Reducibility, Degree Spectra, and Lowness in Algebraic Structures

by

Rebecca M. Steiner

A dissertation submitted to the Graduate Faculty in Mathematics in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

2012

This manuscript has been read and accepted for the Graduate Faculty in Mathematics in satisfaction of the dissertation requirements for the degree of Doctor of Philosophy.

Date	Russell G. Miller Chair of Examining Committee
Date	Józef Dodziuk Executive Officer

Supervisory Committee:

Russell Miller

Roman Kossak

Joel David Hamkins

THE CITY UNIVERSITY OF NEW YORK
2012

Abstract

Reducibility, Degree Spectra, and Lowness
in
Algebraic Structures

by

Rebecca M. Steiner

Advisor: Russell G. Miller

This dissertation addresses questions in computable structure theory, which is a branch of mathematical logic hybridizing computability theory and the study of familiar mathematical structures. We focus on algebraic structures, which are standard topics of discussion among model theorists. The structures examined here are fields, graphs, trees under a predecessor function, and Boolean algebras.

For a computable field F, the *splitting set* S_F of F is the set of polynomials in $F[X]$ which factor over F, and the *root set* R_F of F is the set of polynomials in $F[X]$ which have a root in F. Results of Fröhlich and Shepherdson from 1956 imply that for a computable field F, the splitting set S_F and the root set R_F are Turing-equivalent. Much more recently, in 2010, R. Miller showed that for algebraic fields, if we use a finer measure, the root set actually has slightly higher complexity: for algebraic fields

F, it is always the case that $S_F \leq_1 R_F$, but there are algebraic fields F where we have $R_F \not\leq_1 S_F$.

In the first chapter, we compare the splitting set and the root set of a computable algebraic field under a different reduction: the bounded Turing (bT) reduction. We construct a computable algebraic field for which $R_F \not\leq_{bT} S_F$. We also define a *Rabin embedding* g of a field into its algebraic closure, and for a computable algebraic field F, we compare the relative complexities of R_F, S_F, and $g(F)$ under m-reducibility and under bT-reducibility.

Work by R. Miller in 2009 proved several theorems about algebraic fields and computable categoricity. Also in 2009, A. Frolov, I. Kalimullin, and R. Miller proved some results about the degree spectrum of an algebraic field when viewed as a subfield of its algebraic closure.

In the second chapter, we show that the same computable categoricity results also hold for finite-branching trees under the predecessor function and for connected, finite-valence, pointed graphs, and we show that the degree spectrum results do not hold for these trees and graphs. We also offer an explanation for why the degree spectrum results distinguish these classes of structures: although all three structures are algebraic structures, the fields are what we call *effectively* algebraic.

Every low_n Boolean algebra, for $1 \leq n \leq 4$, is isomorphic to a computable Boolean algebra. It is not yet known whether the same is true for $n > 4$. However, it is known that there exists a low_5 subalgebra of the computable atomless Boolean algebra which,

when viewed as a relation on the computable atomless Boolean algebra, does not have a computable copy.

In the third chapter, we adapt the proof of this recent result to show that there exists a low_4 subalgebra of the computable atomless Boolean algebra $\mathcal{B}$ which, when viewed as a relation on $\mathcal{B}$, has no computable copy. This result provides a sharp contrast with the one which shows that every low_4 Boolean algebra has a computable copy. That is, the spectrum of the subalgebra as a unary relation can contain a low_4 degree without containing the degree $\mathbf{0}$, even though no spectrum of a Boolean algebra (viewed as a structure) can do the same. We also point out that unlike Boolean algebras as structures, which cannot have nth-jump degree above $\mathbf{0}^{(n)}$, subalgebras of $\mathcal{B}$ considered as relations on $\mathcal{B}$ can have nth-jump degree strictly bigger than $\mathbf{0}^{(n)}$.

Acknowledgements

Many thanks to:

- my parents, who think I'm weird but love me anyway;
- Józef Dodziuk, for his reassurance that I'm not any more weird than any other mathematician;
- Carl Goodman, the first to cause my mouth to hang open in fascination with mathematics;
- Nick Metas, who showed me what it means to be passionate about something;
- Wallace Goldberg, for keeping me employed during graduate school;
- Starbucks Coffee, for keeping me caffeinated during graduate school;
- Roman Kossak, who made me learn TeX when I had time to learn TeX;
- Ken Kramer, the oracle for all things algebraic;
- Bob Soare, who (quite literally) wrote the book;
- Julia Knight, for her expert knowledge and unexpected encouragement;

- Alan Sultan, who knew I could, even when I wasn't sure;

and especially

- Russell Miller, for his infinite patience.

Contents

List of Figures

Introduction

Is there an algorithm that will take as input a description of a formal language and a mathematical statement in the language and produce as output either "true" or "false" according to whether the statement is true or false? This question was posed by the German mathematician David Hilbert in 1928 and is known as the *Entscheidungsproblem* (German for "decision problem"). In 1936 Alonzo Church and Alan Turing (independently), influenced by techniques used by Gödel in the proofs of his incompleteness theorems, proved that the answer to the Entscheidungsproblem is "no;" there is no algorithm that can correctly decide the truth of an arbitrary mathematical statement.

Computability grew out of the work of Church and Turing in an attempt to formalize the informal notion of "effective calculation." The form of computability theory we study today was introduced by Turing in 1936. A *Turing machine* is a computer which, provided with a finite program and a natural number n, runs the program on n. The machine may halt after finitely many steps of computation, or it may never halt. If and when the machine halts, it yields an output $f(n)$. A function f from the natural numbers to themselves is *Turing computable* (hereafter simply called

computable) if there is a Turing machine which, on input n, halts with output $f(n)$. The function f is called *total* if it halts on every input.

A set B of natural numbers is *computable* if there is a Turing machine which, on input n, halts with output 1 if n is in B and halts with output 0 if n is not in B. As an example, the set of prime numbers is a computable set; there is a finite algorithm that effectively decides whether an arbitrary natural number is prime. The halting problem, which can be described as the set of indices of Turing programs that halt on input 0, is the most classical example of a noncomputable set.

An *oracle Turing machine* is a machine which, in addition to doing all the things an ordinary Turing machine does, is able to ask questions about membership in a particular fixed set of natural numbers called an *oracle*, to help it carry out its computation. The oracle provides a correct answer to each such question, whether or not the oracle is a computable set, and so oracle Turing machines can compute more sets than ordinary Turing machines can.

A set A of natural numbers is *Turing-reducible* to a set B of natural numbers, written $A \leq_T B$, if there is a Turing machine that correctly answers questions about membership in A when B is used as the oracle set. We say that A *is computable from* B or that B *computes* A. If both $A \leq_T B$ and $B \leq_T A$, we write $A \equiv_T B$, and we say that A and B are *Turing-equivalent*, or that they *have the same Turing degree*. A Turing degree is just an equivalence class under Turing reducibility. The degree of a set A is less than the degree of a set B if and only if $A <_T B$. The higher the degree

of a set, the harder it is to decide membership in that set.

The degree which contains precisely the computable sets is the least degree and is called $\mathbf{0}$. For any set B, the set B' is the set of indices of oracle Turing machines which halt on input 0 when using B as an oracle. The degree of B' is called the *jump* of the degree of B, and is strictly bigger in the semi-lattice of degrees than the degree of B: $\deg(B) <_T \deg(B')$. The degree $\mathbf{0}'$ is the jump of the degree $\mathbf{0}$; $\mathbf{0}'$ happens to be the degree of the halting problem.

Turing reducibility isn't the only way to compare the relative computability of two sets. There is a much stronger reducibility, called m-reducibility, which can separate the complexities of two sets which happen to be Turing-equivalent. A set A is *m-reducible* to a set B, written $A \leq_m B$, if there is a total computable function f such that for every $n \in \omega$, $n \in A$ if and only if $f(n) \in B$. If there happens to be a one-to-one such f, then we say that A is *1-reducible* to B.

We introduce one more reducibility: the bounded Turing reducibility. A set A is bounded Turing-reducible to a set B, written $A \leq_{bT} B$, if A is Turing-reducible to B *and* there is a total computable function g such that for each input n, B-membership can only be determined for natural numbers which are less than or equal to $g(n)$. It is essentially Turing reducibility, but there is a computable bound on how much of the oracle can be used. Bounded Turing reducibility is stronger than Turing reducibility (two sets can be Turing-equivalent but not bT-equivalent) but weaker than m-reducibility (two sets can be bT-equivalent but not m-equivalent).

The first chapter of this dissertation discusses all three of these reducibilities in the context of algebraic fields. For a computable field F, the *splitting set* S_F of F is the set of polynomials in $F[X]$ which factor over F, and the *root set* R_F of F is the set of polynomials in $F[X]$ which have a root in F. We can think of these sets as subsets of ω if we use the canonical bijection of $\omega^{<\omega}$ onto ω to code polynomials as natural numbers. Fröhlich and Shepherdson [8] essentially showed that for a computable field F, the splitting set S_F and the root set R_F are Turing-equivalent. More than fifty years later, R. Miller [21] showed that for algebraic fields, if we use the finer measure of 1-reducibility, the root set actually has slightly higher complexity: for algebraic fields F, it is always the case that $S_F \leq_1 R_F$, but there are algebraic fields F where we have $R_F \not\leq_1 S_F$. Here, we examine the relative complexities of R_F and S_F for a computable algebraic field F under bounded Turing reducibility. Because we always have $S_F \leq_1 R_F$, we must always have $S_F \leq_{bT} R_F$, but we construct a computable algebraic field F for which $R_F \not\leq_{bT} S_F$. The construction involves a fair amount of Galois theory, which the author had to learn expressly for this purpose. It turns out that the lack of bT-reduction from R_F to S_F for some F is equivalent to a purely algebraic fact; indeed, we show that if this fact weren't true, we would always have $R_F \leq_{bT} S_F$. We also introduce a particular kind of computable field-embedding, called a *Rabin embedding*, of an algebraic field into its algebraic closure; for a computable algebraic field F and an arbitrary Rabin embedding g of F into its algebraic closure, we compare the relative complexities of R_F, S_F, and the image $g(F)$ under m-reducibility and under bT-reducibility.

The second chapter of this dissertation takes the concept of algebraicity, which we use in the first chapter in the form of algebraic fields, and examines it under a microscope. Algebraic fields, finite-branching trees under a predecessor function, and connected finite-valence graphs with a constant node are all examples of algebraic structures – each element of each of these kinds of structures has finite orbit under automorphisms. These three types of structures have many properties in common because of their algebraicity, but there is something extra that algebraic fields have that the finite-branching trees and finite-valence graphs don't have. R. Miller [20] has some recent results about computable categoricity properties of algebraic fields; for example, he showed that there is a computable algebraic field which is not computably categorical, yet has a splitting algorithm, and that there is a computable algebraic field which is not even $\mathbf{0}'$-computably categorical. Here we show that the corresponding statements for finite-branching trees under predecessor (and for connected finite-valence graphs with a constant node) are also true: There is a computable finite-branching tree under predecessor which is not computably categorical, yet has computable branching function, and there is a computable finite-branching tree under predecessor which is not even $\mathbf{0}'$-computably categorical. On the other hand, A. Frolov, I. Kalimullin, and R. Miller [9] proved some theorems about the degree spectrum of an algebraic field when viewed as a subfield of its algebraic closure, whose corresponding results for finite-branching trees under predecessor and for connected finite-valence graphs with a constant node don't hold. For example, there are computable algebraic fields whose degree spectrum, when viewed as a subfield of the algebraic closure of $\mathbb{Q}$,

contains exactly one degree, but for every infinite computable finite-branching tree under predecessor (resp. connected finite-valence graph with a constant node), its degree spectrum, if we consider that tree as a subtree of the tree $\omega^{<\omega}$ (resp. a subgraph of the countable random graph), always contains infinitely many degrees. Why do the computable categoricity results carry over while the degree spectra results do not? We attribute this phenomenon to a property of computable algebraic fields which we call "effective algberaicity," which, briefly, means that not only does every element of such a field have finite orbit under automorphisms, but there is a computable bound on the size of that orbit. Computable finite-branching trees under predecessor, as nice as they are to work with as structures, do not have this property, and neither do computable connected finite-valence graphs with a constant node. We go on to argue that a computable structure behaves like an algebraic field with respect to degree spectrum if and only if the structure is effectively algebraic.

The third chapter of this dissertation also concerns degree spectra of structures considered as substructures of bigger structures, but we move to a type of structure we haven't discussed yet – Boolean algebras. It was shown by R. G. Downey and C. G. Jockusch [4] that every low Boolean algebra (considered as a structure in its own right) must be isomorphic to a computable Boolean algebra. A degree $\mathbf{d}$ is defined to be *low* if $\mathbf{d}' \leq_T \mathbf{0}'$, and we generalize this definition to say that $\mathbf{d}$ is low_n if $\mathbf{d}^{(n)} \leq_T \mathbf{0}^{(n)}$. It was then shown by J. J. Thurber [30] that every low_2 Boolean algebra must be isomorphic to a computable Boolean algebra. It was further shown by J. F. Knight and M. Stob [16] that every low_3 Boolean algebra and every low_4

Boolean algebra must be isomorphic to a computable Boolean algebra. It remains unanswered whether it is true for $n > 4$ that every low_n Boolean algebra has a computable copy, but in order to gain some more understanding into the $n = 5$ case, R. Miller [22] proved that there exists a low_5 subalgebra of the computable atomless Boolean algebra $\mathcal{B}$ which, when viewed as a relation on $\mathcal{B}$, is not isomorphic to any computable subalgebra of $\mathcal{B}$. Here, we adapt Miller's proof to show that there also exists a low_4 Boolean subalgebra of $\mathcal{B}$ which, when considered as a relation on $\mathcal{B}$, has no computable copy. This new result provides a sharp contrast with the result of Knight and Stob that every low_4 Boolean algebra has a computable copy. That is, the spectrum of the subalgebra as a unary relation can contain a low_4 degree without containing the degree $\mathbf{0}$, even though no spectrum of a Boolean algebra (viewed as a structure in its own right) can do the same. We go on to show that if a subalgebra of $\mathcal{B}$ is not intrinsically computable (i.e its image under automorphisms of $\mathcal{B}$ isn't always computable), then the degree spectrum of that subalgebra, as a relation on $\mathcal{B}$, is closed upwards, and we give a simple example of what the construction of such a subalgebra would involve. We also cite a theorem of Jockusch and R. I. Soare [13] which states that if a Boolean algebra (as a structure) has nth-jump degree (i.e. if the set of degrees of nth-jumps of structures isomorphic to that Boolean algebra has a smallest member), then that degree must be $\mathbf{0}^{(n)}$; we show here that this is not the case for Boolean subalgebras of $\mathcal{B}$ considered as relations on $\mathcal{B}$. If a subalgebra of $\mathcal{B}$ has nth jump degree, it is possible for that degree to be larger than $\mathbf{0}^{(n)}$.

Chapter 1

Fields and the Bounded Turing Reduction

1.1 Introduction

Given a polynomial $p(X)$ with coefficients in a field F, the questions that are typically asked about $p(X)$ are:

- Does $p(X)$ factor in $F[X]$?
- Does $p(X)$ have a root in F?

It is not always clear which of these two questions is easier to answer. We will address this issue in the case where F is a *computable* field.

Definition 1.1.1 A *computable field* F consists of two computable functions f and g from $\omega \times \omega$ into ω, such that ω forms a field under these functions, with f as the addition and g as the multiplication. Sometimes we refer to F as a *computable presentation* of the isomorphism type of F.

Definition 1.1.2 The *splitting set* S_F for a field F is the set of polynomials in $F[X]$ which factor, i.e. split, into two nonconstant factors in $F[X]$. The *root set* R_F of F is the set of polynomials in $F[X]$ which have a root in F.

Both of these sets are computably enumerable; they are defined by existential formulas.

Turing reducibility is the most common reducibility used by computability theorists to compare the relative complexity of two sets of natural numbers. Finer reducibilities, such as m-reducibility and 1-reducibility, can further separate the relative complexities of sets that sit inside the same Turing degree. A set B is *m-reducible* to a set C, written $B \leq_m C$, if there is a total computable function f where, for all $x \in \omega$, $x \in B$ iff $f(x) \in C$. B is *1-reducible* to C if the function f can be taken to be injective.

For a computable field F, the set S_F and the set R_F are actually always Turing-equivalent, which surprises most mathematicians; the reduction $R_F \leq_T S_F$ is rather obvious (think about it!), but the other direction, $S_F \leq_T R_F$, which was proven in [8], was much more complicated to prove. The sets S_F and R_F are no longer equivalent when we compare them under 1-reducibility; we always have $S_F \leq_1 R_F$, while there are computable fields F for which $R_F \not\leq_1 S_F$ [21].

Another, less-common type of reducibility is the *bounded Turing reducibility*.

Definition 1.1.3 For $A, B \subseteq \omega$, A is *bounded Turing reducible to* B, written $A \leq_{bT} B$, if there exists an oracle Turing functional Φ_e and a total computable function f such that Φ_e^B computes the characteristic function of A and for all $x \in \omega$, the

computation $\Phi_e^B(x)$ asks its oracle questions only about the membership in B of elements $\leq f(x)$. In other words, for all x, $\Phi_e^{B\upharpoonright f(x)}(x) \downarrow = \chi_A(x)$.

Bounded Turing reducibility is sometimes called *weak truth-table reducibility* because it is a weakening of another reducibility known as the truth-table reducibility, but "bounded Turing reducibility" is a much more descriptive name. In many papers, including [29], $A \leq_{wtt} B$ is used in place of $A \leq_{bT} B$.

Bounded Turing (bT) reducibility is finer than Turing reducibility, but still coarser than 1-reducibility. The purpose of this paper is to find out what happens between S_F and R_F under this intermediate reducibility – are they equivalent, as under Turing reducibility, or is one slightly harder to compute than the other, as under 1-reducibility?

Since $S_F \leq_1 R_F$ for any computable algebraic field F, it follows that $S_F \leq_{bT} R_F$, because the function f which serves as the 1-reduction is exactly the computable bound on the R_F oracle. But it turns out that we can construct a computable algebraic field F in which $R_F \not\leq_{bT} S_F$. The interesting thing about this is that the lack of bT-reduction from R_F to S_F is equivalent to a purely Galois-theoretic statement about the splitting set and the root set of a field!

Consider the following fact that will be stated and proved later in Lemma 1.2.11:

Let L be a Galois extension of $\mathbb{Q}$. For any fields F_0 and F_1, with $\mathbb{Q} \subsetneq F_0, F_1 \subsetneq L$,

$$(\forall q(X) \in \mathbb{Q}[X])\ [q \textit{ has a root in } F_0 \iff q \textit{ has a root in } F_1]$$

if and only if

$$(\exists \sigma \in Gal(L/\mathbb{Q}))\ [\sigma(F_0) = F_1].$$

Also consider the statement $\star$, which is almost the same as above, but with the words "factors over" replacing the words "has a root in." The reason there cannot be any bT-reduction from R_F to S_F is that the statement in Lemma 1.2.11 is true, while $\star$ is false. In fact, if $\star$ were to be true also, then there we would always have a bT-reduction from R_F to S_F. This idea will be explored further in Section 4.

A standard reference for algebraic background is [31]; the canonical reference for computability theoretic background is [27], which is soon to be subsumed into the very extensive [28].

1.2 Algebraic Background

We will need the following lemmas in our construction, so we state them here for reference:

Definition 1.2.1 The *elementary symmetric polynomials* in $X_1, \ldots, X_m$ over a field F are the polynomials

$$s_k(X_1, \ldots, X_m) = \sum_{1 \leq i_1 < \cdots < i_k \leq m} X_{i_1} X_{i_2} \cdots X_{i_k} \ \text{(for } 1 \leq k \leq m).$$

The *symmetric polynomials* are the elements of $F[s_1, \ldots, s_m]$ – they are exactly the polynomials in $F[X_1, \ldots, X_m]$ that are invariant under permutations of the variables.

A subfield generated by the elementary symmetric polynomials in some proper nonempty subset of the set of roots of a polynomial $q(X) \in \mathbb{Q}[X]$ is called a *symmetric subfield* in the lattice of subfields of the splitting field of q over $\mathbb{Q}$. We refer the reader to [21] for additional details about these subfields, including the proof of the following lemma.

Lemma 1.2.2 *Let $p(X) \in F[X]$ be a polynomial over a field F and let $E \supseteq F$ be a field extension. Let $\overline{E}$ be the algebraic closure of E, and A the set of roots of $p(X)$ in $\overline{E}$. Assume that every root of $p(X)$ has multiplicity 1. Then the following are equivalent:*

- *$p(X)$ is reducible in $E[X]$.*
- *There exists $I = \{x_1, \ldots, x_m\}$ with $\emptyset \subsetneq I \subsetneq A$ such that for every symmetric polynomial $h \in F[X_1, \ldots, X_m]$ we have $h(x_1, \ldots, x_m) \in E$.*

In other words: p factors over E if and only if E contains one of the symmetric subfields generated by a proper nonempty subset of the set of roots of p.

Definition 1.2.3 For field extensions $E \supseteq \mathbb{Q}$ and $F \supseteq \mathbb{Q}$, E and F are *linearly disjoint* if $E \cap F = \mathbb{Q}$.

Lemma 1.2.4 *For relatively prime numbers m and n, if ζ_m and ζ_n are primitive m-th and n-th roots of unity, respectively, then $\mathbb{Q}(\zeta_m)$ and $\mathbb{Q}(\zeta_n)$ are linearly disjoint.*

Proof. It is not hard to see that we have $\mathbb{Q} \subseteq \mathbb{Q}(\zeta_m) \cap \mathbb{Q}(\zeta_n)$. For the reverse inclusion, it is sufficient to show that $\mathbb{Q}(\zeta_m) \cap \mathbb{Q}(\zeta_n)$ has degree 1 over $\mathbb{Q}$, and this is shown in

[32, Propositions 2.3 & 2.4]. ■

Lemma 1.2.5 *The field $\mathbb{Q}(\zeta_{p^2})$, where ζ_{p^2} is a primitive p^2-th root of unity and p is prime with $p \equiv 1 \pmod 4$, has exactly one subfield of index 2, and this subfield contains the unique subfield of $\mathbb{Q}(\zeta_{p^2})$ of degree 2 over $\mathbb{Q}$.*

Proof. Note that $\mathbb{Q}(\zeta_{p^2})$ is a Galois extension of $\mathbb{Q}$, since ζ_{p^2} generates all of its conjugates, and so $\mathbb{Q}(\zeta_{p^2})$ is the splitting field of the polynomial $X^{p^2} - 1$.

First we show that $\mathbb{Q}(\zeta_{p^2})$ has exactly one subfield of index 2. The Galois group G of $\mathbb{Q}(\zeta_{p^2})$ over $\mathbb{Q}$ is the cyclic group of order $p(p-1)$ (see [32, Theorem 2.5]). By the Fundamental Theorem of Galois Theory (see [5, Theorem 14.14]), a subfield of $\mathbb{Q}(\zeta_{p^2})$ of index 2 corresponds to a subgroup of G of order 2. G certainly has a subgroup of order 2 since the order of G is even. Indeed, $G \cong \mathbb{Z}/p(p-1)\mathbb{Z} = \{0, 1, \ldots, p(p-1)-1\}$ under addition mod $p(p-1)$. So the only possible subgroup of G of order 2 is $\left\{0, \frac{1}{2}p(p-1)\right\}$. If there is exactly one subgroup of G of order 2, then there is exactly one subfield of $\mathbb{Q}(\zeta_{p^2})$ of index 2. In fact, if ζ is a primitive p^2-th root of unity, then $\mathbb{Q}(\zeta+\zeta^{-1})$ is the subfield of $\mathbb{Q}(\zeta)$ such that $[\mathbb{Q}(\zeta) : \mathbb{Q}(\zeta+\zeta^{-1})] = 2$ (see [32, remarks before Proposition 2.15]).

