q}_m$ into $\mathcal{Q}_n$ without changing anything. Likewise, we can take even a finite set of linear orders $\mathcal{N} = \{\mathcal{L}_0, ..., \mathcal{L}_k\}$, and extend it to a countable sequence as in the definition by defining $\mathcal{L}_n = \mathcal{L}_k$ for all $n \geq k$.

Among other uses, this allows us to create subclasses of shuffle sums by placing restrictions on $\mathcal{N}$. And since we will be constructing these shuffle sums by adding points to $\mathbb{Q}$ to create suborders of the appropriate size, restrictions on $\mathcal{N}$ translate to restrictions on how we can add points, which is directly related to the complexity of the degrees we are testing for lowness for $\mathcal{C}$-isomorphism. We will start by considering a fairly simple subclass of shuffle sums, and then extend our results afterwards.

Definition 1.2.2. *Let $\mathcal{C}_{\mathcal{S}0}$ be the class of all shuffle sums of $\mathcal{N} = \{2\}$.*

These are the shuffle sums obtained by replacing every single element in $\mathbb{Q}$ with a pair of points, one of which is a left point (has no predecessor) and one of which is a right point (has no successor). We will show that no Δ^0_2 degree is low for $\mathcal{C}_{\mathcal{S}0}$-isomorphism. This is an interesting result because $\mathbb{Q}$ is computably categorical (any two copies are computably isomorphic), and here we see that even a fairly simple modification is enough to make it so that even the comparatively computationally weak Δ^0_2 sets can all find isomorphisms where no computable one exist.

We will use a similar proof technique to that in Theorem 1.1.3, with one large complication. Any two shuffle sums of the same $\{\mathcal{N}\}$ have uncountably many isomorphisms between them, and so it is no longer sufficient to show that the particular isomorphism that we build is not computable. This will make the diagonalization somewhat more complex, but will also serve to demonstrate the adaptability of the proof technique. The restriction to $\mathcal{N} = \{2\}$ actually makes things more complex than no restriction at all, and we will see how other classes of shuffle sums can be handled more simply in Chapter 2.

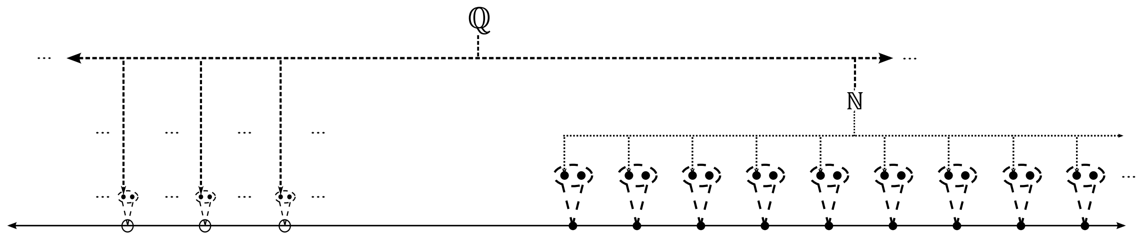
Theorem 1.2.3. *No Δ^0_2 degree is low for $\mathcal{C}_{\mathcal{S}0}$-isomorphism.*

Proof: As before we construct two structures (now shuffle sums) $\mathcal{S}_0$ and $\mathcal{S}_1$, and a Turing functional Γ such that Γ^D is an isomorphism between them.

Let $\mathcal{S}_B$ denote some computable shuffle sum over $\{2\}$ such that the successor relation (and hence whether a point is a left or right point) is computable, and which has a computable, nowhere-dense, increasing subset. Such a shuffle sum could easily be built by starting with $\mathbb{Q}$ and giving each point a successor one at a time. $\mathcal{S}_B$ will work as a "base" shuffle sum - both $\mathcal{S}_0$ and $\mathcal{S}_1$ will be initialized as $\mathcal{S}_B$, and when we decide that one point should no longer be a successor of another, we will build a copy of $\mathcal{S}_B$ between them.

We will always use $\mathbb{Q}$ to refer to the left points of the initial stages ($\mathcal{S}_{0,0}$ and $\mathcal{S}_{1,0}$) of our shuffle sums. We will use $\mathbb{N}$ and "the naturals" to refer to a particular fixed computable, nowhere-dense, increasing subset of $\mathbb{Q}$ (since we are looking at this as an order only, it doesn't really matter which one we choose). $\mathbb{N}$ will be the set of potential witnesses, and so will be handled separately from the other points in the construction. (Note that "natural number" will be referring to such points and not codes as is sometimes common.) We will use notation similar to that in the proof of Theorem 1.1.3. Since $\mathcal{S}_0$ and $\mathcal{S}_1$ will be initialized in exactly the same way, we will refer to the map sending elements of $\mathcal{S}_{0,0}$ to elements with the same code in $\mathcal{S}_{1,0}$ as "the identity." As before, if x is an element of $\mathcal{S}_0$, then we will refer to the element of $\mathcal{S}_1$ with the same code as $\hat{x}$.

Figure 1.2.1: $\mathcal{S}_{0,0}$ and $\mathcal{S}_{1,0}$



Requirements:

R_e: "Φ_e is not an isomorphism."

In this construction there will be injury, so we use the priority order $R_0 > R_1 > R_2 > \ldots$. We will make use$(\Gamma^D(n))$ oracle independent on and approach infinity with $n \in \mathbb{N}$ (again, as elements of $\mathbb{Q} \subset \mathcal{S}_j$, and not as codes).

Construction:

Let D be a non-computable Δ^0_2 set with fixed approximation stages D_s as in the Limit Lemma.

Stage 0: Initialize $\mathcal{S}_{0,0}$ and $\mathcal{S}_{1,0}$ to be identical copies of $\mathcal{S}_B$. Fix an enumeration q_i of $\mathbb{Q} \subset \mathcal{S}_{0,0}$.

Stage $s+1$: For each $i < s+1$, if $\Gamma^{D_{s+1}}(q_i)$ is undefined, set $\Gamma^{D_{s+1}}$ to be the identity on both q_i and its successor. Give attention to R_e for $0 \leq e < s$ as required, and extend any previous definitions of Γ^{D_s} to $\Gamma^{D_{s+1}}$, except where some R_e has already defined $\Gamma^{D_{s+1}}$ differently.

Strategy for R_e at stage s: As before, $x_{e,i}$ will denote the witnesses chosen by R_e, n_e will (at any given stage) be the largest i index of such $x_{e,i}$, and $t_{e,i}$ is the stage number at which that particular $x_{e,i}$ was chosen. R_e needs attention if:

0. R_e has no witness. Then set $x_{e,0}$ to be some large element of $\mathbb{N}$, set $n_e = 0$, and set $t_{e,0} = s$. Set Γ^{D_s} to be the identity on $x_{e,0}$ and on its successor (referred to as $y_{e,0}$), both with large use $\gamma_{e,0}$.

1. If $n_e > 0$, for each $0 \leq i < n_e$, handle any diagonalization that may be required at $x_{e,i}$. (See below.)

2. If $\Phi_{e,s}(x_{e,n_e}) \downarrow$ and we are not currently diagonalized, then increment n_e and, using the newly incremented n_e, remember the current stage number s as t_{e,n_e}, choose the next witness x_{e,n_e} to be some large natural, and set Γ^{D_s} to be the identity on the newly chosen x_{e,n_e} and its successor y_{e,n_e}, both with large use $\gamma_{e,0}$.

Diagonalization: Because (unlike the previous proof) it is not sufficient to show that $\Gamma^D \neq \Phi_e$, we now have cases depending on the value of $\Phi_e(x_{e,i})$. The case that requires the most care is still $\Phi_e(x_{e,i}) = \hat{x}_{e,i}$, but other situations require handling as well. Once again, any time we add points during a diagonalization involving $x_{e,i}$ and define Γ^{D_s} on those points, we give those computations the (oracle independent) use $\gamma_{e,i}$.

Case 1, $\boldsymbol{\Phi_e(x_{e,i}) = \hat{x}_{e,i}}$: Check if the answer to "does $D_s \upharpoonright \gamma_{e,i} = D_t \upharpoonright \gamma_{e,i}$ for some t with $t_{e,i} \leq t \leq t_{e,i+1}$" is different from the answer to "does $D_{s-1} \upharpoonright \gamma_{e,i} = D_t \upharpoonright \gamma_{e,i}$ for some t with $t_{e,i} \leq t \leq t_{e,i+1}$".

(Recall that $\gamma_{e,i}$ is the oracle independent use of $\Gamma^D(x_{e,i})$, that $\Gamma^{D_s}(x_{e,i})$ is defined for the first time at $t_{e,i}$, and that this definition is extended up until the next witness is chosen at $t_{e,i+1}$.)

We will follow a diagonalization procedure similar to the one used in Theorem 1.1.3, where we shift the images of all points in the neighborhood of our witness right when we gain permission to diagonalize, then left when we lose it, adding points to be new images or preimages where required to keep Γ^{D_s} an isomorphism. The primary complication is that we are building a shuffle sum over $\{2\}$, and so can never have more than 2 points in succession. Every time our construction would place more than 2 points in succession, we will build copies of $\mathcal{S}_B$ between them (and add a point if necessary) in such a way as to split the points into two groups of 2. This is easiest to see the first time we diagonalize:

Figure 1.2.2:

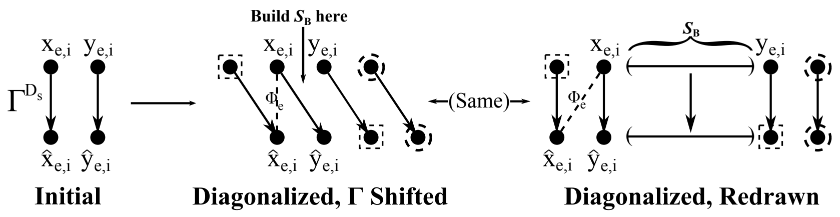
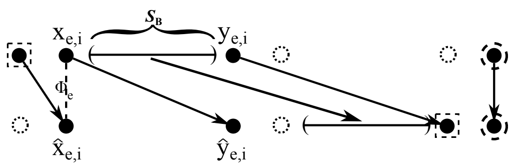


Figure 1.2.2 shows the first diagonalized configuration in two different but equivalent ways. We first shift all of the images one point to the right, adding the points in boxes in the figure to be a preimage for $\hat{x}_{e,i}$ and an image for $y_{e,i}$ (in a way similar to Figure 1.1.2 from the diagonalization procedure in Theorem 1.1.3). But if we stopped there, then we would have a group of three points, which would mean that our structure was no longer the appropriate type of shuffle sum. So we add two more points on the far right (the points in circles in the figure), and build copies of $\mathcal{S}_B$ separating these into two groups of two points. The third portion of the figure shows this same situation redrawn so that Γ^{D_s} is vertical in the diagonalized stage. Since, at this stage, Φ_e is mapping a right point to a left point, Φ_e cannot be an isomorphism and so R_e currently thinks it is winning.

This redrawing stage is primarily helpful because we can imagine that, every time we need to diagonalize or undiagonalize, we slide the points in either $\mathcal{S}_{0,s}$ or $\mathcal{S}_{1,s}$ over so that the desired way of mapping Γ^{D_s} is vertical. This then leaves gaps in the shuffle sums that we fill in with either more copies of $\mathcal{S}_B$ or more individual points. (It is also worth noting that we will treat each separating copy of $\mathcal{S}_B$ as a single object so far as Γ^{D_s} is concerned, even though it is an infinite number of points. We justify this by saying that we can easily either set a canonical method of establishing an isomorphism between such copies that we follow in our definitions of Γ^{D_s} or, if we prefer, we could construct our copies of $\mathcal{S}_B$ in stages along with the rest of $\mathcal{S}_0$ and $\mathcal{S}_1$ so that we are only changing Γ^{D_s} on a finite number of points at a time, but that the structure grows into the desired type of shuffle sum in the limit - either method works.)

For our first undiagonalization, we need to restore $\Gamma^{D_s}(x_{e,i}) = \hat{x}_{e,i}$ and $\Gamma^{D_s}(y_{e,i}) = \hat{y}_{e,i}$. Graphically, we need to slide $\hat{x}_{e,i}$ and $\hat{y}_{e,i}$ to be directly under $x_{e,i}$ and $y_{e,i}$:

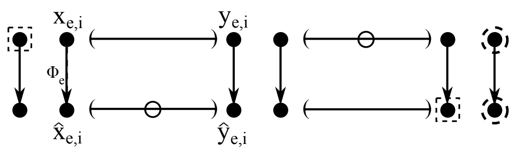
Figure 1.2.3:



Immediately Before First Undiagonalization

This leaves two places where points will need images or preimages when we revert Γ, as well as a point in each shuffle some whose suborder is a singleton that needs a successor to be a pair - we indicate these missing points with dotted circles. To undiagonalize at stage s, fill in the gaps as necessary (where the newly added copies of $\mathcal{S}_B$ are marked with circles), and shift Γ^{D_s} to be vertical:

Figure 1.2.4:



After First Undiagonalization

Note that we maintain the same definition of the neighborhood of a witness that we used in the proof of Theorem 1.1.3, namely the collection of points added while diagonalizing near a particular witness together with the witness itself. This means that in the case of shuffle sums, our neighborhoods can be quiet complex, including entire copies of $\mathcal{S}_B$ and generally being rather large. This means that they are not neighborhoods in the usual order sense, but the word still defines a portion of our order associated with the witness, and in some ways viewable in isolation.

As the process of diagonalizing and undiagonalizing repeats, we do much the same thing. To see that this is always possible, we use the idea of sliding and spreading out the orders during diagonalization. Each point in the neighborhood of our witness will only ever have (at most) two images defined for different $D_s \upharpoonright \gamma_{e,i}$: one for when we are diagonalized and one for when we are not. The order relation on each set of points does not change once it is defined.

This leads to an induction: If all points a and b in the neighborhood of our witness before the n^{th} diagonalization/undiagonalization satisfy that if $a < b$ in $\mathcal{S}_{0,s}$, then

$\Gamma^{D_s}(a) < \Gamma^{D_s}(b)$ in $\mathcal{S}_{1,s}$ (for each s for which $\Gamma^{D_s}(a)$ and $\Gamma^{D_s}(b)$ are defined), then when it comes time to diagonalize (or undiagonalize) at stage t we can "spread out" the orders so that a is directly above $\Gamma^{D_t}(a)$ for each a such that $\Gamma^{D_t}(a)$ is defined. This may leave many "holes", or points that need either images or preimages, and so we fill those in with new points and define Γ^{D_t} accordingly.

After we do so, we still have that if $a < b$ in $\mathcal{S}_{0,s}$, then $\Gamma^{D_s}(a) < \Gamma^{D_s}(b)$ in $\mathcal{S}_{1,s}$ for all such images as are defined, which means that we can do it again when it comes time for the $(n+1)^{th}$ diagonalization/undiagonalization. Since we trivially satisfy the base case (before the first induction, only $x_{e,i}$ and $y_{e,i}$ are near $x_{e,i}$), this means that that we can always change between the definitions of Γ^D that we've made so far without violating the order.

