COMPUTABLE CATEGORICITY FOR ALGEBRAIC FIELDS WITH SPLITTING ALGORITHMS

RUSSELL MILLER AND ALEXANDRA SHLAPENTOKH

ABSTRACT. A computably presented algebraic field F has a *splitting algorithm* if it is decidable which polynomials in $F[X]$ are irreducible there. We prove that such a field is computably categorical iff it is decidable which pairs of elements of F belong to the same orbit under automorphisms. We also show that this criterion is equivalent to the relative computable categoricity of F.

1. INTRODUCTION

Computable categoricity is a basic concept in computable model theory. It holds of those computable structures $\mathfrak{A}$ for which the classical concept of being isomorphic to $\mathfrak{A}$ is equivalent to its analogue in the context of computable structures and computable isomorphisms. Thus it fits squarely into the program of computable model theory, which studies how difficult it is to give effective versions of classical model-theoretic concepts and constructions. (A summary of relevant definitions and standard results appears in Sections 2 and 3.)

Assorted versions of this concept have been proposed. The strongest, *computable stability*, holds of a computable structure $\mathfrak{A}$ iff every classical isomorphism from $\mathfrak{A}$ onto any other computable structure is itself computable. This holds, for instance, for computable presentations of the structure $(\mathbb{Z}, S)$, the integers under the successor function. The most common version, *computable categoricity*, which will be the focus of this paper, is not as stringent: a structure $\mathfrak{A}$ is computably categorical iff, whenever there exists a classical isomorphism f from $\mathfrak{A}$ onto a computable structure $\mathfrak{B}$, there also exists a computable isomorphism g from $\mathfrak{A}$ onto $\mathfrak{B}$, although f itself may fail to be computable. This version has been generalized to *relative computable categoricity*, which requires that for every structure $\mathfrak{B}$ which has domain ω and is isomorphic to the computable structure $\mathfrak{A}$, there must exist an isomorphism from $\mathfrak{A}$ onto $\mathfrak{B}$ which is computable in the Turing degree of the (not necessarily computable) structure $\mathfrak{B}$. Finally, there are weaker versions such as Δ^0_{n+1}-*categoricity*, and relativizations of these: $\mathfrak{A}$ is Δ^0_{n+1}-categorical iff every computable structure $\mathfrak{B}$ isomorphic to $\mathfrak{A}$ has an isomorphism onto $\mathfrak{A}$ which is $\mathbf{0}^{(n)}$-computable. Such

versions essentially study how much information about the structures is needed to compute isomorphisms.

Among these, computable categoricity remains the most widely studied concept. It is often equivalent to relative computable categoricity, but exceptions are known to exist; see [15] for an exception, and [9] for conditions implying equivalence. Traditionally, the main question has been to determine, for a particular class of structures, some structural criterion which is equivalent to computable categoricity. In early examples, from around 1980, Dzgoev, Goncharov, and Remmel showed (independently; see [10, 24]) that a linear order is computably categorical iff it has only finitely many pairs of adjacent elements, and Remmel also showed in [25] that a Boolean algebra is computably categorical iff it has only finitely many atoms. In both cases, the structural criterion identifies the obstacle to computing isomorphisms, in the class of structures under consideration. On the other hand, the criteria equivalent to computable categoricity for trees (viewed either as partial orders, or under the meet relation, and also with distinguished subtrees), established in [16] and [18] by Lempp, McCoy, Miller, and Solomon and in [13] by Kogabaev, Kudinov, and Miller, are not easy to describe in any known way, even though they are "structural," in any reasonable sense of the word. In terms of computational complexity, they are Σ_3^0, as defined in Section 2, just like the conditions for linear orders and Boolean algebras. Indeed, for all of these structures, computable categoricity is equivalent to relative computable categoricity, and it is known from work by Ash, Knight, Manasse, and Slaman in [1], and independently by Chisholm in [2], that the computational complexity of relative computable categoricity is always Σ_3^0.

Our intention in this paper is to give a criterion for computable categoricity for algebraic fields with splitting algorithms. This should be viewed as a first step towards a criterion for computable fields in general, for which the question of computable categoricity has long been studied and has proven highly intractable. The basic definitions regarding computable fields appear in Section 3. Apart from algebraically closed fields, we believe that ours is the first result to characterize computable categoricity for any natural class of fields. (An algebraically closed field is computably categorical iff it has finite transcendence degree over its prime subfield. The forwards direction was established by Ershov in [4]; while the converse follows from the very first known consideration of computable categoricity in the literature, which forms part of the paper [7] by Frohlich and Shepherdson.) It is known that finite transcendence degree does not characterize computable categoricity for fields in general: in [4] Ershov gave a field which is algebraic over its prime subfield, yet is not computably categorical, and more recently Miller and Schoutens disproved the converse in [22], by building a computably categorical field of infinite transcendence degree over $\mathbb{Q}$.

A computable field F is algebraic with a splitting algorithm if F is an algebraic extension of its own prime subfield ($\mathbb{Q}$ or $\mathbb{F}_p$) such that the set of reducible polynomials in $F[X]$ is decidable. Our criterion for computable categoricity, which holds on all such computable fields F, is that the orbit relation on the field must be decidable. In a certain sense, this criterion is not quite as structural as previous ones for other classes of structures; earlier criteria often used attributes, such as finiteness, which are not expressible in first-order model theory, but we are not aware of other classes for which it is necessary to use the notion of decidability

to characterize computable categoricity. Of course, we do not prove here that it is actually necessary to do so for algebraic fields with splitting algorithms either, but the simplicity of the statement of our result, combined with the lack of any other criteria after much study by many researchers, leads us to believe empirically that ours is as "good" a criterion as one is likely to find. In complexity terms, the statement of our criterion is Σ_3^0, just like those for many other classes, and it is equivalent to relative computable categoricity, which would force it to be Σ_3^0 in any case. Indeed, computable categoricity is readily seen to be Σ_3^0-hard for such fields, and so our criterion has the lowest possible computational complexity.

Recent work by Miller also studied categoricity for computable algebraic fields, but using the weaker notion of $\boldsymbol{d}$-computable categoricity. A computable structure $\mathfrak{A}$ is $\boldsymbol{d}$-*computably categorical* if every computable structure isomorphic to it is isomorphic via some $\boldsymbol{d}$-computable isomorphism. This says that the degree $\boldsymbol{d}$ contains sufficiently much information to compute an isomorphism whenever one exists. Normally, for structures which are not computably categorical, one finds that they are $\boldsymbol{0}'$-computably categorical, or $\boldsymbol{0}^{(m)}$-computably categorical for some m, or possibly $\boldsymbol{0}^{(\alpha)}$-computably categorical for some ordinal α, and it is common that, for the least such m, $\boldsymbol{0}^{(m)}$ gives a sharp lower bound for such degrees $\boldsymbol{d}$, especially if we quantify over all structures in a class. The surprising result in [20] was that, although not all computable algebraic fields are $\boldsymbol{0}'$-categorical, there is a degree $\boldsymbol{d}$ with $\boldsymbol{d}' = \boldsymbol{0}''$ such that all such fields are $\boldsymbol{d}$-computably categorical. The unusual degree involved here suggests that the question of computable categoricity for fields, even just for algebraic fields, is somehow not as straightforward as for many other structures.

On the other hand, the current work [11], by Hirschfeldt, Kramer, and the present authors, shows that relative computable categoricity does have a fairly reasonable structural characterization for computable algebraic fields. Their criterion does involve computability, just as does ours in this paper, but it can be expressed in a generally understandable way. However, they also show that not all computably categorical algebraic fields are relatively computably categorical, and that the criterion from this paper, for algebraic fields with splitting algorithms, fails to extend (at least in the obvious ways) to computable algebraic fields without splitting algorithms. Rather than producing a structural criterion, they show that the definition of computable categoricity for algebraic fields is Π_4^0, and that for such fields, the property of being computably categorical is Π_4^0-complete. Thus they establish that no structural criterion can express computable categoricity in any simpler way than the basic definition. Of course, it is still possible that some structural criterion of complexity Π_4^0 (or higher) might characterize computable categoricity in a more transparent way than the definition itself does; but in terms of computational complexity, one cannot simplify the definition at all.

The next two sections of the paper contains definitions and background on computability theory and on computable fields, along with a number of classical results about fields which will be useful later on. This much is sufficient for us to introduce the problem, in Section 4, with some basic cases of algebraic fields and computable categoricity. Sections 5 and 6 describe further concepts necessary for the main theorem: the isomorphism tree, and the orbit relation. Finally, in Section 7, we prove the full result for computable algebraic fields with splitting algorithms.

2. Background in Computability

We recall here the concepts from computability theory which will be essential to our work on fields. Computable functions are defined in [28], and indeed, several very different definitions give rise to the same class of functions. Functions on the set ω of nonnegative integers are usually identified with their graphs in ω^2, and we then code ω^2 into ω, so that the graph corresponds to a subset of ω; conversely, for our purposes, a subset of ω may be identified with its characteristic function. The partial computable functions (those for which the computation procedure halts on certain inputs from ω, but not necessarily on all of them) can be enumerated effectively, and are usually denoted as $\varphi_0, \varphi_1, \ldots$, with the index e coding the program for computing $\varphi_e(x)$ on $x \in \omega$. The domains of these functions constitute the *computably enumerable sets*, and we write W_e for the domain of φ_e. These are precisely the sets which are definable by Σ_1^0 formulas, i.e. sets of the form

$$\{x \in \omega : \exists y_1 \cdots \exists y_m \ R(x, y_1, \ldots, y_m)\},$$

where $m \in \omega$ is arbitrary and R may be any computable subset of ω^{m+1}. We usually write "$\varphi_e(x)\!\downarrow = y$" to indicate that the computation of φ_e on input x halts and outputs y, and so $\varphi_e(x)\!\downarrow$ iff $x \in W_e$; otherwise we write $\varphi_e(x)\!\uparrow$. Also, if the computation halts within s steps, we write $\varphi_{e,s}(x)\!\downarrow$. The set $W_{e,s}$ is the domain of $\varphi_{e,s}$, so $W_e = \cup_s W_{e,s}$. Every set $W_{e,s}$ is computable (although the union W_e may not be), and we take it as a convention of our computations that only numbers $\leq s$ lie in $W_{e,s}$.

More generally, we define the Σ_n^0 formulas by induction on n. The Σ_0^0 formulas are those formulas with free variables $x_1, \ldots, x_m$ which define computable subsets of ω^m. A Π_n^0 formula is the negation of a Σ_n^0 formula (so a Π_1^0 formula is universal, in the same sense that a Σ_1^0 formula is existential), and a Σ_{n+1}^0 formula in the variable x of the form

$$\exists y_1 \cdots \exists y_m \ R(x, y_1, \ldots, y_m),$$

where R is a Π_n^0 formula. Thus the subscript counts the number of quantifier alternations. (The superscript, often omitted here, refers to the fact that we quantify only over natural numbers, not over sets of naturals, or sets of sets of them, etc.) Consecutive like quantifiers can be collapsed to a single quantifier, by the use of computable pairing functions: for each n, there is a computable bijection $\omega^n \to \omega$, denoted by letting $\langle y_1, \ldots, y_n \rangle \in \omega$ be the image of the tuple $(y_1 \ldots, y_n) \in \omega^n$. Thus the Σ_{n+1}^0 formula above could be expressed as $\exists y R'(x, y)$, where $R'(x, \langle y_1, \ldots, y_n \rangle)$ holds iff $R(x, y_1, \ldots, y_n)$ holds. Details can be found in [19], where a more general computable bijection onto ω from the set $\omega^{<\omega}$ of *all* finite tuples of natural numbers is also given. This latter bijection allows us to use a single quantifier to quantify over all polynomials in $F[X]$, for instance, for any computable field F (as defined below).