To show that $\mathbb{Q}(\zeta_{p^2})$ has exactly one subfield of degree 2 over $\mathbb{Q}$, the argument is very similar. Again by the Fundamental Theorem of Galois Theory, a subfield of $\mathbb{Q}(\zeta_{p^2})$ of degree 2 over $\mathbb{Q}$ corresponds to a subgroup of G that has index 2 in G, i.e. has order $\frac{1}{2}p(p-1)$. The only possibility for a subgroup of G of order $\frac{1}{2}p(p-1)$ must contain precisely the "even" elements of G, i.e. $\{0, 2, \ldots, p(p-1)-2\}$ under addition mod

$p(p-1)$. If there is exactly one subgroup of G with index 2 in G, then there is exactly one subfield of $\mathbb{Q}(\zeta_{p^2})$ of degree 2 over $\mathbb{Q}$. In fact, $\mathbb{Q}(\sqrt{p})$ is the subfield of $\mathbb{Q}(\zeta_{p^2})$ such that $[\mathbb{Q}(\sqrt{p}) : \mathbb{Q}] = 2$ (see [5, Example 14.5.1]).

Now to explain why the index 2 subfield must contain the degree 2 subfield: The subgroup H of G of order $\frac{1}{2}p(p-1)$ must contain a subgroup of order 2, since $\frac{1}{2}p(p-1)$ is even (recall that we assumed that $p \equiv 1 (\text{mod } 4)$), and because G has only one of these, H must contain it. The Fundamental Theorem tells us that H containing the subgroup of order 2 corresponds exactly to the subfield of $\mathbb{Q}(\zeta_{p^2})$ with degree 2 over $\mathbb{Q}$ being contained in the subfield of $\mathbb{Q}(\zeta_{p^2})$ of index 2. And since we have the first containment, we must also have the second one. ∎

Corollary 1.2.6 *If K is an extension of $\mathbb{Q}$ of finite degree over $\mathbb{Q}$, then K is linearly disjoint from $\mathbb{Q}(\zeta_{p^2})$ for all but finitely many primes $p \equiv 1 \pmod 4$.*

Proof. If $K \cap \mathbb{Q}(\zeta_{p^2}) \supsetneq \mathbb{Q}$ for infinitely many primes p, then since K has only finitely many subfields between itself and $\mathbb{Q}$, infinitely many $\mathbb{Q}(\zeta_{p^2})$ would all contain the same proper extension of $\mathbb{Q}$. But according to Lemma 1.2.4, the fields $\mathbb{Q}(\zeta_{p^2})$ are pairwise linearly disjoint. So K can only intersect finitely many of the extensions $\mathbb{Q}(\zeta_{p^2})$ in a field bigger than $\mathbb{Q}$. ∎

Lemma 1.2.7 [1] *Suppose L/K is a Galois extension and g is an irreducible polynomial in $L[X]$. Let L_1 be the field extension of K generated by the coefficients of g and let α be a root of g in $\overline{L}$. If $K \subseteq E \subseteq L$, then the minimal polynomial for α over E*

[1] Thanks to Kenneth Kramer for pointing out this group-theoretic fact that was crucial to the result of this chapter.

is

$$f = \prod \sigma_i(g),$$

where $\{\sigma_i\}$ *is the set of distinct embeddings of* L_1 *into* L *over* E*. It follows that*

$$\deg f = [L_1E : E] \deg g.$$

Proof. Let $h = \prod \sigma_i(g)$ and let f be the minimal polynomial for α over E. We will show that $h = f$.

Note first that $h(\alpha) = f(\alpha) = 0$.

We want $\tau(h) = h$ for any $\tau \in \mathrm{Gal}(L/E)$, because then h will be a polynomial over E. But since τ acts by permutation on the cosets, it just rearranges the factors of h, and since polynomial multiplication is commutative, $\tau(h) = h$. Therefore f, the minimal polynomial of α over E, divides h.

Next we show that the $\sigma_i(g)$ are all relatively prime to each other. Suppose that $\sigma_k(g)$ and $\sigma_j(g)$ have a factor in common. Since both of these are irreducible over L, the factor that they have in common must be $\sigma_k(g) = \sigma_j(g)$. In particular, this means that $\sigma_k\sigma_j^{-1}(g) = g$, so $\sigma_k\sigma_j^{-1}$ fixes all coefficients of g, hence fixes L_1, which is generated by the coefficients of g. But $\sigma_k\sigma_j^{-1} \in \mathrm{Gal}(L/E)$, so $\sigma_k\sigma_j^{-1}$ also fixes E. Thus $\sigma_k\sigma_j^{-1}$ is the identity on L_1E, and so $k = j$. So the polynomials $\sigma_i(g)$ are all relatively prime.

Finally, we need irreducibility. We already know that $f|h$, so we just need to show that $h|f$. The polynomial g is irreducible over L. The polynomial $\sigma_i(g)$ is likewise

irreducible over L for each i. Consider f over L: the minimal polynomial for α over L is g, so $g|f$ over L. Then act by σ_i to get $\sigma_i(g)|\sigma_i(f)$. Since σ_i acts trivially on E, $\sigma_i(f) = f$. So, $\sigma_i(g)|f$ for each i. Since the $\sigma_i(g)$ are all relatively prime, their product h must also divide f, and so h and f are actually the same polynomial. ■

Corollary 1.2.8 *For a prime $p \equiv 1$ (mod 4), a polynomial $q(X) \in \mathbb{Q}[X]$ splits (i.e. factors nontrivially) in $\mathbb{Q}(\zeta_{p^2})$ if and only if it splits in the unique index 2 subfield E of $\mathbb{Q}(\zeta_{p^2})$.*

Proof. Let $q(X)$ be an irreducible polynomial in $\mathbb{Q}[X]$. Let g be a factor of q with coefficients in $L = \mathbb{Q}(\zeta_{p^2})$ which is irreducible over L. The field L is Galois over $\mathbb{Q}$, so with $K = \mathbb{Q}$ and with notation as in Lemma 1.2.7, the degree of q is $\deg q = [L_1 : \mathbb{Q}] \deg g$. The degree of an irreducible factor h of q in the index 2 subfield E of L, where h is the factor that is a multiple of g, is $\deg h = [L_1E : E] \deg g$. To have q irreducible over E but split over L, we must have $q = h$ and $[L_1 : \mathbb{Q}] = [L_1E : E] > 1$. If $[L_1E : E] > 1$, then $L_1E = L$ because L is the only proper extension of E within L. Thus $[L_1 : \mathbb{Q}] = [L : E] = 2$. But then by Lemma 1.2.5, we must have $L_1 \subseteq E$, which is a contradiction; since L_1 is generated by the coefficients of g, E would contain these coefficients, and then q would not be irreducible over E. ■

The following lemma dates back to a paper [17] written by Kronecker in 1882, so we refer the reader to the more recent account in [7].

Lemma 1.2.9 *The splitting set of the field $\mathbb{Q}$ is computable. Furthermore, if L is a c.e. subfield of a computable field K and L has a splitting algorithm, then so does*

every finite extension of L inside K, and if K is algebraic over L, these splitting algorithms can be found uniformly in the generators of the extension.

Lemma 1.2.10 *Any isomorphism of extensions of $\mathbb{Q}$ must fix $\mathbb{Q}$ pointwise.*

Lemma 1.2.11 *Let L be a Galois extension of $\mathbb{Q}$. For any fields F_0 and F_1, with $\mathbb{Q} \subsetneq F_0, F_1 \subsetneq L$,*

$$(\forall q(X) \in \mathbb{Q}[X])\ [q \text{ has a root in } F_0 \iff q \text{ has a root in } F_1]$$

if and only if

$$(\exists \sigma \in Gal(L/\mathbb{Q}))\ [\sigma(F_0) = F_1].$$

Proof. For the backward direction, let $\sigma \in \mathrm{Gal}(L/\mathbb{Q})$ with $\mathbb{Q} \subsetneq F_0 \subsetneq L$ and $\sigma(F_0) \neq F_0$. Let $F_1 = \sigma(F_0)$. Then, for $q(X) \in \mathbb{Q}[X]$,

$$q \text{ has a root in } F_0 \implies \exists\ c \in F_0 \text{ with } q(c) = 0$$

$$\implies \exists\ c \in F_0\ \ \sigma(q(c)) = \sigma(0)$$

$$\implies \exists\ c \in F_0\ \ \sigma(q(c)) = 0$$

$$\implies \exists\ c \in F_0\ \ q(\sigma(c)) = 0$$

(because by Lemma 1.2.10, any automorphism of L must fix $\mathbb{Q}$ pointwise and therefore must fix the coefficients of q)

$$\implies q \text{ has a root in } F_1.$$

To show that q has a root in F_0 whenever q has a root in F_1, repeat the above argument with σ^{-1} instead of σ.

For the forward direction, suppose F_0 and F_1 have the same root set with respect to polynomials with rational coefficients, i.e. $R_{F_0} \cap \mathbb{Q}[X] = R_{F_1} \cap \mathbb{Q}[X]$.

Claim: $F_0 \cong F_1$.

According to the Theorem of the Primitive Element (see [31]), every finite algebraic extension of $\mathbb{Q}$ is actually generated by a single element, called a "primitive element." If we know that such an element exists for a given field, we can search through the field until we find one. Find a primitive generator r_0 for F_0 and then find the minimal polynomial $p_0(X)$ over $\mathbb{Q}$ for this generator. Then F_0 certainly has a root of p_0, and by assumption, F_1 also has a root of p_0.

If α and β are two roots of the same irreducible polynomial p over $\mathbb{Q}$, then $\mathbb{Q}(\alpha) \cong \mathbb{Q}[X]/(p(X)) \cong \mathbb{Q}(\beta)$. With this fact in mind, we can see that since F_1 also has a root of the polynomial p_0, then F_1 must contain a subfield isomorphic to $\mathbb{Q}(r_0) = F_0$.

Now find a primitive generator r_1 for F_1 and then find the minimal polynomial $p_1(X)$ for this generator. Then F_1 certainly has a root of p_1, and by assumption, F_0 also has a root of p_1.

Then F_0 contains a subfield isomorphic to $\mathbb{Q}(r_1) = F_1$.

Now, if two fields embed into one another, then the composition of those embeddings is an embedding of the first field into itself, and every embedding of an algebraic field into itself must be an automorphism (see [20, Lemma 2.10]). So the reverse embedding is surjective, and so the two fields must actually be isomorphic. Thus

$F_0 \cong F_1$ via some isomorphism τ. So our claim is proved.

Any isomorphism of extensions of $\mathbb{Q}$ must fix $\mathbb{Q}$ pointwise, as stated earlier in Lemma 1.2.10, and we know that any isomorphism of two subfields of a Galois extension can be effectively extended to an automorphism of that Galois extension (see [31]). These two facts together tell us that τ can be extended to an automorphism σ of L that fixes $\mathbb{Q}$ pointwise, and thus $\sigma \in \mathrm{Gal}(L/\mathbb{Q})$ with $\sigma(F_0) = F_1$. ■

The result in this paper depends almost exclusively on the fact that if we replace the phrase "has a root in" with "factors over" in Lemma 1.2.11, the statement is no longer true; it is the forwards direction that fails. Here is a counterexample: Let L be the field $\mathbb{Q}(\sqrt[8]{2}, i)$, which happens to be the splitting field of the polynomial $x^8 - 2$, and so is Galois over $\mathbb{Q}$. Let $F_0 = \mathbb{Q}(\sqrt{2}, i)$ and $F_1 = \mathbb{Q}(\sqrt[4]{2}, i)$. Given any polynomial q with rational coefficients, q factors over F_0 if and only if it factors over F_1; this is a consequence of Lemma 1.2.7 (F_0 is a subfield of F_1 of index 2, and F_0 contains all the subfields of F_1 of degree 2 over $\mathbb{Q}$, so apply an argument like the one in Corollary 1.2.8. However, F_1 cannot possibly be the image of F_0 under any element of $\mathrm{Gal}(L/\mathbb{Q})$ because F_0 and F_1 are not isomorphic; F_0 is a proper subfield of F_1, so $[F_0 : \mathbb{Q}] < [F_1 : \mathbb{Q}]$.

In Section 1.4 we will take a look at how one would go about building a bT-reduction if the "factors over" analogue of Lemma 1.2.11 were to be true.

1.3 The Construction

Theorem 1.3.1 *There exists a computable algebraic field F, with splitting set S_F and root set R_F, for which $R_F \not\leq_{bT} S_F$.*

We will first construct the computable field F, and then prove that $R_F \not\leq_{bT} S_F$. We effectively enumerate all the partial computable functions $\varphi_0, \varphi_1, \varphi_2, \ldots$ and all the partial computable Turing functionals $\Phi_0, \Phi_1, \Phi_2, \ldots$, and we build F to satisfy, for every e and every i, the requirement

$$\mathcal{R}_{e,i} : \text{If } \Phi_e^{S_F} \text{ and } \varphi_i \text{ are total and } \forall q\ (\Phi_e^{S_F}(q) \downarrow = 0 \iff q \notin R_F), \text{ then}$$
$$(\exists q \in F[X])(\exists y \geq \varphi_i(q))[\Phi_e^{S_F}(q) \text{ asks an oracle question about } y].$$

If every requirement is satisfied, then we will have $R_F \not\leq_{bT} S_F$ because no total computable function can possibly play the role of the bound function on the splitting set oracle.

The construction will happen in stages.

For each e and each i, we will choose a witness polynomial $q_{e,i}(X) \in \mathbb{Q}[X]$, and we will feed it to φ_i and wait, perhaps forever, for $\varphi_i(q_{e,i})$ to halt. If this computation ever does halt, then we will feed $q_{e,i}$ to Φ_e with oracle $S_F \upharpoonright \varphi_i(q_{e,i})$ and wait to see if this second computation halts. If it does, we will ensure that $\Phi_e^{S_F}(q) \downarrow = 0 \iff q \in R_F$.

Given a computable (domain ω) field F, we can list out all the monic polynomials in $F[X]$. We do this by listing out $\omega^{<\omega}(\cong \omega)$ and think of $(a_0, \ldots, a_d) \in \omega^{d+1}$ as representing $a_0 + a_1X + \cdots + a_dX^d + X^{d+1} \in F[X]$.

So we list $F[X] = \{p_0(X), p_1(X), p_2(X), \ldots\} \cong \omega$. S_F is a subset of $F[X]$.

If we use the standard map from ω to $\omega^{<\omega}$, then $p_n(X)$ only uses coefficients from $\{0, 1, \ldots, n\} \subseteq \omega$, the domain of F. If we ensure that the field elements labeled $\{0, 1, \ldots, n\}$ lie within $\mathbb{Q}$, then we will have $\{p_0, p_1, \ldots, p_n\} \subseteq \mathbb{Q}[X]$. So, if the elements $0, 1, \ldots, \varphi_i(q)$ in the domain of F all lie within $\mathbb{Q}$, then we have $\{p_0, p_1, \ldots, p_{\varphi_i(q)}\}$ $\subseteq \mathbb{Q}[X]$. So $S_F \upharpoonright \varphi_i(q)$ only tells us whether elements of $\mathbb{Q}[X]$ split in $F[X]$.

Throughout the construction, we refer to D_s as the set of elements of $\overline{\mathbb{Q}}$ that have been enumerated by the end of stage s, and we refer to F_s as the field that these elements generate. Elements are enumerated into D_s with the goal of eventually enumerating the entire field F_s.

Here is the "basic module" for the construction, i.e. how we satisfy a single requirement $\mathcal{R}_{e,i}$:

Step 0: Initialization.

For the strategy that aims to satisfy the requirement $\mathcal{R}_{e,i}$, choose a prime $p \equiv 1 \pmod 4$, and consider the Galois extension generated by adjoining a primitive p^2-th root of unity ζ_{p^2} to $\mathbb{Q}$, and consider its lattice of subfields over $\mathbb{Q}$. Denote the element ζ_{p^2} by $\beta_{e,i}$ and the element $\zeta_{p^2} + \zeta_{p^2}^{-1}$ by $\alpha_{e,i}$.

We choose, for our "witness" polynomial, the cyclotomic polynomial that has, as its roots, precisely the primitive p^2-th roots of unity. (This is the minimal polynomial of $\beta_{e,i}$ over $\mathbb{Q}$.) We do this because we want a polynomial with coefficients in $\mathbb{Q}$ that has a root in $\mathbb{Q}(\beta_{e,i})$ but no root in any proper subfield of $\mathbb{Q}(\beta_{e,i})$. Call this polynomial $q_{e,i}$.

Step 1: Adjoining $\alpha_{e,i}$.

If and when the computation $\varphi_i(q_{e,i})$ halts, say that it does so at stage s_0. Then at stage s_1 with $s_1 = s_0+1$, we enumerate the element $\alpha_{e,i}$ into D_{s_1} (and hence eventually all $\mathbb{Q}(\alpha_{e,i})$ into the field F_{s_1}) in such a way that the first $\varphi_{i,s_1}(q_{e,i})$ elements of D_{s_1} are elements of the subfield $\mathbb{Q}$.

Once we've put $\alpha_{e,i}$ into the field, then if and when $\Phi_e^{S_{F_{s_1}}\upharpoonright\varphi_{i,s_1}(q_{e,i})}(q_{e,i})$ halts, say that it does so at stage s_2. There are three possible outcomes of this convergence:

- Case 1: $\Phi_{e,s_2}^{S_{F_{s_2}}\upharpoonright\varphi_{i,s_2}(q_{e,i})}(q_{e,i}) \downarrow = 1$. Then according to the splitting set restricted to the first $\varphi_{i,s_2}(q_{e,i})$ elements of the field, F_{s_2} has a root of $q_{e,i}$, which is clearly false, because we haven't put one in!

- Case 2: $\Phi_{e,s_2}^{S_{F_{s_2}}\upharpoonright\varphi_{i,s_2}(q_{e,i})}(q_{e,i}) \downarrow \notin \{0,1\}$. This case is also an immediate win for us, because convergence to something other than 0 or 1 has no meaning in this context. We treat this case exactly like Case 1.

Step 2: Adjoining $\beta_{e,i}$.

- Case 3: $\Phi_{e,s_2}^{S_{F_{s_2}}\upharpoonright\varphi_{i,s_2}(q_{e,i})}(q_{e,i}) \downarrow = 0$. Here lies the potential difficulty, as the re-

stricted splitting set thinks (correctly) that F_{s_2} has no root of $q_{e,i}$. So at stage s_3 with $s_3 = s_2 + 1$, we simply enumerate a root of $q_{e,i}$ (namely $\beta_{e,i}$) into D_{s_3}.

But won't adding this root have possible adverse effects on the splitting set?

Actually, it won't. Recall that the first $\varphi_i(q_{e,i})$ elements of the field lie in $\mathbb{Q}$. Adding an element of $\mathbb{Q}(\beta_{e,i})$ has no effect on the splitting of rational polynomials, because F_{s_1} already contains the field $\mathbb{Q}(\alpha_{e,i})$, and by Corollary 1.2.8, we know that $\mathbb{Q}(\beta_{e,i})$ and $\mathbb{Q}(\alpha_{e,i})$ split exactly the same rational polynomials. If no rational polynomial will be caused to split by the addition of this root, then nothing can enter the restricted splitting set, and so the oracle $S_F \upharpoonright \varphi_i(q_{e,i})$ will not change, and thus the computation will still yield convergence to 0 even though we now have a root of $q_{e,i}$ in the field. So our basic module does indeed satisfy the single requirement $\mathcal{R}_{e,i}$.

We need to be able to deal with many different requirements $\mathcal{R}_{e,i}$ at once. For this, we use the technique of the finite injury priority construction.

Some requirements will be restricted from putting certain elements of $\overline{\mathbb{Q}}$ into D_s. For example, if a requirement wants to be satisfied by putting a root of a particular polynomial into D_s but this root would cause a polynomial in the oracle for a higher-priority requirement to factor, and thus change the oracle for this higher-priority requirement, we cannot allow this. So we prevent requirements from putting anything into D_s that would cause such a factorization.

Here's an example: Suppose a requirement $\mathcal{R}_{c,j}$ has put the element $\zeta_{p^2} + \zeta_{p^2}^{-1}$ into D_s at some stage s after the convergence of $\varphi_j(q_{c,j})$, thus enumerating many polynomials

with coefficients from $\mathbb{Q}(\zeta_{p^2}+\zeta_{p^2}^{-1})$ into $F_s[X]$. Then, at some stage $t > s$, $\varphi_i(q_{e,i})$ with $\langle e,i\rangle < \langle c,j\rangle$ converges so that $\varphi_i(q_{e,i})\downarrow > \varphi_j(q_{c,j})\downarrow$, and now one of the polynomials with coefficients from $\mathbb{Q}(\zeta_{p^2}+\zeta_{p^2}^{-1})$ is sitting in $D_t[X]$ below the $\varphi_i(q_{e,i})$-th element of $D_t[X]$. If later, at a stage after t, $\mathcal{R}_{c,j}$ needs to put the element ζ_{p^2} into the field in order to be satisfied, he may no longer be able to do so, since this may cause one of the polynomials below the bound of the higher-priority $\mathcal{R}_{e,i}$ to split. The requirement $\mathcal{R}_{c,j}$ would be stuck.

We say that the convergence of the aforementioned $\varphi_i(q_{e,i})$ constitutes an *injury* to the requirement $\mathcal{R}_{c,j}$. Because $\mathcal{R}_{c,j}$ may not put into the field any element that might cause changes in the oracle for a higher-priority requirement, its strategy gets *initialized*, i.e. it must start over again from the beginning by choosing a fresh prime. $\mathcal{R}_{c,j}$ can be injured only finitely many times because injury only happens when a higher-priority requirement goes through "step 1" of the basic module. This can't happen more than $2^n - 1$ times if there are n requirements with higher priority than $\mathcal{R}_{c,j}$. Choosing a new prime each of those finitely many times is a way to guarantee that $\mathcal{R}_{c,j}$ can be satisfied while obeying the restrictions set by higher-priority requirements.

At the end of any stage s, we already have a target field F_s, and whatever element gets added to this field at subsequent stages must not cause factorizations that could possibly mess up the restricted splitting sets used as oracles in higher-priority requirements. The set of all the elements that $\mathcal{R}_{e,i}$ wants kept out of D as of stage s

includes:

- any element of $\mathbb{Q}(\zeta_{p^2}) - \mathbb{Q}$, if $\mathcal{R}_{e,i}$ has already chosen its prime p by stage s;
- any element of $\overline{\mathbb{Q}}$ that would cause factorizations in the first $\varphi_i(q_{e,i,s})$ elements of $D_s[X]$, if $\mathcal{R}_{e,i}$ has already seen convergence of $\varphi_{i,s}(q_{e,i})$ by stage s.

The elements that would cause factorizations in the first $\varphi_i(q_{e,i,s})$ elements of $D_s[X]$ are elements of the "symmetric subfields" generated by the elementary symmetric polynomials in proper nonempty subsets of the set of roots of each of the first $\varphi_i(q_{e,i,s})$ elements of $D_s[X]$, by Lemma 1.2.2. So $\mathcal{R}_{e,i}$ wants, at best, these symmetric subfields kept out of D at stage s. It suffices to ensure that D contains no primitive generator of any symmetric subfield. So the elements of $\overline{\mathbb{Q}}$ that $\mathcal{R}_{e,i}$ keeps from going into the field are all in the field generated by a primitive generator for $F_s(\zeta_{p^2})$ and a primitive generator for each of those finitely many symmetric subfields.

Denote by $E_{e,i,s}$ the set that contains ζ_{p^2} and one primitive generator for each of the symmetric subfields for each polynomial that $\mathcal{R}_{e,i}$ wants to keep from factoring.

$\mathcal{R}_{e,i}$ may declare certain elements off-limits to lower-priority requirements, but he must also obey the restrictions set by higher-priority requirements; at the end of stage s he cannot put elements of $\bigcup_{\langle e',i'\rangle<\langle e,i\rangle} E_{e',i',s}$ into D_{s+1}.

Denote by $K_{e,i,s}$ the smallest normal field generated by $\bigcup_{\langle e',i'\rangle<\langle e,i\rangle} E_{e',i',s}$ over F_s. We would like to show that at any given stage s, there is a prime $p \equiv 1 \pmod 4$ sufficiently large so that if we must adjoin a p^2-th root of unity to the current target field F_s for

the sake of a requirement $\mathcal{R}_{e,i}$, we will not cause any additional factorizations in the restricted splitting sets for higher-priority requirements; i.e. $K_{e,i,s}$ will intersect the field $F_s(\zeta_{p^2})$ only in F_s itself. Once we show that there is such a p, we know that there must be a smallest one, and this smallest p is the one we will use in the strategy to satisfy $\mathcal{R}_{e,i}$.

