Applications of computability theory to partial and linear orders

Marie Nicholson, Ph.D.

University of Connecticut, 2017

ABSTRACT

One of the notions central to the study of computable linear orders is that of computable categoricity. Remmel showed that a computable linear ordering L is computably categorical if and only if the (classical) order type of L has only a finite number of successor pairs. In this proof, Remmel assumes that L has infinitely many successor pairs and then constructs another computable linear order R, which is not computably isomorphic to L, and a Δ^0_2-isomorphism f such that $f : L \to R$ is an isomorphism. This shows that L is not computably categorical. The first aim of this dissertation is to investigate under what conditions we can construct f below certain types of Δ^0_2 degrees.

The isomorphism relation on countable posets preserves all structural information but there are $2^{\aleph_0}$ many isomorphism classes of countable posets and the isomorphism relation is Σ^1_1-complete. The second aim of this dissertation is to examine two coarser classifications of partially ordered sets, namely Tukey equivalence and cofinal equivalence. Although these relations preserve less structural information than the isomorphism relation, they still describe some of the essential components of partially ordered sets. However, both Tukey and cofinal equivalence result in just countably

many equivalence classes of countable posets. We examine the complexity of Tukey types and cofinal types of countable posets. In addition, there are various classically equivalent definitions for the notion of Tukey reducibility and we examine how difficult it is to prove their equivalence.

Applications of computability theory to partial and linear orders

Marie Nicholson

M.Sc. Mathematics, University of Connecticut, USA, 2014

B.Sc. Computer Science and Mathematics, University College Cork, Ireland, 2005

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2017

APPROVAL PAGE

Doctor of Philosophy Dissertation

Applications of computability theory to partial and linear orders

Presented by

Marie Nicholson, B.S. Math., M.S. Math.

Major Advisor _____
David Reed Solomon

Associate Advisor _____
Damir D. Dzhafarov

Associate Advisor _____
Fabiana Cardetti

University of Connecticut

2017

ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisor, Reed Solomon, for his continual support and guidance over the past several years. His enthusiasm for the subject was infectious and our meetings were often the highlight of my week—I would always leave his office both enlightened and enthusiastic. I am very grateful for his motivation, patience, feedback and encouragement at every stage in preparing this thesis. Without his input, this thesis would contain far fewer lemmas. I could not have asked for a better advisor or mentor.

I would also like to thank Damir D. Dzhafarov and Fabiana Cardetti for serving on my thesis committee and for their insightful comments.

A very special thank you to Steven Binns, who introduced me to areas of logic that I didn't know existed, turning my flicker of interest into a passion. I would also like to thank Joe McKenna, who encouraged me to apply for grad school at UConn at a time when I was about to give up on the possibility of graduate studies and who was also a wonderful support during my time at UConn.

I would like to thank all the faculty, staff and graduate students of the mathematical department for making UConn a wonderful place where I could grow and learn. I am most grateful for the interesting and stimulating graduate courses in logic and other areas. In particular I would like to thank Johannah Franklin for many a helpful conversation and for being not only a female logician but the first female to teach me at third level. Particular thanks is also due to Monique Roy, for helping me through

the tangle of paperwork involved at every stage. I would not be graduating without her help.

A massive thank you to all my friends who at many times seemed to feel more strongly about my completion of a Ph.D. than I did myself. I would also like to thank my parents, who have always being my biggest fans and encouraged my ability and interest in mathematics from a very young age. Thank you all for believing in me.

To my friends at Coleones, thank you for providing me with a home from home: without you this thesis may have been finished years earlier but it would have been much less fun. To Tarski my cat, who always seemed to sit on whatever piece of paper I was currently using, thank you for reminding me to stand up once in a while, even if it was just to feed you.

To my dearest Bob, a massive thank you for coming on this journey with me: everything is better with you. Thank you sharing in all my highs and lows, for cooking me lots of delicious meals and for proofreading many drafts of this thesis. Without you this thesis would contain fewer commas, more commas or the exact same number of commas distributed differently: at this point it is impossible to tell.

This dissertation is dedicated to the memory of my father, Séan Nicholson, who cut an apple into pieces to teach me about fractions when I was barely old enough to count.

Contents

Chapter 1

Introduction

1.1 Computable Algebra

Computable algebra is the analysis of classical mathematical structures such as rings, graphs or linear orders using the tools of computability theory. We say a structure is computable if its domain is computable and the functions and relations of that structure are also computable. Given a computable structure we then ask whether one can derive algorithms for computing various properties of the structure. If the given property is not computable, then we ask to what extent the computability fails. We also ask questions about the computable content of theorems. Mathematical theorems often assert the existence of an object. We ask how complicated it is to construct these objects. For example, although every vector space has a basis, there is no computable way to find a basis in all cases.

Using the notion of a computable function, it is possible to precisely define what it means for an algebraic structure to be presented computably. We review some of

the basics concepts of computability here. For a more comprehensive introduction, see [**?**].

1.2 Basic Concepts

We let $\{\varphi_e \mid e \in \mathbb{N}\}$ denote all *partial computable functions* from $\mathbb{N}$ to $\mathbb{N}$. Although these functions have no time and memory constraints, they may not necessarily halt. We write $\varphi_e(x) \downarrow$ if φ_e converges on input x and $\varphi_e(x) \uparrow$ if φ_e does not converge on input x. If $\varphi_e(x)$ converges it does so after a finite number of steps. We write $\varphi_{e,s}(x) = y$ if φ_e converges to y on input x in less than s steps and we write $\varphi_e(x) = y$ if there exists s such that $\varphi_{e,s}(x) = y$. Although we consider only subsets of $\mathbb{N}$, with the use of effective coding this includes much more. For example, both the rational numbers and the set of all finite binary strings can be coded in an effective manner into $\mathbb{N}$.

A set $X \subseteq \mathbb{N}$ is called *computable* if its characteristic function is computable. That is, there is an effective procedure for deciding if $n \in X$ or $n \notin X$. A set $X \subseteq \mathbb{N}$ is called *computably enumerable* if X is equal to the domain of φ_e for some e. That is, there is an effective procedure for enumerating all the members of X. There are computably enumerable sets which are not computable. The canonical example is the *halting set* $K = \{e \mid \varphi_e(e) \downarrow\}$.

We say that A is *Turing computable* from B, written as $A \leq_T B$, if there is a Turing functional Φ with B as an oracle which computes A. That is, we can compute A given that B is in memory and we are allowed to ask a finite number of questions of B during the computation that decides whether a particular n is in A. We write

$A = \Phi^B$. We let $\{\Phi_e \mid e \in \mathbb{N}\}$ list the Turing functionals and identify φ_e with $\Phi_e^{\emptyset}$. We write $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$. The equivalence classes under $\equiv_T$ are called *degrees*. The degree of a set is a measure of the computational complexity of the set. The least degree $\mathbf{0}$ denotes the degree of the computable sets. The *use* of a converging functional computation $\Phi_e^A(n)$ is $x+1$ if x is the largest number such that the value $A(x)$ is queried during the computation. We denote this use as use$(\Phi_e^A(n))$.

We relativise the halting set to a set A by defining the *jump* operator on A as $A' = \{e \mid \Phi_e^A(e) \downarrow\}$. We can think of this as the set of programs that will halt on their own names with oracle A. If the degree of A is $\mathbf{a}$, we write the degree of A' as $\mathbf{a}'$. In particular as $\mathbf{0}$ is the degree of $\emptyset$, the degree of the halting set is $\mathbf{0}'$. It is a classical result that the unsolvability of the halting problem generalises to $A <_T A'$. We write $A^{(n)}$ for the n-th iterate of a jump operator applied to A and we have the following hierarchy of degrees $\mathbf{0} < \mathbf{0}' < \mathbf{0}'' < \ldots < \mathbf{0^{(n)}} \ldots$. We say that a degree $\mathbf{a}$ is *arithmetical* if $\mathbf{a} \leq \mathbf{0^{(n)}}$, for some $n \in \mathbb{N}$. We also say a set A with degree $\mathbf{a}$ is *low* if $\mathbf{a}' = \mathbf{0}'$. That is, the jump of A has the least possible degree. There are non-computable low degrees.

The arithmetical degrees are closely related to arithmetical formulas. We say a set A is Σ_1^0 if there exists a computable relation R such that $x \in A$ if and only if $\exists y R(x, y)$. We say a set B is Π_1^0 if there is a computable relation R such that $x \in B$ if and only if $\forall y Q(x, y)$. More generally, we say a set A is Σ_n^0 if there exists a computable relation R such that $x \in A$ if and only if $\exists y_1 \forall y_2 \exists y_3 \forall y_4 \ldots Q y_n R(x, y_1, y_2, \ldots y_n)$ where Q is $\forall$ or $\exists$ depending on whether n is even or odd. Similarly we say that B is Π_n^0 if there exist a computable relation R such that $x \in B$ if and only if $\forall y_1 \exists y_2 \forall y_3 \exists y_4 \ldots Q y_n R(x, y_1, y_2, \ldots y_n)$, where Q is $\forall$ or $\exists$ depending on whether n is even or odd. Finally, we say that A is Δ_n^0 if and only if A is Σ_n^0 and Π_n^0. Note that

the Δ^0_1 sets are the computable sets. It can be shown that if A is Δ^0_2 then $\mathbf{a} \leq \mathbf{0'}$ and more generally a set A is Δ^0_{n+1} if and only if $\mathbf{a} \leq \mathbf{0^{(n)}}$. Also, a set A is computably enumerable if and only if A is Σ^0_1.

The *Schoenfield Limit Lemma* says that A is Δ^0_2 if and only if there exists a computable sequence of sets $\{A_s\}_{s\geq 0}$ such that the $\lim_s A_s(x)$ exists and equals $A(x)$. We say that A is n-c.e. if for each $x \in \mathbb{N}$ the value $A_s(x)$ changes at most n times. A set is A is computably enumerable if and only if it is 1-c.e. and we say that A is d-c.e. if A is 2-c.e., the difference of two *c.e.* sets. A set A is ω-c.e. if there is a computable function f such that for all x, $|\{s : A_s(x) \neq A_{s+1}(x)\}| \leq f(x)$.

1.3 Other Notation

We write ω for the usual order type of $\mathbb{N}$, and η for the order type of the rational numbers $\mathbb{Q}$.

Definition 1.3.1. A *binary tree* is a set T of finite binary strings such that if τ belongs to T and σ is an initial segment of τ then σ also belongs to T. We shall use $2^{<\mathbb{N}}$ to denote the space of finite binary strings.

We say that a binary tree is infinite if it contains strings or arbitrary large length.

1.4 Computable Linear Orders

We will only be concerned with countable linear orders since all computable algebraic structures are countable. We typically identify the domain of countable linear or

partial orders with $\mathbb{N}$, and hence we need only worry about the complexity of the ordering relation.

Definition 1.4.1. We say a linear order $L = (\mathbb{N}, \leq_L)$ is *computable* if the relation $\leq_L$ is computable.

Effective properties of countable linear orderings and their relations have been studied extensively since the 1960's. See Downey's survey article [**?**] for a detailed account of this research. Among the notions central to this theory is that of computable categoricity of a linear order.

Definition 1.4.2. A computable linear order L is *computably categorical* if for any other computable linear order R with $L \cong R$, there exists a total computable function f with $f : L \cong R$.

All finite linear orders are computably categorical, as are some infinite linear orders. The successor relation of a linear order is important in this context.

Definition 1.4.3. We say that (x, y) is a *successor pair* in L if $x <_L y$ and there does not exist a $z \in L$ such that $x <_L z <_L y$.

Any countable linear ordering which contains no successors is computably categorical. That is, η is computably categorical. Given two computable copies, one can perform an effective back-and-forth construction to create an isomorphism. However it is not the case that all infinite linear orders are computable categorical. For example, the natural presentation of $\omega = \{0 < 1 < \ldots\}$ has the property that the successor function is computable. However there is a computable linear order of order type ω whose successor function is not computable. Hence these computable linear orders

are isomorphic but not computably isomorphic. Remmel gave a complete characterisation of computable categorical linear orders.

Theorem 1.4.1. *(Remmel [?]) A computable linear ordering L is computably categorical if and only if the (classical) order type of L has only a finite number of successor pairs.*

A linear order with only finitely many successor pairs is computably categorical because one could begin by correctly mapping the finitely many successor pairs, then run the back-and-forth construction in each of the finitely many infinite intervals with no successors. To show the converse, Remmel assumes that L has infinitely many successor pairs and then constructs another computable linear order R, which is not computably isomorphic to L, and a Δ^0_2-isomorphism f such that $f : L \to R$ is an isomorphism. This shows that L is not computably categorical.

It is natural to ask under what conditions we can construct f below certain types of Δ^0_2 degrees. This is the topic of Chapter 2. We show that for a large class of computable linear orderings, if the successor relation is computable then it is possible to construct f below any non-computable computably enumerable degree. For example, given a computable linear order L with unbounded successors and computable successor relation and an arbitrary non-computable c.e. set D, we construct another a computable linear order $(R, <_R)$ such that R is not computably isomorphic to L but R is D-computably isomorphic to L. Hence R is not computably isomorphic to L and the function witnessing the isomorphism has degree less than that of D.

We also show that we cannot perform Remmel's argument below every non-zero Δ^0_2 degree. To do this, we will construct a computable linear order L and a noncomputable ω-c.e. set D such that for any computable linear order R, if R is classically

isomorphic to L but not computably isomorphic to L, then R is not D-computably isomorphic to L.

Moreover, we show that given any linear order L with infinitely many successor pairs it is possible to construct a computable linear order R which is not computably isomorphic to L but for which there is an isomorphism $f : R \to L$ of low degree.

1.5 Reverse Mathematics

Reverse Mathematics is an area of logic which seeks to identify which set-theoretic axioms are necessary to prove a theorem. The usual axioms of set theory are too expressive for our purpose so we work with subsystems of second order arithmetic Z_2. In this way we are restricting ourselves to countable mathematics. This means that theorems of algebra and combinatorics are restricted to countable structures, while theorems of analysis and topology are restricted to separable spaces. Despite this restriction many classical theorems of mathematics are equivalent to a subsystems of second order arithmetic.

The language of second order arithmetic consists of two types of variables. The first order variables $x, y, z, \ldots$ are intended to range over $\mathbb{N}$ and second order variables $X, Y, Z, \ldots$ are intended to range over subsets of $\mathbb{N}$. In addition we have the usual symbols of first order arithmetic $+$, $\cdot$, and $<$; set membership $\in$; and the constants 0, and 1.

The theory of *second order arithmetic* consists of the following:

- PA^-, discrete ordered semiring axioms for the natural numbers.

- Full comprehension scheme:

$$\exists X \forall x (x \in X \leftrightarrow \varphi(x))$$

  for each formula φ in the language of second order arithmetic, where X does not occur.

- Full induction scheme:

$$(\varphi(0) \wedge \forall n(\varphi(n) \rightarrow \varphi(n+1))) \rightarrow \forall n \varphi(n)$$

  for each formula φ in the language of second order arithmetic.

Comprehension axioms are also known as set existence axioms. This scheme says that if we have a formula in second order arithmetic then the set of numbers which satisfy that formula exists. By restricting the comprehension and induction schemes, we obtain subsystems of Z_2. There are five subsystems of particular interest which are know as the "big five" subsystems of reverse mathematics. In order of increasing strength these are $\mathsf{RCA}_0, \mathsf{WKL}_0, \mathsf{ACA}_0, \mathsf{ATR}_0$ and $\Pi^1_1 - \mathsf{CA}_0$. It turns out that a large number of theorems, in many different areas of mathematics, have been proved to be equivalent to one of these five systems. For a detailed account of this see Simpson's book [**?**]. We will only be concerned with the first three subsystems, $\mathsf{RCA}_0, \mathsf{WKL}_0$ and ACA_0.

The base system for reverse mathematics is called RCA_0 and roughly corresponds to computable mathematics.

Definition 1.5.1. The system RCA_0 consists of PA^- together with Δ^0_1- comprehension and Σ^0_1- induction.

A reasonable amount of mathematics can be developed in RCA_0. Theorems provable in RCA_0 include the intermediate value theorem, Urysohn's Lemma for complete separable metric spaces, the existence of algebraic closures for countable fields and every infinite tree with no dead ends has a path.

The aim of reverse mathematics is to determine a set of axioms which are not only sufficient but also essential to prove a theorems of ordinary mathematics. Working over the base theory RCA_0, we show the equivalence of a theorem T and a set of axioms in second order arithmetic A in the following two steps:

1. We show that the theorem T is provable from RCA_0 together with A.

2. We show that the set of axioms A is provable from RCA_0 together with T.

The first part of the proof shows that the set of axioms A is sufficient to prove a theorem T and the second part of the proof shows that the set of axioms is in fact essential in proving T. We call second part of the proof the "reversal". When both parts are proved we say that T and A are equivalent over RCA_0. In this dissertation, we will show equivalences to WKL_0 and ACA_0.

Definition 1.5.2. WKL_0 consists of RCA_0 together with Weak Königs Lemma, which states that every infinite binary tree has an infinite path.

We know that WKL_0 is a proper extension of RCA_0 as there exists a computable tree T such that no infinite path through T is computable. However WKL_0 is a relatively week extension of RCA_0 since there is a model of WKL_0 in which every set is low.

There are many mathematical constructions that can be thought of as finding infinite paths on infinite binary trees. For example, finding a prime ideal of a given

commutative ring. It can be shown that the use of WKL_0 is not just convenient but in fact essential to this proof, meaning that although these existence theorems deal with different kinds of mathematical objects, they can be thought of as having the same fundamental combinatorial core. At the core of WKL_0 is a compactness argument. In fact, Weak Königs Lemma is equivalent (over RCA_0) to the compactness of 2^ω. Other theorems shown to be equivalent to WKL_0 include the compactness of $[0,1]$ and the fact that any continuous real-valued function on $[0,1]$ has a supremum. We will use the following separation result when proving equivalences over RCA_0.

Lemma 1.5.1. *The following are equivalent over* RCA_0*:*

- WKL_0.
- *For all* $f, g : \mathbb{N} \to \mathbb{N}$*, if* $\forall x \, \forall y (f(x) \neq g(y))$*, then there is a set* X *such that* $\forall x$ $(f(x) \in X$ *and* $g(x) \notin X)$.

Definition 1.5.3. The Arithmetric Comprehension Axiom ACA_0 consists of PA^- together with Σ^0_1- comprehension and Σ^0_1- induction.

Given Σ^0_1- comprehension ACA_0 is stronger than RCA_0. In fact, from Σ^0_1- comprehension we attain comprehension for all arithmetical formulas (formulas with no set quantifiers). ACA_0 is also stronger than WKL_0. This is true since ACA_0 proves that the halting set K exists, while there is a model of WKL_0 which contains only low sets and hence doesn't contain the halting set.

Among other theorems equivalent to ACA_0 over RCA_0 are the Bolzano-Weierstraß Theorem, the existence of maximal ideals for countable commutative rings, the existence of bases for countable vector spaces, and König's Lemma, which states that every infinite finitely branching tree has an infinite path.

We use the following result when proving equivalences to ACA_0 over RCA_0.

Lemma 1.5.2. *The following are equivalent over* RCA_0*:*

- ACA_0.
- *For every one-to-one function* $f : \mathbb{N} \to \mathbb{N}$*, the range of* f *exists.*

1.6 Tukey Types of Partial Orders

Tukey introduced the Tukey ordering to develop the notion of Moore-Smith convergence in topology [**?**]. After its initial success in helping develop general topology, Tukey reducibility was studied as a means of classifying algebraic structures related to partially ordered sets in, for example, Day [**?**], Isbell [**?**] and Todorcevic [**?**]. Its classification is coarser than that of isomorphism and so can be useful in settings where isomorphism is too fine to give a useful classification.

Day studied the Tukey ordering in the more general setting of oriented systems. This is the topic of Section 3.1. A system $(S, <)$ is oriented if it is non-empty, transitive, and for each $s \in S$ there is $s' \in S$ such that $s < s'$. Although we follow Day in using $<$ as notation for a binary relation in an oriented system, $<$ is not assumed to be linear or irreflexive. We say that an oriented system $(S, <)$ is directed if each pair of elements in S have a common upper bound in S. We also say a subset X of S is cofinal in $(S, <)$ if for each $s \in S$ there exists $x \in X$ such that $s < x$.

Let $(S, <_S)$ and $(T, <_T)$ be oriented systems. T is *Tukey reducible* to S, which we will write as $T <_{ty} S$, if there exist functions $f : S \to T$ and $g : T \to S$ such that for each $t \in T$, $g(t) <_S s$ implies $t <_T f(s)$. If two functions are related in this way, we refer to f as an *upper support* for g and g as a *lower support* for f. If f is a function

from S into T, then $<_f$ is a binary relation on $S+T$, the disjoint union of S and T, such that $s <_f t$ if and only if $f(s) <_T t$. If $<_1$ and $<_2$ are two binary relations on S, then $(S, <_1)$ *includes* $(S, <_2)$ if $s <_2 s'$ implies $s <_1 s'$.

Day proved the following theorem in [**?**] about the existence of upper supports of functions.

Theorem 1.6.1. *If $(S, <_S)$ and $(T, <_T)$ are disjoint oriented systems, a function $g : T \to S$ has an upper support if and only if there exists an orientation $<$ of $S+T$ such that*

1. *$<$ includes $<_S, <_T, <_g$,*
2. *T is cofinal in $(S+T, <)$, and*
3. *$<$ agrees with $<_T$ in T.*

Given an orientation $<$ on $S+T$ which satisfies properties 1-3 we can effectively construct a function $f : S \to T$ which is an upper support of g. We will prove that the converse of this theorem is equivalent to Weak Königs Lemma. That is, the following are equivalent over RCA_0:

1. WKL_0.
2. If $(S, <_S)$ and $(T, <_T)$ are disjoint oriented systems and there exists a function $g : T \to S$ which has an upper support, then there exists an orientation $<$ of $S+T$ such that
   - 2.1. $<$ includes $<_S, <_T, <_g$,
   - 2.2. T is cofinal in $(S+T, <)$,

2.3. $<$ agrees with $<_T$ in T.

