Computable Categoricity of Trees of Finite Height

Steffen Lempp
Department of Mathematics
University of Wisconsin-Madison

Charles McCoy
Department of Mathematics
University of Wisconsin-Madison

Russell Miller
Department of Mathematics
Queens College – C.U.N.Y.

Reed Solomon
Department of Mathematics
University of Connecticut*

August 3, 2004

Abstract

We characterize the structure of computably categorical trees of finite height, and prove that our criterion is both necessary and sufficient. Intuitively, the characterization is easiest to express in terms of isomorphisms of (possibly infinite) trees, but in fact it is equivalent to a Σ_3^0-condition. We show that all trees which are not computably categorical have computable dimension ω. Finally, we prove that for every $n \geq 1$ in ω, there exists a computable tree of finite height which is Δ_{n+1}^0-categorical but not Δ_n^0-categorical.

*The first author was partially supported by NSF grant DMS-9732526 and by the Vilas Foundation of the University of Wisconsin. The second author was partially supported by a VIGRE grant to the University of Wisconsin. The third author was partially supported by a VIGRE postdoc under NSF grant number 9983660 to Cornell University. The fourth author was partially supported by an NSF postdoctoral fellowship.

1 Introduction

Computability theorists have developed powerful techniques for studying computational properties of the natural numbers. Many of these techniques can be applied to more general algebraic structures once they are suitably coded into the natural numbers. In this article, we use tools from computability theory to study computational problems for trees of finite height.

We begin with some general definitions and background in computable model theory. Let $\mathcal{A}$ be a countable structure over a fixed computable language whose domain $|\mathcal{A}|$ is a subset of ω. The *degree* of $\mathcal{A}$ is the Turing degree of the atomic diagram of $(\mathcal{A}, a)_{a \in |\mathcal{A}|}$. In particular, if the language is finite, then $\mathcal{A}$ is computable if $|\mathcal{A}|$ is a computable set and the interpretations of the function and relation symbols are all computable. Throughout this paper, we assume that all structures are coded into the natural numbers.

In computable model theory, one frequently works in a given class of countable algebraic structures such as abelian groups, partial orders, fields, or as in this paper, finite height trees. Any computable structure from one of these classes is isomorphic to infinitely many other computable structures. It may happen, however, that two computable structures are isomorphic, yet that the only isomorphisms between them are noncomputable (as maps from one domain to the other). If so, then these structures lie in distinct *computable isomorphism classes* of the isomorphism type of the structure. On the other hand, if there exists a computable function taking one structure isomorphically to the other, then the two structures lie in the same computable isomorphism class.

The *computable dimension* of a computable structure is the number of computable isomorphism classes of that structure. The most common computable dimensions are 1 and ω, and many classes of algebraic structures are known to admit only these computable dimensions. The following theorem is a compilation of results due to Goncharov ([12]); Goncharov, Dzgoev ([13]); Goncharov, Lempp, Solomon ([14]); LaRoche ([22]); Metakides, Nerode ([23]); Nurtazin ([28]); and Remmel ([29], [30]).

Theorem 1.1 *Computable structures in the following classes have computable dimension* 1 *or* ω*: algebraically closed fields, real closed fields, abelian groups, linear orders, Boolean algebras, and ordered abelian groups.*

On the other hand, Goncharov ([11]) proved that for each $0 < n \leq \omega$, there exist structures with computable dimension n. Since then, many classes of structures have been discovered which admit computable dimension n for each $0 < n \leq \omega$. The following examples come from Goncharov ([11]); Goncharov, Molokov, Romanovskii ([15]); Hirschfeldt, Khoussainov, Shore, Slinko ([16]); and Kudinov ([21]).

Theorem 1.2 *For each* $0 < n \leq \omega$*, there are computable structures in the following classes with computable dimension* n*: graphs, lattices, partial orders, nilpotent groups, and integral domains.*

There are many other natural computational questions that one can ask about the members of these algebraic classes. For example, is it possible for the computable dimension of $\mathcal{A}$ to change when a single constant is named? What are the possible degree spectra for a structure or for a relation on a structure within each class? The *degree spectrum* of $\mathcal{A}$ is the set of Turing degrees $\mathbf{d}$ for which there is an isomorphic copy of $\mathcal{A}$ of degree $\mathbf{d}$. The *degree spectrum of a relation U* on $\mathcal{A}$ is the set of degrees $\mathbf{d}$ such that there is an isomorphic computable copy of $\mathcal{A}$ for which the image of U in this copy has degree $\mathbf{d}$.

Hirschfeldt, Khoussainov, Shore and Slinko ([16]) gave highly effective coding methods which show that for the classes of structures from Theorem 1.2, any answer to the above questions which can occur in a countable model, can actually occur within these classes. More specifically, they show that for each of these classes and for each nontrivial countable structure $\mathcal{M}$, there is a structure $\mathcal{A}$ from that class such that

1. the degree spectrum of $\mathcal{M}$ is equal to the degree spectrum of $\mathcal{A}$,

2. the computable dimension of $\mathcal{M}$ is the same as the computable dimension of $\mathcal{A}$,

3. for each $x \in |\mathcal{M}|$, there is an $a \in |\mathcal{A}|$ such that $(\mathcal{M}, x)$ has the same computable dimension as $(\mathcal{A}, a)$,

4. for each $S \subset |\mathcal{M}|$, there is a $U \subset |\mathcal{A}|$ such that the degree spectrum of the relation S with respect to $\mathcal{M}$ is the same as the degree spectrum of U with respect to $\mathcal{A}$.