Lemma 1.3.2 *For each e and each i, at any given stage s, there must be a prime $p \equiv 1$ (mod 4) for which $K_{e,i,s} \bigcap F_s(\zeta_{p^2}) = F_s$, and we can find one effectively.*

Proof. $K_{e,i,s}$ is a finite-degree extension of $\mathbb{Q}$, and so by Lemma 1.2.6, it can only intersect finitely many $\mathbb{Q}(\zeta_{p^2})$ for p prime. So there is a prime $p \equiv 1 \pmod 4$ for which $K_{e,i,s} \bigcap F_s(\zeta_{p^2}) = F_s$.

Here's how we find one: Look at the minimal subfields of $\mathbb{Q}(\zeta_{p^2})$ for the prime p in question, and find a primitive generator for each one. Find the minimal polynomials over $\mathbb{Q}$ for each one of these generators, and check if $K_{e,i,s}$ has a root of any of them (this can be done, by Lemma 1.2.9, since $K_{e,i,s}$ is a finite extension of $\mathbb{Q}$). If $K_{e,i,s}$ has a root of any of these minimal polynomials, then it is not linearly disjoint from $\mathbb{Q}(\zeta_{p^2})$, but if $K_{e,i,s}$ has no roots of any of these minimal polynomials, then it is linearly disjoint from $\mathbb{Q}(\zeta_{p^2})$. ∎

We will choose such a prime in our construction to satisfy $\mathcal{R}_{e,i}$.

We say that the requirement $\mathcal{R}_{e,i}$ "needs attention" at stage $s+1$ if one of the following three things holds:

- a prime has not yet been chosen for this requirement since its last injury;
- a prime and a polynomial $q_{e,i,s}$ have been chosen for this requirement and $\varphi_{i,s}(q_{e,i,s})$ has converged, but we have not yet done anything further to satisfy the requirement;
- a prime and a polynomial $q_{e,i,s}$ have been chosen for this requirement, $\varphi_{i,s}(q_{e,i,s})$ has converged, and $\Phi_{e,s}^{S_{F_s}\restriction\varphi_{i,s}(q_{e,i,s})}(q_{e,i,s}) \downarrow = 0$, but we have not done anything further to satisfy the requirement.

So now we're ready for the actual full construction:

Stage 0: Let $F_0 = \mathbb{Q}$ and $D_0 = \emptyset$.

Stage $s+1$: Of all the requirements $\mathcal{R}_{e,i}$ which "need attention" at this stage, select the one with the least index pair $\langle e, i\rangle$.

As in the basic module, we denote the element ζ_{p^2} by $\beta_{e,i}$ and the element $\zeta_{p^2} + \zeta_{p^2}^{-1}$ by $\alpha_{e,i}$.

- If no prime has been chosen for this requirement's strategy since its last injury, then we choose the smallest prime $p \equiv 1 \pmod 4$ such that $K_{e,i,s} \bigcap F_s(\beta_{e,i}) = F_s$. Take the cyclotomic polynomial that has, as its roots, precisely the primitive p^2-th roots of unity and call this polynomial $q_{e,i,s+1}$. We declare every element of $\mathbb{Q}(\beta_{e,i}) - \mathbb{Q}$ "off-limits" to all lower-priority requirements, i.e. we define $E_{e,i,s+1}$ to be a set of primitive generators, one for each of the intermediate subfields between $\mathbb{Q}$ and $\mathbb{Q}(\beta_{e,i})$. We initialize all requirements with lower priority.

- If $\varphi_{i,s}(q_{e,i,s}) \downarrow$ but we haven't yet done anything further to satisfy $\mathcal{R}_{e,i}$, then we form $D_{s+1} = D_s \cup \{\alpha_{e,i}\}$. (It is important to note here that this new element will appear after the $\varphi_{i,s}(q_{e,i,s})$-th element of D_{s+1}, because there must already be at least s elements in D_s, and $\varphi_i(q_{e,i,s})$, having converged in s steps or fewer, must be $\leq s$.) We define $E_{e,i,s+1}$ to be a (finite) set of primitive generators of the symmetric subfields whose elements, if put into D at any later stage, would cause factorizations in any of the first $\varphi_{i,s}(q_{e,i,s})$ elements of D_{s+1}. We initialize all requirements with lower priority.

- If the element $\alpha_{e,i}$ has already been put into D_s and $\Phi_{e,s}^{S_{F_s} \restriction \varphi_{i,s}(q_{e,i,s})}(q_{e,i,s}) \downarrow = 0$ but we have not yet done anything further to satisfy $\mathcal{R}_{e,i}$, then we form $D_{s+1} = D_s \cup \{\beta_{e,i}\}$.

Let $F = \bigcup_s F_s = \bigcup_s D_s$.

Lemma 1.3.3 *For each e and each i, the requirement $\mathcal{R}_{e,i}$ is injured only finitely often and acts only finitely often.*

Proof. We prove both of these with a simultaneous induction. A requirement may not need to act at all, as in the case where there is no convergence of the function $\varphi_i(q_{e,i,s})$. If and when the first requirement $\mathcal{R}_{0,0}$ acts, it injures all other requirements and causes their strategies to be initialized. $\mathcal{R}_{0,0}$ is never injured; its strategy is never initialized, and so it never needs to act again. By induction, each requirement is injured only finitely often, and if and when it acts after the stage of its final injury, that will be its last action. ∎

If $\mathcal{R}_{e,i}$ acts for the last time (if it ever acts at all) at stage s, then its chosen polynomial does not change after stage s. So $\lim_s q_{e,i,s}$ exists, and we can let $\lim_s q_{e,i,s} = q_{e,i}$.

Lemma 1.3.4 *Suppose the requirement $\mathcal{R}_{e,i}$ has been satisfied and is never injured again. Suppose it was the case that the computation $\varphi_i(q_{e,i})$ did in fact halt, and let t be the stage at which the element $\alpha_{e,i}$ was enumerated into D. Then the above construction guarantees that each of the first $\varphi_{i,t}(q_{e,i,t})$ polynomials in $F_t[X]$ splits over F_t if and only if it splits over F.*

Proof. In the case that $\Phi_e^{S_F \upharpoonright \varphi_i(q_{e,i})}(q_{e,i}) \downarrow = 0$, the element $\beta_{e,i}$ will get enumerated into D, causing the whole field $\mathbb{Q}(\beta_{e,i})$ to go into F, but this will not make any difference to the current splitting set, as a result of both Corollary 1.2.8 and Lemma 1.3.2. And, regardless of whether $\beta_{e,i}$ went into the field, no lower-priority requirement will ever be allowed to put anything into the field that would make any of these first $\varphi_i(q_{e,i})$ polynomials split, as stated explicitly in the construction. No polynomial within this bound that hasn't split by stage t will ever split. So none of the first $\varphi_i(q_{e,i})$ polynomials in $F_t[X]$ will ever be caused to split after stage t. ∎

Lemma 1.3.5 *The requirement $\mathcal{R}_{e,i}$ is satisfied for every $\langle e, i \rangle$.*

Proof. We showed in Lemma 1.3.3 that each requirement $\mathcal{R}_{e,i}$ acts only finitely often. Now to show that this last action by $\mathcal{R}_{e,i}$ actually does result in its satisfaction: In the case that $\varphi_i(q_{e,i})$ never halts, φ_i is not a total function, and so the hypothesis of the requirement is false. In the case that $\varphi_i(q_{e,i})$ halts but $\Phi_e^{S_F \upharpoonright \varphi_i(q_{e,i})}(q_{e,i})$ never halts, then $\Phi_e^{S_F \upharpoonright \varphi_i(q_{e,i})}$ is not a total function, and so again the hypothesis of the requirement

is false. In the case that both functions converge on $q_{e,i}$, the construction ensures that $\Phi_e^{S_F \restriction \varphi_i(q_{e,i})}(q_{e,i}) \downarrow = 0$ iff $q_{e,i} \in R_F$. We never end up with $\Phi_e^{S_F \restriction \varphi_i(q_{e,i})}(q_{e,i}) \downarrow = 0$ and $q_{e,i} \notin R_F$, because if the convergence to zero happens, we force $q_{e,i}$ into R_F, and this doesn't change the convergence to zero, as explained in the proof of Lemma 1.3.4, because no requirement will ever be able to put $q_{e,i}$ into R_F. We also never end up with $\Phi_e^{S_F \restriction \varphi_i(q_{e,i})}(q_{e,i}) \downarrow = 1$ and $q_{e,i} \in R_F$, because we don't put $q_{e,i}$ into R_F unless we have already seen that $\Phi_e^{S_F \restriction \varphi_i(q_{e,i})}(q_{e,i}) \downarrow = 0$. Having the lower-priority requirements respect the restraint set by $E_{e,i}$ ensures this. ■

1.4 Differentiating Between R_F and S_F

As promised at the end of Section 2, we will now consider the (false, in general) analogue to Lemma 2.11 and describe the bT-reduction $R_F \leq_T S_F$ that must exist in cases where this analogue holds.

First, we need a result from Galois theory.

Notation: If H and I are subgroups of a group G, then $HI = \{hi : h \in H \ \& \ i \in I\}$.

The following lemma and its proof were devised for the author by Kenneth Kramer; we are unsure whether it was previously known.

Lemma 1.4.1 *Let E and F be finite algebraic extension fields of $\mathbb{Q}$ with $E \subseteq F$ and let L be the Galois closure of F over $\mathbb{Q}$. Also, let $G = Gal(L/\mathbb{Q})$, $H = Gal(L/E)$, and $J = Gal(L/F)$. Then the following are equivalent:*

(i) There is a proper subgroup I of G containing J such that $HI = G$;

(ii) There is a polynomial $f(X) \in \mathbb{Q}[X]$ which is irreducible over E but reducible over F.

Proof. $(i) \Rightarrow (ii)$: Let $K = L^I$ be the fixed field of I. Write $K = \mathbb{Q}(\alpha)$ where α is a primitive generator of K, and let f be the minimal polynomial of α over $\mathbb{Q}$. Since $I \subsetneq G$, we have $\deg f \geq 2$, and since $J \subseteq I$, we have $K \subseteq F$. Thus $x - \alpha$ is a factor of f in $F[X]$. The compositum KE corresponds by Galois theory to the subgroup $H \cap I$ of G. Thus

$$[E(\alpha) : E] = [KE : E] = [H : H \cap I] = \frac{|H|}{|H \cap I|} = \frac{|HI|}{|I|} = [G : I] = [K : \mathbb{Q}] = \deg f.$$

So f is irreducible over E.

$(ii) \Rightarrow (i)$: Let p be an irreducible factor of f in $F[X]$. Set $S_H = \mathrm{Stab}_H(p) = \left\{h \in H : p^H = p\right\}$ and write $H = \bigcup S_H h_j$ with $1 \leq j \leq n$. Since f is irreducible over E, H acts transitively on the factors of f in $F[X]$. Thus $n = [H : S_H]$ and

$$f = \prod_{j=1}^{n} p^{h_j}.$$

Set $I = S_G = \mathrm{Stab}_G(p)$. We certainly have $J \subseteq I$ and $S_H = H \cap I$. If $g \in G$, we have $p^g = p^{h_j}$ for some j. So $gh_j^{-1} \in I$ and $h_j^{-1}g \in I$, and so $G = IH = HI$. Now

$$[G : I] = \frac{|IH|}{|I|} = \frac{|H|}{|H \cap I|} = [H : S_H] = n.$$

By assumption, $n > 1$, so I is properly contained in G. ∎

Theorem 1.4.2 *Let F be a computable algebraic field and suppose the following property holds:*

For any fields F_0 and F_1, with $\mathbb{Q} \subseteq F_0, F_1 \subseteq F$,

$$(\forall q(X) \in \mathbb{Q}[X])\ [q \in S_{F_0} \iff q \in S_{F_1}]$$

if and only if

$$F_0 \cong F_1.$$

Then $R_F \leq_{bT} S_F$.

Proof. In particular, for every pair of subfields F_0 and F_1 of F with $\mathbb{Q} \subseteq F_0 \subsetneq F_1 \subseteq F$, we are assuming that there is a polynomial with rational coefficients that factors over F_1 but not over F_0.

Given a polynomial $q(X) \in F[X]$ which is irreducible over $\mathbb{Q}$, we want to know whether q has a root in F. (If we happen to start out with a polynomial reducible in $\mathbb{Q}$, then we would factor it into its irreducible factors in $\mathbb{Q}[X]$, which we can do because $\mathbb{Q}$ has a splitting algorithm, as we saw in Lemma 1.2.9, and then continue as follows with each factor.)

q must be a factor of some polynomial p with rational coefficients. p has a splitting field over $\mathbb{Q}$. Call this splitting field P.

We note here that to find the roots of q in F, we only need to find all roots of p in F, since once we have the roots of p in F, we can easily check, by substituting each one into q, which of the roots of p are also roots of q.

There are finitely many subfields between $\mathbb{Q}$ and P. For every pair of subfields $E_0 \subsetneq E_1$ of P, we can check whether (i) of Lemma 1.4.1 holds, telling us whether

there is a rational polynomial which factors over E_1 but not over E_0, and if so, we can search and find one. So we make a list of pairs of these subfields where E_1 is a minimal extension of E_0, and next to each pair, we write down a polynomial in $\mathbb{Q}[X]$ that splits over E_1 but not E_0, if there is one.

So we have this list M of finitely many polynomials. The claim is that if we have an oracle for the splitting set S_F of F restricted to the smallest initial segment of $F[X]$ that contains M (which is a bound that is computable uniformly in p), we can decide whether p has a root in F.

Let E be a subfield of P generated by a single root of p. Then $E_i = \sigma_i(E)$ are exactly the subfields generated by single roots of p, where $\sigma_i \in \mathrm{Gal}(P/\mathbb{Q})$.

Start with the minimal extensions of $\mathbb{Q}$ in P. For each minimal extension K_1 of $\mathbb{Q}$, if q_1 is the polynomial in M that splits over K_1 but not over $\mathbb{Q}$, then we check whether q_1 splits over F. If so, then there is $\sigma_1 \in \mathrm{Gal}(P/\mathbb{Q})$ for which $\sigma_1(K_1) \subseteq F$. (If not, then move onto another minimal extension of $\mathbb{Q}$.)

Then move to the minimal extensions of K_1. For each minimal extension K_2 of K_1, if q_2 is the polynomial in M that splits over K_2 but not over K_1, then we check whether q_2 splits over F. If so, then there is $\sigma_2 \in \mathrm{Gal}(P/\mathbb{Q})$ such that $\sigma_2 \restriction K_1 = \sigma_1 \restriction K_1$ and $\sigma_2(K_2) \subseteq F$. If there is no rational polynomial which splits over K_2 but not over K_1, then K_2 cannot be in F, and neither can anything isomorphic to K_2, so we ignore all the subfields isomorphic to K_2 and all fields containing them.

We continue this way, until either we reach the splitting field P, or we reach some sub-

field K of P where there are no minimal extensions of K for which the corresponding polynomial in M splits over F. In either case, suppose $K = \sigma_n(\ldots\sigma_2(\sigma_1(K_1))\ldots)$. If K contains one of the subfields E_i (which we can check because we know exactly which polynomials in M split over E and which don't), then we will know whether F contains a root of p. F contains a root of p iff K contains an E_i.

Now, $K = P \cap F$, and so the roots of p in F are precisely the roots of p in K, and we can effectively find all the roots of p in K, since K is a finite algebraic extension of $\mathbb{Q}$ and thus has a splitting algorithm by Lemma 1.2.9. And once we have all the roots of p in F, we check which ones of these are also roots of q. ■

1.5 Rabin Images and Reducibilities

There is a third set that can be compared to the root set R_F and the splitting set S_F of a field F under computability-theoretic reductions. This set is the *image* of F under a *Rabin embedding*. We give the definition below.

Definition 1.5.1 Let F and E be computable fields. A function $g : F \to E$ is a *Rabin embedding* if all four of the following hold:

- g is a field homomorphism;
- E is algebraically closed;
- E is algebraic over $g(F)$;
- g is a computable function.

The following is actually a corollary of a theorem of Rabin that can be found in [24]:

Lemma 1.5.2 *For any computable field F, the following are Turing-equivalent:*

1. *the image $g(F)$ of F under any Rabin embedding g;*
2. *the splitting set S_F of F;*
3. *the root set R_F of F.*

So if g is a Rabin embedding of a computable algebraic field F into its algebraic closure, then the image $g(F)$ sits in the same Turing degree as S_F and R_F. The next question is, how does $g(F)$ compare to S_F and to R_F under the bT-reduction?

Theorem 1.5.3 *If F is a computable algebraic field with root set R_F and g is a Rabin embedding of F into its algebraic closure $\overline{F}$, then $R_F \equiv_{bT} g(F)$.*

Proof. First we show that $R_F \leq_{bT} g(F)$: Given a polynomial $p(X)$ in $F[X]$, we want to know whether p has a root in F. Feed p to g to get $(g(p))(X)$ in $(g(F))[X]$. Let $\psi(g(p))$ be the computable function that finds the biggest root of $g(p)$ in $\overline{F}$. Then let $\varphi(p) = \psi(g(p)) + 1$.

Claim: We only need to know $g(F)$ up to the $\varphi(p)$-th element of $\overline{F}$ to decide whether $p \in R_F$.

Proof of claim: We can search through $g(F)$ until we reach the $\varphi(p)$-th element of $\overline{F}$, and by that time we know whether $g(F)$ contains any of the roots of $g(p)$. If $g(F)$ doesn't have any of the roots of $g(p)$, then F doesn't have any of the roots of p,

because g must take roots of p to roots of $g(p)$ and nonroots of p to nonroots of $g(p)$, so $p \notin R_F$. If $g(F)$ has at least one of the roots of $g(p)$, then F must also have one of the roots of p, so $p \in R_F$.

Next we show that $g(F) \leq_{bT} R_F$: Given an element $b \in \overline{\mathbb{Q}}$, we want to know whether $b \in g(F)$. Find the minimal polynomial p for b over $\mathbb{Q}$. Find all the roots of p in $\overline{\mathbb{Q}}$. Consider the subfields of the splitting field of p that are generated by a single root of p, or two roots of p, or three roots of p, etc, up to the subfields generated by $n-1$ roots of p, where $\deg(p) = n$. For each of these finitely many subfields, find a primitive generator α_i, and then find the minimal polynomial q_i for each α_i over $\mathbb{Q}$.

Claim: If we know R_F up to the smallest initial segment of $F[X]$ that contains all the q_i, then we can determine whether $b \in g(F)$.

Subclaim: If we know exactly which q_i have roots in F and which don't, then we can figure out exactly how many roots of p are in F.

Proof of subclaim: For each α_i, ask R_F if q_i has a root in F. If q_i has a root in F, then F contains a subfield isomorphic to $\mathbb{Q}(\alpha_i)$, and this subfield of F contains exactly as many roots of p as $\mathbb{Q}(\alpha_i)$ has. Considering all the q_i together, F contains as many roots of p as are contained in the $\mathbb{Q}(\alpha_i)$ with the greatest number of roots of p in F.

Proof of claim: Once we know how many roots of p are in F, we go and search through F until we find them all. Call these roots b_i. Feed each of the b_i to g and compare $g(b_i)$ to b for each i. If $g(b_i) = b$ for some i, then $b \in g(F)$. If for all i, $g(b_i) \neq b$, then $b \notin g(F)$. ■

Corollary 1.5.4 *If F is a computable algebraic field with splitting set S_F and g is a Rabin embedding of F into its algebraic closure $\overline{F}$, then $S_F \leq_{bT} g(F)$.*

Proof. By Miller's work in [21], we always have $S_F \leq_{bT} R_F$, and by Theorem 1.5.3, we always have $R_F \leq_{bT} g(F)$. So by transitivity, we have $S_F \leq_{bT} g(F)$. ∎

Corollary 1.5.5 *There is a computable algebraic field F with splitting set S_F and Rabin image $g(F)$ for which $g(F) \not\leq_{bT} S_F$.*

Proof. If it were always the case for a computable algebraic field F that $g(F) \leq_{bT} S_F$, then since it is always true that $R_F \leq_{bT} g(F)$, we would have $R_F \leq_{bT} S_F$ by transitivity, which contradicts Theorem 1.3.1. ∎

Now, with respect to m-reducibility, we already know that in a computable algebraic field F, we always have $S_F \leq_m R_F$, while there is a computable algebraic field for which $R_F \not\leq_m S_F$. The remainder of this section will be devoted to determining whether or not we always have m-reductions between R_F and $g(F)$ and between S_F and $g(F)$.

The next reducibility we investigate is m-reducibility from the splitting set S_F of a computable field F to a Rabin image $g(F)$ of F.

Lemma 1.5.6 *If p is a polynomial which is irreducible over an algebraic field F, has prime degree bigger than 2 over F, and has symmetric Galois group over F, then for any two roots α and β of p, we have $F(\alpha) \cap F(\beta) = F$. (In fact, the conclusion still holds even if p doesn't have prime degree over F, but that is beyond the scope of this*

dissertation.)

Proof. Suppose the polynomial p has degree k over F, where k is an odd prime. Then $F(\alpha)$ and $F(\beta)$ are field extensions of F which each have degree k over F. $F(\alpha) \cap F(\beta)$ is also a field extension of F, and sits inside both $F(\alpha)$ and $F(\beta)$, and so its degree over F must divide k, i.e. is either 1 or k. If the degree of $F(\alpha) \cap F(\beta)$ over F is k, then $F(\alpha) = F(\alpha) \cap F(\beta) = F(\beta)$, which in particular means that $\beta \in F(\alpha)$, which is impossible for roots of a polynomial with symmetric Galois group over F. So the degree of $F(\alpha) \cap F(\beta)$ over F is 1, which means that $F(\alpha) \cap F(\beta) = F$. ∎

Lemma 1.5.7 *Let F be a computable algebraic field, and suppose p is a polynomial which has prime degree bigger than 2 over F, is irreducible over F and has symmetric Galois group over F. Let g be a Rabin embedding of F into its algebraic closure, and let $b \in \overline{\mathbb{Q}}$ with $b \notin g(F)$. Let q be the minimal polynomial of b over $\mathbb{Q}$ and let E be the splitting field of q. Then there is a root r of p for which $E \not\subseteq F(r)$.*

Proof. Suppose that for each root r_j of p we have $E \subseteq F(r_j)$. By Lemma 1.5.6, the only elements that lie in $F(r_j)$ for every j already lie in F, so if $E \subseteq F(r_j)$ for every j, then every root of q lies in F, which means in particular that $b \in g(F)$, which is a contradiction. ∎

Theorem 1.5.8 *There exists a computable algebraic field F with splitting set S_F and Rabin image $g(F)$ for which $S_F \not\leq_m g(F)$.*

Proof. We fix a presentation of $\overline{\mathbb{Q}}$, and build F and a Rabin embedding $g : F \to \overline{\mathbb{Q}}$ simultaneously.

Here is the requirement that must be satisfied for each i:

$$\mathcal{R}_i : \text{If } \varphi_i \text{ is total, then } \exists x \, [x \in S_F \iff \varphi_i(x) \notin g(F)].$$

A requirement $\mathcal{R}_i$ with $i \leq s$ "needs attention" at stage $s+1$ if either

- a polynomial p_i has not been chosen for $\mathcal{R}_i$ since its last injury;
- a polynomial $p_{i,s}$ has been chosen for $\mathcal{R}_i$ and $\varphi_i(p_{i,s})$ has converged, but we have not yet done anything further to satisfy this requirement.

As in the proof of Theorem 1.3.1, we refer to D_s as the set of elements of $\overline{\mathbb{Q}}$ that have been enumerated by the end of stage s, and we refer to F_s as the field that these elements generate. Elements are enumerated into D_s with the goal of eventually enumerating the entire field F_s. We also refer to N_s as the set of prime numbers which are no longer allowed to be used as the degree of a witness polynomial for any lower-priority requirement.

The construction:

Stage 0: Let $F_0 = \mathbb{Q}$, $D_0 = \emptyset$, and $N_0 = \{2\}$.

Stage $s+1$: Of all the requirements $\mathcal{R}_i$ with $i < s$ which need attention at this stage, select the one with the least index i.