There is also a notion of the least orientation including multiple orientations. If $<_1, <_2, \ldots, <_n$ are orientations on S then there is a unique minimal orientation $<$ of S which includes all $<_i$; explicitly, $s_0 < s$ if and only if there exists $s_1, s_2, \ldots, s_k = s$ and indices $i_0, i_1, \ldots, i_{k-1}$ such that for each $0 \leq j < k$ the relation $s_j <_{i_j} s_{j+1}$ holds. We will also show the following: The following are equivalent over RCA_0:

1. ACA_0.

2. If $(S, <_S)$ and $(T, <_T)$ are disjoint oriented systems, and $g : T \to S$ has an upper support $f : S \to T$, then there exists a least orientation on $S + T$ which includes $<_S, <_T, <_g, <_f$.

In Section 3.2 we turn our attention to Tukey reducibility and partial orders. If $(P, \leq_P)$ is a partially ordered set then it satisfies the axioms for an oriented system. In the context of partial orders one normally defines Tukey reducibility using the classically equivalent notion of a convergent map. We say a partial ordering $(E, \leq_E)$ is *Tukey reducible* to a partial ordering $(D, \leq_D)$, denoted by $E \leq_{ty} D$, if and only if there is a convergent map from D into E. That is, if and only if there is a function $f : D \to E$ such that for all $e \in E$, there is a $d \in D$ such that $f(c) \geq_E e$ for all $c \geq_D d$. Note that the convergent map in this definition is equivalent to the upper support in the definition of Tukey reducibility above. That is, if you define $g : E \to D$ such that $g(e)$ is a witness d for e in the definition of f being convergent, then you have $\forall e \in E \forall c \in D[g(e) \leq_D c$ implies $e \leq_E f(c)]$ so f is the upper support of g.

We look at various classically equivalent definitions for the notion of Tukey reducibility and examine how difficult it is to prove their equivalence. In what follows

let $(D, \leq_D)$ and $(E, \leq_E)$ be partial orderings. We say a function $f : D \to E$ is *cofinal* if the image of each cofinal subset of D is cofinal in E. Tukey showed in [**?**] that the existence of a cofinal map from $D \to E$ is equivalent to the existence of a convergent map from $D \to E$. We will show that the following are equivalent and their equivalence is provable in RCA_0.

1. There exists a convergent map from D into E.

2. There exists a cofinal map from D into E.

A subset $X \subseteq D$ is called *unbounded* if there is no single $d \in D$ which simultaneously bounds every member of X. That is, for each $d \in D$, there is some $x \in X$ such that $d \not\geq_D x$. A map $g : E \to D$ is called a *Tukey map* or an *unbounded map* if the g-image of each unbounded subset of E is an unbounded subset of D. Schmidt showed in [**?**] that there exists a convergent map from D into E if and only if there is an unbounded map from E into D. We will show that this equivalence requires ACA_0 by showing that there exists a computable convergent map $f : D \to E$ such that if $g : E \to D$ is an unbounded map, then $0' \leq_T g$.

If $D \leq_{ty} E$ and $E \leq_{ty} D$, we say that D is *Tukey equivalent* to E, written as $D \equiv_{ty} E$. The relation $\equiv_{ty}$ is an equivalence relation and the equivalence classes are called Tukey types.

In Section 3.3 we give a characterisation of the Tukey types of computable posets and show that determining this Tukey type of computable posets is $\mathbf{0}'''$-complete.

In Section 3.4 we examine a closely related but finer equivalence relation partial orderings, known as cofinal similarity. Two partial orderings are *cofinally similar* if and only if there is a partially ordered set into which they both embed as cofinal subsets. The equivalence classes are called cofinal types and if P is cofinally similar to

Q, then we write $P \equiv_{cf} Q$. Day showed in [**?**] that for directed sets cofinal similarity coincides with Tukey equivalence. We give a characterisation of the cofinal types of computable posets and show that this has complexity of at most $\Sigma^0_4 \wedge \Pi^0_4$.

Chapter 2

Computable Categoricity of Linear Orders

Definition 2.0.1. We say a linear order $L = (\mathbb{N}, \leq_L)$ is *computable* if the relation $\leq_L$ is computable.

One of the notions central to the study of computable linear orders is that of computably categoricity.

Definition 2.0.2. Given a Turing degree $\mathbf{d}$ and computable linear orders L and R, we say that L is $\mathbf{d}$-*computably isomorphic* to R if there is an isomorphism between L and R which is computable from $\mathbf{d}$. If $\mathbf{d} = \mathbf{0}$ we say that L is *computably isomorphic* to R.

Definition 2.0.3. We call a computable linear ordering L *computably categorical* if any computable linear ordering R isomorphic to L is computably isomorphic to L.

All finite linear orders are computably categorical, as are some infinite linear orders. The successor relation of a linear order is important in this context.

Definition 2.0.4. We say that (x, y) is a *successor pair* in L, written as $x \to_L y$, if $x <_L y$ and there does not exist a $z \in L$ such that $x <_L z <_L y$.

In addition, we define a block of successors as follows.

Definition 2.0.5. We say that (x, y) are in a *block* if there are only finitely many elements between x and y. That is, there exist $z_0, z_1, \ldots, z_n$ such that $z_0 = x$, $z_n = y$ and for each $i < n$ the pair (z_i, z_{i+1}) is a successor pair.

The successor relation of a computable linear order is *co-c.e.*, that is the complement of a c.e. set. Hence the successor relation of a computable linear order is Π^0_1 and the block relation is Σ^0_2. However if the successor relation is computable, the block relation is Σ^0_1.

Remmel gave a complete characterisation of computably categorical linear orders. In [**?**] Remmel showed that a computable linear ordering L is computably categorical if and only if the (classical) order type of L has only a finite number of successor pairs. A linear order with only finitely many successor pairs is computably categorical because one could begin by correctly mapping the finitely many successor pairs, then run the back-and-forth construction in each of the finitely many infinite intervals with no successors. To show the converse, Remmel assumes that L has infinitely many successor pairs and then constructs another computable linear order R, which is not computably isomorphic to L, and a Δ^0_2-isomorphism f such that $f : L \to R$ is an isomorphism. This shows that L is not computably categorical. It is natural to ask under what conditions we can construct f below certain types of Δ^0_2 degrees. Firstly we show that for a large class of computable linear orderings, if the successor relation is computable, then it is possible to construct f below any computably enumerable degree.

2.1 Remmel's construction under a c.e. set

Theorem 2.1.1. *Let $(L, <_L)$ be a computable linear order with infinitely many successor pairs and let D be a non-computable Σ^0_1 set. Suppose the successor relation is computable and the successor pairs satisfy one of the following requirements:*

- *There exists $b \in L \cup \{\infty\}$ such that for all $a <_L b$ there exists a successor pair (x, y) in L such that $a <_L x$ and $y <_L b$.*
- *There exists $b \in L \cup \{-\infty\}$ such that for all $a >_L b$ there exists a successor pair (x, y) in L such that $b <_L x$ and $y <_L a$.*
- *There exists an n such that each block of successors has length less than n.*

Then there is a computable linear order $(R, <_R)$ such that R is not computably isomorphic to L but R is D-computably isomorphic to L.

Proof. This proof will closely follow the notation and argument of Remmel's proof with the addition of a permitting requirement to make $f \leq_T D$.

Without loss of generality we assume that the domain of L is $\mathbb{N}$ and let C be the set of infinitely many successor pairs in L. Let $c_0 < c_1 < c_2 \ldots$ be a list of C in order of magnitude.

We construct a linear order R and a function $f : L \to R$ meeting, for all $e \in \omega$, the following requirements:

- N : R is a computable linear order,
- M : $f : L \to R$ is an isomorphism,
- $\widehat{M}$: $f \leq_T D$, and
- P_e : φ_e is not an isomorphism from L onto R.

Strategy for N:

We will construct a computable linear order R in stages with domain $\mathbb{N}$ such that $R = \cup_s R_s$. At each stage s of our construction, we shall enumerate at least one number into R and specify a linear order $<_R$ on $\{0, 1, \ldots, k_s\}$, where k_s is the largest number enumerated into R at stage s. Once x and y are enumerated into $<_R$ and either $x <_R y$ or $y <_R x$ is specified at some stage s, the order relation on x and y cannot change after this stage, hence ensuring that $R = (\mathbb{N}, <_R)$ is computable.

For notational convenience, we let $r_0^s, r_1^s, \ldots, r_{k_s}^s$ be the elements $\{0, 1 \ldots, k_s\}$ in increasing $<_R$ order. Similarly, $l_0^s, l_1^s, \ldots, l_{k_s}^s$ are the elements $\{0, 1, \ldots, k_s\}$ in increasing $<_L$ order. Therefore, $R_s = (\{0, 1, \ldots, k_s\}, <_R)$ and $L_s = (\{0, 1 \ldots, k_s\}, <_L)$.

Strategy for M:

At each stage we define a finite isomorphism $f_s : \{0, 1, \ldots, k_s\} \to \{0, 1, \ldots, k_s\}$ to be the unique order isomorphism between $(\{0, 1, \ldots, k_s\}, <_L)$ and $(\{0, 1, \ldots, k_s\}, <_R$). Thus for all $i \leq k_s$, we have $f_s(l_i^s) = r_i^s$. We define $f(x) = \lim_s f_s(x)$. The strategy to satisfy P_e may require $f_s(x) \neq f_{s+1}(x)$ for some x and s. Hence we must ensure that $\lim_s f_s(x)$ and $\lim_s f_s^{-1}(x)$ exist for all x. To achieve this our construction will allow the P_e strategy to require $f_s(x) \neq f_{s+1}(x)$ only if $x \notin \{0, 1, \ldots, e\} \cup \{f_s^{-1}(0), f_s^{-1}(1), \ldots, f_s^{-1}(e)\}$. As we have to act to satisfy P_e at most once, the limits will exist.

Strategy for $\widehat{M}$:

Without loss of generality we assume that exactly one number enters D at each stage and let d_s denote this number at stage s. To ensure f is D-computable we allow $f_{s+1}(x) \neq f_s(x)$ only if $x \geq d_s$.

The Strategy for P_e:

The requirement P_e is already permanently satisfied at stage s if either $\varphi_{e,s}$ is not one-to-one and order preserving on $\{x \mid x \leq k_{s-1} \text{ and } \varphi_{e,s}(x) \downarrow\}$ or if there exists $a, b \in L_{s-1}$ such that $a \rightarrow_L b$ but $\varphi_e(a) \not\rightarrow_{R_{s-1}} \varphi_e(b)$. If P_e does not meet these conditions at stage s, then we look for $a, b \in L_{s-1}$ such that:

- $a \rightarrow_L b$,
- $\varphi_{e,s}(a) \downarrow$, $\varphi_{e,s}(b) \downarrow$, and
- $\varphi_{e,s}(a), \varphi_{e,s}(b)$ are successors in R_{s-1}.

If such a pair a, b exists then we define $<_R$ such that $\varphi_e(a) <_R t <_R \varphi_e(b)$ where t is the least number not in R_{s-1}. Hence we will have $a, b \in L$ such that $a \rightarrow_L b$ but $\varphi_e(a) \not\rightarrow_{R_s} \varphi_e(b)$. Hence P_e is satisfied at the end of this stage. However, we also need to modify the current isomorphism f_s so that $f_{s+1}^{-1}(a) \not\rightarrow_L f_{s+1}^{-1}(b)$. We discuss this below in the strategy interaction section.

Blocks:

We define the following block relations:

- $B(l_j, l_k)$ holds if and only if there exist finitely many elements $x_0, x_1, \ldots, x_n \in L$ such that $x_i \rightarrow_L x_{i+1}$ for all $i < n$ and $l_j = x_0$ and $l_k = x_n$.

- $B_s(l_j, l_k)$ holds if and only if there exist finitely many elements $x_0, x_1, \ldots, x_n \in L_s$ such that $x_i \rightarrow_L x_{i+1}$ for all $i < n$ and $l_j = x_0$ and $l_k = x_n$.

Note that $B(l_j, l_k)$ is Σ^0_1 and $B_s(l_j, l_k)$ is computable as the successor relation is computable. We refer to the maximal block of successors containing l_k at stage s as $B_s(l_k)$. That is, $B_s(l_k) = \{l \mid B_s(l, l_k) \text{ or } B_s(l_k, l)\}$. We will refer to the block of successors to the left of l_k at stage s as $LB_s(l_k)$. That is, $LB_s(l_k) = \{l \mid B_s(l, l_k)\}$. Similarly we refer to the block of successors to the right of l_k as $RB_s(l_k)$. That is, $RB_s(l_k) = \{l \mid B_s(l_k, l)\}$.

Strategy Interactions

Recall that the strategy to satisfy M requires that at each stage s we define a finite isomorphism $f_s : \{0, 1, \ldots, k_s\} \rightarrow \{0, 1, \ldots, k_s\}$ to be the unique order isomorphism between L_s and R_s. Hence, if the strategy to satisfy P_e takes action at some stage $s + 1$ by enumerating t into R such that $\varphi_e(a) <_R t <_R \varphi_e(b)$, where $a \rightarrow_L b$, then $f_{s+1}(x) \neq f_s(x)$ on some interval of L. Suppose $\varphi_e(a) = r^s_j$ and $\varphi_e(b) = r^s_{j+1}$ at stage s. Then at stage s we have $f_s(l^s_j) = r^s_j = \varphi_e(a)$ and $f_s(l^s_{j+1}) = r^s_{j+1} = \varphi_e(b)$. If at stage $s + 1$ the strategy P_e enumerates t into R such that $\varphi_e(a) <_R t <_R \varphi_e(b)$, then we have $f_s(l^s_j) <_R t <_R f_s(l^s_{j+1})$. So to satisfy M we must have $f_{s+1}(x) \neq f_s(x)$ for $x \in RB_s(l^s_{j+1})$ or $x \in LB_s(l^s_j)$. More specifically, f_{s+1} must shift the image of each element in $RB_s(l^s_{j+1})$ one element to the left or f_{s+1} must shift the image of each element in $LB_s(l^s_j)$ one element to the right.

To respect requirements M and $\widehat{M}$ the strategy can only do this if the minimum of the $RB_s(l^s_{j+1})$ or the minimum of the $LB_s(l^s_j)$ is greater than e, d_s and all elements of $\{f_s^{-1}(0), f_s^{-1}(1), \ldots, f_s^{-1}(e)\}$. Suppose D permits P_e to makes these changes at some stage $s+1$ and we have $f_{s+1}(x) \neq f_s(x)$ for all $x \in RB_s(l^s_{j+1})$. Since we have shifted the image of each element in $RB_s(l^s_{j+1})$ to the left, there is an element of R which is not currently in the image of f_{s+1}. Specifically, this is the element r^s_v which was the image of the rightmost element l^s_v of the block $RB_s(l^s_{j+1})$ at stage s. Since l^s_v and l^s_{v+1} are not successors, there must be $x \in L$ such that $l^s_v <_L x <_L l^s_{v+1}$. We enumerate elements into L until we see this x and then set $f_{s+1}(x) = f_{s+1}(r^s_v)$. On the other hand if P_e is an isomorphism from L onto R and never gets permission to diagonalise from D, then we will show that D is in fact computable.

Construction:

Stage 0: Let $f_0(0) = 0$.

Stage $s+1$:

Look for the least $e < s$ such that P_e is not satisfied and there exists $n \leq s+1$ which satisfy the following:

1. $c_n = (l^s_i, l^s_{i+1})$ for some $l_i, l_{i+1} \leq k_s$

2. For all $x, y \leq k_s$ if $\varphi_e^{s+1}(x)\downarrow$, $\varphi_e^{s+1}(y)\downarrow$ and $\varphi_e(x), \varphi_e(y) \leq k_s$, then $x <_L y$ if and only if $\varphi_e(x) <_R \varphi_e(y)$,

3. $\varphi_e^{s+1}(l^s_i)\downarrow, \varphi_e^{s+1}(l^s_{i+1})\downarrow$ and $\varphi_e^{s+1}(l^s_i) = r^s_j, \varphi_e^{s+1}(l^s_{i+1}) = r^s_{j+1}$ for some j such that $l_j, l_{j+1} \leq k_s$,

4. If $m < n$ and $c_m = (l^s_a, l^s_{a+1})$ for some $a < k_s$, then $\varphi^{s+1}_e(l^s_a) \downarrow$, $\varphi^{s+1}_e(l^s_{a+1}) \downarrow$, $\varphi^{s+1}_e(l^s_a) = r^s_p, \varphi^{s+1}_e(l^s_{a+1}) = r^s_{p+1}$, for some p such that $p < k_s$, and

5. Either

   5.1. $\min_{\mathbb{N}}\{l \mid l \in RB_s(l^s_{j+1})\} \geq d_{s+1}$,

   5.2. there is no $x \in \{0, 1, \ldots, e\} \cup \{f_s^{-1}(0), f_s^{-1}(1), \ldots f_s^{-1}(e)\}$ such that $x \in RB_s(l^s_{j+1})$, and

   5.3. $l^s_v = \max_L\{l \mid l \in RB_s(l^s_{j+1})\}$ is not the greatest element of L,

   or

   5.1. $\min_{\mathbb{N}}\{l \mid l \in LB_s(l^s_j)\} \geq d_{s+1}$,

   5.2. there is no $x \in \{0, 1, \ldots, e\} \cup \{f_s^{-1}(0), f_s^{-1}(1), \ldots f_s^{-1}(e)\}$ such that $x \in LB_s(l^s_j)$, and

   5.3. $l^s_v = \min_L\{l \mid l \in LB_s(l^s_j)\}$ is not the least element of L.

Note that we know (non-uniformly) whether L has a least or greatest element and what these elements are. Therefore, condition 5 is computable to check. If there is such an $e < s$ and n, then suppose that we are in the first case of condition 5, that is, $\min_{\mathbb{N}}\{l \mid l \in RB_s(l_{j+1})\} \geq d_{s+1}$. The other case is symmetric. Then let $r^s_j <_R k_s + 1 <_R r^s_{j+1}$, $f_{s+1}(l^s_{j+1}) = k_s + 1, f_{s+1}(l^s_{j+2}) = r^s_{j+1}, \ldots, f_{s+1}(l^s_v) = r^s_{v-1}$. We let $f_{s+1}(x) = f_s(x)$ for all $x <_L l^s_{j+1}$ and $x >_L l^s_v$. Currently, r^s_v is not in the image of f_{s+1} and $f_{s+1}(k_s + 1)$ is undefined. To resolve this and also preserve the isomorphism, we continue to enumerate numbers into R, placing each number in the corresponding position in R that it appears in L, as before. For each $x > k_s + 1$ enumerated during this process, we set $f_{s+1}(x) = x$. We terminate this process when we enumerate x and

$l^s_v <_L x <_L l^s_{v+1}$ or in the case that $v = k_s$, then $l^s_v <_L x$. Now we let $f_{s+1}(x) = r^s_v$ and we place x in R in the same position as $k_s + 1$ appears in L and set $f_{s+1}(k_s + 1) = x$. Note that for all $x \leq k_s$ with $x \notin \{l^s_{j+1}, \ldots, l^s_v\}$, we have $f_{s+1}(x) = f_s(x)$. Hence, for all $x \in \{0, \ldots, e\}$ and all $x \in \{f_s^{-1}(0), \ldots, f_s^{-1}(e)\}$ we have $f_{s+1}(x) = f_s(x)$.

If for each $e < s$ no such n exists, we enumerate $k_s + 1$ into R and let its position in $<_R$ correspond to that in $<_L$. That is, let $k_s + 1 <_R r^s_0$ if $k_s + 1 <_L l^s_0$, $r^s_{k_s} <_R k_s + 1$ if $l^s_{k_s} <_L s + 1$ and $r^s_p <_R k_s + 1 <_R r^s_{p+1}$ if $l^s_p <_L k_s + 1 <_L l^s_{p+1}$ for some $p < k_s$. Then we define $f_{s+1}(k_s + 1) = k_s + 1$ and $f_{s+1}(x) = f_s(x)$ for all $0 \leq x \leq k_s$.

Verification

As each stage is effective, $R = (\mathbb{N}, <_R)$ is a computable linear ordering and so requirement N is satisfied. To satisfy requirement M we need to check that for all x, $\lim_s f_s(x)$ exists. Notice that $f_{s+1}(x) \neq f_s(x)$ only if the e chosen at stage $s+1$ is less than x. In this construction we need to act to satisfy P_e at most once, so these limits exist. Similarly, $f = \lim_s f$ is onto R because P_e is restrained from altering f_{s+1} on $\{f_s^{-1}(0), \ldots, f_s^{-1}(s)\}$.

In this construction $f_{s+1}(x) \neq f_s(x)$ only if $x \in RB_s(l_{j+1})$ and $\min_{\mathbb{N}}\{l \mid l \in RB_s(l_{j+1})\} \geq d_{s+1}$, or $x \in LB_s(l_j)$ and $\min_{\mathbb{N}}\{l \mid l \in LB_s(l_j)\} \geq d_{s+1}$. Hence $f_{s+1}(x) \neq f_s(x)$ only if $x \geq d_{s+1}$ and so $\widehat{M}$ is also satisfied. It remains to show that P_e is satisfied for all e.

Assume that all the requirements $P_0, P_1, \ldots, P_{e-1}$ are met by the end of stage s_0 and assume for a contradiction that P_e is not satisfied. That is, we assume that φ_e is one-to-one and order preserving from $(\mathbb{N}, <_L)$ onto $(\mathbb{N}, <_R)$ and each one of the successor pairs $c_0, c_1, c_2, \ldots$ is mapped by φ_e to another successor pair in R. That

is, for all n if $c_n = (x_n, y_n)$, then both $\varphi_e(x_n), \varphi_e(y_n)$ are defined and $\varphi_e(x_n) \to_R \varphi_e(y_n)$. For notational convenience we let $LB_n^s = LB_s(f_s^{-1}(\varphi_e(x_n)))$ and let $RB_n^s = RB_s(f_s^{-1}(\varphi_e(y_n)))$. If our construction did act to satisfy P_e at stage $s+1$, then f_{s+1} would differ from f_s on one of these blocks. Note that because P_i for $i \geq e$ cannot change f_{s+1} on $f_s^{-1}(0), \ldots, f_s^{-1}(e)$, we have $f^{-1}(0) = f_{s_0}^{-1}(0), \ldots, f^{-1}(e) = f_{s_0}^{-1}(e)$. Therefore, we have fixed the values of $f^{-1}(0), \ldots, f^{-1}(e)$ by stage s_0.

