Ordered structures and computability

By Jennifer Chubb

M.S. in Applied Mathematics, George Mason University, 2003

B.S. in Physics and Applied Mathematics, George Mason University, 1999

A Dissertation submitted to

The Faculty of
the Columbian College of Arts and Sciences
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

May 7, 2009

Dissertation directed by

Valentina Harizanov

Professor of Mathematics

The Columbian College of Arts and Sciences of The George Washington University certifies that Jennifer Chubb has passed the Final Examination for the degree of Doctor of Philosophy as of May 7, 2009. This is the final and approved form of the dissertation.

Ordered structures and computability

Jennifer Chubb

Dissertation Research Committee:

Valentina Harizanov (GWU), Professor of Mathematics, Dissertation Director

Ali Enayat (American University), Professor of Mathematics, Reader

Jozef Przytycki (GWU), Professor of Mathematics, Reader

Michael Moses (GWU), Associate Professor of Mathematics, Reader

Dedication

For Valentina, Brian, and my Mom and Dad. And me.

Abstract

We consider three questions surrounding computable structures and orderings.

First, we make progress toward answering a question of Downey, Hirschfeldt, and Goncharov by showing that for a large class of countable linear orderings, the Turing degree spectrum of the successor relation is closed upward in the c.e. degrees.

Next we consider computable partial orders (specifically, finitely branching trees) with, in addition to their ordered structure, finite-range functions (coloring functions) on the collection of chains of length n in the ordering. In the style of Ramsey, we prove the existence of a monochromatic substructure and analyze the axiomatic content of this theorem in the context of reverse mathematics. Then we provide a bound on the complexity of the monochromatic substructure in the case that the coloring function is computable.

Finally, we consider computable algebraic structures (in particular, groups and semigroups) and the algorithmic complexity of orderings of their elements. We give conditions sufficient to ensure that a group has orderings of arbitrary computability theoretic complexity (in a strong sense), discuss an interesting example, and give a useful representation of the orderings of a computable semigroup as paths in a computable binary tree.

Contents

Dedication **iii**

Abstract **iv**

Contents **iv**

1 Introduction **1**

1.1 Basic notions from computability theory 1

2 Linear orderings **8**

2.1 Motivation 8

2.2 The successor relation 11

2.3 Upwards closure of $DgSp_{\mathcal{L}}(Succ_{\mathcal{L}})$ in the c.e. degrees 14

2.4 Concluding remarks and proposed problems 27

3 Colored trees **28**

3.1 Motivation, and some basic definitions and facts 28

3.2 Notation and basic observations 30

3.3 Reverse Mathematics 31

3.4 A Π^0_2 bound 41

3.5 The Π^0_n bound 53

4 Orderings and algebraic structures 65

4.1 Basic notions 65

4.2 An ordering in every tt-degree 71

4.3 Computable copies of $\mathbb{Z}^\omega$ with no computable ordering 74

4.4 Trees of orderings of semigroups 81

Chapter 1

Introduction

In many parts of mathematics, ordered structures play a fundamental role. The natural numbers in their usual ordering, collections of nested sets (ordered by containment), trees, Boolean algebras, and other partial orders—these kinds of structures are pervasive and indispensable in mathematics as we know it. In this dissertation, we consider the computability theoretic properties of ordered *computable structures*. Even when it is possible to know (by running a program) any finite *basic* piece of information about a structure, it is not necessarily the case that we may find any information we like, finite or otherwise. This provides much of the motivation for what follows.

1.1 Basic notions from computability theory

Our formalization begins with the notion of *computability*[1]. A set, function, or relation on the natural numbers, $\mathbb{N}$, is *computable* if there is a Turing machine (or a computer program in

[1] See [52] for a thorough exposition.

an acceptable[2] programming system) that computes membership, output, and truth value, respectively, for any natural number input. Once a programming system is selected and fixed, programs in that language may be enumerated in a systematic way by listing finite syntactically correct strings of symbols from the programming language. We denote this list of programs by $P_0, P_1, \ldots$, and the corresponding partial function on $\mathbb{N}$ computed by the eth program by φ_e. Note that not all syntactically correct programs P_e will be *robust*, i.e., it is possible that an algorithm gets "stuck" or loops infinitely for some inputs. Thus, for many $e \in \mathbb{N}$, it is the case that $\mathrm{dom}(\varphi_e) \subsetneq \mathbb{N}$.

We call the collection of all functions φ_e the *partially* computable functions; computable functions are those with $\mathrm{dom}(\varphi_e) = \mathbb{N}$. A set A is computable if its characteristic function,

$$\chi_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

is computable.

A more general notion, and arguably the deepest and most important in the general theory of computability, is that of *relative computability*. We say a set (or other countable object, suitably coded into the natural numbers) A *is computable relative to* B (another set or object) if, when we have free access to membership information about B, we are able to compute A.

To formalize this, include among the basic commands of the acceptable programming system fixed above, a command that allows a program to query an *oracle set*, $O \subseteq \mathbb{N}$. (We

[2] A programming system is *acceptable* if it is universal, and satisfies a version of the Parameter Theorem. For details, consult [16, 52].

could use, for example,

$$\texttt{If } x \in O \texttt{ goto } L_1\texttt{, else goto } L_2,$$

where L_1 and L_2 are labels of lines of the program.) Augmenting the language of the programming system in this way leads to a different (but again, systematic) enumeration of *oracle programs*, $P_0^O, P_1^O, \ldots$. Depending on which set of natural numbers is used as the oracle set, the corresponding functions φ_e^O vary. In this framework, a set A is computable relative to B (we write $A \leq_T B$ and often say that A is *Turing reducible* to B) if there is an index e so that φ_e^B computes the characteristic function of the set A. Note that included in this list of oracle programs are programs containing no oracle commands. In other words, the collection of non-oracle programs described above are included in the list of oracle programs no matter what oracle set is used. Thus for any set B, if A is computable, $A \leq_T B$.

This reducibility is a pre-ordering[3] on the power set of $\mathbb{N}$. When $A \leq_T B$ and $B \leq_T A$, we say sets A and B are *Turing equivalent* and write $A \equiv_T B$ (for example, for any set A, $A \equiv_T \overline{A}$, where $\overline{A}$ denotes the complement of A). The relation $\equiv_T$ is an equivalence relation on the power set of the natural numbers, and the equivalence classes are called *Turing degrees*. The collection of Turing degrees under the ordering induced by $\leq_T$ forms an upper-semilattice with a least element (the degree of the computable sets). We denote the Turing degree of the computable sets by $\mathbf{0}$ and that of the *halting set*[4], $K = 0' = \{e \mid \varphi_e(e)$

[3] A *pre-ordering* is a reflexive, transitive binary relation.

[4] The *halting set* is the *jump* of the empty set. In general, the jump of set A is

$$A' = \{e \mid \varphi_e^A(e) \text{ eventually halts}\}.$$

The inequality $A \leq_T A'$ is proper.

eventually halts}, by $\mathbf{0}'$.

Some sets, though they may not be computable themselves, may be enumerated by a program. Formally, there is an e so that φ_e is total, and the range of φ_e is exactly the set. These form the important class of *computably enumerable (c.e.)* sets. Provided the set is infinite, we may assume, when it is convenient, that the enumerating function is 1-1. An equivalent characterizing property for a c.e. set is that for some e, the set is exactly the domain of φ_e. Post's theorem asserts that a set is computable if and only if both it and its complement are c.e. A Turing degree is called a *computably enumerable* or *c.e. degree* if it contains a c.e. set.

Now, we give the definition that will facilitate consideration of computability theoretic properties of *structures*. We will present only the most necessary and basic ideas. For a more complete exposition, see, for example, [3] or [28]. All languages are assumed to be finite, and because in the context of computability theory, finite structures are trivial, we will assume all structures to be infinite.

A mathematical structure $\mathcal{A}$ is *computable* if its universe $|\mathcal{A}|$ is a computable set under a suitable coding into the natural numbers (without loss of generality, we may assume it is the natural numbers), and all basic operations and relations are computable[5]. Equivalently, a structure is computable if we can write a computer program that given a quantifier-free formula in the language of $\mathcal{A}$, $\psi(\vec{x})$, as input, and any tuple of elements $\vec{a}$ (of appropriate length) from $|\mathcal{A}|$, will determine whether $\psi(\vec{a})$ holds or not. (In other words, the *atomic diagram* of a computable structure is computable as a set, once suitably coded into the natural

[5]When the language is infinite, we require that the basic relations and functions are *uniformly computable*, see [3] or [28].

numbers.) For example, a linear ordering $\mathcal{L} = (L, <)$ is computable if L is a computable set, and there is a computer program accepting any pair of elements, a and b, of L as input, that outputs 1 (interpreted as "True") if $a \leq_{\mathcal{L}} b$, and 0 (or "False") otherwise.

An *additional relation* on a computable structure is one that is not a basic part of the structure, i.e. it is not part of the language. (For example, if $\mathcal{L} = (\mathbb{N}; <_{\mathcal{L}})$ is a linear ordering, the successor relation, $Succ_{\mathcal{L}}$, is an additional binary relation on $\mathcal{L}$. This example is the topic of Chapter 2.) Even when the underlying structure is computable, an additional relation may not be computable. If the relation is definable in the first-order language of the structure, the defining formula gives some upper bound on the algorithmic complexity of the relation. Furthermore, the algorithmic properties of such a relation may change under isomorphism to other computable copies of the structure. The algorithmic complexity of such a relation therefore depends not necessarily on the isomorphism type, but on the copy of the structure under consideration. For this reason, it is necessary to consider all isomorphic computable copies of a structure, and the complexity of each respective instance of the relation, to gain a complete understanding of the algorithmic properties of the relation itself.

For a given structure $\mathcal{A}$ and definable relation R, we consider the spectrum of Turing degrees realized by the image of that relation in all computable copies of the structure. This collection of Turing degrees is the *degree spectrum*[6] *of* R *in* $\mathcal{A}$,

$$DgSp_{\mathcal{A}}(R_{\mathcal{A}}) = \{\deg(f(R)) \mid f : \mathcal{A} \cong \mathcal{B}, \text{ where } \mathcal{B} \text{ is computable}\}.$$

In the context of computability structure theory the following are two natural questions

[6]This definition was first presented in [32].

that arise concerning ordered structures.

1. Given a computable ordered structure of some kind, how hard (computability theoretically) is it to find some other relation of interest on that structure?

2. Given a computable structure of some kind, how hard is it to order its elements in a way that is relevant to the structure?

In Chapter 2, we will consider the computability theoretic properties of a definable relation (the successor relation) in different computable copies of the same linear ordering.

In Chapter 3, we examine computable partial orderings (in particular, binary trees) having in addition to the ordered structure a computable function of finite range. We call these *colored trees*, and the function a *coloring function*. We will ascertain the algorithmic complexity associated with extracting, in analogy with Ramsey's Theorem, a homogeneous substructure (in this case, a monochromatic embedded binary tree). This relation, "being an element of the monochromatic subtree" is not definable, since there is more than one such substructure that satisfies the criteria. We will give bounds on the complexity of the "simplest" substructure guaranteed to exist and satisfy the description for a given computable copy of the structure.

In Chapter 4, we consider the algorithmic complexity encountered in ordering the elements of computable groups and semigroups so that the ordering (as a binary relation) is invariant under the group operation. Again, since such an ordering need not (indeed, it *cannot*) be unique for a given group, a relation "$x < y$" satisfying the ordering criteria, is not definable, but nonetheless, we can ask how difficult it might be to obtain such an ordering.

We will give conditions ensuring that a group admits an ordering in every *truth-table* degree[7], discuss an interesting example, and consider the space of orderings for a group. We will end the chapter by showing that the collection of bi-orderings of a computable semigroup can be represented as the collection of paths in a computable tree.

[7]The truth-table degrees are a refinement of the Turing degrees and will be formally described in Chapter 4. Intuitively, two sets share the same truth-table degree if they encode the same information, and we can obtain information about one from the other in an entirely robust way.

Chapter 2

Linear orderings

2.1 Motivation

An infinite linear ordering $\mathcal{L}$ is computable just in case its domain is a computable set (without loss of generality, we will assume it is always $\mathbb{N}$), and $<_{\mathcal{L}}$ is a computable relation. Even in such a simple computable structure as this, much that we might like to know can remain algorithmically inaccessible. For example, given x and y, is y the immediate successor of x? Does x have a successor at all? Are there finitely many or infinitely many elements between them? Is x a limit point in $\mathcal{L}$? From the left or right or both? What is the order-type of $\mathcal{L}$?

The defining formulas (not all of them first-order) of these questions give some clue as to how difficult it might be to answer them:

1. y is the immediate successor of x in $\mathcal{L}$ (we write $Succ_{\mathcal{L}}(x, y)$) if and only if

$$Succ_{\mathcal{L}}(x, y) \equiv_{def} x <_{\mathcal{L}} y \wedge \neg(\exists z)[x <_{\mathcal{L}} z <_{\mathcal{L}} y].$$

2. x has a successor in $\mathcal{L}$ if and only if $\exists y\, Succ(x, y)$.

3. There are finitely many elements between x and y in $\mathcal{L}$ (we write $Block_{\mathcal{L}}(x, y)$) if and only if

$$\bigvee_{n\in\omega} (\forall z)[x <_{\mathcal{L}} z <_{\mathcal{L}} y \implies z <_{\mathcal{L}} n].$$

4. x is a limit point from the right in $\mathcal{L}$ if and only if

$$\exists y\, [y >_{\mathcal{L}} x \wedge \forall y\, \neg Succ(x, y)].$$

5. $\mathcal{L}$ has order-type α if and only if $(\exists f)[f : \alpha \cong \mathcal{L}]$.

Intuitively, each existential first order quantifier requires a search through the universe of $\mathcal{L}$ to find an element satisfying some property, and ascertaining non-existence requires an infinite, exhaustive, failed search. Each universal quantifier requires an exhaustive checking that a property holds for every element. The infinite disjunction in (3) behaves similarly to an existential quantifier in practice (once all is translated into the language of arithmetic). In (5), there is a second-order quantifier, which requires a more complicated kind of algorithm.

In some instances, the desired property may be easy to check. For example, if $\mathcal{L}_{\mathbb{Q}}$ is a computable copy of the rational numbers, $Succ_{\mathcal{L}_{\mathbb{Q}}}(x, y)$ is the constant function 0 (for "False"), as is $Block_{\mathcal{L}_{\mathbb{Q}}}$. If $\mathcal{L}_{\omega}$ is the natural numbers in their usual ordering, we can write a program to compute the successor relation:

Given x and y,

if $y = x + 1$ then output 1,

otherwise, output 0.

Thus, the defining formula does not give an exact description of the algorithmic complexity of a relation in all instances, it only provides an upper bound.

This chapter is concerned with the algorithmic complexity of the successor relation in computable linear orderings, which has been of interest in computability theory for some years. In the early 1980's Remmel [45] discovered that a necessary and sufficient condition for a computable linear ordering to be computably categorical[1] is that the successor relation is finite (the result follows from independent work of Dzgoev and Goncharov [23]). For these linear orderings, $DgSp_{\mathcal{L}}(Succ_{\mathcal{L}}) = \{\mathbf{0}\}$ since a finite set is always computable. This result paralleled another theorem of Remmel [44] ensuring computable categoricity of computable Boolean algebras when the *Atom* relation is finite. (An element of a Boolean algebra is an *atom* if it is not the least element and there is no element properly between it and the least one.)

The connection between *Atom* and *Succ* is a strong one, and is established via the *interval algebra* of a given linear ordering. If $\mathcal{L}$ is a linear ordering, then the Boolean algebra generated by the half-open intervals of $\mathcal{L}$ has an atom corresponding to each successor pair in $\mathcal{L}$: If $Succ_{\mathcal{L}}(a, b)$ holds, then $[a, b)$ is a singleton, so non-trivial and an atom in the corresponding interval algebra. If $\mathcal{L}$ is computable so is the induced interval algebra, so it is natural to imagine that *Atom* and *Succ* might exhibit similar algorithmic properties, and indeed in many respects they do. It is thus worth noting that this is not always the case: Downey and Moses [22] gave a construction of a linear ordering having the property that the degree

[1] A computable structure $\mathcal{A}$ is *computably categorical* if there is always a computable isomorphism between computable copies of $\mathcal{A}$.

spectrum of the successor relation is exactly $\{\mathbf{0}'\}$, in contrast to another of Downey's results, that every computable Boolean algebra has a computable copy, $\mathcal{B}$, with $Atom_{\mathcal{B}} \leq 0'$.

In a study of relations on Boolean algebras [17], Downey, Goncharov, and Hirschfeldt asked about the structure of the degree spectrum of the successor relation in linear orderings (assuming it is not finite), specifically, they asked whether the spectrum can be a single degree other than the computable or complete degree. Remmel showed in [44] that the degree spectrum for the atom relation in a computable Boolean algebra is closed upward in the c.e. degrees, provided the relation is infinite. So it is natural to ask if the same is the case for the successor relation. If this is the case, then, of course, the answer to the question of Downey, Goncharov, and Hirschfeldt is no. The main results in this chapter make substantial progress toward answering this question. We will see that for a large class of linear orderings, the degree spectrum of successor is closed upward in the c.e. degrees, and will classify by order type those linear orderings for which the question remains open.

2.2 The successor relation

We begin with an example illustrating an important fact mentioned in the previous section. Namely, the isomorphism type of a computable structure does not specifically determine the algorithmic properties of additional relations. In both examples described above (when *Succ* was finite, and in the example due to Downey and Moses), the isomorphism type of the linear ordering does exactly determine the algorithmic properties of the successor relation. In the example below, this is not the case.

We denote the natural numbers in their usual order by $\mathcal{L}_\omega = (\mathbb{N}, <)$. Of course, in the

usual copy of this structure the successor relation is computable. There are isomorphic linear orderings, though, in which the successor relation is highly non-computable. In fact, it can be arranged that the relation has any c.e. degree:

Example 2.2.1. Let A be any infinite computably enumerable set, and $\{a_0, a_1, a_2, \dots\}$ be a one-to-one computable enumeration of its elements. We construct a computable linear ordering $\mathcal{L}_A$ isomorphic to $\mathcal{L}_\omega$ in which the successor relation is exactly as difficult to compute as the set A itself. The universe of $\mathcal{L}_A$ will be $\mathbb{N}$, and the order relation will be computable.

Construction.

Stage 0. Begin with a skeleton, ordering just the even numbers:

$$0 <_{\mathcal{L}_A} 2 <_{\mathcal{L}_A} 4 <_{\mathcal{L}_A} 6 <_{\mathcal{L}_A} \dots$$

Stage $s+1$. If $a_s = n$, insert $2s+1$ between $2n$ and $2n+2$:

$$2n <_{\mathcal{L}_A} 2s+1 <_{\mathcal{L}_A} 2n+2$$

This ends the construction.

Because A is infinite, all odd numbers are eventually placed so the universe is $\mathbb{N}$. Furthermore, since at most one odd number is inserted between any pair of consecutive even numbers, it is easy to see that the ordering will be isomorphic to $\mathcal{L}_\omega$. To determine if $x <_{\mathcal{L}_A} y$, we can check in cases: If both x and y are even, we use the usual order. If one is odd, say y, then the relation holds only if $(y-1)/2$ is not one of the first $x/2-1$ elements in the (computable) enumeration of A. The remaining cases are similar, and the basic order

relation is thus computable.

It remains to assess the complexity of the successor relation.

$Succ_{\mathcal{L}_A}(x, y)$ holds exactly when

1. both x and y are even and $x/2$ never appears in the enumeration of A,

2. x is even, y is odd, and $x/2$ is $a_{(y-1)/2}$, or

3. x is odd, y is even, and $(y-2)/2$ is $a_{(x-1)/2}$.

The first case is the only one that is not simply computable (because this is where the coding happened). If we know exactly which numbers are in A, we will know if $x/2$ will ever appear in its enumeration. Thus, $Succ_{\mathcal{L}_A} \leq_T A$.

For the other reduction, note that $n \in A$ if and only if $\neg Succ_{\mathcal{L}_A}(2n, 2n+2)$, and so $A \leq_T \overline{Succ_{\mathcal{L}_A}} \equiv_T Succ_{\mathcal{L}_A}$. We conclude that the relation and the set have the same Turing degree.

The example above shows that given any c.e. set, we can construct a computable linear ordering of type ω where the successor relation is exactly as complicated as that set. In fact, this exhausts the possibilities for the complexity of the successor relation in a computable linear ordering in the following sense.

Considering the defining formula for the successor relation, we can easily imagine a program having as its domain exactly the complement of $Succ_{\mathcal{L}}$, provided that the linear ordering $\mathcal{L}$ is computable:

Input x and y.

If $y \leq_{\mathcal{L}} x$, halt and output 1.

If $x <_{\mathcal{L}} y$ search for n properly $<_{\mathcal{L}}$-between x and y.

If such an n is found, halt, output 1.

(If no such n exists, the algorithm will search forever.)

Note that this program halts if and only if $\neg Succ(x, y)$. We see from this that the Turing degree of the successor relation of any computable linear ordering is *always* a c.e. degree. Example 2.2.1 shows that every c.e. Turing degree contains a set that codes the successor relation in a computable linear order of type ω.