- If no polynomial p_i has been chosen for $\mathcal{R}_i$ since its last injury, then we choose a polynomial $p_{i,s+1}(X) \in F_s[X]$ which has prime degree over F_s different from any degree on the "off-limits list" N_s, is irreducible over F_s, and has symmetric Galois group over F_s. (The fact that such a polynomial exists is proven in

Theorem 2.15 of [21].) Let P be the splitting field of $p_{i,s+1}$. We enumerate every prime factor of $[P : F_s]$ into our "off-limits list" N_{s+1} so that no lower-priority requirement can choose a polynomial with one of these degrees, and we initialize all lower-priority requirements.

- If a polynomial $p_{i,s}$ has been chosen for $\mathcal{R}_i$ and $\varphi_i(p_{i,s})$ has converged, but we have not yet done anything further to satisfy this requirement, then we act according to two cases:
  - Case 1: $\varphi_{i,s}(p_{i,s}) \downarrow \in g_s(F_s)$. In this case we just let $F_{s+1} = F_s$.
  - Case 2: $\varphi_{i,s}(p_{i,s}) \downarrow \notin g_s(F_s)$. In this case, according to Lemma 1.5.7, we can put a root of $p_{i,s}$ into F_{s+1} without putting $\varphi_{i,s}(p_{i,s})$ into $g_{s+1}(F_{s+1})$. So we find a root of $p_{i,s}$ that doesn't generate the splitting field $E_{i,s}$ of the minimal polynomial $q_{i,s}$ over F_s of $\varphi_{i,s}(p_{i,s})$, and adjoin this root to D_{s+1}. For any roots of $q_{i,s}$ that are forced into F_{s+1} by this move, define their images under g_{s+1} to be roots of $q_{i,s}$ that are not equal to $\varphi_{i,s}(p_{i,s})$. Then we put the degree of $p_{i,s}$ into N_{s+1} and initialize all lower-priority requirements.

Let $F = \bigcup_s F_s$.

Lemma 1.5.9 *For each i, the requirement $\mathcal{R}_i$ is injured only finitely often and acts only finitely often.*

Proof. Injury would happen if, for example, a requirement $\mathcal{R}_e$ needs to keep $\varphi_e(p_e)$

out of $g(F)$, but later, $\mathcal{R}_i$ with $i < e$ puts a root of p_i into F, and putting this root into the field forces $\varphi_e(p_e)$ in $g(F)$. We would need to initialize $\mathcal{R}_e$'s strategy. To make sure a requirement starts over when it needs to, we declare all $\mathcal{R}_e$ for $e > i$ injured as soon as $\mathcal{R}_i$ has acted, regardless of whether $\varphi_e(p_e)$ for some $e > i$ goes into the field because a root of p_i dragged it along.

Just as in Lemma 1.3.3, we prove the current lemma by induction: each requirement is injured only finitely often, and if and when it acts after the stage of its final injury, that will be its last action. ∎

Lemma 1.5.10 *The requirement $\mathcal{R}_i$ is satisfied for every i.*

Proof. We showed in the previous lemma that each requirement $\mathcal{R}_i$ acts only finitely often. Now to show that this last action by $\mathcal{R}_i$ actually does result in its satisfaction: In the case that $\varphi_i(p_i)$ never halts, φ_i is not a total function, and so the hypothesis of the requirement is false. In the case that φ_i halts on input p_i, the construction ensures that $p_i \in S_F$ iff $\varphi_i(p_i) \notin g(F)$. We never end up with $p_i \in S_F$ and $\varphi_i(p_i) \in g(F)$ for two reasons: First, if $\varphi_i(p_i) \downarrow \in g(F)$, we use the sets N_s to guarantee that p_i stays out of S_F forever, and second, if we do end up putting p_i into S_F, we use Lemma 1.5.7 to make sure that we do so without putting $\varphi_i(p_i)$ into $g(F)$. Likewise, we never end up with $p_i \notin S_F$ and $\varphi_i(p_i) \notin g(F)$, because if $\varphi_i(p_i) \downarrow \notin g(F)$, we put p_i into S_F without letting $\varphi_i(p_i)$ into $g(F)$. ∎

Corollary 1.5.11 *There is a computable algebraic field F with root set R_F and Rabin image $g(F)$ for which $R_F \not\leq_m g(F)$.*

Proof. The field and Rabin image constructed in Theorem 1.5.8 works here as well!

■

Next, we look at a computable algebraic field F where there is no m-reduction from a Rabin image $g(F)$ to the root set R_F.

Lemma 1.5.12 *Let F be a computable algebraic field, let q be a polynomial of odd prime degree with coefficients in F which has symmetric Galois group over F, and let p be a polynomial with coefficients in F which is irreducible over F. Fix an algebraic closure $\overline{F}$ of F, let the roots of q be $y_1, y_2, \ldots, y_d$, let the roots of p be $x_1, x_2, \ldots, x_n$. Suppose there is a root of q which generates a root of p. Then $d = n$, and there is a renumbering of the roots of p for which $F(y_i) = F(x_i)$.*

Proof. First we note that if one root of q generates a root of p, every root of q must generate a root of p, by the symmetry of the Galois group of q over F.

Let y_j be an arbitrary root of q. Then by our hypothesis, there is some x_k for which $F(x_k) \subseteq F(y_j)$. Note that $F(x_k)$ cannot be a proper subfield of $F(y_j)$, because $[F(y_j) : F]$ is prime, and that would force $[F(x_k) : F] = 1$, which would mean that x_k is already in F, which is a contradiction. So the only other option is that $F(x_k) = F(y_j)$.

Also note that there cannot be $l \neq k$ for which $F(x_l) = F(y_j) = F(x_k)$: If there were, then there would have to be an automorphism of the splitting field of p which interchanges x_k and x_l and which leaves y_j fixed. But if there were such an automorphism, then its fixed field would contain $F(y_j)$ but not $F(x_k)$; a clear contradiction.

An automorphism of the splitting field of q which moves y_j to another root of q must also move x_k to another root of p. It follows that there are just as many roots of p as there are of q, and each root y_i of q has its own unique pairing with (by renumbering, if necessary) a root x_i of p for which $F(y_i) = F(x_i)$. ■

Notice that it follows from the proof of Lemma 1.5.12 that each root of p has the same degree over F as a root of q does.

Theorem 1.5.13 *There is a computable algebraic field F with root set R_F and Rabin image $g(F)$ for which $g(F) \not\leq_m R_F$.*

Proof. We fix a presentation of $\overline{\mathbb{Q}}$, and build F and a Rabin embedding $g : F \to \overline{\mathbb{Q}}$ simultaneously.

The requirement that must be satisfied for each i:

$$\mathcal{R}_i\text{: If } \varphi_i \text{ is total, then } \exists x\, [x \in g(F) \iff \varphi_i(x) \notin R_F].$$

A requirement $\mathcal{R}_i$ with $i \leq s$ "needs attention" at stage $s+1$ if either

- an element $b_i \in \overline{\mathbb{Q}}$ has not yet been chosen for this requirement since its last injury;
- an element $b_{i,s} \in \overline{\mathbb{Q}}$ has been chosen for this requirement and $\varphi_{i,s}(b_{i,s})$ has converged to an element outside R_F, but we haven't yet done anything further to satisfy the requirement.

Let D_s, N_s, and F_s be as before.

Stage 0: Let $F_0 = \mathbb{Q}$, $D_0 = \emptyset$, and $N_0 = \{2\}$.

Stage $s+1$: Of all the requirements $\mathcal{R}_i$ with $i < s$ which need attention at this stage, select the one with the least index i.

- If an element $b_i \in \overline{\mathbb{Q}}$ has not been chosen for this requirement since its last injury, then we choose an element $b_{i,s+1} \in \overline{\mathbb{Q}}$ such that the minimal polynomial $q_{i,s+1}$ of $b_{i,s+1}$ over $\mathbb{Q}$ has prime degree different from any degree in the "off-limits" list N_s and has symmetric Galois group over $\mathbb{Q}$. We enumerate the degree of $b_{i,s+1}$ over $\mathbb{Q}$ into N_{s+1} and initialize all lower-priority requirements.

- If $\varphi_{i,s}(b_{i,s}) \downarrow$ but we haven't yet done anything further to satisfy $\mathcal{R}_i$, we act according to three cases:

  - Case 1: $\varphi_{i,s}(b_{i,s}) \downarrow \in R_F$. In this case, simply let $F_{s+1} = F_s$

  - Case 2: $\varphi_{i,s}(b_{i,s}) \downarrow \notin R_F$ and the subfield of $\overline{\mathbb{Q}}$ generated by $g_s(F_s)$ and $b_{i,s}$ contains a root of $\varphi_{i,s}(b_{i,s})$. In this case, appealing to Lemma 1.5.12, we find the root r_1 of an irreducible (over F_s) factor of $\varphi_{i,s}(b_{i,s})$ which generates $b_{i,s}$, choose a root $r_2 \neq r_1$ of $\varphi_{i,s}(b_{i,s})$, enumerate r_2 into D_{s+1} (thereby putting $\varphi_{i,s}(b_{i,s})$ into R_F), and define $g_{s+1}(r_2) = r_2$.

  - Case 3: $\varphi_{i,s}(b_{i,s}) \downarrow \notin R_F$ and we can put $b_{i,s}$ into $g_{s+1}(F_{s+1})$ without forcing a root of $\varphi_{i,s}(b_{i,s})$ into F_{s+1}. In this case, find a root t of $q_{i,s}$ such that $\mathbb{Q}(t)$ does not contain any roots of $\varphi_{i,s}(b_{i,s})$, enumerate t into D_{s+1}, and define $g_{s+1}(t) = b_{i,s}$. Enumerate the prime factors of the degree of $\varphi_{i,s}(b_{i,s})$ over

$\mathbb{Q}$ into N_{s+1}, and initialize all lower-priority requirements.

Let $F = \bigcup_s F_s$.

Lemma 1.5.14 *For each i, the requirement $\mathcal{R}_i$ is injured only finitely often and acts only finitely often.*

Proof. The scenarios where injury happens in this construction are very much like the injury scenario described in Lemma 1.5.9, and the simultaneous induction proceeds exactly as in Lemma 1.5.9. ∎

Lemma 1.5.15 *The requirement $\mathcal{R}_i$ is satisfied for every i.*

Proof. We will show that the last action by $\mathcal{R}_i$ results in its satisfaction. In the case that $\varphi_i(b_i)$ never halts, φ_i is not a total function, and so the hypothesis of the requirement is false. In the case that φ_i halts on input b_i, the construction ensures that $b_i \in g(F)$ iff $\varphi_i(b_i) \notin R_F$. We never end up with $b_i \in g(F)$ and $\varphi_i(b_i) \in R_F$, because if we end up putting b_i into $g(F)$, we do so without ever letting $\varphi_i(b_i)$ into R_F; if we either have $\varphi_i(b_i) \downarrow \in R_F$ or end up putting $\varphi_i(b_i)$ into R_F, we use Lemma 1.5.12 to make sure we keep b_i out of $g(F)$ forever. We also never end up with $b_i \notin g(F)$ and $\varphi_i(b_i) \notin R_F$, because if we have $\varphi_i(b_i) \downarrow \notin R_F$, we either put $\varphi_i(b_i)$ into R_F while keeping b_i out of $g(F)$, or we put b_i into $g(F)$ while keeping $\varphi_i(b_i)$ out of R_F, and Lemma 1.5.12 shows us that we can always do one or the other. The reasons we can keep $\varphi_i(b_i)$ out of R_F (in Case 3) or b_i out of $g(F)$ (in Case 2) forever are the sets N_s – the degrees of these elements can never be used again. ∎

Corollary 1.5.16 *Among the reducibilities $\leq_T$, $\leq_{bT}$, $\leq_m$, and $\leq_1$, the following are the strongest which hold for all computable algebraic fields F:*

- $R_F \leq_T S_F$
- $g(F) \leq_T S_F$
- $g(F) \leq_{bT} R_F$
- $R_F \leq_{bT} g(F)$
- $S_F \leq_{bT} g(F)$
- $S_F \leq_1 R_F$.

It might have seemed as though the root set R_F and a Rabin image $g(F)$ of a computable algebraic field F have the same reducibility strength; after all, they are bT-equivalent and can be m-incomparable. But the last two reducibilities in the list above show us that R_F is just slightly stronger than $g(F)$: the splitting set S_F 1-reduces to R_F but not always to $g(F)$.

Chapter 2

Effective Algebraicity

2.1 Introduction

A field algebraic over the rationals $\mathbb{Q}$ has the following property which computability theorists find very nice. Given any element of the field, that element can be mapped to one of only finitely many elements under an automorphism of the field. If the chosen element happens to be in $\mathbb{Q}$, it must be mapped to itself, and if not, it can only be mapped to another root of its own minimal polynomial over $\mathbb{Q}$.

In recent years, there has been much work done in computable structure theory on algebraic fields; most of that work concerns a property of structures called computable categoricity. A structure $\mathcal{M}$ is *computably categorical* if for every computable structure $\mathcal{N}$ isomorphic to $\mathcal{M}$, there is a *computable* isomorphism from $\mathcal{M}$ to $\mathcal{N}$. This definition has been generalized to arbitrary Turing degrees $\mathbf{d}$: A structure $\mathcal{M}$ is $\mathbf{d}$-*computably categorical* if for every computable structure $\mathcal{N}$ isomorphic to $\mathcal{M}$, there is a $\mathbf{d}$-*computable* isomorphism from $\mathcal{M}$ to $\mathcal{N}$. We now know that there must be a

degree $\mathbf{d}$ with $\mathbf{d}' \leq \mathbf{0}''$ such that every computable algebraic field is $\mathbf{d}$-computably categorical [20]. We also know that there is a computable algebraic field which is not even $\mathbf{0}'$-computably categorical. The property of computable categoricity for algebraic fields is Π^0_4-complete [11]. In addition to the work done on computable categoricity, we also know that the orbit relation on a computable algebraic field is Π^0_2 [23], that the isomorphism type of an algebraic field is completely determined by its Σ_1-theory [9], and that there are computable algebraic fields which, when considered as subfields of the algebraic closure of the rationals, have singleton degree spectra [9].

The concept of algebraicity was developed for fields, but then generalized, so that other kinds of structures may be "algebraic" as well. Algebraicity, as defined below, implies that each element of the structure has a finite orbit under automorphisms. In this article, we look at some of the above-mentioned results for fields in the context of some other "algebraic" structures. Many of the results that hold for fields turn out to hold for these other structures as well, but some of them don't. Just how similar are these structures? Where must we draw the line? And what is the significance of the differences between these structures? We end up attributing the differences to an effectiveness property which algebraic fields have and which the other algebraic structures we consider lack; our definition of an "effectively algebraic" structure will essentially say that there is a computable bound on the size of the orbit under automorphisms. We contrast these other structures with fields to see how algebraicity and effectiveness interact with each other.

In particular, we consider other structures which have the property that their elements have only finitely many possible images under an automorphism. Consider a finite-branching tree. Under a tree automorphism, the root must be mapped to itself, and a node at level n of the tree can only be mapped to another node at level n of the tree. Since each level of the tree is finite, this tree has the desired property.

Also consider a connected graph where each node has only finitely many other nodes adjacent to it, and suppose we add a constant, naming a particular node, to the language. Since this fixed node must map to itself under any automorphism, and a node adjacent to this node must map to another node adjacent to it under any automorphism (and so on), each node in this graph has only finitely many possibilities for where to go under under an automorphism of this graph. So this graph also is "algebraic."

In the following three definitions, we introduce the two new structures which will be compared and contrasted in this paper, and we make precise the terminology we use throughout. See Definitions 1.1.1 and 1.1.2 for the relevant definitions for *computable fields*.

Definition 2.1.1 A *computable tree under predecessor* is a tree with domain ω and a total computable "predecessor function" P where $P(x) = y$ means that y is the immediate predecessor of x. (We keep the function total by defining the predecessor of the root to be the root itself.) A *finite-branching tree* is a tree where each node has finitely many immediate successors. The *branching function* of a finite-branching

tree is a function that takes as input a node on the tree and outputs the number of immediate successors of that node.

We consider trees under predecessor rather than trees as partial orders because more of the results from fields carry over to the trees under predecessor. For example, no tree of height ω under $\leq$ is computably categorical [18], and since we consider only finite-branching trees, the only trees which would be computably categorical are the finite ones (which are intrinsically non-interesting). We note that all trees under predecessor have height $\leq \omega$, since a node at level ω would have no immediate predecessor.

Definition 2.1.2 A *computable graph* is a symmetric, irreflexive graph with domain ω and computable edge relation. A *pointed graph* is a graph with one constant node in a signature with one constant symbol added. A *finite-valence graph* is a graph where each node has finitely many neighbors (i.e. finitely many nodes adjacent to it). The *valence function* of a finite-valence graph is a function that takes as input a node on the graph and outputs the number of neighbors of that node.

In §2.2, we examine the computability-theoretic similarities between the three types of structures, and in §2.3 and §2.4, we examine some properties which distinguish the fields from the trees and the graphs. In §2.5, we explore the significance of those distinguishing properties.

2.2 Computable Categoricity in Algebraic Structures

The theorems we prove in this section about similarities between connected, finite-valence, pointed graphs, finite-branching trees under predecessor, and algebraic fields are mostly analogues of theorems of Russell Miller in [20]. In the next section, we will discuss the differences between algebraic fields and these other structures.

The splitting set of a computable algebraic field is a computably enumerable set (it is defined using only existential quantifiers over a computable matrix) and hence $\mathbf{0}'$-computable. Here are the corresponding results for computable, finite-valence graphs and computable, finite-branching trees under the predecessor function.

Lemma 2.2.1 *If V is the valence function of a computable, finite-valence graph, then $V \leq_T \emptyset'$, and if B is the branching function of a computable, finite-branching tree under predecessor, then $B \leq_T \emptyset'$.*

Proof. Given a node x on the graph and a $\mathbf{0}'$-oracle, we ask whether there is a node adjacent to x. If there is, then we ask whether there is a node different from the first one adjacent to x. We keep going until we get a "no" answer. We know we will get a "no" answer after only finitely many questions because we're dealing with a finite-valence graph (this is why a $\mathbf{0}'$-oracle is sufficient). We do the same thing for a node x on the tree, asking whether x has a successor different from the ones we've already counted, until we get a "no." ∎

Isomorphic computable algebraic fields have Turing-equivalent splitting sets [20, Cor.

2.8]. The corresponding result for isomorphic computable, connected, finite-valence, pointed graphs states that they have Turing-equivalent valence functions, and this result is an easy relativization of the following theorem.

Theorem 2.2.2 *No computable, connected, finite-valence, pointed graph with non-computable valence function is isomorphic to a graph with computable valence function.*

Proof. Let $\mathcal{G}$ and $\mathcal{H}$ be computable, connected, finite-valence graphs. Suppose $\mathcal{G}$ and $\mathcal{H}$ are isomorphic via an isomorphism that takes the constant node $a_0 \in \mathcal{G}$ to the constant node $b_0 \in \mathcal{H}$. Suppose furthermore that $\mathcal{G}$ has computable valence function.

Define $R_n = \{x \in \mathcal{H} : \text{dist}(x, b_0) = n\}$.

We claim first that the function f defined by $f(n) = |R_n|$ is a total computable function: We can compute the number nodes at distance 1 from a_0, because these are just the nodes adjacent to a_0, and the valence function of G is computable. Since $\mathcal{G}$ and $\mathcal{H}$ are isomorphic with $a_0 \mapsto b_0$, there are just as many nodes at distance 1 from b_0 as there are from a_0. For any n, we can compute the number of nodes at distance n from a_0 in $\mathcal{G}$ (these will be the nodes which are adjacent to the nodes at distance $n-1$ from a_0 but which are not at distance less than n from a_0) because the valence of nodes in $\mathcal{G}$ is computable. Since $\mathcal{G}$ and $\mathcal{H}$ are isomorphic with $a_0 \mapsto b_0$, there must be just as many nodes at distance n from b_0 in $\mathcal{H}$.

We claim next that for each n, we can enumerate R_n (uniformly in n) and know when we're done: We know how many nodes are at distance 1 from b_0 (by the first claim),

and so we search for them, and once we've found all of them, we know exactly which ones they are and we know that there aren't any more. For each n, if we know the contents of each of the finite sets $R_1, R_2, \ldots, R_{n-1}$, and if we know how many nodes are in R_n, we can go and search for the nodes that are at distance n from b_0, and once we find them all, we know exactly which ones they are and that there aren't any others.

The last claim is that the valence function of $\mathcal{H}$ must be computable: Let $b \in \mathcal{H}$. The node b must appear in R_n for some n because $\mathcal{H}$ is connected. So we enumerate all the sets R_n until we find b. Say $b \in R_i$. Then we enumerate R_{i+1}, and we find which nodes in R_{i-1}, R_i, and R_{i+1} are neighbors of b. And then we know the valence of b.

So if $\mathcal{G}$ is a computable, connected, finite-valence graph which has computable valence function, then any graph isomorphic to $\mathcal{G}$ also has computable valence function. ■

The following is an easy relativization of the preceding theorem.

Corollary 2.2.3 *Isomorphic computable, connected, finite-valence, pointed graphs have Turing-equivalent valence functions.*

As another corollary, we also have the following, which will be explained immediately below.

Corollary 2.2.4 *Isomorphic computable, finite-branching trees under predecessor have Turing-equivalent branching functions.*

We take a moment to note here that every finite-branching tree under the predecessor

function is a finite-valence, pointed graph, in the following sense: the root of the tree is analogous to the constant node in that it must always be mapped to itself under an automorphism of the structure, and there is an edge between two nodes of the graph if and only if one is the predecessor of the other in the tree. (The converse is not obvious, since a graph can contain a cycle and thus may not look like a tree, even when the constat node is put at the bottom and the graph grows upward.) Thus, when we state a 'for-all' result concerning these structures, we will state it for all finite-valence graphs (and thus it will hold for every finite-branching tree under predecessor), and when we state a 'there-exists' result, we will state it for a finite-branching tree under predecessor (which is also an example of a finite-valence graph). Viewed this way, Corollary 2.2.4 follows immediately from Corollary 2.2.3.

To see the importance of the assumption that $\mathcal{G}$ is connected and has finite valence, we will show in Theorem 2.2.5 that the result fails when the graph need not be connected, and we will show in Theorem 2.2.6 that result fails when the graph (shown as a tree for convenience) need not be finite-valence.

Theorem 2.2.5 *There is a computable, finite-valence graph with noncomputable valence function that is isomorphic to a graph with computable valence function.*

Note: This existence theorem is not stated for a tree because we build our graph here to be disconnected.

Proof. We will build the graph $\mathcal{G}$ with computable valence and the graph $\mathcal{H}$ with noncomputable valence simultaneously. Begin each graph as in the figure below:

Start enumerating nodes $a_0, a_1, \ldots$ in $\mathcal{G}$ and $b_0, b_1, \ldots$ in $\mathcal{H}$, and give each one two neighbors right away.

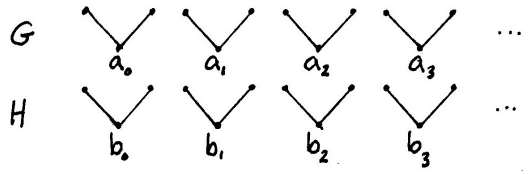

Figure 2.1: beginning of the constructions of $\mathcal{G}$ and $\mathcal{H}$

Then, if and when $\varphi_i(b_i) \downarrow = 2$, we add a third neighbor to b_i. To keep the two graphs isomorphic, we add a new node a_i' to $\mathcal{G}$ and give it three neighbors, and we also add a new node b_i' to $\mathcal{H}$ and give it two neighbors so that an isomorphism can take a_i to b_i'.

Below is what would happen if and when $\varphi_0(b_0) \downarrow = 2$.