These results suggest that the algebraic structure on the members of these classes interacts in a trivial way with the computational structure in the sense that any "pathological" computational behavior which can occur in a countable model can actually occur within these classes of structures. For example, Slaman ([32]) and Wehner ([34]) independently proved that there is a computable model $\mathcal{M}$ whose degree spectrum contains all degrees except $\mathbf{0}$. Therefore, by the result above, there are graphs, lattices, and so on with this property.

Because Property (2) above fails for the classes in Theorem 1.1, the interaction between the algebraic structure on the members of these classes and their computational structure is nontrivial in the sense that the algebraic structure necessarily limits the types of computational behavior that can occur. It is therefore interesting to ask about how the algebraic structure and the computational properties interact. For example, Downey and Jockusch ([7]) showed that every low Boolean algebra has a computable copy. Therefore, it is not possible for a Boolean algebra to have a degree spectrum consisting of all degrees except $\mathbf{0}$. On the other hand, by Miller ([25]), there is a linear order which has copies in every Δ^0_2 degree except $\mathbf{0}$. The question of whether a linear order can have a spectrum consisting of all degrees except $\mathbf{0}$ remains open. The reader is referred to [16] for a more detailed survey of similar results.

We would hope for a fine line separating the classes of structures which behave as in Theorem 1.1 and those which behave as in Theorem 1.2. In the class of groups, the fact that abelian groups fall in Theorem 1.1 and nilpotent groups fall in Theorem 1.2 gives a reasonably sharp distinction. To sharpen the difference further, we could weaken nilpotent groups to torsion free nilpotent groups (that is, nilpotent groups in which no element except the identity has finite order) and we could add structure to the abelian groups by making them ordered. In both cases, the classes retain their previous possible computable dimensions.

For ring structures and ordered structures, the story is quite different. There is a large gap between algebraically or real closed fields (Theorem 1.1) and integral domains (Theorem 1.2). The obvious open question is what are the possible computable dimensions for computable fields. For ordered structures, there is a gap between linear orderings and Boolean algebras (Theorem 1.1) and lattices and partial orderings (Theorem 1.2). Trees are one obvious class of structures which falls within this gap and therefore they are of particular interest. It is not immediately apparent whether one would expect trees to admit only limited coding, like linear orders, or to admit very general coding, like partial orders, with respect to computable dimension and the other properties mentioned above. Our main result says that with respect to computable dimension, they have limited behavior in that they must have computable dimension 1 or ω. It would therefore be interesting to explore the answers to the other computational questions for trees.

In addition to proving that finite height trees must have dimension 1 or ω, we give an algebraic characterization for when they have computable dimension 1. If the computable dimension of $\mathcal{A}$ is 1, we say that $\mathcal{A}$ is *computably categorical.* This notion is somewhat analogous to the concept of categoricity in ordinary model theory: a theory is *categorical* in a given power κ if all models of the theory of power κ are isomorphic. Computable categoricity, however, is a property of structures, not of theories: a computable structure $\mathcal{A}$ is computably categorical if every other computable structure which is isomorphic to $\mathcal{A}$ is computably isomorphic to $\mathcal{A}$.

A standard example of a categorical theory is the theory of dense linear orders without end points, which is categorical in power ω. One proves this by taking two arbitrary countable dense linear orders and building an isomorphism between them by a back-and-forth construction. The same construction allows us to prove that the structure $\mathbb{Q}$ is computably categorical. (More formally, let $(\omega, \prec)$ be a computable linear order isomorphic to $(\mathbb{Q}, <)$. Then $(\omega, \prec)$ is computably categorical.)

Characterizations of computable categoricity have been found for several types of structures. The following examples comes from Goncharov, Dzgoev ([13]); Goncharov, Lempp, Solomon ([14]); and Remmel ([29], [30]).

Theorem 1.3 *The following equivalences for computable categoricity hold.*

1. *A computable linear order is computably categorical if and only if it has a finite number of pairs of adjacent elements.*
2. *A computable Boolean algebra is computably categorical if and only if it has a finite number of atoms.*
3. *A computable ordered abelian group is computably categorical if and only if it has finite rank.*

There are a number of natural generalizations for computable categoricity, two of which are important for this article. A computable structure $\mathcal{A}$ is *relatively computably categorical* if for every isomorphic (not necessarily computable) copy $\mathcal{B}$, there is an isomorphism between $\mathcal{A}$ and $\mathcal{B}$ which is computable from the degree of $\mathcal{B}$. It follows directly from this definition that any structure which is relatively computably categorical is also computably categorical. For linear orders and Boolean algebras, the notions of computable categoricity and relative computable categoricity coincide. However, this is not always the case. In general, computable categoricity does not imply relative computable categoricity without addition assumptions on the structures involved.

Theorem 1.4 (Kudinov ([20])) *There is a computable $\mathcal{A}$ for which the Π^0_1 diagram is decidable and which is computably categorical but not relatively so.*

By the following result of Goncharov, Kudinov's example is the best possible in terms of decidable fragments.

Theorem 1.5 (Goncharov ([8])) *Let $\mathcal{A}$ be a computable structure for which the Π^0_2 diagram is decidable. Then, $\mathcal{A}$ is relatively computably categorical if and only if it is computably categorical.*

In the present paper we consider computable trees of finite height, and develop a structural criterion for such trees which is equivalent to both computable categoricity and relative computable categoricity. There are a number of definitions for trees, but for our purposes, a tree consists of a universe T with a strict partial order $\prec$ on T such that for every $x \in T$, $\prec$ well-orders the set of $\prec$-predecessors of x in T, and such that T contains a least element under $\prec$ called the root. We view our trees as growing upward with the root r at the base. A tree is computable if T is a computable set and $\prec$ is a computable relation. Without loss of generality, we can restrict ourselves to trees whose domain is an initial segment of ω. An *index* for T is then an index for the characteristic function of $\prec$.