Turing reducibility and 1-reducibility are ways of comparing the complexity of subsets $A, B \subseteq \omega$, both defined in [28]. We write $A \leq_T B$ and $A \leq_1 B$, respectively, to denote that A is no more complex than B under these relations. It is well known that, for every $n \in \omega$, there exists a set S which is Σ_{n+1}^0-*complete*: S itself is Σ_{n+1}^0, and every Σ_{n+1}^0 set T has $T \leq_1 S$. Likewise, the complement of S is Π_{n+1}^0-complete. This is regarded as an exact assessment of the complexity of S; among other things, it ensures that S is *not* Π_{n+1}^0, nor Σ_n^0. It should be noted that the class of Σ_0^0 sets and the class of Π_0^0 sets coincide: these are the computable sets,

and no set is Σ_0^0-complete. A set which is both Σ_n^0 and Π_n^0 is said to be Δ_n^0. Every Δ_1^0 set is Δ_0^0, but for greater n these classes no longer coincide. The Δ_{n+1}^0 sets are exactly those which are Turing-reducible to a Σ_n^0-complete oracle set. As a canonical Σ_n^0-complete set, we usually use $\emptyset^{(n)}$, the n-th jump of the empty set, as defined in [28].

Turing reducibility $\leq_T$ is a partial pre-order on the power set $\mathcal{P}(\omega)$. We define $A \equiv_T B$, saying that A and B are *Turing-equivalent*, if $A \leq_T B$ and $B \leq_T A$. The equivalence classes under this relation form the *Turing degrees*, and are partially ordered by $\leq_T$. In fact, they form an upper semi-lattice under $\leq_T$, with least element $\mathbf{0}$, the degree of the computable sets, but no greatest element. One often speaks of a set A as being *computable in a Turing degree $\mathbf{d}$*, meaning that for some (equivalently, for every) $B \in \mathbf{d}$ we have $A \leq_T B$.

3. Useful Results on Fields

Computable fields fit the general definition of computable structures, which is the basis of computable model theory.

Definition 3.1. A structure $\mathcal{S}$ in a finite language is *computable* if its domain is an initial segment of ω, the set of natural numbers, and all functions and relations in $\mathcal{S}$ are computable when viewed as functions and relations on ω.

A structure $\mathcal{M}$ is *computably presentable* if it is isomorphic to a computable structure $\mathcal{S}$, in which case we call $\mathcal{S}$ a *computable presentation* of $\mathcal{M}$.

A computable field F therefore has domain ω, or else $\{0, 1, \ldots, p^k - 1\}$, with computable field operations. Since the symbols 0 and 1 have their own meaning in field theory, we often write the domain of F as $\{x_0, x_1, \ldots\}$ for clarity. If the i-th partial computable function φ_i computes addition on F (so $x_m + x_n = x_{\varphi_i(m,n)}$) and φ_j computes multiplication on F, then $\langle i, j \rangle$ is an *index* for F. These definitions are standard in computable model theory, and we will maintain them here, but they complicate the discussion of computable fields. Herewith our conventions. The standard symbols $+$, $-$, $\cdot$, x^n, and $\frac{x}{y}$, from field theory all refer to the (computable) operations in F, with "$-$" denoting both subtraction and negation as usual. Likewise, 0 and 1 denote the identity elements of F, rather than naming the first two elements of the domain. These will be far more useful for us than the constants and operations on ω would be. On the other hand, we use the symbol $<$ to denote the usual relation on ω, *not on the field F*. Of course, an arbitrary computable field may or may not be orderable at all, let alone computably orderable, so field orders (i.e. linear orders compatible with the field operations) will not enter into our discussion. We will often want to search through the domain ω until we find an element with a certain property, and such a search will simply go through the elements $0, 1, 2, \ldots$ (or $x_0, x_1, x_2, \ldots$), using the $<$ relation on ω. Similarly, phrases such as "the least element satisfying..." will mean the least under $<$ on ω.

At a basic level, [19] and [21] are both useful for definitions about computable fields. They both avoid the notational issue by writing $\{a_0, a_1, \ldots\}$ in place of $\{0, 1, \ldots\}$ as the domain of a field. For serious research on computable fields through the twentieth century, all of [30], [7], [23], [4], [17], and [29] are familiar references.

A field is *algebraic* iff it is an algebraic extension of its prime subfield. (The prime subfield is just the smallest subfield; it is a copy of either the rationals $\mathbb{Q}$ or the p-element field $\mathbb{F}_p$, depending on characteristic.) In this paper, we will restrict

ourselves entirely to algebraic fields, although in characteristic 0, our results carry over to the case of fields of finite transcendence degree over $\mathbb{Q}$, just by fixing a transcendence basis B and treating $\mathbb{Q}(B)$ as the prime subfield, over which the rest of the field is algebraic.

We restrict ourselves further to the case of algebraic fields with splitting algorithms. A computable field F has a *splitting algorithm* if its *splitting set*

$$S_F = \{p \in F[X] : p \text{ is reducible in } F[X]\}$$

is computable. (To clarify: p is *reducible* if it can be expressed as a product of nonconstant polynomials in $F[X]$; it need not split into linear factors, but it must split into at least two proper factors.) The polynomial ring $F[X]$ may be presented effectively (i.e. as a computable ring) by use of the computable bijection from $\omega^{<\omega}$ onto ω described in Section 2. The next result relates S_F to other properties of the field F. This is a direct consequence of Rabin's Theorem, first published in [23] in 1960, and is discussed in more detail in [20], where it appears as Corollaries 2.7 and 2.8.

Lemma 3.2. *Let F be any computable field. Then the following are all Turing-equivalent: the splitting set of F*

$$S_F = \{p(X) \in F[X] : p(X) \text{ has a proper factorization in } F[X]\},$$

the root set *of F*

$$R_F = \{p(X) \in F[X] : (\exists r \in F)p(r) = 0\},$$

the root function g_F *of F*

$$g_F(p) = |\{r \in F : p(r) = 0\}|, \text{ with } dom(g_F) = F[X],$$

and the root multiplicity function *of F (which is the same as g_F, except that roots are counted by multiplicity). Moreover, the Turing reductions are uniform in (an index for) F.*

Moreover, any two isomorphic computable algebraic fields F and $\tilde{F}$ must have Turing-equivalent splitting sets, and the Turing reductions are uniform in F and $\tilde{F}$. Hence, for computable algebraic fields, the Turing degree of each item above is an invariant of the isomorphism type. □

Notice that an algebraic field need not be finitely generated (equivalently, need not have finite degree) over the prime subfield. Indeed, finitely generated computable fields are relatively straightforward objects: for one thing, they always have splitting algorithms, as do all prime fields. This was shown in 1882 by Kronecker, in [14]; a discussion in modern terms appears in [20, §2], where it is also explained how we can determine splitting algorithms, uniformly in the generators, for all finitely generated subfields E of any computable algebraic field F, and how we can use these to determine the Galois group of any finitely generated subfield $E \subseteq F$ over any subfield of E, uniformly in the generators of E and the subfield. We view the Galois group as a permutation group on the generators and their conjugates over the ground field; this is a useful way to consider automorphisms of E as finitary objects rather than as functions.

We add a new notion, the conjugacy function, which is related but not always Turing-equivalent to the sets and functions from Lemma 3.2.

Definition 3.3. Let F be a computable field, with prime subfield Q. The *conjugacy function for F* is the function $h : F \to \omega$ defined by:

$$h(x) = \begin{cases} \uparrow, \text{ if } x \text{ is transcendental over } Q \\ |\{y \in F : x \text{ and } y \text{ are conjugate over } Q\}|, \text{ else.} \end{cases}$$

In general, the conjugacy function is a partial function. Normally its initial step is to find the minimal polynomial of x over Q, and if x is transcendental, then this search will never converge. In practice we are usually concerned with the case of an algebraic extension F over Q, in which case h is total.

The conjugacy function is always computable from the splitting set, or from any of the other Turing-equivalent sets in Lemma 3.2. The splitting set allows us to find the minimal polynomial $q(X)$ of any x algebraic over Q, and then we use the root function to determine $h(x)$. However, the conjugacy function may have strictly smaller Turing degree than those sets. For example, in any normal extension of Q, the conjugacy function is computable, being given by the degree of the minimal polynomial; yet the computable normal algebraic field $\mathbb{Q}[\sqrt{p_n} \mid n \in K]$, generated by the square roots of the primes p_n with n in the halting set K, has splitting set of degree $\mathbf{0}'$.

The analogue of Rabin's Theorem for the conjugacy function is the following.

Proposition 3.4. *Let F be any computable algebraic field with prime subfield Q, and g any computable field embedding of F into a computable presentation $\overline{Q}$ of the algebraic closure of Q. Then the conjugacy function h of F is computable iff the image $g(F)$ is a computable subfield of its normal closure within $\overline{Q}$, i.e. iff there exists a partial computable function ψ whose domain is the normal closure of $g(F)$ within $\overline{Q}$, such that ψ is the characteristic function of $g(F)$ on that domain. More generally, this ψ is Turing-equivalent to h.*

Notice that the normal closure of $g(F)$ within $\overline{Q}$ is independent of the choice of the computable embedding g.

Proof. Suppose first that we have an h-oracle. Given any $x \in \overline{Q}$, ψ waits until some y appears in F such that $g(y)$ and x have the same minimal polynomial over the prime subfield Q of $\overline{Q}$. If this never happens, then x does not lie in the normal closure of $g(F)$, so $\psi(x)$ need not converge. If y appears, then we compute $h(y)$, using the oracle, and find all conjugates $y = y_1, y_2, \ldots, y_{h(y)}$ of y over Q in F. Then $x \in g(F)$ iff $(\exists i \le h(y)) \, g(y_i) = x$.

Conversely, with an oracle for ψ, the program for h accepts any $y \in F$ as input, computes the minimal polynomial $q(X)$ of $g(y)$ over Q, and finds all roots $x_1 = g(y), x_2, \ldots, x_{\deg(q)}$ of $g(y)$ in $\overline{Q}$. Then $\psi(x_i) \downarrow$ for all $i \le \deg(q)$, and $h(y)$ is the number of these i for which $\psi(x_i) = 1$. $\square$

Of course, often one wants to know about conjugates over ground fields other than the prime subfield. If this ground field is algebraic, then the conjugacy function computes this information.

Lemma 3.5. *Let F be a computable field, with prime subfield Q and conjugacy function h. Then for any subfield $E \subseteq F$ algebraic over Q, the function*

$$h_E(x) = \begin{cases} \uparrow, \text{ if } x \text{ is transcendental over } E \\ |\{y \in F : x \text{ and } y \text{ are conjugate over } E\}|, \text{ else} \end{cases}$$

is computable uniformly in oracles for h and the splitting set of E. In particular, if E is generated by a finite set $z_1, \ldots, z_n$ over Q, then we can compute h_E uniformly in h and the tuple $\vec{z}$.

Proof. For any $x \in F$, the splitting set for E allows us to find the minimal polynomial $p(X)$ of x over E. We also determine the minimal polynomial $q(X)$ of x over Q, and search until we have found all roots $x = x_1, x_2, \ldots, x_{h(x)}$ of $q(X)$ in F. Now $p(X)$ must divide $q(X)$ in the ring $E[X]$, so $h_E(x)$ is just the number of these roots x_i satisfying $p(x_i) = 0$.

Here F is not assumed to be algebraic over Q, but E is algebraic. So, if $x \in F$ is transcendental over Q, then x is also transcendental over E, and therefore the computation of $h_E(x)$ described above never halts, which is exactly the prescribed outcome. $\qquad\square$

Next we introduce several standard facts about fields which do not involve computability. The following result appears as Lemma 2.10 in [20].