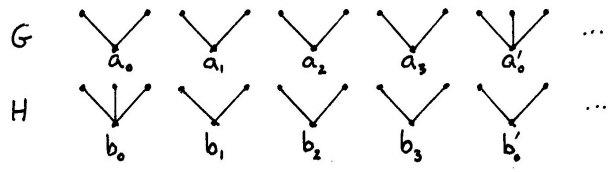

Figure 2.2: $\mathcal{G}$ and $\mathcal{H}$ in the case that $\varphi_0(b_0) \downarrow = 2$

This diagonalization precludes each φ_i from computing the valence of $\mathcal{H}$, yet $\mathcal{G}$ clearly has computable valence. ■

Theorem 2.2.6 *There is a computable tree under the predecessor function with non-computable branching function which is isomorphic to a tree with computable branching function.*

Proof. In the picture of the graph $\mathcal{G}$ immediately above, introduce a new "root" node which has each of the nodes in $\mathcal{G}$ of valence 2 or 3 as its successors:

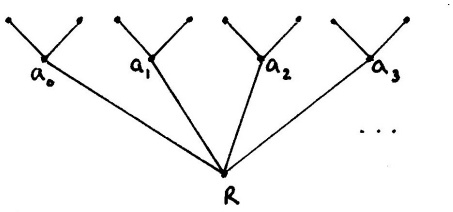

Figure 2.3: tree with noncomputable branching having a copy with computable branching

The resulting tree is no longer finite-branching, and it happens to have finite height. The branching is still computable, as long as we allow the symbol "∞" to be an output of a computable branching function. ■

The next several results from fields which we generalize to graphs and trees have to do with computable categoricity, and more generally, **d**-computable categoricity where **d** is a Turing degree.

Definition 2.2.7 A computable structure $\mathcal{S}$ is said to be *computably categorical* if every computable structure isomorphic to $\mathcal{S}$ is computably isomorphic to $\mathcal{S}$. $\mathcal{S}$ is said to be **d**-*computably categorical* if every computable structure isomorphic to $\mathcal{S}$ is **d**-computably isomorphic to $\mathcal{S}$.

There exists a computable algebraic field which is not computably categorical, yet has a splitting algorithm [20, Thm. 3.4]. Here is the corresponding result for trees:

Theorem 2.2.8 *There is a computable finite-branching tree under the predecessor function which is not computably categorical, yet has computable branching function.*

Proof. We will build, simultaneously, two copies of a computable tree, and diagonalize against all computable possibilities for the isomorphism between them. The corresponding argument for graphs was shown to the author by Valentina Harizanov; the author merely adapted it for trees.

For the first copy of the tree, begin with an infinite "spine.". Label the successor of the root a_0, and its successor a_1, and so on. While we continue to build the infinite spine, we start building two new "arm" branches from each node $a_0, a_1, \ldots$. Label the new successors of a_i with a_i^L and a_i^R.

For the second copy of the tree, do the same, except replace every "a" with a "b".

If and when $\varphi_i(a_i^L)$ and $\varphi_i(a_i^R)$ converge to b_i^L and b_i^R (respectively or irrespectively), stop building the arms from a_i and b_i. If $\varphi_i(a_i^L) \downarrow = b_i^L$ and $\varphi_i(a_i^R) \downarrow = b_i^R$, then give the last node of the a_i^L and b_i^R arms two successors, and we give the last node of the a_R and b_i^L arms three successors. If $\varphi_i(a_i^L) \downarrow = b_i^R$ and $\varphi_i(a_i^R) \downarrow = b_i^L$, do the reverse. This way, we force φ_i to converge to the wrong thing.

Here's what our tree looks like:

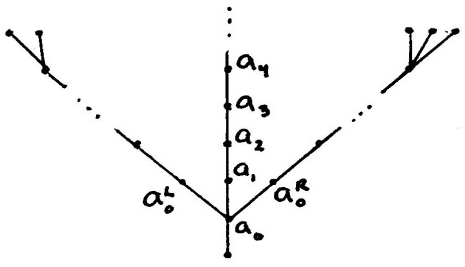

Figure 2.4: tree which isn't computably categorical but has computable branching

The second copy has its fingers arranged either the same way or the opposite way, so the trees are isomorphic, but not via φ_0. If $\varphi_0(a_0^L) \uparrow$ or $\varphi_0(a_0^R) \uparrow$, then we simply have a single infinite stem above each of a_0^L, a_0^R, b_0^L, and b_0^R, so the copies again are isomorphic. Finally, no node in the tree at stage s acquires a successor after stage $s+1$, so the branching is computable. ■

It turns out that we also have a tree with the opposite properties – one which is computably categorical but doesn't have a computable branching function.

Theorem 2.2.9 *There is a computable finite-branching tree under the predecessor function which has noncomputable branching function, yet is computably categorical.*

Proof. Begin with one infinite branch, and label the nodes $a_0, a_1, \ldots$. Let C be an arbitrary noncomputable c.e. set, and let the branching function on the nodes of the infinite branch be as follows:

$$B(a_n) = \begin{cases} 1, & \text{if } n \notin C; \\ 2, & \text{if } n \in C. \end{cases}$$

Here is an example of such a tree:

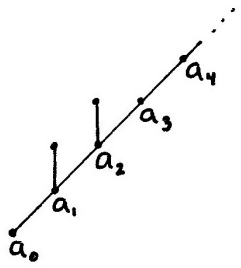

Figure 2.5: tree with noncomputable branching but is computably categorical

In the above example, $0 \notin C$, $1 \in C$, $2 \in C$, $3 \notin C$, $4 \notin C$, etc.

In any computable copy, one can identify the main stem, since it contains exactly those nodes with successors. Computable categoricity follows. ■

There exists a computable algebraic field which is not even $\mathbf{0}'$-computably categorical [20, Thm. 4.1]. Here is the corresponding result for trees:

Theorem 2.2.10 *There is a computable finite-branching tree under the predecessor function which is not even $\mathbf{0}'$-computably categorical.*

Proof. We build, simultaneously, two copies of a computable tree, and diagonalize against all limit-computable possibilities for the isomorphism between them.

For the first copy of the tree, begin with the same infinite "spine" which we saw in the proof of Theorem 2.2.8. Label the successor of the root a_0, and its successor a_1, and so on. While we continue to build the spine, we give each node a_i two new successors. Label the new successors of a_i with a_i^L and a_i^R. For the second copy of the tree, do the same thing, except replace every "a" with a "b".

Let f_i be the function (if any) computably approximated as $f_i(x) = \lim_s \varphi_i(x, s)$.

If either of $f_i(a_i^L)$ and $f_i(a_i^R)$ is undefined, then f_i is not a total function and cannot possibly be an isomorphism, so we need not act to defeat this function. If $f_i(a_i^L)$ and $f_i(a_i^R)$ converge to anything other than b_i^L and b_i^R (respectively or irrespectively), then again f_i has no chance of being an isomorphism from the first tree to the second.

Suppose now that $f_{i,s}(a_i^L)\downarrow$ and $f_{i,s}(a_i^R)\downarrow$ to b_i^L and b_i^R, respectively or irrespectively. Say, without loss of generality, that $f_{i,s}(a_i^L)\downarrow = b_i^L$ and $f_{i,s}(a_i^R)\downarrow = b_i^R$. Then, to defeat

f_i (for the moment) and to keep the two trees isomorphic, we give a_i^R and b_i^L each one successor.

Example: $i = 0$

Figure 2.6: example for $i = 0$ at some stage s

The functions f_i can change their minds this time, if, for example, $f_{i,s+1}(a_i^L) \downarrow \neq f_{i,s}(a_i^L)$. If at some later stage r, $f_{i,r}(a_i^L) \downarrow = b_i^R$ and $f_{i,r}(a_i^R) \downarrow = b_i^L$, then we must defeat f_i once again, while keeping the trees isomorphic. So we give a_i^L one successor, a_i^R's successor one successor, b_i^R one successor, and b_i^R's successor one successor.

Continuing the $i = 0$ example:

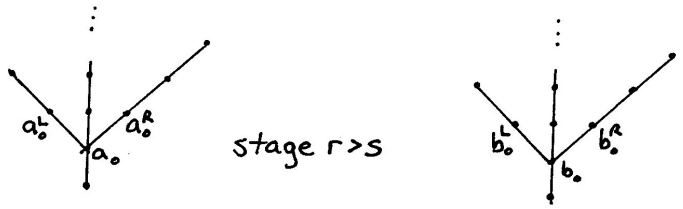

Figure 2.7: example for $i = 0$ at some stage $r > s$

If f_i changes its mind yet again and reflects an isomorphic mapping from the first tree to the second, we use the same strategy to defeat him while keeping the trees isomorphic.

Continuing the $i = 0$ example:

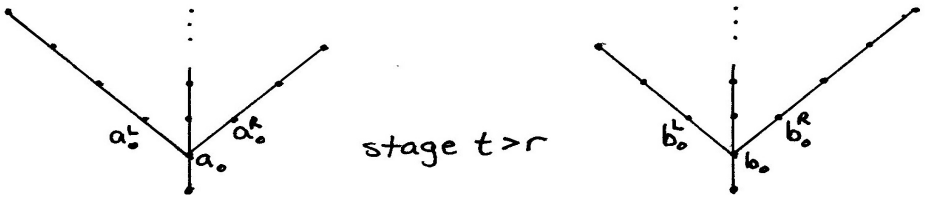

Figure 2.8: example for $i = 0$ at some stage $t > r$

We keep extending the "arms" as the function f_i keeps mimicking an isomorphism. If f_i never settles on a particular choice, then the two trees are still isomorphic, but the isomorphism is not limit-computable by f_i in either case. ■

If two algebraic fields $\mathcal{E}$ and $\mathcal{F}$ are isomorphic and we have an embedding from $\mathcal{E}$ to $\mathcal{F}$, this embedding must actually be an isomorphism [23, Cor. 3.7]. Here is the corresponding result for connected, finite-valence, pointed graphs.

Lemma 2.2.11 *If $\mathcal{G}$ and $\mathcal{H}$ are connected, finite-valence, pointed graphs that are isomorphic to each other via an isomorphism which maps the constant node a_0 in $\mathcal{G}$ to the constant node b_0 in $\mathcal{H}$, then every embedding f of $\mathcal{G}$ into $\mathcal{H}$ with $f(a_0) = b_0$ must be surjective.*

Proof. Let f be an embedding of $\mathcal{G}$ into $\mathcal{H}$ with $f(a_0) = b_0$, and let $b \in \mathcal{H}$.

The node b in $\mathcal{H}$ must be at distance n from b_0 for some n because the graph is connected. Say there are k nodes in $\mathcal{H}$ at distance n from b_0. Then there must also be k nodes in $\mathcal{G}$ at distance n from a_0, because there is an isomorphism that takes a_0 to b_0, and f will map the k nodes in $\mathcal{G}$ to the k nodes in $\mathcal{H}$, injectively and surjectively due to the finiteness of these valences. So b will be the image of some node a in $\mathcal{G}$ at distance n from a_0. ■

Lemma 2.2.12 *If two connected, finite-valence, pointed graphs embed into each other, then they are isomorphic.*

Proof. Let $\mathcal{G}$ and $\mathcal{H}$ be connected, finite-valence, pointed graphs, and suppose $g : \mathcal{G} \hookrightarrow \mathcal{H}$ and $h : \mathcal{H} \hookrightarrow \mathcal{G}$ are pointed graph embeddings. The composite function $g \circ h$ is an embedding of $\mathcal{H}$ into itself, and we know by Lemma 2.2.11 that this embedding must be surjective. Thus the embedding h is surjective, making h an isomorphism from $\mathcal{H}$ onto $\mathcal{G}$. ■

The following theorem is a result of Julia Knight and will be helpful in the proofs of some of our results later on in this section and in the next section.

Theorem 2.2.13 ([15], theorem 1.4′) *Let $\mathcal{A}$ be a structure in a relational language. Then exactly one of the following holds:*

(i) For every degree $\mathbf{d} > deg(\mathcal{A})$, there is some $\mathcal{B} \cong \mathcal{A}$ such that $deg(\mathcal{B}) = \mathbf{d}$.

(ii) There is a finite set $S \subseteq \omega$ such that all permutations of ω which fix S pointwise are automorphisms of $\mathcal{A}$.

The preceding theorem states that unless a structure is "automorphically trivial" (i.e., satisfies condition (ii)), its spectrum is closed upwards in the Turing degrees.

Definition 2.2.14 The *spectrum of a structure* $\mathcal{A}$ is defined to be the set $Spec(\mathcal{A})$ of Turing degrees of structures isomorphic to $\mathcal{A}$. $Spec(\mathcal{A}) = \{\deg(\mathcal{B}) : \mathcal{B} \cong \mathcal{A}\}$.

For more discussion of the spectrum of a structure, see [10].

Frolov, Kalimullin, and Miller [9, Thm. 2] showed that for every algebraic field $\mathcal{F}$,

there exists a set V_F such that $Spec(\mathcal{F}) = \{\mathbf{d} : V_F \text{ is c.e. in } \mathbf{d}\}$. In other words, the ability of a degree to enumerate the set V_F is precisely what the degree needs to be able to compute an isomorphic copy of the field. It turns out that the same is true for every connected, finite-valence, pointed graph.

Theorem 2.2.15 *For every connected, finite-valence, pointed graph $\mathcal{G}$, there exists a set V_G such that $Spec(\mathcal{G}) = \{\mathbf{d} : V_G \text{ is c.e. in } \mathbf{d}\}$.*

Proof. Consider the set of finite, connected subgraphs of $\mathcal{G}$ which contain the constant node. Let V_G be this set.

Suppose $\mathbf{d}$ is a degree which can enumerate V_G. We build a graph $\mathcal{H} \cong \mathcal{G}$ using a $\mathbf{d}$-oracle. As soon as an element $\mathcal{G}_0$ appears in V_G, we let $\mathcal{H}_0 \cong \mathcal{G}_0$. When another element $\mathcal{G}_1$ appears in V_G, if $\mathcal{G}_1$ extends $\mathcal{G}_0$, then we let $\mathcal{H}_1 \cong \mathcal{G}_1$, but if not, we wait for an element of V_G which extends both $\mathcal{H}_0$ and $\mathcal{G}_1$. Suppose $\mathcal{G}_k$ extends both $\mathcal{H}_0$ and $\mathcal{G}_1$. Then we let $\mathcal{H}_1 \cong \mathcal{G}_k$ (noting that $\mathcal{H}_1$ extends $\mathcal{H}_0$ via the embedding of $\mathcal{H}_0$ into $\mathcal{G}_k$). Next, we wait for an element of V_G which extends $\mathcal{H}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$, ..., $\mathcal{G}_{k-1}$. Suppose $\mathcal{G}_n$ does this, where $n > k$. Then we let $\mathcal{H}_2 \cong \mathcal{G}_n$. Next, we wait for an element of V_G which extends $\mathcal{H}_2$, $\mathcal{G}_{k+1}$, $\mathcal{G}_{k+2}$, ..., $\mathcal{G}_{n-1}$. And so on. Let $\mathcal{H} = \bigcup_i \mathcal{H}_i$. The graph $\mathcal{H}$ is a $\mathbf{d}$-computable copy of $\mathcal{G}$, and so by Theorem 2.2.13, we have a copy of $\mathcal{G}$ of degree $\mathbf{d}$.

For the other half, suppose we have a $\mathbf{d}$-computable copy $\mathcal{H}$ of $\mathcal{G}$. Choose a node x_0 in $\mathcal{H}$, use a $\mathbf{d}$-oracle to find the shortest path from x_0 to the constant node of $\mathcal{H}$, and enumerate each initial segment (beginning with the fixed node) of that path. Do the

same thing for another node x_1 in $\mathcal{H}$, and then also enumerate each initial segment of the union of those two paths. Continue like this until we've exhausted all the nodes in $\mathcal{H}$. Every finite graph embeddable in $\mathcal{H}$, and thus embeddable in $\mathcal{G}$, will eventually appear on our list. So $\mathbf{d}$ can enumerate V_G. ■

The next few results from fields which we generalize to graphs and trees still have to do with $\mathbf{d}$-computable categoricity, and they are also corollaries of the Low Basis Theorem of Jockusch and Soare [14]. We first introduce the definition of an *isomorphism tree*, following Definition 5.1 in [20].

Definition 2.2.16 Let $\mathcal{G}$ and $\mathcal{H}$ be computable, connected, finite-valence, pointed graphs which are isomorphic to each other. Fix an enumeration of $\mathcal{G}$, starting with the constant node, such that each node on the list (except the very first one) is adjacent in $\mathcal{G}$ to one of the nodes that came before it on the list. An element of the *isomorphism tree* $T_{\mathcal{G},\mathcal{H}}$ *from* $\mathcal{G}$ *to* $\mathcal{H}$ is an embedding of a finite initial segment of the enumeration of $\mathcal{G}$ into $\mathcal{H}$. The isomorphism tree $T_{\mathcal{G},\mathcal{H}}$, therefore, consists of all such partial embeddings of the enumeration into $\mathcal{H}$, mapping the constant node in $\mathcal{G}$ to the constant node in $\mathcal{H}$. These form a tree under $\subseteq$.

Definition 2.2.17 Let $\mathcal{R}$ and $\mathcal{S}$ be computable finite-branching trees under predecessor which are isomorphic to each other. Fix an enumeration of $\mathcal{R}$, beginning with the root, such that each node on the list (except the first one) is a successor of one of the nodes that came before it on the list. An element of the *isomorphism tree* $T_{\mathcal{R},\mathcal{S}}$ *from* $\mathcal{R}$ *to* $\mathcal{S}$ is an embedding of a finite initial segment of the enumeration of $\mathcal{R}$ into

$\mathcal{S}$. The isomorphism tree $T_{\mathcal{R},\mathcal{S}}$, therefore, consists of all such partial embeddings of the enumeration into $\mathcal{S}$, mapping the root of $\mathcal{R}$ to the root of $\mathcal{S}$. These form a tree under $\subseteq$.

Remark 2.2.18 *The valence function of a connected, finite-valence, pointed graph $\mathcal{G}$ can compute the branching of the isomorphism tree $T_{\mathcal{G},\mathcal{H}}$, where $\mathcal{H} \cong \mathcal{G}$. Likewise, the branching function of a finite-branching tree $\mathcal{R}$ can compute the branching of the isomorphism tree $T_{\mathcal{R},\mathcal{S}}$, where $\mathcal{S} \cong \mathcal{R}$.*

Paths through the isomorphism tree correspond to total embeddings of $\mathcal{G}$ into $\mathcal{H}$ or $\mathcal{R}$ into $\mathcal{S}$, and this correspondence preserves Turing degrees: the degree of the path through the tree is the degree of the embedding. Lemma 2.2.11 shows that these embeddings are precisely the isomorphisms from $\mathcal{G}$ onto $\mathcal{H}$ or $\mathcal{R}$ onto $\mathcal{S}$, justifying the term "isomorphism tree."

The following is a relativization of a theorem which appears in Jockusch and Soare [14, Thm. 2.1].

Theorem 2.2.19 ("Low Basis Theorem," relativized) *Fix any set $V \subseteq \omega$ and let $\mathfrak{T}$ be the class of all those V-computable finite-branching trees $\mathcal{T}$ for which the function $s : \mathcal{T} \to \omega$ given by $s(\sigma) = |\{n \in \omega : \sigma^\frown\langle n\rangle \in \mathcal{T}\}|$ is also V-computable. Then there exists a Turing degree $\mathbf{d}$ with $\mathbf{d}' \leq deg(V')$ such that every infinite $\mathcal{T} \in \mathfrak{T}$ has a $\mathbf{d}$-computable path. (Such a degree is known as a PA-degree relative to V.) Furthermore, there exist two PA-degrees $\mathbf{d}_0$ and $\mathbf{d}_1$ relative to V such that every degree which is both $\leq \mathbf{d}_0$ and $\leq \mathbf{d}_1$ is $\leq deg(V)$ as well.*

For every computable algebraic field $\mathcal{F}$ with root set R, there exists a degree $\mathbf{d}$ low relative to R (i.e. $\mathbf{d}' \leq \deg(R')$) such that $\mathcal{F}$ is $\mathbf{d}$-computably categorical [20, Cor. 5.4]. Here is the corresponding result for graphs:

Corollary 2.2.20 *For every computable, connected, finite-valence, pointed graph $\mathcal{G}$ with valence function V, there exists a degree $\mathbf{d}$ low relative to V (i.e. $\mathbf{d}' \leq deg(V')$) such that $\mathcal{G}$ is $\mathbf{d}$-computably categorical.*

Proof. By Definition 2.2.16, an isomorphism between any two computable isomorphic copies $\mathcal{G}_0$ and $\mathcal{G}_1$ of $\mathcal{G}$ corresponds to a path through the isomorphism tree $T_{\mathcal{G}_0,\mathcal{G}_1}$. For every computable graph $\mathcal{G}_0 \cong \mathcal{G}$, Corollary 2.2.3 tells us that the valence function of $\mathcal{G}_0$ has the same degree as V, and Theorem 2.2.19 provides a degree $\mathbf{d}$ low relative to V for which $T_{\mathcal{G},\mathcal{G}_0}$ has a $\mathbf{d}$-computable path, which corresponds to a $\mathbf{d}$-computable isomorphism. ∎

Miller used Theorem 2.2.19 to show that there is a degree $\mathbf{d}$ with $\mathbf{d}' \leq \mathbf{0}''$ such that every computable algebraic field is $\mathbf{d}$-computably categorical, and in fact, every PA-degree relative to $\mathbf{0}'$ is such a $\mathbf{d}$ [20, Cor. 5.5]. Here is the corresponding result for graphs:

Corollary 2.2.21 *There is a degree $\mathbf{d}$ with $\mathbf{d}' \leq \mathbf{0}''$ such that every computable, connected, finite-valence, pointed graph is $\mathbf{d}$-computably categorical. In fact, every PA-degree relative to $\mathbf{0}'$ is such a $\mathbf{d}$.*

Proof. We saw in Lemma 2.2.1 that the valence function V of a computable, finite-valence graph always satisfies $\deg(V) \leq \mathbf{0}'$. So then $\deg(V)' \leq \mathbf{0}''$, and the result

follows from the Low Basis Theorem. ■

There is a low degree **d** such that every computable algebraic field with a splitting algorithm is **d**-computably categorical [20, Cor. 5.6]. The analogous result also holds for graphs.

Corollary 2.2.22 *There is a low degree* **d** *such that every computable, connected, finite-valence, pointed graph with computable valence function is* **d***-computably categorical.*

Proof. Apply the Low Basis Theorem with $V \leq_T \emptyset$. ■

Definition 2.2.23 The *categoricity spectrum* of a computable structure $\mathcal{S}$ is the set of degrees **d** such that $\mathcal{S}$ is **d**-computably categorical. The *degree of categoricity* of a computable structure is the least degree in its categoricity spectrum, if that least degree exists.

Several recent papers, including [2], [6], and [20], have considered categoricity spectra. The first example of a computable structure with no degree of categoricity appeared in Theorem 4.1 in [20], and the proof given there also applies to connected finite-valence graphs.

Corollary 2.2.24 *If $\mathcal{G}$ is a computable, connected, finite-valence, pointed graph with computable valence function and $\mathcal{G}$ is not computably categorical, then $\mathcal{G}$ has no degree of categoricity. More generally, for any computable, connected, finite-valence, pointed graph $\mathcal{G}$, the degree of categoricity of $\mathcal{G}$, if it exists, must be computable from the*

valence function of $\mathcal{G}$.

We note here that Theorem 2.2.8 builds a finite-branching tree under predecessor, and hence a connected, finite-valence, pointed graph, with no degree of categoricity; the tree we built there has computable branching function but is not computably categorical. So Corollary 2.2.24 is not vacuous.

There is no least degree $\mathbf{d}$ such that every computable algebraic field is $\mathbf{d}$-computably categorical [20, Cor. 5.10]. Computable, connected, finite-valence graphs turn out to have the same property.

Corollary 2.2.25 *There is no least degree* $\mathbf{d}$ *such that every computable, connected, finite-valence, pointed graph is* $\mathbf{d}$*-computably categorical. Likewise, there is no such degree for the computable, connected, finite-valence, pointed graphs which have computable valence function.*

Proof. By Corollary 2.2.21, such a degree $\mathbf{d}$ would have to lie below all PA-degrees relative to $\mathbf{0}'$. Then according to Theorem 2.2.19 with $V = \emptyset'$, $\mathbf{d}$ would have to lie below $\mathbf{0}'$. However, we saw in Theorem 2.2.10 that this can never happen. And the same argument with $V = \emptyset$ and Theorem 2.2.8 gives us the result for the graphs with computable valence function. ∎

The last result we mention here in this section is purely model-theoretic. An algebraic field is determined, up to isomorphism, by its Σ_1-theory [9, Thm. 2]. The following is the result for connected, finite-valence, pointed graphs.