We now turn to the question of whether the Turing degree spectrum of the successor relation in linear orderings is closed upward in the c.e. Turing degrees (provided that it is infinite).

2.3 Upwards closure of $DgSp_{\mathcal{L}}(Succ_{\mathcal{L}})$ in the c.e. degrees

The following theorem (our main result in this chapter) shows that for a large class of linear orderings, the degree spectrum of the successor relation is closed upward in the c.e. degrees. Later, we will see a characterization of those linear orderings for which the question remains open. We will also establish that every upper cone of c.e. Turing degrees may be realized as the degree spectrum of the successor relation in some computable linear ordering.

Theorem 2.3.1 ([9]). *Let $\mathcal{L}$ be a computable linear ordering with domain $\mathbb{N}$ such that*

(U) For every $x \in \mathbb{N}$ there are $a, b \in \mathbb{N}$ with $Succ_{\mathcal{L}}(a, b)$ and $x <_{\mathcal{L}} a$.

Let A be a c.e. set so that $Succ_{\mathcal{L}} \leq_T A$. Then there exists a computable linear ordering $\mathcal{M} \cong \mathcal{L}$ with $Succ_{\mathcal{M}} \equiv_T A$.

Proof. Let the computable linear ordering $\mathcal{L}$ and c.e. set A satisfy the hypotheses of the theorem. Without loss of generality, we assume A is infinite.

We construct the computable linear ordering $\mathcal{M}$ in stages, with $|\mathcal{M}| = \mathbb{N}$ so that $\mathcal{M} \cong \mathcal{L}$. Our aim is to code the set A into the complement of $Succ_{\mathcal{M}}$, as was done in Example 2.2.1. Our construction is complicated by the fact that we do not know *a priori* the order type of $\mathcal{L}$, as we did in the example.

At each stage s of the construction we will specify the linear ordering $<_{\mathcal{M}}$ on $\{0, \ldots, s\}$, uniquely determining a partial isomorphism $f_s : \{0, \ldots, s\} \to \{0, \ldots, s\}$ from $\mathcal{L}$ to $\mathcal{M}$. In other words, for all $i < j \leq s$, $i <_{\mathcal{L}} j$ holds in $\mathcal{L}$ if and only if $f_s(i) <_{\mathcal{M}} f_s(j)$ holds in $\mathcal{M}$.

To simplify our exposition, we introduce the following scheme for representing the elements of $\mathcal{L}$ and $\mathcal{M}$ under consideration at stage s. Let $l_0^s, \ldots, l_s^s$ be the elements of the set $\{0, \ldots, s\}$ in increasing $<_{\mathcal{L}}$-order, and $r_0^s, \ldots, r_s^s$ be the elements of $\{0, \ldots, s\}$ in increasing $<_{\mathcal{M}}$-order. Thus for all $j \leq s$, we have $f_s(l_j^s) = r_j^s$. It will also be convenient to take $l_{-1}^s <_{\mathcal{L}} x <_{\mathcal{L}} l_0^s$ and $l_s^s <_{\mathcal{L}} x <_{\mathcal{L}} l_{s+1}^s$ to simply mean $x <_{\mathcal{L}} l_0^s$ and $l_s^s <_{\mathcal{L}} x$, respectively.

As explained above, the complement of the successor relation of $\mathcal{L}$,

$$\overline{Succ_{\mathcal{L}}} = \{(x, y) \mid x \not<_{\mathcal{L}} y \vee \exists u(x <_{\mathcal{L}} u <_{\mathcal{L}} y)\},$$

is a c.e. set. We define a computable approximation of $\overline{Succ_{\mathcal{L}}}$ by finite sets, $\{\overline{C_s}\}_{s \in \omega}$, as follows:

$$\overline{C_s} = \{\langle x, y\rangle \mid (x, y \leq s) \ \wedge \ (x \not<_{\mathcal{L}} y \vee (\exists u \leq s)[x <_{\mathcal{L}} u <_{\mathcal{L}} y])\},$$

where $\langle \cdot, \cdot \rangle$ is a fixed computable pairing bijection[2].

[2]For example, we may take $\langle n, m\rangle = 2^n(2m + 1) - 1$, which is a computable bijection from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$.

The complement of these finite sets gives, for each s, an infinite computable collection of coded pairs $\langle x, y\rangle \in C_s$ approximating the set of successor pairs of $\mathcal{L}$ at stage s. Let $\{c_0^s < c_1^s < \dots\}$ be an enumeration of elements of C_s in the usual order of magnitude, and $c_n = \lim_s c_n^s$ (observe that this limit, of course, exists). The collection $\{c_n\}_{n\in\omega}$ of these limit values is an enumeration of all the successor pairs of $\mathcal{L}$.

Now recursively define the following family of computable functions. Let

$$g_s(0,0) = c_0^s, \text{ and}$$
$$g_s(e,d) = \min\{c_n^s = \langle a, b\rangle \mid (\forall \langle e', d'\rangle < \langle e, d\rangle)[g_s(e', d') < c_n^s] \ \wedge \ d <_{\mathcal{L}} a\}.$$

For each n, c_n^s converges to c_n, and $\mathcal{L}$ satisfies condition (U) so it is clear that for all e and d, the limit, $g(e,d) = \lim_s g_s(e,d)$ exists. It will be equal to the least coded successor pair $c_n = \langle a, b\rangle$ such that for all $\langle e', d'\rangle < \langle e, d\rangle$ we have $g(e', d') < c_n$ and $d <_{\mathcal{L}} a$. If we consider the collection of successor pairs coded by $g(e, \cdot)$ for a fixed e, we see that it contains pairs with arbitrarily large (with respect to $<_{\mathcal{L}}$) first coordinates. So for each e, $g(e, \cdot)$ generates a unique sequence of successor pairs witnessing that (U) holds of $\mathcal{L}$. These sequences of codes of ordered pairs are pairwise disjoint.

For the arbitrarily selected infinite c.e. set A, let $\{A_s\}_{s\in\omega}$ be a computable sequence of finite sets such that $A_s \subseteq A_{s+1}$ and $A = \cup_s A_s$. The construction aims to satisfy two types of requirements.

The first type of requirement serves to ensure that the set A is coded into the complement of $Succ_{\mathcal{M}}$. These are met by arranging a pair of witnesses for each $e \in \mathbb{N}$, a^e and b^e, so that $e \in A$ if and only if (a^e, b^e) is not a successor pair of $\mathcal{M}$. The witnesses will be approximated

at each stage, and in the end we will have $a^e = \lim_s a^e_s$ and $b^e = \lim_s b^e_s$.

Natural number e *is coded in* $\mathcal{M}$ *at stage* s if and only if $a^e_s <_{\mathcal{M}} b^e_s$ and either

1. $e \notin A_s$ and there is no $x \leq s$ so that $a^e_s <_{\mathcal{M}} x <_{\mathcal{M}} b^e_s$ (in other words, (a^e_s, b^e_s) appears to be a successor pair at this stage), or

2. $e \in A_s$ and there is an $x \leq s$ so that $a^e_s <_{\mathcal{M}} x <_{\mathcal{M}} b^e_s$.

If no such pair of witnesses a^e_s and b^e_s exists at this stage, e *is not coded in* $\mathcal{M}$ *at stage* s. An $e \in \mathbb{N}$ may be coded in $\mathcal{M}$ at stage s and not coded at some $s' > s$. We will see that this decoding can only happen at only finitely many stages, and that eventually e remains coded in $\mathcal{M}$ via the same fixed pair a^e and b^e.

The second type of requirement will guarantee that $f = \lim_s f_s$ exists and is an isomorphism from $\mathcal{L}$ to $\mathcal{M}$. To meet these, a *restraint function*, $r(e, s)$, defined in the construction ensures f_s and the collection of successor pairs do not change on elements $<_{\mathcal{L}} r(e, s)$ as a result of action taken to code $e' > e$ into $\mathcal{M}$.

Construction.

Stage 0. Let $|\mathcal{M}| = \{0\}$, $<_{\mathcal{M}} = \emptyset$, and $f_0(0) = 0$. For all e, initialize $r(e, 0) = e$, and set $a^e_0 = b^e_0 = -1$ so that e is not coded in $\mathcal{M}$ at stage 0.

Stage $s + 1$. We begin this stage with $(\{0, \ldots, s\}, <_{\mathcal{M}})$ and f_s from the previous stage.

Our first action will be to define $(\{0, \ldots, s + 1\}, <_{\mathcal{M}})$, extending the ordering defined in stage s. Simultaneously, we define f_{s+1}, which need not be an extension of f_s. The next task will be to update the values a^e_{s+1}, b^e_{s+1}, and $r(e, s + 1)$ to the current stage for all $e \in \mathbb{N}$.

Find the least $e \leq s$ that is not coded in $\mathcal{M}$ at stage s for which there is a $d \leq s$ satisfying the following conditions:

- $g_{s+1}(e,d) = \langle l_i^s, l_{i+1}^s \rangle$ for some $i+1 < s$,
- $r(e,s) <_{\mathcal{L}} l_i^s$,
- $r(e,s) <_{\mathcal{L}} s+1$, and
- e is not coded in at stage s.

If there is such an e, take the least suitable d (and the corresponding i).

Now define $(\{0,\ldots,s+1\}, <_{\mathcal{M}})$ and f_{s+1} in two cases.

Case 1. There are three instances of this case:

(i) there is no such e, or

(ii) there is such an $e \notin A_{s+1}$, or

(iii) there is such an $e \in A_{s+1}$, and $a_s^e = -1$.

If any of the above holds and $l_m^s <_L s+1 <_L l_{m+1}^s$, declare $r_m^s <_{\mathcal{M}} s+1 <_{\mathcal{M}} r_{m+1}^s$. Note that if (ii) or (iii) holds, $m \neq i$ because for some n, $g_{s+1}(e,d) = c_n^{s+1} = \langle l_i^s, l_{i+1}^s \rangle$, and so this pair is not in $\overline{C_{s+1}}$. Since both $l_i^s \leq s$ and $l_{i+1}^s \leq s$, it must be that there is no $u \leq s+1$ between them; in particular, $s+1$ is not between them.

Let f_{s+1} be the unique isomorphism between $(\{0,\ldots,s+1\}, <_{\mathcal{L}})$ and $(\{0,\ldots,s+1\}, <_{\mathcal{M}})$. (That is, $f_{s+1}(s+1) = s+1$ and $f_{s+1}(x) = f_s(x)$ for all $x \leq s$.)

Case 2. If there is such an $e \in A_{s+1}$ (satisfying the bulleted conditions above), and $a_s^e \neq -1$.

Let $a_s^e <_{\mathcal{M}} s+1 <_{\mathcal{M}} b_s^e$. (In this case, the construction will ensure that $r(e,s) <_{\mathcal{L}} f_s^{-1}(a_s^e)$. More on this below.)

Let f_{s+1} be the unique isomorphism between $(\{0,\ldots,s+1\}, <_{\mathcal{L}})$ and $(\{0,\ldots,s+1\}, <_{\mathcal{M}})$. (Hence $f_{s+1}(x) = f_s(x)$ for all $x \leq_{\mathcal{L}} r(e,s)$, since $r(e,s) <_{\mathcal{L}} s+1$ and $r(e,s) <_{\mathcal{L}} f_s^{-1}(a_s^e)$.)

Now, we can define $a_{s+1}^{e'}$ and $b_{s+1}^{e'}$ for all $e' \in \mathbb{N}$ as follows:

1. If no e satisfied the conditions, set $a_{s+1}^{e'} = a_s^{e'}$ and $b_{s+1}^{e'} = b_s^{e'}$ for all $e' \in \mathbb{N}$.

2. If there was an e that satisfied the conditions, let $a_{s+1}^{e'} = a_s^{e'}$ and $b_{s+1}^{e'} = b_s^{e'}$ for all $e' < e$. For $e' > e$, set $a_{s+1}^{e'} = b_{s+1}^{e'} = -1$. (It is this detaching of markers that ensures that the parenthetical remarks in Case 2 above hold.)

   For e itself, recall that $g_{s+1}(e,d) = (l_i^s, l_{i+1}^s)$ for some $i+1 < s$, and make the following assignments:

   (a) If $e \notin A_{s+1}$, set $a_{s+1}^e = r_i^s$ and $b_{s+1}^e = r_{i+1}^s$;

   (b) If $e \in A_{s+1}$ and $a_s^e = -1$, let $a_{s+1}^e = r_i^s$ and $b_{s+1}^e = r_{i+2}^s$;

   (c) If $e \in A_{s+1}$ and $a_s^e \neq -1$, let $a_{s+1}^e = a_s^e$ and $b_{s+1}^e = b_s^e$.

All that remains is to update the restraint function. For all $e' \in \mathbb{N}$, we let $r(e', s+1)$ be the $<_{\mathcal{L}}$-maximal element of

$$\{0,\ldots,e'\} \cup \{f_{s+1}^{-1}(i) \mid i \leq e'\} \cup \{f_{s+1}^{-1}(a_{s+1}^x), f_{s+1}^{-1}(b_{s+1}^x) \mid x < e' \wedge a_{s+1}^x \neq -1\}.$$

This completes the construction.

Because the relative positions of elements of $\mathcal{M}$ never change, $\mathcal{M} = (\mathbb{N}, <_{\mathcal{M}})$ is a computable linear ordering.

Remark 2.3.2. *It is easy to see that if action is taken to code e into $\mathcal{M}$ at stage $s+1$, then $f_{s+1}(x) = f_s(x)$ for every $x \leq_{\mathcal{L}} r(e,s)$, and if no coding takes place at this stage, this is true for all $x \leq s$. We also note here that $r(e,s)$ is increasing in e for each s.*

We now give (and establish some basic facts about) a function $\mathfrak{s} : \mathbb{N} \to \mathbb{N}$, that will be used in the lemmas that follow. Inductively define the function $\mathfrak{s}(e)$ to be the least s so that $s \geq \max(\{\mathfrak{s}(e') \mid e' < e\})$ and e is coded into $\mathcal{M}$ for all stages $s' \geq s$. The maximum of the empty set is taken to be zero.

Lemma 2.3.3. *The function $\mathfrak{s}$ exists and $\mathfrak{s} \leq_T A$.*

Proof. Inductively assume that $\mathfrak{s}(e')$ exists for $e' < e$, and that it can be computed from A.

Let $s_e = \max\{\mathfrak{s}(e') \mid e' < e\}$. By hypothesis, all $e' < e$ are coded in at stage s_e and at all subsequent stages. Thus, $a^{e'} = \lim_s a_s^{e'} = a_{s_e}^{e'}$ and $b^{e'} = \lim_s b_s^{e'} = b_{s_e}^{e'}$, so $r(e) = \lim_s r(e,s) = r(e,s_e)$ as a consequence of Remark 2.3.2 and the definition of $r(e,s)$. At s_e, e is *not* coded in $\mathcal{M}$. Both $a_{s_e}^e$ and $b_{s_e}^e$ have been reset to -1. (It follows from the definition of $\mathfrak{s}$ that s_e will be the stage where $e-1$ gets coded into $\mathcal{M}$ for the last time.)

Suppose $e \in A$. Let $s_1 \geq s_e$ be the least s so that $e \in A_s$. Let $s_2 \geq s_1$ be the least stage so that there is a $d \leq s_2$ with $g_{s_2+1}(e,d) = \langle l_i^{s_2}, l_{i+1}^{s_2} \rangle$ where $i+1 < s_2$, $r(e,s_2) = r(e) <_{\mathcal{L}} l_i^{s_2}$, and $r(e) <_{\mathcal{L}} s_2 + 1$. Because $\mathcal{L}$ satisfies condition (U), and the function g_s converges to true successor pairs of $\mathcal{L}$, s_2 must exist. We have $\mathfrak{s}(e) = s_2 + 1$.

If $e \notin A$, let $s_1 \geq s_e$ be the least stage so that there is a $d \leq s_1$ with $g_{s_1+1}(e,d) = g(e,d) = \langle a, b\rangle$ with $a, b \leq s_1$, and $b <_{\mathcal{L}} l_{s_1}^{s_1}$, and $r(e) <_{\mathcal{L}} a$. Then if there is a stage $s \geq s_1$ at which e is not coded in $\mathcal{M}$, it will be coded in at the first stage $t > s$ having $r(e) <_{\mathcal{L}} t$, which must occur since $\mathcal{L}$ satisfies condition (U). Clearly, it will remain coded in $\mathcal{M}$ at all subsequent stages. Thus, $\mathfrak{s}(e)$ exists.

We proceed to compute $\mathfrak{s}(e)$ from A for $e \notin A$. Recall that $Succ_{\mathcal{L}} \leq_T A$. Using $Succ_{\mathcal{L}}$ we can find stage s_1 described above. If e is not coded in $\mathcal{M}$ at this stage, find the least $t > s_1$ having $r(e) <_{\mathcal{L}} t$. Then $\mathfrak{s}(e) = t$.

If e is coded in at s_1, find the stage s, $s_e < s \leq s_1$, where it became coded in $\mathcal{M}$. If $(f_{s_1}^{-1}(a_{s_1}^e), f_{s_1}^{-1}(b_{s_1}^e)) \in Succ_{\mathcal{L}} \leq_T A$, then $\mathfrak{s}(e) = s$. Otherwise, there must be some element, n, $<_{\mathcal{L}}$-between $f_{s_1}^{-1}(a_{s_1}^e)$ and $f_{s_1}^{-1}(b_{s_1}^e)$, and necessarily $n > s_1$. At stage n, this e will become not coded in $\mathcal{M}$. (The restraints $r(e', s)$ for all $e' \geq e$ $<_{\mathcal{L}}$-exceeds n, so at this stage the construction will be in Case 1.) Find the least stage $t > n$ with $r(e) <_{\mathcal{L}} t$, and we will have $\mathfrak{s}(e) = t$.

Thus we see that $\mathfrak{s}$ is well-defined and that it may be computed from A, since A computes $Succ_{\mathcal{L}}$ by assumption.

□

In many ways, $\mathfrak{s}$ acts as a modulus of convergence. We now exploit this property to show that the partial functions, f_s, do indeed converge to a limit that is an isomorphism. The relative complexity of the isomorphism is also established.

Lemma 2.3.4. *The limit, $f = \lim_s f_s$, exists and is an isomorphism from $\mathcal{L}$ to $\mathcal{M}$. Furthermore, $f \leq_T \mathfrak{s}$.*

Proof. From Lemma 2.3.3 and Remark 2.3.2 it follows that for all $s' \geq \mathfrak{s}(e)$, we will have $f_{s'}(e) = f_{\mathfrak{s}(e)}(e)$ and $f_{s'}^{-1}(e) = f_{\mathfrak{s}(e)}^{-1}(e)$, since $e \leq_{\mathcal{L}} r(e,s') = r(e)$ and $f_{s'}^{-1}(e) \leq_{\mathcal{L}} r(e)$. Therefore, $\lim_s f_s(e)$ and $\lim_s f_s^{-1}(e)$ exists for all $e \in \mathbb{N}$, and $f \leq_T \mathfrak{s}$.

Since f_s is an isomorphism from $(\{0,\ldots,s\}, <_{\mathcal{L}})$ onto $(\{0,\ldots,s\}, <_{\mathcal{M}})$ and converges, its limit, $f = \lim_s f_s$, is also an isomorphism from $\mathcal{L}$ onto $\mathcal{M}$.

□

Finally, we are able to complete the proof of the theorem by showing that $Succ_{\mathcal{M}}$ has the same Turing degree as the set A.

Lemma 2.3.5. *$Succ_{\mathcal{M}} \equiv_T A$.*

Proof. We begin by establishing the reduction $Succ_{\mathcal{M}} \leq_T A$. From Lemma 2.3.3 we have that $\mathfrak{s} \leq_T A$. From Lemma 2.3.4, we have $f \leq_T \mathfrak{s}$. Clearly $Succ_{\mathcal{M}} \leq_T f \oplus Succ_{\mathcal{L}}$, and since each of f and $Succ_{\mathcal{L}}$ is Turing reducible to A, we see that $Succ_{\mathcal{M}} \leq_T A$ as well.

To see that A can be computed from $Succ_{\mathcal{M}}$, we show that $Succ_{\mathcal{M}}$ computes $\mathfrak{s}$. Then, recalling our original coding strategy and properties of $\mathfrak{s}$, i.e.,

$$e \in A \iff (a^e_{\mathfrak{s}(e)}, b^e_{\mathfrak{s}(e)}) \notin Succ_{\mathcal{M}},$$

we can finish the proof by observing that

$$A \leq_T Succ_{\mathcal{M}} \oplus \mathfrak{s} \leq_T Succ_{\mathcal{M}}.$$

It remains only to prove that $\mathfrak{s}$ can be computed from $Succ_{\mathcal{M}}$. For some $e \in \mathbb{N}$, inductively assume $Succ_{\mathcal{M}}$ computes $s_e = \max\{\mathfrak{s}(e') | e' < e\}$. Simultaneously attempt to compute s^e_{in}

and s^e_{out}, defined to be

$$s^e_{in} \equiv_{def} \text{ the least } s > s_e \text{ such that } e \in A_s, r(e) <_{\mathcal{L}} s+1, \text{ and}$$
$$(\exists d \leq s)[g_{s+1}(e,d) = \langle l^s_i, l^s_{i+1}\rangle \wedge (r(e) <_{\mathcal{L}} l^s_i) \wedge (i+1 < s)]; \text{ and}$$

$$s^e_{out} \equiv_{def} \text{ the least } s > s_e \text{ such that } (a^e_s, b^e_s) \in Succ_{\mathcal{M}}.$$

If s^e_{in} exists, $\mathfrak{s}(e) \leq s^e_{in}$, and we can calculate $\mathfrak{s}(e)$ since we have this upper bound. In this case, of course, $e \in A$. If s^e_{out} exists, $\mathfrak{s}(e) = s^e_{out}$ and $e \notin A$. Because $\mathfrak{s}(e)$ exists by Lemma 2.3.3, this search must halt. So using oracle $Succ_{\mathcal{M}}$ we can recursively compute the function $\mathfrak{s}(e)$, thus $\mathfrak{s} \leq_T Succ_{\mathcal{M}}$, and the proof of the lemma is complete.