Because we are only concerned with trees of finite height, we can define the level of a node $x \in T$ by

$$\text{level}_T(x) = |\{y \in T : y \prec x\}|.$$

A more formal definition sets the level of the root to be 0 and inductively defines $\text{level}_T(x) = \sup\{\text{level}_T(y) + 1 : y \prec x\}$, thereby also covering the case of an element with infinitely many predecessors. The level of a node is not generally computable, but it can be approximated from below by the computable function $f(x, s) = |\{y < s : y \prec x\}|$ which is increasing in the variable s and which has the property that for all $x \in T$, $\text{level}_T(x) = \lim_s f(x, s)$. The height of T is defined by

$$\text{ht}(T) = \sup_{x \in T}(\text{level}_T(x) + 1).$$

A *path* γ through T is a maximal linearly ordered subset of T. Thus, for a finite height tree, the height of T is the greatest n such that T contains a path with n elements. In a tree of finite height n, we say that a node is *established* if it lies on a path of length n, and (for computable trees) *established at stage s* if it lies on a path of length n which is contained in the approximation T_s at stage s. (If a node is established at stage s, then at that stage we know what its level in T must be, since no more predecessors of the node can appear at later stages.)

We define our structural criterion for computable categoricity by induction on the height of the tree.

Definition 1.6 Let $(T, \prec)$ be a tree of finite height, and x a node of T, with immediate successors $\{x_i : i \in I\}$. Let $T[x_i] = \{y \in T : x_i \preceq y\}$. We say that x is *of strongly finite type* if it satisfies the following conditions:

i. There are only finitely many isomorphism types in the set $\{T[x_i] : i \in I\}$, each of which is of strongly finite type; and

ii. For each j and k in I, if $T[x_j]$ embeds into $T[x_k]$, then either $T[x_j]$ and $T[x_k]$ are isomorphic, or the isomorphism type of $T[x_k]$ appears only finitely often in $\{T[x_i] : i \in I\}$.

T itself is of strongly finite type if every node in T is of strongly finite type, or equivalently, if the root node is of strongly finite type. (By part (i), it is also equivalent to require that every minimal ω-branch point in T be of strongly finite type.)

Notice that $\text{ht}(T[x_i]) < \text{ht}(T)$ for every $i \in I$, so that the concept is well-defined for every tree of finite height. Also, finite trees are automatically of strongly finite type, having no ω-branch points. We also have a weaker criterion.

Definition 1.7 Using the same notation, we say that x is *of finite type* if it satisfies:

i. There are only finitely many isomorphism types in the set $\{T[x_i] : i \in I\}$, each of which is of finite type; and

ii. Every isomorphism type which appears infinitely often in the set $\{T[x_i] : i \in I\}$ is of strongly finite type; and

iii. For each j and k in I, if $T[x_j]$ embeds into $T[x_k]$, then either $T[x_j]$ and $T[x_k]$ are isomorphic, or the isomorphism type of $T[x_j]$ appears only finitely often in $\{T[x_i] : i \in I\}$, or the isomorphism type of $T[x_k]$ appears only finitely often in $\{T[x_i] : i \in I\}$.

T itself is of finite type if every node in T is of finite type. Again, this is equivalent to every minimal ω-branch point being of finite type.

We can now state our main theorem.

Theorem 1.8 *For a computable tree $(T, \prec)$ of finite height, the following are equivalent:*

1. T is of finite type;

2. T is computably categorical;

3. T has finite computable dimension;

4. T is relatively computably categorical.

The proof of Theorem 1.8 is contained in Sections 2, 3 and 4. In Section 2, we show that finite type implies relative computable categoricity, which in turn implies computable categoricity. In Sections 3 and 4, we show that any tree which is not of finite type cannot be computably categorical, which in turn implies that it is not relatively computably categorical. The proof proceeds by induction, with Section 3 containing the base case and Section 4 containing the inductive argument. This establishes the equivalence of conditions 1, 2 and 4. We also show, in Section 4, that if a computable tree does not have finite type, then it must have infinite computable dimension. This establishes the equivalence of condition 3 with the other conditions.

To our knowledge, this is the first example of a structural criterion for computable categoricity which needs to be defined by recursion. Notice, however, that this criterion only applies to trees of finite height. The following result handles the case of infinite height trees.

Theorem 1.9 (Miller ([24])) *The computable dimension of every computable tree of infinite height is ω (regardless of whether or not the tree has ω-branch points).*

Together, Theorems 1.8 and 1.9 show that trees, like linear orders, cannot exhibit the behavior of the structures listed in Theorem 1.2. Chisholm ([4]) has some related unpublished work for intrinsically 1-computable trees. A computable tree T is 1-computable if its Σ^0_1 diagram is computable and T is intrinsically 1-computable if every computable copy of T is 1-computable. Chisholm proved that every intrinsically 1-computable tree is computably categorical. Notice, however, that intrinsic 1-computability is a strong assumption for trees, as Chisholm also showed that every intrinsically 1-computable tree is intrinsically decidable. (That is, every computable copy of the tree is decidable.)