Theorem 2.2.26 *Two connected, finite-valence, pointed graphs are isomorphic if and only if they satisfy the same Σ_1 sentences.*

Proof. One direction is clear. For the other direction, let $\mathcal{G}$ and $\mathcal{H}$ be infinite, connected, finite-valence, pointed graphs, and suppose they satisfy the same Σ_1 sentences.

Consider the "isomorphism tree" for $\mathcal{G}$ embedding into $\mathcal{H}$ (without assuming in advance that $\mathcal{G}$ and $\mathcal{H}$ are isomorphic). Note that the assumption about Σ_1 sentences proves this to be an infinite finite-branching tree, which by König's Lemma must have an infinite path, i.e. an embedding of $\mathcal{G}$ into $\mathcal{H}$. If we do the same for an embedding of $\mathcal{H}$ into $\mathcal{G}$ and then apply Lemma 2.2.12, we must have $\mathcal{G}$ and $\mathcal{H}$ isomorphic. ■

2.3 Degree Spectra of Relations

In this section we will address the computability-theoretic differences between algebraic fields and connected, finite-valence, pointed graphs (along with finite-branching trees under predecessor). We will build up to the main difference (Theorems 2.3.13 and 2.3.14) with some preliminary definitions and facts.

Definition 2.3.1 The *orbit* of an element a of a structure $\mathcal{M}$ is defined to be the following set:

$$\{b \in \mathcal{M} \ : \ \exists \sigma \in \mathrm{Aut}(\mathcal{M}) \ \ \sigma(a) = b\}\,.$$

Note the similarity this definition has with Definition 2.4.5: the orbit is the equivalence class of a under the orbit relation.

Consider building an isomorphism tree between two computable algebraic fields. When examining the choices of elements to which to map a given element a of one of the fields, we see that there is a pre-imposed computable bound on the size of the orbit of a; a can only be sent to a root of its own minimal polynomial over $\mathbb{Q}$, and that minimal polynomial has fixed degree, no matter which elements of the field have not actually been enumerated into the field yet. (Not all roots of that minimal polynomial are necessarily in the orbit of a, even assuming the field contains such roots; the point is simply that we have a computable bound on the orbit size.) We can compute the minimal polynomial of any given element over $\mathbb{Q}$, and hence, we can compute its degree.

Now consider building an isomorphism tree between two computable, connected, finite-valence, pointed graphs where the constant node of one has already been mapped to the constant node of the other. When examining the choices of elements to which to map a given element x of one of the graphs, we have no such pre-imposed computable bound on the size of the orbit of x. We know its orbit must be finite; because the graph has finite valence, there are only finitely many nodes at the same distance from the constant node as x. But because nodes may continue to be enumerated into the graphs, we have no idea how big it might be.

This difference between computable, connected, finite-valence, pointed graphs and computable algebraic fields has some degree-theoretic significance which will appear at the end of this section, but first we have some background to cover.

We begin with the idea of a structure which is 'universal' for connected, finite-valence graphs in the sense that every connected, finite-valence graph can be embedded into this structure. We already have a structure which is 'universal' for algebraic fields of characteristic 0; every algebraic field can be embedded into the algebraic closure $\overline{\mathbb{Q}}$ of $\mathbb{Q}$.

Definition 2.3.2 A *countable random graph* is a countable graph with the property that for every two disjoint finite sets of nodes in the graph, there is a node (or equivalently, infinitely many nodes) adjacent to every node in the first set and not adjacent to any node in the second set.

Remark 2.3.3 *A note about how we build our computable random graphs: The ordered pairs of disjoint finite sets of nodes in the random graph can be put in an ω-ordering. As the random graph is built, each new node is placed into the graph on behalf of an already-existing pair of disjoint finite sets of nodes and is made to be adjacent to each node in the first set and nonadjacent to each node in the second set. Each new node is a witness for exactly one such pair, and we enumerate nodes into the graph in such a way that each pair has an infinite "arsenal" of witness nodes. So, the "arsenals" of witness nodes partition the graph.*

There is only one countable random graph up to isomorphism; this can be shown by a back-and-forth construction. This countable random graph is indeed 'universal' for countable, connected, finite-valence graphs; every finite-valence graph can be embedded into the random graph. For finite-branching trees under predecessor, we have a

'universal' structure which is much easier to visualize than the random graph; every finite-branching tree can be embedded into the tree $\omega^{<\omega}$.

Are there other structures besides the random graph and the tree $\omega^{<\omega}$ which are 'universal' for connected finite-valence graphs and finite-branching trees under predecessor, respectively? Of course. However, the following definition will give some intuition as to why we chose these structures as opposed to others which might do the same job.

Definition 2.3.4 A structure $\mathcal{M}$ is said to be ultrahomogeneous if the following are equivalent when A is a finite subset of $\mathcal{M}$ and $\overline{a}$ and $\overline{b}$ are tuples of elements from $\mathcal{M}$:

- $\overline{a}$ and $\overline{b}$ have the same atomic type over A;
- $\exists\, \sigma \in \mathrm{Aut}_A(\mathcal{M})$ with $\sigma(\overline{a}) = \overline{b}$.

In layman's terms, in an ultrahomogeneous structure, elements which look the same really are the same, up to automorphism. Both the random graph and the tree $\omega^{<\omega}$ under predecessor are ultrahomogeneous. The idea is that ultrahomogeneity is necessary in order for every computable structure to have a computable embedding into each computable copy of the universal structure.

Theorem 2.3.5 *The tree $\omega^{<\omega}$ is the only countable ultrahomogeneous tree under predecessor which is universal for finite-branching trees.*

Proof. Let $\mathcal{T}$ be a countable tree under predecessor which is both ultrahomogeneous

and universal for finite-branching trees. Because $\mathcal{T}$ must contain a copy of every finite-branching tree, $\mathcal{T}$ must have infinitely many nodes at level 1. The nodes of $\mathcal{T}$ at level 1 must all have the same number of immediate successors because of ultrahomogeneity, and because $\mathcal{T}$ must contain a copy of every finite-branching tree, no finite number of successors will do. So each node at level 1 must have infinitely many successors at level 2. And so on, recursively. Thus $\mathcal{T} \cong \omega^{<\omega}$. ■

Theorem 2.3.6 *The random graph is the only countable ultrahomogeneous graph which is universal for connected finite-valence graphs.*

Proof. Let $\mathcal{G}$ be a countable graph which is both ultrahomogeneous and universal for connected finite-valence graphs. Choose two finite disjoint sets A and B of nodes in $\mathcal{G}$, and consider the edge relation among these nodes. There is a finite (and thus finite-valence) graph which consists of precisely this subgraph of $\mathcal{G}$ along with two additional nodes: one adjacent to every node in A and none of the nodes in B, and one adjacent to every node in B and none of the nodes in A. We can make this graph connected by inserting an edge between the two additional nodes. Because $\mathcal{G}$ is universal, it must embed this graph, and so the image of the graph will contain some subgraph of $\mathcal{G}$ which looks like $A \cup B$ along with these two extra nodes. Now, $\mathcal{G}$ is ultrahomogeneous, so if for the copy of $A \cup B$ in the image of the finite graph there is a node adjacent to each node in the copy of A and no node in the copy of B, and vice versa, there must be such nodes for the actual A and B. Since A and B were chosen arbitrarily, this has to be the case for every pair of finite disjoint sets of

nodes in $\mathcal{G}$, and so $\mathcal{G}$ will have the randomness property. Thus $\mathcal{G}$ is isomorphic to the random graph. ■

It turns out that we already have terminology for the unique countable ultrahomogeneous structure which is universal for a particular class of structures. W. Hodges [12] gave it the name Fraïssé Limit, after the famous theorem of Fraïssé, which appears below as Theorem 2.3.7. We must first note, however, that although Theorem 2.3.7 stipulates that the class of structures be a class of finitely-generated structures, it still gives us the universal structure we want: for example, if we start with the class of all finite-branching trees, then take the subclass of all finitely-generated finite-branching trees, and then take the Fraïssé limit of this subclass, this Fraïssé limit (which turns out to be the tree $\omega^{<\omega}$) is a structure which is countable, ultrahomogeneous, and also universal for the *entire* class of all finite-branching trees.

Theorem 2.3.7 ([12], Theorem 6.1.2) *Let L be a countable signature and let $\mathcal{K}$ be a non-empty countable set of finitely generated L-structures which satisfy the following three conditions:*

- *If $\mathcal{A} \in \mathcal{K}$ and $\mathcal{B}$ is a finitely generated substructure of $\mathcal{A}$, then $\mathcal{B}$ is isomorphic to some structure in $\mathcal{K}$;*
- *If $\mathcal{A} \in \mathcal{K}$ and $\mathcal{B} \in \mathcal{K}$, then there is $\mathcal{C} \in \mathcal{K}$ such that both $\mathcal{A}$ and $\mathcal{B}$ are embeddable in $\mathcal{C}$;*
- *If $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ are in $\mathcal{K}$ and $e : \mathcal{A} \hookrightarrow \mathcal{B}$, $f : \mathcal{A} \hookrightarrow \mathcal{C}$ are embeddings, then there are $\mathcal{D} \in \mathcal{K}$ and embeddings $g : \mathcal{B} \hookrightarrow \mathcal{D}$ and $h : \mathcal{C} \hookrightarrow \mathcal{D}$ such that $g \circ e = h \circ f$.*

Then there is an L-structure $\mathcal{F}$ (called the Fraïssé Limit of $\mathcal{K}$), unique up to isomorphism, such that $\mathcal{F}$ is countable and ultrahomogeneous, and the set $\mathcal{K}$ is precisely the class of finitely generated substructures of $\mathcal{F}$.

Lemma 2.3.8 *Let $\mathcal{T}$ be a finite-branching tree under predecessor. If f and g each embed $\mathcal{T}$ into $\omega^{<\omega}$, then there is an automorphism of $\omega^{<\omega}$ mapping $f(\mathcal{T})$ onto $g(\mathcal{T})$.*

Proof. We will define the (likely noncomputable) automorphism $h : \omega^{<\omega} \to \omega^{<\omega}$ recursively as follows. Map the root of ω^ω to itself. For $x \in \omega^{<\omega}$ where the predecessor y of x is already in the domain of h, we have two cases: If $x \in f(\mathcal{T})$, then define $h(x) = g(f^{-1}(x))$. Otherwise, define $h(x)$ to be the least successor of $h(y)$ not in $g(\mathcal{T})$ which is not already in the range of h. ∎

Lemma 2.3.9 *Let $\mathcal{G}$ be a connected, finite-valence graph, and let $\mathfrak{R}$ be a copy of the random graph. If f and g each embed $\mathcal{G}$ into $\mathfrak{R}$, then there is an automorphism of $\mathfrak{R}$ mapping $f(\mathcal{G})$ onto $g(\mathcal{G})$.*

Proof. We will define the automorphism $h : \mathfrak{R} \to \mathfrak{R}$ as follows. For an arbitrary $n \in \mathfrak{R}$, if $n \in f(\mathcal{G})$, then define $h(n) = g(f^{-1}(n))$. If $n \notin f(\mathcal{G})$, find the smallest distance r (which is either 1 or 2) from n to a node in $f(\mathcal{G})$, let $\{m_0, m_1, \ldots, m_k\}$ be the nodes at distance r from n which are in $f(\mathcal{G})$, find $g(f^{-1}(m_i))$ for each of these, and define $h(n)$ to be the least node in $\mathfrak{R}$ outside $g(\mathcal{G})$ which has the same relationship to the nodes $g(f^{-1}(m_i))$ as n has to the nodes m_i. ∎

Definition 2.3.10 The *spectrum of a relation R on a computable structure $\mathcal{M}$* is defined to be the set $\mathrm{DgSp}_{\mathcal{M}}(R)$ of Turing degrees of all images of R under isomorphisms

from $\mathcal{M}$ onto other computable structures.

Keep in mind that even though Definitions 2.2.14 and 2.3.10 each define a kind of spectrum of degrees, they are different definitions. They are related, however, as we will soon see in Theorems 2.3.13 and 2.3.14.

Theorem 2.3.11 *For every finite-branching tree $\mathcal{T}$ under predecessor, there is an embedding of $\mathcal{T}$ into a computable copy of $\omega^{<\omega}$ such that the image of this embedding, as a relation on $\omega^{<\omega}$, has the same degree as $\mathcal{T}$.*

Proof. Let $\mathcal{T}$ have degree $\mathbf{d}$, and let D be an arbitrary set of degree $\mathbf{d}$.

Construction of the embedding: Send the root of $\mathcal{T}$ to the root of $\omega^{<\omega}$. No node of $\mathcal{T}$ gets mapped into $\omega^{<\omega}$ unless its predecessor has already been mapped into $\omega^{<\omega}$. When we choose a node at level k of $\mathcal{T}$ to send into $\omega^{<\omega}$, say the node is chosen at stage n of the construction. If $k-1 \in D$, we send this node to the $(2n)$-th least node above its predecessor's image, and if $k-1 \notin D$, we send it to the $(2n+1)$-st least node above its predecessor's image.

Claim 1: We can compute D from an oracle for the image of $\mathcal{T}$.

Proof of Claim 1: Given $n \in \omega$, if we want to know whether n is in D, we find a node at level n in $\omega^{<\omega}$ which has at least one successor in the image of $\mathcal{T}$. Check the position of the least of these successors (how many nodes less than him are above his predecessor). Say he is the k-th least node above his predecessor. If k is even, then $n \in D$, and if k is odd, then $n \notin D$.

Claim 2: We can compute the image of $\mathcal{T}$ from a D-oracle.

Proof of Claim 2: Given a node $x \in \omega^{<\omega}$, if we want to know whether x is in the image of $\mathcal{T}$, we locate the level of x in $\omega^{<\omega}$ and we locate its position n above its predecessor. If n is even, write $n = 2m$ for some m, and if n is odd, write $n = 2m+1$ for some m. Run the D-computable construction of the embedding up to the end of stage m. If x isn't in the image by that point, then it won't ever go in.

Thus the image of $\mathcal{T}$, as a relation on $\omega^{<\omega}$, has degree $\mathbf{d}$. ∎

Theorem 2.3.12 *For every connected, finite-valence, pointed graph $\mathcal{G}$, there is an embedding of $\mathcal{G}$ into a computable, pointed copy of the random graph $\mathcal{H}$ such that the image of this embedding, as a relation on $\mathcal{H}$, has the same degree as $\mathcal{G}$.*

Proof. Let $\mathcal{G}$ have degree $\mathbf{d}$, and let D be an arbitrary set of degree $\mathbf{d}$.

Construction of the embedding: At stage 0, map the constant node a_0 of $\mathcal{G}$ to the constant node (which will now be called $g(a_0)$) in $\mathcal{H}$. At stage $n+1$, choose a node of $\mathcal{G}$ which is adjacent to something already mapped, and call this node a_{n+1}. Say that a_{n+1} is adjacent to each node in some subset $A \subseteq \{a_0, \ldots, a_n\}$ and not adjacent to any node in $\{a_0, \ldots, a_n\} \setminus A$. In the arsenal (see Remark 2.3.3) of nodes adjacent to each node in $g(A)$ and not adjacent to any node in $\{g(a_0), \ldots, g(a_n)\} \setminus g(A)$, map a_{n+1} to the $(2n)$-th least node if $n \in D$, and to the $(2n+1)$-st least node if $n \notin D$.

Claim 1: We can compute D from an oracle for the image of $\mathcal{G}$.

Proof of Claim 1: Given $n \in \omega$, if we want to know whether n is in D, we search

through $\mathcal{H}$ until we find a node in $g(\mathcal{G})$ in the m-th position in its arsenal where $m = 2n$ or $m = 2n + 1$. (Such a node must occur in $\mathcal{H}$ because this is how we coded D into the image of $\mathcal{G}$.) If $m = 2n$, then $n \in D$, and if $m = 2n + 1$, then $n \notin D$.

Claim 2: We can compute the image of $\mathcal{G}$ from a D-oracle.

Proof of Claim 2: Given a node $x \in \mathcal{H}$, if we want to know whether x is in the image of $\mathcal{G}$, we find the unique arsenal in $\mathcal{H}$ which contains x, and find x's position in the arsenal. Say x is the m-th least node in its arsenal. If m is even, write $m = 2k$ for some k, and if m is odd, write $m = 2k + 1$ for some k. Run the D-computable construction of the embedding up to the end of stage $k + 1$. If x isn't in the image by then, it won't ever be.

Thus the image of $\mathcal{G}$, as a relation on $\mathcal{H}$, has degree $\mathbf{d}$. ■

Theorem 2.3.13 *Let $\mathcal{T}$ be an infinite computable finite-branching tree under the predecessor function and let $\mathcal{C}$ be a computable copy of $\omega^{<\omega}$ under the predecessor function. Then for every embedding f of $\mathcal{T}$ into $\mathcal{C}$, if we consider $f(\mathcal{T})$ as a relation on $\mathcal{C}$, then $DgSp_C(f(\mathcal{T})) = Spec(\mathcal{T})$.*

Proof. Because of Lemma 2.3.8, all we need to show is that there *exists* an embedding f of $\mathcal{T}$ into $\mathcal{C}$ such that if we consider $f(\mathcal{T})$ as a relation on $\mathcal{C}$, $DgSp_C(f(\mathcal{T})) = Spec(\mathcal{T})$; for any other embedding $g : \mathcal{T} \to \mathcal{C}$, there is an automorphism of $\mathcal{C}$ sending $f(\mathcal{T})$ to $g(\mathcal{T})$, and thus $DgSp_C(g(\mathcal{T})) = DgSp_C(f(\mathcal{T}))$.

Let $\mathbf{d} \in Spec(\mathcal{T})$. Then we have a presentation $\mathcal{T}_d$ of $\mathcal{T}$ of degree $\mathbf{d}$. We use Theorem

2.3.11 to get an embedding of $\mathcal{T}_d$ into $\mathcal{C}$ where the image of $\mathcal{T}_d$ has degree $\mathbf{d}$ as a subtree of $\omega^{<\omega}$. We can do this for each degree in $Spec(\mathcal{T})$, but to show that $\mathbf{d} \in DgSp_{\omega^{<\omega}}(f(\mathcal{T}))$, we need to show that if we did this for isomorphic trees $\mathcal{T}_d$ and $\mathcal{T}_r$ of degrees $\mathbf{d}$ and $\mathbf{r}$, respectively, where f embeds $\mathcal{T}_d$ into $\omega^{<\omega}$ and g embeds $\mathcal{T}_r$ into $\omega^{<\omega}$, then there is an automorphism of $\omega^{<\omega}$ which is an isomorphism of $f(\mathcal{T}_d)$ onto $g(\mathcal{T}_r)$. This follows from Lemma 2.3.8.

Now let $\mathbf{d} \in DgSp_C(f(\mathcal{T}))$. This means there is a copy $\mathcal{C}_0$ of $\mathcal{C}$ which has a subtree $\mathcal{T}_0 \cong f(\mathcal{T})$ such that $\deg(\mathcal{T}_0) \leq \mathbf{d}$. But then $\mathcal{T}_0$ as a structure is just a copy of $\mathcal{T}$, and since it has degree $\leq \mathbf{d}$, $\mathbf{d} \in Spec(\mathcal{T})$ by Theorem 2.2.13. ∎

Theorem 2.3.14 *Let $\mathcal{G}$ be an infinite computable, connected, finite-valence, pointed graph and let $\mathfrak{R}$ be a computable copy of the random graph. Then for every embedding f of $\mathcal{G}$ into $\mathfrak{R}$, if we consider $f(\mathcal{G})$ as a relation on $\mathfrak{R}$, then $DgSp_{\mathfrak{R}}(f(\mathcal{G})) = Spec(\mathcal{G})$.*

Proof. The proof here is the same as the proof of Theorem 2.3.13, except we replace each occurrence of the word "tree" with the word "graph," and instead of using Lemma 2.3.8, we use Lemma 2.3.9. ∎

If an algebraic field $\mathcal{F}$ is normal over its prime subfield $\mathbb{Q}$, then no matter what embedding f we choose to use to embed $\mathcal{F}$ into $\overline{\mathbb{Q}}$, $DgSp_{\overline{\mathbb{Q}}}(f(\mathcal{F}))$ contains exactly one degree [9, Cor. 4]. In fact, even if $\mathcal{F}$ is "almost normal" [9, Defn. 1] over $\overline{\mathbb{Q}}$, $DgSp_{\overline{\mathbb{Q}}}(f(\mathcal{F}))$ is still a singleton, for any embedding $f : \mathcal{F} \hookrightarrow \overline{\mathbb{Q}}$. We can't get a singleton spectrum for a finite-branching tree under predecessor or a connected, finite-valence, pointed graph, as long as these structures are infinite. We don't have

a concept analogous to normality for these trees and graphs, so it makes sense that we don't get the same kind of result that we have for fields.

Corollary 2.3.15 *Under the conditions of Theorems* 2.3.14 *and* 2.3.13*, where* $\mathcal{G}$ *is an infinite connected, finite-valence, pointed graph and* $\mathcal{T}$ *is an infinite finite-branching tree, we cannot get a singleton spectrum for* $DgSp_{\mathfrak{R}}(f(\mathcal{G}))$ *or for* $DgSp_C(f(\mathcal{T}))$*.*

Proof. The structures $\mathcal{G}$ and $\mathcal{T}$ are infinite, so $Spec(\mathcal{G})$ and $Spec(\mathcal{T})$ are closed upwards, by Theorem 2.2.13. ∎

2.4 The Isomorphism Problem

Definition 2.4.1 The *index set* for a class $\mathfrak{K}$ of structures is the set

$$\{e : \varphi_e \text{ is the characteristic function of a structure in } \mathfrak{K}\}$$

where the characteristic function of a structure is defined to be the characteristic function of its atomic diagram.

The index set for algebraic fields is known to be Π^0_2 (and in fact, is Π^0_2-complete) [23]. Since there is only one function (and no relations) in the atomic diagram of a finite-branching tree under predecessor, we can express the index set for the class $\mathfrak{T}$ of such trees, up to computable isomorphism, as:

$$\{i : \varphi_i \text{ is the predecessor function of a structure in } \mathfrak{T}\}.$$

This set is Π^0_3, as its predicate can be written as the conjunction of the following four statements:

- $\forall x\ \exists s\ \varphi_{i,s}(x)\downarrow$;
- $\exists x\ \exists s\ [\varphi_{i,s}(x) = x]\ \&\ \forall y\ \forall x\ \forall t\ [(\varphi_{i,t}(x) = x\ \&\ \varphi_{i,t}(y) = y) \longrightarrow y = x]$;
- $\forall x\ \exists n\ \exists s\ \left[\varphi_{i,s}^{n}(x) = \varphi_{i,s}^{n+1}(x)\right]$;
- $\forall x\ \exists k\ \forall z\ \forall s\ [\varphi_{i,s}(z) = x \longrightarrow z < k]$.

It can also be shown that this set is Π_3^0-complete, but we omit those details here.

Definition 2.4.2 ([1], Definition 3.1) If $A \subseteq B$ and Γ is a complexity class (e.g., Π_2^0), then A is defined to be Γ *within* B if there is some $M \in \Gamma$ such that $A = M \cap B$.

Definition 2.4.3 The *isomorphism problem* for a class $\mathfrak{K}$ of structures is the set

$$\{(i,j) : \varphi_i \text{ and } \varphi_j \text{ are characteristic functions of isomorphic structures in } \mathfrak{K}\}.$$

The isomorphism problem for computable algebraic fields is Π_2^0 within the index set for these fields; i.e. isomorphism between two computable algebraic fields is a Π_2^0 property. It turns out that isomorphism between two computable, finite-branching trees under predecessor (or two computable finite-valence, pointed graphs) is also a Π_2^0 property.

Theorem 2.4.4 *The isomorphism problem for computable, finite-branching trees under predecessor is* Π_2^0 *within the index set for such trees.*

Proof. In the notation of Definition 2.4.2, A is the isomorphism problem for computable, finite-branching trees under predecessor, and if C is the index set for these trees, then $B = C \times C$. We need to show that there is a Π_2^0 relation M such that

$A = M \cap B$.

Suppose we have two computable, finite-branching trees $\mathcal{T}$ and $\mathcal{R}$ under predecessor, and suppose that at each stage n of the construction of $\mathcal{T}$, there is a stage m of the construction of $\mathcal{R}$ for which there is a predecessor-tree embedding $f_n : \mathcal{T}_n \hookrightarrow \mathcal{R}_m$. We produce a predecessor-tree embedding $f : \mathcal{T} \hookrightarrow \mathcal{R}$.