□

This completes the proof of the theorem.

□

The result in Theorem 2.3.1 applies to a somewhat broader class of linear orderings than just those satisfying condition (U). First, for any linear ordering $\mathcal{L}$, the degree spectrum of the successor relation in $\mathcal{L}$ will be identical to that of the reverse ordering, $\mathcal{L}^*$. So a descending sequence of successor pairs satisfying a symmetric condition (U^*) is similarly sufficient.

Additionally, suppose that $\mathcal{L}$ is a computable linear ordering in which (U) does not hold. Then $\mathcal{L}$ may be decomposed as $\mathcal{L} = \mathcal{L}_2 + \mathcal{L}_1$, where $\mathcal{L}_1$ has order type 1 or $1 + \eta$. The ordering $\mathcal{L}_2$ is an initial segment of $\mathcal{L}$ and is computable since it is definable with a quantifier-free formula with finitely many parameters (in fact, a single parameter suf-

fices). Its successor relation is at most finitely different from that of $\mathcal{L}$. Consequently, $DgSp_{\mathcal{L}}(Succ_{\mathcal{L}}) = DgSp_{\mathcal{L}_2}(Succ_{\mathcal{L}_2})$, and if $\mathcal{L}_2$ satisfies (U), $DgSp_{\mathcal{L}}(Succ_{\mathcal{L}})$ will be closed upward in the c.e. degrees.

This process may be iterated any finite number of times to obtain a computable initial segment $\mathcal{M}$ of $\mathcal{L}$ with $DgSp_{\mathcal{L}}(Succ_{\mathcal{L}}) = DgSp_{\mathcal{M}}(Succ_{\mathcal{M}})$. If $\mathcal{M}$ satisfies (U), then $DgSp_{\mathcal{M}}(Succ_{\mathcal{M}})$, and hence $DgSp_{\mathcal{L}}(Succ_{\mathcal{L}})$, will be closed upward in the c.e. degrees.

If this decomposition process continues *ad infinitum*, we cannot apply Theorem 2.3.1. We characterize these types of linear orderings (R_1, R_2, R_3, and R_4) in the following corollary.

Corollary 2.3.6 ([9])**.** *Let M be a computable linear ordering with infinitely many successor pairs. If M is not of the form R_i for $1 \leq i \leq 4$, where R_i is given below, then $DgSp_M(Succ_M)$ is closed upward in the c.e. degrees.*

$R_1 = F_1 + \omega + R + \omega^* + F_2$,

$R_2 = n_0 + \eta + n_1 + \eta + \cdots + R + \omega^* + F_2$,

$R_3 = F_1 + \omega + R + \cdots + \eta + n_1' + \eta + n_0'$,

$R_4 = n_0 + \eta + n_1 + \eta + \cdots + R + \cdots + \eta + n_1' + \eta + n_0'$,

where F_1 and F_2 are arbitrary (possibly empty) linear orderings with finitely many successor pairs, $n_i, n_i' \geq 1$ are finite blocks of the respective size, and R may be any countable linear ordering.

Proof. Let $\mathcal{L}$ be a computable linear ordering for which (U) does not hold, and that $\mathcal{L}$ cannot be decomposed via finitely many iterations of the process described above to a sum $\mathcal{L} = \mathcal{M} + L_n + \ldots + L_1$ where $\mathcal{M}$ is a computable initial segment of $\mathcal{L}$ satisfying (U).

We decompose $\mathcal{L}$ as described above to obtain

$$\mathcal{L} = R + \ldots + L_n + \ldots + L_2 + L_1,$$

where each $\mathcal{L}_i$ is of type 1 or $1 + \eta$.

Case 1. If for some k, and all $i > k$, L_i is of type 1, then $R + \ldots + L_n + \ldots + L_{k+1}$ is of type $R + \omega^*$. The remaining part of the decomposition, $F = L_k + L_{k-1} + \ldots + L_2 + L_1$ is a finite sum of orderings of type 1 or $1 + \eta$, and thus F can have only finitely many successor pairs. In this case, we have

$$\mathcal{L} = R + \omega^* + F,$$

where F is a linear ordering having finitely many successor pairs.

Case 2. If Case 1 does not hold, then for each k, there is $j > k$ so that L_j is of type $1 + \eta$. Hence, at most finitely many blocks of type 1 may appear consecutively. If n such blocks appear, they may be represented as a single block, n. In this case, we have

$$\mathcal{L} = R + \ldots + \eta + n_1 + \eta + n_0,$$

where the n_i's are appropriate finite blocks.

Recalling that for any linear ordering $\mathcal{L}$, $DgSp_{\mathcal{L}}(Succ_{\mathcal{L}}) = DgSp_{L^*}(Succ_{L^*})$, we arrive at the four decompositions above. □

We end the section with a nice consequence (Theorem 2.3.8) of Theorems 2.3.1 and 2.3.7 (the latter below). Earlier in this chapter, we mentioned an example of a linear ordering in

which the successor relation has degree $\mathbf{0}'$ in every computable copy [22]. Its existence is a consequence of the following theorem.

Theorem 2.3.7 ([22])**.** *For any non-computable c.e. set C, there is a computable linear ordering $\mathcal{L}$ such that $Succ_{\mathcal{L}} \equiv_T C$ and $C \leq_T Succ_{\mathcal{L}'}$ for every computable linear ordering $\mathcal{L}' \cong \mathcal{L}$. Furthermore, $\mathcal{L}$ has the form*

$$\mathcal{L} = \mathcal{I}_0 + \mathcal{L}_0 + \mathcal{I}_1 + \mathcal{L}_1 + \ldots,$$

where each $\mathcal{I}_i$ is a block of length $i+3$ and $\mathcal{L}_i$ has order type η or $(\eta + 2 + \eta) \cdot \tau$ for some τ.

In Theorem 2.3.7, the witnessing computable linear ordering $\mathcal{L}$ is constructed so that $DgSp_L(Succ_L) \subseteq \mathcal{R}(\geq deg(C))$, where $\mathcal{R}(\geq deg(C))$ is the collection of all c.e. Turing degrees $\geq deg(C)$. Because of their form, the orderings satisfy the condition (U) in Theorem 2.3.1, and we have the following.

Theorem 2.3.8 ([9])**.** *For any c.e. degree $\mathbf{a}$, there is a linear ordering $\mathcal{L}$ so that the degree spectrum of $Succ_{\mathcal{L}}$ is exactly the upper cone of c.e degrees determined by $\mathbf{a}$, that is, $DgSp_{\mathcal{L}}(Succ_{\mathcal{L}}) = \mathcal{R}(\geq \mathbf{a})$.*

Proof. If $\mathbf{a}$ is computable then the result follows from Example 2.2.1. Let $\mathbf{a}$ be a non-computable c.e. degree. Theorem 2.3.7 yields a linear ordering $\mathcal{L}$ with $deg(Succ_{\mathcal{L}}) = \mathbf{a}$ that satisfies condition (U) of Theorem 2.3.1, and ensures that $DgSp_{\mathcal{L}}(Succ_{\mathcal{L}})$ is contained in the cone above $Succ_{\mathcal{L}}$. Theorem 2.3.1 says that $DgSp_{\mathcal{L}}(Succ_{\mathcal{L}})$ contains that cone. □

Thus, every upper cone of c.e. degrees can be realized as the degree spectrum of the successor relation in the computable copies of some linear ordering.

2.4 Concluding remarks and proposed problems

Research continues in the study of the degree spectrum of the successor relation. Downey, Lempp, and Wu [18] have recently shown that, provided the successor relation is infinite, the degree spectrum of successor always contains $\mathbf{0}'$.

Many questions remain in the path to a complete understanding of these spectra and the possibilities surrounding them. For example,

1. When the spectrum is closed upwards, is it *necessarily* a cone? A filter?

2. If not, what is possible? Can, for example, the spectrum be the union of finitely many cones with incomparable roots?

Chapter 3

Colored trees

3.1 Motivation, and some basic definitions and facts

In this chapter, we will analyze the algorithmic complexity of a version of Ramsey's theorem adapted to trees.

For a set of natural numbers A, let $[A]^n$ be the collection of (unordered) n-element subsets of A. Recall that Ramsey's theorem for a fixed n and k is the following.

Theorem 3.1.1 (RT^n_k)**.** *For any $f : [\mathbb{N}]^n \to \{0, \ldots, k-1\}$ there exists an infinite subset $A \subseteq \mathbb{N}$ so that $f \upharpoonright [A]^n$ is constant.*

The set A is a *monochromatic* or *homogeneous* subset for the coloring function f. In 1971, Specker [58] demonstrated that the effective version of Ramsey's theorem does not hold, i.e., that there are computable colorings for which there exist no monochromatic subset. In 1972, Carl Jockusch ([36]) considered the complexity of the monochromatic subset and described the difficulty involved in extracting such a set when the coloring function is computable in

terms of the arithmetical hierarchy of sets. He showed that if a coloring function f of n-element subsets of $\mathbb{N}$ is a computable one, then there is a Π^0_n subset of the natural numbers, A, so that $[A]^n$ is monochromatic for f. He thus gives an upper bound on the complexity of the requisite homogeneous subset. Furthermore, for $n \geq 2$, Jockusch gives a construction of a computable coloring of n-element sets for which there is no Σ^0_n monochromatic subset, thus demonstrating that the bound is sharp.

Subsequently, a great deal of work has been done in attempts to assess the relative axiomatic content of a variety of combinatorial theorems in the program of *reverse mathematics* established by Friedman and Simpson [50]. Certain instances and formulations of Ramsey's theorem have been unyielding to analysis in terms of the usual hierarchy of systems of arithmetic (the Big Five) studied in this program (see, for example, [8]). In particular, exact description of the axiomatic content of Ramsey's theorem for $n = 2$ remains elusive. The full version of Ramsey's theorem, "For all n and all k, RT^n_k holds," is known to be equivalent (over RCA_0) to ACA_0 plus the statement that for any set A and natural number n, the nth jump of A, $A^{(n)}$, exists [50]. (This is true in the usual model of ACA_0, but not provable. These axiom systems will be described in Section 3.3.)

The computable content, and especially the reverse mathematics, of classical combinatorial theorems, and their adaptations have been extensively studied, for example [10, 1, 7, 8, 33, 34, 35, 36, 48]. Trees are common structures that are relevant throughout mathematics, and adaptations of Ramsey's theorem to trees have been introduced in other contexts, for example, in [24, 40, 59].

Here we consider an adaptation of Ramsey's theorem to trees that may lead to insight into the reverse mathematics of Ramsey's theorem itself. In particular, we assess the algorithmic

complexity of the requisite homogeneous substructure in terms of the arithmetical hierarchy of sets. As we have already mentioned, a number of tree versions of Ramsey's theorem appear in the literature (see, for example, [24], [40], or [59]), and from some of them may be derived a restricted version of Theorem 3.2.1 with n fixed as 1. Furthermore, in those cases a more restrictive definition of isomorphism is used.

3.2 Notation and basic observations

Let $\mathcal{T}$ be a partial ordering. We denote by $[\mathcal{T}]^n$ the collection of chains of length n (n-chains) in $\mathcal{T}$. (Note that $[\mathcal{T}]^1 = \mathcal{T}$.) The collection of all finite strings of 0's and 1's is $2^{<\omega}$ and for the usual tree partial ordering of this set by initial segments we write $\sqsubset$. (In other words, for strings σ and τ, we write $\sigma \sqsubset \tau$ if σ is an initial segment of τ.)

A (k, n)-*colored binary tree* $\mathcal{T}$ is a partial ordering of $|\mathcal{T}| \subset \omega$ together with function $f : [\mathcal{T}]^n \to \{0, \ldots, k\}$ and bijection $g : |\mathcal{T}| \to 2^{<\omega}$ satisfying

$$x <_{\mathcal{T}} y \iff g(x) \sqsubset g(y).$$

Such a tree is computable if $|\mathcal{T}|$ is a computable set, and f and g are computable functions. The function f gives a coloring of the n-chains by finitely many colors.

A tree S is a *subtree of tree* $\mathcal{T}$ *isomorphic to* $2^{<\omega}$ if $|S| \subseteq |\mathcal{T}|$ and

1. S has a least element under the ordering induced by $\mathcal{T}$, and

2. each element of S has exactly two immediate successors under the induced ordering.

If a subtree of $\mathcal{T}$ is the collection of all nodes above some given node τ, we call it the *full*

subtree above τ.

Without loss of generality, when a tree is computable we assume it is the full binary tree, $2^{<\omega}$. Subtrees of such a tree will be identified with the corresponding subset of nodes and induced ordering. A subtree of a colored tree is *monochromatic for* f if the coloring function f is constant on the subset of nodes.

Now we may state the adaptation of Ramsey's theorem to trees that we will consider below.

Theorem 3.2.1 ([10])**.** *(TT^n_k) Let* $f : [2^{<\omega}]^n \to \{0, \ldots, k-1\}$ *be a* (k, n)*-coloring of the binary tree. Then there exists a subtree* S *of* $2^{<\omega}$ *that is isomorphic to the full binary tree, so that* $f \restriction [S]^n$ *is constant. In other words,* S *is a monochromatic subtree for* f*.*

Later we will consider the algorithmic complexity of the subtree, and will make this assessment in terms of the arithmetical hierarchy of sets. Recall that a set of natural numbers is Σ^0_n (Π^0_n) if it is definable in the language of first order arithmetic by a formula having n alternating quantifier blocks beginning with $\exists$ ($\forall$). A set that is computable in $0^{(n)}$ is in the intersection of Σ^0_n and Π^0_n, and is called a Δ^0_n set. A set that is Σ^0_{n+1} is enumerable in $0^{(n)}$. (See [52] for details.)

First, we will see some properties of this theorem from the perspective of *reverse mathematics*.

3.3 Reverse Mathematics

Broadly, the goal of the program of reverse mathematics is to assess the axiomatic content of theorems (expressed in the language of second order arithmetic) relative to a weak base

system (RCA_0) by showing equivalence to one of a number of standard axioms or axiom schemes that may be added to RCA_0 to obtain a larger fragment of the full second order arithmetic. Since we are working in second order arithmetic, our language is two-sorted, with variables for both natural numbers, and sets of natural numbers.

The system, RCA_0, (the acronym RCA is an abbreviation for *recursive comprehension axiom*) has a first order part described by the same collection of axioms as *Peano Arithmetic*, except that it lacks full induction. In RCA_0, induction is limited and holds only for Σ^0_1 formulas. In addition, there is a second order comprehension scheme for Δ^0_1 formulas (this scheme asserts that all computable sets exist in any model of RCA_0). If a theorem is provable in RCA_0, then an effective version of that theorem holds. This is a consequence of the fact that there is a minimal ω-model of RCA_0, and it is our usual arithmetic, except that only the computable sets provably exist.

The system ACA_0 (for *arithmetical comprehension axiom*) consists of the axioms of RCA_0 together with a comprehension scheme for sets definable by arithmetical formulas (i.e., by a Σ^0_n formula for some n). ACA_0 also has a minimal ω-model, one consisting of only the arithmetical sets. When a theorem ψ is provable in ACA_0, and at the same time, the theorem ψ together with the axioms of RCA_0 prove the existence of arithmetical sets (i.e., prove arithmetical comprehension itself), then, in light of such a reversal, we can say that ψ is equivalent to ACA_0 over RCA_0. When this is the case, we can conclude that no effective version of ψ can hold, since a model necessarily contains non-computable sets.

In the case of Ramsey's theorem for $n = 1$ (the pigeonhole principle for infinite sets), it is a simple matter to compute the subset of $\mathbb{N}$ on which a computable coloring function takes a particular value. Determining which component of the partition is infinite is a more

complicated question. In our case, we are searching not only an infinite component of the coloring partition, but one exhibiting a particular *structure*—a binary subtree.

We begin with a basic case for $n = 1$ and $k = 2$.

Lemma 3.3.1 ([10])**.** *(RCA_0) Let $f : 2^{<\omega} \to \{\mathsf{red}, \mathsf{blue}\}$ be a $(2, 1)$-coloring of the full binary tree. For any node σ of the tree either, (1) there is subtree isomorphic to $2^{<\omega}$ with root above σ and in which every node (except possibly the root) is colored* red*, or (2) there is a $\tau \supset \sigma$ so that every node above τ is colored blue by f (i.e., a full subtree above some τ).*

Proof. Given f and σ, we may make a systematic search for a subtree, S.

Construction.

Stage 0. Find the least node $\tau_{\langle\rangle} \sqsupseteq \sigma$ for which $f(\tau_{\langle\rangle}) = \mathsf{red}$. Add this node to S.

Stage s+1. For each node τ_α of length s already in S, find the least pair of incomparable nodes above τ that are both colored red. Call these $\tau_{\alpha^\frown 0}$ and $\tau_{\alpha^\frown 1}$, and add them to S.

This ends the construction.

If the search ever fails, we have found a full blue subtree and realized (2) in the statement of the Lemma. If it succeeds, we have a subcollection of nodes, all colored red by f, and the indices of the markers assigned to them provide the isomorphism to $2^{<\omega}$.

Note that if f is computable, the given algorithm is computable in $0'$. (We would need $0'$ to tell us that the "least incomparable extensions" that we might be looking for do not exist.) □

We will refer to the red subtree constructed in Lemma 3.3.1 (when it exists) as the *standard* red *subtree for f above σ*. For any node σ the collection of all nodes extending it is the *full subtree above σ*.

As commented in the proof, $0'$ is a sufficiently strong oracle to execute this construction when f is computable. We will see below that for the case when $n = 1$ there is always a computable monochromatic tree, though it is not calculable in a uniform way. In the instance where the search in the proof of Lemma 3.3.1 fails, there is a $\tau \sqsupseteq \sigma$ for which all extending nodes are colored blue, so this monochromatic subtree, $\mathcal{T}_\tau$ is computable. If the algorithm never encounters a full blue subtree, the search succeeds, and a red subtree will be enumerated. We can do better if we are willing to sacrifice uniformity.

Theorem 3.3.2. *If $f : 2^{<\omega} \to \{\mathsf{red}, \mathsf{blue}\}$ is a computable $(2,1)$-coloring for which there exist no* full *(so necessarily computable) monochromatic subtrees, then there is a computable (but not full) monochromatic subtree for f that is isomorphic to the full binary tree.*

Proof. If there is no full monochromatic subtree of any color, then for any τ there must exist an incomparable pair of red (and blue, for that matter) nodes extending it. We enumerate the monochromatic subtree S in the following algorithm, ensuring it is isomorphic to $2^{<\omega}$.

Construction.

Stage 0. Attach the name $\sigma_{\langle\rangle}$ to the least node colored red by f.

Stage s+1. We begin this stage with a collection of leaves, σ_τ for all $\tau \in 2^s$. For each σ_τ find the least pair of incomparable red nodes extending σ_τ and attach names $\sigma_{\tau^\frown 0}$ and $\sigma_{\tau^\frown 1}$ to them. (Note that this systematic search is guaranteed to halt by the observation made just before the beginning of the construction.)

Let $S = \{\sigma_\tau \mid \tau \in 2^{<\omega}\}$.

This ends the construction.

The indices provide the isomorphism, and S is clearly c.e. To see that S is computable,

note that we can determine if node $\alpha \in 2^{<\omega}$ is an element of S by executing the construction above for $|\alpha|$ steps, where $|\alpha|$ denotes the length of the node. If α is not in S at that stage, it will never enter. $\square$

These computable subtrees exist when the coloring function is computable and the proof requires only Σ^0_1 induction, so they exist in RCA_0. Thus Lemma 3.3.1 holds in RCA_0.

We now generalize from two to k colors. The proof uses induction on Σ^0_2 formulas, and consequently, the statement of the theorem refers to $RCA_0 + \Sigma^0_2\text{-}IND$.

Theorem 3.3.3 ([10])**.** *($RCA_0 + \Sigma^0_2\text{-}IND$) Let $f : 2^{<\omega} \to \{0, \ldots k-1\}$. Then there is subtree S that is isomorphic to $2^{<\omega}$ and monochromatic for f.*

Proof. Define a set of colors $C = \{j \mid (\exists\sigma)(\forall\tau \sqsupset \sigma)[j \leq f(\tau)]\}$. This set is non-empty ($0 \in C$) and finite (it is a subset of $\{0, \ldots, k-1\}$), so it has a maximum, j_M.

By the defining property, there is a node τ_M so that all nodes above τ_M are colored by colors $\geq j_M$. If $j_M = k-1$ then all nodes above τ_M have color j_M and there is a full monochromatic subtree.