Lemma 3.6. *For an algebraic field F, every endomorphism (i.e. every injective homomorphism from F into itself) is an automorphism.* $\qquad\square$

Corollary 3.7. *If $F \cong \tilde{F}$ are isomorphic algebraic fields, and $f : F \to \tilde{F}$ is a field embedding with $f(1) \neq 0$, then the image of f is all of $\tilde{F}$. That is, such an f must be an isomorphism.* $\qquad\square$

Before continuing to Proposition 3.8, we explain the intuition behind the *embedding tree*, which will be stated formally in Definition 5.1, but is used here as well. To construct a field embedding from E into F effectively, one usually begins with the prime subfield Q of E (which is always computably enumerable within E, and can be mapped effectively onto the prime subfield of F, provided the fields have the same characteristic). Then one extends this partial embedding f_0 to the least element x_0 in the domain $\{x_0, x_1, \ldots\}$ of E (which might already lie in Q, of course), then the second-least, and so on. The options for the image of x_{s+1} depend on the choices we made for the images of $x_0, \ldots, x_s$, naturally. At each stage, the number of ways to extend the embedding $f_s : Q(x_0, \ldots, x_{s-1}) \hookrightarrow F$ to x_s is bounded by the degree of the minimal polynomial of x_s over $Q(x_0, \ldots, x_{s-1})$, and there might be no way at all to do so, even assuming that E embeds into F, because we may have made bad choices at previous stages. Our choices thus naturally give rise to the *embedding tree*, with a root node representing the partial embedding f_0 (to which every embedding of E into F restricts), and with each node σ of length $|\sigma| = s$ having one immediate successor for each of the finitely many possible images in F of x_s, given our previous choices. Hence one may view σ as a node in the tree $F^{<\omega}$ of finite sequences of elements of F: the sequence $\langle y_0, \ldots, y_{s-1} \rangle$ describes the map with $x_i \mapsto y_i$ for all $i < s$, and this sequence lies in the isomorphism tree iff that map extends to an embedding of $Q(x_0, \ldots, x_s)$ into F. The infinite paths through this tree correspond precisely to the embeddings of E into F. Moreover, the tree itself is computable, in the sense that we can decide exactly which sequences in $F^{<\omega}$ lie in the tree. Nevertheless, some or all of the paths through the tree may be noncomputable.

Proposition 3.8. *Let E and F be algebraic field extensions of a common subfield Q. Then E embeds into F over Q iff every finitely generated subfield of E containing Q embeds into F over Q.*

Proof. The key to the nontrivial direction is König's Lemma, applied to the *embedding tree* $T_{E,F;Q}$ for E and F over Q, as we now explain. Write the domain of E as $E = \{x_0, x_1, \ldots\}$, with ω as the domain of F. For each s, pick any $q_s \in Q[X_0, \ldots, X_s]$ such that $q_s(x_0, \ldots, x_{s-1}, X)$ is the minimal polynomial of x_s over $Q[x_0, \ldots, x_{s-1}]$. We consider the set $T_{E,F;Q}$ of finite sequences σ of natural numbers:

$$T_{E,F;Q} = \{\sigma : (\forall s < |\sigma|) \; q_s(\sigma(0), \ldots, \sigma(s)) = 0\}.$$

Notice that if $\sigma = (a_1, \ldots, a_n) \in T_{E,F;Q}$, then so is $\tau = (a_1, \ldots, a_m)$ for each $m = |\tau| \leq n = |\sigma|$. That is, $T_{E,F;Q}$ is closed under initial segments, so we view it as a tree.

Sequences $\sigma \in T_{E,F;Q}$ correspond to embeddings of $Q[x_0, \ldots, x_{|\sigma|-1}]$ into F, since for each s less than the length $|\sigma|$ of the sequence, $\sigma(s)$ satisfies the same minimal polynomial over $Q(\sigma(0), \ldots, \sigma(s-1))$ in F that x_s satisfies over $Q(x_0, \ldots, x_{s-1})$ in E. Clearly $T_{E,F;Q}$ is finite-branching: the number of successors of σ is at most the degree of the variable $X_{|\sigma|}$ in $q_{|\sigma|}$. (A more general formal description appears as Definition 5.1 below.)

König's Lemma states that every finite-branching tree with infinitely many nodes must contain an infinite path. By assumption, every subfield $Q[x_0, \ldots, x_n]$ embeds into F, so $T_{E,F;Q}$ contains a node of each length $n \in \omega$. Therefore, $T_{E,F;Q}$ contains an infinite path, which defines an embedding of F into E. $\qquad\square$

A less slick but intuitively clearer proof of this result (with Q as the prime subfield) appears within the proof of [8, Appendix A, Thm. 2]. Sometimes we will write just $T(E, F)$, in which case Q is to be understood as the prime subfield of both. The concept of the embedding tree foreshadows the *isomorphism trees* in Definition 5.1 below.

Corollary 3.9. *Two algebraic fields E and F are isomorphic over a common subfield Q iff every finitely generated subfield (containing Q) of each one embeds over Q into the other.*

Proof. By Proposition 3.8, the latter condition is equivalent to E and F both embedding into each other over Q. But then the composition of these two embeddings is an automorphism, by Lemma 3.6. $\qquad\square$

The upshot of this corollary is that often, to build a computable field F isomorphic to a given computable field E, we can simply construct E as a union of nested, uniformly computable fields $E_s \subseteq E_{s+1}$ with each E_s isomorphic to the subfield $F_s \subseteq F$ generated by the first s elements of F. There is no need for the isomorphisms $f_s : F_s \to E_s$ to have a limit; Corollary 3.9 does all the work for us.

For this paper we have two new definitions, arising out of the standard concept of conjugacy. Examples appear in Section 4.

Definition 3.10. Let $E \subseteq F$ be any field extension. Two elements $a, b \in F$ are *conjugate over E* if they have the same minimal polynomial in $E[X]$. It is well known that then the subfields $E[a]$ and $E[b]$ are isomorphic, via a map fixing E pointwise with $a \mapsto b$. We say that a and b are *true conjugates in F over E* if there exists an automorphism ψ of F with $\psi(a) = b$ and $\psi{\restriction}E$ being the identity map. If a and b are conjugate over E but are not true conjugates in F over E, we call them *false conjugates in F over E*.

Finally, we give the computability-theoretic version of the classical Theorem of the Primitive Element.

Theorem 3.11 (Effective Theorem of the Primitive Element). *Let E and F be computable fields, with $E \subseteq F$ a separable algebraic extension and with E c.e. as a subfield of F. Then for any elements $x_1, \ldots, x_n \in F$, we can effectively find a primitive generator $y \in F$ for these elements. That is, we can find a $y \in F$ such that $E[y] = E[x_1, \ldots, x_n]$. The procedure for finding y is uniform in n, the generators $\langle x_1 \ldots, x_n \rangle$, the enumeration of E within F, and the field operations in F.*

Proof. The existence of such an element y is the classical theorem see e.g. [30, p. 139], and is made effective by a direct search for y. An arbitrary $y \in F$ generates $E[x_1, \ldots, x_n]$ iff

$$(\exists p \in E[X_1, \ldots, X_n])(\exists q_1, \ldots, q_n \in E[Y])[y = p(x_1, \ldots, x_n) \ \& \ \text{all } x_i = q_i(y)],$$

and so we can find y, with the uniformities described above. For a proof giving an actual formula for the generator y, see [6]. □

4. A First Example

We start with an example, to introduce the concepts that will be used later in our analysis of computable categoricity for algebraic fields. Let F_0 be a computable presentation of the normal algebraic extension of the rationals by the square roots of all rational primes: $F_0 = \mathbb{Q}[\sqrt{p_0}, \sqrt{p_1}, \ldots]$. The domain of F_0 is ω, and we use $\sqrt{p_i}$ to refer to the lesser of the two square roots of p_i in F_0, under the ordering $<$ on the domain ω. Now, for any $W \subseteq \omega$, let F_W be the extension of F_0 in which we adjoin a square root of $\sqrt{p_i}$ iff $i \in W$. Notice that no $-\sqrt{p_i}$ acquires a square root of its own during this process, whether or not $i \in W$. (To see this, embed F_W into the field $\mathbb{R}$.) So F_W is not normal over $\mathbb{Q}$ (unless $W = \emptyset$), although it is normal over F_0, which in turn is normal over $\mathbb{Q}$. F_W is computably presentable iff W is computably enumerable, in which case we take F_W to denote a computable presentation built over our original presentation of F_0. The domain of F_0 can be the set of even elements of ω, for instance, with the odd elements added as numbers i appear in W and dictate the adjoinment of square roots of $\sqrt{p_i}$ and the new elements they generate.

Notice that for all $i \in W$, $\sqrt{p_i}$ and $-\sqrt{p_i}$ are false conjugates in F_W over $\mathbb{Q}$, as in Definition 3.10: they both have minimal polynomial $X^2 - p_i \in \mathbb{Q}[X]$, but neither can be mapped to the other by any automorphism of F_W over $\mathbb{Q}$, since $\sqrt{p_i}$ has a square root of its own and $-\sqrt{p_i}$ does not. On the other hand, for $i \notin W$, $\pm\sqrt{p_i}$ are true conjugates in F_W over $\mathbb{Q}$, since there is an automorphism of F_W mapping one to the other.

Before stating the main result for these fields F_W, we recall the concept of computable inseparability.

Definition 4.1. Two sets P and N are *computably inseparable* if there is no computable set C with $P \subseteq C$ and $N \subseteq \overline{C}$.

A standard example (see [28, I.4.22]) has $P = \{n \in \omega : \varphi_n(n) \downarrow = 0\}$ and $N = \{n \in \omega : \varphi_n(n) \downarrow = 1\}$. (Recall that "$\varphi_e(x) \downarrow = 0$" means that the e-th partial computable function, when run with the input x, halts and outputs 0.) The following result was first proven by Yates, who saw that it followed from a construction of Friedberg; but the proof was only published by Cleave [3] in 1970, some years later.

Theorem 4.2 (Friedberg & Yates [3]). *Every noncomputable c.e. set is the union of two disjoint, computably inseparable c.e. subsets.* □

Conversely, if W is computable, then for every partition of W into c.e. sets A and B, both A and B are computable, since the complement $\overline{A}$ of the c.e. set A is the set $(\overline{W} \cup B)$, which is also c.e.

Proposition 4.3. *For any c.e. W, the computable field F_W is computably categorical iff W cannot be partitioned into computably inseparable c.e. subsets.*

Corollary 4.4. *F_W is computably categorical iff W is computable.* □

The corollary is immediate, using Theorem 4.2. The point of Proposition 4.3 is its proof (below), rather than the result itself, since the proof illustrates the usefulness of the concept of computable inseparability (see Definition 4.1).