We define f on the root of $\mathcal{T}$ to be the root of $\mathcal{R}$.

For a node y_1 on level 1 of $\mathcal{T}$, the set $\{f_n(y_0) : n \in \omega\}$ is finite: all its elements must be on level 1 of $\mathcal{R}$. So the sequence $0 < 1 < 2 < \cdots$ has an infinite subsequence $s_0^1 < s_1^1 < s_2^1 < \cdots$ of stages such that $f_{s_n^1}(y_0)$ is the same for all n. We define $f(y_0)$ to be this value (or the least such, if there are more than one).

For a node y_k of the tree for which f has been defined on its predecessor, the set $\left\{f_{s_n^k}(y_k)\right\}$ is finite. So the sequence $s_0^k < s_1^k < s_2^k < \cdots$ has an infinite subsequence $s_0^{k+1} < s_1^{k+1} < s_2^{k+1} < \cdots$ of stages such that $f_{s_n^{k+1}}(y_k)$ is the same for all n. We define $f(y_k)$ to be this value (or the least such).

This defines f on all of $\mathcal{T}$, and since its restriction to $\mathcal{T}_n$ is a tree embedding for every n, it is clear that f embeds $\mathcal{T}$ into $\mathcal{R}$.

Similarly, if at each stage m of the construction of $\mathcal{R}$ there is also a stage n of the construction of $\mathcal{T}$ for which there is an embedding $g_m : \mathcal{R}_m \hookrightarrow \mathcal{T}_n$, a similar argument builds an embedding $g : \mathcal{R} \hookrightarrow \mathcal{T}$. And if there is an embedding in each direction, we know by Lemma 2.2.12 there is an isomorphism $h : \mathcal{T} \to \mathcal{R}$.

So the existence of an isomorphism from one tree to another is equivalent to the following statement

$$\forall n\ \exists m\ (\mathcal{T}_n \hookrightarrow \mathcal{R}_m\ \&\ \mathcal{R}_n \hookrightarrow \mathcal{T}_m)$$

which is Π^0_2. Let M be the set $\{(a,b) : \forall s\ \exists t\ (\mathcal{T}_{a,s} \hookrightarrow \mathcal{T}_{b,t}\ \&\ \mathcal{T}_{b,s} \hookrightarrow \mathcal{T}_{a,t})\}$. Now, $A \subseteq M \cap B$ because if (i,j) is a pair of indices of isomorphic finite-branching trees under predecessor, then (i,j) certainly belong to both M and B. Also, $M \cap B \subseteq A$ because if (a,b) is a pair of indices for two finite-branching trees under predecessor which also lies in M, then by what we've just shown, the trees given by the indices (a,b) are actually isomorphic. Thus $A = M \cap B$ with M Π^0_2.

Notice that the proof above is really just König's Lemma: We define the embedding tree from $\mathcal{T}$ into $\mathcal{R}$ just by analogy to the isomorphism tree (see Definition 2.2.17). We point out that paths through the embedding tree correspond precisely to embeddings. If there is no path, then by König's Lemma, the embedding tree must be finite, and so there is a level of the embedding tree with no nodes, i.e. some finite substructure of $\mathcal{T}$ does not embed into $\mathcal{R}$. ■

Definition 2.4.5 The orbit relation on a structure $\mathcal{M}$ is the set

$$R_{\mathcal{M}} = \{(x,y) : \exists \sigma \in \mathrm{Aut}(\mathcal{M})\ [\sigma(x) = y]\}.$$

The orbit relation on a computable algebraic field has been shown to be Π^0_2 [23, §6]. We show, as a corollary of Theorem 2.4.4, that the orbit relation on a computable finite-branching tree under predecessor is likewise Π^0_2.

Corollary 2.4.6 *The orbit relation on a computable, finite-branching tree under predecessor is* Π^0_2.

Proof. Two nodes a and b of the tree $\mathcal{T}$ are in the same orbit if and only if

(i) every finite subtree $\mathcal{T}_0$ of $\mathcal{T}$ with $a \in \mathcal{T}_0$ has a tree-embedding $g : \mathcal{T}_0 \hookrightarrow \mathcal{T}$ with $g(a) = b$, and

(ii) every finite subtree $\mathcal{R}_0$ of $\mathcal{T}$ with $b \in \mathcal{R}_0$ has a tree-embedding $h : \mathcal{R}_0 \hookrightarrow \mathcal{T}$ with $h(b) = a$.

The conditions in (i) and (ii) are both Π^0_2 statements, and according to the proof of Theorem 2.4.4, they are sufficient conditions for having an automorphism of $\mathcal{T}$ which maps a to b. ∎

We saw in Theorem 2.2.26 that connected, finite-valence, pointed graphs, just like algebraic fields, are determined, up to isomorphism, by their Σ_1-theory.

An algebraic field is not just determined by its Σ_1-theory; it is determined by its *single-quantifier* Σ_1-theory, and indeed by the set of formulas which hold in the field and are of the form $\exists x\ p(x) = 0$ as p ranges over $\mathbb{Q}[X]$. The reason behind this is the "Theorem of the Primitive Element" [5, Thm. 14.25], which states that each finitely generated algebraic extension of the rationals is in fact generated by a single element algebraic over the rationals. This is not true for a connected, finite-valence, pointed graph or a finite-branching tree under predecessor.

Theorem 2.4.7 *There exist two finite-branching trees under predecessor which are*

not isomorphic, yet satisfy exactly the same one-quantifier Σ_1 sentences.

Proof. With a single existential quantifier, we can say $\exists x R(x)$, where R is a quantifier-free predicate. The only atomic predicates in one variable are statements of the form $P^n(x) = P^m(x)$, where P is the predecessor function of the tree. For $0 \leq n < m$, the formula $P^n(x) = P^m(x)$ is true of exactly those x at levels $\leq n$ in the tree. All quantifier-free predicates are built from atomic predicates and their negations. In other words, $R(x)$ can only give us finitely or cofinitely many possibilities for the level of x in the tree. So as long as two trees have the same height (finite or countably infinite), they will satisfy exactly the same one-quantifier Σ_1 sentences.

In the case of finite-valence pointed graphs, where we don't have any functions, the only nontrivial atomic predicates in one variable are $E(x, c)$, $x = c$, and their negations, where E is the edge relation on the (symmetric, irreflexive, connected) graph and c is the constant node. So, up to one-quantifier-Σ_1-equivalence, there are only three classes of these graphs:

(i) graphs containing only c;

(ii) graphs containing ≥ 2 nodes, all (except c) adjacent to c; and

(iii) graphs containing a node x with $x \neq c$ and x not adjacent to c.

Any two graphs in the same category will satisfy exactly the same one-quantifier-Σ_1 sentences, but may not be isomorphic. ∎

2.5 Effective Algebraicity

The key to understanding the model-theoretic reason why these differences in the degree spectra occur lies in the concept of algebraic types. The following definition was inspired by field theory [26], but we'll see that it can be used in other contexts as well.

Definition 2.5.1 An *algebraic type* in a theory $\mathfrak{T}$ is a type which has only finitely many realizations in each model of $\mathfrak{T}$. An *algebraic structure* is a structure whose elements realize only algebraic types.

Given an element of an algebraic field, we can find a quantifier-free algebraic formula which generates the atomic type of the element; we find the minimal polynomial $p(X)$ of the element over $\mathbb{Q}$ or over $\mathbb{F}_p$, and write $p(X) = 0$. (Note that this formula may not generate a complete type. For example, it doesn't tell us whether an element has a square root in the field. However, over the theory ACF_0 or ACF_p, it does generate a complete type.)

Definition 2.5.2 An algebraic structure is *pointwise-identifiable* if there is a computable function which takes an element b of the structure as input and outputs a formula which generates the atomic type of b over a particular theory. This formula is called an *identifying formula* for b.

Computable algebraic fields, computable finite-branching trees under predecessor and computable, connected, finite-valence, pointed graphs are pointwise-identifiable. In

the field, the identifying formula of an element is its minimal polynomial over $\mathbb{Q}$ or $\mathbb{F}_p$. In the tree, the identifying formula of a node b looks something like $P(P(P(b))) = P(P(P(P(b)))) \neq P(P(b))$, where P is the predecessor function. The preceding formula says that x is at level 3 in the tree.

Sometimes an identifying formula knows how many elements can possibly satisfy it, and sometimes it doesn't. The theory of fields says that the type generated by the minimal polynomial can be realized at most n times, where n is the degree of that polynomial, in *any* model of the theory. Given an element of a finite-branching tree under predecessor, however, the best we can do is find a partial type satisfied by the element, for example,"I am on level 3 of the tree." This partial type is not algebraic in the sense that there is no n for which the type can be realized at most n times in any model of the theory of trees under predecessor. To turn this partial type into an algebraic type, we could say, for example, "I am on level 3 of the tree and there are at most 2 nodes on level 3." If a is the element in question and P is the predecessor function on the tree, we can write this as

$$P(P(P(a))) = P(P(P(P(a)))) \neq P(P(a)) \quad \&$$

$$\forall x_0 \forall x_1 \forall x_2$$

$$[\, \forall i \leq 2\, [P(P(P(x_i))) = P(P(P(P(x_i)))) \neq P(P(x_i))] \Longrightarrow \exists i < j \leq 2\, (x_i = x_j)\,].$$

This is an algebraic type; it can be realized at most twice in any model of the theory of trees under predecessor. But how can we know, given this type and a node on such a tree, whether that node satisfies this type? How can we know how many nodes

are on a given level of the tree? The point is that we can't. These graphs and trees *are* algebraic structures; the complete types satisfied by their elements are indeed algebraic types. But the algebraicity of these structures is not *effective*. There is no way to *check* whether a given algebraic type is satisfied by a given element, and this is essentially due to the presence of quantifiers in the formulas of the type.

There is another difference between the algebraic fields and the finite-branching trees under predecessor (or connected, pointed, finite-valence graphs), and this one has to do with the "closures" of these structures which we spoke of in §3. Notice that the "closure" of an algebraic field of characteristic zero is the field $\overline{\mathbb{Q}}$, which is itself an algebraic structure, while the "closure" of a finite-branching tree under predecessor (resp. a connected, pointed, finite-valence graph) is the tree $\omega^{<\omega}$ (resp. the pointed countable random graph), which is not an algebraic structure. Taking the "closure" of these non-effectively algebraic structures destroys the finiteness of the orbit size. This leads us to offer the following further definitions and theorem.

Definition 2.5.3 A *weakly effective presentation* of a class of computable structures is a uniformly computable enumeration of the structures, possibly with repetitions, where the set of generators of each structure is c.e. (The "weakness" of the presentation is that at any given stage of the enumeration, we may not have all the generators of a particular structure, even if the structure is finitely generated.)

We now formally define the notion of "effective algebraicity:"

Definition 2.5.4 An algebraic structure $\mathcal{A}$ is *effectively algebraic* if there is a com-

putable function f such that $f(a)$ is a bound on the size of the orbit of the element a for each $a \in \mathcal{A}$.

Definition 2.5.5 A class of algebraic structures is defined to be a *closed class* if it satisfies the three conditions in Theorem 2.3.7.

Definition 2.5.6 A *uniform class* $\mathfrak{P}$ *of effectively algebraic structures* is defined to be a class of algebraic structures in a fixed theory, with a weakly effective presentation, which has the following properties:

- $\mathfrak{P}$ is a closed class of structures;
- There is a computable function g such that if φ_e enumerates a weakly effective presentation of $\mathfrak{P}$, then $g(\langle e, i, z\rangle)$ is a uniform computable bound on the size of the orbit of the element z in the i-th structure in that enumeration.

Definition 2.5.7 The *closure* of a closed class of countable algebraic structures is defined to be the Fraïssé limit (see Theorem 2.3.7) of its subclass of finitely-generated substructures.

Theorem 2.5.8 *Let* $\mathfrak{P}$ *be a closed class of algebraic structures with a weakly effective presentation. Then* $\mathfrak{P}$ *is uniformly effectively algebraic if and only if its closure is effectively algebraic.*

Proof. $\Longrightarrow$ Let $\mathfrak{P}$ be a uniform family of effectively algebraic structures, and let g be the uniform bound function on orbit size for elements of structures in $\mathfrak{P}$. Let $\mathcal{R}$ be the closure of $\mathfrak{P}$. Choose e such that program e weakly-effectively enumerates the

structures of $\mathfrak{P}$.

For each pair of elements (x, r) where x is an element of a structure $\mathcal{A}_i$ in $\mathfrak{P}$ and r is an element of $\mathcal{R}$ such that there is an embedding of $\mathcal{A}_i$ into $\mathcal{R}$ with $x \mapsto r$, program e has an "evil-twin" program e' which also weakly-effectively enumerates the structures of $\mathfrak{P}$, and which operates as follows: If the orbit of r in $\mathcal{R}$ contains elements which the current orbit of x in $\mathcal{A}_i$ does not contain, then program e' takes those additional elements in the orbit of r and adds them to the orbit of x in $\mathcal{A}_i$. Program e' still enumerates the structures of $\mathfrak{P}$ and nothing else: The structure which was the i-th structure in the original enumeration will be added by e' later in the enumeration, and the new i-th structure is still finitely generated by generators from $\mathcal{R}$.

Let $b \in \mathcal{R}$. Search through the structures $\mathcal{A}_i$ in the enumeration given by program e until we find one, using the identifying formulas of elements, which embeds into $\mathcal{R}$ in such a way that b is in the range of that embedding. Suppose it is the element z of the structure $\mathcal{A}_i$ which gets mapped to b.

Claim: $g(\langle e', i, z\rangle)$ is a bound on the size of the orbit of b in $\mathcal{R}$.

Proof of Claim: Suppose the orbit of b contains more than $g(\langle e', i, z\rangle)$ elements. Then program e' will put these additional elements into $\mathcal{A}_i$ to force the orbit of z to contain as many elements as does the orbit of b in $\mathcal{R}$, which is greater than $g(\langle e', i, z\rangle)$, contradicting the uniform effective algebraicity of the class $\mathfrak{P}$.

So define $f(b) = g(\langle e', i, z\rangle)$. We do the same sort of thing for the other elements of $\mathcal{R}$. Then f is a computable bound on the orbit size for elements of $\mathcal{R}$.

$\Longleftarrow$ Let $\mathfrak{P}$ be a class of algebraic structures, and let $\mathcal{R}$ be its closure. Suppose $\mathcal{R}$ is effectively algebraic via a computable function f. Choose an arbitrary element z of an arbitrary structure $\mathcal{A}_i$ in $\mathfrak{P}$. Embed $\mathcal{A}_i$ into $\mathcal{R}$, using the identifying formulas of the elements of $\mathcal{A}_i$ and of $\mathcal{R}$, and find the image b of z. Use the computable bound function on orbit size in $\mathcal{R}$ to find a bound on the size of the orbit of b. Set $g(\langle e, i, z\rangle)$ equal to that bound. If we do this for each element of each structure in $\mathfrak{P}$, then g is uniform. ■

Corollary 2.5.9 *Every uniform class* $\mathbf{K}$ *of effectively algebraic structures has the property that singleton spectra are possible for infinite members of* $\mathbf{K}$*, while no class of algebraic structures which isn't uniformly effectively algebraic can have this property.*

Proof. Suppose $\mathcal{K}$ is the closure of a uniform class $\mathbf{K}$ of effectively algebraic structures. Then $\mathcal{K}$, which is effectively algebraic, considered as a substructure of itself, is always computable (its characteristic function is the one which converges to 1 on every input), and thus $\mathrm{DgSp}_{\mathcal{K}}(\mathcal{K}) = \{\mathbf{0}\}$. In the non-effectively algebraic case, we use a coding argument very much like the one in the proof of Theorem 2.3.11 to code an arbitrary infinite structure $\mathcal{A}$ in $\mathbf{K}$ into the closure $\mathcal{K}$ of $\mathbf{K}$ in such a way that $\mathrm{DgSp}_{\mathcal{K}}(\mathcal{A})$ is upwards-closed. ■

Chapter 3

Low$_n$ Boolean Subalgebras

3.1 Introduction

Researchers in computability theory have been coding (or attempting to code) low$_n$ sets into Boolean algebras for at least the past twenty-five years.

Definition 3.1.1 A set $A \subseteq \omega$ is called *low* if $A' \leq_T \mathbf{0}'$. We relativize this to say that A is *low$_n$* if $A^{(n)} \leq_T \mathbf{0}^{(n)}$. Note that for every n, if a set A is low$_n$, then it is low$_{n+1}$.

The question which has been the focus of much of this work is: For which n is it true that every low$_n$ Boolean algebra is isomorphic to a computable Boolean algebra? Recall from Definition 2.2.14 that the *spectrum of a structure* $\mathcal{A}$ is the set $\mathrm{Spec}(\mathcal{A})$ of Turing degrees of structures isomorphic to $\mathcal{A}$: $\mathrm{Spec}(\mathcal{A}) = \{\deg(\mathcal{D}) : \mathcal{D} \cong \mathcal{A}\}$. With this definition, we can rephrase the above question:

Question 3.1.2 *For which n is it true that the spectrum of a low$_n$ Boolean algebra must contain the degree $\mathbf{0}$?*

Question 3.1.2 has been settled for $n = 1$, 2, 3, and 4, all in the affirmative [4] [30] [16], as described in §3.2. That is, if the spectrum of a Boolean algebra contains a low, low_2, low_3, or low_4 degree, it must also contain the degree $\mathbf{0}$.

The goal of this chapter isn't to contribute to the solution of Question 3.1.2, but rather to contribute to the solution of a different (but related) question: If $\mathcal{B}$ is the computable atomless Boolean algebra and $\mathcal{A}$ is a subalgebra which is low_n within $\mathcal{B}$, must there be a computable subalgebra $\mathcal{D}$ such that $(\mathcal{B}, \mathcal{A}) \cong (\mathcal{B}, \mathcal{D})$? Recall from Definition 2.3.10 that the *spectrum of a relation R on a computable structure $\mathcal{M}$* is the set $\text{DgSp}_{\mathcal{M}}(R)$ of Turing degrees of all images of R under isomorphisms from $\mathcal{M}$ onto other computable structures: $\text{DgSp}_{\mathcal{M}}(R) = \{\deg(S) : (\exists \mathcal{B} \leq_T \emptyset)\, [(\mathcal{B}, S) \cong (\mathcal{M}, R)]\}$. With this definition, it is equivalent to ask:

Question 3.1.3 *For which n is it true that the spectrum of a low_n Boolean algebra, viewed as a unary relation on some fixed copy of $\mathcal{B}$, must contain the degree $\mathbf{0}$?*

Question 3.1.3 has recently been answered in the negative for $n = 5$ [22], also described in §3.2, which is a case still unsettled for the former question.

The computable atomless Boolean algebra $\mathcal{B}$ is the *Fraïssé limit* (see Theorem 2.3.7) of the class of finite Boolean algebras. Structures are often considered as unary relations on a Fraïssé limit. For example, the countable dense linear ordering $\mathcal{E}$ is the Fraïssé limit of the class of finitely-generated linear orderings, and in [10], linear orderings are considered as unary relations on a fixed computable copy of $\mathcal{E}$. As the reader may recall from §2.3, the tree $\omega^{<\omega}$ is the Fraïssé limit of the class of finite trees,

and the countable random graph is the Fraïssé limit of the class of finite connected graphs. For other examples of Fraïssé limits, see [3].

A negative answer to Question 3.1.3 for $n = 1$, 2, 3, or 4 would be significant because it would provide us with an example of a spectrum of a subalgebra as a unary relation on the computable atomless Boolean algebra $\mathcal{B}$ which cannot possibly be the spectrum of a Boolean algebra viewed as a structure in its own right – the spectrum of the subalgebra would contain a low_n degree for $n = 1$, 2, 3, or 4 without containing the degree $\mathbf{0}$, and we know that no spectrum of a Boolean algebra as a structure has this property [16].

In this chapter, we adapt the proof of the result that the spectrum of a low_5 subalgebra of $\mathcal{B}$, viewed as a relation on $\mathcal{B}$, need not contain the degree $\mathbf{0}$ to show that the spectrum of a low_4 subalgebra of $\mathcal{B}$, viewed as a relation on $\mathcal{B}$, need not contain the degree $\mathbf{0}$. Thus we provide a negative answer to the $n = 4$ case for the spectrum of a subalgebra. The cases $n \leq 3$ are the subject of current work by the author. We also point out that this implies another spectral difference between Boolean algebras as structures and Boolean subalgebras as relations on $\mathcal{B}$. A theorem of Jockusch and Soare in [13] says that if a Boolean algebra has n-th jump degree $\mathbf{d}$ (defined below in Definition **??**), where $n < \omega$, then $\mathbf{d} = \mathbf{0}^{(n)}$. In other words, no Boolean algebra as a structure can have n-th jump degree larger than $\mathbf{0}^{(n)}$. We show below in Corollary 3.3.6 that a Boolean subalgebra of $\mathcal{B}$, considered as a relation on $\mathcal{B}$, *can* have n-th jump degree strictly larger than $\mathbf{0}^{(n)}$.

3.2 The History

The first result on low_n Boolean algebras and computable copies was the result of Downey and Jockusch for $n = 1$ which appeared in 1994.

Theorem 3.2.1 [4, Thm. 1] *Every low Boolean algebra is isomorphic to a computable one.*

They converted an argument about Boolean algebras to an argument about linear orderings, and then applied a theorem of Remmel [25, Thm. 2.1] which gives sufficient conditions for two linear orderings to have isomorphic interval algebras.

The result for $n = 2$ was the work of Thurber, and it appeared very soon after the $n = 1$ result. Thurber used the same sort of linear ordering argument Downey and Jockusch used.

Theorem 3.2.2 [30, Thm. 1] *Every low_2 Boolean algebra is isomorphic to a computable one.*

Six years after the original result, Knight and Stob came out with the results for $n = 3$ and $n = 4$.

Theorem 3.2.3 [16, Cor. 3.3] *Every low_3 Boolean algebra is isomorphic to a computable one.*

Theorem 3.2.4 [16, Cor. 5.3] *Every low_4 Boolean algebra is isomorphic to a computable one.*

Knight and Stob, unlike Downey and Jockusch and Thurber, argued in terms of Boolean algebras directly.

In an effort to attempt a result for $n = 5$, Miller, in 2011, instead ended up with a result just as interesting about low_5 subalgebras of the computable atomless Boolean algebra $\mathcal{B}$ when viewed as a relation on $\mathcal{B}$.

Theorem 3.2.5 [22, Thm. 2.4] *Let* $\mathbf{c}$ *be any Turing degree which is not* low_4*. Then there exists a Boolean subalgebra* $\mathcal{A}$ *of the computable atomless Boolean algebra* $\mathcal{B}$ *for which* $DgSp_{\mathcal{B}}(\mathcal{A})$ *contains* $\mathbf{c}$ *but does not contain* $\mathbf{0}$*. In particular, there are* low_5 *degrees for which this works.*

Proof. We present a sketch of this proof.

Let $\mathcal{B}$ be a Boolean algebra and let $\mathcal{A}$ be a subalgebra of $\mathcal{B}$. An element $x \in \mathcal{B}$ is called an $\mathcal{A}$-*supremum* if x is the least upper bound in $\mathcal{B}$ of an infinite set of $\mathcal{A}$-atoms. If such an x is not the union of two disjoint $\mathcal{A}$-suprema, then x is called a *single* $\mathcal{A}$-*supremum* (also called a *1-atom of* $\mathcal{A}$ in the literature), and a *k-fold* $\mathcal{A}$-*supremum* if x is the union of k disjoint single $\mathcal{A}$-suprema. The property of being a k-fold $\mathcal{A}$-supremum is $\Sigma_4^{\mathcal{A}}$ uniformly in k.

Given an arbitrary nonlow_4 degree $\mathbf{c}$ and an arbitrary $C \in \mathbf{c}$, Miller uses a C-oracle to construct a subalgebra $\mathcal{A}$ of $\mathcal{B}$ so that $\mathbf{c} \in \text{DgSp}_{\mathcal{B}}(\mathcal{A})$. He builds $\mathcal{A}$ to satisfy the following:

$$n \in C^{(4)} \iff \exists x \in \mathcal{A} \, [\, x \text{ is a } 2^n\text{-fold } \mathcal{A}\text{-supremum }].$$

(For the detailed constructions of the subalgebras $\mathcal{A}$ and $\mathcal{D}$, see Miller's proof in [22].)