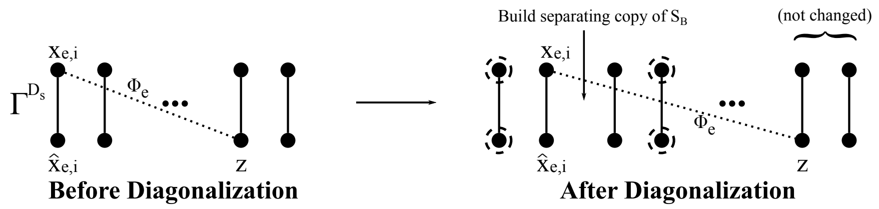
Further, we can carry this out in such a way that $x_{e,i}$ is always a right point and that, in the diagonalized stages, $\hat{x}_{e,i}$ is always a left point, and thus that we are actually diagonalized. Should we follow the "lining up" portion of procedure in a way that fails to do this, we can always add the appropriate number of successors or predecessors surrounding each point and then separate them off into the appropriate pairs. If $\hat{x}_{e,i}$ would be a right point, for example, we can separate it from its predecessor by copies of $\mathcal{S}_B$ then give it and its old predecessor new successors. Since we control where we place the new points when raising singletons to pairs in the lining up portion of the procedure in the first place, we should be able to avoid having to address this situation separately, but it turns out that we do not need to be especially careful to do so.

Case 2, $\Phi_e(x_{e,i}) \downarrow = z \neq \hat{x}_{e,i}$: We can assume that $z \in \mathcal{S}_{1,s}$ (if not, just pause this requirement until we see it enter).

Here we will have three subcases depending on the potential for injury. The third will be handled differently; but in each of the first two we take advantage of the fact that we can change $x_{e,i}$ to be a left or right point without changing any computations of Γ^{D_s} and without changing the position of z in its pair. This allows us to make sure that $x_{e,i}$ and z are never both left points or both right points at the same time (procedure below).

If $z \notin \mathbb{N}$, then check and see if $x_{e,i}$ and z satisfy "$x_{e,i}$ and z are currently either both initial points or both not initial points of their respective pairs." If we do not satisfy this statement, do nothing this stage - we're already diagonalized. If so, build separating copies of $\mathcal{S}_B$ between $x_{e,i}$ and its successor. Determine whether z is the initial point of its pair. If so, give $x_{e,i}$ and $\hat{x}_{e,i}$ predecessors; if not, give them successors. Add a point to be the successor of the now pairless point:

Figure 1.2.5:



This diagonalization works by forcing $x_{e,i}$ and $\hat{z}$ to never both be left or both be right points at the same time, making Φ_e fail to be an isomorphism.

Note that in this case we will remain diagonalized forever. Whether it is because $x_{e,i}$ is an initial point and $\hat{z}$ is not or vice versa will change each time the shuffle sum changes near z, which will only ever happen if z was added as part of a diagonalization procedure. But such changes will only happen finitely often since only diagonalizations from Case 1 will change whether a given non-natural is an initial point of a pair more than once. These changes only happen when D changes below a certain use, and D is Δ^0_2. So in this subcase, R_e will never choose another witness and we will win forever in a way that will stabilize. This fact that z can only change between being a left or right point finitely often is crucial to making this case succeed in diagonalizing, and ensuring that this we always have this finiteness is why we have three subcases of Case 2.

(Note that diagonalizations of this type do not change whether z is an initial point or not. Only the requirement that added z to the orders, if there is such a requirement, has the ability change whether z is a left or right point; so we need not worry about a case of two requirements taking turns switching z between being a left or right point infinitely often in an attempt to win.)

If $z \in \mathbb{N}$ and if $z \neq \hat{x}_{k,i}$ for any higher priority requirement R_k then diagonalize as above.

If z was a (hatted) witness for a lower priority R_n, tell R_n to no longer use z as a witness. The construction may still have to change computations to match definitions made by R_n, so that Γ^D will be well defined, but it is now required to always add points and begin separations to keep $z \in \mathcal{S}_1$ in whatever state (left or right point) it is in at this stage.

In some cases, it may still be possible for R_n to successfully use z as a witness, but we could have a situation where $\Phi_e(x_{e,i}) = \Phi_n(x_{n,j}) = \Phi_m(x_{m,k}) = \Phi_e(x_{e,i})$ so that we end up with a triangle issue where (following the equality from left to right) $x_{e,i}$ being a left point requires that $x_{n,k}$ is a right point, which requires that $x_{m,j}$ be a left point, which requires that $x_{e,i}$ be a right point (a contradiction), etc, so that the three requirements can never all be met at once.

If we did try to diagonalize in such a case then each requirement would, in changing its witness from being left to right, change a "z point" for another requirement from being left to right and hence necessitate that the other requirement change its witness from right to left. In this way, we could end up with our requirements continually forcing each other to flip their witnesses from being left points to right or right points to left, in a way that would never stabilize and never have all requirements satisfied. (Note that for a shuffle sum that so that $\mathcal{N} \ni \omega$ we could change things so that this wouldn't be an issue, but even then there is no particular reason to do so.)

If z is a witness for a higher priority requirement, then do not change anything about either shuffle sum. We will never try to diagonalize at $x_{e,i}$ by adding points, but we will count the number of times R_e sees Φ_e send its witnesses to some higher priority (hatted) witness. If this has happened more than $\Sigma_{i<e}(n_i + 1)$ times (the total number of higher priority witnesses chosen by this stage) by our stage s, then Φ_e maps more points to higher priority witnesses than there are higher priority witnesses (so far), and so is not 1-1, hence not an isomorphism.

If R_e ever thinks it has won in this way, then it has won forever simply because we have seen Φ_e map a finite set onto a smaller finite set. If, excluding this subcase, it is not possible for a requirement to choose infinitely many witnesses (as will be shown), then a simple induction argument shows that if Φ_e is not 1-1, then R_e will not choose infinitely many witnesses either: R_0 has no higher priority requirements, so will not enter this subcase ever, and so (by assumption) will choose only finitely many witnesses. But then R_1 can only enter this subcase finitely often (the number of witnesses chosen by R_0), and (by the assumption, again) can only choose finitely many witnesses for other reasons, so will only choose finitely many total witnesses. Then R_2 can only enter this subcase finitely often, and so on.

Verification of Strategy: Our verification follows the same form as in Theorem 1.1.3. As mentioned above, all injury is finite, and in the same way as before it is sufficient to show that no n_e can increase forever. But this could only happen for exactly the same reasons:

For a fixed e, suppose then that n_e increases forever. Then γ_{e,n_e} and t_{e,n_e} also increase forever, and Γ cannot retain permission to diagonalize at any $x_{e,i}$.

If we chose infinitely many witnesses, then all but finitely many must try to diagonalize via case 1 (via the induction argument above), and hence we are in the same position as previously.

To compute $D(y)$: Run the construction until we find an i such that $y < \gamma_{e,i}$ and either $D_t(y) = 1$ or $D_t(y) = 0$ for all t between the corresponding $t_{e,i}$ and $t_{e,i+1}$ (and $x_{e,i}$ tries to diagonalize via case 1). If $D(y) \neq D_t(y)$ for all such t then we know that $D \upharpoonright \gamma_{e,i} \neq D_t \upharpoonright \gamma_{e,i}$ for any such t and so we must eventually get (and retain) permission to diagonalize at $x_{e,i}$. But by our supposition, this does not happen. So we must have $D(y) = D_t(y)$ for t in the interval.

Therefore, if n_e approaches infinity for any e, then D is computable. But D is not computable. Therefore n_e must have a finite limit for all e, and so all R_e are met.

$\square$

Notes on generalization of the proof: Since a shuffle sum (thought of as $\mathbb{Q}$ where each element is replaced with an assigned an order type) can have the order type of elements on a nowhere dense set changed to any other order type contained in $\mathcal{N}$ without changing the isomorphism type of the shuffle sum, it doesn't take much to modify this proof to work for any $\mathcal{N} \ni 2$. We can simply add the requirement to the construction of the initial $\mathcal{S}_B$ that each n in the nowhere dense set $\mathbb{N}$ be in a suborder of length 2. We would have to change “pair” to “suborder” in the case 2 diagonalization procedure, but this would cause no problems - the diagonalization would still work since it depended only on which points were first points and which weren’t, and the resulting structures would still be the appropriate type of shuffle sum.

Further, we used $\mathcal{N} = \{2\}$, but we could just as easily have used $\mathcal{N} = \{3\}$ or any other order that has an initial point - or for that matter, a final point, with another slight modification. This leads to a generalization:

Proposition 1.2.4. *Let $\mathcal{C}_{\mathcal{S},\mathcal{N}}$ be the class of shuffle sums of $\mathcal{N}$. If $\mathcal{N}$ contains at least one order $\mathcal{L}$ with a least or greatest element and with $|\mathcal{L}| \geq 2$, then no Δ^0_2 degree is low for $\mathcal{C}_{\mathcal{S},\mathcal{N}}$-isomorphism.*

Since the class of all shuffles sums contains shuffle sums of these types, then we get the following corollary.

Corollary 1.2.5. *Let $\mathcal{C}_{\mathcal{S}}$ be the class of all shuffle sums. Then no Δ^0_2 set is low for $\mathcal{C}_{\mathcal{S}}$-isomorphism.*

Proof: This follows immediately from Theorem 1.2.3 (or Proposition 1.2.4) and Proposition 0.1.7.

□

Since isomorphisms between shuffle sums are very far from unique, we do not so easily get a complete categorization result as we did in Corollary 1.1.4. As we will see in

Section 1.3, this uniqueness is not absolutely required to get a complete categorization, so it may be that there is a property of (perhaps a subclass of) shuffle sums that will play a similar role, but this is not yet known. We will discuss this further in the shuffle sum summary in Section 3.3.

1.3 Equivalence Structures and Δ_2^0 Degrees

Definition 1.3.1. *Let $\mathscr{L} = \{E\}$ be a language, where E is a binary relation symbol. An equivalence structure is an $\mathscr{L}$-structure satisfying the axioms stating that E is an equivalence relation.*

We now turn to equivalence structures. Rather than focus on the actual relation E itself, however, it tends to be easier to focus on the equivalence classes. Since asking whether two elements are in the same equivalence class is equivalent to asking if they are related by E, this doesn't change anything on the computability level, but it does allow us to build our equivalence structures by adding points to equivalence classes, and hence defining the relation E implicitly rather than explicitly.

Isomorphisms between equivalence structures are not in general unique, since there can be multiple equivalence classes of the same size, and even if the structures only have one class of each size, elements of any equivalence class can be mapped to any other element of the equivalence class of the corresponding size. In this way, we appear to be in a situation similar to the shuffle sums. For the class of all equivalence structures the situations are remarkably similar, as we will see in the coding constructions in Chapter 2. Further, like the class of shuffle sums, the class of equivalence structures can be restricted to subclasses with varying (known) behaviors.

In this section, we examine equivalence structures with exactly one equivalence class of each finite size. As mentioned, isomorphic copies of such structures will have many (uncountably many, in fact) isomorphisms between them, but we still have a quasi-uniqueness property: An isomorphism may be able to send a particular point to several different points, but each equivalence class in the domain structure must be mapped to the unique equivalence class of the same size in the range structure. This will turn out to be enough, and in fact will make this structure behave more similarly to the orders of type ω in Theorem 1.1.3 than the shuffle sums in Theorem 1.2.3, despite the uncountable number of isomorphisms.

Theorem 1.3.2. *Let $\mathcal{C}_{\mathcal{E}0}$ be the class of all equivalence structures which consist of exactly one equivalence class of each finite size. Then no non-computable Δ^0_2 set is low for $\mathcal{C}_{\mathcal{E}0}$-isomorphism.*

Proof: As before, we fix a non-computable Δ^0_2 D and construct two equivalence structures $\mathcal{E}_0$ and $\mathcal{E}_1$ such that $\mathcal{E}_0 \cong_D \mathcal{E}_1$, and $\mathcal{E}_0 \not\cong_{\Delta^0_1} \mathcal{E}_1$. We will initialize $\mathcal{E}_0$ and $\mathcal{E}_1$ to contain one equivalence class of each finite size in such a way that for every element, the (initial) size of its equivalence class is computable and we can find an element of each class based on its (initial) size. This will allow us to pick "large" witnesses from classes where we have not yet defined our isomorphism. As is common, we will use

$[x]$ to denote the equivalence class of x. We continue to use the notational convention of denoting the element of $\mathcal{E}_1$ with the same code as $x \in \mathcal{E}_0$ by $\hat{x}$.

We will also be building a Turing functional Γ so that Γ^D will end up being an isomorphism, and for reasons similar to before will make sure that the use of $\Gamma^D(x)$ in oracle independent and is the same on each element in a fixed initial equivalence class, while increasing towards infinity as the size of the initial equivalence classes increases towards infinity.

Construction: The general idea is much the same as the ω case - the main difference being adding a point to shift images becomes adding an equivalence class of the appropriate size and then increasing the sizes of other equivalence classes. The fact that D is Δ_2^0 will guarantee that everything remains finite.

Requirements:

R_e: Φ_e is not an isomorphism.

We will meet this requirement by ensuring that Γ^D is an isomorphism, and choosing witnesses $x_{e,i}$ and diagonalizing so that for some i, $\Phi_e(x_{e,i}) \notin [\Gamma^D(x_{e,i})]$.

Stage 0: Initialize $\mathcal{E}_0$ and $\mathcal{E}_1$ as described above.

Stage $s+1$: Give attention to R_0 up to R_s as required (see strategy below). For all x for which $\Gamma^{D_{s-1}}(x)$ is defined, and for which no R_e defined $\Gamma^{D_s}(x)$ this stage, define

$\Gamma^{D_s}(x) = \Gamma^{D_{s-1}}(x)$. For all $y \in \mathcal{E}_{0,s}$ with $|[y]| < s$ such that $y \in [x]$ for some $x \in \mathcal{E}_{0,0}$, and $\Gamma^{D_s}(x)$ is undefined, set $\Gamma^{D_s}(y) = y$.

The condition "for all $y \in \mathcal{E}_{0,s}$ with $|[y]| < s$ such that $y \in [x]$ for some $x \in \mathcal{E}_{0,0}$" is more complicated than the corresponding portion the ω construction in Theorem 1.1.3 because our requirements may have increased the size of our initial equivalence classes. All this does is to ensure that we define Γ^{D_s} on all y in the appropriate equivalence classes.

Strategy for R_e at stage s: Once again, R_e can need attention for three reasons, and possibly for several at once.

0. If R_e does not have any witnesses, choose a large witness $x_{e,0}$ from $\mathcal{E}_{0,0}$ and set $t_{e,0} = s$, $\Gamma^{D_s}(x_{e,0}) = \hat{x}_{e,0}$ (and define Γ^{D_s} to be the identity on the rest of $[x_{e,0}]$) and $n_e = 0$ with large use $\gamma_{e,i}$.