Once we know that there are computable trees of finite height with infinite computable dimension, it is natural to ask exactly how difficult it is to compute an isomorphism between arbitrary pairs of such trees. (For computable trees without the restriction to finite height, the isomorphism problem is Σ^1_1-complete as computable trees can be used to code arbitrary computable ordinals.) In Section 5 we begin to answer this question for finite height trees by showing that no degree $\mathbf{0^{(n)}}$ is capable of computing an isomorphism between every pair of isomorphic computable trees of finite height. More specifically, a computable structure $\mathcal{A}$ is called Δ^0_n-categorical if for every computable $\mathcal{B}$ isomorphic to $\mathcal{A}$, there is a Δ^0_n-isomorphism from $\mathcal{A}$ to $\mathcal{B}$. In this notation, Δ^0_1-categoricity is equivalent to computable categoricity. In Section 5, we show the following theorem.

Theorem 1.10 *For every $n \geq 1$ there is a computable tree of finite height which is Δ^0_{n+1}-categorical but not Δ^0_n-categorical.*

Another natural question to ask is how difficult it is to express the property "T is computably categorical". On its face, our structural criterion is an analytic predicate of similar complexity to stating the definition of computable categoricity. However, Ash, Knight, Mannasse and Slaman ([3]) showed that for any computable language $\mathcal{L}$, a computable $\mathcal{L}$-structure is relatively computably categorical if and only if it has a formally Σ^0_1 Scott family. In Section 2, we show that computable trees of finite height and finite type have formally Σ^0_1 Scott families consisting of finitary formulas. (In fact, any formally Σ^0_1 Scott family can be transformed into one consisting of finitary formulas.) It is known (see Proposition 6.10 in Ash, Knight ([2])) that the existence of such families is described by a Σ^0_3 condition. Therefore, since computable categoricity and relative computable categoricity coincide for trees of finite height, there is a Σ^0_3 predicate which expresses "T is computably categorical". Theories are known to exist in which the property of computable categoricity is strictly more complex than Σ^0_3; we refer the reader to [35] for details.

It is important to note that our definition of tree is based on a partial order $\prec$. In other papers, "tree" is sometimes defined using the infimum function $\wedge$, where the infimum $x \wedge y$ of x and y is the $\prec$-maximal z such that $z \preceq x$ and $z \preceq y$. One can define $\prec$ from $\wedge$ by $a \prec b$ if and only if $a \wedge b = a$. Therefore, the notions of tree in terms of $\wedge$ and $\prec$ are classically bi-interpretable, and if $(T, \wedge)$ is a computable tree, then so is the corresponding $(T, \prec)$. However, the computability of $\prec$ need not imply computability of $\wedge$. Therefore, by choosing a definition based on $\prec$ rather than $\wedge$, we consider a broader class of computable trees. In Section 6, we give a brief discussion and conjecture about criteria for computable categoricity of trees in the language of the infimum.

By a *homomorphism* of trees, we mean a map which respects the partial orders, but need not preserve infima. Similarly, an *embedding* is a one-to-one homomorphism $T \hookrightarrow T'$. We will frequently use the equivalence relation $\equiv$ given by

$$T \equiv T' \iff T \hookrightarrow T' \hookrightarrow T.$$

A tree $(T', \prec')$ is a *subtree* of $(T, \prec)$ if $T' \subseteq T$ and the inclusion map respects the partial orders. Thus the infimum of two elements in T may not be the same as their infimum in T'. (This broader notion of subtree is another reason for choosing a definition of tree based on $\prec$ rather than $\wedge$.) Also, the root of T may be distinct from the root of T', as in the case of the subtrees $T[x]$, which we will consider frequently. If $x \in T$, then $T[x] = \{y \in T : x \preceq y\}$. The partial order on $T[x]$ is the restriction to $T[x]$ of the partial order $\prec$ on T. Therefore $T[x]$ is a subtree of T with root x. If x is an immediate successor of the root in T, then we refer to $T[x]$ as a successor tree (of the root) in T. We define the *height of T above x* by

$$\mathrm{ht}_x(T) = \mathrm{ht}(T[x]).$$

2 Relatively computably categorical trees

In [8], Goncharov gave a syntactic condition which, under some extra hypotheses, is equivalent to computable categoricity. In [3] Ash, Knight, Mannasse, and Slaman proved that this condition is actually equivalent, with no extra hypotheses needed, to the stronger notion of relative computable categoricity. In fact, they proved a more general result which applies to relative α-categoricity for any $\alpha < \omega_1^{CK}$. In order to understand these statements fully, one would need to know about computable infinitary formulas as defined by Ash. However, all formulas we will need in this paper are finitary. Thus we state the relevant definitions and results correctly and completely, but we provide some clarifying remarks.

Definition 2.1 Let $\mathcal{L}$ be a computable language, and let $\mathcal{A}$ be a computable $\mathcal{L}$-structure. A *formally* Σ_1^0 *Scott family* for $\mathcal{A}$ is a c.e. collection Θ of computable Σ_1 formulas (possibly infinitary) so that

1. there is a finite tuple $\vec{c}$ so that for any $\theta(\vec{x}) \in \Theta$, all of the parameters appearing in θ are among $\vec{c}$;
2. for each tuple $\vec{a} \in \mathcal{A}$ of distinct elements, there is a $\theta(\vec{x}) \in \Theta$ so that $\mathcal{A} \models \theta(\vec{a})$;
3. for each $\theta \in \Theta$, each tuple $\vec{a} \in \mathcal{A}$ of distinct elements and each tuple $\vec{a}' \in \mathcal{A}$ of distinct elements, if $\mathcal{A} \models \theta(\vec{a}) \wedge \theta(\vec{a}')$, then $(\mathcal{A}, \vec{a}) \cong (\mathcal{A}, \vec{a}')$.