Proof. First suppose that there is no partition of W into computably inseparable c.e. sets, and let $\tilde{F}$ be a computable field isomorphic to F_W. Define

$$P = \{i \in \omega : \exists x, y, z \in \tilde{F}[x < y \,\&\, x^2 = y^2 = \tilde{p}_i \,\&\, z^2 = x]\}$$
$$N = \{i \in \omega : \exists x, y, z \in \tilde{F}[y < x \,\&\, x^2 = y^2 = \tilde{p}_i \,\&\, z^2 = x]\}.$$

(Here $<$ refers to the standard order on the domain ω of $\tilde{F}$, but $\tilde{p}_i$ is the i-th rational prime in $\tilde{F}$, and squares refer to the field multiplication in $\tilde{F}$.) The computability of $\tilde{F}$ shows that P and N are both c.e., and since $\tilde{F} \cong F_W$, we know that $P \cup N = W$ and $P \cap N = \emptyset$. By assumption, therefore, there exists a computable C with $P \subseteq C$ and $N \subseteq \overline{C}$. We define our computable isomorphism $f : F_W \to \tilde{F}$ beginning with the computable subfield $F_0 \subseteq F_W$. f is uniquely determined on $\mathbb{Q}$ within F_0. If $i \in C$, then we map $\sqrt{p_i}$ (from F_W) to the lesser square root of $\tilde{p}_i$ in $\tilde{F}$, with $-\sqrt{p_i}$ therefore mapped to the greater one. In this situation we know that either $i \notin W$ (in which case $\sqrt{p_i}$ can be mapped to either square root of $\tilde{p}_i$) or $i \in P$ (in which case we made the correct choice by mapping $\sqrt{p_i}$ to the lesser square root of $\tilde{p}_i$). If $i \notin C$, we do the opposite, mapping $\sqrt{p_i}$ to the greater square root of $\tilde{p}_i$ in $\tilde{F}$. With $i \notin C$, we know that either $i \notin W$ or $i \in N$, and so again the choice we made was a correct choice. This much is readily computable, since C is computable.

Moreover, as elements x outside of F_0 appear in F_W, we can compute the (unique) extension of f to those elements: if a square root z_i of $\sqrt{p_i}$ ever appears in F_W, then $i \in W = P \cup N$, and our choice of $f(\sqrt{p_i})$ using C ensures that $f(\sqrt{p_i})$ has a square root of its own in $\tilde{F}$, to which we map z_i. Since x must have been generated by finitely many of these z_i, this will eventually allow us to extend f to x. Clearly, then, f is a computable embedding of F_W into $\tilde{F}$, and by Corollary 3.7, f must be an isomorphism. Thus F_W is computably categorical.

Now suppose there exist disjoint, computably inseparable sets P and N whose union equals W. We define a computable field F isomorphic to F_W by starting with F_0 and enumerating P and N. Whenever any i appears in P, we adjoin a square root of $\sqrt{p_i}$ to F, while when an i enters N, we adjoin a square root of $-\sqrt{p_i}$ to F. Since P and N form a partition of W, this F must be isomorphic to F_W. However, if $g : F_W \to F$ is any isomorphism, then $C = \{i \in \omega : g(\sqrt{p_i}) < g(-\sqrt{p_i})\}$ is computable in g and must contain P while not intersecting N. (Here again $<$ is the standard order on the domain ω of F.) Therefore no such g can be computable, and F is a computable field isomorphic to F_W but not computably isomorphic to it. □

Relativizing this proof yields an immediate corollary.

Corollary 4.5. *For c.e. sets W, the above field F_W is $\boldsymbol{d}$-computably categorical iff the degree $\boldsymbol{d}$ computes separations of every partition of W into two c.e. subsets. That is, the degrees with this property form the* categoricity spectrum *of F_W, as defined in [5].*   □

We note that the splitting set of this field F_W is Turing-equivalent to W, hence is in general noncomputable. There is a similar procedure which builds a computable field K_W with computable splitting set, and with the same categoricity spectrum: the basic strategy is that, if n enters W at stage s, one should adjoin a root of a polynomial $q_n(\sqrt{p_n}, Y)$ in such a way that $q_n(-\sqrt{p_n}, Y)$ has no root. By ensuring that these polynomials all have distinct prime degrees, we can make certain that $q_n(-\sqrt{p_n}, Y)$ never acquires a root at any other stage, and by making that prime degree d be $> s$, we can keep the splitting set of F computable. Unfortunately, $q_n(\sqrt{p_n}, Y)$ cannot simply say $Y^d = \sqrt{p_n}$, because then the negative of this root would be a d-th root of $-\sqrt{p_n}$. For the existence of the requisite polynomials q_n, see [20, Prop. 2.15], and for the full construction see Theorem 3.4 there. With this note, we have informally established the following corollary.

Corollary 4.6. *Computable categoricity for computable algebraic fields with splitting algorithms is Σ_3^0-hard.*

Proof. This means that there is a 1-reduction f from some Σ_3^0-complete set S: for all $e \in S$, $f(e)$ is the index of an algebraic field with a splitting algorithm, which is computably categorical iff $e \in S$. We choose the set $\mathbf{Rec} = \{e : W_e \leq_T \emptyset\}$ to serve as S, and $f(e)$ is the field index produced by the above construction on W_e. For details, see [28, §IV.3].   □

It was also shown in [20] that a field such as K_W (with $W >_T \emptyset$), with a splitting algorithm but not computably categorical, cannot have any least Turing degree in its categoricity spectrum. By Corollary 4.5, this shows that for noncomputable c.e. sets W, there is no least degree which computes a separation of every partition of W into two c.e. subsets. Of course, this result can be shown directly, without the excursus into computable model theory! The key to the direct proof, and also to the result in [20], is the Low Basis Theorem of Jockusch and Soare, from [12].

With the preceding results, we have answered a question which Hirschfeldt posed upon encountering categoricity spectra for algebraic structures.

Corollary 4.7. *There exists a computable algebraic field whose categoricity spectrum contains precisely the PA-degrees, i.e., those Turing degrees which compute a complete extension of Peano arithmetic.*

Proof. Let A and B be two effectively inseparable disjoint c.e. sets, such as $\{n : \varphi_n(n) \downarrow = 0\}$ and $\{n : \varphi_n(n) \downarrow = 1\}$, and set $W = A \cup B$. If the field K_W is $\boldsymbol{d}$-computably categorical, then $\boldsymbol{d}$ must compute a separation of A and B, by Corollary 4.5. It then follows by known results of Scott and Solovay (see [26], or [27, Thm. 6.6]) that this $\boldsymbol{d}$ must be a PA-degree. Conversely, since K_W has a splitting algorithm, the embedding tree $T_{K_W, E, \mathbb{Q}}$ has computable branching whenever E is a computable field isomorphic to K_W, and therefore every PA-degree $\boldsymbol{d}$ can compute a path through this tree, i.e. an isomorphism from K_W onto E. Thus K_W is $\boldsymbol{d}$-computably categorical for precisely the PA-degrees $\boldsymbol{d}$.   □

Effective inseparability of A and B is a stronger property than computable inseparability, and not every c.e. set $W >_T \emptyset$ can be partitioned into effectively inseparable c.e. subsets. Indeed, one can produce a computable algebraic field with a splitting algorithm, for which the categoricity spectrum contains certain non-PA degrees as well, yet does not contain the degree $\mathbf{0}$.

5. Isomorphism Trees for Fields

Computable separability arises frequently in discussions of computable trees and paths through them. In light of the connection seen in the preceding section between computable separability and computably categorical fields, it is not surprising that, in discussing isomorphisms between fields, we will make great use of the *isomorphism tree*. This concept was introduced in [20, §5] and used in Proposition 3.8 above; it may be useful for the reader to refer back now to the discussion there. Here we give a full definition, generalizing the work in [20] to the case where the two fields are not necessarily isomorphic. $F^{<\omega}$ denotes the set of finite sequences σ of elements of F (or, formally, functions $\sigma : \{0, 1, \ldots, |\sigma| - 1\} \to F$, where $|\sigma|$ is the length of σ).

Definition 5.1. Let E and F be computable algebraic fields of the same characteristic, and let $\langle x_0, x_1, \ldots \rangle_{i \in J}$ be a computable sequence (finite or infinite) of elements of E which together generate E over its prime subfield Q. Let $\langle q_i \rangle_{i \in J}$ be polynomials such that every $q_i \in Q[X_0, \ldots, X_i]$ and for each i, $q_i(x_0, \ldots, x_{i-1}, X)$ is the minimal polynomial of x_i over $Q(x_0, \ldots, x_{i-1})$. The *embedding tree* I_{EF} is the computable tree

$$I_{EF} = \{\sigma \in F^{<\omega} : (\forall i < |\sigma|)\tilde{q}_i(\sigma(0), \ldots, \sigma(i)) = 0\},$$

where $\tilde{q}_i$ is the image of q_i under the unique embedding of Q into F. The height of I_{EF} is $|J| + 1$ for finite generating sets J, or ω if $|J| = \omega$, and I_{EF} must be finite-branching: for each $\sigma \in I_{EF}$ of any length n, there are only finitely many $\tau \in I_{EF}$ such that $\sigma \subseteq \tau$ and $|\tau| = n+1$, because there are only finitely many roots of $q_{n+1}(\sigma(0), \ldots, \sigma(n-1), X)$ in F.

When E and F are known to be isomorphic, the term *isomorphism tree* is often used (with Corollary 3.7 as justification). If $E = F$ and $I = \langle x_0, x_1, \ldots \rangle$ simply lists the domain of F in order (as elements of ω), then we write I_F for I_{FF} and call this the *automorphism tree* of F. (We sometimes abuse this terminology by writing I_F even when a different generating sequence is being used, as long as it is clear which sequence it is.) The *identity path* in the automorphism tree I_F contains all nodes of the form $\langle x_0, x_1, \ldots, x_i \rangle$; clearly this is a path through I_F, corresponding to the identity automorphism, since the map it defines sends each x_i to x_i itself. The *level function* for F is the function giving (for each n) the number of nodes at level n in I_F.

In Definition 5.1 in [20], the sequence $\langle x_i \rangle_{i \in I}$ was assumed simply to be the domain of F. For the purposes of this paper, we generalized to arbitrary computable generating sequences, but this does not change the central point about isomorphism trees, which is the following.

Lemma 5.2. *If $F \cong \tilde{F}$, then there is a bijection $\psi \mapsto h_\psi$ from the set of field isomorphisms $\psi : F \to \tilde{F}$ onto the set of paths through $I_{F\tilde{F}}$, with $\psi \equiv_T h_\psi$ for all*

such isomorphisms ψ. (For each path $h \in \tilde{F}^\omega$ through $I_{F\tilde{F}}$, we write ψ_h for the unique field isomorphism such that $h = h_{\psi_h}$, namely $\psi_h(x_i) = h(i)$.)

Proof. For each ψ, the path h_ψ is the set of all nodes $\sigma \in I_{F\tilde{F}}$ of the form $\langle \psi(x_0), \psi(x_1), \ldots, \psi(x_{|\sigma|-1}) \rangle$. Since ψ is an isomorphism, this is clearly a path in $I_{F\tilde{F}}$, and conversely, for any path h, setting $\psi(x_i) = \sigma(i)$ for the unique $\sigma \in h \cap \tilde{F}^{(i+1)}$ defines a field embedding ψ, which must be an isomorphism by Corollary 3.7, with $h = h_\psi$. The Turing equivalence is immediate from these definitions. Also, when $\tilde{F} = F$, the image h_{id} of the identity automorphism is the identity path through I_F. $\qquad\qquad\square$

We note that Lemma 5.2 holds independently of the choice of generating sequence of F. The isomorphism trees for two different generating sequences may not be isomorphic to each other as trees, but nevertheless the set of paths through each one still corresponds to the set of isomorphisms from F onto $\tilde{F}$. The polynomials q_i are not uniquely determined by the generating sequence; one can use any q_i, satisfying the condition that $q_i(x_0, \ldots, x_{i-1}, X)$ be the minimal polynomial of x_i over $Q(x_0, \ldots, x_{i-1})$, and $I_{F\tilde{F}}$ will come out exactly the same. Finally, replacing $\tilde{F}$ by a different field isomorphic to F would not change anything about the isomorphism tree except the names of the nodes.