1. If $n_e > 0$, for each $0 \leq i < n_e$, check if the answer to "does $D_s \upharpoonright \gamma_{e,i} = D_t \upharpoonright \gamma_{e,i}$ for some t with $t_{e,i} \leq t \leq t_{e,i+1}$" is different from the answer to "does $D_{s-1} \upharpoonright \gamma_{e,i} = D_t \upharpoonright \gamma_{e,i}$ for some t with $t_{e,i} \leq t \leq t_{e,i+1}$". For each i, if there is such a change, then follow the diagonalization procedure below.

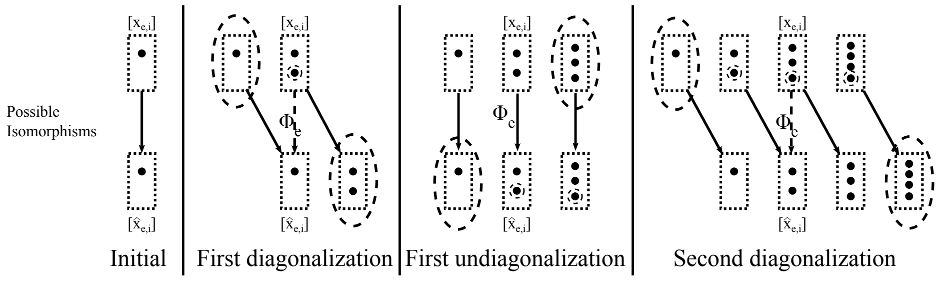
2. If $\Phi_{e,s}(x_{e,n_e}) \downarrow \in [\hat{x}_{e,n_e}]$ and either $n_e = 0$ or $\Gamma^{D_s}(x_{e,i}) \in [\hat{x}_{e,i}]$ for all $0 \leq i < n_e$, then increment n_e and, using the newly incremented n_e, remember the current stage number s as t_{e,n_e}, pick a new large witness x_{e,n_e} from $\mathcal{E}_{0,0}$, and define

$\Gamma^{D_s}(x_{e,n_e}) = \hat{x}_{e,n_e}$ (and define Γ^{D_s} to be the identity on the rest of $[x_{e,n_e}]$) with large use $\gamma_{e,i}$.

Diagonalization procedure for 1.: To diagonalize, we make $|[x_{e,i}]| \neq |[\hat{x}_{e,i}]|$. Since we are only trying to diagonalize at $x_{e,i}$ if $\Phi_e(x_{e,i}) \in [\hat{x}_{e,i}]$, this will ensure Φ_e does not respect the size of equivalence classes and so is not an isomorphism.

The diagonalization procedure we use to accomplish this follows a pattern similar to adding points from the ω case, except that adding a point to the left of the neighborhood translates to adding a new class of the same size as the smallest class in the neighborhood and increasing the size of all other equivalence classes of size greater than or equal to this newly added class by one. Similarly with adding a point on the right:

Figure 1.3.1:



In Figure 1.3.1, the rectangles represent equivalence classes. Entire classes or individual points that are circled are added at that stage. Note that the figure only shows the neighborhood of $[x_{e,i}]$, what is not shown is that we must add a point to every single equivalence class of of larger size each time we diagonalize or undiagonalize (and define Γ^{D_s} accordingly, in each equivalence class for which $\Gamma^{D_{s-1}}$ is already defined on some member). However since D is Δ^0_2, no equivalence class will gain more than finitely many points in this way, so we maintain both the uniqueness of classes of a given size and the finiteness of all classes.

The solid arrows in Figure 1.3.1 indicate which equivalence classes a possible isomorphism may map to each other. In particular, Φ_e follows the dashed arrow, and so cannot be an isomorphism in the diagonalized stages. We build Γ^D in such a way that it always follows the solid lines (details below).

Note that, in this construction, the neighborhood of a witness $x_{e,i}$ refers to all those points in $[x_{e,i}]$ together with those points in the equivalences classes that were added as part of diagonalizing at $x_{e,i}$ (and similarly for $\hat{x}_{e,i}$). In particular, the neighborhood does not include the points that we add to equivalence classes (other than $[x_{e,i}]$) that were not themselves added as part of the diagonalization.

The diagonalization procedure in more detail is as follows: If R_e just gained permission to diagonalize at stage s, and if the minimum size of the equivalence classes in the neighborhood of $x_{e,i}$ is m and the maximum size of such classes is M then,

1. For each class of size greater than or equal to m in $\mathcal{E}_0$, add one new point.

2. Add a new class of size m to $\mathcal{E}_0$.

3. For each class of size greater than or equal to $M + 1$ in $\mathcal{E}_1$, add one new point.

4. Add a new class of size $M+1$ to $\mathcal{E}_0$.

5. For each class in $\mathcal{E}_0$ containing some element on which $\Gamma^{D_{s-1}}$ is defined, define Γ^{D_s} to map this class to the class of the same size in $\mathcal{E}_1$.

   a. For points in the neighborhood of any $x_{n,j}$, give Γ^{D_s} use $\gamma_{n,j}$.

   b. For other points, give Γ^{D_s} the same use as other members of their respective equivalence classes.

The m and M are to ensure that we add classes on the "left" and "right" of the neighborhoods respectively, and hence maintain that each class in $\mathcal{E}_0$ has at most two images defined under Γ^{D_s} for differing s. To see why we do this, we think of these classes as lined up in order of their size. The requirement that we have a unique class of each size means that if we want to add a class between classes of size k and $k+1$, we have to increase the sizes of all classes of size $k+1$ or larger by one point. This can be thought of as "shifting" these classes over to make room for a new class of size $k+1$.

In this way, step 1 in the procedure above can be visualized as shifting all equivalence classes (in $\mathcal{E}_0$) one unit to the right, starting from the far left of the neighborhood of $x_{e,i}$. This makes room for a new class of size m on the far left of the neighborhood. Step 2 adds that class. Likewise, step 3 makes room for a class of size $M+1$ in $\mathcal{E}_1$, and step 4 adds that class.

Step 5 defines Γ^{D_s} in exactly the way we would expect, making sure to give all points the appropriate use: we do not want to add points to the neighborhood of some other

$x_{n,j}$ that cannot have their images shifted when R_n needs to diagonalize or undiagonalize there.

Similarly, to undiagonalize at stage s, we do the following:

1. For each class of size greater than or equal to $M + 1$ in $\mathcal{E}_0$, add one new point.
2. Add a new class of size $M + 1$ to $\mathcal{E}_0$.
3. For each class of size greater than or equal to m in $\mathcal{E}_1$, add one new point.
4. Add a new class of size m to $\mathcal{E}_0$.
5. For each class in $\mathcal{E}_0$ containing some element on which $\Gamma^{D_{s-1}}$ is defined, define Γ^{D_s} to map this class to the class of the same size in $\mathcal{E}_1$.
   a. For points in the neighborhood of any $x_{n,j}$, give Γ^{D_s} use $\gamma_{n,j}$.
   b. For other points, give Γ^{D_s} the same use as other members of their respective equivalence classes.

This procedure functions much the same as the diagonalization procedure, except that we make room on the right in $\mathcal{E}_0$ and on the left in $\mathcal{E}_1$, so as to make $|[x_{e,i}]| = |[\hat{x}_{e,i}]|$ once again.

Verification of strategy for R_e: We follow much the same pattern as in the ω case in Theorem 1.1.3. Should R_e stop choosing new witnesses, we still win the requirement for the same reason: R_e can not change its mind about why it thinks it is diagonalized on a finite set of witnesses infinitely often, and if it ever thinks that it is not diagonalized then it picks a new witness. It remains to show that R_e cannot choose infinitely many witnesses.

Suppose R_e does choose infinitely many witnesses, and hence γ_{e,n_e} and t_{e,n_e} approach infinity. To compute $D(y)$: Run the construction until we find an i such that $y < \gamma_{e,i}$ and either $D_t(y) = 1$ or $D_t(y) = 0$ for all t between the corresponding $t_{e,i}$ and $t_{e,i+1}$.

If it were not the case that $D(y) = D_t(y)$ for all such t, then we know that $D \restriction \gamma_{e,i} \neq D_t \restriction \gamma_{e,i}$ for any such t and hence we must eventually get (and retain) permission to diagonalize at $x_{e,i}$ (via reason to diagonalize 1.). But by our supposition, this does not happen. So we must have $D(y) = D_t(y)$ for t in the interval.

Therefore D is computable. But this is a contradiction, hence our R_e cannot choose infinitely many witnesses, and we win R_e.

$\square$

Note that in this proof, it was not crucial that we have classes of each size - we could have done much the same proof for even any infinite c.e. set of sizes (and perhaps more complicated sets of sizes) with just a little more work: We do the diagonalization as above, but treat these initial classes as kind of a tail. If A is our c.e. set of sizes, then each time we see an n enter A above the smallest size of a class in the tail, we increase all sizes of classes in the tail until the smallest has size n, and no longer consider that class, now of size n, to be in the tail. If we see n enter A below the smallest size of the class in the tail, we simply add a class of that size and then ignore it from then on. To maintain the correct sizes, instead of increasing the size of each class by one when the diagonalization procedure needs to make room for another class, we will increase the size of each class to the size of the next larger class. This will be important in Chapter 3. In some ways, this is analogous to the shuffle sum proof, where we did not exactly need $\mathcal{N} = \{2\}$, only that $\mathcal{N}$ contain a well order or anti-well order.

We might expect the fact that there are uncountably many isomorphisms between any two structures in $\mathcal{C}_{\mathcal{E}0}$ to cause the construction to behave similarly to the construction we used for shuffle sums, but it didn't actually have much impact on the proof. In the shuffle sum case, we had to handle cases where a computable function Φ_e mapped the witness $x_{e,i}$ not only to an element different from $\hat{x}_{e,i}$, but possibly even to some other $\hat{x}_{n,j}$, which led to several complications. When dealing with equivalence structures, if $\Phi_e(x_{e,i}) \notin [\hat{x}_{e,i}]$, we just won; and if $\Phi_e(x_{e,i}) \in [\hat{x}_{e,i}]$ but $\Phi_e(x_{e,i}) \neq \hat{x}_{e,i}$ we did exactly the same thing we would have done had $\Phi_e(x_{e,i}) = \hat{x}_{e,i}$.

In this way, though there is definitely not a unique isomorphism, the fact that each equivalence class has a unique possible image under isomorphism is a sort of quasi-uniqueness property that plays a similar role. This quasi-uniqueness property causes behavior very similar to what we saw in the ω case - even to the point of leading towards a complete categorization corollary similar to Corollary 1.1.4.

Corollary 1.3.3. *A degree* **a** *is not low for $\mathcal{C}_{\mathcal{E}0}$-isomorphism if and only if* **a** *computes* *a non-computable Δ_2^0 degree.*

Proof: ($\Leftarrow$): In the same way as before, this direction is an obvious result of Proposition 0.1.6 together with Theorem 1.3.2.

($\Rightarrow$): This time we take advantage of both the uniqueness of equivalence classes of a given size in our structures, and the fact that elements in the same equivalence class are algebraically indistinguishable - that is, so long as a function respects equivalence classes and is 1-1 and onto, it is an isomorphism regardless of what it does with the individual elements.

We start by defining the set $A_{\mathcal{E}_0,\mathcal{E}_1} = \{\langle x, y\rangle : \alpha(x) = y \text{ for some isomorphism } \alpha : \mathcal{E}_0 \to \mathcal{E}_1\}$. Note that this is equivalent to the set $\{\langle x, y\rangle : (|[x]| = |[y]|) \wedge (x \in \mathcal{E}_0) \wedge (y \in \mathcal{E}_1)\}$, and is also the union of the graphs of all isomorphisms.

If $\mathcal{E}_0$ and $\mathcal{E}_1$ are isomorphic, $A_{\mathcal{E}_0,\mathcal{E}_1}$ can always compute an isomorphism between them: simply do a back and forth argument, making sure to respect the equivalence relation at each stage. In particular, this means that if $\mathcal{E}_0 \ncong_{\Delta_1^0} \mathcal{E}_1$, then $A_{\mathcal{E}_0,\mathcal{E}_1}$ is not computable.

Further, if there is exactly one equivalence class of each size then any isomorphism can compute $A_{\mathcal{E}_0,\mathcal{E}_1}$: for a fixed isomorphism f, $\langle x, y\rangle \in A_{\mathcal{E}_0,\mathcal{E}_1}$ if and only if $f(x) \in [y]$ (and $[y]$ is computable), since $[y]$ is the unique equivalence class in $\mathcal{E}_1$ of the same size as $[x]$ in $\mathcal{E}_0$.

And finally, if $\mathcal{E}_0$ and $\mathcal{E}_1$ have only finite equivalent classes, then $\mathbf{0}'$ can always compute an isomorphism between them. For any x (since our sizes are finite), $\mathbf{0}'$ can compute $|[x]|$ by alternating between asking if there is another element in the class, then finding it if so. $\mathbf{0}'$ can then compute an isomorphism by finding equivalence classes of the same size and matching them up.

In particular, for any $\mathcal{E}_0$ and $\mathcal{E}_1$ in $\mathcal{C}_{\mathcal{E}0}$, $\mathbf{0}'$ can compute an isomorphism between them, and that isomorphism can compute $A_{\mathcal{E}_0,\mathcal{E}_1}$. Which means that $A_{\mathcal{E}_0,\mathcal{E}_1}$ is Δ^0_2.

By definition, any degree $\mathbf{a}$ that is not low for $\mathcal{C}_{\mathcal{E}0}$-isomorphism must compute some isomorphism f between two equivalence structures $\mathcal{E}_0$ and $\mathcal{E}_1$ that are not computably isomorphic. And since f is an isomorphism, it must compute $A_{\mathcal{E}_0,\mathcal{E}_1}$, which is a non-computable Δ^0_2 set by virtue of being Turing below $\mathbf{0}'$ and being able to compute an isomorphism between the non-computably isomorphic structures itself.

Therefore any degree $\mathbf{a}$ that is not low for $\mathcal{C}_{\mathcal{E}0}$-isomorphism computes some non-computable Δ^0_2 set.

$\square$

A few notes: First, this corollary is similar to the Corollary 1.1.4 not just in its statement, but also in its proof. The set $A_{\mathcal{E}_0,\mathcal{E}_1}$ is the union of all isomorphisms between $\mathcal{E}_0$ and $\mathcal{E}_1$; in the ω case the corresponding union of all isomorphisms $A_{\leq_0,\leq_1}$ is the union of a set of exactly one isomorphism. This set $A_{\leq_0,\leq_1}$ satisfies exactly the same conditions as were required in the proof of 1.3.3: for structures that are not computably isomorphic, it can't be computable; it must be below $\mathbf{0}'$, and any degree that can compute an isomorphism must compute it. All three of these features were necessary in the proof of Corollary 1.1.4, they were just more obvious.

Second, the uniqueness of equivalence classes of a given size played a crucial role in the corollary, and we will see in Chapter 2 that if this condition is removed, Corollary 1.3.3 fails.