If $j_M \lneq k-1$, then by maximality, $j_M + 1$ is not in C, and so above each node in $2^{<\omega}$ there is a node τ so that $j_M + 1 > f(\tau)$. In particular, if we consider nodes above τ_M, above each of these nodes there is a node with color $< j_M + 1$, but all have color $\geq j_M$. Hence above each node extending τ_M there is a node of color *exactly* j_M. From these we can construct a monochromatic subtree S isomorphic to $2^{<\omega}$ of color j_M.

Construction.

Stage 0. Let $\sigma_{\langle\rangle}$ be the least extension τ of τ_M having $f(\tau) = j_M$, and add $\sigma_{\langle\rangle}$ to S.

Stage $s+1$. For each leaf σ_α in S at the end of stage s (i.e., nodes with subscript α of

length s), let $\sigma_{\alpha^\frown 0}$ and $\sigma_{\alpha^\frown 1}$ be the least extensions of $\sigma_\alpha^\frown 0$ and $\sigma_\alpha^\frown 1$ of color j_M. (Existence of such nodes is guaranteed by maximality of j_M as discussed in the previous paragraph.) Add these nodes to S.

This ends the construction.

The isomorphism is provided by the indices. □

Note that as in the case with two colors, this tree is computable when f is (and j_M is given). Calculating j_M is not uniformly computable—in the proof above, the set C is defined by a Σ^0_2 statement.

Of course the infinite pigeonhole principle (RT^1_k) follows almost immediately from TT^1_k: Given f, a k-coloring of the natural numbers, define a $(1,2)$-coloring g on the tree $2^{<\omega}$ by setting $g(\sigma) = i$ when $g(|\sigma|) = i$, where $|\sigma|$ is the length of the node σ. Once a monochromatic tree is found, the collection of lengths of nodes appearing in that tree will be a monochromatic subset for f. (We will use this level coloring argument again later.)

It is known that the infinite pigeonhole principle is equivalent to the bounding principle $\mathsf{B}\Pi^0_1$ (see [34] or [8]) which is strictly weaker than Σ^0_2-IND. We conclude that the strength of TT^1_k lies between $\mathsf{B}\Pi^0_1$ and Σ^0_2-IND.

In this preliminary case when $n = 1$, we are able to guarantee the existence of a computable monochromatic substructure (though not in a uniform way, exactly as is the case for Ramsey's theorem for $n = 1$). (As we will see in the next section, when $n \geq 2$ the lowest guaranteed complexity is Π^0_n.)

We proceed with our assessment of the axiomatic strength of the the tree theorems, now proving the case when $n = 2$, which provides a base case for the inductive proof for larger n.

Theorem 3.3.4 ([10]). (ACA_0) *For all k, TT^2_k. That is, for any $(k,2)$-coloring of $2^{<\omega}$, there is a monochromatic subtree isomorphic to $2^{<\omega}$.*

We first prove the theorem for the case when $k = 2$, and will later extend the argument to k colors.

Proof for $k = 2$. Suppose $f : [2^{<\omega}]^2 \to \{\mathsf{red}, \mathsf{blue}\}$ is a $(2,2)$-coloring of the full binary tree. Given any $\sigma \in 2^{<\omega}$, we define a family of induced coloring maps on single nodes,

$$f_\sigma : \{\tau \in 2^{<\omega} \mid \tau \supset \sigma\} \to \{\mathsf{red}, \mathsf{blue}\},$$

by setting $f_\sigma(\tau) = f(\sigma, \tau)$.

Define p_σ, T_σ, and c_σ as follows. Set $p_{\langle\,\rangle} = \langle\,\rangle$ and $T_{\langle\,\rangle} = 2^{<\omega}$. Suppose p_σ and T_σ have been defined. If there is a full blue subtree of T_σ above p_σ using f_{p_σ}, then make the following assignments:

- $c_\sigma = \mathsf{blue}$.
- Let $p_{\sigma^\frown 0}$ and $p_{\sigma^\frown 1}$ be the first two incomparable nodes of the full blue subtree of T_σ above p_σ using f_{p_σ}.
- For $\varepsilon \in \{0,1\}$, set $T_{\sigma^\frown \varepsilon} = \{\tau \in T_\sigma \mid \tau \sqsupseteq p_{\sigma^\frown \varepsilon}\}$.

If there is no full blue subtree of T_σ above p_σ using f_{p_σ}, then make the following assignments:

- $c_\sigma = \mathsf{red}$.
- Let $p_{\sigma^\frown 0}$ and $p_{\sigma^\frown 1}$ be the first two incomparable nodes of the standard red tree of T_σ above p_σ using f_{p_σ}.

- For $\varepsilon \in \{0,1\}$, $T_{\sigma^\frown\varepsilon}$ consists of those nodes of the standard red tree of T_σ above p_σ using f_{p_σ} which extend $p_{\sigma^\frown\varepsilon}$.

Let $S = \{p_\sigma \mid \sigma \in 2^{<\omega}\}$. Since p_σ is arithmetically definable from the values of p_τ and c_τ for $\tau \sqsubset \sigma$, $\mathsf{ACA_0}$ proves the existence of S. By the construction, whenever $\sigma \sqsubset \tau \in 2^{<\omega}$, we have $f(p_\sigma, p_\tau) = c_\sigma$. The map $p_\sigma \mapsto c_\sigma$ is a $(2,1)$-coloring of S, and its existence is provable in $\mathsf{ACA_0}$. Since $\mathsf{ACA_0}$ implies Σ^0_2-IND, Theorem 3.3.3 holds in ACA_0, and we may apply it here to obtain a color c and a subtree T of S such that $p_\sigma \in T$ implies $c_\sigma = c$. Consequently, if $p_\sigma \sqsubset p_\tau$ are elements of T, then $f(p_\sigma, p_\tau) = c$, completing the proof. □

We now extend the result to $(k,2)$-colorings.

Proof of Theorem 3.3.4 for k colors. Suppose $f : [2^{<\omega}]^2 \to k$ is a finite coloring of pairs of comparable nodes of $2^{<\omega}$. Given any $\sigma \in 2^{<\omega}$, we define an induced map on single nodes $f_\sigma : \{\tau \in 2^{<\omega} \mid \tau \supset \sigma\} \to k$ by setting $f_\sigma(\tau) = f(\sigma, \tau)$.

Define p_σ, T_σ, and c_σ as follows. Set $p_{\langle\,\rangle} = \langle\,\rangle$ and $T_{\langle\,\rangle} = 2^{<\omega}$. Given p_σ and T_σ, define c_σ as follows. Let j be the least integer such that there is no $p \supset p_\sigma$ in T_σ such that $\forall \tau \in T_\sigma(\tau \supset p \to j < f_{p_\sigma}(\tau))$. Since j is the least such integer, there is a $p \supset p_\sigma$ in T_σ such that $\forall \tau \in T_\sigma(\tau \supset p \to j \leq f_{p_\sigma}(\tau))$. Fix this p, and note that by the definition of j, there is no $q \supset p$ in T_σ such that $\forall \tau \in T_\sigma(\tau \supset q \to j < f_{p_\sigma}(\tau))$. If we treat the color j as red and the colors greater than j as blue, by Theorem 3.3.1, the standard j-colored subtree of T_σ above p using f_{p_σ} exists and is isomorphic to $2^{<\omega}$. Call this tree T. Let $c_\sigma = j$. Let $p_{\sigma^\frown 0}$ and $p_{\sigma^\frown 1}$ be the two level one elements of T. For $\varepsilon \in \{0,1\}$, let $T_{\sigma^\frown\varepsilon}$ be the subtree of T with root $p_{\sigma^\frown\varepsilon}$.

Let $S = \{p_\sigma \mid \sigma \in 2^{<\omega}\}$. Since p_σ is arithmetically definable from the values of p_τ and c_τ

for $\tau \sqsubset \sigma$, $\mathsf{ACA_0}$ proves the existence of S. By the construction, whenever $\sigma \sqsubset \tau \in 2^{<\omega}$, we have $f(p_\sigma, p_\tau) = c_\sigma$. The map $p_\sigma \mapsto c_\sigma$ is a finite coloring of S, and its existence is provable in $\mathsf{ACA_0}$. Since $\mathsf{ACA_0}$ implies $\Sigma^0_2 - \mathsf{IND}$, an application of Theorem 3.3.3 yields a color c and a subtree T of S such that $p_\sigma \in T$ implies $c_\sigma = c$. Consequently, if p_σ and p_τ are nodes in T with $\sigma \sqsubset \tau$, then $p_\sigma \sqsubset p_\tau$ and $f(p_\sigma, p_\tau) = c$, completing the proof. □

We complete the proof of $\mathsf{TT}^{\mathsf{n}}_{\mathsf{k}}$ in $\mathsf{ACA_0}$ using the following inductive step. We abbreviate $\forall k(\mathsf{TT}^{\mathsf{n}}_{\mathsf{k}})$ by TT^{n}.

Theorem 3.3.5 ([10])**.** ($\mathsf{ACA_0}$) *For all $n \geq 1$, TT^{n} implies $\mathsf{TT}^{\mathsf{n+1}}$.*

Proof. We will generalize the proof of Theorem 3.3.4 to handle higher exponents by constructing a subtree S such that the color of any $(n+1)$-chain is determined by its first n elements, and applying TT^{n} to S to obtain the desired monochromatic tree. Suppose $f : [2^{<\omega}]^{n+1} \to k$ is a finite coloring of the $(n+1)$-chains in $2^{<\omega}$. If $P = \{p_\tau \mid \tau \sqsubseteq \sigma\}$ is the complete path of nodes leading to and terminating in p_σ, we define an induced coloring of single nodes $\tau \supset p_\sigma$ by setting

$$f_{p_\sigma}(\tau) = \prod_{\vec{m} \in [P]^n} \mathsf{pr}(\vec{m})^{f(\vec{m},\tau)}.$$

where if k is the integer code for the sequence $\vec{m}$, then $\mathsf{pr}(\vec{m})$ is the k^{th} prime. Note that f_{p_σ} uses no more than $k^{|P|^n}$ colors.

Define p_σ, T_σ, and c_σ as follows. Set $p_{\langle\,\rangle} = \langle\,\rangle$ and $T_{\langle\,\rangle} = 2^{<\omega}$. Given p_σ and T_σ, let c_σ be the greatest integer in the range of f_{p_σ} such that there is a $p \supset p_\sigma$ in T_σ such that

$$\forall \tau \in T_\sigma \ (\tau \supset p \to c_\sigma \leq f_{p_\sigma}(\tau)).$$

Fix the least such p, and note that the standard c_σ-colored subtree of T_σ above p using f_{p_σ} exists and is isomorphic to $2^{<\omega}$. Call this tree T. Let $p_{\sigma^\frown 0}$ and $p_{\sigma^\frown 1}$ be the two level one elements of T and let $T_{\sigma^\frown \varepsilon}$ be the subtree of T with root $p_{\sigma^\frown \varepsilon}$ for $\varepsilon \in \{0,1\}$.

Since p_σ is arithmetically definable from the values of p_τ and c_τ for $\tau \sqsubset \sigma$, $\mathsf{ACA_0}$ proves the existence of the tree $S = \{p_\sigma \mid \sigma \in 2^{<\omega}\}$. By the construction of S, given any increasing sequence of elements of S of the form

$$p_1 \sqsubset p_2 \sqsubset \cdots \sqsubset p_n \sqsubset p_{n+1} \sqsubset p_{n+2},$$

we have $f_{p_n}(p_{n+1}) = f_{p_n}(p_{n+2})$, and so $f(p_1, \ldots, p_n, p_{n+1}) = f(p_1, \ldots, p_n, p_{n+2})$. Consequently, the function $g : [S]^n \to k$ defined for $p_{\sigma_1} \sqsubset \cdots \sqsubset p_{\sigma_n}$ by

$$g(p_{\sigma_1}, \ldots, p_{\sigma_n}) = f(p_{\sigma_1}, \ldots, p_{\sigma_n}, p_{\sigma_n^\frown 0})$$

indicates the color of the $(n+1)$-chains extending $(p_{\sigma_1}, \ldots, p_{\sigma_n})$. By $\mathsf{TT^n}$ there is a subtree of S which is isomorphic to $2^{<\omega}$, monochromatic for g, and hence monochromatic for f. $\square$

We made use of ACA_0 in proving the previous theorem, and the following reversal shows that this was, in fact, unavoidable.

Theorem 3.3.6 ([10]). *For $n \geq 3$ and $k \geq 2$, $\mathsf{RCA_0}$ proves that the following are equivalent:*

(1) $\mathsf{ACA_0}$*,*

(2) $\mathsf{TT^n}$*, and*

(3) $\mathsf{TT^n_k}$*.*

Proof. Theorems 3.3.4 and 3.3.5 show that (1) implies (2). Since (3) is a special case of (2), it remains only to show that (3) implies (1).

Note that (3) implies Ramsey's theorem for n-element sets and two colors by another level coloring argument. Let f be a 2-coloring of $[\mathbb{N}]^n$. Define $\widetilde{f}$, a 2-coloring of $[2^{<\omega}]^n$, as follows. For an n-chain of nodes $\langle \sigma_1, \ldots, \sigma_n \rangle$, set $\widetilde{f}(\langle \sigma_1, \ldots, \sigma_n \rangle) = f(\{|\sigma_1|, \ldots, |\sigma_n|\})$, where $|\sigma|$ denotes the length of σ. From any monochromatic subtree for $\widetilde{f}$, we can construct an infinite monochromatic set for f. Whenever $n \geq 3$, Ramsey's theorem for n-tuples and two colors implies ACA_0 (see Lemma III.7.5 of [50]). This fact closes the circuit of implications. □

We have shown that TT^3_2 implies ACA_0, but the exact strength of TT^2 and TT^2_2 remain open. Using the level coloring argument of Theorem 3.3.6, it is easy to show that TT^2 implies Ramsey's theorem for pairs, but whether or not the converse is provable in RCA_0 is not known.

3.4 A Π^0_2 bound

We now turn to an analysis of the complexity of the monochromatic substructures for $n = 2$. Subsequently, we will use this as a base case for proving the Π^0_n bound for computable colorings of n-chains.

For the case when $n = 2$, we cannot expect to find a computable (or even c.e.) monochromatic substructure.

Theorem 3.4.1 ([10])**.** *If $n \geq 2$ then there is a computable 2-coloring of $[2^{<\omega}]^n$ with no Σ^0_n monochromatic subtree.*

Proof. Consider the following theorem due to Jockusch [36], and proceed with a level coloring argument.

Theorem 3.4.2 ([36])**.** *If $n \geq 2$ then there is a computable 2-coloring of $[\mathbb{N}]^n$ with no Σ^0_n monochromatic subset.*

Let f be a computable 2-coloring of $[\mathbb{N}]^n$ having no Σ^0_2 monochromatic subset as in Jockusch's theorem. Define $\widetilde{f}$, a 2-coloring of $[2^{<\omega}]^n$, as follows. For an n-chain of nodes $\langle \sigma_1, \ldots, \sigma_n \rangle$, set $\widetilde{f}(\langle \sigma_1, \ldots, \sigma_n \rangle) = f(\{|\sigma_1|, \ldots, |\sigma_n|\})$, where $|\sigma|$ denotes the length of σ.

Clearly $\widetilde{f}$ is a computable coloring of $[2^{<\omega}]^n$. Furthermore, if there is a Σ^0_n monochromatic substructure S for $\widetilde{f}$, the collection of natural numbers realized as the *lengths* of the nodes in S is a Σ^0_2 homogeneous subset for f, and we have a contradiction. □

Thus, we cannot expect a computable coloring of 2-chains to have a Σ^0_2 homogeneous substructure. In the following theorem we show that at worst, we will be able to find a Π^0_2 monochromatic substructure.

Theorem 3.4.3 ([10])**.** *Every computable finite coloring of pairs of comparable nodes of $2^{<\omega}$ has a Π^0_2 monochromatic subtree that is isomorphic to $2^{<\omega}$.*

Proof. We will carry out the proof for two colors, and then indicate how to extend the result to an arbitrary finite number of colors. Suppose $f : [2^{<\omega}]^2 \to \{\mathsf{red}, \mathsf{blue}\}$ is a computable two coloring of the pairs of comparable nodes of $2^{<\omega}$. Any computable monochromatic subtree would be Π^0_2 definable, so for the remainder of the proof we may assume no computable monochromatic subtree exists.

We will show that the complement of the desired monochromatic subtree is computably enumerable in $0'$ and then apply the strong hierarchy theorem [52]. Initially, we will need

to enumerate the complement of a tree S, and it is from this tree that we will extract our monochromatic subtree. This enumeration, computable from $0'$, will be built using movable markers.

Intuitively, the markers used in this proof eventually settle on nodes corresponding to nodes of S, above each of which S is monochromatic in a certain sense (described in the next paragraph). By making initial guesses at associated colors and allowing for later revisions, we can execute the construction using only a $0'$ oracle. The selection of the nodes is complicated by our goal of arranging for the monochromatic tree to be isomorphic to $2^{<\omega}$, especially when the color blue is assigned to a node.

For each $\alpha \in 2^{<\omega}$ we will have a marker p_α. We write p_α^s to indicate the location of p_α at stage s, and we will use the colors $c_\alpha^s \in \{\mathsf{red}, \mathsf{blue}\}$. At each stage, if $\alpha \sqsubset \beta$ and p_α^s and p_β^s are in use, we require $p_\alpha^s \sqsubset p_\beta^s$ and $f(p_\alpha^s, p_\beta^s) = c_\alpha^s$. We will frequently write $f_{p_\alpha}(p_\beta)$ for $f(p_\alpha, p_\beta)$.

At each stage s, we will also have a finite set M^s consisting of all $\alpha \in 2^{<\omega}$ such that p_α is in use, and a finite set E^s which is a correct initial segment of the complement of S. (M is for *map* and E is for *ejected*.) In selecting locations for newly introduced markers, we will be careful to avoid elements of E^s.

With each $\alpha \in 2^{<\omega}$ and stage s, we will associate a tree T_α^s of possible extensions of p_α^s. Just as we intend for p_α^s to converge to a node in S, we intend for T_α^s to converge to a tree isomorphic to $2^{<\omega}$. However, T_α^s may be a finite tree at some stages, due to erroneous selections. Regardless of the size of T_α^s, it is completely described by E^s together with a finite sequence of pairs called a *descriptor*.

Descriptors are defined inductively as follows. The sequence of no pairs, $\langle\ \rangle$, is a descriptor

for $2^{<\omega}$. Writing $d(T^s_\alpha)$ for the descriptor of T^s_α, if $p \in T^s_\alpha$ then $d(T^s_\alpha)^\frown(p, \mathsf{red})$ is the descriptor for the tree obtained by following the algorithm for constructing the standard red subtree of T^s_α above p using f_p, avoiding all nodes in E^s. In executing the algorithm to find the standard red subtree, we will assume that all elements at level k are determined before any elements at level $k+1$. Similarly, $d(T^s_\alpha)^\frown(p, \mathsf{full})$ is the descriptor for the tree of all elements of T^s_α extending the least extension of p which lies above all elements of E^s. Call this least extension the root. Because of the way we will construct E^s, the root of the tree with descriptor $d(T^s_\alpha)^\frown(p, \mathsf{full})$ is always a proper extension of p. Because of the way descriptors are defined, for every s and α, T^s_α is either isomorphic to $2^{<\omega}$ or finite. Since descriptors are always finite, they can be encoded by an integer and tagged onto markers.

In the following construction, the behavior of each marker is very limited. Initially, we place p_α and guess that red is the appropriate color for c_α. As long as no difficulties arise in locating extensions of p_α in standard red subtrees, p_α and c_α remain unchanged. If the search for extensions fails, then (some) p_α must have a full blue subtree for f_{p_α}. In this case, we change c_α to blue and attempt to move p_α to a successor of its current location. This move is a necessary complication, allowing us to decode a monochromatic subtree from the current tree. If c_α is blue at stage s, then p_α will not be moved unless the descriptor $d(T^s_\alpha)$ is shortened, or for some $\beta \sqsubset \alpha$, p_β is modified.

Construction.

Stage 0. Let $p^0_{\langle\,\rangle} = \langle\,\rangle$, $c^0_{\langle\,\rangle} = \mathsf{red}$, $d(T^0_{\langle\,\rangle}) = (\langle\,\rangle, \mathsf{red})$, $E^0 = \emptyset$, and $M^0 = \{\langle\,\rangle\}$. Thus, we have assigned the empty node the color red, will search for successors of this marker in the standard red subtree of $2^{<\omega}$ for $\langle\,\rangle$ using $f_{\langle\,\rangle}$, have determined no elements in the complement

of the tree from which we will extract our monochromatic tree, and have placed exactly one marker, corresponding to the location of $\langle\ \rangle$ in $2^{<\omega}$. All other markers are unassigned.

Stage $s+1$. We will use two cases to describe the action at this stage.