Lemma 5.3. *For computable algebraic fields $F \cong \tilde{F}$, and for a fixed generating sequence for F, we always have an isomorphism of trees:*

$$I_{F\tilde{F}} \cong I_F.$$

Indeed, there is a canonical 1-1 map from field isomorphisms $\psi : F \to \tilde{F}$ to tree isomorphisms $H_\psi : I_F \to I_{F\tilde{F}}$, i.e. to bijective maps H which preserve the successor relation on the trees. Hence there is also a canonical 1-1 map from paths through $I_{F\tilde{F}}$ to such tree isomorphisms, with $H_\psi \equiv_T \psi$ uniformly in ψ.

Proof. Fix any field isomorphism ψ, and, for $\sigma \in I_F$ with $|\sigma| = n$, define H_ψ on I_F by

$$H_\psi(\sigma) = \langle \psi(\sigma(0)), \ldots, \psi(\sigma(n-1)) \rangle \in I_{F\tilde{F}}.$$

Clearly H_ψ is a tree isomorphism: $H_\psi(\sigma) \in I_{F\tilde{F}}$ because

$$q_n(\psi(\sigma(0)), \ldots, \psi(\sigma(n-1))) = \psi(q_n(\sigma(0), \ldots, \sigma(n-1))) = \psi(0) = 0,$$

using the definition of I_F to see that $q_n(\sigma(0), \ldots, \sigma(n-1)) = 0$, and bijectivity and preservation of the successor relations on the trees follow from ψ being a field isomorphism.

The definition of H_ψ shows it to be computable from ψ. Conversely, if we know H_ψ, we can apply it to nodes $\langle x_0, \ldots, x_s \rangle$ on the identity path in I_F, yielding $\langle \psi(x_0), \ldots, \psi(x_s) \rangle$, thus computing ψ on the generating sequence $\langle x_s \rangle_{s \in I}$ for the field F. $\qquad\qquad\square$

The canonical extension of the pairing need not be the only possible extension to an isomorphism from I_F onto $I_{F\tilde{F}}$, and so the 1-1 map in Lemma 5.3 need not be onto. Any tree isomorphism $H : I_F \to I_{F\tilde{F}}$ must map the identity path to some path h through $I_{F\tilde{F}}$, and thus gives rise to a field isomorphism ψ_h, although H_{ψ_h} may not equal H. For example, if F has generating sequence x_0, x_1, x_2, where $x_0 \in \mathbb{R}$ and $x_2 \notin \mathbb{R}$ are cube roots of 2 and x_1 is a root of $p(x_0, Y)$ for some $p \in \mathbb{Q}[X, Y]$ such that the resulting field F contains no roots of $p(x_2, Y)$ or $p(\overline{x_2}, Y)$, then the only nontrivial automorphism of F is complex conjugation, with $x_2 \mapsto \overline{x_2}$, yet the

tree I_F for this generating sequence has four automorphisms. The automorphism tree I_F contains two terminal nodes $\langle x_2 \rangle$ and $\langle \overline{x_2} \rangle$ at level 1, which represent what happens when one maps x_0 to x_2 (or to $\overline{x_2}$) and then finds that there is no element of F to which to map x_1. There is an automorphism θ of I_F which fixes each path through I_F, but interchanges these two terminal nodes. This θ is not equal to H_ψ, no matter which of the two automorphisms of F we try: H_{id} fixes I_F pointwise, and when ψ is complex conjugation, H_ψ interchanges the two paths through I_F. So the tree isomorphism θ is not the image of any field isomorphism under the canonical 1-1 map from Lemma 5.3.

The Turing degree of the branching of an automorphism tree I_F is of interest, especially in light of its use in [20]. Recall that the level function for F gives the number of nodes at each level n in I_F.

Lemma 5.4. *For any algebraic field F, the level function l for F is Turing-computable in the conjugacy function h for F.*

Proof. The number $l(\sigma)$ of immediate successors of a node $\sigma \in I_F$ is exactly the number of conjugates of $x_{|\sigma|}$ in F over the subfield $Q[x_0, \ldots, x_{|\sigma|-1}]$. We then appeal to Lemma 3.5. □

The interesting part, however, is that l need not be Turing-equivalent to h, and indeed the Turing degree of l need not even be invariant under distinct presentations of the field F. Thus, Lemma 3.2 applies to the root set, the splitting set, the root function, and the conjugacy function, but *not* to the level function! To see this, consider the following micro-example. Suppose that F has domain $\{z_0, z_1, \ldots\}$, and that $z_0^2 = z_5^2 = 2$, $z_1^2 = z_2^2 = z_0$, $z_3^2 = z_4^2 = z_1$, and that if a certain m appears in the halting set K, then z_5 later acquires two square roots of its own in F. If $m \notin K$, then z_5 has no square roots in F, and in no case does z_5 have any fourth roots in F.

Now if we build the automorphism tree I_F using the presentation $F = \{x_0 = z_0, x_1 = z_1, \ldots\}$ exactly as above, then the level function l_F is not computable: I_F contains the nodes $\langle x_0, x_1 \rangle$ and $\langle x_0, x_2 \rangle$, and also contains $\langle x_5 \rangle$, but it is impossible to decide whether $\langle x_5 \rangle$ has no immediate successors or two, since this depends on whether $m \in K$. (One would repeat this strategy, adding other field elements to F as roots of primes other than 2, to code whether each other $m' \in \omega$ lies in K or not.)

On the other hand, now consider the distinct presentation $E = \{y_0 = z_3, y_1 = z_0, y_2 = z_1, y_3 = z_2, y_4 = z_4, \ldots, y_i = z_i, \ldots\}$ of the same field F. Using this presentation, we get a different sequence of minimal polynomials: y_0 has minimal polynomial $Y_0^8 - 2$, so $q_0(Y_0) = Y_0^8 - 2$. Then y_1 has minimal polynomial $Y_1 - y_0^4$ over $\mathbb{Q}[y_0]$, since $y_1 = y_0^4$ is in this field. That is, $q_1(Y_0, Y_1) = Y_1 - Y_0^4$. Next, $q_2(Y_0, Y_1, Y_2) = Y_2 - Y_0^2$, since $y_2 = z_1 = z_3^2 = y_0^2$, and $q_3 = Y_3 + Y_2$, and $q_4 = Y_4 + Y_0$, and thereafter each q_i is the same as in the previous presentation, except with the roles of Y_0 and Y_3 reversed. This seemingly trivial difference between E and F changes the structure of the corresponding automorphism tree I_E: now the only nodes at level 1 are $\langle y_0 \rangle$ and $\langle y_4 \rangle$, each with exactly one successor at level 2, exactly one at level 3, at level 4, at level 5, and still exactly one at level 6 (these being $\langle y_0, y_1, y_2, y_3, y_4, y_5 \rangle$ and $\langle y_4, y_1, y_2, y_3, y_0, y_5 \rangle$), and so on. So in this case the level function for I_E is computable! Since the eighth roots of 2 appeared first, the question of how many fourth roots of 2 lie in E is obviated. It is possible, of course, that two more fourth roots of 2 (specifically, square roots of z_5) may appear in E,

as z_i and z_j for some large $i < j$, but if so, then we will simply have $q_i = X_i^2 - X_5$ and $q_j = X_j + X_i$ for the presentation F, and likewise for the presentation E, and we will be able to compute the number of nodes at those levels in each presentation. In E, the appearance of z_3 before z_0 and z_5 allows the level function l_E to evade the noncomputable question of whether z_5 has square roots.

Of course, this micro-example is really the basic module for the construction of two computable presentations E and F of a single field, with the property that I_E has computable level function but I_F does not. In the basic module above, one would adjoin two square roots of z_5 when and if 0 enters K. Likewise, for every m, start with a full complement of p_m-th roots of 2, exactly one of which has p_m-th roots and also (p_m^2)-th roots of its own. If m enters K, then adjoin all remaining (p_m^2)-th roots of 2 to the field. The presentations differ exactly as in the basic module, with the (p_m^2)-th roots of 2 appearing before the (p_m^3)-th roots of 2 in the field F, but after them in the field E. One can compute (for arbitrary m) the number of (p_m^3)-th roots of 2 in this field, but not the number of (p_m^2)-th roots.

6. Orbit Relations on Fields

In this section we wish to consider the action on a field F of the automorphism group $\mathrm{Aut}(F)$ of F. We continue to assume that F is computable and algebraic. Of course, the automorphism group may have cardinality as high as 2^ω, making it difficult to present in an effective fashion. However, since the field F may be viewed as the union of an effectively presented chain of finitely generated subfields, we will be able to make substantial use of the following definition.

Definition 6.1. Let F be any computable field, and let G be any subgroup of the automorphism group of F. The *full action of G on F* is the set

$$\{\langle a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}\rangle : (\exists \sigma \in G)(\forall i < n)\sigma(a_i) = b_i\},$$

where $\vec{a}$ and $\vec{b}$ are tuples of elements of F, with every n allowed as their common length. When F is algebraic, we will be able to restrict our attention to the *action of G on F*, which by definition is the set

$$\{\langle a, b\rangle : a, b \in F \ \& \ (\exists \sigma \in G)\sigma(a) = b\}.$$

If this set is computable, we say that *G acts computably on F*.

If $G = \mathrm{Aut}(F)$, we also call these the *full orbit relation* and the *orbit relation* of F, respectively.

It is quite possible for an uncountable G to act computably on F: for example, the entire automorphism group G of (a computable presentation of) the algebraic closure $\overline{Q}$ has size 2^ω, and contains elements of every Turing degree, yet its full action on $\overline{Q}$ is computable. This will follow from Lemma 6.2 and Corollary 6.5 below.

Since we restrict our attention in this paper to algebraic fields, we will only consider the action of a G on an F, not the full action. The following lemma justifies this.

Lemma 6.2. *Let F be a computable algebraic field of characteristic 0, and G a subgroup of $\mathrm{Aut}(F)$. Set B to be the action of G on F, and A the full action. Then $B \equiv_1 A$. In particular, each is computable iff the other is, and each is c.e. iff the other is.*

In characteristic p, the proof below ensures that $B \equiv_m A$. 1-equivalence would be false in a finite field F.

Proof. $B \leq_1 A$ is immediate, so we build a function g to show $A \leq_1 B$. Let $(a_0, \ldots, a_n; b_0, \ldots, b_n) \in F^{2n+2}$. First we check whether the map $a_i \mapsto b_i$ extends to an isomorphism from $Q(\vec{a})$ onto $Q(\vec{b})$, where Q is the prime subfield of F. This is computable, since we need only find the minimal polynomial of each a_i over $Q(a_0, \ldots, a_{i-1})$ and check that b_i satisfies the corresponding polynomial over $Q(b_0, \ldots, b_{i-1})$, with each a_j $(j < i)$ mapped to b_j to determine the coefficients in this corresponding polynomial. If $a_i \mapsto b_i$ does not extend to an isomorphism, then clearly $\langle \vec{a}, \vec{b} \rangle$ does not lie in the full action of G on F, and we define $g(\langle \vec{a}, \vec{b} \rangle)$ to lie outside B. (To ensure injectivity, let it be the $\langle \vec{a}, \vec{b} \rangle$-th element of some infinite c.e. subset of $\overline{B}$.) If it does extend to an isomorphism, we use Theorem 3.11 to find a single element $a \in Q(\vec{a})$ and polynomials $p \in Q[X_0, \ldots, X_n]$ and $q_i \in Q[X]$ such that $a = p(\vec{a})$ and $a_i = p_i(a)$ for each $i \leq n$. (Since there are infinitely many such a, we may choose ours to preserve injectivity of g.) By the isomorphism above, each $b_i = p_i(p(\vec{b}))$ as well. So $\langle \vec{a}, \vec{b} \rangle$ lies in A iff $g(\langle \vec{a}, \vec{b} \rangle) = \langle a, p(\vec{b}) \rangle$ lies in B. $\qquad\square$

On its face, membership in B_F is a Σ_1^1 property: it demands the existence of a function from ω to ω satisfying certain arithmetic properties. In fact, though, the algebraicity of the field F makes B_F (and its computable isomorph A_F) vastly simpler than this. The proof demonstrates the usefulness of the isomorphism trees defined in Section 5.