Note that since we talk about a c.e. set of formulas, we must have some way of assigning code numbers to the computable formulas. Note also that any finitary existential formula is (logically equivalent to) a computable Σ_1^0 formula. Again, throughout this paper, we will deal only with finitary formulas.

Theorem 2.2 (Ash-Knight-Mannasse-Slaman [3]) *Let $\mathcal{L}$ be a computable language, and let $\mathcal{A}$ be a computable $\mathcal{L}$-structure. Then $\mathcal{A}$ is relatively computably categorical iff $\mathcal{A}$ has a formally Σ_1^0 Scott family.*

The forward direction of this result is difficult and requires a forcing construction. The other direction follows from a straightforward back-and-forth argument. In this section, we use the easy direction of Theorem 2.2 to show that a tree with finite type is relatively computably categorical, and thus computably categorical.

Definition 2.3 Using the same notation as the introduction, we say that x has *weakly finite type* if there are only finitely many isomorphism types in the set $\{T[x_i] : i \geq 1\}$, each of which has weakly finite type. T itself has weakly finite type if every ω-branch point in T has weakly finite type.

Definition 2.4 Let T be a tree of finite height with root node r, and let $x \in T$, $x \neq r$. T_x is defined to be $(\{z \in T : z \text{ is not comparable to } x\} \cup \{r\}, \prec_T)$.

We need to prove a few facts about this operation.

Lemma 2.5 *Let T be a tree of finite height with root node r. Let $x, y \in T$, $x, y \neq r$, and x, y incomparable.*

1. *T_x is a tree.*
2. *$(T_x)_y = (T_y)_x$.*
3. *If T has weakly finite type, then so does T_x.*

Proof. 1. The ordering relation on T_x is inherited from T, and the root node of T, by definition, is in T_x.

2. Since each tree inherits its relation from T, we need only show that the two trees have the same underlying set. Note that r is a member of both trees. Let $z \neq r$ be a member of $(T_x)_y$. Then z is incomparable with x, since it belongs to T_x. (The operation never adds elements to a tree, so $(T_x)_y \subseteq T_x$.) And by definition, z is incomparable with y in T, since the relation on T_x is inherited from T. Consequently, $z \in (T_y)_x$. By symmetry, $(T_x)_y = (T_y)_x$.

3. We induct on the height of T. First note that the definition only applies to trees of height ≥ 2. If $\mathrm{ht}(T) = 2$, then x is an immediate successor of r with no successors, so the claim is obviously true.

Let $\mathrm{ht}(T) = n + 1$. If x is an immediate successor of r, then $T_x = T - T[x]$, which has weakly finite type. If x is strictly above an immediate successor r_1 of r, then we must first understand exactly how T_x looks. Let $(r_1^i)_{i \in I}$ be the immediate successors of r_1 in the tree $(T[r_1])_x$. (The set I might be finite or infinite.) Then the tree T_x is as follows:

1. $T - T[r_1] \subseteq T_x$; and
2. for each $i \in I$, the tree $((T[r_1])_x)[r_1^i]$ is attached directly above r.

By induction, $(T[r_1])_x$ has weakly finite type. Consequently, each ω-branch point of $(T[r_1])_x$ has only finitely many types directly above it. Therefore, there are only finitely many isomorphism types in the set $\{((T[r_1])_x)[r_1^i] : i \in I\}$. And so, if r is an ω-branch point in T_x, then it has weakly finite type. Moreover, since $(T[r_1])_x$ has weakly finite type and T has weakly finite type, all other ω-branch points of T_x have weakly finite type. Thus, T_x has weakly finite type. ∎

Based on part 2 of the preceding lemma, if x, y are incomparable nodes in T, we write $T_{x,y}$ for $(T_x)_y = (T_y)_x$. One further piece of notation we use is to denote the isomorphism type of a tree $T[x]$ by $ot(T[x])$.

Lemma 2.6 *If T has weakly finite type, then the set $\{ot(T[x]) : x \in T\}$ is finite.*

Proof. We induct on the height of T. If $\mathrm{ht}(T) = 1$, then it's clear. Let $\mathrm{ht}(T)$ be $n+1$, let r be its root node and its immediate successors be members of the sequence $(r_i)_{i\in I}$. Whether I is finite or infinite, the set $\{ot(T[r_i]) : i \in I\}$ is finite, since T has weakly finite type. Moreover, by the induction hypothesis, for each $i \in I$, the set $\{ot(T[z]) : z \succ r_i\}$ is finite. Consequently, $\{ot(T[x]) : x \in T\} = \{ot(T)\} \cup \{ot(T[r_i]) : i \in I\} \cup \bigcup_{i\in I}\{ot(T[z]) : z \succ r_i\}$ is finite. (Even if I is infinite, there are only finitely many different sets in this union by the comments above.) ■

Based on Lemma 2.6, we could restate Definition 2.3 by saying the a finite height tree T has weakly finite type if and only if it has finitely many orbits under automorphisms of T.

Lemma 2.7 *If T has weakly finite type and T has root node r, then the set $\{ot(T_x) : x \neq r\}$ is finite.*

Proof. We induct on the height of T. If $\mathrm{ht}(T) = 2$, then there is only one element in the set $\{ot(T_x) : x \neq r\}$.