As action was never taken to satisfy P_e then e did not satisfy the requirements $1-5$, with any n, at any stage $s+1 > s_0$. However, for each n, there exists a stage $s_n > s_0$ such that, for all $m \leq n$:

- $\varphi_{e,s_n}(x_m) \downarrow, \varphi_{e,s_n}(y_m) \downarrow$, and
- $x_m, y_m, \varphi_e(x_m), \varphi_e(y_m) \leq s_n$.

Hence e and n satisfy the requirements $1-4$ at all stages $s+1 > s_n$. So for each $s+1 > s_n$ it must be the case that we cannot change f on the LB_n^s or RB_n^s. However for a large class of computable linear orderings, we claim that this forces D to be computable. Before we look at the different cases we will prove the following lemma.

Lemma 2.1.1. *If e is the least number such that P_e is not satisfied and the s_n are as before, then for all $s \geq s_n$*

- $f_{s+1}^{-1}(\varphi_e(x_n)) = f_s^{-1}(\varphi_e(x_n))$, *and*
- $f_{s+1}^{-1}(\varphi_e(y_n)) = f_s^{-1}(\varphi_e(y_n))$.

Proof. Suppose at some stage $s+1 > s_n$ we have $f_{s+1}^{-1}(\varphi_e(x_n)) \neq f_s^{-1}(\varphi_e(x_n))$ or $f_{s+1}^{-1}(\varphi_e(y_n)) \neq f_s^{-1}(\varphi_e(y_n))$ but yet P_e is not satisfied. Note that $f_{s+1}(x) \neq f_s(x)$ if and only if at stage $s+1$ our construction acted to satisfy P_i, for some $i > e$, using a pair $c_m = (x_m, y_m)$ and $x \in RB_s(f_s^{-1}(\varphi_i(y_m)))$ or $x \in LB_s(f_s^{-1}(\varphi_i(x_m)))$.

So either $f_s^{-1}(\varphi_e(y_n)) \in RB_s(f_s^{-1}(\varphi_i(y_m)))$ and the current strategy requires f to change on this block or $f_s^{-1}(\varphi_e(x_n)) \in LB_s(f_s^{-1}(\varphi_i(x_m)))$ and the current strategy requires f to change on this block. Suppose we are in the former case. If the strategy to satisfy P_i can edit f on $RB_s(f_s^{-1}(\varphi_i(y_m)))$ at stage $s+1$ then

- $\min_{\mathbb{N}} RB_s(f_s^{-1}(\varphi_i(y_m))) > d_{s+1}$,
- $(\{0,1,\ldots,i\} \cup \{f_s^{-1}(0), f_s^{-1}(1), \ldots f_s^{-1}(i)\}) \cap RB_s(f_s^{-1}(\varphi_i(y_m))) = \emptyset$, and
- $\max_L RB_s(f_s^{-1}(\varphi_i(y_m)))$ is not the greatest element of L.

Since $f_s^{-1}(\varphi_e(y_n)) \in RB_s(f_s^{-1}(\varphi_i(y_m)))$ we have $RB_s(f_s^{-1}(\varphi_e(y_n))) \subset RB_s(f_s^{-1}(\varphi_i(y_m)))$ and so

- $\min_{\mathbb{N}} RB_s(f_s^{-1}(\varphi_e(y_n))) > d_{s+1}$,
- $(\{0,1,\ldots,e\} \cup \{f_s^{-1}(0), f_s^{-1}(1), \ldots f_s^{-1}(e)\}) \cap RB_(f_s^{-1}(\varphi_e(y_n))) = \emptyset$ as $e < i$, and
- $\max_L RB_s(f_s^{-1}(\varphi_e(y_n)))$ is not the greatest element of L.

As e and n satisfy requirements 1-5 at stage $s+1$ and $e < i$, our construction would act to satisfy P_e before P_i at this stage. □

We return to our claim that D is computable if P_e is not satisfied. So for each $m \in \mathbb{N}$ we need to be able to identify a stage t such that:

- For all $s > t$ there exists n such that $\min(RB_n^s) > m$,
- $(\{0,1,\ldots,e\} \cup \{f^{-1}(0), f^{-1}(1), \ldots f^{-1}(e)\}) \cap RB_n^s = \emptyset$, and
- If L has a maximum element it is not in RB_n^s.

or

- For all $s > t$ there exists n such that $\min(LB_n^s) > m$,
- $(\{0, 1, \ldots, e\} \cup \{f^{-1}(0), f^{-1}(1), \ldots f^{-1}(e)\}) \cap LB_n^s = \emptyset$, and
- If L has a minimum element it is not in LB_n^s.

Case 1: There exists $b \in L \cup \{\infty\}$ such that for all $a <_L b$ there exists a successor pair (x, y) in L such that $a <_L x$ and $y <_L b$. Given m, we want to know if $m \in D$. Without loss of generality, we can assume that $m > e$. Let $a = \max_L\{x \mid x \leq_{\mathbb{N}} m \text{ and } x <_L b\} \cup \{f^{-1}(u) \mid u \leq e \text{ and } f^{-1}(u) <_L b\}$. We wait for a stage t such that there exists a pair $c_p = (x_p, y_p)$ such that $a <_L x_p <_L y_p <_L b$ and both $f_t(x_p)$ and $f_t(y_p)$ are in the range of φ_e. That is, $y_p = f_t^{-1}(\varphi_e(y_n))$ for some successor pair $c_n = (x_n, y_n)$. As P_e is not satisfied, by lemma 2.1.1, we have $f_s^{-1}(\varphi_e(y_n)) = y_p$ for all $s > t$. By our choice of y_p we have that $\min_{\mathbb{N}} RB_n^s > m$ and $(\{0, 1, \ldots e\} \cup \{f^{-1}(0), f^{-1}(1), \ldots f^{-1}(e)\}) \cap RB_n^s = \emptyset$ for all $s > t$. Hence our construction can act to satisfy P_e if at any stage $s \geq t$ if $d_s < m$. As P_e was never satisfied, we conclude that $D_t \upharpoonright m = D \upharpoonright m$. Hence D is computable.

Case 2: There exists $a \in L \cup \{-\infty\}$ such that for all $b >_L a$ there exists a successor pair (x, y) in L such that $a <_L x$ and $y <_L a$. This argument is symmetric to case 1.

Case 3: There exists an r such that each block of successors has length less than r. Again, given m, we want to know if $m \in D$. We wait for a stage t such that there exists a pair $c_p = (l_q^t, l_{q+1}^t)$ and $r - 2$ elements greater than l_{q+1}^t such that $\min_{\mathbb{N}}(l_q^t, l_{q+1}^t, l_{q+2}^t, \ldots, l_{q+r-1}^t) >_L \{0, 1, \ldots, m\} \cup \{0, 1, \ldots, e\} \cup \{f_s^{-1}(0), f_s^{-1}(1), \ldots f_s^{-1}(e)\}$ and $f_t(l_q^t), f_t(l_{q+1}^t)$ are in the range of φ_e. That is, $l_{q+1}^t = f_t^{-1}(\varphi_e(y_n))$, for some successor pair $c_n = (x_n, y_n)$. As P_e is not satisfied, by lemma 2.1.1, we have $f_s^{-1}(\varphi_e(y_n)) = l_{q+1}^t$ for all $s > t$. Since a block of successors has size at most r, we have that

$\min_{\mathbb{N}} RB_n^s > m$ and $\{0, 1, \ldots e\} \cap RB_n^s = \emptyset$ for all $s > t$. Hence our construction can act to satisfy P_e if at any stage $s \geq t$ if $d_s < m$. As P_e was never satisfied, we conclude that $D_t \upharpoonright m = D \upharpoonright m$. Hence D is computable.

□

2.2 A failure of permitting in Remmel's construction.

In the following theorem we see that it is not possible to perform Remmel's argument below every non-zero Δ^0_2 degree. We construct a computable linear order L and an ω-c.e. set D such that any function f demonstrating that L is not computably categorical cannot be computed from D.

Theorem 2.2.1. *There exists a non computable ω-c.e. set D, and a computable linear order L with infinitely many successor pairs such that, if R is a computable linear order which is D-computably isomorphic to L, then R is computably isomorphic to L.*

Proof. In the case that φ_i defines a linear order, L_i denotes the linear order generated by the i-th partial computable function. Without loss of generality we assume that the domain of L_i is $\mathbb{N}$ and so $L_i = (\mathbb{N}, \leq_i)$ where $\leq_i$ is the binary relation computed by φ_i.

We need to build a computable linear order L and an ω-c.e. set D meeting, for all $j, e, i \in \omega$, the following requirements:

- $D_j : \Phi_j \neq D$, and
- $R_{\langle e,i\rangle}$: If $\Phi^D_e : L_i \to L$ is an isomorphism, then there exists a computable isomorphism $\Delta : L_i \to L$.

Note that Δ is a partial computable function built by one of our strategies.

Constructing a computable linear order L

We will construct $L = \cup_s L_s$ in stages with domain $\mathbb{N}$. To construct L we replace each rational in $\mathbb{Q}$ with a finite block of successors. The natural numbers are added

to L in numerical order and we let m denote the next natural number to be added. We begin with $m = 0$. Let $\mathbb{Q} = \{q_0, q_1, \ldots, q_k, \ldots\}$ be a computable enumeration of $\mathbb{Q}$ and let p_k denote the kth prime, where $p_0 = 2$. At stage s we add a finite block of successors to L, known as b_s of length p_s, and specify the linear ordering $<_L$ on $L_s = \cup_s b_s$. The block b_s in L will occupy the position which corresponds to the position q_s in $\mathbb{Q}$. That is $b_j <_L b_k$ in L_s if and only if $q_j <_{\mathbb{Q}} q_k$. We write $b_j <_L b_k$ to mean that for any $x \in b_j$ and $y \in b_k$ we have $x <_L y$. More specifically at stage s, we add $m, m+1, \ldots, m+(p_s-1)$ to L, increment m by p_s and define $<_L$ on these elements as follows: $m <_L m+1 <_L \ldots <_L m+(p_s-1)$. We then refer to this as the block b_s. Hence the domain of L will be $\mathbb{N}$ and L will consist of infinitely many blocks of successors with distinct length. An $R_{\langle e,i\rangle}$ strategy may cause b_s to grow so that it can diagonalise but we will ensure that the block of successors b_s will have length p_s^n for some n at every stage $t \geq s$. Even though b_s may expand at some stage, the initial and end elements will remain unchanged. We denote the length of b_s by $|b_s|$. As each block in L will have unique length, if $L \cong L_i$ then there is a unique isomorphism from L_i onto L.

Constructing an ω-c.e. set D

In the usual way, the D_j strategies ensure that D is not computable. Each D_j strategy will choose a witness x and keep x out of D. If at any stage $\Phi_j(x) = 0$, then the D_j strategy will enumerate x into D and stop. On the other hand, the $R_{\langle e,i\rangle}$ strategy will sometimes need to reset an initial segment of D. However we will be able to computably bound the number of times $D(x)$ will change, hence making D an ω-c.e. set.

Defining the pre-images of the blocks

Within the strategy $R_{\langle e,i\rangle}$ we will refer to the pre-image of b_k in L_i, under Φ_e^D as a_k. We will define a_k at some stage s and after this stage the initial element init_{a_k} and end element end_{a_k} of a_k will never change. So for all $t \geq s$, we define $a_{k,t} = \{a \in L_i \mid \text{init}_{a_k} \leq_{L_i} a \leq_{L_i} \text{end}_{a_k}\}$. Hence if L_i adds a new element to the interior of a_k we consider it part of a_k but if it adds a predecessor to init_{a_k} or a successor to end_{a_k} we do not consider these part of a_k. We refer to the number of elements in this interval as the length of a_k, written as $|a_k|$.

Constructing Δ

If φ_i does define a linear order L_i and $\Phi_e^D : L_i \to L$ is an isomorphism, then the $R_{\langle e,i\rangle}$ strategy must construct a computable isomorphism $\Delta : L_i \to L$. Only this $R_{\langle e,i\rangle}$ makes definitions for this Δ whereas other $R_{\langle e,i\rangle}$-strategies define their own versions of Δ. The basic strategy for constructing Δ is to copy Φ_e^D as it converges onto blocks in L. That is, we set $n = 0$ and wait for a stage s_n where we see that Φ_e^D is onto b_n. We then define a_n to be the pre-image of b_n at this stage, by defining $\Delta(a) = \Phi_e^{D_{s_n}}(a)$ for all $a \in a_n$. We then increment n by 1 and repeat.

If n tends to infinity and for each n, the interval a_n, the block b_n and the isomorphism between them $\Phi_e^D(a_n)$ do not change after stage s_n, then we will have defined a computable function Δ from L_i to L. Note that we say $\Phi_e^D(a_n)$ does not change after stage s_n, if for all $a \in a_n$, we have $\Phi_e^D(a) = \Delta(a)$ at all $s > s_n$. However if a_n, b_n or $\Phi_e^D(a_n)$ do change at any stage, then the corresponding $R_{\langle e,i\rangle}$ strategy must respond. Note that $\Phi_e^D(a_n)$ may change at some stage $s > s_n$, if a D_j requirement adds an element to D below the use of $\Phi_e^D(a_n)$ or another $R_{\langle e',i'\rangle}$ requirement changed D below

this use. Although each strategy define their own Δ and a_n intervals, all strategies are working with the same blocks in L. Hence if one strategy makes changes to a block, it will effect all other strategies. We now describe the $R_{\langle e,i\rangle}$ strategy's response:

Either a_n or b_n have increased in size: That is, at some stage $s > s_n$, the linear order L_i added a new element to a_n or some $R_{\langle e,i\rangle}$ strategy expanded the block b_n. We now distinguish three cases:

Case 1: If $|a_n| > |b_n|$ we set $D \upharpoonright s_n = D_{s_n} \upharpoonright s_n$ and restrain any numbers from entering b_n and $D \upharpoonright s_n$ from this stage on and stop. In doing so we have ensured that Φ_e^D cannot be an isomorphism from L_i to L.

Case 2: If $|a_n| < |b_n|$ then we wait for a_n to expand and converge onto b_n again. Alternatively, a different set of elements in L_i may converge onto b_n. In that case either Φ_e^D is not 1-1 from L_i into L or Φ_e^D has changed on a_n.

Case 3: If $|a_n| = |b_n|$ and $\Phi_e^D(a_n)$ has not changed since stage s_n, then a_n and b_n have new elements in corresponding positions and we define $\Delta(a) = \Phi_e^D(a)$ for the new elements in a_n.

The computation $\Phi_e^D(a_n)$ has changed: A D_j strategy may enumerate an element into D below the use of a computation $\Phi_e^D(a_n)$ after the stage s_n, or another $R_{\langle e,i\rangle}$ strategy may change D below this use, at which we defined $\Delta(a_n)$. If this happens, then $\Phi_e^D(a_n)$ may change or not converge at all. If $\Phi_e^D(a_n)$ no longer converges, than we simply wait to see if it converges again. If not, then it is not an isomorphism from L_i into L.

However, if $\Phi_e^D(a_n)$ does converge but has changed since stage s_n, this presents us with a problem. The computable function Δ which we are constructing may no longer be extendable to an isomorphism, while Φ_e^D may be an isomorphism. The $R_{\langle e,i\rangle}$ strategy responds by preventing the function Φ_e^D from possibly being an isomorphism

from L_i to L. It does so as follows:

It begins by restraining any new numbers from entering $D \upharpoonright s_n$ or the block b_n. Hence we have preserved this computation.

Case 1: $\Phi_e^D(a_n)$ is not part of one block in L. In this case we wait. The elements of a_n in L are being mapped onto multiple blocks in L. If Φ_e^D will extend to an isomorphism then the interval a_n must be infinite and so at some stage we will see that $|a_n| > |b_n|$. The $R_{\langle e,i \rangle}$ strategy now restores the computation at stage s_n by setting $D \upharpoonright s_n = D_{s_n} \upharpoonright s_n$. Now the elements that belonged to a_n at stage s_n are being mapped into b_n once more and this computation cannot change after this stage. As $|a_n| > |b_n|$ and $|b_n|$ will never increase, the function Φ_e^D cannot extend to an isomorphism and so the strategy stops.

Case 2: $\Phi_e^D(a_n)$ is part of a block b_k in L. In this case we add new elements to the interior of b_k until $|b_k| > |b_n|$ and in this process break at least one successor pair in b_k. So if Φ_e^D is to extend to an isomorphism then L_i must expand a_n to the size of b_k. If this happens, then we will have $|a_n| > |b_n|$ and proceed as in Case 1.

The tree of strategies

The full construction takes place on a tree of strategies $T = \Lambda^{<\omega}$ where

$$\Lambda = \{s_{\text{fin}} <_\Lambda \infty <_\Lambda \ldots d_n <_\Lambda r_n <_\Lambda w_n <_\Lambda \ldots <_\Lambda b_1 <_\Lambda r_1 <_\Lambda w_1 <_\Lambda$$
$$b_0 <_\Lambda r_0 <_\Lambda w_0 <_\Lambda w_I <_\Lambda w\}$$

is the set of outcomes of a strategy. We effectively order the requirements as follows: Let $D_0 < R_0 < D_1 < R_1 < \ldots < D_j < R_j < \ldots$. Furthermore, all strategies $\alpha \in T$

of length j are assigned to the jth requirement. The linear ordering $<_\Lambda$ on the set of outcomes Λ induces a lexicographical ordering of the tree of strategies T. That is, $\alpha <_{lex} \beta$ if there are $\gamma \in T$ and $o_\alpha <_\Lambda o_\beta$ such that $\gamma^\frown\langle o_\alpha\rangle \subseteq \alpha$ and $\gamma^\frown\langle o_\beta\rangle \subseteq \beta$. Note that $\alpha <_{lex} \beta$ means that α is left of β on the tree of strategies. We say that α has higher priority than β, written as $\alpha < \beta$, if $\alpha <_{lex} \beta$ or $\alpha \subset \beta$.

Other restraints

We let s_α denote the first stage at which the strategy α is eligible to act after last being initialised. As we have seen, the $R_{\langle e,i\rangle}$ strategies may expand a block in L. We need to ensure that each block gets changed at most finitely often. To do this we do not allow an $R_{\langle e,i\rangle}$ strategy α to change any block b_k where $k < s_\alpha$.

Similarly to ensure that there exists some s where $D \upharpoonright s = D_s \upharpoonright s$ we require that the witness chosen by the D_j strategy is greater than s_α.

Notation

As a notational convenience, we will write $\Delta(a_n) = b_n$ at stage s to mean that $\Delta(a) = \Phi_e^{D_s}(a)$ for each $a \in a_n$. Similarly we will say that $\Delta(a_n) \neq b_n$ at stage s if for some $a \in a_n$ we have $\Delta(a) \neq \Phi_e^{D_s}(a)$. We will also write $\Phi_e^D(a_n)\uparrow$ to mean that for some $a \in a_n$ we have $\Phi_e^D(a)\uparrow$. Similarly we will write $\Phi_e^D(a_n)\downarrow$ to mean that for all $a \in a_n$ we have $\Phi_e^D(a)\downarrow$. We let $\Phi_e^D(x)[s]$ denote the computation $\Phi_{e,s}^{D_s}(x)$.

The full strategies

Strategy for D_j:

1. Pick a unused witness x larger than any number mentioned so far in the construction and keep x out of D. In particular x is chosen to be greater than any restraints placed on D by higher priority requirements.

2. Wait for a stage such that $\Phi_j(x) = 0$.

3. Enumerate x into D and stop.

Outcomes:

w: Wait at step 2 infinitely often. Then $D(x) = 0 \neq \Phi_j(x)$.

s_{fin}: Stop at step 3 after x has been enumerated into D. Then $D(x) = 1 \neq 0 = \Phi_j(x)$.

Strategy for $R_{\langle e,i\rangle}$ requirement α:

Note that if at any point, we see that φ_i violates one of the Π^0_1 conditions to be a linear order or that Φ_e^D is not one-to-one or order preserving on L_i, then we stop the current action for α and take outcome s_{fin}. All parameters are local to this strategy but we suppress the subscript α for readability, and often suppress the stage number subscript as well. When needed, we will use n_α or $n_{\alpha,s}$ to denote the parameter n for the requirement α at stage s.

(A) Set-up phase. At the first stage at which α is eligible to act, choose the parameter v large and take outcome w_I. At future α-stages, check whether

- $\Phi_e^D(x)[s]$ converges for all $x \leq v$, and
- $\Phi_e^D[s]$ maps finite intervals in L_i onto each of the finite blocks $b_{0,s}, \ldots, b_{v,s}$.

If no, take outcome w_I. If yes, then for $j \leq v$, define a_j^{init} and a_j^{end} so that

$$a_{j,s} = \{x \in L_i : a_j^{\text{init}} \leq_{L_i} x \leq_{L_i} a_j^{\text{end}}\}$$

is the finite interval mapped onto b_j. For stages $t \geq s$, we let $a_{j,t}$ denote the set of $x \in L_i$ such that $a_j^{\text{init}} \leq_{L_i} x \leq_{L_i} a_j^{\text{end}}$. Let u_v denote the maximum use of the computations seen in this phase. Set $\Delta(a_j) = \Phi_e^D(a_j)$ for $j \leq v$ and proceed to (B).

(B) Main loop. At the first stage we enter this main loop, set the parameter $n = v + 1$.

(B.1) Check whether L_i has placed any new elements in some a_j for $j < n$ so that $|a_{j,s}| > |b_{j,s}|$. If so, take outcome s_{fin} at this and each future α-stage. If not, check whether $\Phi_e^D(n)[s]$ converges and whether $\Phi_e^D[s]$ maps a finite interval in L_i onto the block $b_{n,s}$. If not, take outcome w_n. If so, proceed to (B.2).