Case 1. For each leaf $\beta \in M^s$, we can locate incomparable proper extensions $p_{\beta,0}$ and $p_{\beta,1}$ of p^s_β in T^s_β. Note that we can determine whether or not this case holds on the basis of finitely many queries to $0'$. When this case holds, do the following:

1. For each leaf $\beta \in M^s$ and each $\varepsilon \in \{0,1\}$:

   - Set $p^{s+1}_{\beta^\frown\varepsilon} = p_{\beta,\varepsilon}$ and $c^{s+1}_{\beta^\frown\varepsilon} = \mathsf{red}$;
   - add all elements of M^s and $\beta^\frown\varepsilon$ to M^{s+1};
   - Let the descriptor for $T^{s+1}_{\beta^\frown\varepsilon}$ be $d(T^s_\beta)^\frown(p_{\beta,\varepsilon}, \mathsf{red})$.

2. For all other α, set $p^{s+1}_\alpha = p^s_\alpha$, $c^{s+1}_\alpha = c^s_\alpha$, and $d(T^{s+1}_\alpha) = d(T^s_\alpha)$.

3. Let $L = \max\{lh(p^{s+1}_\alpha) \mid \alpha \in M^{s+1}\}$, and set

$$E^{s+1} = E^s \cup \{\tau \in 2^{<\omega} \mid lh(\tau) < L \wedge \forall \alpha \in M^{s+1}(\tau \neq p^{s+1}_\alpha)\}.$$

Case 2. Case 1 fails, so there is a leaf $\beta \in M^s$ with no incomparable proper extensions of p^s_β in T^s_β. Using $0'$ we can fix such a leaf β.

Intuitively, whenever this situation arises, we need to create a blue marker. For example, if c^s_β is red and we can find no such extensions, then we should change c^s_β to blue. Though not as obvious, blue markers with no extensions arise from erroneous red nodes in descriptors. To complicate matters, simply changing the color of a marker creates problems with extracting

the final monochromatic tree from the tree under construction. Consequently, in this case we will move some marker and color it blue.

We will search each tree B in a list for a pair of nodes $p_0 \sqsubset p_1$ such that $\forall \tau \in B(p_1 \sqsubset \tau \rightarrow f_{p_0}(\tau) = \mathsf{blue})$. The trees fall into two categories. If $\alpha \sqsubseteq \beta$ and $c^s_\alpha = \mathsf{red}$, then the descriptor $d(T^s_\alpha)$ is of the form $d^\frown(p^s_\alpha, \mathsf{red})$. For each (possibly empty) sequence $p_{\alpha,0} \sqsubset p_{\alpha,1} \sqsubset \cdots \sqsubset p_{\alpha,k}$ of nodes in the tree with descriptor $d^\frown(p^s_\alpha, \mathsf{full})$, add the tree with descriptor

$$d^\frown(p^s_\alpha, \mathsf{full})^\frown(p_{\alpha,0}, \mathsf{red})^\frown \ldots ^\frown (p_{\alpha,k}, \mathsf{red})$$

to the list. If $\alpha \sqsubseteq \beta$ and $c^s_\alpha = \mathsf{blue}$, then the descriptor $d(T^s_\alpha)$ may be of the form $d^\frown(p_{\alpha,0}, \mathsf{red})^\frown \ldots ^\frown (p_{\alpha,k}, \mathsf{red})^\frown(p_{\alpha,k+1}, \mathsf{full})^\frown(p_{\alpha,k+2}, \mathsf{full})$ where $k \geq 0$. If so, then for each $j \leq k$ add the tree with the descriptor

$$d^\frown(p_{\alpha,0}, \mathsf{red})^\frown \ldots ^\frown (p_{\alpha,j-1}, \mathsf{red})^\frown(p_{\alpha,j}, \mathsf{full})$$

to the list. Search all trees in the list until p_0 and p_1 as described at the beginning of this paragraph are found. (We allow p_0 to be the root of a tree; in particular, if the descriptor of B terminates in $(p_{\alpha,j}, \mathsf{full})$, we may let p_0 be the root of B, which is the least extension of $p_{\alpha,j}$ lying above all elements of E^s.) A proof that this search always terminates is given in Claim 3.4.4 below. Remember the descriptor of the tree for which the search succeeded, including the value of α.

Suppose we have found p_0, p_1, and $\alpha \sqsubseteq \beta$ as specified in the preceding paragraph. Do the following:

1. Let $M^{s+1} = \{\gamma \in M^s \mid \gamma \not\sqsupset \alpha\}$.

2. For $\gamma \in M^{s+1} - \{\alpha\}$, let $p_\gamma^{s+1} = p_\gamma^s$, $c_\gamma^{s+1} = c_\gamma^s$, and $d(T_\gamma^{s+1}) = d(T_\gamma^s)$.

3. Let $E^{s+1} = E^s$.

Denote the descriptor of the tree for which the search succeeded by d_0 and do the following:

1. Set $c_\alpha^{s+1} = \mathsf{blue}$ and $p_\alpha^{s+1} = p_0$.

2. Let $d(T_\alpha^{s+1}) = d_0^\frown(p_0, \mathsf{full})^\frown(p_1, \mathsf{full})$.

This completes the construction.

The next four claims show that the construction yields the desired enumeration of the complement of the preliminary tree, from which we will extract our monochromatic tree.

Claim 3.4.4. The search described in Case 2 of Stage $s+1$ always terminates.

Proof. Suppose there is a leaf $\beta \in M^s$ with no proper extensions of p_β^s in T_β^s. The absence of extensions indicates that T_β^s is not isomorphic to $2^{<\omega}$, so T_β^s must be finite. Each initial segment of the descriptor $d(T_\beta^s)$ is a descriptor for some tree. Since the empty sequence is the descriptor for $2^{<\omega}$, there must be a first pair (p, c) such that the initial segment of $d(T_\beta^s)$ terminating in (p, c) describes a finite tree.

If d is the descriptor for a tree isomorphic to $2^{<\omega}$ containing p, then $d^\frown(p, \mathsf{full})$ is also isomorphic to $2^{<\omega}$. Thus the pair (p, c) in the preceding paragraph must be of the form (p, red). The node p must either be a p_α for some $\alpha \sqsubseteq \beta$, or a node on a path leading to some p_α for which $c_\alpha^s = \mathsf{blue}$. We will consider these situations in order.

First suppose (p, c) is of the form (p_α, red) for some $\alpha \sqsubseteq \beta$ where $c_\alpha^s = \mathsf{red}$. Then $d(T_\beta^s)$ is of the form $d^\frown(p_\alpha^s, \mathsf{red})^\frown\hat{d}$. (Note that the following holds when $\hat{d} = \emptyset$.) The tree with

descriptor $d^\frown(p^s_\alpha, \mathsf{full})$ is isomorphic to $2^{<\omega}$. Suppose by way of contradiction that the search fails. That is, given any (possibly empty) sequence $p_{\alpha,0} \sqsubset p_{\alpha,1} \sqsubset \cdots \sqsubset p_{\alpha,k}$ of nodes in the tree with descriptor $d^\frown(p^s_\alpha, \mathsf{full})$, if we let B be the tree with descriptor

$$d^\frown(p^s_\alpha, \mathsf{full})^\frown(p_{\alpha,0}, \mathsf{red})^\frown \ldots ^\frown (p_{\alpha,k}, \mathsf{red}),$$

then there is no pair $p_0 \sqsubset p_1$ in B such that $\forall \tau \in B\ (p_1 \sqsubset \tau \to f_{p_0}(\tau) = \mathsf{blue})$. Consequently, for any such B, p_0, and p_1, there is a $\tau \supset p_1$ in B such that $f_{p_0}(\tau) = \mathsf{red}$. We can use this feature to construct a computable monochromatic red tree as follows.

Let B denote the tree with descriptor $d^\frown(p^s_\alpha, \mathsf{full})$. Let $q_{\langle\ \rangle}$ denote the root of this tree; that is, $q_{\langle\ \rangle}$ is the least extension of p^s_α lying above all elements of E^s. By the preceding paragraph, there is no $p_1 \supset q_{\langle\ \rangle}$ such that $\forall \tau \in B\ (p_1 \sqsubset \tau \to f_{q_{\langle\ \rangle}}(\tau) = \mathsf{blue})$. Let $B_{\langle\ \rangle}$ be the tree with descriptor $d^\frown(q_{\langle\ \rangle}, \mathsf{red})$. By Lemma 3.3.1, $B_{\langle\ \rangle}$ is isomorphic to $2^{<\omega}$. Suppose q_α and B_α are defined and B_α is isomorphic to $2^{<\omega}$. Let $q_{\alpha^\frown 0}$ and $q_{\alpha^\frown 1}$ be the first pair of incomparable elements of B_α. For each $\varepsilon \in \{0, 1\}$, treating $q_{\alpha^\frown \varepsilon}$ as p_0, by the preceding paragraph, the tree with descriptor

$$d^\frown(q_{\langle\ \rangle}, \mathsf{red})^\frown \ldots (q_\alpha, \mathsf{red})^\frown (q_{\alpha^\frown \varepsilon}, \mathsf{red})$$

(which will be $B_{\alpha^\frown \varepsilon}$) is isomorphic to $2^{<\omega}$. Note that if $\alpha \sqsubset \beta$, then $q_\beta \in B_\alpha$, so $f_{q_\alpha}(q_\beta) = \mathsf{red}$. Thus $\{q_\alpha \mid \alpha \in 2^{<\omega}\}$ is a computable monochromatic tree for f.

The existence of a computable monochromatic tree for f contradicts the first paragraph of the proof of this theorem. Consequently, when c^s_α is red, the search must terminate,

completing the proof for this situation.

Now suppose (p, c) is of the form $(p_{\alpha,j}, \mathsf{red})$ for some $\alpha \sqsubseteq \beta$ where $c^s_\alpha = \mathsf{blue}$. Then $d(T^s_\beta)$ is of the form

$$d^\frown(p_{\alpha,j}, \mathsf{red})^\frown \ldots ^\frown (p_{\alpha,k}, \mathsf{red})^\frown(p_{\alpha,k+1}, \mathsf{full})^\frown(p_{\alpha,k+2}, \mathsf{full})^\frown\hat{d}.$$

(Note that the following holds if $\hat{d} = \emptyset$ and also if $j = k$.) Since $(p_{\alpha,j}, \mathsf{red})$ was the first pair yielding a finite tree, the tree with descriptor $d^\frown(p_{\alpha,j}, \mathsf{full})$ is isomorphic to $2^{<\omega}$. As in the preceding paragraphs, if we cannot find p_0 and p_1 satisfying the search, then we can construct a computable monochromatic tree, yielding a contradiction. This completes the proof of the claim that the search always succeeds. □

Claim 3.4.5. For every $\alpha \in 2^{<\omega}$, the following limits exist: $\lim_s p^s_\alpha = p_\alpha$, $\lim_s c^s_\alpha = c_\alpha$, and $\lim_s d(T^s_\alpha) = d_\alpha$.

Proof. Consider the possible behaviors for $p^s_{\langle\,\rangle}$. If $p^s_{\langle\,\rangle}$ is never moved, then $p_{\langle\,\rangle} = \langle\,\rangle$, $c_{\langle\,\rangle} = \mathsf{red}$, and $T_{\langle\,\rangle}$ is the tree with descriptor $(\langle\,\rangle, \mathsf{red})$. At some stage s, $p^s_{\langle\,\rangle}$ may be moved, in which case $c^s_{\langle\,\rangle} = \mathsf{blue}$ and $T^s_{\langle\,\rangle}$ has a new descriptor of some length n. At any successive stage t, $c^t_{\langle\,\rangle} = \mathsf{blue}$ and if $p^t_{\langle\,\rangle}$ moves, then the descriptor of $T^t_{\langle\,\rangle}$ is shortened. Consequently, the process must eventually converge to a limiting p_α and d_α.

If p_α, c_α and d_α have achieved their limits at stage s, then the only allowable changes in $p^t_{\alpha^\frown\varepsilon}$, $c^t_{\alpha^\frown\varepsilon}$, and $d(T^t_{\alpha^\frown\varepsilon})$, for $\varepsilon \in \{0, 1\}$ and $t > s$ are exactly those in the preceding paragraph. Thus, all the markers must achieve their limits.

Furthermore, each time Case 2 of Stage $s + 1$ is executed, either M^s is decreased in size,

or for some $\alpha \in M^s$, either c^s_α is changed from red to blue or the descriptor of T^s_α is shortened. Since M^s and all descriptors are finite, Case 1 of Stage $s+1$ must occur infinitely often. Consequently, once p_α achieves its limit, $p^s_{\alpha^\frown\varepsilon}$ must eventually be introduced, and will also achieve its limit. Thus, for every $\alpha \in 2^{<\omega}$, p_α and c_α are assigned, and T_α is a nonempty tree. □

Claim 3.4.6. For every $\alpha \in 2^{<\omega}$, if $\alpha \sqsubset \beta$ then $p_\alpha \sqsubset p_\beta$, $d_\alpha \sqsubset d_\beta$, and $f_{p_\alpha}(p_\beta) = c_\alpha$.

Proof. Detailed examination of the construction shows that for each s, if $\alpha \sqsubset \beta \in M^s$, then $p^s_\alpha \sqsubset p^s_\beta$, $p^s_\beta \in T^s_\alpha$, and $d(T^s_\beta)$ extends $d(T^s_\alpha)$. Consequently, $f_{p^s_\alpha}(p^s_\beta) = c^s_\alpha$ and $T^s_\beta \subset T^s_\alpha$. Since these relationships are preserved at each stage, they must hold in the limit. □

Claim 3.4.7. $\overline{\{p_\alpha \mid \alpha \in 2^{<\omega}\}} = \bigcup_s E^s$.

Proof. As shown in Claims 3.4.5 and 3.4.6, for each $\alpha \in 2^{<\omega}$, p_α exists, and if $\beta \sqsupset \alpha$, then $p_\beta \sqsupset p_\alpha$. Thus, the length of p^s_α can be forced to exceed any fixed bound in $\mathbb{N}$ by picking suitably large values of s and α. By virtue of the definition of E^{s+1} in Case 1 of Stage $s+1$ (which occurs infinitely often), $\bigcup_s E_s \supseteq \overline{\{p_\alpha \mid \alpha \in 2^{<\omega}\}}$. Since each T^s_α is defined so as to avoid elements of E^s, no p_α can be an element of $\bigcup_s E_s$. □

Summarizing the proof to this point, we have a subtree $\{p_\alpha \mid \alpha \in 2^{<\omega}\}$ which is isomorphic to $2^{<\omega}$, and satisfies $f_{p_\alpha}(p_\beta) = c_\alpha$ whenever $\alpha \sqsubset \beta$. Furthermore, the complement of this set is the union of finite sets each of which can be computed with the aid of $0'$. Consequently, the complement is computably enumerable in $0'$. Thus, we have found the desired tree, with complement computably enumerable in $0'$. It remains to extract a monochromatic subtree and describe an enumeration for its complement.

First, suppose there is an α such that for all $\beta \sqsupseteq \alpha$, $c_\beta = \mathsf{red}$. Then the subtree $T = \{p_\beta \mid \beta \sqsupseteq \alpha\}$ is the desired monochromatic red tree. To enumerate the complement of T, repeat the construction, adding p^s_γ to E^s whenever $p^s_\gamma \not\sqsupset p_\alpha$. Since the complement of T is computably enumerable in $0'$, by the strong hierarchy theorem T is Π^0_2 definable.

Finally, suppose that for every α there is a $\beta \sqsupset \alpha$ such that $c_\beta = \mathsf{blue}$. We repeat the construction, adding new markers $\{t_\alpha \mid \alpha \in 2^{<\omega}\}$, new finite subsets of the complement of the monochromatic tree $\{F^s \mid s \in \mathbb{N}\}$, and new maps $\{N^s \mid s \in \mathbb{N}\}$ where N^s contains those α for which t_α is attached at stage s.

Run the construction until the first c^s_α is set to blue. Let $t^s_{\langle\,\rangle} = p^s_\alpha$, $N^s = \{\langle\,\rangle\}$, and $F^s = \{p^s_\beta \mid \beta \in M^s \wedge \beta \neq \alpha\}$. Note that if $p^s_\beta \in F^s$, then $c^s_\beta = \mathsf{red}$.

At stage $s+1$, execute the process for constructing S, and then consider three cases.

Case 1. For each leaf $\beta \in N^s$, given that $t^s_\beta = p^s_\gamma$, suppose we can locate extensions $p^s_{\delta_0} \sqsupset p^s_{\gamma^\frown 0}$ and $p^s_{\delta_1} \sqsupset p^s_{\gamma^\frown 1}$ such that $c^s_{\delta_0} = \mathsf{blue}$ and $c^s_{\delta_1} = \mathsf{blue}$. In this case, do the following:

1. For each leaf $\beta \in N^s$ and each $\varepsilon \in \{0,1\}$,
   - set $t^{s+1}_{\beta^\frown\varepsilon} = p^s_{\delta_\varepsilon}$, and
   - add all elements of N^s and $\beta^\frown\varepsilon$ to N^{s+1}.
2. For all other $\alpha \in N^s$, set $t^{s+1}_\alpha = t^s_\alpha$.
3. Define F^{s+1} by the equation

$$F^{s+1} = F^s \cup \{\tau \in 2^{<\omega} \mid \exists \alpha \in M^{s+1}(\tau = p^{s+1}_\alpha) \wedge \forall \alpha \in N^{s+1}(\tau \neq t^{s+1}_\alpha)\}.$$

Case 2. For some $\beta \in N^s$, a predecessor of t^s_β is moved. Let p^s_δ be this predecessor node.

Because of the way nodes are added in Case 1, there is a unique least $\alpha \sqsubseteq \beta$ such that $t^s_\alpha \sqsupseteq p^s_\delta$. Find this α, and do the following:

1. Let $N^{s+1} = \{\gamma \in N^s \mid \gamma \not\sqsupset \alpha\}$.

2. For $\gamma \in N^{s+1} - \{\alpha\}$, let $t^{s+1}_\gamma = t^s_\gamma$.

3. Let $F^{s+1} = F^s$.

4. Let $t^{s+1}_\alpha = p^{s+1}_\delta$.

This last step is possible because p^s_δ was moved to a previously unassigned location, guaranteeing that $p^s_\delta \notin F^s$. Furthermore, since p^s_δ moved, $c^{s+1}_\delta = \mathsf{blue}$.

Case 3. If neither Case 1 nor Case 2 holds, let $N^{s+1} = N^s$, $F^{s+1} = F^s$, and $t^{s+1}_\alpha = t^s_\alpha$ for all $\alpha \in N^{s+1}$.

It is not difficult to show that for each α, the limit $t_\alpha = \lim_s t^s_\alpha$ exists, and that it marks some p_γ such that $c_\gamma = \mathsf{blue}$. Also, $\overline{\{t_\alpha \mid \alpha \in 2^{<\omega}\}} = \bigcup_s E_s \cup \bigcup_s F_s$, so $\{t_\alpha \mid \alpha \in 2^{<\omega}\}$ is the complement of a set which is computably enumerable in $0'$. Thus, in this situation, $\{t_\alpha \mid \alpha \in 2^{<\omega}\}$ is a blue monochromatic tree which is Π^0_2 definable.

We have completed the proof for two colors. To extend the result to an arbitrary finite number of colors, we modify the construction, assigning colors 0 through k in order. Initially c^s_α is assigned 0. In Case 2 of Stage $s+1$, if c^s_α is assigned $j < k$ then we search for $p_0 \sqsubset p_1$ such that $\forall \tau \in B(p_1 \sqsubset \tau \to j < f_{p_0}(\tau))$ and set $c^{s+1}_\alpha = j+1$. The color k behaves like blue in the original construction.

The claims are proved as before, and again yield a tree with each c_α in $\{0, \ldots, k\}$. Pick the

least j such that there is an α such that for all $\beta \sqsupseteq \alpha$, $c_\beta \leq j$. If $j = 0$, then $T = \{p_\beta \mid \beta \sqsupseteq \alpha\}$ is the desired monochromatic subtree. Otherwise, rerun the construction using new markers to extract a j-colored subtree. The Π^0_2 bound follows as before. $\square$

3.5 The Π^0_n bound

In the previous section, we showed that when $n = 1$, there is a computable monochromatic substructure (though for more than two colors, this fact is not provable in RCA_0). In this section, we give definitions and develop some machinery that will allow us to use the $n = 2$ case shown above as a base for an induction yielding the appropriate bound for $n > 2$.

Theorem 3.5.1 ([10])**.** *Let $n \geq 2$. If $f : [2^{<\omega}]^n \to k$ is computable, then there is a Π^0_n monochromatic subtree isomorphic to $2^{<\omega}$ for f.*

The theorem will be proven at the end of this chapter. It follows in perfect analogy with Jockusch's result [36] that for $n \geq 2$, if f is a computable partition of $[\mathbb{N}]^n$, then there is an infinite Π^0_n subset of $\mathbb{N}$ on which f is constant.

Before we begin talking about colorings of the n-chains of $2^{<\omega}$, we make some basic definitions and observations. For now, let $f : 2^{<\omega} \to k$. Also (and always), for each $\alpha \in 2^{<\omega}$, we let T_α denote the full subtree of $2^{<\omega}$ rooted at α, that is $T_\alpha = \{\tau \in 2^{<\omega} \mid \alpha \sqsubseteq \tau\}$. Now we can define *color blocks* and *avoiding trees*.

Definition 3.5.2. A *color block for f* is a set of $k + 1$ chains with the following properties:

1. Each chain consists of k nodes, exactly one of each color.

2. Any two nodes chosen from distinct chains are incomparable.

Definition 3.5.3. For $c < k$, we say f has a *full c-avoiding tree* if there is some node τ such that for all $\sigma \sqsupset \tau$, $f(\sigma) \neq c$.