Let $\mathrm{ht}(T) = n+1$. Let the immediate successors of r be members of the sequence $(r_i)_{i\in I}$; and let $r_1, \ldots, r_k$ be the successors so that for all $i \in I$, there is $1 \leq j \leq k$ with $T[r_i] \cong T[r_j]$. By induction, we know that for each $1 \leq j \leq k$, there are $p_j \in \mathbb{N}$ and $r_j^1, \ldots, r_j^{p_j} \succ r_j$ so that for each $x \succ r_j$, there is $1 \leq q \leq p_j$ with $(T[r_j])_x \cong (T[r_j])_{r_j^q}$. It is clear from the description of T_x in the proof of Lemma 2.5 that the set $\{ot(T_{r_j}) : 1 \leq j \leq k\} \cup \bigcup_{1\leq j\leq k}\{ot(T_{r_j^q}) : 1 \leq q \leq p_j\}$ is equal to $\{ot(T_x) : x \neq r\}$. ■

The proof of the next two lemmas will use induction on the *degree of ω-branching*, which we define formally below. Intuitively, T has degree m ω-branching if and only if the following two conditions hold: first, there is a chain of elements in T which contains m many elements which are ω branching and second, for any chain of elements in T, there are at most m many elements in the chain which are ω-branching.

Definition 2.8 Let T be a finite height tree.

1. T has degree 0 ω-branching iff it is finite;
2. T has degree $n+1$ ω-branching iff
   a. T does not have degree n ω-branching; and
   b. if x is a minimal ω-branch point of T with immediate successors $r_i, i \geq 1$, then for each $i \geq 1$, there is $m \leq n$ so that $T[r_i]$ has degree m ω-branching.

Lemma 2.9 *If T_1 and T_2 have weakly finite type and $T_1 \not\to T_2$, then there is a finite subtree $T_1' \subseteq T_1$ such that $T_1' \not\to T_2$.*

Proof. We induct on the height of tree T_1. If $\mathrm{ht}(T_1) = 1$, then $T_1 \hookrightarrow T_2$.

Let T_1 have height $n+1$. We induct on the degree of ω-branching in T_1. First, if the degree is 0, then T_1 is finite, so $T_1' = T_1$.

Let T_1 have degree $m+1$ ω-branching. We induct on the height of T_2. If $\mathrm{ht}(T_2) = 1$, then T_1 is infinite, but T_2 is finite, so the desired conclusion is clear.

Let T_2 have height $p+1$. We induct on the degree of ω-branching in T_2. If the degree is 0, then then T_1 is infinite, but T_2 is finite.

Let T_2 have degree $q+1$ ω-branching.

Case 1: The root node r of T_1 is finite branching with immediate successors $r_1, \ldots, r_k$. By induction on the height of T_1, for each $i \in \{1, \ldots, k\}$ and each $s \in T_2$, if $T_1[r_i] \not\hookrightarrow T_2[s]$, then there is a finite subtree $(T_1[r_i])^s \subseteq T_1[r_i]$ so that $(T_1[r_i])^s \not\hookrightarrow T_2[s]$. Consequently, by Lemma 2.6, for each $i \in \{1, \ldots, k\}$, there is a single finite $T_1[r_i]' \subseteq T_1[r_i]$ so that for all $s \in T_2$, if $T_1[r_i] \not\hookrightarrow T_2[s]$, then $T_1[r_i]' \not\hookrightarrow T_2[s]$. (We may assume that $r_i \in T_1[r_i]'$ for each $i \in \{1, \ldots, k\}$.)

Define $T_1' \subseteq T_1$ as follows:

1. $r \in T_1'$;
2. $T_1[r_i]' \subseteq T_1'$ for each $i \in \{1, \ldots, k\}$.

We claim that $T_1' \not\hookrightarrow T_2$. Assume otherwise that $f : T_1' \hookrightarrow T_2$. Then f maps $r, r_1, \ldots, r_k$ to some $s, s_1, \ldots s_k$ so that

1. s is below all of $s_1, \ldots, s_k$;
2. no two of $s_1, \ldots, s_k$ are comparable; and
3. $T_1[r_i]' \hookrightarrow T_2[s_i]$ for all $1 \leq i \leq k$.

Consequently, by the way in which we defined each $T_1[r_i]'$,

1. s is below all of $s_1, \ldots, s_k$;
2. no two of $s_1, \ldots, s_k$ are comparable; and
3. $T_1[r_i] \hookrightarrow T_2[s_i]$ for all $1 \leq i \leq k$.

Therefore, $T_1 \hookrightarrow T_2$, a contradiction.

Case 2: The root node r is an ω-branch node with immediate successors $r_i, i \geq 1$, where $r_1, r_2, \ldots, r_k, r_{k+1}, \ldots, r_l$ are such that

1. $T_1[r_1], \ldots, T_1[r_k]$ are all of the direct successor trees of r whose isomorphism types occur finitely often directly above r;

2. $T_1[r_{k+1}], \ldots, T_1[r_l]$ are direct successor trees of r whose isomorphism types occur infinitely often directly above r;

3. for all j, j' with $k+1 \leq j < j' \leq l$, $T_1[r_j] \ncong T_1[r_{j'}]$; and

4. for all $i \geq 1$, there is $j \leq l$ so that $T_1[r_i] \cong T_1[r_j]$.

Case 2a: The root node s of T_2 is an ω-branch node of T_2 with immediate successors $s_i, i \geq 1$, where $s_1, s_2, \ldots s_t, s_{t+1}, \ldots, s_u$ have properties analogous to those of $r_1, \ldots, r_k, r_{k+1} \ldots, r_l$. Consider $T_3 \subset T_1$ defined as follows:

1. $r \in T_3$;

2. $T_1[r_i] \subset T_3$ for each $1 \leq i \leq k$.