Third, the finiteness of the equivalence classes came into play in showing that $A_{\leq_0,\leq_1}$ is Δ^0_2, but this finiteness isn't strictly necessary. If we allow only finitely many equivalence classes of infinite size, we can obtain a similar result to Corollary 1.3.3:

Proposition 1.3.4. *Let $\mathcal{C}_{\mathcal{E}0'}$ be the class of equivalence structures with exactly one equivalence class of each finite size and finitely many classes of infinite size. A degree* $\mathbf{a}$ *is not low for $\mathcal{C}_{\mathcal{E}0'}$-isomorphism if and only if* $\mathbf{a}$ *computes a non-computable Δ^0_2 degree.*

Proof: ($\Leftarrow$): The proof of Theorem 1.3.2 still works if we allow any number of (even infinitely many) equivalence classes of infinite size, so long as we maintain the uniqueness of classes of finite size - we can just place these infinite classes in the initialization of each structure and then ignore them. Proposition 0.1.6 then gives us this direction.

($\Rightarrow$): We add a way of handling the classes of infinite size to the proof of this direction from 1.3.3: Since structures in $\mathcal{C}_{\mathcal{E}0'}$ have only finitely many equivalence classes of infinite size, for any isomorphic $\mathcal{E}_0$ and $\mathcal{E}_1$ in $\mathcal{C}_{\mathcal{E}0'}$ there is a finite (hence computable) set $I_{\mathcal{E}_0,\mathcal{E}_1}$ of pairs $\langle x, 0\rangle$ and $\langle y, 1\rangle$ for exactly one representative of each infinite equivalence class $[x]$ in $\mathcal{E}_0$ and $[y]$ in $\mathcal{E}_1$.

We can use this set $I_{\mathcal{E}_0,\mathcal{E}_1}$ to see that $A_{\mathcal{E}_0,\mathcal{E}_1}$ is still Δ^0_2. $\mathbf{0}' \oplus I_{\mathcal{E}_0,\mathcal{E}_1}$ computes $A_{\mathcal{E}_0,\mathcal{E}_1}$ as follows: To see if $\langle a, b\rangle \in A_{\mathcal{E}_0,\mathcal{E}_1}$, search $I_{\mathcal{E}_0,\mathcal{E}_1}$ to see if $a \in [x]$ for some x such that $\langle x, 0\rangle \in I_{\mathcal{E}_0,\mathcal{E}_1}$ and if $b \in [y]$ for some y such that $\langle y, 1\rangle \in I_{\mathcal{E}_0,\mathcal{E}_1}$. Since $I_{\mathcal{E}_0,\mathcal{E}_1}$ is finite, these searches are bounded, and so we do not need the $\mathbf{0}'$ oracle for this portion of the computation.

If both searches result in yes answers, then $\langle a, b\rangle \in A_{\mathcal{E}_0,\mathcal{E}_1}$, since $|[a]|$ and $|[b]|$ are both countably infinite. If exactly one of the searches returns a yes answer, then $\langle a, b\rangle \notin A_{\mathcal{E}_0,\mathcal{E}_1}$ since $|[a]|$ and $|[b]|$ are different. If neither search returns a yes answer, then $|[a]|$ and $|[b]|$ are both finite, so $\mathbf{0}'$ can determine if they are the same size and hence whether $\langle a, b\rangle$ is in $A_{\mathcal{E}_0,\mathcal{E}_1}$. Since $I_{\mathcal{E}_0,\mathcal{E}_1}$ is computable, $\mathbf{0}' \geq_T \mathbf{0}' \oplus I_{\mathcal{E}_0,\mathcal{E}_1} \geq_T A_{\mathcal{E}_0,\mathcal{E}_1}$.

Likewise, any isomorphism f can compute $f \oplus I_{\mathcal{E}_0,\mathcal{E}_1}$, which can compute $A_{\mathcal{E}_0,\mathcal{E}_1}$ in a similar way. Here we especially take advantage of the fact that $I_{\mathcal{E}_0,\mathcal{E}_1}$ can directly compute whether an element of $\mathcal{E}_0$ or $\mathcal{E}_1$ is in an equivalence class of infinite size.

And finally, a simple back and forth argument shows that $A_{\mathcal{E}_0,\mathcal{E}_1}$ can still always compute an isomorphism.

Since any degree **a** that is not low for $\mathcal{C}_{\mathcal{E}0'}$-isomorphism must compute an isomorphism between non-computably isomorphic members of $\mathcal{C}_{\mathcal{E}0'}$, and this isomorphism must compute the non-computable Δ^0_2 set $A_{\mathcal{E}_0,\mathcal{E}_1}$, this completes the proof.

$\square$

In the proof of Proposition 1.3.4, the restriction to a finite number of infinite equivalence classes guarantees that we can compute whether or not a particular element is in an infinite class. Without this restriction, this need not be possible. In such a situation we can still place an upper bound on the set $A_{\mathcal{E}_0,\mathcal{E}_1}$ (namely $\mathbf{0}''$), however it is no longer necessarily the case that every isomorphism can compute $A_{\mathcal{E}_0,\mathcal{E}_1}$, and so the proof breaks down.

And finally, though it is not currently known whether or not there is some trick that will result in a complete categorization in the shuffle sum case in Theorem 1.2.3, trying to directly define a similar set, $A_{\mathcal{S}_0,\mathcal{S}_1}$ to be the union of all isomorphisms between $\mathcal{S}_0$ and $\mathcal{S}_1$ does not immediately work, since that set is not always obviously able to compute an isomorphism. $A_{\mathcal{S}_0,\mathcal{S}_1}$ would be able to successfully match up left points with left points and right points with right points, and it can respect the order while doing so, but this is not enough since such a function need not respect the successor relation. Neither is it obviously the case that any isomorphism can compute $A_{\mathcal{S}_0,\mathcal{S}_1}$. This is discussed further in Section 3.3.

Chapter 2

Coding Constructions

We will now use coding techniques to obtain further results for less restricted classes of structures. These techniques can be simpler but, particularly in the case of shuffle sums, do not always give us as much control over the resulting structure. The basic idea behind these techniques is to code information about appropriate sets into the isomorphisms between structures (but not necessarily into a single structure alone).

2.1 Linear Orders of Type ω and Δ^0_2 Degrees: Coding Method

We begin by presenting an alternate proof of Theorem 1.1.3 for the purpose of illustrating the coding process in a simple case. In this proof, we will code a Δ^0_2 set into the isomorphism between two specific copies of ω. Once again, the uniqueness of this isomorphism from Lemma 1.1.2 will simplify things, however the role is less important here than it was previously. In fact, in future constructions we will actively work to destroy any sort of uniqueness property in some cases.

First we note that ω is an example of a scattered linear order:

Definition 2.1.1. *A linear order $\mathcal{L}$ is scattered if and only if the linear order $\mathbb{Q}$ does not embed into $\mathcal{L}$.*

We will see a distinction between the interaction of scattered linear orders and the (non-scattered) shuffle sums with the coding techniques used in this chapter. Recall:

Theorem 1.1.3. *No non-computable Δ_2^0 set D is low for ω-isomorphism.*

Proof: Fix an arbitrary non-computable Δ_2^0 set D with approximation stages D_s. Assume $D_0(n) = 1$ for all n. (This allows us to write simpler conditions for the coding cases by ensuring that every n leaves D before it enters D.)

As before, we initialize $\leq_0$ and $\leq_1$ to be identical copies of ω, ensuring that we can compute the n^{th} point of each order at this initial stage for every n. However, rather than diagonalize against every computable isomorphism individually, we will instead construct our orders so that the unique isomorphism has the same degree as the set D. This will both guarantee that $\leq_0$ and $\leq_1$ are not computably isomorphic, and that they are D-isomorphic.

Let x_n and $\hat{x}_n$ denote the n^{th} points of our initial orders $\leq_{0,0}$ and $\leq_{1,0}$ respectively. We will build our orders so that if α is the unique isomorphism, then $n \in D$ if and only if $\alpha(x_n) = \hat{x}_n$. Since D is assumed non-computable, this will show that $\leq_0$ and $\leq_1$ are not computably isomorphic. This will also ensure that D can compute the unique isomorphism between $\leq_0$ and $\leq_1$, which will complete the proof.

Construction:

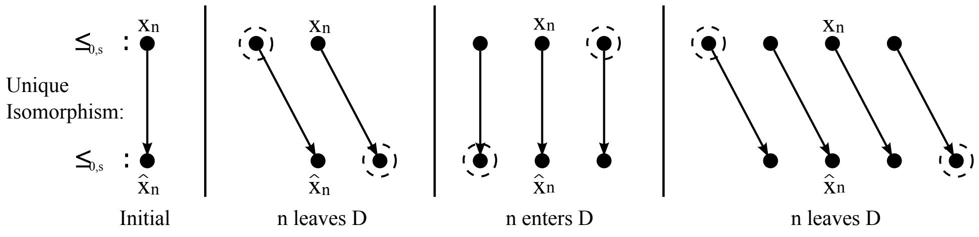
Stage 0: Initialize $\leq_{0,0}$ and $\leq_{1,0}$ as described.

Stage $s+1$: For each n, add points near x_n as necessary according to the cases below.

Case 1: If $n \in D_s \setminus D_{s-1}$, then add one point to the right of the neighborhood of x_n and another to the left of the neighborhood of $\hat{x}_n$.

Case 2: If $n \in D_{s-1} \setminus D_s$, then add one point to the left of the neighborhood of x_n and another to the right of the neighborhood of $\hat{x}_n$.

Figure 2.1.1:



Since D is Δ^0_2, this process will eventually stabilize, guaranteeing that we will maintain order type ω.

Note that Figure 2.1.1 is very similar to Figure 1.1.2 from the diagonalization proof of Theorem 1.1.3. The procedure for adding points is in fact identical, only the conditions for when to follow this procedure have changed: rather than explicitly building an isomorphism, and then changing the orders to try force this isomorphism to be different from a specific Φ_e at a witness, we instead change the orders in such a way that the isomorphism will be able to compute D in the limit.

It's worth emphasizing that we are not explicitly constructing an isomorphism, but rather modifying the structures so that any (the) isomorphism α satisfies $n \in D_s$ if and only if $\alpha(x_n) = \hat{x}_n$. The uniqueness of isomorphisms between copies of ω means that there isn't much difference in this case, but in the more complicated proofs to follow where we handle showing that multiple (possibly uncountably many) sets aren't low for $\mathcal{C}$-isomorphism (for a particular $\mathcal{C}$) per construction, this is fairly important.

Verification: By construction, the unique isomorphism α can compute D via $n \in D$ if and only if $\alpha(x_n) = \hat{x}_n$.

It remains to show that D can compute an isomorphism. Let Γ^D be defined as follows: For each n,

Case $n \in D$: Define $\Gamma^D(x_n) = \hat{x}_n$. Then watch the construction. We know that each time the neighborhood grows as in Case 2 in the construction, that it will later grow as in Case 1, so simply wait until it does so, and define Γ^D appropriately.

Case $n \notin D$: Watch the construction and wait until the first time we add points via Case 2 (which must happen if $n \notin D$). Set $\Gamma^D(x_n)$ equal to the new successor of $\hat{x}_n$, and likewise map the new predecessor of x_n to $\hat{x}_n$. Then each time the neighborhood grows via Case 1, we know that it will later grow via Case 2, so simply wait until it does so, and define Γ^D appropriately.

This shows that $\leq_0 \cong_D \leq_1$ and $\leq_0 \not\cong_{\Delta^0_1} \leq_1$. Therefore D is not low for ω-isomorphism.

$\square$

One might try to extend this method to showing that no non-computable Σ^0_2 set D is low for $\mathcal{C}_{\mathcal{L}}$ isomorphism. We could do much the same construction, and though we wouldn't end up with an order of type ω because of the potentially infinite oscillation between Cases 1 and 2 in the construction, we would end up with a well defined linear order - the only difference being that some neighborhoods would grow to be $\mathbb{Z}$ when $n \notin D$. Further, D would still be able to compute an isomorphism in exactly the same way.

The problem would come when we tried to compute D from an isomorphism. If the neighborhood of x_n (hence $\hat{x}_n$) grew to be a copy of $\mathbb{Z}$, there would be many different ways to map the neighborhood of x_n to the neighborhood of $\hat{x}_n$. In particular, there would both be isomorphisms that both send x_n to $\hat{x}_n$ and to some other element of the neighborhood of $\hat{x}_n$. In this way, our method above would have some isomorphisms that thought that n was in D, and others that thought it was not. This destroys are method of showing that all isomorphisms must compute D, which in turn destroys our method for showing that the two structures were not computably isomorphic.

So the exact method we used here would not work, and in fact we know from Theorem

0.2.3 that there is a Σ^0_2 set that is low for isomorphism, so no other trick will work either.

However, if we move from Σ^0_2 sets to separating sets D of pairs of computably inseparable Σ^0_2 sets A and B, this is no longer an issue. We may still end up with an infinite amount of oscillation that causes portions of our structure to grow into copies of $\mathbb{Z}$, but a separating set will (except in one special case) generally have values n so that the value of $D(n)$ does not affect whether or not D is a separating set. A and B will interact in such a way that we can ensure that the infinite growth and hence the loss of the (local) uniqueness of the isomorphisms only occurs on these n. In this way, every isomorphism will compute a separating set D, though perhaps not the same one, and since A and B were computably inseparable, this ensures that no isomorphism is computable.

This is the general strategy of the separating set proofs that follow, and in fact it works for a scattered linear order construction remarkably similar to the construction we just saw. However, this scattered linear order construction has an additional complication, so we delay scattered linear orders until Section 2.4, and use equivalence structures to demonstrate how using separating sets affects the constructions.

2.2 Equivalence Structures and Separating Sets

Definition 2.2.1. *Let A and B be disjoint subsets of $\mathbb{N}$. We say that D separates A and*

B iff $D \supseteq A$ and $D \cap B = \emptyset$.

This definition does depend on the order in which we are given A and B, but since we are ultimately interested in computability properties, the fact that "D separates A and B" is equivalent to "$\bar{D}$ separates B and A" (and $D \cong_T \bar{D}$) means that the difference between separating A and B and separating B and A does not cause problems.

As mentioned at the end of Section 2.1, using separating sets in our coding constructions allows us to have varying isomorphisms and still maintain that each isomorphism computes some (non-computable) separating set. However, if we wish to show that no separating set is low for isomorphism, we also need to show that each of these (possibly uncountably many) sets computes an isomorphism. To handle this, we end up not only allowing for varying isomorphisms but requiring them.

For this reason, we use equivalence structures as our first example. We will present two results using linear orders as well, but the added structure of linear orders gives rise to complications that equivalence structures avoid.

Theorem 2.2.2. *Let $\mathcal{C}_{\mathcal{E}}$ be the class of all equivalence structures. No separating set for any pair of computably inseparable Σ^0_2 sets A and B is low for $\mathcal{C}_{\mathcal{E}}$-isomorphism.*

Proof: Fix computably inseparable Σ^0_2 sets A and B with construction stages A_s and B_s.