The following lemma ensures that we may always assume that we have one or the other of these two objects. Note that although the lemma is stated and proved for $2^{<\omega}$, its consequence applies to T_α for any $\alpha \in 2^{<\omega}$, as do the consequences of the lemmas and theorems in the previous section.

Lemma 3.5.4. *Either there is a c such that f has a full c-avoiding tree or there is a color block for f.*

Proof. We will search $2^{<\omega}$ for a color block for f. If the search fails, it is because we have discovered a full c-avoiding tree for some c.

Begin by selecting $k+1$ pairwise incomparable nodes in $2^{<\omega}$, the least $k+1$ such nodes will do. For each node σ in this collection, do the following:

> Let $\sigma_0 = \sigma$. For $0 \leq i \leq k-2$, given σ_i, let σ_{i+1} be the least node extending σ_i with $f(\sigma_{i+1}) \neq f(\sigma_j)$ for $j \leq i$, if such a node exists. (Note that deciding the existence of such a σ_{i+1} requires a query to $0'$ when f is computable.)

If this search fails for some i, it is because for each node τ extending σ_i there is some $j \leq i$ so that $f(\tau) = f(\sigma_j)$, thus T_{σ_i} is c-avoiding for any $c \notin \{f(\sigma_j) | j \leq i\}$.

If the search does not fail, we have successfully constructed $k+1$ non-intersecting chains, each consisting of k distinctly colored nodes. Furthermore, when f is computable, this construction may be carried out with only finitely many (in fact at most $(k-1)(k+1)$) queries to $0'$. $\square$

In our later constructions, when a color block exists, we will exploit the following basic fact to work toward building our homogeneous set.

Lemma 3.5.5. *If there is a color block for f, then there is a monochromatic subtree isomorphic to $2^{<\omega}$ in which the first level (as distinct from the zeroth level) consists of (exactly two) nodes in the color block.*

Proof. Let B be a color block for f. For each $i \leq k$, let μ_i be the maximal node in the ith chain.

By Theorem 3.3.3, each tree T_{μ_i} has a monochromatic subtree isomorphic to $2^{<\omega}$, and as there are $k+1$ such trees, at least two of them are of the same color, say $c < k$. Assume T_{μ_i} and T_{μ_j} have monochromatic subtrees S_i and S_j, respectively, of color c with corresponding roots μ_i and μ_j. Let σ_i and σ_j be the elements of the ith and jth chains in B having color c. Then $T = \{\langle\,\rangle, \sigma_i, \sigma_j\} \cup (S_i \backslash \{\mu_i\}) \cup (S_j \backslash \{\mu_j\})$ is a monochromatic subtree of $2^{<\omega}$ isomorphic to $2^{<\omega}$ having its first level nodes in the color block B. □

For what follows, we will need a slightly more flexible definition of color blocks.

Definition 3.5.6. If f is a k-coloring and S is a subset of the colors, an *S-color block for f* is a collection of $|S|+1$ chains, each of which is composed of exactly one node from each color in S, satisfying the incomparability requirement in the definition of a color block.

The definition below describes the main tool we will need in our construction of the homogeneous substructure.

Definition 3.5.7. An *f-forest* is a collection of finite binary trees with nodes from a sequence of disjoint sets of nodes, $\langle L_i \rangle_{i \in \mathbb{N}}$, defined as follows:

Let $L_0 = \{\langle\ \rangle\}$. Note that the range of f is $\{0, \ldots, k-1\}$, and attach the tag $(\{0, \ldots, k-1\}, \langle\ \rangle)$ to $\langle\ \rangle$.

Suppose that L_n is defined. If some $\sigma \in L_n$ has a tag then do the following:

1. If the tag on σ is (S, τ) and $|S| > 1$, then check for an S-color block above τ using the algorithm described in the proof of Lemma 3.5.4.

   (a) If such a color block is located, add all the nodes in the color block to L_{n+1}. Whenever μ is the supremum of a chain in the color block attach the tag (S, μ) to μ. Remove the tag from σ.

   (b) If no such color block exists, then for some $c \in S$ and some β above τ there is a c-avoiding tree above β. Change the tag on σ to $(S - \{c\}, \beta)$.

2. If the tag on σ is (S, τ) and $|S| = 1$, then the tree above τ is monochromatic. Add $\tau^\frown 0$ and $\tau^\frown 1$ to L_{n+1}. Attach the tag $(S, \tau^\frown 0)$ to $\tau^\frown 0$ and $(S, \tau^\frown 1)$ to $\tau^\frown 1$. Remove the tag from σ.

If no element of L_n has a tag, then the calculation of L_{n+1} is complete, and L_{n+1} is now defined.

The f-forest consists of all finite monochromatic binary subtrees T such that for all k less than or equal to the height of T, the k^{th} level of T consists of exactly 2^k elements from L_k.

This completes the definition of the f-forest, and we conclude by noting that the f-forest is a partially ordered set under extension.

In the following, given any finite tree T, let $\ulcorner T \urcorner$ denote some canonical integer code for T.

Lemma 3.5.8. *If f is computable, then the f-forest is computable from $0'$. Furthermore, there is a function g computable from $0'$ such that for all n, if T is an element of the f-forest of height n, then $\ulcorner T \urcorner \leq g(n)$.*

Proof. Let T be a finite tree of height n, and $\mathcal{F}$ the f-forest for our computable f. To determine whether $T \in \mathcal{F}$, we first (computably) ensure that it is monochromatic and isomorphic to $2^{\leq n}$. Then check for each $k \leq n$ that

$$(\forall \sigma \in T)[\sigma \text{ has } k \text{ predecessors in } T \implies \sigma \in L_k].$$

Thus membership in $\mathcal{F}$ reduces to finitely many questions about membership in the sets L_k for $k \leq n$. By the proof of Lemma 3.5.4 , the construction of the (finite) set L_k can be carried out with the assistance of $0'$.

Since each set L_k is finite, there are only finitely many trees of a given height n belonging to $\mathcal{F}$. With the aid of $0'$, we may find these and set $g(n)$ to exceed the largest of their canonical indices. □

We now give an important lemma.

Lemma 3.5.9. *If f is computable, then there is a monochromatic subtree T isomorphic to $2^{<\omega}$ such that $T' \leq 0''$.*

Proof. Let f be a computable k-coloring and $\mathcal{F}$ the associated f-forest. Consider $\mathcal{F}$ as a partial order under inclusion and note that it has the structure of a finitely branching tree,

though not typically binary. By Lemma 3.5.8 if the trees that are the nodes of this partial order are identified with their canonical codes, we have a function $g \leq 0'$ bounding each level of $\mathcal{F}$.

To see that $\mathcal{F}$ is infinite, suppose by way of contradiction that there is an upper bound on the height of the trees in $\mathcal{F}$. In this case, we may find a coloring h such that the maximum height of a tree in the h-forest is minimal among all choices of colorings. Let H be a tree from the h-forest that has this maximal height, which we will denote by n. By Lemma 3.5.4 and the definition of an h-forest, $n \geq 1$. Let L_1 be the first level of the h-forest. If L_1 consisted of nodes taken from a monochromatic subtree, then H could be extended without bound in the h-forest. Consequently, L_1 is a color block. Let j be the number of colors in L_1 and let $\mu_0, \ldots, \mu_j$ be the maximal elements of the $j+1$ chains in L_1. For each μ_i, the h-forest on T_{μ_i} contains a monochromatic tree M_{μ_i} of height at least n. (Recall that n is the minimum of the maximum height of a tree in the forest over all possible colorings.) Since there are $j+1$ of these trees and only j colors, two must match; suppose that M_{u_i} and M_{u_j} match, where $i \neq j$. Let σ_i (respectively σ_j) be the node in the chain in L_1 below μ_i (respectively μ_j) of the same color as M_{u_i} (which is the same color as M_{u_j}). Then $\{\sigma_i, \sigma_j\} \cup M_{\mu_i} \cup M_{\mu_j}$ is a monochromatic tree in the h-forest of height at least $n+1$, yielding a contradiction. Thus the elements of $\mathcal{F}$ are unbounded in height, and S is an infinite finitely branching tree with each level bounded by the function g.

In light of these observations, we see that the (relativized) Jockusch-Soare low basis theorem ([37]) may be applied to obtain a path P in S that is low over $0'$, that is, $P' \leq 0''$. This path is a sequence of nested finite monochromatic binary trees. Since each is as full as possible for its height (the nth node in P is a tree isomorphic to $2^{\leq n}$), their union yields a

monochromatic subtree T, isomorphic to $2^{<\omega}$. Finally, since $T \leq P$ and $P' \leq 0''$, we have $T' \leq 0''$. $\square$

Before we can take the final steps in proving Theorem 3.5.14, we need one more definition and a few more lemmas. These parallel and extend those that we have just seen. They are the analogues for a family of finite colorings instead of a single coloring.

In what follows, we suppose that for each $\alpha \in 2^{<\omega}$, the function $f_\alpha : T_\alpha \to k_\alpha$ is a finite coloring of T_α. Suppose the initial segments of α are $\alpha_0 \sqsubset \alpha_1 \sqsubset \alpha_2 \sqsubset \cdots \sqsubset \alpha_n = \alpha$. Define $k^*_\alpha = \{(j_0, \ldots, j_n) \mid \forall i \; j_i < k_{\alpha_i}\}$, and $f^*_\alpha : T_\alpha \to k^*_\alpha$ by $f^*_\alpha(\tau) = (f_{\alpha_0}(\tau), f_{\alpha_1}(\tau), \ldots, f_{\alpha_n}(\tau))$.

Now we recursively define the $\langle f_\alpha \rangle$-forest for a family of colorings $\{f_\alpha\}_{\alpha \in 2^{<\omega}}$.

Definition 3.5.10. An $\langle f_\alpha \rangle$-forest is defined in terms of a sequence of levels $\langle L_i \rangle_{n \in \mathbb{N}}$. The levels are defined as follows:

Let $L_0 = \{\langle\,\rangle\}$. Note that $f^*_{\langle\,\rangle} = f_{\langle\,\rangle}$ and that the range of $f_{\langle\,\rangle}$ is $\{0, 1, \ldots, k_{\langle\,\rangle} - 1\}$. Attach the tag $(\{0, 1, \ldots, k_{\langle\,\rangle} - 1\}, \langle\,\rangle)$ to $\langle\,\rangle$.

Suppose that L_n is defined. If some $\sigma \in L_n$ has a tag, then do the following:

1. If the tag on σ is (S, τ) and $|S| > 1$, then check for an S color block for f^*_σ above τ using the algorithm from the proof of Lemma 3.5.4.

   (a) If such a color block is located, add all the nodes in the color block to L_{n+1}. Whenever μ is the supremum of a chain in the color block, define

   $$S_\mu = \{(v_0, v_1, \ldots, v_{|\sigma|}, \ldots v_{|\mu|}) \in k^*_\mu \mid (v_0, v_1, \ldots, v_{|\sigma|}) \in S\},$$

   and attach the tag (S_μ, μ) to μ. Remove the tag from σ.

(b) If no such color block is found, then for some $c \in S$ and some β above τ there is a c-avoiding tree for f^*_σ above β. Change the tag on σ to $(S - \{c\}, \beta)$.

2. If the tag on σ is (S, τ) and $|S| = 1$, then the tree above τ is monochromatic for f^*_σ. Add $\tau^\frown 0$ and $\tau^\frown 1$ to L_{n+1}. For each $\varepsilon \in \{0, 1\}$, define

$$S_{\tau^\frown \varepsilon} = \{(v_0, \ldots, v_{|\sigma|}, \ldots, v_{|\tau^\frown \varepsilon|}) \in k^*_{\tau^\frown \varepsilon} \mid (v_0, \ldots, v_\sigma) \in S\},$$

and attach the tag $(S_{\tau^\frown \varepsilon}, \tau^\frown \varepsilon)$ to $\tau^\frown \varepsilon$. Remove the tag from σ.

If no element of L_n has a tag, then the calculation of L_{n+1} is complete, and L_{n+1} is defined. The $\langle f_\alpha \rangle$-forest consists of all finite binary subtrees T such that

1. the k^{th} level of T contains exactly 2^k elements from L_k, and

2. if σ, τ_1, and τ_2 are nodes of T and τ_1 and τ_2 both extend σ, then $f^*_\sigma(\tau_1) = f^*_\sigma(\tau_2)$.

In short, the trees in an $\langle f_\alpha \rangle$-forest are constructed in much the same way as in an f-forest. If T is a tree in an $\langle f_\alpha \rangle$-forest and τ is in T, then f_τ is monochromatic on the nodes of T above τ.

Lemma 3.5.11. *Suppose that for each $\alpha \in 2^{<\omega}$, $f_\alpha : T_\alpha \to k_\alpha$ is a finite coloring of T_α. If $\langle f_\alpha \rangle_{\alpha \in 2^{<\omega}}$ is a uniformly computable collection of finite colorings, then the $\langle f_\alpha \rangle$-forest is computable from $0'$. Furthermore, there is a function g such that $g \leq 0'$ and for all n, if T is a height n element of the $\langle f_\alpha \rangle$-forest, then $\ulcorner T \urcorner \leq g(n)$.*

Idea of proof: This proof is essentially the same as that of Lemma 3.5.8. The only alteration in the computation of the levels L_k arises from the fact that the size of the color blocks may

increase from one level to the next. However, given a node σ in L_k, the number of nodes in the color block above σ is at most n^2+n, where n is the cardinality of the range of f^*_σ. Since each level L_k is finite and computable from $0'$, using only $0'$ we can also compute membership in the $\langle f_\alpha \rangle$-forest and the bounding function g. $\square$

Lemma 3.5.12. *If $\langle f_\alpha \rangle$ is as in Lemma 3.5.11, then there is a subtree T such that the following conditions hold.*

1. *T is isomorphic to $2^{<\omega}$.*
2. *For each $\sigma \in T$, f_σ is constant on $\{\tau \in T \mid \tau \sqsupset \sigma\}$.*
3. *$T' \leq 0''$.*

Proof. This proof parallels that of Lemma 3.5.9, substituting applications of Lemma 3.5.11 for uses of Lemma 3.5.8. In particular, we begin by letting $\mathcal{F}$ denote the $\langle f_\alpha \rangle$-forest and order its elements by inclusion. By Lemma 3.5.11, $\mathcal{F}$ is a finitely branching tree bounded by a function g, and both $\mathcal{F}$ and g can be computed from $0'$.

The argument that $\mathcal{F}$ is infinite is similar but not identical to that in the proof of Lemma 3.5.9. Suppose by way of contradiction that there is an upper bound on the height of elements of $\mathcal{F}$. In this case, we may find a sequence of colorings $\langle h_\alpha \rangle$ such that the maximum height of a tree in the $\langle h_\alpha \rangle$-forest is minimal among all choices of colorings. Let H be a tree from the $\langle h_\alpha \rangle$-forest that has this maximal height, which we will denote by n. Since the first level of an $\langle h_\alpha \rangle$-forest is the same as the first level of an $h_{\langle\,\rangle}$-forest, by Lemma 3.5.4 we have $n \geq 1$. Let L_1 be the first level of the $\langle h_\alpha \rangle$-forest. We consider two cases.

First suppose L_1 consists of nodes taken from a monochromatic subtree for $h_{\langle\,\rangle}$; denote these by τ_0 and τ_1. For each nonempty $\alpha \in 2^{<\omega}$, define $h^{\tau_0}_\alpha$ by $h^{\tau_0}_\alpha(\beta) = h_{\tau_0 \widehat{\ } \alpha}(\tau_0 \widehat{\ } \beta)$ and also

define $h^{\tau_0}_{\langle\ \rangle}(\beta) = h^*_{\tau_0}(t_0^\frown\beta)$. Define $h^{\tau_1}_\alpha$ similarly, and note that for $i \in \{0,1\}$ the trees of the $\langle h^{\tau_i}_\alpha\rangle$-forest are the extensions of τ_i in the $\langle h_\alpha\rangle$-forest. By our choice of n, the $\langle h^{\tau_0}_\alpha\rangle$-forest and the $\langle h^{\tau_1}_\alpha\rangle$-forest each contain a tree of height n; call them T_{τ_0} and T_{τ_1}. Then $\{\langle\ \rangle\} \cup T_{\tau_0} \cup T_{\tau_1}$ is a tree in the $\langle h_\alpha\rangle$-forest of height $n+1$, contradicting the choice of n.

Now suppose L_1 consists of nodes in a color block for $h_{\langle\ \rangle}$ and let $\mu_0, \ldots, \mu_j$ be the maximal elements of the $j+1$ chains in L_1. Note here that the cardinality of the range of $h_{\langle\ \rangle}$ on nodes in and above the chains is j. As in the previous paragraph, construct the induced sequences of colorings for each μ_i, and a monochromatic tree M_{μ_i} of height n for each μ_i. Two of these, say M_{μ_i} and M_{μ_j}, must agree in the first component of their coloring. Pick σ_i in the chain below μ_i so that $\{\sigma_i\} \cup (M_{\mu_i} - \{\mu_i\})$ is monochromatic for $h_{\langle\ \rangle}$. Choose σ_j for M_{μ_j} similarly, and note that $\{\langle\ \rangle, \sigma_i, \sigma_j\} \cup (M_{\mu_i} - \{\mu_i\}) \cup (M_{\mu_j} - \{\mu_j\})$ is a tree of height at least $n+1$ in the $\langle h_\alpha\rangle$-forest, contradicting the choice of n and completing the proof that $\mathcal{F}$ is infinite.

We have shown that $\mathcal{F}$ is an infinite finitely branching tree, computable from $0'$, with levels bounded by a function g which is also computable from $0'$. By the relativized Jockusch-Soare low basis theorem [37], S has a path P such that $P' \leq 0''$. The desired tree T is the union of the elements in this path. Note that the definition of an $\langle f_\alpha\rangle$-forest ensures that for all $\sigma \in T$, f^*_σ is constant on $\{\tau \in T \mid \tau \sqsupset \sigma\}$ and so f_σ is also constant on this set. □

Lemma 3.5.13. *Suppose $n > 1$ and $f : [2^{<\omega}]^{n+1} \to k$ is computable. There is a tree T which is isomorphic to $2^{<\omega}$ such that the following hold:*

1. *$T' \leq 0''$.*

2. *If $\sigma_1, \ldots, \sigma_n$ is a sequence of n comparable elements of T and τ_1 and τ_2 are extensions*

of σ_n*, then* $f(\sigma_1, \ldots, \sigma_n, \tau_1) = f(\sigma_1, \ldots, \sigma_n, \tau_2)$.

Proof. Define $\langle f_\alpha \rangle_{\alpha \in 2^{<\omega}}$ as follows. If $\mathrm{lh}(\alpha) < n$, let $f_\alpha(\tau) = 0$ for all τ. If $\mathrm{lh}(\alpha) \geq n$, let $\vec{\sigma}_1, \ldots, \vec{\sigma}_n$ be an enumeration of the n-chains of nodes at or below α. For $\tau \sqsupset \alpha$, let $f_\alpha(\tau) = \prod_{j \leq n} \mathsf{pr}(\sigma_j)^{f(\vec{\sigma}_j, \tau)}$. Apply Lemma 3.5.12 to $\langle f_\alpha \rangle$ to obtain a tree T.

Note that T is isomorphic to $2^{<\omega}$ and that $T' \leq 0''$. Let $\vec{\sigma}$ denote a sequence $\sigma_1, \ldots, \sigma_n$ of comparable elements of T. By Lemma 3.5.12, f_{σ_n} is constant on $\{\tau \in T \mid \tau \sqsupset \sigma\}$. Let $\tau_1, \tau_2 \in T$ extend σ_n. Then $f_{\sigma_n}(\tau_1) = f_{\sigma_n}(\tau_2)$. From the definition of f_{σ_n}, we have

$$\prod_{j \leq n} \mathsf{pr}(\sigma_j)^{f(\vec{\sigma}_j, \tau_1)} = \prod_{j \leq n} \mathsf{pr}(\sigma_j)^{f(\vec{\sigma}_j, \tau_2)},$$

and so $f(\vec{\sigma}, \tau_1) = f(\vec{\sigma}, \tau_2)$ as desired. □

Theorem 3.5.14. *If* $f : [2^{<\omega}]^n \to k$ *is computable, then there is a* Π^0_n *monochromatic set for* f.

Proof. Now that the machinery is in place, we can essentially follow the proof of Theorem 5.5 of [36]. We use induction on n.