Proposition 6.3. *Let F be a computable algebraic field. Then an arbitrary pair $\langle a; b \rangle$ of elements of F lies in the field's orbit relation B_F iff:*

- *b is conjugate to a over the prime subfield Q; and*
- *$(\forall p \in Q[X, Y])[p(a, Y)$ has a root in $F \implies p(b, Y)$ has a root in $F]$.*

It follows that the orbit relation B_F of F is Π_2^0. More specifically, B_F is Π_1^S, where S is the splitting set of F, and so if F has a splitting algorithm, then its orbit relation is co-c.e.

By Lemma 6.2, these results also apply to A_F, of course. The surprise is that the second condition is not symmetric in a and b; this is essentially a consequence of Corollary 3.7, which can be used to show that the second condition is equivalent to the same statement with a and b interchanged.

Proof. If $\langle a, b \rangle \in B_F$, then the two conditions given are immediate. For the backwards direction, let $\{y_0, y_1, \ldots\}$ be the domain of F. (Of course, this domain is really ω; we write y_n instead of n to avoid confusion with the language of fields.) Given $\langle a, b \rangle \in F \times F$, define $x_0 = a$ and $x_{s+1} = y_s$ for all n. So $\langle x_s \rangle_{s \in \omega}$ is a computable generating sequence for F, and we may construct the automorphism tree I_F relative to this sequence, along with the sequence $\langle q_s \rangle_{s \in \omega}$ of minimal polynomials, as in Definition 5.1. Thus $q_0(X)$ is the minimal polynomial of a over the prime subfield Q. Of course, some $n > 0$ has $x_n = a$, but this only means that $q_n(X_0, \ldots, X_n) = X_n - X_0$.

The first condition in the proposition is that $q_0(b) = 0$. Since we are assuming that both conditions hold, $\langle b \rangle$ is a node at level 1 in I_F, and we claim that $\langle b \rangle$ lies on a path through I_F. To see this, fix any $n \in \omega$ and consider a primitive generator $u \in F$ of the subfield $Q(a, x_1, x_2, \ldots, x_n)$. Choose $p \in Q[X, Y]$ so that $p(a, Y)$ is the minimal polynomial of u over $Q(a)$. Since F satisfies the second

condition in the proposition, $p(b, Y)$ must also have a root v in F, and so there is an isomorphism $h : Q(a, u) \to Q(b, v)$ with $h(a) = b$ and $h(u) = v$. But then $\langle b, h(x_1), h(x_2), \ldots, h(x_n) \rangle \in I_F$, by the definition of I_F. Since this holds for all n, the node $\langle b \rangle$ has arbitrarily long successors in I_F, and so, applying König's Lemma to the finite-branching tree I_F, $\langle b \rangle$ lies on a path through I_F. This path defines an automorphism of F mapping a to b, so $\langle a, b \rangle \in B_F$ as required.   □

Indeed, the preceding proof showed more than Proposition 6.3 stated. We could have used any generating set for F in place of the domain $\{y_0, y_1, \ldots\}$, and constructed the automorphism tree I_F relative to this sequence (with a attached as the first element of the sequence). The proof really showed that $\langle a, b \rangle \in B_F$ iff the node $\langle b \rangle$ had successors at arbitrary high levels in I_F.

Corollary 6.4. *Let $\{z_0, z_1, \ldots\}$ be a computable sequence of elements of a computable algebraic field F with prime subfield Q, such that*

$$Q \subseteq Q(z_0) \subseteq Q(z_1) \subseteq \cdots \ \text{and} \ \cup_s Q(z_s) = F.$$

A pair $\langle a, b \rangle$ of elements of F lies in the field's orbit relation B_F iff a and b are conjugate over Q and, for every s and every $p_s \in Q[X, Y]$ such that $p_s(a, Y)$ is the minimal polynomial of z_s over $Q(a)$, it holds that $p_s(b, Y)$ has a root in F.

Proof. For each s, choose y_s so that $p_s(b, y_s) = 0$. We apply Corollary 3.9 to F and the subfield of F generated by $\{b, y_0, y_1, \ldots\}$, with the common subfield being $Q(a)$ (within F) and $Q(b)$ (within $Q(b, y_0, y_1, \ldots)$), identified via the isomorphism mapping a to b. Thus these two fields are isomorphic over the common subfield, so we have a field embedding of F into itself mapping a to b. By Corollary 3.7, this embedding is an automorphism of F.   □

If F is a normal algebraic extension of Q, then for any $a, b \in F$ which are conjugate over Q, both conditions in Proposition 6.3 hold. We state the obvious corollary.

Corollary 6.5. *All normal computable algebraic fields have computable orbit relation.*   □

7. Fields with Splitting Algorithms

Theorem 7.1. *Let F be a computable algebraic field with a splitting algorithm. Then F is computably categorical iff the orbit relation of F (or equivalently, the full orbit relation of F) is computable.*

Proof. We prove the forwards direction as Proposition 7.2 below. For the converse (which is the easier direction), we work under the weaker assumption that F has a computable presentation with computable full orbit relation and with computable level function, as in Definition 5.1. Of course, all computable algebraic fields with splitting algorithms have computable level functions, so this will suffice. Since computable categoricity is a property of the isomorphism type, we may assume F itself to be the computable presentation which has a computable level function l_F. We take the domain of F to be ω.

Suppose the full orbit relation A of F is computable, and let $\tilde{F} \cong F$ be a computable copy of F. We build a computable embedding $f : F \to \tilde{F}$ as follows. The prime subfield $F_0 = Q$ is a computable subfield of F (since F is algebraic) and

has a unique embedding f_0 into $\tilde{F}$, which is computable and extends to the given isomorphism θ_0 (not necessarily computable) from F onto $\tilde{F}$.

We now proceed by recursion on s. Given the embedding $f_s : F_s \hookrightarrow \tilde{F}$ which we have already built, consider the least element z of $(F - F_s)$, and let $F_{s+1} = F_s[z]$, on which we will define f_{s+1} to extend f_s. By induction, F_s must then be generated by the finite set $\{0, 1, \ldots, z-1\}$, and we also know inductively that f_s extends to some isomorphism θ_s from F onto $\tilde{F}$. The splitting algorithm for F_s lets us find the minimal polynomial $q(Z) \in F_s[Z]$ of z. Now the roots $z' \in F$ with $q(z') = 0$ correspond precisely to the nodes $\sigma \in I_F$ of the form $\langle 0, 1, \ldots, z-1, z' \rangle$, so let $d = l_F(\langle 0, 1, \ldots, z-1 \rangle)$ be the number of such roots. Then we can find all roots of $q(Z)$ in F: let them be $z = z_1 < z_2 < \cdots < z_d$. From the computable set A, we can determine exactly which tuples $\vec{x}_i = \langle 0, 1, \ldots, (z-1), z, 0, 1, \ldots, (z-1), z_i \rangle$ lie in A; these z_i are the true conjugates of z in F over F_s. For each $i \leq d$ with $\vec{x}_i \notin A$, we know that the node $\sigma_i = \langle 0, 1, \ldots, z-1, z_i \rangle$ is nonextendible in I_F, and so we use the function l_F to find a level which contains no successors of σ_i. (König's Lemma shows that such a level must exist.) Thus we compute a single level n such that no σ_i with $\vec{x}_i \notin A$ has any successor at level n in I_F. Then we turn to the isomorphism tree $I_{F\tilde{F}}$. θ_s shows that the node $\tau = \langle f_s(0), \ldots, f_s(z-1) \rangle$ is extendible in $I_{F\tilde{F}}$; indeed it is the image of $\langle 0, \ldots, z-1 \rangle$ under the isomorphism H_{θ_s} from Lemma 5.3. Moreover, from H_{θ_s} and our knowledge of I_F, we know that any immediate successor of this τ which extends to level n must be extendible. So we enumerate $I_{F\tilde{F}}$ until we find in it some node ρ at level n extending τ, and we define $f_{s+1}(z) = \rho(z)$. Now $F_{s+1} = F_s[z]$, so f_{s+1} is uniquely defined on F_{s+1}. Thus F_{s+1} is generated by $\{0, \ldots, z\}$, and moreover f_{s+1} extends to some isomorphism θ_{s+1} from F onto $\tilde{F}$, namely the (not necessarily unique) path through $\rho \restriction (z+1)$ in $I_{F\tilde{F}}$. These were all the inductive facts we needed in order to continue to the next stage.

This process computes f_{s+1}, uniformly in s, so $f = \cup_s f_s$ is a computable embedding of F into $\tilde{F}$. By Corollary 3.9, it must be an isomorphism. Thus F is computably categorical. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

To help the reader, we give a quick translation of the preceding construction into the language of fields, without using isomorphism trees. To find an image f_{s+1} for z in $\tilde{F}$, we find all conjugates of z over F_s in F, and use the computable set A_F to determine which are true conjugates. For each false conjugate z_i, we search for a polynomial $p_i \in F_s[Z, Y]$ which shows z_i to be false, namely, a polynomial such that $p_i(z, Y)$ has a root in F but $p_i(z_i, Y)$ does not. Proposition 6.3 shows that we will eventually find such a polynomial. (Of course, we are using the computability of S_F here to determine that $p_i(z_i, Y)$ has no root in F.) Using f_s, we then find the corresponding polynomials $\tilde{p}_i \in (f_s(F_s))[Z, Y]$, and search for any $\tilde{z} \in \tilde{F}$ which satisfies the (image in $(f_s(F_s))[X]$ of the) minimal polynomial of z over F_s and such that every $\tilde{p}_i(\tilde{z}, Y)$ has a root in $\tilde{F}$. We eventually must find such a $\tilde{z}$ (since $\theta_s(z)$ is such an element), and when we do, we know that it must be a true conjugate of $\theta_s(z)$, since a false conjugate $\tilde{z}'$ would be the image of some false conjugate z_i of z over F_s in F, and therefore $\tilde{p}_i(\tilde{z}', Y)$ would not have had a root in $\tilde{F}$. So it is safe for us to define $f_{s+1}(z) = \tilde{z}$, and θ_{s+1} is the composition of θ_s with an automorphism of $\tilde{F}$ sending $\theta_s(z)$ to its true conjugate $\tilde{z}$ over the image $f_s(F_s)$.

Proposition 7.2. *If an algebraic computable field F with a splitting algorithm is computably categorical, then the orbit relation B_F of F is computable.*

Proof. We will construct a computable field $\tilde{F}$, isomorphic to F, in such a way that the existence of any computable isomorphism from F onto $\tilde{F}$ will allow us to compute B_F. This is sufficient to prove the proposition. (It is also the contrapositive of the usual argument for this sort of theorem. In other contexts, mathematicians have often taken the given property – in this case, the noncomputability of B_F – and used it to build the second structure $\tilde{F}$ by a construction which diagonalizes against all possible computable isomorphisms. We believe that this would be feasible in the present case, and that the construction would not be substantially different from ours, but we see our argument as more direct.)