(B.2) Let s_0 denote the α-stage at which the convergences in (B.1) occur. Define a_n^{init}, a_n^{end} and $a_{n,s}$ as above and let u_n denote the maximum use for $\Phi_e^D(j)[s_0]$ and $\Phi_e^D(a_j)[s_0]$ for $j \leq n$. Define $\Delta(a_n) = \Phi_e^D(a_n)[s_0]$ and take outcome ∞.

(B.3) At the next α-stage s_1 after s_0, act as follows based on the case satisfied.

(B.3.1) There is a $j \leq n$ such that $|b_{j,s_1}| > |b_{j,s_0}|$. At this stage and future α-stages, check if $a_{j,s}$ has grown so that $|a_{j,s}| = |b_{j,s_1}|$ and $\Phi_e^D(a_j)[s] = b_j$. If not, take outcome $\alpha * r_n$. If so, define $\Delta(a) = \Phi_e^D(a)[s]$ for any a in $a_{j,s}$ on which Δ was not previously defined, increment n by 1 and return to (B.1).

(B.3.2) For each $j \leq n$, we have $|b_{j,s_1}| = |b_{j,s_0}|$, but $D_{s_1} \upharpoonright u_n \neq D_{s_0} \upharpoonright u_n$. In this case, enter the recovery phase below.

(B.3.3) For each $j \leq n$, we have $|b_{j,s_1}| = |b_{j,s_0}|$, and $D_{s_1} \upharpoonright u_n = D_{s_0} \upharpoonright u_n$. In this case, increment n by 1 and return to (B.1).

(C) Recovery phase. We continue to let s_0 and s_1 denote the α-stages referenced above.

(C.1) Check whether $\Phi_e^D(x)[s]$ converges for all $j \leq n$ and $x \in a_{j,s_0}$. If not, take outcome r_n. If so, proceed to (C.2).

(C.2) Let $s_2 > s_1$ be the α-stage at which we see these convergences. Check if $\Phi_e^D(x)[s_2] = \Phi_e^D(x)[s_0]$ for all $j \leq n$ and $x \in a_{j,s_0}$. If so, increment n by 1 and return to (B.1). If not, proceed to (C.3).

(C.3) We have at least one $j \leq n$ and $x \in a_{j,s_0}$ such that $\Phi_e^D(x)[s_2] \neq \Phi_e^D(x)[s_0]$. We act according to which of the following two cases applies.

(C.3.1) There is a $j \leq n$ and an L_i-successor pair $x, y \in a_{j,s_0}$ such that $\Phi_e^D(x)[s_2], \Phi_e^D(y)[s_2] \in b_{k,s_2}$ for some $k \neq j$. In this case, we add new elements to b_{k,s_2+1} between $\Phi_e^D(x)[s_2]$ and $\Phi_e^D(y)[s_2]$ so that $|b_{k,s_2+1}|$ is a power of the k-th prime p_k and so that the interval between $\Phi_e^D(x)[s_2]$ and $\Phi_e^D(y)[s_2]$ has size greater than b_{j,s_2}. We take outcome d_n. At future α-stages, we check whether the L_i-interval between x and y has grown to have size greater than $|b_{j,s_2}|$. If not, we take outcome d_n. If so, we reset $D_s \upharpoonright u_n = D_{s_0} \upharpoonright u_n$ and take outcome s_{fin} at all future α-stages.

(C.3.2) If there is no such index j and pair x, y, then we take outcome d_n without expanding any b_{p,s_2} blocks. At future α stages s, we check whether there is an index $j \leq n$ such that $|a_{j,s}| > |b_{j,s}|$. If not, we

take outcome $\alpha * d_n$. If so, we we reset $D_s \upharpoonright u_n = D_{s_0} \upharpoonright u_n$ and take outcome s_{fin} at all future α-stages.

Outcomes:

w_n: Wait at step (B.1) forever for some n. Then Φ_e^D is not an isomorphism form L_i onto L.

r_n: Wait at step (B.3.1) or (C.1) forever. Then for some n, either $|a_n| < |b_n|$ or Φ_e^D does not converge on some $a \in a_n$ and hence is not an isomorphism from L_i onto L.

d_n: Wait at step (C.3) forever. Then Φ_e^D is not an isomorphism from L_i onto L.

∞: Eventually reach step (B.1) for all n. Then we have defined a computable isomorphism Δ from L_i onto L.

s_{fin}: Stop because L_i is not a linear order or Φ_e^D is not one-to-one or order preserving. Alternatively stop at step (B.1) or (C.3) because for some n, either $|a_n| > |b_n|$. In all cases, Φ_e^D is not an isomorphism from $L_i \to L$.

The construction

The construction is performed in ω many stages s. Each stage consists of substages $t \leq s$. At substage t of stage s, a strategy α of length t with the current correct guess about the outcomes of the strategies $\beta \subset \alpha$ is eligible to act. Strategy α will then proceed according to its strategy from where it left off the last time it was eligible to act. At the end of each substage $t \leq s$, the strategy α will determine its outcome o and thus the strategy $\alpha^\frown\langle o\rangle$ is eligible to act at the next substage $t+1$. We define

the current true path TP_s at stage s to be the longest strategy eligible to act at stage s. At the end of stage s, we initialise all strategies to the right of TP_s in the tree of strategies. Initialising a strategy means cancelling all of its parameters and requiring it to restart as though it had never acted. This completes the description of the construction.

Verification

Denote by $TP \in [T]$ the true path of the construction, i.e. the lim inf of TP_s as s tends to infinity. Note that TP will be an infinite path through T. We observe that the true outcome is the leftmost current outcome that is achieved infinitely often. Either the infinitary outcome ∞ achieved at infinitely many stages or a finitary outcome (s_{fin}, w, w_n, r_n, d_n or w_I) achieved at cofinitely many stages. Note that our construction ensures the following facts, once a strategy on the true path has been initialised for the last time:

- A true finitary outcome once current must by current from that stage on.
- In the case of a true infinitary outcome, any current finitary outcome, once no longer current, can never be current again.
- If α is on the true path and α is last initialised at stage s, then no β to the left of α on the tree of strategies is eligible to act after stage s.

Lemma 2.2.1. *Let α be an $R_{\langle e,i\rangle}$ strategy. If α leaves the set-up phase at stage s, then no requirement can injure the computations $\Phi_e^D(a_j)[s]$ for $j \leq v$ by changing D below u_v without initialising α.*

Proof. Assume α is not initialised after s. First, we show that no D_j-strategy can enumerate a number into D below u_v after stage s. Let β be a D_j-strategy. If β is to the left of α on the tree of strategies, then β cannot be eligible to act. If $\beta * w \subseteq \alpha$ and β enumerates an element into D, then the path moves to the left of α and α is initialised. If $\beta * s_{\text{fin}} \subseteq \alpha$, then β has already enumerated its witness into D by stage s. If β is to the right of α, then β is initialised at stage s and will only pick witnesses larger than u_v at future stages. Similarly, since α first takes an outcome other than $\alpha * w_I$ at stage s (and will not take outcome $\alpha * w_I$ again unless initialised), if β is below α on the tree of strategies, it will choose witnesses greater than u_v. Therefore, no D_j strategy can enumerate a number into D below u_v unless α is initialised.

Second, let β be an $R_{\langle e',i'\rangle}$ strategy. Such a strategy could cause a change in D below u_v in its recovery phase by resetting D in (C.3.1) or (C.3.2). If β is to the left of α, it will not be eligible to act. If $\beta * s_{\text{fin}} \subseteq \alpha$, then β will continue to take outcome $\beta * s_{\text{fin}}$ unless it is initialised, in which case, α will be initialised as well. If $\beta \subset \alpha$ but $\beta * s_{\text{fin}} \not\subseteq \alpha$, then when β resets D, it moves the path left of α and initialises α. Finally, suppose β is to the right of α or below an α outcome other than $\alpha * w_I$. For a contradiction, let $t > s$ be the least stage at which such a β strategy changes D below u_v and fix the β strategy which makes the change. β must be acting in (C.3.1) or (C.3.2) by resetting $D_t \upharpoonright u_{n,\beta} = D_{s_0} \upharpoonright u_{n,\beta}$ for an appropriate β-parameter $u_{n,\beta}$ and a stage $s_0 < t$ from β action in (B.3.2). However, β was initialised at stage s (or had not acted before stage s) and hence $s < s_0$. Therefore, $D_{s_0} \upharpoonright u_{n,\beta}$ already disagreed with D_s below u_v, contradicting the minimality of t. □

A similar analysis will show the following two lemmas.

Lemma 2.2.2. *Let α be an $R_{\langle e,i\rangle}$ strategy. While α waits at (B.1) no requirement*

can injure the computations $\Phi_e^D(a_j)[s]$ *for* $j < n$ *without initialising* α.

Lemma 2.2.3. *Let* α *be an* $R_{\langle e,i\rangle}$ *strategy. While* α *waits at (B.3.1) no requirement can injure the computations* $\Phi_e^D(a_j)[s]$ *for* $j \leq n$ *without initialising* α.

Lemma 2.2.4. *Let* α *be an* $R_{\langle e,i\rangle}$ *strategy. If* α *resets* D *in (C.3.1) or (C.3.2) at stage* s*, then no strategy can injure the computations* $\Phi_e^D(a_j)[s]$ *without initialising* α.

Proof. No strategy β to the left of α can act without initialising α. If $\beta \subset \alpha$ changed $D \upharpoonright u_n$, either by enumerating an element into D for a D_j strategy or resetting D for an $R_{\langle e',i'\rangle}$ strategy, then β takes outcome $\beta * s_{\text{fin}}$ for the first time. Therefore, $\beta * s_{\text{fin}}$ is to the left of α and initialises α. Suppose $\alpha * s_{\text{fin}} \subseteq \beta$ or β is to the right of α. If $\alpha * s_{fin} \subseteq \beta$ then β has never acted before stage s and if β is to the right of α on the tree of strategies it is initialised at stage s. In either case, the same argument as in the last paragraph of the proof of Lemma 2.2.1 shows that β cannot change D below u_n.

□

Lemma 2.2.5. *Let* α *be an* $R_{\langle e,i\rangle}$ *strategy such that* α *leaves the set-up phase at stage* s*. Unless* $R_{\langle e,i\rangle}$ *is initialised, at each future* α*-stage* $t > s$*,* $b_{j,t} = b_{j,s}$ *for all* $j \leq v$.

Proof. Assume α is not initialised after s. A block $b_{j,t}$ can only be expanded by an R-strategy acting in (C.3.1), so let β be an $R_{\langle e',i'\rangle}$-strategy which is not to the left of α on the tree of strategies.

First, suppose that $\beta \subset \alpha$. When β acts in (C.3.1), it switches from outcome $\beta * \infty$ or $\beta * r_n$ to $\beta * d_n$. Since α has been eligible to act before and is not initialised by this change, we must have $\beta * \infty \subseteq \alpha$. However, once β takes outcome $\beta * d_n$ (since β cannot be initialised again), it either always takes outcome $\beta * d_n$ or it switches to

$\beta * s_{\text{fin}}$. In the former case, there are no more α-stages, and in the later case, α is initialised.

Second, suppose β either is to the right of α or is below α but not below $\alpha * w_I$ (since the strategies below $\alpha * w_I$ will not be eligible to act after stage s unless α is initialised). For a contradiction, assume β is the first strategy to expand a block $b_{p,s'}$ with $p \leq v_\alpha$ at a stage $s' > s$. Thus, $b_{p,s'} = b_{p,s}$ before β acts in (C.3.1) to increase the size of $b_{p,s'+1}$. Since β was initialised at stage s or was not eligible to act until after stage s, $v_\beta > v_\alpha$ and so $p < v_\beta$. Fix the elements x and y from (C.3.1) such that $\Phi_{e'}^D(x)[s']$ and $\Phi_{e'}^D(y)[s']$ are in $b_{p,s'}$. By lemma 2.2.1 applied to β, $\Phi_{e'}^D(a_{p,\beta})[s']$ remains the same as when β left the set-up phase. Since $b_{p,s'} = b_{p,s}$, we have that $\Phi_{e'}^D(a_{p,\beta})[s']$ maps onto $b_{p,s'}$ and so $\Phi_{e'}^D[s']$ is not one-to-one. Therefore, by construction, rather than expanding $b_{p,s'}$, β takes outcome s_{fin} to preserve the failure of one-to-oneness.

□

A similar analysis will show the following lemmas.

Lemma 2.2.6. *Let α be an $R_{\langle e,i\rangle}$ strategy which starts waiting at (B.1) in a loop cycle at stage s. For each α-stage $t > s$ at which R returns to this same cycle at (B.1), we have $b_{j,t} = b_{j,s}$ for all $j \leq n$.*

Lemma 2.2.7. *Let α be an $R_{\langle e,i\rangle}$ strategy which starts waiting at (B.3.1) in a loop cycle at stage s. For each α-stage $t > s$ at which α returns to this same cycle at (B.3.1), we have $b_{j,t} = b_{j,s}$ for all $j \leq n$.*

Lemma 2.2.8. *Let α be an $R_{\langle e,i\rangle}$ strategy which enters (C.1) or (C.3.2) at stage s_2. Unless α is initialised, at each α-stage $t > s_2$ at which α returns to the same cycle of (C.1) or (C.3.2), we have $b_{j,t} = b_{j,s_2}$ for all $j \leq n$.*

Lemma 2.2.9. *Let α be an $R_{\langle e,i\rangle}$ strategy which enters (C.3.1) at stage s_2. Unless α is initialised, at each α-stage $t > s_2$ at which α returns to the same cycle of (C.3.1), we have $b_{j,t} = b_{j,s_2+1}$ for all $j \leq n$.*

The above lemmas show that if α is an $R_{\langle e,i\rangle}$ strategy, which has defined $\Delta_\alpha(a_j)$ for $j \leq n$, then it can only be injured when it takes outcome ∞ at some stage s. This is because some strategy $\beta \supset \alpha * \infty$ may have started acting at an earlier stage. If β is an $R_{\langle e,i\rangle}$ strategy it may expand some block b_j for some $j \leq n$ or reset D causing a change to $D \upharpoonright u_n$. Similarly, if β is a D-strategy, it could enumerate a witness chosen at an earlier stage into D, again causing a change to $D \upharpoonright u_n$. The next lemma shows that at most one of these can occur each time α takes outcome ∞.

Lemma 2.2.10. *Let α be an $R_{\langle e,i\rangle}$ strategy which starts takes outcome $\alpha * \infty$ at stage s_0. Unless α is initialised, at the next α-stage s_1, we could have that $|b_{j,s_1}| > |b_{j,s_0}|$ for some $j \leq n$ or that $D_{s_1} \upharpoonright u_n \neq D_{s_0} \upharpoonright u_n$, but not both.*

Proof. Suppose $\beta \supset \alpha * \infty$ injures α at stage s_0. If β is a D-strategy that enumerates a new element into $D \upharpoonright u_n$ or an R-strategy that changes $D \upharpoonright u_n$ in $(C.3)$, then β takes the new outcome s_{fin} at stage s_0. No R-strategy $\beta' \supset \beta * s_{\text{fin}}$ can cause a block b_{j,s_0} to grow at stage s_0 because the strategies are acting for the first time. If β is an R-strategy that expands the block b_j for some $j \leq n$, then it takes a new outcome $\beta * d_n$ at stage s_0. No $\beta' \supset \beta$ can change $D_{s_0} \upharpoonright u_n$ at stage s_0 because they are acting for the first time. Hence only $\beta \supset \alpha * \infty$ can injure α at stage s_0.

In addition, we claim that no other strategy can change $D_{s_0} \upharpoonright u_n$ or cause a block b_j with $j \leq n$ to grow before stage s, without initialising α. If $\beta \subset \alpha$ causes either of these changes, it would move the path to the left of α, initialising α. If β is to the left of α, it cannot act without initialising α.

Finally, suppose β is to the right of α. If β is a D-strategy, it will only choose witnesses $x > u_n$ and so will not change $D \upharpoonright u_n$. Assume β is an $R_{\langle e',i' \rangle}$ strategy. In this case, β cannot increase the size of a block b_j for $j \leq n$ by the argument given in the last paragraph of the proof of Lemma 2.2.5, and β cannot change $D \upharpoonright u_n$ by the argument given in the last paragraph of the proof of Lemma 2.2.1.

□

The following lemma shows that D is ω-c.e.

Lemma 2.2.11. *Let β be a D-strategy and let s be a β-stage at which β enumerates its current witness x_β into D. The only strategies which could later remove x_β from D or add x_β to D are the R-strategies α such that $\alpha * \infty \subseteq \beta$ and each such strategy can remove or add x at most once.*

Proof. An R-strategy α can only change D when it acts in (C.3.1) or (C.3.2), and it can only reach the recovery phase if D changes below its current use $u_{n_\alpha,\alpha}$ between a stage s_0 when α takes outcome $\alpha * \infty$ and the next α-stage s_1. We claim that no R-strategies α except those with $\alpha * \infty \subseteq \beta$ can satisfy the following conditions: there is an $\alpha * \infty$-stage $s_0 \leq s$ but no subsequent α-stage before stage s, such that α has not been initialised between s_0 and the end of stage s, and x_β is less that the current α use $u_{n_\alpha,\alpha}$. If $\alpha * \infty$ is to the left of β on the tree of strategies, then β was initialised at the last $\alpha * \infty$ stage and hence chose its witness greater than the current α use $u_{n_\alpha,\alpha}$. If α is to the right of β or below $\beta * w$, then α is initialised at the end of stage s. If α is below $\beta * s_{\text{fin}}$, then α has not been eligible to act yet and so there have been no $\alpha * \infty$-stages. Therefore, the only R-strategies which could respond to this initial enumeration of x into D are those such that $\alpha * \infty \subseteq \beta$.

It remains to see what happens if one of these R-strategies moves x in or out of D. Let $\alpha * \infty \subseteq \beta$. If α resets D in its recovery phase (and so removing or adding x), then it takes outcome $\alpha * s_{\text{fin}}$. This initialises any R-strategy α' such that $\alpha * \infty \subseteq \alpha'$. In particular, if $\alpha' * \infty \subseteq \beta$, α' cannot subsequently cause a change in x. Furthermore, by the same analysis as in the previous paragraph, after α resets D, the only strategies which could cause a further change in the status of x are R-strategies γ such that $\gamma * \infty \subseteq \alpha$. The statement of the lemma now follows. □

Lemma 2.2.12. *Every strategy $\alpha \subset TP$ ensures the satisfaction of its requirement.*

Proof. To see this we proceed by simultaneous induction on the length of α. Fix s_α least such that $\alpha \subseteq TP_{s_\alpha}$ and α has not been initialised since stage s_α. We now distinguish cases for α.

Case 1: α is an D_j-strategy: Then α picks a permanent witness x at stage s_α.

If at any α-stage $s > s_\alpha$, we have $\Phi_j(x) = 0$ then α enumerates x into D at stage s. Hence $D_s(x) = 1 \neq \Phi_j(x)$. We claim that no strategy β will remove x from D after stage s, and hence D_j will be satisfied. We saw in lemma 2.2.11 that the only strategies that could remove x from D or add x to D after stage s are the R-strategies β such that $\beta * \infty \subset \alpha$. However if this happens at some stage $s_1 > s$ then β takes outcome s_{fin} at stage s_1, initialising α after stage s_α for a contradiction.

Otherwise $D(x) = 0 \neq \Phi_j(x)$. As each D_j strategy selects a unique witness, no other D_j strategy will enumerate x into D. In addition, no R-strategy β will add x to D since all computations copied by these strategies will have $D(x) = 0$.

Case 2: α is an $R_{\langle e,i \rangle}$-strategy:

If α stops at some α-stage $s > s_\alpha$ because L_i is not a linear order then no other strategy can injure this and so α is satisfied.

If at some α-stage $s > s_\alpha$ it happens that Φ_e^D is not one-to-one or order preserving, then α stops and we claim that no other strategies will change $D \upharpoonright s$ after this stage. That is, $D_s \upharpoonright s = D_{s'} \upharpoonright s$ for all $s' > s$. No β to the left of α will be eligible to act after stage s_α by assumption. Note that any D-strategy or R-strategy which changes D, switches its output to s_{fin}. Since α has the correct guess about all the strategies $\beta \subset \alpha$, no $\beta \subset \alpha$ changes $D \upharpoonright s$ after stage s. Any $\beta \supset \alpha$ or β to the right of α which act after stage s will be initialised for the first time after stage s. Hence any witnesses chosen by D-strategies will be greater than s and any computations copied by $R_{\langle e,i\rangle}$-strategies will have $D \upharpoonright s = D_s \upharpoonright s$ as in the proof of Lemma 2.2.1. Hence α is satisfied.

If α takes outcome $\alpha * w_I$ at every α-stage after s_α, then either $\Phi_e^D(x)$ diverges for some $x \leq v$ or Φ_e^D does not map onto the blocks $b_0, \ldots b_v$. In either case, $R_{\langle e,i\rangle}$ is satisfied. Therefore, assume that α eventually leaves the set up phase.

Suppose at some α-stage $s > s_\alpha$, the α strategy takes outcome s_{fin} in $(B.1)$, because $|a_{j,s}| > |b_{j,s}|$. As α is never initialised again there are infinitely many α-stages, the same analysis used to prove Lemma 2.2.2 and 2.2.6 shows that $b_{j,t} = b_{j,s}$, for all $t > s$ and that $\Phi_e^D(a_j) = \Phi_{e,s}^{D_s}(a_j)[s]$. Therefore, Φ_e^D cannot be an isomorphism from L_i to L, so $R_{\langle e,i\rangle}$ is satisfied.

If α waits forever at step (B.1) for some for some n, taking outcome $\alpha * w_n$, then Φ_e^D does not converge onto the block b_n in L. Therefore, Φ_e^D is not an isomorphism from $L_i \to L$ and so α is satisfied.