The case $n = 1$ follows from Theorem 3.3.3 and $n = 2$ follows from Theorem 3.4.3. Suppose the theorem holds for some $n \geq 2$, we will prove it for $n + 1$. Let $f : [2^{<\omega}]^{n+1} \to k$ be computable. Find T as in Lemma 3.5.13. Given any sequence $\sigma_0, \ldots, \sigma_{n-1}$ of comparable elements of T, let σ_n be the least extension of σ_{n-1} in T and define

$$\hat{f}(\sigma_0, \ldots, \sigma_{n-1}) = f(\sigma_0, \ldots, \sigma_{n-1}, \sigma_n).$$

Note that $\hat{f}$ is computable from T. By the induction hypothesis, there is a monochromatic

tree $\hat{T}$ for $\hat{f}$ which is Π^0_n in T. Since $\hat{T}$ is monochromatic for f, it remains only to show that $\hat{T}$ is Π^0_{n+1}. Since $\hat{T}$ is Π^0_n in T, there is a T-computable $(n+1)$-place predicate R such that for all τ

$$\tau \in \hat{T} \leftrightarrow \forall x_1 \ldots Q x_n \; R(\tau, x_1, \ldots, x_n)$$

where Qx_nR is one of $\exists x_n$ and $\forall x_n$. The predicate Qx_nR is computable in T' and hence in $0''$. Applying Post's hierarchy theorem (see, for example, [52]), we may replace Qx_nR by either a Σ^0_3 or a Π^0_3 predicate, depending on whether Qx_nR is $\exists x_n$ or $\forall x_n$. The resulting predicate is the required Π^0_{n+1} definition of $\hat{T}$. $\square$

Chapter 4

Orderings and algebraic structures

In this chapter, the structures we consider do not have an ordering as part of their signature. Instead, we consider a computable copy of some structure and ask how hard it is (computability theoretically) to find a linear ordering of the elements of the universe so that the ordering is respected by the basic functions in the signature of the structure. Specifically, we consider computable groups and semigroups, and begin by giving some basic introduction to ordered groups. See [5], [25], or [38] for additional information and details.

4.1 Basic notions

A *partial left ordering* of a group or semigroup G is a partial ordering, $<_l$, of its elements satisfying for all $x, y, z \in G$

$$x <_l y \implies zx <_l zy.$$

A *partial right ordering* of G is defined similarly, $x <_r y \implies xz <_r yz$. A *partial bi-ordering* of G is a partial ordering that is simultaneously left and right invariant.

A partial left, right, or bi-ordering is a left, right, or bi-ordering if it is a total ordering. We say a group is *left-orderable* if it admits a left-ordering of its elements, and *bi-orderable* if it admits a bi-ordering.

Immediately from the definition, it is clear that a bi-orderable group or semigroup is trivially left- and right-orderable, and a left-orderable group is right-orderable[1]. Of course a right-orderable group is also left-orderable, but neither is necessarily bi-orderable. Furthermore, any group or semigroup that admits a total ordering of any kind must be torsion-free.

For groups, it is frequently convenient to present a partial or total ordering in a different but equivalent way. For both left- and bi-orderings of groups, it is sufficient to specify the collection of those elements that are $\geq e_G$, where e_G is the identity in the group. This collection is called the *positive cone* for the ordering $<$ and we will usually denote it as $P_<$ or P_G, but will drop subscripts when the context is clear.

To see that it is sufficient to specify only the positive cone to completely describe a left- (or bi-) ordering (or partial ordering) of a group, note that

$$a < b \iff e < a^{-1}b.$$

It is simple to check that the correspondence between orderings and cones is a bijection, and we will often identify an ordering $<$ with its positive cone $P_<$. Note further that when G is computable, this bijection preserves the Turing degree [57].

We may also take this opportunity to make note of an important fact. The subsets of group G that are positive cones of a linear left-ordering of its elements can be characterized

[1]If $<_l$ is a left ordering on a group G, a right ordering can be defined by $a <_r b$ if and only if $b^{-1} <_l a^{-1}$.

as follows. Let P be a subset of group G. Then P is the positive cone of some left-ordering of G if and only if

1. $e_G \in P$ and P is a semigroup, i.e., $P \cdot P \subseteq P$.

2. For each $g \in G$ with $g \neq e_G$, exactly one of g and g^{-1} is in P.

If in addition we have

3. For each $g \in G$, $gPg^{-1} \subseteq P$,

then P is the positive cone of a bi-ordering of G. Furthermore, we may observe that if we relax condition 2 from "exactly" to "at most", P defines a *partial* left- or bi-ordering (depending on satisfaction of 3).

Given a partial left-ordering P of G, it is natural to ask if that ordering extends to a total left-ordering. For group elements $g_1, \ldots, g_n$, let $\mathrm{sgr}(g_1, \ldots, g_n)$ be the semigroup generated by these elements and $P^+ = P - \{e_G\}$.

Theorem 4.1.1 ([11])**.** *A partial left ordering P on group G can be extended to a total left ordering if, and only if, for each tuple of non-identity group elements $g_1, \ldots, g_n$, there exist a tuple $\varepsilon_1, \ldots, \varepsilon_n \in \{1, -1\}$ such that*

$$e \notin \mathrm{sgr}(P^+ \cup \{g_1^{\varepsilon_1}, \ldots g_n^{\varepsilon_n}\}).$$

Note that when P is chosen to be the trivial partial order $\{e_G\}$, Theorem 4.1.1 gives the usual semigroup criteria for left-orderability due to Conrad [11].

For a group G, we call the collection of all left orderings $LO(G)$, and the collection of all bi-orderings $O(G)$. Sikora [49] introduced a natural topology on the collection of all possible

orderings of a given group. The structure of this space varies widely depending on the group and has been studied, for example, in [14], [42], and [49]. In the much more general setting of orderable *magmas* (sets with a binary operation), the same topology may be defined, and the space is compact [14].

For countable orderable groups, the collection of possible orderings may be realized as paths on a subtree of the binary tree representing the characteristic functions of the respective positive cones. The collection of paths inherits the usual topology on paths of trees, and thus the space of orderings is always a closed subspace of Cantor space. For a computable group, this tree maybe computably constructed, and the space of orderings forms a Π^0_1 *class*, an computably closed subspace of Cantor space (for more, see [6]).

For computable fields, the collection of field orderings is always a Π^0_1 class, and there is a 1-1 Turing degree preserving bijection between any Π^0_1 class and the orderings of some orderable computable field [39]. There can be no such exact correspondence for groups. This is a consequence of a well-known result of Jockusch and Soare: there is a Π^0_1 class, with all elements pairwise Turing incomparable [37]. Solomon [53] considered abelian and nilpotent groups and was able to show that no such correspondence exists in even a weak way for these groups.

In general, the question of whether a given group is left-orderable is not an easy one to answer, but for a computable group, the problem can be reduced to determining whether some computable binary tree has an infinite path. A modification of a construction of Solomon [57] allows us to see this. We present first a non-computable construction of a tree, the paths of which are the left orderings admitted by a given countable group G.

Let $G - \{e_G\} = \{g_0, g_1, g_2, \ldots\}$, and for a node $\sigma \in 2^{<\omega}$, define $S_\sigma = \mathrm{sgr}(g_0^{\sigma(0)}, \ldots, g_{|\sigma|}^{\sigma(|\sigma|)})$.

Construction.

Stage 0. Add $\langle\ \rangle$ to T.

Stage s+1. For each leaf σ in T at the end of stage s, add $\sigma^\frown i$ to T (where $i = 0$ or 1) if and only if $S_{\sigma^\frown i}$ extends to a full ordering of G. (Here we are taking advantage of the condition in Theorem 4.1.1.)

This ends the construction.

This construction produces a leafless tree with paths corresponding exactly to the characteristic functions of positive cones of orderings in $LO(G)$. (The construction may be modified to produce a tree with paths corresponding to $O(G)$; it is only necessary to define S_σ as the *normal* subsemigroup generated by those elements.)

In the case that G is computable, we can produce a computable tree having paths corresponding to the orderings in $LO(G)$, but the construction requires some guessing and as a consequence will not be leafless. The construction for the tree of bi-orderings of a computable group appears in [57], and a hybrid of Solomon's construction and the one we have just seen yields the tree of left orderings of a computable group. (Essentially, it is only necessary to relax the normality condition in Solomon's construction.) We present the modified construction here for the reader's convenience.

Assume G is computable, and let $\{g_i\}_{i\in\omega}$ be a computable enumeration of $G - \{e_G\}$.

Construction.

Stage 0. Initialize $T = \{\langle\ \rangle\}$, and $S^0_{\langle\rangle} = \emptyset$.

Stage $s+1$. Now, for each leaf σ in T at the end of stage s, do the following: If $e_G \notin S_\sigma$, add both $\sigma^\frown 0$ and $\sigma^\frown 1$ to T.

Now, define $S_{\sigma^\frown 1}$ to be

$$S_\sigma \cup \{g_s\} \cup \{ab \mid a, b \in S_\sigma \cup \{g_s\}\},$$

and define $S_{\sigma^\frown 0}$ to be

$$S_\sigma \cup \{g_s^{-1}\} \cup \left\{ab \mid a, b \in S_\sigma \cup \{g_s^{-1}\}\right\}.$$

This ends the construction.

The tree T is easily seen to be computable. If X is a path in T, then the collection

$$P_X = \{g_i \mid X(i) = 1\} \cup \{g_i^{-1} \mid X(i) = 0\} \cup \{e_G\}$$

is easily seen to satisfy the conditions (1) and (2) above, guaranteeing it is the positive cone of a left ordering of G. Furthermore, it is clear that P_X has the same Turing degree as X.

Later, we will give a construction producing a computable tree, and the paths of this tree will be in Turing degree preserving bijective correspondence with the bi-orderings of a given computable semigroup. Because there is no simple corresponding notion of "positive cone" for a semigroup[2] the construction is necessarily somewhat different.

General criteria ensuring that a group be orderable or left-orderable have been studied extensively in the past (see, for example, [5], [25], and [38] for detailed expositions). Here we are concerned with the question of the algorithmic complexity of the orders on some computable group, and whether a computable group might admit a computable ordering of its elements[3].

[2]See [25] for a discussion of notions of cones for semigroups.

[3]For a discussion on the Turing degrees attained by orders of groups in the case when the group is abelian

In the following section, we see that conditions sufficient to ensure that a group admits a left ordering in each Turing degree similarly ensure there is an ordering of arbitrary *truth-table (tt-)*degree (these degrees are a refinement of the Turing degrees).

4.2 An ordering in every *tt*-degree

We begin by introducing two notions of reducibility that are strictly stronger than Turing reducibility.

A set A is *weak truth-table (wtt-) reducible* to a set B if $A \leq_T B$ and the computation of A from B is *predictable* in that the amount of information sufficient to compute A from B is itself computable. Formally, there is an $e \in \mathbb{N}$ and a computable function f so that

$$A(x) = \varphi_e^{B \upharpoonright f(x)}(x),$$

where $B \upharpoonright n$ indicates the restriction of the characteristic function of B to its first n bits. In this case, we write $A \leq_{wtt} B$.

A set A is *truth-table reducible* to a set B if $A \leq_{wtt} B$ and in addition, the algorithm wtt-computing A from B is entirely robust. In other words, it halts on every input even if the oracle is not giving correct information about B. Formally, for all $x \in \mathbb{N}$ and all $Y \subseteq \mathbb{N}$, the computation $\varphi_e^Y(x)$ must halt, though it need not take the value $A(x)$.

Obviously $\leq_{tt} \implies \leq_{wtt} \implies \leq_T$.

The *wtt*-degrees and *tt*-degrees are defined in exact analogy with the Turing degrees. Turing degrees may (but do not always) shatter into a countable infinity of *wtt*-degrees, and

or, more generally, 2-step nilpotent, see [57].

these in turn may shatter into smaller *tt*-degrees. For more information on these strong reducibilities and the induced upper-semilattices of strong degrees, see [43, 41].

In the following theorem, we give conditions sufficient to guarantee that the group G admits a left-ordering in each tt-degree. Groups satisfying these conditions admit left-orderings of their elements having arbitrary (in a strong sense) algorithmic complexity. This improves upon a result of [13], where the same conditions are shown to guarantee an ordering of arbitrary Turing degree.

Theorem 4.2.1. *Let G be a computable group, and $\mathcal{P}$ a c.e. family of finite subsets of $G - \{e\}$ satisfying the following conditions for every $p \in \mathcal{P}$.*

1. $e \notin \mathrm{sgr}(p)$,
2. $(\exists r_0, r_1 \in \mathcal{P})(\exists g \in G)[r_0, r_1 \supset p \wedge g \in r_0 \wedge g^{-1} \in r_1]$, *and*
3. $(\forall g \in G, g \neq e)(\exists r \in \mathcal{P})[r \supseteq p \wedge (g \in r \vee g^{-1} \in r)]$.

Then, G admits a left ordering in every tt-degree.

Proof. We construct a map $\mathcal{T} : 2^{<\omega} \to \mathcal{P}$ so that if $\sigma \sqsubset \tau$ then $\mathcal{T}(\sigma) \subset \mathcal{T}(\tau)$, and for any $X \subseteq \mathbb{N}$ we have $P_X^+ = \bigcup_n \mathcal{T}(X \upharpoonright n)$ has for every P_X^+ is a pure[4] subsemigroup not containing $\{e\}$. We will see that $P_X^+ \equiv_{tt} X$, and so $P_X = P_X^+ \cup \{e\}$ is a left-ordering of G of the same (arbitrary) *tt*-degree of X.

Let $G - \{e\} = \{g_0, g_1, \ldots\}$ and $\mathcal{P} = \{p_0, p_1, \ldots\}$ be computable enumerations of $G - \{e\}$ and $\mathcal{P}$, respectively.

Construction.

[4] A subsemigroup of a group G is *pure* if contains at most one of g and g^{-1} for each $g \in G$.

Stage 0. Set $\mathcal{T}(\langle\ \rangle) = p_0$.

Stage $s+1$. At the beginning of this stage, we have $\mathcal{T}$ defined on $2^{\leq s}$. For each σ of length s, find the first r_0 and r_1 in $\mathcal{P}$ and the first g in G witnessing the satisfaction of condition (2) for $p = \mathcal{T}(\sigma)$. We choose them so that $g \in r_1$ and $g^{-1} \in r_0$. Find the first g_{j_0} and g_{j_1} so that neither these elements nor their inverses are in r_0 and r_1, respectively. Let r_0' (and r_1', respectively) be the first element of $\mathcal{P}$ extending r_0 (r_1) containing g_{j_0} or $g_{j_0}^{-1}$ (g_{j_1} or $g_{j_1}^{-1}$). Such r_i' exist by condition (3).

Set $\mathcal{T}(\sigma^\frown i) = r_i'$ for $i = 0, 1$.

This completes the construction.

Now, let $\mathbf{x}$ be an arbitrary tt-degree, and X an infinite set with $deg_{tt}(X) = \mathbf{x}$. Define $P_X^+ = \bigcup_s \mathcal{T}(X \upharpoonright s)$.

Since for each s, $\mathrm{sgr}(\mathcal{T}(X \upharpoonright s))$ is a pure subsemigroup not containing e, we have that $P_X^+ = \mathrm{sgr}(P_X^+)$ is as well. (Note that this equality holds since if $a, b \in P_X^+$, but $ab \notin P_X^+$, then for some i, $g_i = (ab)^{-1} \in P_X^+$. All three elements will have entered by some stage j and this information is captured by some node $\sigma \sqsubset X$ of length j. The element of $\mathcal{P}$ mapped to this node by $\mathcal{T}$ will fail condition (1).) Furthermore, it is clear that for each $g_i \neq e$, either g_i or g_i^{-1} enters P_X^+ by stage $i+1$, so $P_X = P_X^+ \cup \{e\}$ defines a full left-ordering of G.

To see that $P_X \leq_{tt} X$, we observe the following. To determine if $g_i \in P_X$, we use the fact that for each σ of length $i+1$, either g_i or g_i^{-1} is in $\mathcal{T}(\sigma)$. So we have

$$g_i \in P_X \iff g_i \in \mathcal{T}(X \upharpoonright (i+1)).$$

Note that this is a tt-reduction as the initial segment of length $i+1$ of any set X is in the domain of $\mathcal{T}$.

For the reverse reduction, $X \leq_{tt} P_X$, we check to see if $i \in X$ via the following algorithm:

Construct $\mathcal{T}$ until the domain includes all nodes of length $i+1$.

If $P_X \supseteq \mathcal{T}(\sigma)$ for some σ of length $i+1$, then $i \in X$ if and only if the last bit of σ is 1.

If P_X does not contain the image of any σ of length $i+1$, output 0.

If P_X is in fact the partial order defined from the construction above, this algorithm yields the correct answer to the membership question on X. If it is not, it halts, but may not yield a correct answer. Thus the algorithm is a valid tt-reduction, and we have $\deg(P) = \mathbf{x}$.

□

4.3 Computable copies of $\mathbb{Z}^\omega$ with no computable ordering

In 1986, Downey and Kurtz [21] gave a construction of a computable copy G of $\mathbb{Z}^\omega = \oplus_\omega \mathbb{Z}$ admitting no computable ordering of its elements. Because the group is computable, the space of orders correspond to the paths in a computable tree via a construction presented by Solomon [57]. It is shown in [57] that any computable copy of this group has orders in every Turing degree $\geq \mathbf{0}'$. This can be viewed as a consequence of the fact that that the (computable) divisible closure of G, $\overline{G}$, is isomorphic to $\mathbb{Q}^\omega$, and any ordering on one of them, yields an ordering on the other of the same Turing degree [51]. As a vector space,

any computable (or c.e.) copy of $\overline{G}$ has a dependence algorithm of c.e. degree, and it is always possible to produce an ordering in any Turing degree $\geq_T$ the degree of the dependence algorithm[5] as there is a basis for the space in each of these degrees [39]. Thus any computable copy of $\mathbb{Z}^\omega$ must admit an ordering in each Turing degree $\geq \mathbf{0}'$.

However, for any such computable copy of this group, the tree of orderings is computable, so the Jockusch-Soare Low Basis Theorem [37] applies and we observe that there necessarily exists an ordering of low Turing degree.

It is natural to ask whether the low ordering of the group constructed in [21] is a consequence of $\overline{G}$ having a dependence algorithm of this same (or lower) low degree. (Of course $\overline{G}$ cannot have a computable dependence algorithm.) We imagine there are constructions of computable or c.e. copies of $\mathbb{Q}^\omega$ having dependence algorithms of degree $\mathbf{0}'$ (in fact, of degree $\mathbf{a}$ for any c.e. degree $\mathbf{a}$), but to this author's knowledge no assessment of the algorithmic complexity of orderings of their elements has been made. (Perhaps each existing example admits orderings of every Turing degree. The computable examples must all have an ordering of low degree.)

In the case of a computable abelian group with infinite rank, all Turing degrees $\geq \mathbf{0}'$ occur in the spectrum, and by the low basis theorem, an ordering of low degree must exist. Here we construct a computable abelian group of infinite rank admitting no computable orderings having the additional property its computable divisible closure has a complete dependence algorithm.

In the following construction, we produce a computable copy of $\mathbb{Z}^\omega$ so that the dependence

[5]The *dependence algorithm* for a computable or c.e. vector space is the collection of all finite dependant subsets of its universe, suitably coded into the natural numbers.

algorithm in the computable divisible closure must be complete (i.e. it must compute the halting set, K). The group is computable so, by the discussion above, it must have an ordering of low Turing degree. We conclude that a torsion-free abelian group, and indeed the copy of the vector space over $\mathbb{Q}$ that is its computable divisible closure, may admit orderings of properly lower Turing degree than the corresponding dependence algorithm.

It is not known whether this spectrum of degrees of orderings for this group is closed upward in the Turing degrees.

Theorem 4.3.1. *There is a computable copy V of $\mathbb{Q}^\omega$ having no computable ordering and a complete dependence algorithm.*

Proof. We obtain a computable copy G of $\mathbb{Z}^\omega$, the computable divisible closure of which is the group V we seek.

The domain of G will be $\mathbb{N}$, and we simultaneously construct the multiplication table, $\cdot$, of G and the isomorphism f to $\mathbb{Z}^\omega$. The multiplication table will be computable, the isomorphism will not be. In fact, f will be 2-c.e.

We will use 0 to denote the identity of G and $\mathbb{Z}^\omega$.

We aim to satisfy the following requirements for all e:

C_e: If $\langle i, j\rangle = e$, then $i \cdot j$ is defined.

I_e: There is an i so that $e \cdot i = 0$.

N_e: φ_e does not compute an ordering of G.

D_e: $e \in K$ iff $5e$ and $5e + 1$ are dependent.

The priority ordering is $N_0 < D_0 < C_0 < I_0 < N_1 < \dots$. Let $K_0 \subseteq K_1 \subseteq \cdots$ be a computable approximation of the halting set K by finite sets.

A requirement R_e is *active at stage s* if it is the least requirement satisfying the following. R_e is

C_e and $e = \langle i, j \rangle$, $f(i)$ and $f(j)$ exist, and there is no k so that $i \cdot j = k$.

I_e and $f(e)$ is defined but there is no k so that $e \cdot k = 0$.

D_e and (1) f is not defined on $5e, 5e+1$, or (2) f is defined on $5e$ and $5e+1$, $e \in K_s$, and e has not *been coded* (defined later).

N_e and (1) f is not defined on $5e+2, 5e+3$, or (2) f is defined on $5e+2$ and $5e+3$, $\varphi_{e,s}(5e+2)\downarrow$, $\varphi_{e,s}(5e+3)\downarrow$, and e has not *been defeated* (defined later).

Construction.

Stage 0. Set $f(0) = 0$. That's it.

Stage s+1. Let $n = \max(\mathrm{dom}(f))$, and R_e the highest priority requirement that is active.

- If R_e is C_e, let k be the least natural number $\geq n$ that is $5m+4$ for some $m \in \omega$. Set $f(k) = f(i) + f(j)$ where $\langle i, j \rangle = e$. Now $i \cdot j = k$.