We start by enumerating a generating set for the given field F. Set F_0 to be the prime subfield of F, either $\mathbb{Q}$ or $\mathbb{F}_p$, with z_0 as the multiplicative identity element of F. Given F_s, choose the least number $y \in F - F_s$, and let F_{s+1} be the normal closure of $F_s \cup \{y\}$ within F. Since F has a splitting algorithm, this is computable: we can determine the minimal polynomial $p(X) \in F_0[X]$ of y over F_0 and then find all of its roots in F, and F_{s+1} is generated by these roots over F_s. Being finitely generated over F_s (and hence over F_0, by induction), F_{s+1} has a splitting algorithm, uniformly in its generators and hence uniformly in s, and thus is a decidable subset of the algebraic field F. We set z_{s+1} to be the least primitive generator of F_{s+1} over F_0, and list out all of its conjugates over F_0 in F, each of which is another primitive generator of F_{s+1}:

$$z_{s+1} = z_{s+1}^0 < z_{s+1}^1 < \cdots < z_{s+1}^{d_{s+1}}.$$

(The superscripts here are not exponents, of course, but merely indices.) Again, all of this is computable uniformly in s. Notice that

$$d_{s+1} + 1 = [F_{s+1} : F_0]$$

is precisely the size of the automorphism group of F_{s+1}; this is at most the degree of the minimal polynomial $p_{s+1}(X) \in F_0[X]$ of z_{s+1} over F_0, and in general is not equal to that degree, since F_{s+1} may fail to be normal over F_0. However, every automorphism σ of F_{s+1} must fix setwise (although not necessarily pointwise) every F_t with $t \leq s$, since each F_t is normal within F, hence normal within F_{s+1}. (In particular, $\sigma(z_t)$ must equal some F_0-conjugate of z_t, which is to say, some primitive generator z_t^i of F_t.)

The requirements for our construction are simply stated:

$$\mathcal{R}_e : \quad \varphi_e \text{ is not an isomorphism from } F \text{ onto } \tilde{F}.$$

If all these requirements (for all e) were true of the field $\tilde{F}$, then there would be no computable isomorphism from F onto $\tilde{F}$. Our construction of $\tilde{F}$ addresses these requirements individually, for each e, and attempts to satisfy each one. Requirements such as these are used throughout computability theory; the reader unfamiliar with them should consult [28]. Normally, to prove computable non-categoricity of F, one would build a computable field $\tilde{F}$ isomorphic to F for which every $\mathcal{R}_e$ holds. As we are proving the contrapositive, our F is assumed to be computably categorical, and so we will not succeed in satisfying all of these requirements, but our construction will attempt to do so nevertheless, using the indices e to assign priorities to each requirement, with a lower index denoting a higher priority. The least e for which we fail to satisfy $\mathcal{R}_e$ will be the key to our decision procedure for B_F.

Now we construct $\tilde{F}$, in stages, with each $\tilde{F}_s$ isomorphic to F_s. Of course $\tilde{F}_0 = F_0$ is just a computable copy of the prime subfield, and we fix

$$f_0 : F_0 \to \tilde{F}_0$$

to be the (unique) isomorphism between them. All requirements $\mathcal{R}_e$ are unsatisfied at stage 0. At the end of each stage s, we will have a field $\tilde{F}_s$ and an isomorphism f_s onto it from F_s, all computable uniformly in s. Moreover, for each $t \leq s$, we also know z_t and all z_t^i, for $i \leq d_t$. We proceed as follows to build $\tilde{F}_{s+1}$.

For each $t \leq s$ and $i \leq d_t$, we consider the automorphism σ_t^i of F_t defined by $\sigma_t^i(z_t) = z_t^i$; these are precisely the automorphisms of F_t (over F_0, which is rigid). We do the same for the automorphisms σ_{s+1}^k of F_{s+1}, defined by $\sigma_{s+1}^k(z_{s+1}) = z_{s+1}^k$, for $k \leq d_{s+1}$. Since all this is computable, we may decide, for each t and i, whether σ_i^t extends to an automorphism of F_s and/or to an automorphism of F_{s+1}. Certainly σ_t^0, the identity, extends to σ_{s+1}^0, but other σ_t^i may or may not extend to F_s, and those which do may or may not extend to F_{s+1}. However, by normality of F_s within F_{s+1}, σ_t^i can only extend to an automorphism of F_{s+1} if it extends to an automorphism of F_s.

We search for the least $e \leq s$, if any, for which $\mathcal{R}_e$ is not yet satisfied and there exists some $t \leq s$ and $j \leq d_t$ for which $\varphi_{e,s}(z_t^i) \downarrow$ for all $i \leq d_t$ and σ_t^j extends to an automorphism σ of F_s but does not extend to any automorphism of F_{s+1}. (Our conventions about $\varphi_{e,s}(z_t^i)$ and $W_{e,s}$ were described in Section 2.) If there is no such e, then we define $f_{s+1}{\restriction}F_s = f_s$ and let $\tilde{F}_{s+1}$ contain all of $\tilde{F}_s$, along with fresh elements to be the images of the elements of $(F_{s+1} - F_s)$ under f_{s+1}.

If such an e does exist, then we act to satisfy $\mathcal{R}_e$, using the value j and the automorphism σ of F_s found above. First, though, for all of the (finitely many) elements $x \in W_{e,s} \cap F_t$, we find a polynomial $q(Z) \in F_0[Z]$ with $q(z_t) = x$ and check whether $q(\varphi_e(z_t)) = \varphi_e(x)$. If this fails for any x, then φ_e cannot be an isomorphism, so we act just as we did (above) when e did not exist, and declare $\mathcal{R}_e$ satisfied. Otherwise, we know that $\varphi_{e,s}$ maps the set $\{z_t^0, \ldots, z_t^{d_t}\}$ bijectively onto the set $\{f_s(z_t^0), \ldots, f_s(z_t^{d_t})\}$. Fix the m such that $f_s(z_t^m) = \varphi_e(z_t^0)$. If no automorphism of F_s maps z_t^0 to z_t^m, then $(f_s^{-1} \circ \varphi_e)$ cannot be an automorphism, so φ_e cannot be an isomorphism, and once again we just extend f_s to f_{s+1}, fill in $\tilde{F}_{s+1}$ with fresh elements, and declare $\mathcal{R}_e$ satisfied. Otherwise, there is an automorphism τ of F_s with $\tau(z_t^0) = z_t^m$, and we let $f_{s+1}{\restriction}F_s = f_s \circ \tau \circ \sigma^{-1}$, and form $\tilde{F}_{s+1}$ by adding fresh elements to $\tilde{F}_s$ to be the images of the elements of F_{s+1} under f_{s+1}. Lemma 7.3 below will show that in this case, φ_e cannot be an isomorphism from F onto $\tilde{F}$. So we declare $\mathcal{R}_e$ satisfied, and end the stage.

This builds a computable field $\tilde{F}$, which we claim is isomorphic to F. Of course, we made no attempt during the construction to ensure that $\lim_s f_s(x)$ must exist for $x \in F$. However, every finitely generated subfield of F embeds into some F_s and hence (via f_s) into $\tilde{F}_s$, whence into $\tilde{F}$. A symmetric argument with $\tilde{F}$ and F interchanged also holds, leaving Corollary 3.9 to prove that $F \cong \tilde{F}$ over F_0. Of course, the isomorphism need not be computable.

Lemma 7.3. *If $\mathcal{R}_e$ is ever declared satisfied during this construction, then φ_e is not an isomorphism from F onto $\tilde{F}$.*

Proof. Let $s + 1$ be the stage at which $\mathcal{R}_e$ is declared satisfied. If φ_e were an isomorphism from F onto $\tilde{F}$, then it would have to restrict to an isomorphism from

F_{s+1} onto $\tilde{F}_{s+1}$, because F_{s+1} is normal within F and $\tilde{F}_{s+1}$ is the image of F_{s+1} under the isomorphism f. Similarly, $\varphi_e{\restriction}F_s$ would have to be an isomorphism. To prove the lemma, therefore, we will show that $\varphi_e{\restriction}F_s$ and $\varphi_e{\restriction}F_{s+1}$ cannot both be isomorphisms.

First, if any $\varphi_{e,s}(x)$ converged to a value other than a root of $q(\varphi_e(z_t))$ (where $q(Z)$ is the minimal polynomial of x over F_0, as in the construction), then clearly $\varphi_e{\restriction}F_s$ is not an isomorphism. Also, we chose m to satisfy $\varphi_e(z_t^0) = f_s(z_t^m)$, so if no automorphism of F_s maps z_t^0 to z_t^m, then $(f_s^{-1}{\circ}\varphi_e){\restriction}F_s$ cannot be an automorphism, and $\varphi_e{\restriction}F_e$ cannot be an isomorphism

In the remaining case, σ_s^j is known to extend to an automorphism σ of F_s, but not to any automorphism of F_{s+1}, and we have an automorphism τ of F_s with $\tau(z_t^0) = z_t^m$. In this case

$$f_{s+1}(z_t^j) = (f_s \circ \tau \circ (\sigma^{-1}))(z_t^j) = f_s(z_t^m) = \varphi_e(z_t^0).$$

With the new elements added to form $\tilde{F}_{s+1}$, this means that φ_e cannot restrict to an isomorphism from F_{s+1} onto $\tilde{F}_{s+1}$, because if it were, then $(f_{s+1}^{-1} \circ \varphi_e)$ would be an automorphism of F_{s+1} mapping z_t^0 to z_t^j, which would necessarily extend σ_s^j, and we chose j precisely so that no such extension of σ_s^j exists. $\square$

For the sake of readers who saw "$\mathcal{R}_e$" and expected an injury construction, we note that no conflict exists between the different requirements here: Lemma 7.3 shows that there was no need to preserve the satisfaction of $\mathcal{R}_e$ once it was established. The normality of each F_{s+1} within F took care of that. Moreover, the above argument did not require that $\lim_s f_s$ itself exist, let alone that it be an isomorphism, and so, when satisfying a requirement, the construction makes no effort to have f_{s+1} agree with f_s at all. Therefore, there are no injuries in this construction. The only need for priority arose in choosing which $\mathcal{R}_e$ to satisfy, at a stage at which more than one requirement might have been satisfiable, and the only reason for taking the least e in those situations was that this is the simplest way to ensure that each requirement which can be satisfied at infinitely many stages does indeed eventually become satisfied.

Of course, it remains to show that this field $\tilde{F}$ really does prove the desired result. We have seen that $\tilde{F} \cong F$, and by the computable categoricity of F, this means that there exists a computable isomorphism φ_e from F onto $\tilde{F}$. We fix the least such index e and the largest stage s_0 at which a requirement $\mathcal{R}_i$ with $i < e$ was satisfied. By Lemma 7.3, $\mathcal{R}_e$ was never satisfied, so at all stages $s > s_0$, it did not fulfill the conditions which the construction posed in order to be satisfied.

We now describe an algorithm for deciding whether a pair $\langle z_t, z_t^n \rangle$, with t and $n \le d_t$ arbitrary, lies in the orbit relation B_F of F. We may assume that $t > s_0$, since the answers for all $t \le s_0$ and $n \le d_t$ constitute finitely much information. Find the least s_1 such that $\varphi_{e,s_1}(z_t^i){\downarrow}$ for all $i \le d_t$. Now $z_t^0 \ge t-1$ (since $z_t^0 \notin F_{t-1}$ and, by construction, each subfield F_s contains the elements $0, \ldots, s-1$), and hence $\varphi_e(z_t)$ requires at least $t-1$ steps to converge, forcing $s_1 \ge t-1 \ge s_0$. As above, we may compute all automorphisms of F_{s_1} and check whether any of them maps z_t to z_t^n. If not, then clearly $\langle z_t, z_t^n \rangle \notin B_F$, since any automorphism mapping z_t to z_t^n would restrict to an automorphism of the subfield F_{s_1}, it being normal within F. We claim that if some automorphism ρ of F_{s_1} does have $\rho(z_t) = z_t^n$, then $\langle z_t, z_t^n \rangle \in B_F$.