The statement

$$\exists x \in \mathcal{A} \; [\; x \text{ is a } 2^n\text{-fold } \mathcal{A}\text{-supremum }]$$

is Σ^0_4 in $\mathcal{A}$, and hence computable in $\mathcal{A}^{(4)}$. And because this statement is equivalent to n being in $C^{(4)}$, we have $C^{(4)} \leq_T \mathcal{A}^{(4)}$.

Now, suppose $\mathbf{0} \in \mathrm{DgSp}_{\mathcal{B}}(\mathcal{A})$. In other words, suppose there is a computable subalgebra $\mathcal{D}$ of $\mathcal{B}$ such that $(\mathcal{B}, \mathcal{D}) \cong (\mathcal{B}, \mathcal{A})$. Then the uniform $\Sigma^{\mathcal{A}}_4$ definition of k-fold $\mathcal{A}$-suprema would convert to a uniform $\Sigma^{\mathcal{D}}_4$ definition of k-fold $\mathcal{D}$-suprema on $(\mathcal{B}, \mathcal{D})$, which of course means just a Σ^0_4 definition, since $\mathcal{D}$ is computable. Thus it is Σ^0_4 whether $\mathcal{D}$ contains a 2^n-fold $\mathcal{D}$-supremum, and this in turn implies that $C^{(4)} \leq_T \emptyset^{(4)}$. This is a contradiction because we chose C to be nonlow_4. ■

3.3 The Main Result

Theorem 3.3.1 *Let* $\mathbf{c}$ *be any Turing degree which is not* low_3*. Then there exists a Boolean subalgebra* $\mathcal{A}$ *of the computable atomless Boolean algebra* $\mathcal{B}$ *for which* $DgSp_{\mathcal{B}}(\mathcal{A})$ *contains* $\mathbf{c}$ *but does not contain* $\mathbf{0}$*.*

Since there exists a degree which is low_4 but not low_3, we have an immediate corollary:

Corollary 3.3.2 *There is a* low_4 *degree* $\mathbf{c}$ *and a Boolean subalgebra* $\mathcal{A}$ *of* $\mathcal{B}$ *for which* $DgSp_{\mathcal{B}}(\mathcal{A})$ *contains* $\mathbf{c}$ *but does not contain* $\mathbf{0}$*.*

Proof. (of Theorem 3.3.1) The construction of a $\mathcal{D}$ for which $(\mathcal{B}, \mathcal{D}) \cong (\mathcal{B}, \mathcal{A})$ and

$\deg(\mathcal{D}) = \mathbf{c}$ follows exactly as in the proof of Theorem 3.2.5 and [22], even though we are dealing with a nonlow_3 degree here, rather than a nonlow_4 degree. This is still a $C^{(4)}$-construction.

It is actually a small (yet powerful) fact from a classic result known as the "Jump Theorem" which allows us to use a low_3 argument with a low_4 construction; we will use this fact to show that $\text{DgSp}_{\mathcal{B}}(\mathcal{A})$ does not contain the degree $\mathbf{0}$.

Fact 3.3.3 [27, "Jump Theorem" 3.4.3(v)] *Let $A \subseteq \omega$ and $B \subseteq \omega$. Then $A \leq_T B$ if and only if $A' \leq_1 B'$.*

The reason this works is that the Turing reduction $C^{(4)} \leq_T \mathcal{A}^{(4)}$ is actually much stronger than just a Turing reduction. Recall that according to Miller's construction in [22], we have

$$n \in C^{(4)} \iff \exists x \in \mathcal{A} \, [\, x \text{ is a } 2^n\text{-fold } \mathcal{A}\text{-supremum }], \tag{3.1}$$

where the property of being a 2^n-fold $\mathcal{A}$-supremum is a $\Sigma_4^{\mathcal{A}}$ property of x, uniformly in n. It turns out that we actually have a 1-reduction from $C^{(4)}$ to $A^{(4)}$: Let f be a total computable function with the property that for all n,

$$f(n) \in \mathcal{A}^{(4)} \iff \mathcal{A} \text{ contains a } 2^n\text{-fold } \mathcal{A}\text{-supremum.} \tag{3.2}$$

Then by equivalences (3.1) and (3.2), $n \in C^{(4)}$ iff $f(n) \in \mathcal{A}^{(4)}$. Thus $C^{(4)} \leq_1 \mathcal{A}^{(4)}$ via this f.

For any $\mathcal{D}$ for which $(\mathcal{B}, \mathcal{D}) \cong (\mathcal{B}, \mathcal{A})$, if $\mathcal{A}$ has a k-fold $\mathcal{A}$-supremum, then $\mathcal{D}$ also has a k-fold $\mathcal{D}$-supremum: limits must map to limits under an isomorphism (same

for limits of limits, etc.), and so k-fold suprema are isomorphism-invariant. Thus $n \in C^{(4)}$ iff $f(n) \in \mathcal{D}^{(4)}$. So $C^{(4)} \leq_1 \mathcal{D}^{(4)}$.

If there were a computable such $\mathcal{D}$, then we would have $C^{(4)} \leq_1 \emptyset^{(4)}$, and then by Fact 3.3.3, we'd have $C^{(3)} \leq_T \emptyset^{(3)}$, which is a contradiction since we chose C to be nonlow_3. ■

Corollary 3.3.4 *The spectrum of a Boolean subalgebra of $\mathcal{B}$ can fail to be the spectrum of any Boolean algebra (as a structure).*

Proof. The spectrum of the Boolean subalgebra of $\mathcal{B}$ which we constructed in Theorem 3.3.1 contains a low_4 degree but not the degree $\mathbf{0}$. The spectrum of a Boolean algebra (as a structure in its own right) cannot contain a low_4 degree without containing the degree $\mathbf{0}$, as we stated in Theorem 3.2.4. ■

Theorem 3.3.5 *If $\mathcal{A}$ is a Boolean subalgebra of the computable atomless Boolean algebra $\mathcal{B}$ and $\mathcal{A}$ is not intrinsically computable, then $DgSp_{\mathcal{B}}(\mathcal{A})$ is closed upwards in the Turing degrees.*

Proof. In [3], the authors proved that if $\mathcal{F}$ is the Fraïssé limit of a class $\mathbf{K}$ of finite structures over a finite language $\mathcal{L}$ where $\text{Th}_{\mathcal{L}}(\mathbf{K})$ is computably axiomatizable and locally finite (every finitely-generated structure is actually finite), then if R is a unary relation on $\mathcal{F}$ which is not intrinsically computable, $\text{DgSp}_{\mathcal{F}}(R)$ is upward closed under Turing reducibility [3, Cor. 5.2]. These hypotheses all hold for the atomless Boolean algebra $\mathcal{B}$ – Boolean algebras are locally finite because the Boolean algebra generated by n elements has 2^n elements – and so if $\mathcal{A}$ is the unary relation on $\mathcal{B}$, we have

exactly the result we want. ■

To give the reader an idea of what the proof involves in the case of Boolean algebras, we present an example. We begin with the four-element Boolean algebra, and we describe the coding process.

Let R be a unary relation on $\mathcal{B}$. Fix any degrees $\mathbf{d} < \mathbf{c}$, and suppose that $S \in \mathbf{d}$ and $(\mathcal{B}, R) \cong (\mathcal{B}, S)$. Let $\tilde{\mathcal{B}}$ be another computable copy of $\mathcal{B}$, and fix a set $C \in \mathbf{c}$ to be our oracle. We will build a C-computable isomorphism g from $\mathcal{B}$ onto $\tilde{\mathcal{B}}$ such that $g(S) \equiv_T C$. This will prove the upwards closure of $\mathrm{DgSp}_{\mathcal{B}}(R)$.

At each stage $s+1$ of the construction, the partial isomorphism g_s will be extended to g_{s+1} by extending the domain D_s to D_{s+1}, extending the range R_s to R_{s+1}, and defining g_{s+1} on the finitely many new elements in $D_{s+1} - D_s$. The R_{s+1} will be computable, uniformly in s, without any C-oracle, whereas computing D_{s+1} and g_{s+1} will require a C-oracle.

Suppose, in our example, that at stage $s+1$ with s even, $D_s \cong R_s \cong$ the four-element Boolean algebra. The atoms of R_s will be called y_1 and y_2.

We partition $\tilde{\mathcal{B}}$ into equivalence classes where two elements of $\tilde{\mathcal{B}}$ are equivalent if they have the same quantifier-free type over $\{y_1, y_2\}$. There are $3^2 = 9$ equivalence classes:

- $Z_1 = \left\{x \in \tilde{\mathcal{B}} : x = y_1 \cup y_2\right\}$
- $Z_2 = \left\{x \in \tilde{\mathcal{B}} : x = \emptyset\right\}$

- $Z_3 = \left\{x \in \tilde{\mathcal{B}} : x = y_1\right\}$
- $Z_4 = \left\{x \in \tilde{\mathcal{B}} : x = y_2\right\}$
- $Z_5 = \left\{x \in \tilde{\mathcal{B}} : x \supseteq y_1 \ \ \& \ \ \emptyset \subsetneq x \cap y_2 \subsetneq y_2\right\}$
- $Z_6 = \left\{x \in \tilde{\mathcal{B}} : x \supseteq y_2 \ \ \& \ \ \emptyset \subsetneq x \cap y_1 \subsetneq y_1\right\}$
- $Z_7 = \left\{x \in \tilde{\mathcal{B}} : \emptyset \subsetneq x \cap y_1 \subsetneq y_1 \ \ \& \ \ \emptyset \subsetneq x \cap y_2 \subsetneq y_2\right\}$
- $Z_8 = \left\{x \in \tilde{\mathcal{B}} : \emptyset \subsetneq x \cap y_1 \subsetneq y_1 \ \ \& \ \ x \cap y_2 = \emptyset\right\}$
- $Z_9 = \left\{x \in \tilde{\mathcal{B}} : x \cap y_1 = \emptyset \ \ \& \ \ \emptyset \subsetneq x \cap y_2 \subsetneq y_2\right\}$

Notice that the classes Z_1, Z_2, Z_3, and Z_4 in our example each have size 1.

For each i, $1 \leq i \leq 9$, let $x_{i,1}$ be the least element of Z_i which is not in R_s, and choose one element of Z_i to represent each possible type of an element of Z_i over $R_s \cup \{x_{i,1}\}$. Depending on the i, there could be as many as 63 of these types, so instead of listing out all the possible types for each i, we pick a nontrivial example and list the 7 types of an element of Z_6 over $R_s \cup \{x_{6,1}\}$:

- $x_{6,1}$ already represents the set $\{y \in Z_6 : y = x_{6,1}\}$.
- $x_{6,2}$ will represent the set $\{y \in Z_6 : y \subsetneq x_{6,1}\}$.
- $x_{6,3}$ will represent the set $\{y \in Z_6 : x_{6,1} \subsetneq y \subsetneq \mathbb{Q}\}$.
- $x_{6,4}$ will represent the set

  $\{y \in Z_6 : y_2 \subsetneq y \cap x_{6,1} \subsetneq x_{6,1} \ \ \& \ \ \emptyset \subsetneq y \cap (\mathbb{Q} - \{x_{6,1}\}) \subsetneq (\mathbb{Q} - \{x_{6,1}\})\}$.

- $x_{6,5}$ will represent the set

$\{y \in Z_6 : y \cap x_{6,1} = y_2 \ \ \& \ \ \emptyset \subsetneq y \cap (\mathbb{Q} - \{x_{6,1}\}) \subsetneq (\mathbb{Q} - \{x_{6,1}\})\}$.

- $x_{6,6}$ will represent the set $\{y \in Z_6 : y \supseteq (\mathbb{Q} - \{x_{6,1}\}) \ \ \& \ \ y \cap x_{6,1} \supsetneq y_2)\}$.
- $x_{6,7}$ will represent the set $\{y \in Z_6 : y \cap x_{6,1} = y_2 \ \ \& \ \ y \supseteq (\mathbb{Q} - \{x_{6,1}\})\}$.

For each i, $1 \leq i \leq 9$, let $A_i = g_s^{-1}(Z_i)$.

We claim that for some i, $1 \leq i \leq 9$, we have both $A_i \cap S \neq \emptyset$ and $A_i \cap \overline{S} \neq \emptyset$. If there were no such i, then for each A_i, either $A_i \subseteq \overline{S}$ or $A_i \subseteq S$, and in that case, S could be expressed as the union of the A_i's for which $A_i \subseteq S$. But each of these A_i's is definable without quantifiers over the finite set D_s, making S definable without quantifiers over D_s. S was not definable this way by hypothesis, as we chose the degree of S arbitrarily.

So we find an i for which $A_i \cap S \neq \emptyset$ and $A_i \cap \overline{S} \neq \emptyset$. For each $j < i$, we look for (type-wise) appropriate preimages in A_j of the $x_{j,1}, \ldots, x_{j,k_j}$, which are either all in S or all in $\overline{S}$. If we find such elements for each $j < i$, we stick with the i that we first found for which $A_i \cap S \neq \emptyset$ and $A_i \cap \overline{S} \neq \emptyset$. If there is a $j < i$ for which we don't find appropriate preimages which are all inside or all outside S, then let the least such j be the new i.

Suppose, in our example, that $i = 6$. Let $b_6 \in A_6 \cap S$ and let $c_6 \in A_6 \cap \overline{S}$. If $e \in C$, we define $g(b_6) = x_{6,1}$, and we choose the $x_{6,m}$ $(2 \leq m \leq 7)$ which has the same quantifier-free type over $x_{6,1}$ as c_6 has over b_6; we define $g(c_6) = x_{6,m}$ for this m. If $e \notin C$, we define $g(c_6) = x_{6,1}$, and we define $g(b_6) = x_{6,n}$ for the appropriate n. In

either case, we choose appropriate preimages for the remaining five $x_{6,i}$ and all the $x_{j,i}$ for $j \neq 6$.

At the next stage (an even stage), we take care to ensure that the isomorphism g ends up being total. We do this by making sure to define g on the least element of the origin algebra $\mathcal{B}$. Since no coding happens at the even stages, we omit the details.

Verification. We built g using knowledge of S and C, and since $S \leq_T C$ by design, we have $g \leq_T C$. The finite ranges R_s are uniformly computable in s (without a C-oracle). At odd stages we ensure that g is onto (by always choosing a preimage for the least element of $\tilde{\mathcal{B}}$ not yet in the range), and at even stages we ensure that g is total. At each stage s, g_s is a partial isomorphism, so $g : \mathcal{B} \to \tilde{\mathcal{B}}$ is an isomorphism.

It remains to show that $C \leq_T g(S)$. So let $e \in \omega$. We will determine whether e is in C or not using only a $g(S)$-oracle. We need to know what went into the range of g at stage $2e+1$, so we compute $R_{2e+1} - R_{2e} = \{x_{i,j} : 1 \leq i \leq n,\ 1 \leq j \leq k_i\}$. We can do this because the range does not depend on C at all. Next, we find the least r such that there exists m with $[\,[x_{r,1} \in g(S)\ \&\ x_{r,m} \notin g(S)] \vee [x_{r,1} \notin g(S)\ \&\ x_{r,m} \in g(S)]\,]$. Such an r exists because this is how we coded C into g. (For each $j < r$, the $x_{j,i}$ will either all be in $g(S)$ or all be outside $g(S)$, because their preimages will either all be in S or all be outside of S, by construction.) Then $e \in C$ if and only if $x_{r,1} \in g(S)$. ■

When Miller wrote up his proof of Theorem 3.2.5, he neglected to describe the spectrum of the Boolean algebra he built. So we describe it here.

Corollary 3.3.6 *The spectrum of the Boolean subalgebra constructed in [22] and used here in Theorem 3.3.1 is the set of all Turing degrees* $\mathbf{d}$ *such that*

$$\mathbf{c}''' \leq \mathbf{d}'''$$

where $\mathbf{c}$ *is the nonlow*$_4$ *or nonlow*$_3$ *degree in the theorem.*

Proof. Let $\mathcal{D}$ be a copy of the $\mathcal{A}$ we built, where $\deg(\mathcal{D}) = \mathbf{d}$. We know from the construction of $\mathcal{A}$ that $n \in C^{(4)} \iff \exists x \in \mathcal{A}$ [x is a 2^n-fold $\mathcal{A}$-supremum]. Since $\mathcal{D} \cong \mathcal{A}$ and suprema are isomorphism-invariant, we have

$$n \in C^{(4)} \iff \exists x \in \mathcal{D} \text{ [} x \text{ is a } 2^n\text{-fold } \mathcal{D}\text{-supremum]}.$$

In the proof of Theorem 3.3.1 we defined f for which $f(n) \in \mathcal{A}^{(4)}$ means $\mathcal{A}$ contains a 2^n-fold $\mathcal{A}$-supremum. So $C^{(4)} \leq_1 \mathcal{D}^{(4)}$ via this f, and by Fact 3.3.3, we have $C'''' \leq_T \mathcal{D}'''$.

Now let D be a set of degree $\mathbf{d}$, and suppose $C'''' \leq_T D'''$. By Fact 3.3.3, we have $C^{(4)} \leq_1 D^{(4)}$. Just as in Miller's construction in [22], we use a D-oracle to build a Boolean subalgebra $\mathcal{R}$ of $\mathcal{B}$ with $(\mathcal{B}, \mathcal{A}) \cong (\mathcal{B}, \mathcal{R})$. This gives us $\mathcal{R} \leq_T D$, and now that we have Theorem 3.3.5 to give us the upward closure of $\mathrm{DgSp}_{\mathcal{B}}(\mathcal{A})$, we can say that $\mathbf{d} \in \mathrm{DgSp}_{\mathcal{B}}(\mathcal{A})$. ∎

Last but not least, we recall a well-known theorem of Jockusch and Soare about Boolean algebras as structures which we now know does not hold for Boolean subalgebras as relations on $\mathcal{B}$.

Definition 3.3.7 For $n \in \omega$, the *n-th jump degree of a structure* $\mathcal{M}$ is the least

degree among the degrees of n-th jumps of structures isomorphic to $\mathcal{M}$, if such a least degree exists.

Theorem 3.3.8 (Jockusch & Soare) [13, Thm. 1.3] *Let $n < \omega$. If a Boolean algebra has nth-jump degree $\mathbf{d}$, then $\mathbf{d} = \mathbf{0}^{(n)}$.*

This theorem, along with Corollary 3.3.6, differentiates the jump degree spectrum of a Boolean algebra and the jump degree spectrum of a Boolean subalgebra of $\mathcal{B}$. Theorem 3.3.8 states that the n-th jump degree of a Boolean algebra (as a structure) can't be any larger than $\mathbf{0}^{(n)}$, while Corollary 3.3.6, via $n = 3$, shows that the n-th jump degree of a Boolean subalgebra of $\mathcal{B}$, considered as a relation on $\mathcal{B}$, *can* be larger than $\mathbf{0}^{(n)}$.

Notice how this also distinguishes Boolean subalgebras of $\mathcal{B}$ from substructures of the Fraïssé limits we met in §2.3. Recall from Theorem 2.3.14 (resp. Theorem 2.3.13) that spectra of finite-valence connected subgraphs (resp. finite-branching subtrees) of the random graph (resp. the tree $\omega^{<\omega}$) are exactly the spectra of finite-valence connected graphs (resp. finite-branching trees) as structures; we now know that spectra of subalgebras of $\mathcal{B}$ are not the same as spectra of Boolean algebras as structures.

Bibliography

[1] W. Calvert; The isomorphism problem for computable abelian p-groups of bounded length, *Journal of Symbolic Logic*, **70** (2005), 331-345.

[2] B. F. Csima, J. N. Y. Franklin, & R. A. Shore; Degrees of categoricity and the hyperarithmetic hierarchy, *in preparation*.

[3] B. F. Csima, V. S. Harizanov, R. G. Miller, & A. Montalbán; Computability of Fraïssé Limits, *Journal of Symbolic Logic* **76** (2011), 66-93.

[4] R. Downey & C. G. Jockusch; Every low Boolean algebra is isomorphic to a recursive one, *Proceedings of the American Mathematical Society* **122** (1994), 871-880.

[5] D. S. Dummit & R. M. Foote; *Abstract Algebra*, third edition, (Hoboken, New Jersey: John Wiley & Sons Inc., 2004).

[6] E. B. Fokina, I. Kalimullin, & R. G. Miller; Degrees of categoricity of computable structures, *Archive for Mathematical Logic*, **49** (2010), 51-67.

[7] M. D. Fried & M. Jarden, *Field Arithmetic* (Berlin: Springer-Verlag, 1986).

[8] A. Fröhlich & J. C. Shepherdson; Effective procedures in field theory, *Phil. Trans. Royal Soc. London, Series A* **248** (1956) 950, 407-432.

[9] A. Frolov, I. Kalimullin, & R. G. Miller; Spectra of algebraic fields and subfields, *Mathematical Theory and Computational Practice – Fifth Conference on Computability in Europe, Lecture Notes in Computer Science 5635* (Berlin: Springer-Verlag, 2009), 232-241.

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

[11] D. R. Hirschfeldt, K. Kramer, R. G. Miller, & A. Shlapentokh; Relative computable categoricity and computable dimension for algebraic fields, *in preparation*.

[12] W. Hodges; *A shorter model theory*, (Cambridge Univ. Press, 1997).

[13] C. G. Jockusch & R. I. Soare; Boolean algebras, stone spaces, and the iterated Turing jump, *Journal of Symbolic Logic* **59** (1994), 1121-1138.

[14] C. G. Jockusch & R. I. Soare; Π^0_1-classes and degrees of theories, *Transactions of the American Mathematical Society* **173** (1972), 33-56.

[15] J. F. Knight; Degrees coded in jumps of orderings, *Journal of Symbolic Logic* **51** (1986), 1034-1042.

[16] J. F. Knight & M. Stob; Computable Boolean algebras, *Journal of Symbolic Logic* **65** (2000), 1605-1623.

[17] L. Kronecker; Grundzüge einer arithmetischen Theorie der algebraischen Größen, *J. f. Math.* **92** (1882), 1-122.

[18] R. G. Miller; Computability, definability, categoricity, and automorphisms, *Ph.D. dissertation*.

[19] R. G. Miller; Computable fields and Galois theory, *Notices of the American Mathematical Society* **55** (2008), 798-807.

[20] R. G. Miller; **d**-Computable categoricity for algebraic fields, *Journal of Symbolic Logic* **74** (2009), 1325-1351.

[21] R. G. Miller; Is it harder to factor a polynomial or to find a root?, *Transactions of the American Mathematical Society* **362** (2010), 5261-5281.

[22] R. G. Miller; Low_5 Boolean subalgebras and computable copies, *Journal of Symbolic Logic* **76** (2011), 1061-1074.

[23] R. G. Miller & A. Shlapentokh; Computable categoricity for algebraic fields with splitting algorithms, *in preparation*.

[24] M. Rabin; Computable algebra, general theory, and theory of computable fields, *Transactions of the American Mathematical Society* **95** (1960), 341-360.

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

[26] P. Rothmaler; *Introduction to Model Theory. Algebra, Logic and Applications, Vol. 15* (Gordon and Breach, 2000).

[27] R. I. Soare; *Recursively Enumerable Sets and Degrees* (Berlin: Springer-Verlag, 1987).

[28] R. I. Soare; *Computability Theory and Applications: The Art of Classical Computability,* two volumes (Heidelberg: Springer-Verlag, to appear).

[29] R. M. Steiner; Computable fields and weak truth-table reducibility, *Programs, Proofs, Processes - Sixth Conference on Computability in Europe,* eds. F. Ferreira, B. Lowe, E. Mayordomo, & L.M. Gomes, *Lecture Notes in Computer Science 6158* (Berlin: Springer-Verlag, 2010), 394-405.

[30] J. J. Thurber; Every low_2 Boolean algebra has a recursive copy, *Proceedings of the American Mathematical Society* **123** (1995), 3859-3866.

[31] B. L. van der Waerden; *Algebra*, volume I, trans. F. Blum & J.R. Schulenberger (New York: Springer-Verlag, 1970 hardcover, 2003 softcover).

[32] L. C. Washington; *Introduction to Cyclotomic Fields* (New York: Springer-Verlag, 1982).