- If R_e is I_e, let k be the least natural number $\geq n$ that is $5m+4$ for some $m \in \omega$. Set $f(k) = -f(e)$. Now $k \cdot e = 0$.

- If R_e is D_e and is active by virtue of part (1) of the condition, define $f(5e) = 0^{4e}1$ and $f(5e+1) = 0^{4e+1}1$. If D_e is active by virtue of (2), these definitions have already

been made, and $e \in K_s$. We now take action so that e *is coded* into the group. For $k \in \mathrm{dom}(f)$, we write $k[\ell]$ for the ℓth bit of $f(k)$.

Let $m = 1 + \max\{k[4e+1] | k \in \mathrm{dom}(f)\}$ and make the following update to f. Note that this redefinition of the isomorphism does not affect the part of the multiplication table for G that has been defined so far. For each $k \in \mathrm{dom}(f)$, set

$$f(k)[i] = \begin{cases} f(k)[i], & i \neq 4e+1, 4e+2 \\ 0, & i = 4e+2 \\ f(k)[4e+1] + 2mf(k)[4e+2], & i = 4e+1. \end{cases}$$

Here, the f on the left of the equals sign is the new value, and that on the right is the old value. Now $f(5e+1) = 2mf(5e)$, and $5e$ and $5e+1$ are dependent.

- If R_e is N_e and is active by virtue of part (1) of the condition, define $f(5e+2) = 0^{4e+2}1$ and $f(5e+3) = 0^{4e+3}1$. If N_e is active by virtue of (2), these definitions have already been made, and $\varphi_{e,s} \downarrow$ for both $5e+2$ and $5e+3$. We interpret φ_e as computing an ordering $<_e$ in the following sense. First, it must be total, and supposing that, we take $\varphi_e(a) = 0$ to mean that $a <_e 0$, and $\varphi_e(a) \neq 0$ to mean that $0 <_e a$.

Case 1. Both $\varphi_e(5e+2)$ and $\varphi_e(5e+3)$ are non-zero. Then both are positive for $<_e$, and the idea is to update f so that an appropriate linear combination of them is zero. Let $m = 1 + \max\{k[4e+3] \mid k \in \mathrm{dom}(f)\}$, and set

$$f(k)[i] = \begin{cases} f(k)[i], & i \neq 4e+3, 4e+4 \\ 0, & i = 4e+4 \\ f(k)[4e+1] - 2mf(k)[4e+2], & i = 4e+1. \end{cases}$$

Now $f(5e+3) + 2mf(5e+2) = 0$, which is inconsistent with $<_e$.

Cases 2–4. In all other cases we make similar arrangements, always producing a map so that the group constructed is inconsistent with the possible ordering being considered.

The appropriate lemmas are easy to check, and are much like those in [21], except that it is necessary to check the degree of the dependence algorithm, and we proceed with this task.

Let D be the dependence algorithm for V, the computable divisible closure of $G \cong \mathbb{Z}^\omega$. Let g be the computable embedding from G into V.

Lemma 4.3.2. $D =_T K$.

Proof. For any computable (or c.e.) vector space, the dependence algorithm is c.e., so always $D \leq_T K$.

To check to see if $n \in K$, ask D if $g(5n)$ and $g(5n+1)$ are dependent. They are if and only if $n \in K_s$ for some s. □

□

Corollary 4.3.3. *There is a computable copy of $\mathbb{Z}^\omega$ admitting no computable ordering, for which the dependence algorithm of the computable divisible closure is complete.*

When an orderable group has a computable copy admitting no computable orderings, there are interesting consequences for the (topological) space of (left- or bi-) orderings of that group. We gave a non-effective algorithm for obtaining a tree with paths corresponding exactly to the left orderings of a countable group. In the case that the group is computable, this algorithm can be modified and a computable tree having paths in Turing degree preserving correspondence to left-orderings of the group may be obtained. In [57], an effective construction of the tree of bi-orderings for a computable group is given. In all cases, these collections of paths inherit the natural topology on the paths of $2^{<\omega}$, and thus form a subspace of Cantor space.

The space of left- and bi-orderings of groups have been studied, for example, in [12, 13, 42, 49]. An orderable group has either finitely many or uncountably many left-orderings of its elements. For bi-orderings, a countable infinity of orderings becomes accessible [4]. Of interest to many is the question of whether, for a given group, either the space of left-orderings or the space of bi-orderings is homeomorphic to the Cantor space in which it is embedded.

Because an isolated path in a computable tree must be computable, we can make the observation that a tree T having no computable paths must have that $[T]$ (if it is not empty) is homeomorphic to the Cantor space. (A non-empty closed subspace of Cantor space having no isolated points is perfect.) Thus, a countable group having a computable copy admitting no computable orderings must (since these orderings correspond to the paths in the appropriately obtained tree) have that its space of orderings is a Cantor space. The topology of the space of orderings will not change under isomorphism from one copy of the group to another, so in the event that such a copy can be found, this describes the space of

orderings for that group's isomorphism class.

For example, this observation, together with the copy of $\mathbb{Z}^\omega$ given in Corollary 4.3.3 (or the Downey–Kurtz group), make short work of the fact that $\mathbb{Z}^\omega$ has a Cantor space as its space of orderings. (Both of these give a computable copy admitting no computable orderings.) This result was originally proved in [12].

4.4 Trees of orderings of semigroups

As mentioned above, for a computable group, we can construct computable trees with paths exactly corresponding to left- or bi-orderings of the group. Such a collection of paths is called a Π^0_1 class. A great body of research exists in the study of Π^0_1 classes (see, for example, [6]). A collection of sets of natural numbers forms a Π^0_1 class exactly when their characteristic functions may be realized as the infinite paths through some computable binary tree. Given that the collection of left- or bi-orderings of a computable group may be realized as such a class, and basis theorems (for example the Jockush-Soare Low Basis Theorem mentioned above) about Π^0_1 classes may thus be applied to the collection of orderings, we see the utility in this viewpoint.

Consequently, it is worthwhile to cast the space of orderings of other computable algebraic structures onto the collection of paths through some computable tree. Below we obtain such a tree for a computable semigroup, though because there is no single viable notion of a positive cone for an ordered semigroup (as there is in the case of groups), we cannot adapt the construction for groups directly.

Recall that a *semigroup* is a set with an associative operation. An *ordered semigroup* is

a semigroup S together with an order, $<$, on its elements satisfying for all $x \in S$

$$a < b \implies (xa < xb) \wedge (ax < bx).$$

Theorem 4.4.1. *Let S be a computable semigroup. There is a computable tree T and a Turing degree preserving bijection between the paths of T and the bi-orderings on S.*

Proof. Let $\{p_0, p_1, \ldots\}$ be a computable listing of non-diagonal elements of $S \times S$.

Paths in T will roughly correspond to characteristic functions of index sets of sets of pairs of elements of S: For $f \in [T]$, define $<_f = \{p_i \mid f(i) = 1\} \cup \{p_i^* \mid f(i) = 0\}$, where $p_i^* = (b, a)$ when $p_i = (a, b)$.

For an ordering $<$ of S, define

$$f_<(i) = \begin{cases} 1, & p_i = (a, b) \wedge a < b \\ 0, & p_i = (a, b) \wedge b < a. \end{cases}$$

It is clear that this is a bijection and preserves Turing degree.

In the construction that follows we will use approximations of orderings, $\leq_\sigma$, defined at each stage and corresponding to nodes added to the tree.

Construction:

Stage 0. Set $T_0 = \{\langle\ \rangle\}$, and $\leq_{\langle\ \rangle} = \emptyset$.

Stage $s + 1$. For each node, σ of T_s having length s, do the following:

- If either $\leq_\sigma$ contains (a, b) and (b, a) for any $a \neq b$, or if $(a, b) = p_i$ and $(b, a) = p_j$ for $i, j < s$ and $\leq_\sigma$ contains neither, then this node is terminal.

- Otherwise, add both $\sigma^\frown 1$ and $\sigma^\frown 0$ to T_{s+1} and

    1. add p_s to $\leq_{\sigma^\frown 1}$ and p_s^* to $\leq_{\sigma^\frown 0}$,

    2. add (s, s) to $\leq_{\sigma^\frown i}$ for $i = 0, 1$,

    3. for all (a, b) and (c, d) in $\leq_{\sigma^\frown i}$, add (ac, bd) and (ca, db) to $\leq_{\sigma^\frown i}$, and

    4. when (a, b) and (b, c) are in $\leq_{\sigma^\frown i}$, add (a, c) to $\leq_{\sigma^\frown i}$.

This completes the construction.

It is clear that T is computable.

Lemma 4.4.2. *Let $f \in [T]$. Then $<_f$ is an ordering.*

Proof. Suppose $f \in [T]$

1. $a \not<_f a$. For no i is $p_i = (a, a)$.

2. For $a \neq b$, exactly one of $a <_f b$ and $b <_f a$ holds. If $a \neq b$, then $p_i = (a, b)$ for some $i \in \omega$, and f takes a value on i. If $p_j = (b, a)$, f must take the opposite value on j, for it is a path, and otherwise, $f \upharpoonright j + 1$ would be terminal.

3. If $a <_f b$ and $b <_f c$, then $a <_f c$. Assume otherwise, and that $p_i = (a, b)$, $p_j = (b, c)$, and $p_k = (c, a)$. Let $s = \max\{i, j, k\}$. Then $f \upharpoonright s + 1$ is terminal, but again, f is a path.

4. If $a <_f b$ and $x \in S$, then $ax <_f bx$ and $xa <_f xb$. Let $p_i = (a, b)$, and $s = \max\{i, x\}$. Steps 2 and 4 in the construction provide that $p_j = (ax, bx)$ and $p_k = (xa, xb)$ are added to $\leq_{f\upharpoonright s+1}$. It follows (as in 2 just above) that f must take the value 1 on both j and k.

□

Lemma 4.4.3. *Let $<$ be an ordering of S. Then $f_< \in [T]$.*

Proof. Let $<$ be an ordering of S and define $f_<$ as described above.

Claim 4.4.4. $\lneq_{f_< \restriction n} =_{def} \leq_{f_< \restriction n} - \{(x,x)\}_{x \in S} \subset <$.

Proof. By induction. Of course this is true for $n = 0$.

Assume it is true for n. Then $\lneq_{f_< \restriction n}$ is part of a consistent ordering of S, so for no distinct x and y will we have (x,y) and (y,x) in $\lneq_{f_< \restriction n}$. Thus, at stage $n+1$, the node $f_< \restriction n$ is extended in both directions, and either p_n or p_n^* is added to $\lneq_{f_< \restriction n+1}$. Assume, without loss of generality, that $f_<(n) = 1$ and $p_n = (a,b)$ is added. By definition of $f_<$, we must have $a < b$. Similarly, all other pairs added to $\lneq_{f_< \restriction n+1}$ at this stage must be part of any consistent ordering of S containing $\lneq_{f_< \restriction n}$ and p_n. Thus $\lneq_{f_< \restriction n+1} \subset <$, and we have the claim. □

The proof we have given for the claim also establishes the lemma.

□

□

In the case of groups, the natural topology on this tree (though the paths are not positive cones) coincides with that which would be obtained from the trees having paths corresponding to positive cones. This is seen immediately from the dual characterizations of the topology on the space of ordering given by Sikora in [49], and the proofs of their equivalence[6].

[6]The topology was presented as induced from a metric on the orderings, as well as being characterized as the topology obtained from a particular subbasis. (The subbasis consisted of sets of the form $U_{(a,b)} = \{< \mid <$ is an ordering of the group, and $a < b\}$.) For the countable case, these are easily seen to be equivalent to the topology inherited from the tree.

Thus, the basis theorems of Π^0_1 classes apply as well to semigroups, and we can conclude that a computable copy of an orderable semigroup will admit a low ordering of its elements. Furthermore, and as in the case with groups, we can conclude that the space is isomorphic to Cantor space when there is a computable copy having no computable ordering.

Bibliography

[1] B. Anderson and J. Hirst, Partitions of trees and ACA'0, (preprint).

[2] S. Arwon, Y. Kim, and A. Rhemtulla, On finitely determined total orders, *74th Workshop on General Algebra*, June 2007, Tampere University of Technology, Tampere, Finland.

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

[4] R.N. Buttsworth, A family of groups with a countable infinity of full orders, *Bulletin of the Australian Mathematics Society* 4 (1971) 97–104.

[5] R.T. Botto Mura and A.H. Rhemtulla, *Orderable groups*, Lecture Notes in Pure and Applied mathematics 27, Marcel Dekker, Inc., New York–Basel (1977).

[6] D. Cenzer and J.B. Remmel, *Effectively closed sets*, in preparation.

[7] D. Cenzer and J.B. Remmel, Proof-Theoretic Strength of the Stable Marriage Theorem and Other Problems *Reverse Mathematics*, edited by S. Simpson, ASL Lecture Notes in Logic 21, AK Peters (2005), 67–103.

[8] P.A. Cholak, C.G. Jockusch, and T.A. Slaman, On the strength of Ramsey's theorem for pairs, *The Journal of Symbolic Logic* 66 (2001), no. 1, 1–55.

[9] J. Chubb, A. Frolov, V.S. Harizanov, Degree spectra of the successor relation of computable linear orderings. *Archive for Mathematical Logic* 48 (2009) 7-13.

[10] J. Chubb, J. Hirst, and T. McNicholl, Reverse mathematics, computability, and partitions of trees. *Journal of Symbolic Logic* 74 (2009) 201-215.

[11] P.F. Conrad, Right ordered groups, *Mich. Math Journal* 6 (1959), 267–275.

[12] M. Dabkowska, *Turing Degree Spectra of Groups and Their Spaces of Orders*, Ph.D. dissertation, George Washington University, 2006.

[13] M. Dabkowska, M. Dabkowski, V. Harizanov, and A. Togha, Spaces of orders and their Turing degree spectra, submitted to *Annals of Pure and Applied Logic*.

[14] M. Dabkowska, M. Dabkowski, V. Harizanov, J. Przytycki, and M. Veve, Compactness and spaces of left orders, *Journal of Knot Theory and Its Ramifications* 16 (2007), pp. 257-266.

[15] M.K. Dabkowski, J.Przytycki, A.A. Togha, Non-left-orderable 3-manifold groups, *Canadian Math. Bull.*, 48(1), 2005, 32–40.

[16] M.D. Davis, R. Sigal, and E.J. Weyuker, *Computability, Complexity, and Languages*, Morgan Kauffman, San Francisco, 1994.

[17] R.G. Downey, S.S. Goncharov and D. Hirschfeldt, Degree spectra of relations on Boolean algebras, *Algebra and Logic* 42 (2003), pp. 105–111.

[18] R.G. Downey, S. Lempp, G. Wu, On the complexity of the successivity relation in computable linear orderings, *in preparation*.

[19] R.G. Downey, Every recursive Boolean algebra is isomorphic to one with incomplete atoms, *Annals of Pure and Applied Logic* 60 (1993), pp. 193–206.

[20] R. Downey and R. Solomon, Reverse mathematics, archimedean classes, and Hahn's Theorem, *Reverse Mathematics 2001*, edited by Stephen Simpson, AK Peters, 2005, 147–163.

[21] R.G. Downey, and S.A. Kurtz, Recursion theory and ordered groups. *Annals of Pure and Applied Logic* 32 (1986), no. 2, 137–151.

[22] R.G. Downey and M.F. Moses, Recursive linear orders with incomplete successivities, *Transactions of the American Mathematical Society* 326 (1991), pp. 653–668.

[23] V.D. Dzgoev, and S.S. Goncharov, Autostability of models, *Algebra and Logic* 19 (1980), pp. 45–58 (Russian), pp. 28–37 (English translation).

[24] W. Deuber, R.L. Graham, H.J. Prömel, and B. Voigt, A canonical partition theorem for equivalence relations on $\mathbf{Z}^t$, *Journal of Combinatorial Theory. Series A* 34 (1983), 331–339.

[25] L. Fuchs, *Partial ordered algebraic systems*, International series of monographs on pure and applied mathematics, Oxford, New York, Pergamon Press, 1963.

[26] S. Goncharov, S. Lempp, and R. Solomon, The computable dimension of ordered abelian groups, *Advances in Mathematics* 175, 2002, 102–143.

[27] V.S. Harizanov, Turing degrees of certain isomorphic images of recursive relations, *Annals of Pure and Applied Logic* 93 (1998), 103–113.

[28] V.S. Harizanov, Pure Computable Model Theory, *Handbook of Recursive Mathematics: Volume 1: Recursive Model Theory (Studies in Logic and the Foundations of Mathematics)*. Ed. Iurii Leonidovich Ershov. North Holland, 1998

[29] V.S. Harizanov, Uncountable degree spectra, *Annals of Pure and Applied Logic* 54 (1991), 255–263.

[30] V.S. Harizanov and R. Miller, Spectra of structures and relations, *Journal of Symbolic Logic* 72 (2007) 1, 324-348.

[31] V.S. Harizanov, Some effects of Ash-Nerode and other decidability conditions on degree spectra, *Annals of Pure and Applied Logic* 55 (1991), pp. 51–65.

[32] V.S. Harizanov, *Degree Spectrum of a Recursive Relation on a Recursive Structure*, PhD dissertation, University of Wisconsin, Madison, 1987.

[33] D.R. Hirschfeldt and R.A. Shore, Combinatorial principles weaker than Ramsey's Theorem for pairs, *The Journal of Symbolic Logic* 72 (2007) 171–206.

[34] J.L. Hirst, *Combinatorics in Subsystems of Second Order Arithmetic*, The Pennsylvania State University, Ph.D. Thesis (1987).

[35] T.L. Hummel, Effective versions of Ramsey's theorem: avoiding the cone above $\mathbf{0}'$, *The Journal of Symbolic Logic* 59 (1994), no. 4, 1301–1325.

[36] C.G. Jockusch, Ramsey's theorem and recursion theory. *Journal of Symbolic Logic* 37 (1972) 268–280.

[37] C.G. Jockusch, Jr. and R.I. Soare, Π^0_1 classes and degrees of theories. *Transactions of the Amererican Mathematical Society* 173 (1979), pp. 33-56.

[38] V.M. Kopytov and N.Y. Medvedev, *Right-Ordered Groups*, Siberian School of Algebra and Logic, Consultants Bureau, New York, 1996.

[39] G. Metakides, A. Nerode, Effective content of field theory, *Annals of Mathematical Logic* 17 (1979), no. 3, 289–320.

[40] K.R. Milliken, A Ramsey theorem for trees, *Journal of Combinatorial Theory* Series A 26 (1979), no. 3, 215–237.

[41] J. Mohrherr, Index Sets and Truth-Table Degrees in Recursion Theory, PhD Dissertation, University of Illinois at Chicago, 1982.

[42] A. Navas, On the dynamics of (left) orderable groups, arXiv:0710.2466v5

[43] P. Odifreddi, *Classical Recursion Theory*, North-Holland, Amsterdam, 1989.

[44] J.B. Remmel, Recursive isomorphism types of recursive Boolean algebras, *Journal of Symbolic Logic* 46 (1981), pp. 572–594.

[45] J.B. Remmel, Recursively categorical linear orderings, *Proceedings of the American Mathematical Society* 83 (1981), pp. 387–391.

[46] A. Rogalski, Reverse mathematics on lattice ordered groups, Ph.D. thesis, University of Connecticut, 2007.

[47] J. Rosenstein, *Linear Orderings*, Academic Press, 1982.

[48] D. Seetapun and T.A. Slaman, On the strength of Ramsey's theorem, *Notre Dame Journal of Formal Logic* Special Issue: Models of arithmetic, 36 (1995), no. 4, 570–582.

[49] A.S. Sikora, Topology on the spaces of orderings of groups, *Bulletin of the London Mathematical Society* 36, (2004) 519-526.

[50] S.G. Simpson, *Subsystems of second order arithmetic*, Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1999.

[51] R.L. Smith, Two theorems on autostability in p-groups, in *Logic Year 1979–80*, vol. 859 of *Lecture Notes in Mathematics*, A. Dold and B. Eckmann, eds. Springer–Verlag, New York, 302–311.

[52] R.I. Soare, *Recursively Enumerable Sets and Degrees, A Study of Computable Functions and Computably Generated Sets*, Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1987.

[53] R. Solomon, Π^0_1 classes and orderable groups. *Annals of Pure and Applied Logic* 115(2002) 279–302.

[54] R. Solomon, Ordered groups: A case study in reverse mathematics, *Bulletin of Symbolic Logic* 5 (1999) 45–58.

[55] R. Solomon, $\Pi^1_1 - CA$ and order types of countable ordered groups *Journal of Symbolic Logic* 66 (1), March 2001, 192–206.

[56] R. Solomon, Reverse mathematics and ordered groups *Notre Dame Journal of Formal Logic* 39 (2), Spring 1998, 157–189.

[57] R. Solomon, Reverse math and ordered groups, Ph.D. Dissertation, Cornell University, 1998.

[58] E. Specker, Ramsey's Theorem does not hold in recursive set theory, *Logic Colloquium '69, Studies in Logic and the Foundations of Mathematics*, North-Holland, Amsterdam, 1971.

[59] J. Stern, A Ramsey theorem for trees, with an application to Banach spaces, *Israel Journal of Mathematics* 29 (1978) 179–188.