To see this, we induct on stages $s \geq s_1$, claiming that σ_t^n extends to an automorphism of F_s for all such s. For s_1 this already holds, since

$$\rho(z_t) = z_t^n = \sigma_t^n(z_t)$$

and z_t generates the domain F_s of σ_t^n. So we consider an arbitrary $s + 1 > s_1$. Now since $\mathcal{R}_e$ does *not* become satisfied at stage $s+1$ (and no $e' < e$ ever becomes satisfied after stage s_0), e must not fulfill the conditions in the construction for choosing the requirement to be satisfied at stage $s + 1$. But with $s \geq s_1$, $\varphi_{e,s}(z_t^i)\downarrow$ for all $i \leq d_t$, and σ_t^n extends (by inductive hypothesis) to an automorphism of F_s. If σ_t^n failed to extend to an automorphism of F_{s+1}, then the construction would have chosen e and acted to satisfy $\mathcal{R}_e$ at this stage, destroying the isomorphism φ_e. This did not happen, so σ_t^n must extend to an automorphism of F_{s+1}. This completes the induction.

The extension of σ_t^n to an automorphism τ of the whole field F is now accomplished by application to Corollary 3.9. We let $Q = F_t$, as a subfield of F, and have $E = F$. The subfield Q of E is in fact F_t as well, but we identify the two copies of F_t via σ_t^n, rather than via the identity. Corollary 3.9 then yields an isomorphism between F and E over the common subfield, i.e. an automorphism of F extending σ_t^n, as desired.

Finally, for an arbitrary pair $\langle a, b \rangle$ of elements of F, we find an s with $a, b \in F_s$ and determine all automorphisms (if any) of F_s mapping a to b. Each of these automorphisms is equal to σ_s^i for some $i \leq d_s$, and so for each such i, we check whether $\langle z_s, z_s^i \rangle \in B_F$. If this holds for any $i \leq d_s$, then also $\langle a, b \rangle \in B_F$, as witnessed by the automorphism(s) of F extending σ_s^i. Conversely, any automorphism of F mapping a to b would restrict to an automorphism of F_s, which would then equal σ_s^i for one of these i. Thus B_F is computable. $\square$

8. Relativizing the Results

For simplicity, we proved Theorem 7.1 above in a non-relativized form. However, the argument in one direction relativizes easily to any degree $\boldsymbol{d}$, producing the following result. Recall that a computable structure $\mathfrak{A}$ is $\boldsymbol{d}$-*computably categorical* if every computable structure $\mathfrak{B}$ classically isomorphic to $\mathfrak{A}$ is $\boldsymbol{d}$-computably isomoprhic to $\mathfrak{A}$.

Proposition 8.1. *Let F be a computable algebraic field with a splitting algorithm, and fix any Turing degree $\boldsymbol{d}$. If the orbit relation of F (or equivalently, the full orbit relation of F) is $\boldsymbol{d}$-computable, then F is $\boldsymbol{d}$-computably categorical.* $\square$

The proof of Proposition 7.2, however, relativizes to the statement that, for each computable algebraic field F with a splitting algorithm, there exists a $\boldsymbol{d}$-computable field $\tilde{F} \cong F$ such that any $\boldsymbol{d}$-computable isomorphism from F onto $\tilde{F}$ would allow one to compute B_F. This is not sufficient to prove the converse of Proposition 8.1, and in fact the converse turns out to be false, by the following argument.

By [20], there exists a computable algebraic field F, with a splitting algorithm, such that there is no least Turing degree $\boldsymbol{d}$ for which F is $\boldsymbol{d}$-computably categorical. Indeed, it is proven there that there exist Turing degrees $\boldsymbol{c}$ and $\boldsymbol{d}$ whose infimum is $\boldsymbol{0}$, such that F is both $\boldsymbol{c}$-computably categorical and $\boldsymbol{d}$-computably categorical. If the orbit relation B_F for this field were both $\boldsymbol{c}$-computable and $\boldsymbol{d}$-computable, then it would be computable, and F would have been computably categorical. Consequently, one of these degrees (say $\boldsymbol{d}$) has the property that $\deg(B_F) \not\leq_T \boldsymbol{d}$, even

though F is $\boldsymbol{d}$-computably categorical. So the converse of Proposition 8.1 fails. It would be of interest to determine whether perhaps there exists a computable field E isomorphic to this F, for which $\boldsymbol{d}$ does compute B_E (in which case necessarily $\boldsymbol{c}$ would not compute B_E, by the argument above). If so, then the Turing degree of the orbit relation would not be invariant under isomorphisms between computable algebraic fields with splitting algorithms, even though (by Theorem 7.1) the decidability of the orbit relation is invariant.

9. Further Notes

The article [11] is complementary to this one in a number of respects. It considers relative computable categoricity for computable algebraic fields, and also examines the possible computable dimensions of such fields. Its most relevant results for us, however, are negative ones: it is shown in [11, Theorem 4.5] that there exists a computable algebraic field with computable orbit relation which is not computably categorical, and it is shown in [11, Theorem 5.1] that there exists a computably categorical algebraic field F such that B_F is not even Σ_2^0, let alone computable. One might have hoped for Proposition 7.2 to generalize to all computable algebraic fields; alternatively, one might have rephrased Theorem 7.1 to say that computable categoricity is equivalent to computable enumerability of B_F (which is exactly the content of the proof, B_F being Π_1^0 for any computable field with a splitting algorithm). The results in [11] dash these hopes, establishing that both directions of the desired equivalence are false.

It is noted in [11] that the definition of computable categoricity, which is normally of complexity Π_1^1, drops to complexity Π_4^0 when one restricts the discussion to computable algebraic fields. Essentially this follows from Corollary 3.9 above (with Q as the prime subfield), which reduces the complexity of the isomorphism relation on such fields dramatically. Moreover, [11, Theorem 6.4] proves that for algebraic fields, computable categoricity is Π_4^0-complete, and thus is quantifiably more difficult than computable categoricity for algebraic fields with splitting algorithms. This re-establishes the negative results from the preceding paragraph. Also, the proof in [11] that computable categoricity does not imply relative computable categoricity is already of interest, since to our knowledge, all previous results giving structural criteria for computable categoricity in commonplace mathematical classes also implied relative computable categoricity. So there is concrete evidence that fields, and even just algebraic fields, constitute a more challenging class of structures for this question than did the previous classes studied.

References

1. C.J. Ash, J.F. Knight, M.S. Manasse, & T.A. Slaman; Generic copies of countable structures, *Annals of Pure and Applied Logic* **42** (1989), 195–205.
2. J. Chisholm; On intrisically 1-computable trees, unpublished MS.
3. J.P. Cleave; Some properties of recursively inseparable sets, *Zeits. Math. Logik Grund. Math.* **16** (1970) 2, 187–200.
4. Yu.L. Ershov; Theorie der Numerierungen, *Zeits. Math. Logik Grund. Math.* **23** (1977), 289–371.
5. E. Fokina, I. Kalimullin, & R.G. Miller; Degrees of categoricity of computable structures, *Archive for Mathematical Logic* **49** (2010) 1, 51–67.
6. M.D. Fried & M. Jarden, *Field Arithmetic* (Berlin: Springer-Verlag, 2005).
7. A. Frohlich & J.C. Shepherdson; Effective procedures in field theory, *Phil. Trans. Royal Soc. London, Series A* **248** (1956) 950, 407–432.

8. A. Frolov, I. Kalimullin, & R.G. Miller; Spectra of algebraic fields and subfields, *Mathematical Theory and Computational Practice: Fifth Conference on Computability in Europe, CiE 2009*, eds. K. Ambos-Spies, B. Löwe, & W. Merkle, *Lecture Notes in Computer Science* **5635** (Berlin: Springer-Verlag, 2009), 232–241. (For Appendix A, see `qcpages.qc.cuny.edu/~rmiller/research.html`.)

9. S.S. Goncharov; Autostability and computable families of constructivizations, *Algebra and Logic* **14** (1975), 647–680 (Russian), 392–409 (English translation).

10. S.S. Goncharov & V.D. Dzgoev; Autostability of models, *Algebra and Logic* **19** (1980), 45–58 (Russian), 28–37 (English translation).

11. D.R. Hirschfeldt, K. Kramer, R.G. Miller, & A. Shlapentokh; Categoricity properties for computable algebraic fields, to appear in the *Transactions of the American Mathematical Society*.

12. C.G. Jockusch & R.I. Soare; Π_1^0-classes and degrees of theories, *Transactions of the American Mathematical Society* **173** (1972), 33–56.

13. N. Kogabaev, O. Kudinov, & R.G. Miller; The computable dimension of I-trees of infinite height, *Algebra and Logic* **43** (2004) 6, 393–407.

14. L. Kronecker; Grundzüge einer arithmetischen Theorie der algebraischen Größen, *J. f. Math.* **92** (1882), 1–122.

15. O.V. Kudinov; An autostable 1-decidable model without a computable Scott family of $\exists$ formulas, *Algebra and Logic* **35** (1996), 255–260 (English translation).

16. S. Lempp, C. McCoy, R.G. Miller, & R. Solomon; Computable categoricity of trees of finite height, *Journal of Symbolic Logic* **70** (2005), 151–215.

17. G. Metakides & A. Nerode; Effective content of field theory, *Annals of Mathematical Logic* **17** (1979), 289–320.

18. R.G. Miller; The computable dimension of trees of infinite height, *Journal of Symbolic Logic* **70** (2005), 111–141.

19. R.G. Miller, Computable fields and Galois theory, *Notices of the American Mathematical Society* **55** (August 2008) 7, 798–807.

20. R.G. Miller; $\boldsymbol{d}$-Computable categoricity for algebraic fields, *The Journal of Symbolic Logic* **74** (2009) 4, 1325–1351.

21. R.G. Miller, Computability and differential fields: a tutorial, to appear in *Differential Algebra and Related Topics: Proceedings of the Second International Workshop*, eds. L. Guo & W. Sit (World Scientific, 2011), ISBN 978-981-283-371-6. Also available at `qcpages.qc.cuny.edu/~rmiller/research.html`.

22. R.G. Miller & H. Schoutens; Computably categorical fields via Fermat's Last Theorem, to appear in *Computability*.

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

24. J.B. Remmel; Recursively categorical linear orderings, *Proceedings of the American Mathematical Society* **83** (1981), 387–391.

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

26. D. Scott; Algebras of sets binumerable in complete extensions of arithmetic, in *Recursive Function Theory*, vol. 5 of *Proceedings of Symposia in Pure Mathematics* (Providence: American Mathematical Society, 1962), 117–121.

27. S. Simpson; Degrees of unsolvability: a survey of results, in *Handbook of Mathematical Logic*, ed. J. Barwise (Amsterdam: North-Holland, 1977), 631–652.

28. R.I. Soare; *Recursively Enumerable Sets and Degrees* (New York: Springer-Verlag, 1987).

29. V. Stoltenberg-Hansen & J.V. Tucker; Computable rings and fields, in *Handbook of Computability Theory*, ed. E.R. Griffor (Amsterdam: Elsevier, 1999), 363–447.

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

Department of Mathematics, Queens College – C.U.N.Y., 65-30 Kissena Blvd., Flushing, New York 11367 U.S.A.; Ph.D. Programs in Mathematics and Computer Science, C.U.N.Y. Graduate Center, 365 Fifth Avenue, New York, NY 10016 U.S.A.
   E-mail address: `Russell.Miller@qc.cuny.edu`

East Carolina University, Department of Mathematics, Greenville, NC 27858 U.S.A.
   E-mail address: `shlapentokha@ecu.edu`