By the previous paragraph, we can assume that for each $n \geq v$, if α starts (B.1) with parameter n, then there is a future α-stage s_n at which α takes outcome $\alpha * \infty$. To prove that α satisfies its requirement, we split into two cases. For the first case, assume that we never enter (C.3) in the recovery phase. Once we have analyzed this

case, we will return to what happens if α reaches (C.3).

Assume that α never reaches (C.3). Fix n and s_n as above and consider what happens at the first α-stage $s'_n > s_n$. By Lemma 2.2.10, we could have $|b_{j,s'_n}| > |b_{j,s_n}|$ for some $j \leq n$ or $D_{s'_n} \upharpoonright u_n \neq D_{s_n} \upharpoonright u_n$ or neither, but not both. Consider each of the three possible cases separately.

First, if $|b_{j,s'_n}| = |b_{j,s_n}|$ for all $j \leq n$ and $D_{s'_n} \upharpoonright u_n = D_{s_n} \upharpoonright u_n$, then we have $\Delta(a_j) = \Phi_e^D(a_j)[s_n] = \Phi_e^D(a_j)[s'_n]$ is correctly defined and onto b_j for each $j \leq n$. We return to (B.1) with the parameter n incremented by 1.

Second, suppose $|b_{j,s'_n}| > |b_{j,s_n}|$ for some $j \leq n$, so α acts in (B.3.1). Since α is never initialized again, by Lemmas 2.2.3 and 2.2.7, at future stages $s > s'_n$ the computations $\Phi_e^D(a_j)[s] = \Phi_e^D(a_j)[s_n] = \Phi_e^D(a_j)[s'_n]$ remain intact and the blocks $b_{j,s} = b_{j,s'_n}$ for $j \leq n$ do not grow while α waits for L_i to increase the size of the blocks a_j for each $j \leq n$ such that $|b_{j,s'_n}| > |b_{j,s_n}|$. If some a_j block never grows to the proper size then $R_{\langle e,i\rangle}$ is satisfied since Φ_e^D does not map a_j onto b_j. Therefore, we can assume that whenever α enters (B.3.1), the a_j blocks do grow to the appropriate size. Let s''_n denote the stage at which this occurs for parameter n. We expand the definition of Δ to correctly map the new elements of each a_{j,s''_n} to the correct elements of $b_{j,s''_n} = b_{j,s'_n}$ and note that we return to (B.1) with $\Delta(a_j) = \Phi_e^D(a_j)[s''_n]$ correctly defined and onto b_j for all $j \leq n$.

Third, suppose $D_{s'_n} \upharpoonright u_n \neq D_{s_n} \upharpoonright u_n$ and α enters the recovery phase at (C.1). By Lemma 2.2.8, at all stages $s > s'_n$, we have $b_{j,s} = b_{j,s'_n} = b_{j,s_n}$ for all $j \leq n$ while α waits for the computations $\Phi_e^D(a_j)[s]$ to converge. If some computation never converges, then Φ_e^D is not total and $R_{\langle e,i\rangle}$ is satisfied. Therefore, we can assume that whenever α enters (C.1) with parameter n, there is a stage $s''_n > s'_n$ at which $\Phi_e^D(a_j)[s''_n]$ converges for all $j \leq n$. As we are assuming that we never reach (C.3), we

have $\Phi_e^D(a_j)[s_n''] = \Phi_e^D(a_j)[s_n]$ for all $j \leq n$. Since we have $b_{j,s_n''} = b_{j,s_n}$, we return to (B.1) with $\Delta(a_j) = \Phi_e^D(a_j)[s_n'']$ correctly defined and onto b_j for all $j \leq n$.

From the analysis of these three cases, we note the following. If α never reaches (C.3), then either α satisfies its requirement by getting stuck in (B.3.1) or (C.1) forever, or for every $n \geq v$, α returns to (B.1) at a stage $t_n > s_n$ with the parameter n incremented by 1 and $\Delta(a_j) = \Phi_e^D(a_j)[t_n]$ correctly defined and onto b_j for all $j \leq n$. If α is never stuck in (B.3.1) or (C.1), then this condition says that $\Delta = \Phi_e^D$ is one-to-one, order preserving and onto every b_j. Therefore, Δ is an isomorphism and $R_{\langle e,i\rangle}$ is satisfied.

Finally, we consider the case when α reaches (C.3) with parameter n. As above, let s_n'' be the α-stage at which α sees the convergence computations in (C.1). First, suppose that α acts in (C.3.1) with witnesses x and y and increases the size of $b_{k,s_n''+1}$. By Lemma 2.2.9, the computations $\Phi_e^D(a_j)[s] = \Phi_e^D(a_j)[s_n'']$ remain intact while α waits in (C.3.1) for the L_i interval between x and y to grow. In particular, if this interval does not grow to match the size of the interval between $\Phi_e^D(x)[s_n'']$ and $\Phi_e^D(y)[s_n'']$, then Φ_e^D does not map the L_i interval between x and y onto the L interval between $\Phi_e^D(x) = \Phi_e^D(x)[s_n'']$ and $\Phi_e^D(y) = \Phi_e^D(y)[s_n'']$, and so α satisfies its requirement. Therefore, assume that the L_i interval does grow to have this size, and hence grows larger than $|b_{j,s_n''}|$ at an α-stage s. In this case, α resets $D_s \upharpoonright u_n = D_{s_n} \upharpoonright u_n$. By Lemma 2.2.4, $D \upharpoonright u_n = D_s \upharpoonright u_n$ and hence $\Phi_e^D(x), \Phi_e^D(y)$ map into $b_{j,s}$. However, the size of the L_i interval is greater than the size of $b_{j,s}$ and the size of b_j will not change after stage s. Hence Φ_e^D cannot be one-to-one and order preserving, so $R_{\langle e,i\rangle}$ is satisfied.

The case when α acts in (C.3.2) is similar. Note that in (C.3.2), there are $x, y \in a_{j,s_n''}$ for some $j \leq n$ such that $\Phi_e^D(x)[s_n'']$ and $\Phi_e^D(y)[s_n'']$ are in different b_k blocks.

Since blocks are never combined and are dense, if Φ_e^D is onto L, it will eventually have to put unboundedly many points between x and y. Therefore, this interval will grow larger than $|b_{j,s''_n}|$ and the analysis from here is as in the previous paragraph.

□

This completes the proof. □

2.3 Remmel's construction below a low set

For any linear order L with infinitely many successor pairs, there is a computable linear order R which is not computably isomorphic to L but is isomorphic to L by a low isomorphism.

Theorem 2.3.1. *Let $(L, <_L)$ be a computable linear order with infinitely many successor pairs. There exists a non-computable low set D and computable linear order $(R, <_R)$ such that R is not computable isomorphic to L but R is D-computably isomorphic to R.*

Proof. We need to construct a computable enumerable set D and a computable function Γ, as well as a computable linear order R, such that for all partial computable functions Φ and for all $e \in \omega$, the following requirements hold:

- N : R is a computable linear order,
- M : $f : L \to R$ is an isomorphism,
- $\widehat{M}$: $f \leq_T D$,
- D_e : $D'(e) = \lim_t \Gamma(e, t)$, and
- P_e : φ_e is not an isomorphism from L onto R.

By Schoenfield's limit lemma, the D_e-requirements will ensure lowness. Note that if we satisfy these requirements we also ensure that D is not computable. Requirement M ensures that f is an isomorphism from L onto R, while the P_e requirements ensure that there is no computable isomorphism from L onto R. Hence f is not computable and by requirement $\widehat{M}$, we have that $f \leq_T D$. Therefore D is not computable.

The strategies N, M and $\widehat{M}$ will be met as in Theorem 2.1.1, while D_e will be met in the standard finite injury manner. The P_e requirement is met as in Remmel's argument with the addition of permitting which is simpler than Theorem 2.1.1 because we are constructing D.

Strategy for N:

We will construct a computable linear order R in stages with domain $\mathbb{N}$ such that $R = \cup_s R_s$. At each stage s of our construction we shall enumerate at least one number into R and specify a linear order $<_R$ on $\{0, 1, \ldots, k_s\}$, where k_s is the largest number enumerated into R at stage s.

For notational convenience, we let $r_0^s, r_1^s, \ldots, r_{k_s}^s$ be the elements $\{0, 1 \ldots, k_s\}$ in increasing $<_R$ order. Similarly, $l_0^s, l_1^s, \ldots, l_{k_s}^s$ are the elements $\{0, 1, \ldots, k_s\}$ in increasing $<_L$ order. Therefore, $R_s = \langle \{0, 1, 2, \ldots, k_s\}, <_R \rangle$ and $L_s = \langle \{0, 1 \ldots, k_s\}, <_L \rangle$.

Strategy for M

At each stage we define a finite isomorphism $f_s : \{0, 1, \ldots, k_s\} \to \{0, 1, \ldots, k_s\}$ to be the unique order isomorphism between L_s and R_s. Thus for all $i \leq k_s$ we have that $f_s(l_i^s) = r_i^s$. We define $f(x) = \lim_s f_s(x)$. The strategy to satisfy P_e will require $f_s(x) \neq f_{s+1}(x)$ for some x and s. Hence we must ensure that $\lim_s f_s(x)$ and $\lim_s f_s^{-1}(x)$ exists for all x. We will allow a strategy to act to satisfy P_e at stage s if the strategy ensures $f_s(x) = f_{s-1}(x)$ for all $x \in \{0, 1, \ldots, e\} \cup \{f^{-1}(0), f^{-1}(1), \ldots, f^{-1}(e)\}$. As we have to act to satisfy P_e at most finitely often, the limits $\lim_s f_s(x)$ and $\lim_s f_s^{-1}(x)$ will exist.

Strategy for $\widehat{M}$:

We denote by d_s, the least number to enter D at stage s. To ensure f is D-computable we allow $f_{s+1}(x) \neq f_s(x)$ if and only if $x \geq d_s$.

Strategy for D_e:

1. Start by setting $\Gamma(e,t) = 0$ at each stage t.
2. Wait for $\Phi_e^D(e)$ to converge.
3. Protect $\Phi_e^D(e)$ by restraining $D \upharpoonright \text{use}(\Phi_e^D(e))$ and set $\Gamma(e,t) = 1$ at each stage t from now on.

Outcomes for the D_e strategy:

w: Wait at step 2 forever: Then $D'(e) = 0$ and $\Gamma(e,t) = 0$ for all t.

s: Reach step 3: Then $D'(e) = 1$ and $\Gamma(e,t) = 1$ for cofinitely many t.

Strategy for P_e:

Let $c_0, c_1, c_2, \ldots$ denote the successor pairs in L. Fix an ordering of $\mathbb{N}^2$ of order type $\mathbb{N}$. For each stage s, let $C_s = \{\langle x,y\rangle \mid x,y \in L_s \text{ and } x \rightarrow_{L_s} y\}$ be the set of pairs which currently look like successor pairs. These are ordered by the fixed order on $\mathbb{N}^2$ and we let $c_0^s, c_1^s, c_2^s, \ldots, c_{k_s}^s$ be an enumeration of these pairs.

The basic strategy is as in Remmel, with the addition of permitting. To set up the permitting, we set a parameter d to be a number not currently in D which we can later use when altering f_{s+1}. We wish to find a pair $c_n^s = \langle x_n, y_n\rangle$ in L_s such that

$\varphi_e^{s+1}(x) \to_{R_s} \varphi_e^{s+1}(y)$ in R_s. We then add $s+1$ to R such that $\varphi_e^{s+1}(x) <_R s+1 <_R \varphi_e^{s+1}(y)$. If $c_n^s = \langle x_n, y_n \rangle$ turns out to be a successor pair then we have satisfied P_e. This strategy requires that f_{s+1} differs from f_s on some interval. To define this interval we let $I_n^{s+1} = \{l \mid s+1 \leq_L l \leq_L f_s^{-1}(\varphi_e(x_n))\}$ if $s+1 <_L f_s^{-1}(\varphi_e(x_n))$ and let $I_n^{s+1} = \{l \mid f_s^{-1}(\varphi_e(y_n)) \leq_L l \leq_L s+1\}$ if $s+1 >_L f_s^{-1}(\varphi_e(x_n))$. We may change f_s on the interval I_n^{s+1} if the following two requirements hold. Firstly, we have $I_n^{s+1} \cap (\{0,1,\ldots,e\} \cup \{f_s^{-1}(0), f_s^{-1}(1), \ldots, f_s^{-1}(e)\}) = \emptyset$. This ensures that $\lim_s f_s(x)$ and $\lim_s f_s^{-1}(x)$ exist. Secondly, we have $\min_{\mathbb{N}} I_n^{s+1} \geq d$. We add to D, which ensures that $f \leq_T D$ despite changing f_s to f_{s+1}. We now outline the strategy in more detail.

At each stage $s+1$:

1. Set the parameter d to be the least number not in D but larger than any restrictions placed on D so far in the construction.

2. At each subsequent stage, check that the following conditions hold:

   - φ_e is one-to-one and order preserving on $\{x \in L_s \mid \varphi_{e,s}(x)\downarrow\}$, and
   - If $x, y \in L_s$, $x \to_{L_s} y$ and $\varphi_e^{s+1}(x)\downarrow, \varphi_e^{s+1}(y)\downarrow$ then $\varphi_e^{s+1}(x)\downarrow \to_{R_s} \varphi_e^{s+1}(y)\downarrow$.

   If either of these are not true then do nothing.

3. Otherwise look for the least potential successor pair $c_n^s = \langle x_n, y_n \rangle$ such that:

   3.1. $\varphi_e^{s+1}(x_n)\downarrow, \varphi_e^{s+1}(y_n)\downarrow$,

   3.2. $\varphi_e^{s+1}(x_n), \varphi_e^{s+1}(y_n) \in R_s$,

   3.3. $\min_{\mathbb{N}}\{l \mid l \in I_n^{s+1}\} \geq d$, and

3.4. there is no $x \in \{0, 1, \ldots, e\} \cup \{f_s^{-1}(0), f_s^{-1}(1), \ldots, f_s^{-1}(e)\}$ such that $x \in I_n^{s+1}$.

If there is no such pair then wait.

4. Otherwise the strategy will act to satisfy P_e. Place $\varphi_e(x_n) <_R t <_R \varphi_e(y_n)$ where t is the least number not in R and enumerate d into D. So we have $x_n, y_n \in L_{s+1}$ such that $x_n \to_{L_{s+1}} y_n$ but $\varphi_e(x_n) \not\to_{R_{s+1}} \varphi_e(y_n)$. Assume that $s+1 <_L f_s^{-1}(\varphi_e(x_n))$ and so $I_n^{s+1} = \{l \mid s+1 \leq_L l \leq_L f_s^{-1}(\varphi_e(x_n))\}$. The other case is symmetric. Suppose $l_i^s <_L l_{i+1}^s <_L \ldots <_L l_j^s = f_s^{-1}(\varphi_e(x))$ are the elements of I_n^{s+1}. Note that in L_{s+1} we have $s+1 <_L l_i^s$ and no elements between these. We now define f_{s+1} so that $f_{s+1}(s+1) = r_i^s$, for each $i \leq k < j$ we have $f_{s+1}(l_k^s) = r_{k+1}^s$, and $f_{s+1}(l_j^s) = s+1$. To ensure $f <_T D$ our strategy requires that $f_{s+1}(l) \neq f_s(l)$ only if $D_{s+1} \upharpoonright l \neq D_s \upharpoonright l$. As we have enumerated d into D, the unique isomorphism $f_{s+1} : L_{s+1} \to R_{s+1}$ requires $f_{s+1}(l) \neq f_s(l)$ if and only if $l \in I_n^{s+1}$ and $\min_{\mathbb{N}}\{l \mid l \in I_n^{s+1}\} \geq d$ we have that $f <_T D$.

Outcomes for the P_e strategy:

s: Stop at step 2 cofinitely often: Then φ_e is not one-to-one, order preserving or the image of a successor pair in L is not a successor pair in R.

w: Wait at step 3 cofinitely often: Then φ_e only converges on finitely many successor pairs.

The current outcome of a P_e strategy is s if the strategy has stopped at step 2 and w if the strategy is waiting at step 3. We let $\Lambda = \{s <_\Lambda w\}$ be the set of outcomes.

Tree of Strategies:

Let $T = \Lambda^{<\omega}$ be the tree of strategies. We order the requirements as follows: $D_0 > P_0 > D_1 > P_1 > \ldots$ and assign to all strategies $\alpha \in T$ of length r the rth requirement in the list.

Construction:

Each stage s consists of substages $t \leq s$. At sub-stage t, the strategy α of length t eligible to act is chosen such that for all $\beta \subset \alpha$, we have that $\beta^\frown\langle o\rangle \subseteq \alpha$ if and only if β has current outcome o. Strategy α will then proceed according to its description from where it left off the last time it was eligible to act. We define the current true path TP_s at stage s to be the longest strategy eligible to act at stage s and we initialise all strategies to the right of TP_s.

Furthermore, if at stage s nothing has been enumerated into R_s, then we enumerate the least number not in R_s, say z, and place it in the corresponding position in R as it appears in L. So $f_{s+1}(z) = z$.

Verification:

We define the true path TP to be the limit of the current true path. The limit will exist since the true outcome of any strategy is current cofinitely often.

Lemma 2.3.1. *Every strategy $\alpha \subseteq TP$ ensures the satisfaction of its requirement. In fact, if $\alpha \subset TP$ then $\alpha * s$ or $\alpha * w$ is on TP_s for cofinitely many s. Furthermore, if α is a P_e strategy and $\alpha \subset TP_s$ for all $s \geq t$, then $f_s(x)$ and $f_s^{-1}(x)$ are constant for all $s \geq t$ and $x \leq e$.*

Proof. The strategy α will first be eligible to act at some stage, s_0 say. At some stage $s_1 \geq s_0$ all strategies $\beta \subset \alpha$ will have permanently achieved their final outcome and from then on α will be eligible to act at each stage. Hence no β incomparable with α is eligible to act after stage s_1.

Case 1: α is a D_e-strategy. If $\Phi_e^D(e)$ converges at some stage, this strategy switches to outcome $\alpha * s$, which protects this computation by placing a restraint on D. Any d selected by a P_e strategy after this stage will be greater than this restraint and so will not affect the computation $\Phi_e^D(e)$.

Case 2: α is a P_e-strategy. After stage s_1, the outcome of $\beta \subset \alpha$ will not change. Hence the restraint placed on D by these strategies will not change and so the parameter d will not change after stage s_1. If α stops at step 2 then φ_e is not one-to-one, order preserving or the image of a successor pair in L is not a successor pair in R. If α waits at step 3 forever, then φ_e only converges on finitely many successor pairs.

Suppose to the contrary that φ_e converges on infinitely many successor pairs $c_0, c_1, c_2, \ldots$. Since P_e did not permanently stop at some stage stage, φ_e is one-to-one, order preserving and maps successor pairs in L onto successor pairs in R. Let $m = \max\{e, d, f_s^{-1}(0), f_s^{-1}(1), \ldots, f_s^{-1}(e)\}$ and let $a_0 <_L a_1 <_L \ldots <_L a_m$ be the l-ordering of the elements $\{0, 1, \ldots, m\}$. We can partition L into the intervals $(-\infty, a_0), (a_0, a_1), \ldots (a_{m-1}, a_m), (a_m, \infty)$. By our assumption on φ_e we know there are infinitely many successor pairs in one of these intervals. Hence at some stage s_k the function φ_e will converge on a pair $c_k = \langle x_k, y_k \rangle \in L_{s_k}$ such that $f_{s_k}^{-1}(\varphi_e(x_k))$ and $f_{s_k}^{-1}(\varphi_e(y_k))$ lies in an infinite interval. It follows that $\min_{\mathbb{N}}\{l \mid l \in I_k^{s+1}\} > m$ at infinitely many stages $s+1 > s_k$. That is, $\{0, 1, \ldots, e, f_s^{-1}(0), f_s^{-1}(1), \ldots, f_s^{-1}(e)\} \notin I_k^{s+1}$ and $\min\{l \mid l \in I_k^{s+1}\} > d$ infinitely often. So there is a stage $s+1 > s_1$ large enough such that $x_k \rightarrow_{L_s} y_k$ $\varphi_{e,s}(x_k)\downarrow$, $\varphi_k^s(y_k)\downarrow$, $\varphi_{e,s}(x_k), \varphi_{e,s}(y_k) \in R_s$ and $I_k^{s+1} > m$. In this

case c_k satisfies the conditions in step 3 and so the strategy would act to satisfy P_e. Hence if α waits at step 3 forever, φ_e cannot converge on infinitely many successor pairs.

Note that for any x only P_e strategies with $e < x$ can change $f_s(x)$ and $f_s^{-1}(x)$. So if α is a P_e strategy and $\alpha \subset TP_s$ for all $s \geq t$, then $f_s(x)$ and $f_s^{-1}(x)$ are constant for all $s \geq t$ and $x \leq e$.

□

As our construction incorporated the strategies for N, M, and $\widehat{M}$, these requirements are also met.

□

Chapter 3

Tukey types of partial orders

Tukey introduced the Tukey ordering to develop the notion of Moore-Smith convergence in topology [**?**]. After its initial success in helping develop general topology, Tukey reducibility was studied as a means of classifying algebraic structures related to partially ordered sets in, for example, Day [**?**], Isbell [**?**] and Todorcevic [**?**]. Its classification is coarser than that of isomorphism and so can be useful in settings where isomorphism is too fine to give a useful classification.

The isomorphism relation on countable posets preserves all the structural information but there are $2^{\aleph_0}$ many isomorphism classes of countable posets and the isomorphism relation is Σ^1_1-complete. On the other hand, the Tukey relation on a countable poset preserves some but not all of the structural information, and has only countably many equivalence classes. Cofinal equivalence is closely related to Tukey equivalence of posets and although it is finer than Tukey equivalence it also gives countably many equivalence classes. Both Tukey and cofinal equivalence classes are described in terms of notions of reducibility and can be described arithmetically.