If $T_3 \not\hookrightarrow T_2$, then by induction on the degree of ω-branching in T_1, there is a finite subtree $T_1' \subseteq T_3 \subseteq T_1$ so that $T_1' \not\hookrightarrow T_2$. So assume that $T_3 \hookrightarrow T_2$. Thus, it cannot be the case that each $T_1[r_{k+1}], \ldots, T_1[r_l]$ embeds into one of $T_2[s_{t+1}], \ldots, T_2[s_u]$. Otherwise, $T_1 \hookrightarrow T_2$. (T_2 with countably many more copies of each of $T_2[s_{t+1}], \ldots T_2[s_u]$ attached directly above s is a tree isomorphic to T_2 itself.)

Let $T_1[r_{\alpha_1}], \ldots, T_1[r_{\alpha_v}]$ be a list of all the trees among $T_1[r_1], \ldots, T_1[r_k]$ which individually do not embed into any $T_2[s_{t+1}], \ldots, T_2[s_u]$. Similarly, let $T_1[r_{\beta_1}], \ldots, T_2[r_{\beta_w}]$ be a list of all those among $T_1[r_{k+1}], \ldots, T_1[r_l]$ with this property. Consider the tree $T_1^* \subseteq T_1$ defined as follows:

1. $r \in T_1^*$;

2. $T_1[r_{\alpha_1}], \ldots, T_1[r_{\alpha_v}] \subset T_1^*$;

3. for each $i \in \{1, \ldots, w\}$, T_1^* contains infinitely many copies of $T_1[r_{\beta_i}]$ directly above r.

This tree cannot be embedded into the subtree $\{s\} \cup T_2[s_1] \cup \cdots \cup T_2[s_t]$; otherwise $T_1 \hookrightarrow T_2$. By induction on the degree of ω-branching of T_2, there is a finite number μ and a tree $\hat{T}_1$ so that

1. $r \in \hat{T}_1$;

2. $T_1[r_{\alpha_1}], \ldots, T_1[r_{\alpha_v}] \subset \hat{T}_1$;

3. for each $i \in \{1, \ldots, w\}$, $\hat{T}_1$ contains exactly μ copies of $T_1[r_{\beta_i}]$ directly above r; and

4. $\hat{T}_1 \not\hookrightarrow \{s\} \cup T_2[s_1] \cup \cdots \cup T_2[s_t]$.

In fact, $\hat{T}_1 \not\hookrightarrow T_2$. Why? Assume otherwise that $f : \hat{T}_1 \hookrightarrow T_2$. None of the roots of the copies of $T_1[r_{\alpha_1}], \dots, T_1[r_{\alpha_v}], T_1[r_{\beta_1}], \dots, T_2[r_{\beta_w}]$ can be mapped to a point in $T_2[s_i]$ with $i \geq t+1$. Thus, $f : \hat{T}_1 \hookrightarrow \{s\} \cup T_2[s_1] \cup \cdots \cup T_2[s_t]$, a direct contradiction. And so, by induction on the degree of ω-branching of T_1, there is a finite subtree $T_1' \subseteq \hat{T}_1 \subset T_1$ such that $T_1' \not\hookrightarrow T_2$.

Case 2b: The root node s of T_2 has finitely many immediate successors. We proceed by induction on the number of immediate successors s has. First assume that s has only one immediate successor s_1. Since the root node r of T_1 has infinitely many immediate successors, we know, by induction on the height of T_2, that there is a finite tree $T_1' \subset T_1$ so that

1. $r \in T_1'$;
2. r has at least two immediate successors in T_1'; and
3. $T_1' \not\hookrightarrow T_2 - \{s_1\} (\cong T_2[s_1])$.

Of course, this implies that $T_1' \not\hookrightarrow T_2$; if $f : T_1' \hookrightarrow T_2$, then $f(r) \in T_2[s_1]$ or $s_1 \notin \text{range}(()f)$, since s_1 is the sole immediate successor of s, and r has at least two.

Assume that s has $t+1$ immediate successors $s_1, \dots, s_{t+1}$ in T_2. Of course, for each $j \in \{1, \dots, t+1\}$, $T_1 \not\hookrightarrow T_2 - T_2[s_j]$. Therefore, by induction on the number of immediate successors of the root node s, for each $j \in \{1, \dots, t+1\}$, there is a tree $(T_1)^j \subset T_1$ and a finite number μ_j so that

1. $r \in (T_1)^j$;
2. for each $i \in \{1, \dots, k\}$, $T_1[r_i] \subset (T_1)^j$;
3. for each $i \in \{k+1 \dots, l\}$, $(T_1)^j$ contains exactly μ_j many copies of $T_1[r_i]$ directly above r; and
4. $(T_1)^j \not\hookrightarrow T_2 - T_2[s_j]$.

Let $\mu = \max\{\mu_j : j \in \{1, \dots, t+1\}\}$. Define T_1^* as follows:

1. $r \in T_1^*$;
2. for each $i \in \{1, \dots, k\}$, $T_1[r_i] \subset T_1^*$;
3. for each $i \in \{k+1 \dots, l\}$, T_1^* contains exactly μ many copies of $T_1[r_i]$ directly above r.