We will construct two equivalence structures $\mathcal{E}_0$ and $\mathcal{E}_1$ such that any isomorphism α between them can compute a set D that separates A and B (and hence that $\mathcal{E}_0$ and

$\mathcal{E}_1$ are not computably isomorphic, since A and B are computably inseparable). We will then verify that our construction also allows any separating set D to compute an isomorphism, which will prove the theorem.

Construction:

We initialize $\mathcal{E}_{0,0}$ to contain distinct equivalence classes $[x_n] \ni x_n$ and $[y_n] \ni y_n$ of size p_n for each prime p_n, in such a way that we can compute $\{x_n\}$ and $\{y_n\}$ as sequences. We initialize $\mathcal{E}_{1,0}$ to be identical to $\mathcal{E}_{0,0}$, but with elements corresponding to x_n and y_n denoted by $\hat{x}_n$ and $\hat{y}_n$.

We will then watch each n enter and leave A and B, and increase the sizes of equivalence classes appropriately so that if α is an isomorphism, then the set D defined by $n \in D$ if and only if $\alpha(x_n) \in [\hat{x}_n]$ is a separating set. We will also make sure that our construction requires neither $\alpha(x_n) \in [\hat{x}_n]$ nor $\alpha(x_n) \notin [\hat{x}_n]$ when $n \notin (A \cup B)$ (and hence there are separating sets that both contain and do not contain n) for the purposes of ensuring that any separating set computes an isomorphism.

Each time we need to increase the size of an equivalence class of size p_n^k for some n and k, we will increase its size to p_n^{k+1}. It is not crucial that we use sizes of prime power, but it allows us to avoid having changes in the size of one equivalence class affect the size of another. Unlike in Theorem 1.3.2, equivalence classes may grow infinitely often, so we do need some method avoiding most of this interaction - otherwise an infinite number of changes at x_n could destroy the coding at x_m for all $m > n$.

Stage 0: Initialize as described above

Stage $s+1$: Check if $n \in A_s$ and if $n \in B_s$, and add points as required according the cases below. (See Figure 2.2.1.)

Assume that at each stage, and for each n, $\max(\{|[x_n]|, |[\hat{x}_n]|, |[y_n]|, |[\hat{y}_n]|\}) = p_n^k$.

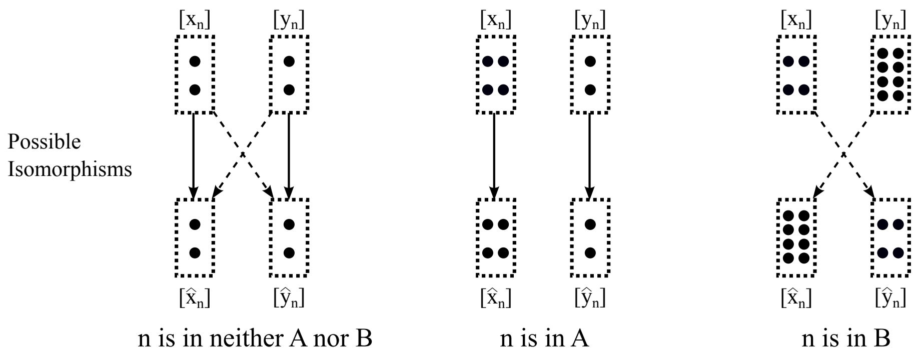
Case 0: $n \in A_s$ and $n \in B_s$. Then do nothing this stage. We know that A and B are disjoint, so this condition will not persist.

Case 1: $n \in A_s$ and $n \notin B_s$, and $|[x_n]| = |[\hat{y}_n]|$. Then add points as required to raise $|[x_n]|$ and $|[\hat{x}_n]|$ to size p_n^{k+1}, and $|[y_n]|$ and $|[\hat{y}_n]|$ to size p_n^k.

Case 2: $n \notin A_s$ and $n \in B_s$, and $|[y_n]| = |[\hat{y}_n]|$. Then add points as required to raise $|[y_n]|$ and $|[\hat{x}_n]|$ to size p_n^{k+1}, and $|[x_n]|$ and $|[\hat{y}_n]|$ to size p_n^k.

Case 3: $n \notin A_s$ and $n \notin B_s$. Then add points as required to raise each of $|[x_n]|$, $|[\hat{x}_n]|$, $|[y_n]|$, and$|[\hat{y}_n]|$ to size p_n^k.

Figure 2.2.1:



Isomorphisms that map $[x_n]$ to $[\hat{x}_n]$ and $[y_n]$ to $[\hat{y}_n]$ are said to map vertically, in reference to the representation of the equivalence classes in Figure 2.2.1. Likewise, isomorphisms that map $[x_n]$ to $[\hat{y}_n]$ and $[y_n]$ to $[\hat{x}_n]$ are said to map diagonally.

Each case of the construction ensures that for a given n, our construction is in one of the three configurations shown in Figure 2.2.1. Ignoring Case 0 situations, Case 1 changes the structure so that only vertical isomorphisms are possible when $n \in A_s$, Case 2 only allows diagonal isomorphisms when $n \in B_s$, and Case 3 allows both vertical and diagonal isomorphisms when $n \notin (A_s \cup B_s)$. Note that only Cases 1 through 3 add points to the structures, and that no case adds points twice in a row. (If Case 1 wants to make it so that only vertical isomorphisms is possible, and this is already the case, it doesn't do anything, and similarly with the other cases.)

Since A and B are Σ^0_2, it is possible for our construction to switch between Case 1 and Case 2 infinitely often, causing all four equivalence classes to grow to infinite size. However, this can only happen when $n \notin (A \cup B)$, since if $n \in A$ then $n \in A_s$ for all sufficiently large s (because A is Σ^0_2). This means that after we reach this stage, the construction will only switch between Case 1 and Case 0 - which means that we will stop adding points. Likewise if $n \in B$.

Verification: First we show that $\mathcal{E}_0 \not\cong_{\Delta^0_1} \mathcal{E}_1$. Let α be an arbitrary isomorphism from $\mathcal{E}_0$ to $\mathcal{E}_1$. It suffices to show that the set D defined by $n \in D$ if and only if $\alpha(x_n) \in [\hat{x}_n]$ is a separating set, since A and B are computably inseparable and α computes D.

If we do not have infinite oscillation, this is clear from the construction: n must be in any separating set if $n \in A$, and if $n \in A$ we must have $\alpha(x_n) \in [\hat{x}_n]$. Hence $n \in D$ since $[x_n]$ and $[\hat{x}_n]$ are the only equivalence classes of their size in each structure, so we have $n \in D$ when required. Similarly, if $n \in B$ then $\alpha(x_n) \notin [\hat{x}_n]$, so $n \notin D$ for all n for which n must not be in any separating set.

And as mentioned, we only have infinite oscillation if $n \notin (A \cup B)$ - but in this case the value of $D(n)$ does not affect whether D is a separating set. It is possible in such a case that $\alpha(x_n) \in [x_m]$ for some $m \neq n$ if we had infinite oscillation for both n and m, but in this case, we simply don't place n in D, and since $n \notin (A \cup B)$ this is not an issue.

Therefore any isomorphism computes a separating set.

It remains to show that any separating set D can compute an isomorphism. But we can do this simply by defining an isomorphism Γ^D using a back and forth argument (so that we do not miss points in the case of infinite equivalence classes), where if $n \in D$ we map vertically and if $n \notin D$ we map diagonally. Again, we know from the construction that if $n \in D$, then isomorphisms can map vertically, since $n \in D$ implies that either $n \in A$ (in which case vertical isomorphism are required) or $n \notin (A \cup B)$ (in which case vertical isomorphisms are possible). Similarly if $n \in B$.

Therefore any separating set computes an isomorphism.

$\square$

Since every Δ^0_2 set is Σ^0_2, we get the following corollary:

Corollary 2.2.3. *No separating set for any pair of computably inseparable Δ^0_2 sets A and B is low for $\mathcal{C}_{\mathcal{E}}$-isomorphism.*

But we can actually do better: in the proof of Theorem 2.2.2, we only ended up with equivalence classes of infinite size when at least one of A and B changed its mind infinitely often on some n. If A and B were Δ^0_2, then this would not happen, and our witnessing structures will be in the class of equivalence structures with no infinite equivalence classes. This leads to the following proposition:

Proposition 2.2.4. *Let $\mathcal{C}_{\mathcal{E}1}$ be the class of equivalence structures with no infinite equivalence classes. Then no separating set for any pair of computably inseparable Δ^0_2 sets A and B is low for $\mathcal{C}_{\mathcal{E}1}$-isomorphism.*

Further, for every non-computable Δ^0_2 D, both D and $\bar{D}$ are Σ^0_2 and so form a computably inseparable pair of Σ^0_2 sets. Since D separates D and $\bar{D}$, we get the following corollary:

Corollary 2.2.5. *No Δ^0_2 is low for $\mathcal{C}_{\mathcal{E}}$-isomorphism.*

We could also get a version of Corollary 2.2.5 from Proposition 2.2.4. Both versions are less general Theorem 1.3.2, however - we present this corollary to demonstrate that a corollary of this form will always result from a proof that no separating set of computably inseparable Σ^0_2 sets is low for $\mathcal{C}$-isomorphism.

And as a final corollary:

Corollary 2.2.6. *No separating set for any pair of computably inseparable c.e. sets A and B is low for $\mathcal{C}_{\mathcal{E}}$-isomorphism.*

If we take Corollary 2.2.6 from Proposition 2.2.4, we can restrict $\mathcal{C}_{\mathcal{E}}$ to $\mathcal{C}_{\mathcal{E}1}$, and in fact we can actually restrict to an even smaller class if we like: if we run the construction in Theorem 2.2.2 with c.e. sets A and B, we will only be able to grow each equivalence class at most once. This means that while we still have arbitrarily large equivalence classes, we have no infinite classes and have more control over which finite sizes will be in the resulting structures. Following the construction from Theorem 2.2.2 will result in equivalence structures with at most two classes of size p_n and at most one class of size p_n^2, though we could control the sizes in any number of ways.

2.3 Shuffle Sums and Separating Sets

Recall that a shuffle sum over a set of suborders $\mathcal{N} = \{\mathcal{L}_n\}$ is obtained by partitioning $\mathbb{Q}$ into a collection of dense sets $\{\mathcal{Q}_n\}_{n\in\omega}$, and replacing each point in $\mathcal{Q}_n$ with the suborder $\mathcal{L}_n$.

In Chapter 1, the subclass of shuffle sums were the most complicated of the structures that we considered, to the point that the construction for the (scattered) order of type ω was more closely related to the construction for the subclass of equivalence structures $\mathcal{C}_{\mathcal{E}0}$ than to the construction for its fellow linear orders. For the coding arguments, we see this relation flip: now shuffle sums behave more like equivalence structures, and scattered orders behave oddly.

This is a result of the density of the sets $\mathcal{Q}_n$ in the partition. In Theorem 1.2.3, we did the diagonalization on the individual suborders of the shuffle sum. Now we remove the restriction that $\mathcal{N} = \{2\}$, and so we can use the sizes of the suborders associated with each $\mathcal{Q}_n$ for coding in much the same way that we used the sizes of the equivalence classes in Theorem 2.2.2. Density comes into play when it comes time to match up the suborders.

Suppose we are building two shuffle sums $\mathcal{S}_0$ and $\mathcal{S}_1$, and at some stage $\mathcal{S}_0$ contains the suborders $A_i < ... < B_j < ... < C_k$, and that $\mathcal{S}_1$ contains corresponding suborders $\hat{A}_i < ... < \hat{B}_j < ... < \hat{C}_k$ of the same length (at this stage). So long as we maintain that $\mathcal{S}_0$ and $\mathcal{S}_1$ have the same associated $\mathcal{N}$ (hence isomorphism type), we can change the length of B_j in $\mathcal{S}_0$ and be guaranteed that there is a suborder of the new length of B_j between $\hat{A}_i$ and $\hat{C}_k$ in $\mathcal{S}_1$ (on either side of $\hat{B}_j$). We can also find a suborder in $\mathcal{S}_1$ of the same length as $\hat{B}_j$ on both the right and left of B_j, and so there is still an isomorphism. We will be changing the order types of all suborders associated with a particular $\mathcal{Q}_n$, but the principle is much the same.

Theorem 2.3.1. *No degree that computes a separating set for computably inseparable Σ_2^0 A and B sets is low for $\mathcal{C}_{\mathcal{S}}$-isomorphism.*

Proof: Recall that the choice of dense sets in the partition does not affect the isomorphism type. We take advantage of this by instead partitioning $\mathbb{Q}$ into uniformly computable dense sets $\{\mathcal{Q}_n, \mathcal{P}_n\}_{n\in\omega}$, with a fixed, uniformly computable $x_n \in \mathcal{Q}_n$, and initially setting points in both $\mathcal{Q}_n$ and $\mathcal{P}_n$ to all have finite order type of length p_n (the n^{th} prime).

This will still be a shuffle sum, since the union of two dense sets is dense. We will then change the order types of the points in $\mathcal{Q}_n$ and $\mathcal{P}_n$ as we see n enter and leave our fixed A and B, so that for any isomorphism α, the set defined by "$D \ni n$ if and only if $\alpha(x_n) \in \hat{\mathcal{Q}}_n$" separates A and B. This will also allow any separating set to compute an isomorphism, as we will verify. Note that x_n will always be an initial point

As before, rather than replacing points with orders, our construction starts with $\mathbb{Q}$ and then adds points to bring the order types of each point in each $\mathcal{Q}_n$ and $\mathcal{P}_n$ to the correct length. Since all of our orders $\mathcal{L}_n$ are well orders, we will always add point on the right (both initially and throughout the construction this time) so that all of the rationals, particularly each x_n, will always be an initial point of their orders. It is worth noting that this feature of the construction will make the successor relation computable in the resulting orders, even though this need not generally be the case for shuffle sums.

Construction:

Stage 0: Initialize to shuffle sums $\mathcal{S}_0$ and $\mathcal{S}_1$ as described above, ensuring that $\mathcal{Q}_n$, $\mathcal{P}_n$, and $\{x_n\}$ are uniformly computable. Denote the subsets of $\mathcal{S}_1$ corresponding to $\mathcal{Q}_n$ and $\mathcal{P}_n$ in $\mathcal{S}_0$ by $\hat{\mathcal{Q}}_n$ and $\hat{\mathcal{P}}_n$.

Stage $s+1$: For each n, check if $n \in A_s$ and if $n \in B_s$, and add points as required according to the cases below.

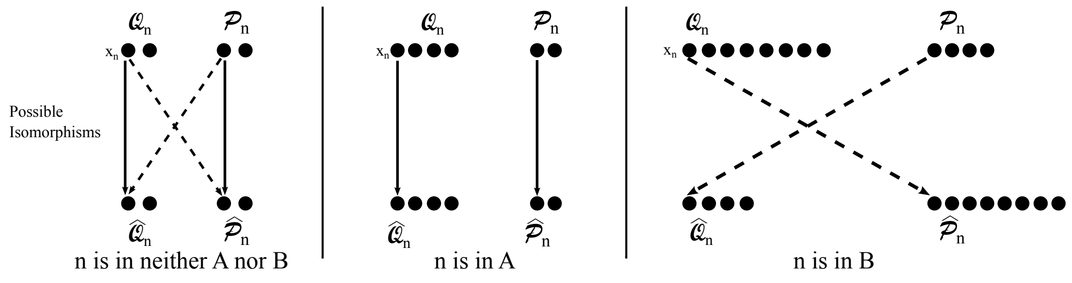
Case 0: If $n \in A_s$ and $n \in B_s$, do nothing this stage. We know that A and B are disjoint, so this condition will not persist.