In addition to classifying posets, Tukey equivalence and cofinal equivalence have been studied in the more general setting of oriented systems and in the more restricted setting of directed posets. Day showed in [**?**] that for directed sets cofinal similarity coincides with Tukey equivalence. There are various classically equivalent definitions for the notion of Tukey reducibility and we will examine how difficult it is to prove their equivalence. We begin be discussing Tukey reducibility in the context of oriented systems.

3.1 Oriented Systems

Day studied the Tukey ordering in the more general setting of oriented systems. We begin with the basic definitions associated with oriented systems.

Definition 3.1.1. An *oriented system* is a pair $(S, <)$ such that

- S is a non-empty set;
- $<$ is a transitive relation on S; and
- each element in S has a successor in S, that is for each $s \in S$ there is an $s' \in S$ such that $s < s'$.

$(S, <)$ is *directed* if it is non-empty, transitive and each pair of elements in S have a common successor in S.

Note that, despite using the notation $<$, an oriented system $(S, <)$ is not necessarily irreflexive. Also the relation $<$ may have cycles.

Definition 3.1.2. A subset S' of S is *cofinal* in $(S, <)$ if each element of S has a successor in S'. That is, for each $s \in S$, there exists $s' \in S'$ such that $s < s'$.

Definition 3.1.3. Let $(S, <_S)$ and $(T, <_T)$ be oriented systems. T is *Tukey reducible* to S, which we will write as $T <_{ty} S$, if there exist functions $f : S \to T$ and $g : T \to S$ such that for each $t \in T$, $g(t) <_S s$ implies $t <_T f(s)$. If two functions are related in this way, we refer to f as an *upper support* for g and g as a *lower support* for f.

Definition 3.1.4. If S and T are non-empty sets then $S + T$ denotes the disjoint union of S and T. Formally, $S + T = (S \times \{0\}) \cup (T \times \{1\})$. However when working with $S + T$ we will treat S and T as though they are disjoint and regard $S + T$ as $S \cup T$.

Definition 3.1.5. If $(S, <_S)$ and $(T, <_T)$ are oriented systems and f is a function from S into T, then $<_f$ is a binary relation on $S + T$ such that $s <_f t$ if and only if $f(s) <_T t$.

Definition 3.1.6. Let S be any set and let $<_1$ and $<_2$ be two binary relations on S, then $<_1$ *includes* $<_2$ if $s <_2 s'$ implies $s <_1 s'$.

Lemma 3.1.1. *If $<_1, <_2, \ldots, <_n$ are binary relations on S such that for each $s \in S$ there exists s' and i such that $s <_i s'$, then there classically exists a unique minimal orientation $<$ on S which includes all $<_i$; explicitly, $s_0 < s$ if and only if there exists $s_1, s_2, \ldots, s_k = s$ and integers $i_0, i_1, \ldots, i_{k-1}$ such that for each $0 \leq j < k$ the relation $s_j <_{i_j} s_{j+1}$ holds. Furthermore, the existence of $<$ is equivalent to ACA_0 by a proof similar to Theorem 3.1.3.*

In Theorem 2.1 in [**?**], Day proved the following theorem about the existence of upper supports of functions. If $(S, <_S)$ and $(T, <_T)$ are disjoint oriented systems, a function $g : T \to S$ has an upper support if and only if there exists an orientation $<$ of $S + T$ such that

1. $<$ includes $<_S, <_T, <_g$,

2. T is cofinal in $(S+T, <)$,

3. $<$ agrees with $<_T$ in T.

We will show that one direction of this equivalence is provable in RCA_0 while the other direction is equivalent to WKL_0.

Theorem 3.1.1. (RCA_0)

If $(S, <_S)$ and $(T, <_T)$ are disjoint oriented systems, g is a function from T into S and there exists an orientation $<$ of $S+T$ such that

1. *$<$ includes $<_S, <_T, <_g$;*

2. *T is cofinal in $(S+T, <)$; and*

3. *$<$ agrees with $<_T$ in T.*

Then g has an upper support.

Proof. Suppose an orientation $<$ of $S+T$ exists with properties 1 to 3. By the fact that T is cofinal in $(S+T, <)$, we may define $f : S \to T$ such that $s < f(s)$ for each $s \in S$. Then, for any $t \in T$, if $g(t) <_S s$, then $t <_g s$ and so $t < s < f(s)$. By transitivity of $<$ and property 3 we have $t <_T f(s)$. Hence f is an upper supporting function for g.

□

Day's proves the converse direction by using the least orientation on $S+T$ which includes $<_S, <_T, <_g, <_f$. We will later show that the existence of this least orientation implies ACA_0.

Theorem 3.1.2. *The following are equivalent over* RCA_0*:*

1. WKL_0.
2. *Let* $(S, <_S)$ *and* $(T, <_T)$ *be disjoint oriented systems and let* $g : T \to S$ *be a function which has an upper support* $f : S \to T$*. Then there exists an orientation* $<$ *of* $S + T$ *such that*
   - 2.1. $<$ *includes* $<_S, <_T, <_g$*;*
   - 2.2. T *is cofinal in* $(S + T, <)$*; and*
   - 2.3. $<$ *agrees with* $<_T$ *in* T*.*

Proof. We prove the two directions of this theorem separately.

WKL_0 suffices to prove such an orientation exists

Let $f : S \to T$ be an upper supporting function to $g : T \to S$. We will construct $(S + T, <)$ using WKL_0 such that

(i) $(S + T, <)$ includes $<_S, <_T, <_g, <_f$; and

(ii) $<$ agrees with $<_T$ on T.

An orientation satisfying these criteria will satisfy 2.1. to 2.3.. It is clear that it will satisfy 2.1. and 2.3.. We show that it also satisfies 2.2., that is T is cofinal in $(S + T, <)$, since $(S + T, <)$ includes $<_f$. First recall that $s <_f t$ if and only if $f(s) <_T t$. Since every element of T has a successor, for every $s \in S$ there exists $t \in T$ such that $f(s) <_T t$. Hence T is cofinal in $(S + T, <)$. Therefore it suffices to construct $(S + T, <)$ which satisfies (i) and (ii).

Suppose $S = \{2n \mid n \in \mathbb{N}\}$ and $T = \{2n+1 \mid n \in \mathbb{N}\}$. We define a tree $\mathfrak{T} \subseteq 2^{<\mathbb{N}}$ such that any path through $\mathfrak{T}$ codes an orientation of $S + T$ which satisfies (i) and (ii). We view each sequence $\sigma \in 2^{<\mathbb{N}}$ as a sequence of pairs of elements from $\mathbb{N}$ and we think of σ as giving a finite approximation, $<_\sigma$, to an orientation on $S + T$. That is, we interpret $\sigma(\langle n, m\rangle) = 1$ as specifying $n <_\sigma m$ and $\sigma(\langle n, m\rangle) = 0$ as specifying $n \not<_\sigma m$. We put $\sigma \in \mathfrak{T}$ if and only if, for all $l, n, m < |\sigma|$:

1. $m <_S n$, $m <_T n$, $m <_g n$ or $m <_f n$ and $\langle m, n\rangle \in \mathrm{dom}(\sigma)$, imply $\sigma(\langle m, n\rangle) = 1$;

2. $\sigma(\langle l, m\rangle) = 1$, $\sigma(\langle m, n\rangle) = 1$ and $\langle l, n\rangle \in \mathrm{dom}(\sigma)$ imply $(\langle l, n\rangle) = 1$; and

3. $m, n \in T$ and $m \not<_T n$ imply $\sigma(\langle m, n\rangle) = 0$.

It is clear that $\mathfrak{T}$ is a tree and we will now verify that it is infinite. Fix $N \in \mathbb{N}$ and for each pair $\langle n, m\rangle < N$ define $\sigma(\langle n, m\rangle) = 1$ if and only if there is a sequence $x_0, \ldots, x_k \in S + T$ such that

- each $x_i < N$;
- for each $i < k$ there exists $j \in \{S, T, f, g\}$ such that $x_i <_j x_{i+1}$; and
- $x_0 = n$ and $x_k = m$.

Clearly σ satisfies 1 and 2. Since f is an upper support of g, we show that σ also satisfies 3. Suppose $\sigma(\langle t, t'\rangle) = 1$ for $t, t' \in T$. Hence there exists a sequence $x_0, \ldots, x_k \in S + T$ such that $x_i <_j x_{i+1}$ for some $j \in \{S, T, f, g\}$ and $x_0 = t$ and $x_k = t'$. If each x_i in this sequence is an element of T, then we have $t <_T t'$. In addition we claim that if one of the x_i's in the sequence is an element of S, then we also have $t <_T t'$. Suppose that there exists an i such that $x_i \in T$ and $x_{i+1} \in S$. Then we have $x_i <_g x_{i+1}$, that is $g(x_i) <_S x_{i+1}$. Since $t' \in T$, for some $i + 1 \leq j < k$ we

have $x_{i+1} \leq_S x_j$ where $x_{j+1} \in T$. So we have $f(x_j) <_T x_{j+1}$. Hence $x_i <_g x_{i+1} \leq_S x_j$ implies $g(x_i) <_S x_{i+1} \leq_S x_j$. So $g(x_i) <_S x_j$. Since f is an upper support of g, we have that $g(x_i) <_S x_j$ implies $x_i <_T f(x_j)$. However $x_j <_f x_{j+1}$ implies $f(x_j) <_T x_{j+1}$ and so $x_i <_T x_{j+1}$. Hence if $\sigma(\langle t, t' \rangle) = 1$, there exists a sequence connecting them which is comprised solely of elements of T and so $t <_T t'$. So σ satisfies 3 also. Hence $\sigma \in \mathfrak{T}$ and so $\mathfrak{T}$ is infinite.

By WKL_0, $\mathfrak{T}$ has a path. If p is a path, define $<_p$ on $S + T$ by $n <_p m$ if and only if $p(\langle n, m \rangle) = 1$. Criterion 2 for a string to enter $\mathfrak{T}$ implies that $<_p$ is transitive. By criterion 1 we have that $<_p$ includes $<_S$ and $<_T$ so every element of $S + T$ has a successor under $<_p$ in $S+T$. Hence $(S+T, <_p)$ is an orientation including $<_S, <_T, <_g$ and $<_f$, and by criterion 3 we have that $<_p$ agrees with $<_T$ on T. This orientation satisfies 2.1. to 2.3..

The existence of such an orientation implies WKL_0

To show WKL_0, it suffices to consider an arbitrary pair of one-to-one functions $h, l : \mathbb{N} \to \mathbb{N}$ with disjoint ranges and show that there is a set X such that for all m, $h(m) \in X$ and $l(m) \notin X$. We will build two orientations $(S, <_S)$ and $(T, <_T)$ and a function $g : T \to S$ such that g has an upper supporting function $f : S \to T$ and given any orientation, $<$, on $S + T$ which satisfies 2.1. to 2.3., there exists X such that for all m, $h(m) \in X$ and $l(m) \notin X$.

Constructing the orientations $(S, <_S)$ and $(T, <_T)$

The orientation $(T, <_T)$ will contain two chains, each of which contains a maximal element. T will consist of $\{t_i \mid i \in \mathbb{N}\}$ and two maximal elements $\{t', t''\}$. More

specifically, we will have $t_0 <_T t_2 <_T t_4 <_T \ldots <_T t_{2k} <_T t_{2(k+1)} <_T \ldots <_T t'$ and $t_1 <_T t_3 <_T t_5 <_T \ldots <_T t_{2k+1} <_T t_{2k+3} \ldots <_T t''$. We also have $t' <_T t'$ and $t'' <_T t''$. Hence $(T, <_T)$ is transitive and each element in T has a successor.

The elements of $(S, <_S)$ will satisfy $s <_S s$, so each element will have a successor. Additionally, $(S, <_S)$ will form an antichain except that we will make $s_{2l(k)} <_S s_{2k+1}$ to code facts about the range of the function l. The even and odd indexed elements of S will not necessarily be enumerated into S in the order of their indices. At each stage k, will will enumerate one more element into S with an odd index, specifically s_{2k+1}. On the other hand, we will enumerate as many of the even index elements of S as necessary. This will depend on the values of $h(k)$ and $l(k)$.

The idea behind the construction

The idea behind the construction is that if n is in the range of h, then we will have $t_0 < s_{2n}$ in any orientation $(S + T, <)$ which satisfies 2.1. to 2.3.. In addition, if n is in the range of l, then we will ensure $t_0 \not< s_{2n}$ in any orientation $(S + T, <)$ which satisfies 2.1. to 2.3.. In the former case, when n is in range of h, in order to ensure $t_0 < s_{2n}$ we must, at some stage in our construction, set $g(t_i) = s_{2n}$ for some $t_i >_T t_0$. Note that the index i must be even. In the latter case, when n is in the range of l, we want $t_0 \not< s_{2n}$. To ensure this, we will force s_{2n} to have a successor t in T where $t = t_i$ and i is odd or $t = t''$. Therefore, as any orientation $(S + T, <)$ which satisfies 2.1. to 2.3. is transitive and must agree with $<_T$ on T, it will not be possible for $s_{2n} > t_0$.

The construction

At stage 0:

- Enumerate $t_0, t_1, t^{'}$ and $t^{''}$ into T.
- Add the following relations to $(T, <_T)$: $t_0 <_T t^{'}$, $t_1 <_T t^{''}$, $t^{'} <_T t^{'}$ and $t^{''} <_T t^{''}$.
- Enumerate s_1 and $\{s_{2n} \mid n < \max\{0, h(0), l(0)\}\}$ into S.
- Let $s <_S s$ for all s enumerated at this stage.
- Let $g(t_0) = s_{2h(0)}$ and $g(t_1) = s_1$.
- Let $s_{2l(0)} <_S s_1$.

At each stage $k > 0$:

- Enumerate t_{2k}, t_{2k+1} into T.
- Let $t_{2n-2} <_T t_{2k} <_T t^{'}$ and $t_{2n-1} <_T t_{2k+1} <_T t^{''}$ for all $1 \leq n \leq k$.
- Enumerate s_{2k+1} and, as necessary, $\{s_{2n} \mid n < \max\{k, h(k), l(k)\}\}$ into S.
- Let $s <_S s$ for all s enumerated at this stage.
- Let $g(t_{2k}) = s_{2h(k)}$ and $g(t_{2k+1}) = s_{2k+1}$.
- Let $s_{2l(k)} <_S s_{2k+1}$.

Note that the ordering between any pair of elements in T is determined at the first stage when both are declared in T. That is, if t_i, t_j have been enumerated into T by stage k and we do not declare $t_i <_T t_j$ at stage k, then $t_i \not<_T t_j$. Similarly, the ordering between any pair of elements in S is determined at the first stage when both are declared in S. That is, if s_i, s_j have been enumerated into S by stage k and we do not declare $s_i <_S s_j$ at stage k, then $s_i \not<_S s_j$.

The upper support f exists

We may define an upper supporting function $f : S \to T$ as follows: $f(s_{2n}) = t^{'}$ and $f(s_{2n+1}) = t^{''}$ for each $n \in \mathbb{N}$.

If n is in the range of h, then $g(t_{2k}) = s_{2n}$ for some k and s_{2n} has no successor except itself. Therefore $f(s_{2n}) = t^{'} >_T t_{2k}$ is an upper support for g in this case. If n is not in the range of h, then s_{2n} is not in the range of g and s_{2n} has no predecessors in S, so $f(s_{2n}) = t^{'}$ has no restrictions. Since $g(t_{2n+1}) = s_{2n+1}$ and s_{2n+1} has no successor except itself, $f(s_{2n+1}) = t^{''} >_T t_{2n+1}$ is an upper support for g.

The separating set

Suppose $(S+T, <)$ satisfies 2.1. to 2.3. and let $X = \{n \mid s_{2n} > t_0\}$. We claim that if n is in the range of h, then n is an element of X. Suppose $n = h(k)$ for some k. In this case, at stage k of the construction, we set $g(t_{2k}) = s_{2n}$. So we have, $t_0 <_T t_{2k} <_g s_{2n}$. As $(S + T, <)$ includes $<_T$ and $<_g$, we also have $t_0 < s_{2n}$. Hence $n \in X$.

We also claim that if n is in the range of l, then n is not an element of X. Suppose $n = l(k)$ for some k. In this case, at stage k of the construction, we set $g(t_{2k+1}) = s_{2k+1}$ and $s_{2n} <_S s_{2k+1}$. As T is cofinal in $(S + T, <)$, there is some $t \in T$ such that $s_{2k+1} < t$. As $t_{2k+1} <_g s_{2k+1}$ and $(S + T, <)$ is transitive, includes $<_g$ and must agree with $<_T$ on T, we have that this successor t of s_{2k+1} must also be a successor of t_{2k+1} in T. Hence $t = t_i$ where i is odd or $t = t^{''}$. Since $s_{2n} <_S s_{2k+1}$ we also have $s_{2n} < t$. Again since $(S + T, <)$ is transitive and must agree with $<_T$ on T we cannot have $t_0 < s_{2n}$ as this would imply that $t_0 <_T t$. Therefore n is not in X. We conclude that X is a separating set for the ranges of h and l, as desired.

□

Theorem 3.1.3. *The following are equivalent over* RCA_0*:*

1. ACA_0.
2. *If* $(S, <_S)$ *and* $(T, <_T)$ *are disjoint oriented systems, and* $g : T \to S$ *has an upper support* $f : S \to T$*, then there exists a least orientation on* $S + T$ *which includes* $<_S, <_T, <_g, <_f$.

Proof. First, we show that 1 suffices to show 2 since the least orientation of $S + T$ which includes $<_S, <_T, <_g, <_f$ is arithmetically definable. That is, $x_0 < x_1$ if and only if there exist $u_0, u_1, \ldots, u_k \in S + T$ such that $x_0 = u_0$, $x_1 = u_k$ and for all $i < k$ there exists $j \in \{S, T, g, f\}$ such that $u_i <_j u_{i+1}$.

Secondly, we show that 2 implies ACA_0 by showing that the range of each one-to-one function exists. Let $h : \mathbb{N} \to \mathbb{N}$ be one-to-one. We construct computable orientations $(S, <_S)$ and $(T, <_T)$ and functions $g : T \to S$ and $f : S \to T$ where f is an upper support of g, such that the least orientation $<$ of $S+T$ including $<_S, <_T, <_g$ and $<_f$ codes the range of h.

Let $S = \{s_i^n \mid i \text{ is even and } n \in \mathbb{N}\} \cup \{s_i \mid i \text{ is odd}\}$, let $s_i^0 <_S s_i^1 <_S s_i^2, <_S \ldots$ where i is even, and let $s_i <_S s_i$ where i is odd. Let $T = \{t_0, t_1, t_2, \ldots\}$ and let $t_i <_T t_{i+1}$ if i is even and $t_i <_T t_i$ for all i. So $(S, <_S)$ and $(T, <_T)$ are orientations in RCA_0.

Let $g(t_i) = s_{i+1}$ if i is even and let $g(t_i) = s_i$ if i is odd. Let $f(s_i) = t_i$ if i is odd. When $i = 2e$ is even, define $f(s_{2e}^n) = t_{2e}$ if there exists $m < n$ such that $h(m) = e$, and define $f(s_{2e}^n) = t_{2e+1}$ otherwise. It is straightforward to check that f is an upper support for g.

Let $<$ be the least orientation on $S + T$ which includes $<_S, <_T, <_g$ and $<_f$. We claim that $s_{2e}^0 < t_{2e}$ if and only if e is in the range of h. Suppose e is in the range

of h and fix the unique m such that $h(m) = e$. Then $f(s_{2e}^{m+1}) = t_{2e}$. So we have $s_{2e}^0 <_S s_{2e}^{m+1} <_f t_{2e}$. Hence $s_{2e}^0 < t_{2e}$. Suppose e is not in the range of h. Then $h(m) \neq e$ for all m and we have $f(s_{2e}^m) = t_{2e+1}$ for all m. Therefore there is no $x \in S + T$ and $r \in \{S, T, f, g\}$ such that $x <_r t_{2e}$. It follows that $s_{2e}^0 \not< t_{2e}$.

□

3.2 Tukey reducibility and partial orders

We will now introduce Tukey reducibility for general partial orders.

Definition 3.2.1. We say that $f : D \to E$ is a *convergent map* from D into E if for each $e \in E$ there is a $d \in D$ such that $f(c) \geq_E e$ for all $c \geq_D d$.

Definition 3.2.2. We say a partial ordering $(E, \leq_E)$ is *Tukey reducible* to a partial ordering $(D, \leq_D)$, written as $E \leq_{ty} D$, if and only if there is a convergent map from D into E.

Note that the convergent map in this definition is equivalent to the upper support in the definition of Tukey reducibility given by Day in [**?**]. We will now look at various classically equivalent definitions for the notion of Tukey reducibility and examine how difficult it is to prove their equivalence. In what follows let $(D, \leq_D)$ and $(E, \leq_E)$ be partial orderings.

Definition 3.2.3. (Classical) We say a function $f : D \to E$ is *cofinal* if the image of each cofinal subset of D is cofinal in E.

Definition 3.2.4. (RCA_0) We say a function $f : D \to E$ is *cofinal* if for every cofinal

$X \subseteq D$, we have that f maps X cofinally into E. That is,

$$\forall e \in E \; \exists x \in X (e \leq_E f(x)).$$

In the above definition of a cofinal map we don't assume that the image of the cofinal set exists.

Theorem 3.2.1. *The following are equivalent over* RCA_0.

1. *There exists a convergent map from D into E.*

2. *There exists a cofinal map from D into E.*

Proof. Suppose $f : D \to E$ is a convergent map and suppose $X \subseteq D$ is cofinal in D. We claim that f maps X cofinally into E. Let $e \in E$. We know that there exists $d_e \in D$ such that $f(c) \geq_E e$ for all $c \geq_D d_e$. Since X is cofinal in D, there exists $x \in X$ such that $x \geq_D d_e$. Since f is convergent, $f(x) \geq_E e$. Hence f maps X cofinally into E. In fact, f is a cofinal map.

Suppose $f : D \to E$ is a cofinal map. We will show that f is also a convergent map from D into E. Suppose for a contradiction that f is not a convergent map from D into E. That is, there is some $e \in E$ such that for every $d' \in D$ there exists $d \geq_D d'$ with $f(d) \not\geq_E e$. Let $X_e = \{d \in D \mid f(d) \not\geq_E e\}$. By assumption, X_e is cofinal in D. So f maps X_e cofinally into E. Hence there exists $d \in X_e$ such that $f(d) \geq_E e$, contradicting the definition of X_e. So f is a convergent map from D into E. □

Definition 3.2.5. (Classical definition) A subset $X \subseteq D$ is called *unbounded* if there is no single $d \in D$ which simultaneously bounds every member of X. That is, for

each $d \in D$, there is some $x \in X$ such that $d \not\geq_D x$. A map $g : E \to D$ is called a *Tukey map* or an *unbounded map* if the g-image of each unbounded subset of E is an unbounded subset of D.

In RCA_0 the image of a set may not necessarily exist so we have the following definition.

Definition 3.2.6. (RCA_0) A map $g : E \to D$ is called a *Tukey map* or an *unbounded map* if for every $X \subseteq E$ which is unbounded we have

$$\forall d \in D \; \exists x \in X (d \not\geq_D g(x)).$$

If $(E, \leq_E)$ is an antichain, then every subset of size at least two is unbounded.

Schmidt showed that there exists a convergent map from D into E if and only if there is an unbounded map from E into D. We show here that this equivalence requires ACA_0.

Theorem 3.2.2. *There exist computable partial orders $(D, \leq_D)$ and $(E, \leq_E)$ with a convergent map $f : D \to E$ such that if $g : E \to D$ is an unbounded map, then $0' \leq_T g$.*

Proof. We construct $E = \{e_i \mid i \in \mathbb{N}\}$ as an infinite antichain with $e \leq_E e$ for all $e \in E$. Our construction will require two elements to code if $0 \in K$, four elements to code if $1 \in K$, eight elements to code if $2 \in K$ and in general 2^{t+1} elements to code if $t \in K$. We will use $\{e_0, e_1\}$ to code if $0 \in K$ and $\{e_2, e_3, e_4, e_5\}$ to code if $1 \in K$ and so on. For notational convenience we will let n_t denote the index of the first element of E used to code if $t \in K$ and let m_t denote the index of the last element of E used to code if $t \in K$. Hence $n_0 = 0, m_0 = 1, n_1 = 2, m_1 = 5$ and

in general $n_t = \sum_{k=1}^{t} 2^k = 2^{t+1} - 2$ for $t \geq 1$ and $m_t = (\sum_{k=1}^{t+1} 2^k) - 1$. Note that $\{e_{n_t}, e_{n_t+1}, \ldots, e_{m_t}\}$ consists of 2^{t+1} elements.

We construct $D = \{d_i^j \mid i \in \mathbb{N} \text{ and } j \geq t \text{ where } n_t \leq i < n_{t+1}\}$ to consist of infinitely many unbounded chains as follows:

$$d_0^0 <_D d_0^1 <_D d_0^2 <_D d_0^3 <_D d_0^4 <_D \ldots d_0^j <_D d_0^{j+1} <_D \ldots$$

$$d_1^0 <_D d_1^1 <_D d_1^2 <_D d_1^3 <_D d_1^4 <_D \ldots d_1^j <_D d_1^{j+1} <_D \ldots$$

$$\ldots$$

$$d_{n_t}^t <_D d_{n_t}^{t+1} <_D d_{n_t}^{t+2} <_D \ldots d_{n_t}^j <_D d_{n_t}^{j+1} <_D \ldots$$

$$d_{n_t+1}^t <_D d_{n_t+1}^{t+1} <_D d_{n_t+1}^{t+2} <_D \ldots d_{n_t+1}^j <_D d_{n_t+1}^{j+1} <_D \ldots$$

$$\ldots$$

$$d_{m_t}^t <_D d_{m_t}^{t+1} <_D d_{m_t}^{t+2} <_D \ldots d_{m_t}^j <_D d_{m_t}^{j+1} <_D \ldots$$

$$\ldots$$

That is, $d_i^j <_D d_i^{j+1}$ for all i and $j \geq t$ where $n_t \leq i < n_{t+1}$. The subscript denotes the chain that the element belongs to and the superscript denotes the stage at which the element is enumerated. We will add more relations to $<_D$ during the construction. We define a convergent function $f : D \to E$ as $f(d_i^j) = e_i$ for all i and j.

The idea behind the construction is that any unbounded map will need to map the 2^{t+1} unbounded elements $\{e_{n_t}, e_{n_t+1}, \ldots e_{m_t}\}$ to 2^{t+1} unbounded elements of D. If we see $\varphi_t(t)$ converge at stage s then we create a directed set containing all the elements currently in D with index greater than n_t. Therefore at the end of that

stage the largest possible unbounded set will contain at most one element from each chain with index less than n_t, there are $2^{t+1} - 2$ of these, and 1 element from the newly constructed directed set. Hence the largest possible unbounded set contains at most $2^{t+1} - 1$ many elements. So if $g : E \to D$ is an unbounded map, at least one of $g(e_{n_t}), g(e_{n_t+1}), \ldots, g(e_{m_t})$ must be enumerated after stage s. By taking s_t to be the largest superscript in the set $\{g(e_{n_t}), g(e_{n_t+1}), \ldots, g(e_{m_t})\}$ we have $t \in K$ if and only if $t \in K_{s_t}$.

The construction:

At each stage s:

- Enumerate $e_{n_s}, e_{n_s+1}, \ldots, e_{m_s}$ into E.
- Let $e_i \leq_E e_i$ for e_i enumerated at this stage.
- Enumerate $d^s_0, d^s_1, \ldots d^s_{n_s}, \ldots d^s_{m_s}$.
- Let $d^s_i \leq_D d^s_i$ for each d^s_i enumerated at this stage.
- Let $d^j_i <_D d^s_i$ for all $j < s$ where d^j_i exists.
- Let $f(d^j_i) = e_i$ for all $d^s_0, d^s_1, \ldots d^s_{n_s}, \ldots d^s_{m_s}$.
- For each $t < s$, check if $t \in K_s$ and $t \notin K_{s-1}$. If this is the case then let $d^j_i <_D d^s_{n_t}$ for all d^j_i with $n_t < i \leq m_s$ and $j \leq s$.

Note that if d^j_i is added to D after stage s, then $j > s$.

The function $f : D \to E$ is a convergent map

Notice that if $d_i^j <_D d_{i'}^{j'}$ then either $i = i'$ and $j < j'$ or $i' < i$. For any fixed i, let s be such that $n_s \leq i \leq m_s$. There are at most s many stages t at which we set $d_{i'}^t > d_i^t$ for some $i' < i$. Therefore there is a stage u such that for all i', j' we have $d_i^u <_D d_{i'}^{j'}$ if and only if $i = i'$ and $u < j'$. Since $f(d_i^j) = e_i$ for all i, the function f is a convergent map from D into E.

Any unbounded function from E into D codes the halting set

Suppose $g : E \to D$ is an unbounded function. Let

$$s_t = \max\{j \mid d_i^j = g(e_k) \text{ for some } s_{n_t} \leq k \leq s_{m_t}\}.$$

We claim that $t \in K$ if and only if $t \in K_{s_t}$. If $t \notin K$, then $t \notin K_{s_t}$ also. Suppose that $t \in K$. Then, at some stage s, we had $t \in K_s$ and $t \notin K_{s-1}$. Hence at stage s of the construction we let $d_i^j <_D d_{n_t}^s$ for all d_i^j with $n_t < i \leq m_s$ and $j \leq s$. Hence $D_t = \{d_i^j \mid n_t \leq i \leq m_s \text{ and } j \leq s\}$ is a bounded set. Since g is an unbounded function at most one of the elements $\{g(e_{n_t}), g(e_{n_t+1}) \ldots g(e_{m_t})\}$ can be in D_t and there are at most $n_t = 2^{t+1} - 2$ distinct directed sets with subscript less than n_t. Hence at least one of the 2^{t+1} elements $\{g(e_{n_t}), g(e_{n_t+1}) \ldots g(e_{m_t})\}$ must have been enumerated after stage s and hence must have superscript $s' > s$. Hence s_t as defined above is greater than s and so $t \in K_{s_t}$ also.

□

3.3 Tukey types of computable partial orders

Definition 3.3.1. If $P \leq_{ty} Q$ and $Q \leq_{ty} P$ we say that P is *Tukey equivalent* to Q, written as $P \equiv_{ty} Q$.

The relation $\equiv_{ty}$ is an equivalence relation and the equivalence classes are called *Tukey types*. Tukey types are themselves partially ordered by the Tukey reduction $\leq_{ty}$.

We now consider the complexity of computing Tukey types of computable posets. In what follows, we let $1 = \{e\}$ denote the directed set with just one element and we let ω denote $\mathbb{N}$ under the usual ordering.

3.3.1 Tukey types of directed sets

Tukey showed in [**?**] that the Tukey types of countable directed partial orders are 1 and ω. First we consider directed sets with a maximal element.

Definition 3.3.2. Let $(P, \leq_P)$ be a poset. We say that $a \in P$ is a *maximal element* if for all $p \in P$ such that $p \geq_P a$ we have $p = a$.

Definition 3.3.3. Let $(P, \leq_P)$ be a poset. We say that $a \in P$ is a *greatest element* if for all $p \in P$ we have $p \leq_P a$.

Note that in directed posets a maximal element is also a greatest element.

Lemma 3.3.1. *The Tukey type of a directed poset with a greatest element is* 1.

Proof. We note that for any directed poset D we have $1 \leq_{ty} D$. Let $1 = \{e\}$ and let $f : D \to 1$ be defined by $f(d) = e$ for all $d \in D$. Then $f : D \to 1$ is a cofinal map.

Conversely, let d_m denote the greatest element of D and let $g : 1 \to D$ be defined by $g(e) = d_m$. As $g : 1 \to D$ is also a cofinal map, we have $D \leq_{ty} 1$. Hence $D \equiv_{ty} 1$. □

In particular, any finite directed set is Tukey equivalent to 1. Also notice that two partial orders do not need to have the same cardinality to be Tukey equivalent. Next we consider directed sets without greatest elements.

Lemma 3.3.2. *The Tukey type of a countable directed set without a greatest element is* ω.

Proof. Let $\{d_0, d_1, d_2, \ldots\}$ be an enumeration of D. We define a cofinal map $f : \omega \to D$ as follows. Let $f(0) = d_0$ and let $f(n) = d_{n_k}$ where $d_{n_k} >_D d_i$ for all $i \leq n$. Since f is a cofinal map from $\omega \to D$ we have $D \leq_{ty} \omega$. Note that f is also an unbounded map from $\omega \to D$ and so $\omega \leq_{ty} D$. Hence $D \equiv_{ty} \omega$. □

In Lemma 3.3.2 if D is computable, then f is also computable. More generally, Lemma 3.3.2 holds for any directed set whose cofinality is $\aleph_0$. We also note that 1 is strictly less than ω in the Tukey ordering.

Lemma 3.3.3. *The Tukey type* 1 *is strictly less than* ω *in the Tukey ordering.*

Proof. As before $1 \leq_{ty} D$ for any directed set. Hence $1 \leq_{ty} \omega$. However there cannot be an unbounded map from ω into 1. Any infinite set in ω is unbounded but the set $\{e\}$ is bounded. So all unbounded subsets of ω are mapped to a bounded set in 1. Hence $1 <_{ty} \omega$. □

Hence the Tukey type 1 is characterised by the directed posets with a greatest element and the Tukey type ω contains all countable directed sets without a greatest element. Alternatively we may think of the posets of type ω as those which have a cofinal ω-chain.

So, given a computable directed poset, if we can determine whether or not it has a greatest element, then we know its Tukey type. This has Σ^0_2 complexity.

3.3.2 Maximal Tukey type of computable partial orders

We now consider an arbitrary computable poset. Given a poset $(P, \leq_P)$, the function $f : P \to 1$ defined by $f(p) = e$ is cofinal and so $1 \leq P$ for all posets. As before, unless $(P, \leq_P)$ has a greatest element, 1 is strictly less than P in the Tukey ordering.

We now ask if the partial ordering of Tukey types of countable posets have a greatest element.

Definition 3.3.4. Let $(P, \leq_P)$ be a poset. We say that the elements a and b are *incompatible* in P, if there is no $c \in P$ such that $a <_P c$ and $b <_P c$.

Definition 3.3.5. A *strong antichain* in a poset $(P, \leq_P)$ is a subset A of P in which each pair of distinct elements are incompatible in P.

The existence of a strong antichain characterises the maximal element in the partial ordering of countable posets.

Lemma 3.3.4. *If $(P, \leq_P)$ and $(Q, \leq_Q)$ are countable posets and P has a infinite strong antichain, then $Q \leq_{ty} P$.*

Proof. Suppose A is an infinite strong antichain in P and let $\{a_1, a_2, \dots\}$ be an enumeration of A. Also let $\{q_0, q_1, \dots\}$ be an enumeration of Q. We define $g : Q \to P$ by $g(q_i) = a_i$. Since every pair of distinct elements in A are unbounded, g is an unbounded map from Q into P. Hence $Q \leq_{ty} P$. □

Also, if P has an infinite strong antichain and $P \leq_{ty} Q$, then Q also has an infinite strong antichain.

Lemma 3.3.5. *Suppose P is a countable poset with an infinite strong antichain and let $g : P \to Q$ be an unbounded map. Then Q also has an infinite strong antichain.*

Proof. Suppose $A = \{a_0, a_1, \ldots a_i, \ldots\}$ is an infinite strong antichain in the poset P. Since $g : P \to Q$ is an unbounded map, every pair of distinct elements in A must be sent to a pair of incompatible elements in Q. Hence $g(A) = \{g(a) \mid a \in A\}$ is a strong antichain in Q. □

Hence a poset $(P, \leq_P)$ is in the maximal Tukey type of countable partial orders if and only if $(P, \leq_P)$ has an infinite strong antichain. It was shown by Frittation and Marcone in [**?**] that the existence of an infinite strong antichain in a partial order is equivalent to the existence of arbitrarily large finite strong antichain. So $(P, \leq_P)$ has an infinite strong antichain if and only if for all $n \in \mathbb{N}$ there exists a strong antichain of size n. That is,

$$(\forall n \in \mathbb{N})(\exists a_1, a_2, \ldots, a_n \in P)(\forall i \neq j \leq n)(\forall p \in P)(a_i \not\leq_P p \text{ or } a_j \not\leq_P p).$$

This is a Π^0_3 statement. In fact, the index set of computable posets with an infinite strong antichain is Π^0_3-complete.

Lemma 3.3.6. *There is a computable sequence of posets $\langle P_n \mid n \in \mathbb{N}\rangle$ such that P_n has an infinite strong antichain if and only if W_n is co-infinite.*

Proof. We construct the poset $(P_n, \leq_{P_n})$ in stages. At each stage we enumerate two elements, a_s and x_s, into P_n. We let $x_i \leq_{P_n} x_s$ for each $i \leq s$. In addition, if $i \leq s$ is enumerated into W_n by stage s, we let $a_i \leq_{P_n} x_s$. The a_i's will form a antichain and

the x_i's will form an infinite chain as follows:

$$x_0 \leq_{P_n} x_1 \leq_{P_n} x_2 \leq_{P_n} \ldots x_i \leq_{P_n} x_{i+1} \leq_{P_n} \ldots.$$

We claim P_n has an infinite strong antichain if and only if W_n is co-infinite. The poset P_n has an infinite strong antichain if and only if there are infinitely many a_i such that $a_i \not\leq_{P_n} x_s$ for any s. This happens if and only if there are infinitely many i such that $i \notin W_n$. That is, W_n is co-infinite. □

We let ∞ denote the Tukey type which consists of all posets with an infinite strong antichain. Hence given a countable poset $(P, \leq_P)$ we have $1 \leq_{ty} P \leq_{ty} \infty$.

3.3.3 Tukey type and lean cofinal subsets

Recall that two partially ordered sets P and Q are Tukey equivalent if there exists a cofinal map from P into Q and a cofinal map from Q into P. Hence we seek to understand the cofinal subsets of a partial order. In a sense, we wish to find the 'simplest' or 'leanest' cofinal subsets. We have seen that two directed sets with greatest elements are Tukey equivalent. Any pair of functions between these partially ordered sets, which send the greatest element of one set to the greatest element of the other set, witnesses this equivalence. Although any set containing the greatest element is cofinal, Tukey equivalence is determined by the fact that both posets have greatest elements. We have also seen that two directed sets without greatest elements are Tukey equivalent. This equivalence is witnessed by functions which sends cofinal ω-chains in one set to cofinal ω-chains in the other set. In this case the 'simplest' or 'leanest' type of cofinal subset is an ω-chain. Informally, we would like to think of

the simplest cofinal subset as a cofinal subset which does not contain any elements not necessary to make it cofinal. Diestel and Pikhurko captured this notion when discussing the cofinality of partially ordered sets in [**?**] and [**?**]. They defined a *lean* subset as follows.

Definition 3.3.6. Let $(P, \leq_P)$ be a poset. The least cardinality of a cofinal subset of P is the *cofinality* $\mathrm{cf}(P)$ of P.

Definition 3.3.7. An cofinal set $A \subseteq P$ is *lean* if every subset $A' \subseteq A$ of cardinality at least $\mathrm{cf}(A)$ is cofinal in P.

Hence the Tukey types of directed sets are precisely the lean cofinal subsets. However not every poset has a lean cofinal subset. For example, an infinite strong antichain does not have a lean cofinal subset. The unrelated disjoint union of two ω-chains does not have a lean cofinal set either. In this case, any cofinal set will contain infinitely many elements from each ω-chain and the cofinality is $\aleph_0$. A cofinal set restricted to one ω-chain is a subset of cardinality $\aleph_0$ but it is not cofinal in the poset. In fact, Diestel and Pikhurko showed in [**?**] that any countable poset with a lean cofinal set is directed.

3.3.4 Decomposing partial orders into directed components

We now seek to compute the Tukey type of posets which are not directed and don't have an infinite strong antichain. As a first step, we decompose posets into directed components, where possible. It is always possible to decompose a poset into directed subsets by decomposing it into single points. However, the unrelated disjoint union of two ω-chains clearly has two directed parts. So we would like to partition a set

into as few directed subsets as possible. We may think of this as decomposing a poset into paths and stars. In [**?**] and [**?**], Diestel and Pikhurko captured this notion in a *essential directed* subset; a directed subset which every cofinal subset must intersect with. Following their notation, we begin with some definitions.

Definition 3.3.8. The *up-closure* of $A \subseteq P$ is

$$\lfloor A \rfloor_P = \{x \in P \mid \text{for some } a \in A \text{ we have } a \leq x\}$$

and the *downward closure* of $A \subseteq P$ is

$$\lceil A \rceil_P = \{x \in P \mid \text{for some } a \in A \text{ we have } x \leq a\}.$$

In this way, we think of the ordering as vertical.

Definition 3.3.9. If $A \subseteq P$ is not cofinal in P, we say A is *small* in P. If A is small in P, then we say that the complement of A in P is *essential*.

Hence any cofinal subset of P intersects all of the essential subsets of P. The canonical example of an essential subset is the up-closure of a single point x in P, that is $\lfloor x \rfloor_P$. In fact, it is straightforward to check that a subset of P is essential if and only if it contains the up closure of at least one point in P. In [**?**], Diestel showed that any poset without an infinite antichain may be decomposed into finitely many essential directed sets and the decomposition is unique up to Tukey equivalence. We will strengthen this result by showing that a partially ordered set may be decomposed into finitely many essential directed sets if and only if it does not contain a strong antichain.

Lemma 3.3.7. *If $(P, \leq)$ is a countable partially ordered set, then P can be partitioned into finitely many essential directed subsets if and only if P does not contain an infinite strong antichain.*

Proof. Suppose P contains an infinite strong antichain. By definition, a directed subset of P may contain at most one element of the strong antichain. Hence P cannot be partitioned into finitely many directed sets.

Conversely, suppose P cannot be partitioned into finitely many directed sets. We will show by induction on $n \geq 2$ that P has a strong antichain of size n. By Frittaion and Marcone [**?**], this suffices to show that P has an infinite strong antichain. For the base case, we show that P has a strong antichain of size 2. Take $a_0 \in P$. Then $\lfloor a_0 \rfloor_P$ is an essential set. Either $\lfloor a_0 \rfloor_P$ is directed or not. First consider the case when $\lfloor a_0 \rfloor_P$ is directed. Then there must be $a_1 \in P$ such that $a_1 \not\leq a$ for any $a \in \lfloor a_0 \rfloor_P$. If such an a_1 does not exist, then P is a directed set and hence may be partitioned into one directed essential set. Hence $\{a_0, a_1\}$ forms a strong antichain of size two. For the second case, suppose that $\lfloor a_0 \rfloor_P$ is not directed. Then there exists $a_1 > a_0$ and $a_2 > a_0$ such that a_1 and a_2 are incompatible. So $\lfloor a_1 \rfloor_P$ and $\lfloor a_2 \rfloor_P$ are disjoint essential sets. Now $\{a_1, a_2\}$ forms an strong antichain of size two.

For the induction case, suppose we have a strong antichain $a_0, a_1, \ldots, a_{n-1}$ of size n. We show that there exists a strong antichain of size $n+1$. As $a_0, a_1, \ldots, a_{n-1}$ forms a strong antichain $\lfloor a_0 \rfloor_P$, $\lfloor a_1 \rfloor_P$, $\lfloor a_2 \rfloor_P$, $\ldots$, $\lfloor a_{n-1} \rfloor_P$ are disjoint essential sets. If $\lfloor a_0 \rfloor_P$, $\lfloor a_1 \rfloor_P$, $\lfloor a_2 \rfloor_P$, $\ldots$, $\lfloor a_{n-1} \rfloor_P$ are also directed sets, then there exists a_n such that $a_n \not\leq a$ for any a in $\lfloor a_i \rfloor_P$ where $0 \leq i < n$. If this were not the case, it would be possible to partition P into n many essential directed sets. Hence $\{a_0, a_1, \ldots, a_{n-1}, a_n\}$ is a strong antichain of size $n+1$. Otherwise, suppose that at least one of $\lfloor a_0 \rfloor_P$,

$\lfloor a_1 \rfloor_P$, $\lfloor a_2 \rfloor_P$, ..., $\lfloor a_{n-1} \rfloor_P$ is not directed. Take the least such a_i. Since $\lfloor a_i \rfloor_P$ is not directed we have $a_n > a_i$ and $a_{n+1} > a_i$ such that a_n and a_{n+1} are not computable in P. Hence $\{a_0, a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, a_{n+1}\}$ is a strong antichain of size $n+1$.