Case 1: If $n \in A_s$ and $n \notin B_s$ and if the lengths of the orders in $\mathcal{Q}_n$ are equal to the lengths of the orders in $\hat{\mathcal{P}}_n$ and are both p_n^k for some k, then add points as necessary so that the lengths of the orders in $\mathcal{Q}_n$ and $\hat{\mathcal{Q}}_n$ are p_n^{k+1}, and the orders in $\mathcal{P}_n$ are $\hat{\mathcal{P}}_n$ all length p_n^k.

Case 2: If $n \notin A_s$ and $n \in B_s$, and if the lengths of the orders in $\mathcal{Q}_n$ are equal to the lengths of the orders in $\hat{\mathcal{Q}}_n$ and are both p_n^k for some k, then add points as necessary so that the lengths of the orders in $\mathcal{Q}_n$ and $\hat{\mathcal{P}}_n$ are p_n^{k+1}, and the orders in $\mathcal{P}_n$ are $\hat{\mathcal{Q}}_n$ all length p_n^k.

Case 3: If $n \notin A_s$ and $n \notin B_s$, then add points as required to bring the lengths of all orders in $\mathcal{Q}_n$, $\mathcal{P}_n$, $\hat{\mathcal{Q}}_n$ and $\hat{\mathcal{P}}_n$ up to the same length.

Figure 2.3.1:



(Note: Suborders of $\mathcal{Q}_n$ need not be less than suborders of $\mathcal{P}_n$. See explanation below.)

Recall that $\mathcal{Q}_n$ and $\mathcal{P}_n$ are dense in $\mathbb{Q}$. Figure 2.3.1 shows how the lengths of the orders in each dense set will grow, and indicates in each case whether points in $\mathcal{Q}_n$ are required to be mapped to points in $\hat{\mathcal{Q}}_n$ or $\hat{\mathcal{P}}_n$. The density condition means that there will always be both points of $\hat{\mathcal{P}}_n$ that are less than and greater than any particular point in $\hat{\mathcal{Q}}_n$, and so we can always match up the suborders that make up the shuffle sums so that the we respect the order relation both within each suborder and between suborders. The diagram does not indicate that an isomorphism will map suborders $N_i < M_j$ to $\hat{M}_j < \hat{N}_i$, but is only indicates which sets of the partition have suborders of the same length that can therefore be mapped to each other.

The condition in Case 1 that the lengths of the orders in $\mathcal{Q}_n$ are equal to the lengths of the orders in $\hat{\mathcal{P}}_n$ (and the similar condition in Case 2) are there so that we do not add points unnecessarily. Each case will add points when it sees the appropriate conditions on A_s and B_s if it was not the most recent case to add points.

Finally note that the lengths of all suborders will be finite unless we alternate between Case 1 and 2 infinitely often and so grow the suborders of points in $\mathcal{P}_n$ and $\mathcal{Q}_n$ to be copies of ω. As in Theorem 2.2.2, this can only happen if $n \notin (A \cup B)$. In particular, if $n \in A$ (and so if n must be in every separating set), then the lengths of all suborders of points in $\mathcal{Q}_n$ and $\hat{\mathcal{Q}}_n$ will all be the same finite size, and no other suborders will have that length. Likewise, if $n \in B$ (and so n cannot be in any separating set), the suborders of points in $\mathcal{Q}_n$ and $\hat{\mathcal{P}}_n$ will have the same finite length, and no other suborders will have that length.

Verification: First, we note that by construction, if α is an isomorphism, then the set defined by "$D \ni n$ if and only if $\alpha(x_n) \in \hat{\mathcal{Q}}_n$" is a separating set. The way in which we increased the sizes of the suborders guarantees that any $n \in A$ will be in such a D, and no $n \in B$ will be. (The cases where α can map x_n to an element of $\hat{\mathcal{Q}}_k$ or $\hat{\mathcal{P}}_k$ for $k \neq n$ only arise when $n \notin (A \cup B)$, and in those cases it does not matter whether or not we place n in D.) And since every isomorphism computes a separating set, and A and B are computably inseparable, this means that there are no computable isomorphisms.

It remains to show that every separating set can compute an isomorphism. Let D be an arbitrary set that separates A and B. Using a back and forth argument, define $\Gamma^D : \mathbb{Q} \to \mathbb{Q}$ to send all points in each $\mathcal{Q}_n$ to points in $\hat{\mathcal{Q}}_n$ if $n \in D$, and with those in $\hat{\mathcal{P}}_n$ if $n \notin D$; and to send all points in each $\mathcal{P}_n$ to those in $\hat{\mathcal{P}}_n$ if $n \in D$, and with $\hat{\mathcal{Q}}_n$ if $n \notin D$. Density allows us to do this in a way that preserves the order relation on $\mathbb{Q}$.

We can then extend Γ^D to be an isomorphism from $\mathcal{S}_0$ to $\mathcal{S}_1$ by watching the construction add points - each time we add a successor y to $x \in \mathcal{S}_0$, we know that $\Gamma^D(x) \in \mathcal{S}_1$ will also gain a successor, and that x and $\Gamma^D(x)$ will be in suborders of the same length. So we can define $\Gamma^D(y)$ to be the successor of $\Gamma^D(x)$ as soon as this successor is placed in the shuffle sum. Since (unlike in the diagonalization procedure in Theorem 1.2.3) we never change the successor once it is defined, and since every point in our final shuffle sums will either be an element of $\mathbb{Q}$ or some k^{th} successor of such an element, Γ^D will be an isomorphism.

□

Note that in way similar to Theorem 2.2.2, if we restrict A and B to be Δ^0_2, then there can be no infinite oscillation between Case 1 and Case 2 in the construction. This means that all suborders will be finite, and so the same construction gives the following result:

Proposition 2.3.2. *Let $\mathcal{C}_{S1}$ be the class of shuffle sums where $\mathcal{N} \subseteq \omega$. No degree that computes a separating set for computably inseparable Δ^0_2 sets A and B is low for $\mathcal{C}_{S1}$-isomorphism.*

Likewise, we can restrict A and B to be c.e. and obtain:

Corollary 2.3.3. *No degree that computes a separating set for computably inseparable c.e. sets A and B is low for $\mathcal{C}_{S1}$-isomorphism.*

Where $\mathcal{C}_{S1}$ can be restricted even smaller if desired, as in Section 2.2.

2.4 Scattered Linear Orders and Separating Sets

Our final coding argument will involve scattered linear orders, and will closely follow the proof of Theorem 0.2.2 by Franklin and Solomon, modified to work with Σ^0_2 sets and build a scattered order.

The coding argument is very similar to those in Theorems 2.2.2 and 2.3.1, with one exception. In a linear order, an isomorphism must respect the order of suborders, not just their size: an isomorphism cannot take groups of points A and B where all the points in A are less than those in B, and map A to $\hat{A}$ and B to $\hat{B}$ if all the points in $\hat{B}$ are less than those in $\hat{A}$. In the case of equivalence structures, there was no relation between distinct equivalence classes, so there was no analogue of this issue at all. In the case of shuffle sums, density allowed us to skirt this issue by finding suborders of the appropriate length where ever we cared to look.

In each of these cases, we could essentially consider equivalence classes of different size, or the collections of suborders associated with a $\mathcal{Q}_n$ or $\mathcal{P}_n$, as free floating objects - even when there were relations between them as in the case of the suborders in the shuffle sums, these relations had no real affect on the resulting structures.

The scattered linear order that we build will not have an analogue of density or any other way to be viewed as a collection of "free floating" substructures. This will primarily affect our construction in the case where we need to allow for multiple isomorphisms.

Theorem 2.4.1. *Let $\mathcal{C}_{Sc}$ be the class of scattered linear orders. No degree that computes a separating set for computably inseparable Σ^0_2 sets is low for $\mathcal{C}_{Sc}$-isomorphism.*

Proof: Fix two computably inseparable Σ^0_2 sets A and B with approximation stages A_s and B_s. We construct two linear orders $\leq_0$ and $\leq_1$ such that any separating set D can compute an isomorphism between $\leq_0$ and $\leq_1$ and any isomorphism can compute a separating set of A and B. Since A and B are computably inseparable, this will guarantee that $\leq_0$ and $\leq_1$ are not computably isomorphic.

Construction: We will initialize $\leq_0$ and $\leq_1$ to be identical copies of ω with the n^{th} elements denoted by by x_n and $\hat{x}_n$ respectively. We will then grow the orders as we see each n enter and leave A_s and B_s in such a way that, for any isomorphism α, the set D defined by "$n \in D$ if and only if $\alpha(x_n) = \hat{x}_n$" will always separate A and B. This will also allow us to define a Turing functional Γ such that Γ^D will be an isomorphism for any separating set D.

As before, we will use the phrases "near x_n" and "neighborhood of x_n" to refer to x_n and all of the points we have added for the purpose of affecting the isomorphisms at x_n. We will run the construction in such a way that any isomorphism must map the neighborhood of x_n to the neighborhood $\hat{x}_n$, so that each of these neighborhoods can be considered separately from each other.

Further, at each stage the neighborhoods of all x_n and $\hat{x}_n$ will be finite, so that there will be a unique isomorphism between them, and in fact between $\leq_{0,0}$ and $\leq_{1,0}$. During the construction, we will check whether this unique isomorphism (at this finite stage) must map x_n to $\hat{x}_n$ or not as part of our conditions for adding more points. We will grow some neighborhoods to have the order type of $\mathbb{Z}$, but will maintain that any isomorphism must still map the neighborhood of x_n to the neighborhood $\hat{x}_n$ in the limit.

Stage 0: Initialize $\leq_{0,0}$ and $\leq_{1,0}$ to be identical copies of ω, with the n^{th} elements denoted by by x_n and $\hat{x}_n$ respectively.

Stage $s + 1$: For each n, check if $n \in A_s$ and if $n \in B_s$, and add points as required according to the cases below.

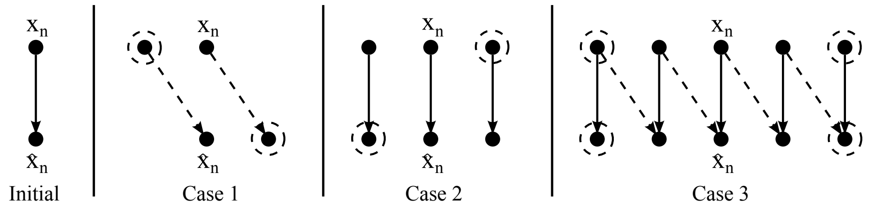
Case 0: If $n \in A_s$ and $n \in B_s$, do nothing this stage. We know that A and B are disjoint, so this condition will not persist.

Case 1: If $n \notin A_s$ and $n \in B_s$, and if the unique isomorphism (at this stage) maps x_n to $\hat{x}_n$, then add points on the far left of the neighborhood of x_n and on the far right of the neighborhood of $\hat{x}_n$, so that the isomorphism cannot map x_n to $\hat{x}_n$.

Case 2: If $n \in A_s$ and $n \notin B_s$ and if the unique isomorphism (at this stage) does not map x_n to $\hat{x}_n$, then add points on the far right of the neighborhood of x_n and on the far left of the neighborhood of $\hat{x}_n$, so that the unique isomorphism must map x_n to $\hat{x}_n$.

Case 3: If $n \notin A_s$ and $n \notin B_s$, then add a point both on the far left and right of both the neighborhood of x_n and $\hat{x}_n$.

Figure 2.4.1:



In Figure 2.4.1, the points each case adds are circled. The diagram shows Case 3 acting after Case 2, and so the dashed lines in Case 3 do not represent a possible isomorphism this stage, however in the limiting structure they will represent an isomorphism if the neighborhoods of x_n and $\hat{x}_n$ grow to be $\mathbb{Z}$. In particular, these dashed lines are not an isomorphism at any given stage, and so are not considered by the conditions for adding points in Cases 1 and 2. The conditions in these cases are there to ensure that Case 1 and 2 only add points if it's required to either force isomorphisms to or not to send x_n to $\hat{x}_n$.

The result is that Cases 1 and 2 will always add points alternately (with respect to each other, Case 3 may add points between them): Case 1 will add points the first time we see $n \notin A_s$ and $n \in B_s$, and then each time we see $n \notin A_s$ and $n \in B_s$ after Case 2 has added points (and Case 2 will act similarly). Case 3 will add points each stage s where $n \notin A_s$ and $n \notin B_s$ regardless of what else is going on.

Further, the neighborhoods of x_n and $\hat{x}_n$ will grow to be $\mathbb{Z}$ if and only if $n \notin A$ and $n \notin B$: If $n \in A$, then $n \in A_s$ for all s after some point, so we will only ever enter Case 1 or 0 after that point (and so will no longer add points near x_n or $\hat{x}_n$). Similarly if $n \in B$.

If $n \notin A$ and $n \notin B$, then we will see $n \notin A_s$ for an infinite number of stages s, and $n \notin B_t$ for an infinite number of stages t. If these stages coincide infinitely often, then we will enter Case 3 infinitely often, and so will add an infinite number of points on each side of x_n and $\hat{x}_n$, in a way that grows the neighborhoods towards $\mathbb{Z}$.

If the stages where $n \notin A_s$ and $n \notin B_t$ do not coincide infinitely often, then we must have that $n \notin A_s$ and $n \in B_s$ as well as $n \in A_t$ and $n \notin B_t$ infinitely often. This means we must switch between Case 1 and Case 2 infinitely often, which will also add an infinite number of points on each side of x_n and $\hat{x}_n$, growing the neighborhoods towards $\mathbb{Z}$.

And finally, note that a function mapping $\leq_0$ to $\leq_1$ is an isomorphism if and only if it is an isomorphism when restricted to each neighborhood of each x_n. We can see this via an induction argument: the neighborhoods of x_0 and $\hat{x}_0$ are either both the same finite size, or both copies of $\mathbb{Z}$ and are at the beginning of their respective orders, so that they must be mapped to each other. Then the neighborhoods of x_1 and $\hat{x}_1$ are either both the same finite size, or both copies of $\mathbb{Z}$ and are at the beginning of the rest of their respective orders, and so must be mapped to each other, and so forth.

Verification: First we verify that every isomorphism computes a separating set (hence there is no computable isomorphism). Let $\alpha : \leq_0 \rightarrow \leq_1$ be an isomorphism, and define D by $n \in D$ if and only if $\alpha(x_n) = \hat{x}_n$. By construction, if $n \in A$, then any isomorphism must have $\alpha(x_n) = \hat{x}_n$ and so $n \in D$. Similarly if $n \in B$ our construction prohibits $\alpha(x_n) = \hat{x}_n$, and so requires that $n \notin D$. (If $n \notin A$ and $n \notin B$, then our construction doesn't require $\alpha(x_n)$ to have any particular value except that it must be in the neighborhood of $\hat{x}_n$.) Since the construction requires that $n \in D$ if $n \in A$ and $n \notin D$ if $n \in B$, any set D so defined (for a fixed isomorphism α) separates A and B.