As there exists a strong antichain of size n for every $n \in \mathbb{N}$, there exists an infinite strong antichain. □

Remark 3.3.1. To construct an infinite strong antichain in Lemma 1 we need to determine when the up-closure of a point is directed, which is a Π^0_2 statement. We also need to determine if a point x is below any point in the up-closure of another point $\lfloor a \rfloor_P$. This is a Σ^0_1 statement.

Lemma 3.3.8. *If $P = \bigcup_{0 \leq i \leq n} A_i$ is a partition of P into essential directed subsets, then every essential directed subset of P is Tukey equivalent to one of the A_i.*

Proof. Let A be an essential directed subset of P and let $a \in A$ be such that $\lfloor a \rfloor_P \subseteq A$. As $P = \bigcup_{0 \leq i \leq n} A_i$ is a partition of P, we have $a \in A_i$ for some i. We claim that A and such an A_i are Tukey equivalent. Since A_i is essential, $\lfloor b \rfloor_P \subseteq A_i$ for some $b \in P$. As A_i is directed, there is $c \in A_i$ such that $c \geq a$ and $c \geq b$. As $\lfloor a \rfloor_P \subseteq A$ we have $c \in A$ also. Now we have $\lfloor c \rfloor_P \subseteq \lfloor a \rfloor_P \cap \lfloor b \rfloor_P \subseteq A \cap A_i$. Moreover since A and A_i are directed sets, $\lfloor c \rfloor_P$ is cofinal in both sets. Hence any functions between A and A_i which are the identity function on $\lfloor c \rfloor_P$ are cofinal functions between the two sets. So A and A_i are Tukey equivalent. □

Furthermore, the number of essential directed subsets n in the partition is unique.

Lemma 3.3.9. *Let $P = \bigcup_{0 \leq i \leq n} A_i$ and $B = \bigcup_{0 \leq i \leq m} B_i$ be two partitions of P into essential directed subsets. Then $m = n$.*

Proof. Suppose for a contradiction that $m \neq n$. Without loss of generality, suppose $m > n$. Fix $b_0, b_1, \ldots, b_m$ such that $\lfloor b_i \rfloor_P \subseteq B_i$. Then $\{b_0, b_1, \ldots b_m\}$ is a strong antichain. Since $m > n$, there must be a b_i, b_j with $i \neq j$ and some A_k such that $b_i, b_j \in A_k$. However since A_k is a directed set, the set $\{b_i, b_j\}$ is directed, contradicting the fact that $\{b_i, b_j\}$ is a strong antichain. □

Therefore there is a most one partition of a partially ordered set into directed essential sets, up to Tukey equivalence.

3.3.5 Constructing essential directed sets from a maximal strong antichain

As noted before, Marcone and Frittation in [**?**] showed that the existence of an infinite strong antichain is equivalent to the existence of arbitrary large strong antichains. Hence, if a poset P does not contain an infinite strong antichain, then there is a finite bound on the size of the strong antichains in P. If $\{a_1, a_2, \ldots, a_n\}$ form a strong antichain of the maximal size, then the upward closures of these points, $\lfloor a_1 \rfloor_P$, $\lfloor a_2 \rfloor_P, \ldots, \lfloor a_n \rfloor_P$, are essential directed sets. In addition, every element of P is below some point in these sets. So, let A_i be the downward closure of these sets $\lfloor a_i \rfloor_P$. By removing overlaps from the A_i, we have partitioned P into finitely many essential directed sets.

Assuming that a poset P does not contain a infinite strong antichain, we can construct a maximal strong antichain in P as follows. Take $a_0 \in P$. If $\lfloor a_0 \rfloor_P$ is not directed, then we take two points, a_0' and a_0'', above a_0 which are not computable. If the upward closure of either a_0' or a_0'' are not directed then we take two points above each of those which are not computable. Since P does not have an infinite strong

antichain, this process will terminate after finitely many splittings and we will have a strong antichain A_0 such that, for each $a \in A_0$, the upward closure $\lfloor a \rfloor_P$ is directed. Now if each point in P lies below one of these directed sets, then we have found a maximal strong antichain. Otherwise we take a_1 such that a_1 is incompatible with any element of A_0. As with a_0, we continue to split above a_1 until we have a strong antichain A_1 such that $\lfloor a \rfloor_P$ is directed for each $a \in A_1$. Now, if each point in P lies below one of the directed sets, $\lfloor a \rfloor_P$ for $a \in A_0 \cup A_1$, then we have found a maximal strong antichain. Otherwise we take an element of P which is incompatible with $a \in A_0 \cup A_1$ and proceed as before. Since P does not contain an infinitely strong antichain, this process must eventually terminate.

Remark 3.3.2. If we know a poset P does not contain a infinite strong antichain, then we can construct a maximal strong antichain in P if we can determine when the up-closure of a point is directed and if all points in P lie below some point in finitely many directed sets. Both of these questions have Π^0_2 complexity.

3.3.6 Tukey Type of partitioned partially ordered sets

Recall that the Tukey type of a directed set is 1 if that set has a greatest element and the Tukey type of a directed set is ω otherwise. Given a partition of a poset P into essential directed sets, one would hope that the Tukey type of the essential directed subsets would determine the Tukey type of the poset. To this end, we say that a poset P has type (n, m), when P partitions into $n + m$ many directed sets and n of those sets have greatest elements where $n, m \in \mathbb{N}$. We show that two posets are Tukey equivalent if and only if they have the same type.

Lemma 3.3.10. *If P has type (n_1, m_1) and Q has type (n_2, m_2), then $P \leq_{ty} Q$ if and*

only if $n_1 + m_1 \leq n_2 + m_2$ *and* $m_1 \leq m_2$.

Proof. Assume $n_1 + m_1 \leq n_2 + m_2$. We show that $P \leq_{ty} Q$ by constructing a cofinal map from Q into P. Take partitions of P and Q into finitely many essential directed sets of their respective types. Let $p_1, p_2, \ldots, p_{n_1}$ be the greatest elements of the directed sets with greatest elements in the partition and let $D_1, D_2, \ldots D_{m_1}$ denote the directed sets without greatest elements in the partition. Similarly, let $q_1, q_2, \ldots, q_{n_2}$ denote the greatest elements of the directed sets with greatest elements in the partition of Q and let $E_1, E_2, \ldots, E_{m_2}$ denote the directed sets without a greatest element in the partition of Q.

We show that $P \leq_{ty} Q$ by constructing a cofinal map from Q to P. Define $f : E_i \to D_i$ for $1 \leq i \leq m_1$ as follows. Suppose $E_i = \{e_0, e_1, \ldots, e_n, \ldots\}$ and $D_i = \{d_0, d_1, d_2, \ldots, d_n, \ldots\}$ and let $d_{k_0} \leq_P d_{k_1} \leq_P \ldots \leq_P d_{k_n} \leq_P \ldots$ be a cofinal chain in D_i such that $d_j \leq_P d_{k_l}$ for all $j \leq l$. Then, for each $i \leq m_1$, we define $f : E_i \to D_i$ by $f(e_n) = d_{k_n}$ for all $n \in \mathbb{N}$. Hence, the image under f of any infinite subset in E_i is an infinite subsequence of $\{d_{k_l}\}_{l \in \mathbb{N}}$ and so is cofinal in D_i. We map the E_i, for $m_1 < i \leq m_2$, onto the greatest elements of P as follows. Let $f(q) = p_{i-m_1}$ for all $q \in E_i$. If $n_1 < m_2 - m_1$, then map the remaining E_i's onto an arbitrary element of P. We also map the greatest elements in Q onto any remaining greatest elements of P as follows. For $1 \leq i \leq n_2$, let $f(q_i) = p_{m_2 - m_1 + i}$. If $p_{m_2 - m_1 + i}$ does not exist, then we map q_i onto an arbitrary element of P.

We claim that f is a cofinal map from Q into P. Suppose X is a cofinal subset of Q. Hence X contains a cofinal subset E_i^* of E_i for each $1 \leq i \leq m_2$. Since each E_i does not have a greatest element, each E_i^* has cardinality $\aleph_0$. By the definition of f, the image of E_i^* under f is a cofinal chain in D_i for each $1 \leq i \leq m_1$. Hence

$f(X)$ is cofinal in each D_i for each $1 \leq i \leq m_1$, as $m_1 \leq m_2$. It remains to show that $f(X)$ is onto the greatest elements in the partition of P. Since X is cofinal it also contains the greatest elements $q_1, \ldots, q_{n_2}$. As $n_1 + m_1 \leq n_2 + m_2$ and each $E_{m_1+1}, \ldots, E_{m_2}, q_1, \ldots, q_{n_2}$ in Q is mapped onto a distinct p_i for $1 \leq i \leq n_1$ until all the p_i are in the image of f, we know $f(X)$ contains the greatest elements $p_1, \ldots p_{n_1}$ also. Hence f is a cofinal map from Q into P and so $P \leq_{ty} Q$.

Conversely suppose $n_1 + m_1 \not\leq n_2 + m_2$ or $m_1 \not\leq m_2$. We show that it is not possible to have $P \not\leq_{ty} Q$. First we consider the case when $n_1 + m_1 > n_2 + m_2$. Then P contains a strong antichain, A, of size $n_1 + n_2$. Recall that $P \leq_{ty} Q$ if and only if there is an unbounded map g from P into Q. Suppose such a map exists. As each pair of elements in A are unbounded, their image under g must be unbounded in Q. Hence $f(A)$ is a strong antichain of size $n_1 + m_1$ in Q. However the greatest size of a strong antichain in Q is $n_2 + m_2 < n_1 + m_1$. Hence there cannot be an unbounded map from P into Q.

Suppose that $n_1 + m_1 \leq n_2 + m_2$ but $m_1 > m_2$ and that $g : P \to Q$ is an unbounded map. Again P contains a strong antichain $A = \{a_1, a_2, \ldots, a_{n_1+m_1}\}$ of size $n_1 + m_1$. If $p_1 \in \lfloor a_i \rfloor$ and $p_2 \in \lfloor a_j \rfloor$ where $i \neq j$ then $\{p_1, p_2\}$ is unbounded and hence $g(p_1)$ and $g(p_2)$ are contained in distinct directed sets in Q. In addition, the upward closure of m_1 many elements in A are unbounded. Let $B = \{b_1, b_2, \ldots, b_{m_1}\}$ be a strong antichain of length m_1 such that if $b \in B$, then $\lfloor b \rfloor$ is unbounded. Hence $g(\lfloor b_1 \rfloor)$, $g(\lfloor b_2 \rfloor), \ldots g(\lfloor b_{m_1} \rfloor)$ are unbounded sets contained in distinct directed subsets of Q. However, Q has just m_2 many distinct unbounded directed subsets and $m_2 < m_1$. Hence there is no unbounded map from P into Q.

□

So, once we have partitioned a poset into essential directed sets, we can determine the Tukey type of the poset by checking whether or not each essential directed set in the partition has a greatest element. Hence we have the following theorem, initially proved by Day [**?**] in the context of oriented systems.

Theorem 3.3.1. *(Day [?]) If $(P, \leq)$ is a computable poset, then the Tukey type is that of an infinite strong antichain, ∞, or (n, m) where P decomposes into $n + m$ many essential directed sets and n of them have a greatest element.*

Proof. This follows from the previous lemmas. □

For $i \in \mathbb{N}$, let $\leq_i$ be the binary relation computable by the partial computable function φ_i on $\mathbb{N}$. We let $P_i = (\mathbb{N}, \leq_i)$ and note that saying that P_i is a partial order is a partial order is a Π^0_2 statement since it requires saying φ_i is total. We denote the index set of Tukey types as $I_{ty} = \{\langle e, i\rangle \mid P_e \text{ and } P_i \text{ are partial orders and } P_e \equiv_{ty} P_i\}$.

Corollary 3.3.1. *I_{ty} is Turing computable from $\mathbf{0}'''$.*

Proof. We have $\langle e, i\rangle \in I_{ty}$ if and only if φ_e and φ_i define partial orders on $\mathbb{N}$ (which are Π^0_2 conditions) plus one of the following holds:

1. both P_e and P_i have infinite strong antichains (which are Π^0_3 conditions), or
2. there are n and m such that each of P_e and P_i decompose into exactly $n + m$ many essential directed sets and n of them have maximal elements.

Condition 2 is Σ^0_3: there exist $m, n \in \mathbb{N}$ such that

- for all $x_1, \ldots, x_{n+m+1}$, there are $u, v \leq n + m + 1$ with $u \neq v$, such that x_u and x_v are compatible (this is a Π^0_2 condition), and

- there exist elements $x_1, \ldots, x_{n+m}$ such that
  - for all $u \neq v \leq n+m$, x_u and x_v are incompatible (Π^0_1 condition), and
  - for all $u \leq n$, there is a maximal element above x_u (Σ^0_2 condition), and
  - for all $n < v \leq n+m$, there is not a maximal element above x_v (Π^0_2 condition).

So condition 2 has the form

$$\exists\big(\Pi^0_2 \wedge \Sigma^0_3\big)$$

and hence is Σ^0_3. Altogether, the statement defining I_{ty} has the form

$$\Pi^0_2 \wedge \big(\Pi^0_3 \vee \Sigma^0_3\big).$$

As $\mathbf{0}'''$ can answer Π^0_3 and Σ^0_3 questions, we have $I_{ty} \leq \mathbf{0}'''$.

□

We also have the following corollary of Lemma 3.3.6.

Lemma 3.3.11. *Let Q be a fixed computable poset consisting of an infinite antichain. There is a computable sequence of posets $\langle P_n \mid n \in \mathbb{N}\rangle$ such that $\{n \mid P_n \equiv_{ty} Q\} \equiv_T \mathbf{0}'''$.*

Proof. Let $\langle P_n \mid n \in \mathbb{N}\rangle$ be the sequence constructed in Lemma 3.3.6. By construction, the set $\{n \mid P_n$ has an infinite strong antichain$\} \equiv_T \mathbf{0}'''$. The current lemma now follows from the fact that $P_n \equiv_{ty} Q$ if and only if P_n has a strong infinite antichain. □

Hence the complexity of the index set of Tukey types I_{ty} is $\mathbf{0}'''$. In comparison the isomorphism problem, $\{(a,b) \mid P_a, P_b$ are computable posets and $P_a \cong P_b\}$, is Σ^1_1-complete. Clearly the isomorphism type of a partial order gives us more information than the Tukey type. However if the Tukey type of a partial order is (n,m), where $n, m \in \mathbb{N}$, then we have a pretty good idea what the partial order looks like. However if the Tukey type is ∞ then, for example, it may contain a binary tree, $\aleph_0$ many ω-chains, or an infinite antichain. Hence we turn to cofinal similarity to give us a better picture in this case.

3.4 Cofinal similarity of computable partial orders

Definition 3.4.1. Two partially ordered sets are *cofinally similar* if there exists a third partial order into which they both embed cofinally. That is, there is a partially ordered set R with cofinal sets $C_P, C_Q \subseteq R$ such that P is order isomorphic to C_P and Q is order isomorphic to C_Q.

Cofinal similarity turns out to be an equivalence relation, and the equivalence classes are called *cofinal types*. If P is cofinally similar to Q, then we write $P \equiv_{cf} Q$. Day showed in [**?**] that for directed sets cofinal similarity coincides with Tukey equivalence. He also showed that cofinal similarity and Tukey equivalence coincide when a partially ordered set can be partitioned into finitely many essential directed sets. However, not all posets with an infinite strong antichain are cofinally similar. For example, an infinite strong antichain cannot preserve order and embed cofinally into a poset which has a cofinal binary tree or cofinal set which is isomorphic to $\aleph_0$ many ω-chains. Similarly a partial order with $\aleph_0$ many ω-chains cannot embed

cofinally into a partial order with a cofinal binary tree.

In what follows, let $(P, \leq_P)$ be a partially ordered set.

Definition 3.4.2. We say that B is an *essential cofinal subset* of P if B is cofinal in some essential subset C of P.

Everywhere branching trees are important in this context.

Definition 3.4.3. Let P be a poset. An $A \subseteq P$ is called an *everywhere branching tree* if for all $a \in A$ there exists incompatible elements $a_1, a_2 \geq_P a$.

Following Day's notation in [**?**] we make the following definitions.

Definition 3.4.4. Let $P_1 = \{p \in P \mid \lfloor p \rfloor \text{ is directed}\}$ and $P_2 = \{q \in P \mid \forall p \geq_P q\, (p \notin P_1)\}$.

So we have determined two disjoint subsets of P. The first part P_1 consists of directed subsets of P and the second part P_2 consists of a maximal everywhere branching tree in P. Note that if P is computable, then P_1 is Π^0_2 and P_2 is Π^0_3.

As already mentioned, every poset with no infinite antichain can be partitioned into finitely many essential directed sets. In addition to this, Diestel and Pikhurko showed in [**?**] that a countable partially ordered set $(P, \leq)$ does not have an essential cofinal subset isomorphic to an ever-branching tree if and only if it admits a partition into essential directed sets.

As infinitely many copies of $2^{<\omega}$ or $\omega^{<\omega}$ can embed into a single copy, we need only consider the existence or absence of an everywhere branching tree. Again, following Day's notation, we make the following definition.

Definition 3.4.5. Let $\mathcal{E}_0$ denote the partial order that consists of countably many disjoint copies of $\omega^{<\omega}$.

Day proved in [**?**] Lemma 7.2 that a countable poset P is ever-branching (meaning that $P_2 = P$) if and only if P has a cofinal subset isomorphic to $\mathcal{E}_0$. Also in Theorem 4.8 [**?**], Day gives the following properties, which outline how a countable poset P can be decomposed to understand its cofinal parts.

- P_1 and P_2 are disjoint and at least one is non-empty.
- P_1 and P_2 are each upward closed.
- $P_1 + P_2$ is cofinal in P.
- If P_2 is non-empty, then $(P_2, \leq_P)$ is an everywhere branching poset and hence has a cofinal subset isomorphic to $\mathcal{E}_0$ by Lemma 7.2 in Day.
- If P_1 is non-empty, then $(P_1, \leq_P)$ can be partitioned into essential directed sets, each of which either has a maximal element or has a cofinal ω-chain. However, unlike the Tukey type case there may be infinitely many of each of these types of essential directed subsets.

So, to determine the cofinal type of a countable partially ordered set we need to determine if it contains any essential cofinal subsets isomorphic to a everywhere branching tree and then partition the remainder of the poset into essential directed sets. Hence, with each countable poset P we associate a tuple (n_P, m_P, l_P) such that n_P, where $0 \leq n_P \leq \aleph_0$, is the number of essential directed subsets with a greatest element in a partition of P, and m_P, where $0 \leq m_P \leq \aleph_0$, is the number of essential directed subsets without a greatest element in a partition of P, and l_P, where $l_P \in \{0, 1\}$ denotes the existence of a essential cofinal subset isomorphic to an ever-branching tree. If $l_P = 1$, then P contains an essential cofinal subset isomorphic

to an ever-branching tree and if $l_P = 0$ it does not. Day found a classification of countable oriented systems under cofinal similarity in terms of these three elements.

Lemma 3.4.1. *[?, Corollary 7.4] Two countable partially ordered sets are cofinally similar if and only if their associated triples are the same.*

Now that we know the tuple (n_P, m_P, l_P) associated with a computable partial order P determines the cofinal type, we ask how complicated it is to find the tuple (n_P, m_P, l_P). To do this we need to determine if a computable poset P contains an essential cofinal subset which is isomorphic to an everywhere branching tree. First, we make the following observation.

Theorem 3.4.1. *The complexity of finding the cofinal type of a computable poset is at most* $\Sigma^0_4 \wedge \Pi^0_4$.

Proof. Given a computable poset P, to compute the tuple (n_P, m_P, l_P) we need to find out if P contains a cofinal essential everywhere branching tree and then decompose the remainder of P into essential directed posets.

Firstly, we need to determine if P contains a cofinal essential everywhere branching tree. This is equivalent to asking if P_2 is non-empty, which is Σ^0_4 question. This determines the third component of the cofinal type.

Secondly, we need to determine the number of directed components in P_1 and whether or not these directed components have a maximal element. Note that an element is maximal in $(P_1, \leq_P)$ if and only if it is maximal in $(P, \leq_P)$. To be a maximal point in P is a Π^0_1 condition. Therefore, to ask if there are infinitely many is a Π^0_3 question and to ask if there are exactly n many maximal points is a $\Sigma^0_2 \wedge \Pi^0_2$ question (i.e. there are n many maximal points and there are not $n+1$ many maximal

points). The answers to these questions determine the first component of the cofinal type.

Finally, we need to determine how many maximal directed sets in P_1 have cofinal type ω. To ask if there are infinitely many, we need to ask whether for every n, there are n many distinct elements $x_1, \ldots, x_n$ in P_1 such that each pair $x_i \neq x_j$ is incompatible. Since being in P_1 is Π^0_2 and being incompatible is Π^0_1, this question is Π^0_4. To ask if there are exactly n many such points, we need to ask whether there are n many pairwise incompatible elements of P_1 (a Σ^0_3 condition) but not $n+1$ such elements (a Π^0_3 condition). □

It follows that the triple describing the cofinal type of a computable partial order can be uniformly computed by $\mathbf{0}''''$. We conjecture that this bound is sharp. That is, the complexity of the index set

$$I_{cf} = \{\langle e, i\rangle \mid P_e, P_i \text{ are posets and } P_e \equiv_{cf} P_i\}$$

is exactly $\mathbf{0}''''$. The proof of this conjecture is ongoing research.