It remains to show that every separating set computes an isomorphism. Let D be a set that separates A and B. Construct Γ^D as follows: As our construction adds points near x_n, define Γ^D to follow the solid lines in Figure 2.4.1 if $n \in D$ and the dashed lines if not.

That is, if $n \in D$, we will map x_n to $\hat{x}_n$ then watch the construction to see if we add a new predecessor or successor to x_n as part of the coding for n. If we do, we know that we will eventually add a predecessor and successor to $\hat{x}_n$, and so we wait until that happens and map the successor of x_n to the successor of $\hat{x}_n$, and the same with their predecessors. Then we continue to watch the construction until (should it happen) the successor of x_n is given a new successor or the predecessor of x_n is given a new predecessor, at which point we wait until the same happens for the corresponding points near $\hat{x}_n$ and map the new successors to each other and the new predecessors to each other, and so on.

If $n \in D$, then either $n \in A$ or $n \notin (A \cup B)$. If $n \in A$, then the neighborhoods of x_n and $\hat{x}_n$ will both be finite and such that Γ^D will agree with the unique isomorphism between them. If $n \notin (A \cup B)$, then both neighborhoods will be copies of $\mathbb{Z}$, and, since Γ^D is a bijection between these neighborhoods that respects the successor, it will be an isomorphism in this case as well.

We construct Γ^D similarly if $n \notin D$, except with images shifted one point to the right (still mapping the neighborhood of x_n to the neighborhoods of $\hat{x}_n$). Γ^D restricted to neighborhoods of x_n for $n \notin D$ will be an isomorphism for similar reasons to the above.

Therefore Γ^D is an isomorphism between the neighborhoods of each x_n and corresponding $\hat{x}_n$, and hence an isomorphism between $\leq_0$ and $\leq_1$.

$\square$

As before, we required the construction to allow multiple (distinguishable) images under isomorphisms for each x_n if $n \notin (A \cup B)$. There will be separating sets $D_0 \ni n$ and $D_1 \not\ni n$, and so any uniform procedure for computing isomorphisms from separating sets will have to be able to define an isomorphism correctly whether n is in the separating set or not (without being able to know whether or not n is in A or B). In this particular proof, we accomplished this by growing neighborhoods to be copies of $\mathbb{Z}$. Such infinite growth will happen at least once so long as $A \neq \bar{B}$, which is a special case discussed below.

Further, these copies of $\mathbb{Z}$ were built into the order at x_n for each $n \notin (A \cup B)$, regardless of whether or not infinite oscillation was involved. Even if A and B had been Δ^0_2 or c.e., we would still have built copies of $\mathbb{Z}$. This means that while we still get corollaries corresponding to the cases where A and B are Δ^0_2 or c.e., the proof of Theorem 2.4.1 does not provide a method to restrict the resulting class of structures when we do so.

This is distinct from the behavior of the constructions in Theorems 2.2.2 and 2.3.1. The lack of a "free floating" mechanism meant that the neighborhood of x_n always had to be mapped to the neighborhood of $\hat{x}_n$, and so to allow x_n to be able to have distinct images within the neighborhood of $\hat{x}_n$required growing both neighborhoods to (at least) $\mathbb{Z}$.

In order for a point x_n to be able to have multiple images under different isomorphisms, it has to be part of some suborder that has no beginning or end point.

This means that even if we restrict A and B to be Δ^0_2 or c.e., the construction we used will result in structures of the same form (a concatenation of copies $\mathbb{Z}$ and finite orders). This contrasts with the results from looking at equivalence structures and shuffle sums, where restricting the complexity of A and B allows us to prove an analogous result for a smaller class of structures.

Though we have not shown a way to further restrict the class of structures as we restrict the sets A and B, Theorem 2.4.1 leads to several corollaries.

Corollary 2.4.2. *No degree that computes a separating set for computably inseparable Δ^0_2 sets A and B is low for $\mathcal{C}_{Sc}$-isomorphism.*

Proof: Every Δ^0_2 set is Σ^0_2.

□

Corollary 2.4.3. *No degree that computes a non-computable Δ^0_2 set is low for $\mathcal{C}_{Sc}$-isomorphism.*

Proof: Every non-computable Δ^0_2 set D is a separating set for the computably inseparable sets D and $\bar{D}$.

□

Corollary 2.4.3 is the only exception to the infinite growth discussed above. If we run the construction with $A = \bar{B}$, for any n we will only enter Case 3 finitely often (eventually n will enter A or B), and since both A and B are Δ^0_2, we will only switch between Case 1 and Case 2 finitely often. In this situation, our resulting structure will be a copy of ω - and in fact, if we run the construction in Theorem 2.4.1 with a Δ^0_2 set A and $B_s(n) = 1 - A_s(n)$, then we end up with a construction identical to that used to prove Theorem 1.1.3 in Section 2.1.

Corollary 2.4.4. *No degree that computes a separating set for computably inseparable c.e. sets A and B is low for $\mathcal{C}_{Sc}$-isomorphism. (Same as Theorem 0.2.2 (Franklin, Solomon)[3] with padding copies of $\mathbb{Q}$ removed.)*

Proof: Every c.e. set is Σ^0_2.

□

As mentioned, the proof of Theorem 2.4.1 follows a very similar construction to Franklin and Solomon's construction in their proof of Theorem 0.2.2. It is worth noting that the limiting structure will always be a sequence of copies of $\mathbb{Z}$ separated by arbitrarily large finite linear orders regardless of if our sets A and B are Σ^0_2, Δ^0_2, or c.e. sets. This results from the need to allow two possible images under isomorphism for x_n in the case that $n \notin (A \cup B)$.

Chapter 3

Summary and Examination of Results

There are essentially four properties that came into play for each of the three broad classes of structures that we built: the ability to grow parts of the structure while maintaining computability and structure type, the uniqueness (or quasi-uniquenes) of isomorphisms between copies of a structure, the ability to impose local (quasi-)uniqueness on the isomorphisms for given neighborhoods, and finally the ability to destroy such local uniqueness.

Each of the structures that we built had the ability to grow, though to differing degrees. We could add any arbitrarily large but finite number points to any neighborhood in the copies of ω we built, but we could not add infinitely many without destroying the order type. Likewise with new equivalence classes in $\mathcal{C}_{\mathcal{E}0}$ (though we could perhaps have defined $\mathcal{C}_{\mathcal{E}0}$ differently so that adding an infinite number of equivalence classes were possible, Theorem 1.3.2 did not require us to do so). The classes $\mathcal{C}_{\mathcal{E}}$, $\mathcal{C}_{\mathcal{S}}$, and $\mathcal{C}_{\mathcal{S}c}$ all allowed for infinite growth, but in many cases we could restrict to a smaller class of structures that disallowed neighborhoods to grow infinitely (with varying definitions of neighborhood) and still obtain similar results.

Global isomorphism uniqueness, or the quasi-uniqueness in the case of $\mathcal{C}_{\mathcal{E}0}$, was important to complete categorization, whereas the ability to create some sort of local uniqueness was required to be able to code. The ability to destroy this local uniqueness, either at a specific stage or in the limiting structure, allowed the constructions for separating sets, where we essentially wanted to code positive information about two computably inseparable sets A and B without directly coding any negative information (except that which could be determined from positive information and the fact that A and B were disjoint).

We built equivalence structures, scattered linear orders and a specific type of non-scattered linear order known as a shuffle sums. We might have expected the two types of linear orders to behave similarly, however our results are dissimilar enough that we handle these two subclasses of linear orders separately. Here we present our results, examine which results we do not have but might expect to have, and discuss some of the issues that would have to be overcome in obtaining such results.

3.1 Equivalence Structures

We proved the following results for equivalence structures:

(i) No degree that computes a separating set for computably inseparable Σ_2^0 sets is low for $\mathcal{C}_{\mathcal{E}}$-isomorphism. (Theorem 2.2.2).

(ii) No degree that computes a separating set for computably inseparable Δ_2^0 sets is low for $\mathcal{C}_{\mathcal{E}1}$-isomorphism. (Proposition 2.2.4).

(iii) No degree that computes a separating set for computably inseparable c.e. sets is low for $\mathcal{C}_{\mathcal{E}1'}$-isomorphism. (Corollary 2.2.6, modified as discussed).

(iv) A degree **a** is not low for $\mathcal{C}_{\mathcal{E}0}$-isomorphism if and only if **a** computes a non-computable Δ^0_2 degree. (Corollary 1.3.3).

We start with equivalence structures because the results have the fewest complications. We first note that only (iv) is a complete categorization. Proving (iv) was somewhat more complicated than showing the analogous result in the case of scattered linear orders, but was accomplished using a quasi-uniqueness property: there are many isomorphisms, but each equivalence class (for pairs of structures in $\mathcal{C}_{\mathcal{E}0}$) can have exactly one image.

Results (i) - (iii) are not known to be complete categorizations, and in fact, it is not even known whether (i) holds if $\mathcal{C}_{\mathcal{E}}$ is replaced by $\mathcal{C}_{\mathcal{E}1}$ or $\mathcal{C}_{\mathcal{E}1'}$ (though (iv) gives us that it does not hold with $\mathcal{C}_{\mathcal{E}0}$). Each of these results relied on the ability to impose and then destroy a local version of the quasi-uniqueness property used in (iv): we could add points so that each equivalence class had either one or two possible images under isomorphism. Infinite growth only arose when we had to create and destroy this property infinitely many times (in distinction with our proof in the case of scattered linear orders), and so we were able to further restrict the class of equivalence structures in the cases of computably inseparable for Δ^0_2 or c.e. sets.

While we do not know if (i) could be proved for the class of structures in (ii) or (iii), we do see a nice relationship between the structures and the computably inseparable sets. We expect a proof involving Σ_2^0 sets in this way to involve some sort of infinite oscillation, which in these proofs tends to lead towards infinite growth, so it would be somewhat surprising if we could prove an analogue of (i) using only equivalence structures with no equivalence classes of infinite size. Likewise, we expect proofs involving Δ_2^0 sets to involve (at least) arbitrarily large but finite growth, and proofs involving c.e. sets to involve (at least) very limited growth. If the coding could be switched from purely relying on the sizes of the equivalence classes to somehow relying on the number of equivalence classes, then it may be possible to further restrict the classes of structures in (i) and (ii), but at this time no such method is known.

3.2 Scattered Linear Orders

We proved the following results for scattered linear orders:

(i) No degree that computes a separating set for any pair of computably inseparable Σ_2^0 sets A and B is low for $\mathcal{C}_{Sc}$-isomorphism. (Theorem 2.4.1).

(ii) No degree that computes a separating set for any pair of computably inseparable Δ_2^0 sets A and B is low for $\mathcal{C}_{Sc}$-isomorphism. (Corollary 2.4.2).

(iii) No degree that computes a separating set for any pair of computably inseparable c.e. sets A and B is low for $\mathcal{C}_{Sc}$-isomorphism. (Corollary 2.4.4).

(iv) A degree **a** is not low for ω-isomorphism if and only if **a** computes a non-computable Δ^0_2 degree. (Corollary 1.1.4).

Note that only result (iv) is known to be a complete categorization (the only if direction resulting from the uniqueness of isomorphisms between copies of ω and the fact that $\mathbf{0}'$ can always compute the successor). It is not currently known whether result (i) can be extended to a complete categorization though the proof of (i) shows that (ii) and (iii) cannot be.

Results (ii) and (iii) are direct corollaries of the (i). The other types of structures that we examined each have similar results, but we were able to restrict to smaller subclasses in each case by modifying the proofs of the analogues of result (i).

In the construction in (i), we used only linear orders that could be obtained by replacing each point in ω with either a copy of $\mathbb{Z}$ or a finite sequence of points. As discussed in Section 2.4, our proof method requires that the orders we build even in cases (ii) and (iii) contain distinct suborders without endpoints for each $n \notin (A \cup B)$. As part of this, even the finite sequences between copies of $\mathbb{Z}$ could grow to be arbitrarily long, even if A and B were Δ^0_2 or c.e.

There does not appear to be an easy way around this. We could try to modify the method so that a single construction and Turing functional does not have to show every D separating A and B fails to be low for $\mathcal{C}_{Sc}$-isomorphism, but a counting argument shows that there will still be at least one pair of structures and Turing functional that have to work for uncountably many D, and so we are likely to have the same issue. We might try to modify the construction so that we can control the lengths of the finite sequences that are not contained in any copy of $\mathbb{Z}$ more carefully, but the obvious ways of doing so destroy either our method of computing isomorphisms from separating sets, or the computability of the structure itself.

This is not to say that no smaller class of structures can satisfy results (ii) or (iii) (or (i) for that matter), only that any construction would have to overcome these issues.

3.3 Shuffle Sums

We proved the following results for shuffle sums:

(i) No degree that computes a separating set for computably inseparable Σ^0_2 sets is low for $\mathcal{C}_{\mathcal{S}}$-isomorphism. (Theorem 2.3.1).

(ii) No degree that computes a separating set for computably inseparable Δ^0_2 sets is low for $\mathcal{C}_{\mathcal{S}1}$-isomorphism. (Proposition 2.3.2).

(iii) No degree that computes a separating set for computably inseparable c.e. sets is low for $\mathcal{C}_{\mathcal{S}1'}$-isomorphism. (Corollary 2.3.3, modified as discussed).

(iv) No degree that computes a non-computable Δ^0_2 degree is low for $\mathcal{C}_{\mathcal{S}0}$-isomorphism. (Theorem 1.2.3).

Unlike in the cases of scattered linear orders and equivalence structures, we have not shown that (iv) is a complete categorization. Even if we use the trick from equivalence structures and define $A_{\mathcal{S}_0,\mathcal{S}_1} = \{\langle x, y\rangle : y = f(x)$ for some isomorphism $f : \mathcal{S}_0 \to \mathcal{S}_1\}$, we have problems with two of the three requirements needed for this to give us a complete categorization corollary. It is true that $\mathbf{0}'$ can compute any such $A_{\mathcal{S}_0,\mathcal{S}_1}$ for $\mathcal{S}_0, \mathcal{S}_1 \in \mathcal{C}_{\mathcal{S}0}$, but we have issues both computing an isomorphism from $A_{\mathcal{S}_0,\mathcal{S}_1}$ and having any isomorphism compute $A_{\mathcal{S}_0,\mathcal{S}_1}$.

In this case, $A_{\mathcal{S}_0,\mathcal{S}_1}$ is exactly the pairs of points $\langle x, y\rangle$ with $x \in \mathcal{S}_0$ and $y \in \mathcal{S}_1$ where x and y are either both left points or both right points. But the fact that there is an isomorphism mapping x to y does not tell us which it is, so a single isomorphism (in particular, an isomorphism of some non-computable Δ^0_2 degree below the degree of the successor relation) has no particular way of determining whether or not two points that it does not map to each other can be mapped to each other. Further, even knowing which points are all left points and which are all right points is not sufficient to compute an isomorphism, since it would be easy to map a left point x to a left point $\hat{x}$, and map the successor of x to some right point that is not the successor of $\hat{x}$.

These were problems that equivalence structures did not have, since elements of the same equivalence class are algebraically indistinguishable, whereas elements of the same suborder, or even elements with the same positions in suborders of the same length, are not.

When we lifted the restriction on the lengths of the suborders and looked at separating sets as in (i), however, shuffle sums began to behave very similarly to equivalence structures, where the collections of suborders associated with each dense $\mathcal{Q}_n$ or $\mathcal{P}_n$ began to take on a role similar to that of equivalence classes $[x_n]$ and $[y_n]$. The distinction between these collections of suborders and equivalence classes is most apparent in that we had to use a back and for argument to match up the the initial points of the suborders associated with $\mathcal{Q}_n$ in the constructions for each of the results (i) through (iii), where as equivalence structures only needed a back and for argument in their version of case (i).

On shuffles sums, "local uniqueness" of the isomorphism took two forms: In the proof of (iv), this role was filled by the property that given two particular suborders of the same length in $\mathcal{S}_0$ and $\mathcal{S}_1$, there is a unique isomorphism between those suborders. In the proof of (i), we used a less specific property that any isomorphism can only map points of $\mathcal{Q}_n$ to points of $\hat{\mathcal{Q}}_n$ if the suborders associated with each are of the same length - all other restrictions on isomorphisms were made simple to satisfy by the construction.

When the construction in (i) required us to destroy uniqueness, we could do this by just increasing the length of suborders associated with $\mathcal{Q}_n$ or $\mathcal{P}_n$. This meant that we only had to increase the lengths of these orders when the sets A and B we were separating actually changed, which in turn meant that if the sets A and B were restricted to be Δ_2^0 or c.e., we could change these lengths less often and so prove (ii) and (iii) for smaller collections of structures.

Note though, that our construction actually made the successor function partial computable in this construction, and in general the successor can be more complicated than that. This means that trying to extend results (i) through (iii) to complete categorizations, even if such ended up being possible in the case of equivalence structures, may run into the same issues we had attempting to do so with (iv).

3.4 Relationship between Subclass and Degree Progression

Our major results for each of our structures are theorems of the form (with the appropriate subclasses of structures inserted):

Theorem Templates 3.4.1.

(i) No separating set for computably inseparable Σ_2^0 sets is low for $\mathcal{C}$-isomorphism.

(ii) No separating set for computably inseparable Δ_2^0 sets is low for $\mathcal{C}_1$-isomorphism.

(iii) No separating set for computably inseparable c.e. sets is low for $\mathcal{C}_{1'}$-isomorphism.

(iv) No degree that computes a non-computable Δ_2^0 set is low for $\mathcal{C}_0$-isomorphism.

(v) Any degree that is not low for $\mathcal{C}_0$-isomorphism computes a non-computable Δ_2^0 set.

Where equivalence structures and scattered linear orders have a result of the form of each template, and shuffle sums have results of the form (i) through (iv).

In the case of equivalence structures, for example, we can let $\mathcal{C} = \mathcal{C}_{\mathcal{E}}$, $\mathcal{C}_1 = \mathcal{C}_{\mathcal{E}1}$, and $\mathcal{C}_0 = \mathcal{C}_{\mathcal{E}0}$, and let $\mathcal{C}_{1'}$ be one of the many possible such subclasses mentioned at the end of Section 2.2. Then each of the templates above becomes an actual proven result (In order: Theorem 2.2.2, Proposition 2.2.4, appropriately modified Corollary 2.2.6, Theorem 1.3.2, and Corollary 1.3.3.)

But while each of these results is proven, we do not know if we could also let $\mathcal{C} = \mathcal{C}_{\mathcal{E}1}$ instead of $\mathcal{C}_{\mathcal{E}}$, for instance, and still prove Template (i). Likewise, in the case of shuffle sums, we do not know if Template (v) holds for $\mathcal{C}_{\mathcal{S}0}$. In this way the known relationships between the relevant classes $\mathcal{C}$, $\mathcal{C}_1$, $\mathcal{C}_{1'}$, and $\mathcal{C}_0$ vary for each of the broad classes of structures. In each case, we have subclasses of structures to fill each role, but beyond that less is clear.

We do, however, have some general relationships between these classes. Let $\mathcal{C}$, $\mathcal{C}_1$, $\mathcal{C}_{1'}$, and $\mathcal{C}_0$ all be subclasses of the same class of structures, such that Templates (i) through (iv) (but not necessarily (v)) hold. Then we can put an order on the "strengths" of each subclass with respect to these templates, as well as on the templates themselves.

Definitions 3.4.2.

- *We say that Template (x) $\succeq$ Template (y) if and only if any class of structures that satisfies Template (x) also satisfies Template (y).*
- *Let $\mathcal{C}_\alpha$ and $\mathcal{C}_\beta$ be subclasses of the same class of $\mathscr{L}$-structures for some $\mathscr{L}$. Then we say $\mathcal{C}_\alpha \succeq \mathcal{C}_\beta$ if and only if each of Templates (i) through (iv) that is satisfied*

by $\mathcal{C}_\beta$ is also satisfied by $\mathcal{C}_\alpha$.

Simple computability properties give us some information on these relations immediately:

Proposition 3.4.3. *The following partial order on templates holds: (i) $\succeq$ (ii) $\succeq$ (iii) and (ii) $\succeq$ (iv).*

Further, let $\mathcal{C}$, $\mathcal{C}_1$, $\mathcal{C}_{1'}$, and $\mathcal{C}_0$ be subclasses as above satisfying (i) through (iv) in order, and suppose that $\mathcal{C}_{1'} \not\succeq \mathcal{C}_1$ and $\mathcal{C}_0 \not\succeq \mathcal{C}_1$. Then $\mathcal{C} \succeq \mathcal{C}_1 \succeq \mathcal{C}_0$, and $\mathcal{C}_1 \succeq \mathcal{C}_{1'}$.

Proof: These results, as well as transitivity of $\succeq$ follow from three computability facts.

(a) Any class that satisfies (i) also satisfies (ii): Δ^0_2 sets are Σ^0_2.

(b) Any class that satisfies (ii) also satisfies (iv): Any non-computable Δ^0_2 set D is a separating set for the computably inseparable Δ^0_2 sets D and $\bar{D}$.

(c) Any class that satisfies (ii) also satisfies (iii): C.e. sets are Δ^0_2.

The (partial) order on templates follows directly from points (a) through (c). Together with the supposition that $\mathcal{C}_{1'} \not\succeq \mathcal{C}_1$ and $\mathcal{C}_0 \not\succeq \mathcal{C}_1$, the order on templates leads directly to the order on subclasses. The supposition is necessary to prevent cases where (for instance) $\mathcal{C}_{1'}$ satisfies Templates (i) through (iv) and $\mathcal{C}_1$ satisfies exactly (ii) through (iv), in which case we would not have $\mathcal{C}_1 \succeq \mathcal{C}_{1'}$.

This is essentially a way of enforcing a naming convention. Any collection of subclasses can be put in a partial order using $\succeq$ according to which of Templates (i) through (iv) they satisfy. Requiring $\mathcal{C}_{1'} \not\succ \mathcal{C}_1$ and $\mathcal{C}_0 \not\succ \mathcal{C}_1$ is a way of saying that after these subclasses are put in this partial order, we don't give the weaker classes the names we would like to use for the stronger classes. Note that we do not require that $\mathcal{C}_{1'} \not\succeq \mathcal{C}_1$ and $\mathcal{C}_0 \not\succeq \mathcal{C}_1$; these classes could be the same strength.

□

Proposition 3.4.3 gives the order relations between templates and between subclasses that are known to always hold. We note that we cannot show that (iii) $\succeq$ (iv) using the same type of argument in the above proposition, since there are Δ^0_2 degrees (namely, 1-generics) that do not compute a separating set for any pair of computably inseparable c.e. degrees. We do know that (iv) $\succeq$ (iii) is not true as a general rule, since (v) holds of $\mathcal{C}_{\mathcal{E}0}$: Since only those degrees that compute a non-computable Δ^0_2 degree are not low for $\mathcal{C}_{\mathcal{E}0}$-isomorphism, there are degrees that compute separating sets for computably inseparable c.e. sets that are low for $\mathcal{C}_{\mathcal{E}0}$-isomorphism. Hence $\mathcal{C}_{\mathcal{E}0}$ does not satisfy (iii). This also shows that $\mathcal{C}_{\mathcal{E}0} \not\succeq \mathcal{C}_{\mathcal{E}1'}$.

However while we have not proven that it holds in general, our results so far are in agreement with the statement (iii) $\succeq$ (iv). In the case of scattered linear orders, our candidate for $\mathcal{C}_0$ is $\mathcal{C}_\omega$, and the only candidate we found for $\mathcal{C}_{1'}$ is in fact $\mathcal{C}_{Sc}$, the class of scattered linear orders (restricted down to the class of orders consisting of sequences of copies of $\mathbb{Z}$ and finite linear orders, if we prefer). But then we have $\mathcal{C}_{Sc} \succeq \mathcal{C}_{\omega}$ simply because $\mathcal{C}_{Sc} \supseteq \mathcal{C}_{\omega}$. We get the same result in the cases of shuffle sums, though less directly. While $\mathcal{C}_{\mathcal{S}1'}$ does not contain $\mathcal{C}_{\mathcal{S}0}$, it does contain shuffle sums where at least one suborder is a well order, which was all we needed to achieve result (iv). Thus $\mathcal{C}_{\mathcal{S}1'} \succeq \mathcal{C}_{\mathcal{S}0}$.

The case of equivalence structures is less straightforward. As mentioned, the diagonalization argument from Theorem 1.3.2 can be done using a c.e. set of sizes. We can modify that argument again by making our "tail" out of sizes disjoint from prime powers, and initializing and increasing sizes so that classes containing our witnesses are always the only classes of their size and each end up having size p_n for some n that is in either A or B for computably inseparable c.e. sets A and B. This the resulting structures likely will not work for the coding argument showing that any set separating A and B are not low for $\mathcal{C}_{\mathcal{E}1'}$-isomorphism, but they will be isomorphic to the structures that do, and so will be in the class $\mathcal{C}_{\mathcal{E}1'}$.

However, while we do not yet have a counterexample for either $\mathcal{C}_{1'} \succeq \mathcal{C}_0$ or (iii) $\succeq$ (iv), there are other structures (certain highly specialized subclasses of directed graphs) that we have not discussed where such a relation seems at least unlikely. Further even in these three cases it has not been shown that there are not weaker classes of structures satisfying (iii) but not (iv), so it would seem premature to conjecture that $\mathcal{C}_{1'} \succeq \mathcal{C}_0$ holds in general.

The next question is whether the relations we do have are strict. The complete categorization Template (v) holds for equivalence structures with $\mathcal{C}_0 = \mathcal{C}_{\mathcal{E}0}$ (Corollary 1.3.3), and for scattered orders with $\mathcal{C}_0 = \mathcal{C}_\omega$ (Corollary 1.1.4). This shows that for these structures, $\mathcal{C}_0 \not\succeq \mathcal{C}_{1'}$, since there are separating sets of c.e. sets that compute no Δ^0_2 degrees, and hence must be low for $\mathcal{C}_\omega$ and $\mathcal{C}_{\mathcal{E}0}$ isomorphism. It is not known whether $\mathcal{C}_{\mathcal{S}0}$ (or any other subclass of shuffle sums) satisfies the complete categorization Template (v). It is not known whether any of the other relations among these subclasses are strict, or, in the case of scattered linear order, whether there are even distinct candidates for $\mathcal{C}$, $\mathcal{C}_1$, $\mathcal{C}_{1'}$.

Our known results in this area as follows:

- Equivalence structures: $\mathcal{C}_\mathcal{E} \succeq \mathcal{C}_{\mathcal{E}1} \succeq \mathcal{C}_{\mathcal{E}1'} \succ \mathcal{C}_{\mathcal{E}0}$.
- Shuffle sums: $\mathcal{C}_\mathcal{S} \succeq \mathcal{C}_{\mathcal{S}1} \succeq \mathcal{C}_{\mathcal{S}1'} \succeq \mathcal{C}_{\mathcal{S}0}$.
- Scattered linear orders: $\mathcal{C}_{Sc} \succ \mathcal{C}_\omega$.

From the discussion of each of these structures in their respective sections, it seems reasonable to conjecture that $\mathcal{C}_\mathcal{E} \succ \mathcal{C}_{\mathcal{E}1}$ and $\mathcal{C}_\mathcal{S} \succ \mathcal{C}_{\mathcal{S}1}$. While these results have not been proven, the additional freedom in the more general subclasses seems to play a necessary role in their respective results. It may be possible to control the various sizes of portions of our structures in the various proofs in such a way as to prove results of type (ii) using isomorphic copies structures arrived at in the proofs of (iii), so it would be premature to conjecture $\mathcal{C}_{\mathcal{E}1} \succ \mathcal{C}_{\mathcal{E}1'}$ or $\mathcal{C}_{\mathcal{S}1} \succ \mathcal{C}_{\mathcal{S}1'}$.

It also seems reasonable to conjecture that $\mathcal{C}_{\mathcal{S}1'} \succ \mathcal{C}_{\mathcal{S}0}$, but more so because it appears that $\mathcal{C}_{\mathcal{S}0}$ lacks the freedom to handle separating sets than because we have evidence to suggest that $\mathcal{C}_{\mathcal{S}0}$ might satisfy Template (v).

And finally, in the case of scattered linear orders, we did not even find distinct candidates to fill the roles of $\mathcal{C}$, $\mathcal{C}_1$, $\mathcal{C}_{1'}$. This of course does not show that there are not such subclasses; however, the necessity for any scattered linear order to have suborders without end points to be able to allow multiple isomorphisms, together with the simplicity of the orders constructed in Theorem 2.4.1, suggests that if there are distinct classes of scattered linear orders $\mathcal{C}, \mathcal{C}_1, \mathcal{C}_{1'}$ satisfying their respective templates, then it would at least not be surprising if they satisfy $\mathcal{C} \preceq \mathcal{C}_1 \preceq \mathcal{C}_{1'}$.

Bibliography

[1] Denis R. Hirschfeldt, Bakhadyr Khoussainov, Richard A. Shore and Arkadii M. Slinko, “Degree spectra and computable dimensions in algebraic structures,” *Annals of Pure and Applied Logic* 115 (2002), 71-113

[2] Asher M. Kach, “Computable shuffle sums of ordinals,” *Archive for Mathematical Logic*, 47:3 (2008), 211-219

[3] Johanna N.Y. Franklin and Reed Solomon, “Degrees that are low for isomorphism,” *Computability* (2014)

[4] B. A. Anderson and B.F. Csima, ”Degrees that are not degrees of categoricity,” *Notre Dame Journal of Formal Logic*