Assume, without loss of generality, that the immediate successors of r in T_1^* can be listed as $r_1, \dots, r_w$. If for some $v_1, \dots, v_w \in T_2$, $f : T_1^* \hookrightarrow T_2$ and $f(r_i) = v_i$ for $1 \leq i \leq w$, then the following must be true:

a. for each $1 \leq j \leq t+1$, there is $1 \leq i \leq w$ so that $v_i \succeq s_j$; and

b. $(T_1 - T_1^*) \cup \{r\} \not\hookrightarrow (T_2)_{v_1,\ldots,v_w}$ (where $(T_2)_{v_1,\ldots,v_w}$ is defined in Definition 2.5).

Otherwise, $(T_1)^j \hookrightarrow T_2 - T_2[s_j]$ for some j or $T_1 \hookrightarrow T_2$.

Notice that if $v_1, \ldots, v_w$ satisfy a. above, then $\mathrm{ht}((T_2)_{v_1,\ldots,v_w}) < \mathrm{ht}(T_2)$. And so, by induction on the height of tree T_2, it must be the case that for each tuple $\vec{v} = v_1, \ldots, v_w \in T_2$ which satisfy a. and b. above, there is a finite number $\kappa_{\vec{v}}$ and a subtree $T^{\vec{v}} \subseteq (T_1 - T_1^*) \cup \{r\}$ so that

1. $r \in T^{\vec{v}}$;

2. for each $i \in \{k+1, \ldots, l\}$, $T^{\vec{v}}$ contains exactly $\kappa_{\vec{v}}$ many copies of $T_1[r_i]$ directly above r; and

3. $T^{\vec{v}} \not\hookrightarrow (T_2)_{v_1,\ldots,v_w}$.

Therefore, by Lemma 2.7, there is a single number κ and a single tree $\hat{T} \subseteq (T_1 - T_1^*) \cup \{r\}$ so that

1. $r \in \hat{T}$;

2. for each $i \in \{k+1, \ldots, l\}$, $\hat{T}$ contains exactly κ many copies of $T_1[r_i]$ directly above r; and

3. for each list of incomparable elements $v_1, \ldots, v_w$ which satisfy a. and b. above, $\hat{T} \not\hookrightarrow (T_2)_{v_1,\ldots,v_w}$.

Finally, define $\hat{T}_1 \subseteq T_1$ as follows:

1. $r \in \hat{T}_1$;

2. for each $i \in \{1, \ldots, k\}$, $T_1[r_i] \subset \hat{T}_1$;

3. for each $i \in \{k+1, \ldots, l\}$, $\hat{T}_1$ contains exactly $\mu + \kappa$ many copies of $T_1[r_i]$ directly above r.

A straightforward argument similar to ones previously given shows that $\hat{T}_1 \not\hookrightarrow T_2$. Therefore, by induction on the degree of ω-branching in T_1 it must be the case that there is a finite $T_1' \subseteq \hat{T}_1 \subset T_1$ so that $T_1' \not\hookrightarrow T_2$. ∎

Lemma 2.10 *If T_1 and T_2 have strongly finite type, and $T_1 \hookrightarrow T_2 \hookrightarrow T_1$, then $T_1 \cong T_2$.*

This lemma need not hold if either T_1 or T_2 is only of finite type. For instance, let T_1 be the tree $\omega^{<3}$, with $x \preceq y$ iff x is an initial segment of y, and delete $\{\langle 0, n\rangle : n \in \omega\}$ from T_1 to get T_2 (or let T_2 be any other tree of height 3 in which T_1 embeds). Recall that the equivalence relation $\equiv$ on all trees is defined by

$$T \equiv T' \iff T \hookrightarrow T' \hookrightarrow T.$$

So the lemma says that for trees of strongly finite type, $\equiv$ and $\cong$ are identical.

Proof. We induct on the height of T_1. If $\mathrm{ht}(T_1) = 1$, then the result is obvious.

Let $\mathrm{ht}(T_1) = n+1$. We induct on the degree of ω-branching of T_1. If the degree is 0, then T_1 is finite. Therefore, if $T_1 \hookrightarrow T_2 \hookrightarrow T_1$, then both trees are finite of the same size, so any embedding must be an isomorphism.

Let the T_1 have degree $q+1$ ω-branching. Let r be the root node of T_1 and s be the root of T_2. It is clear that $\mathrm{ht}(T_1) = \mathrm{ht}(T_2)$, and that in any embedding of one into the other, the root node must be mapped to the root node.

Case 1: r has finitely many immediate successors. First, we argue that s must have the same number of immediate successors. Let r have immediate successors $r_1, \ldots, r_m$. Let p be the number of trees among $T_1[r_1], \ldots, T_1[r_m]$ which have height equal to n. Since $T_1 \hookrightarrow T_2 \hookrightarrow T_1$, the number of such immediate successors of s must also be p, and in the embeddings, a successor at the base of a tree of height n must be mapped to a successor at the base of a tree of height n.

If r has no more immediate successors, then s cannot either, because $T_2 \hookrightarrow T_1$. Otherwise, let h_1 be the greatest number $< n$ so that r has an immediate successor r_i with $\mathrm{ht}(T_1[r_i]) = h_1$, and let p_1 be the number of such r_i's. First, notice that s cannot have a successor s_i with $h_1 < \mathrm{ht}(T_2[s_i]) < n$, since $T_2 \hookrightarrow T_1$, and all of the successors r_i of r with $\mathrm{ht}(T_1[r_i]) > h_1$ are images of successors s_i of s with $\mathrm{ht}(T_2[s_i]) = n$. Moreover, s must have exactly p_1 immediate successors s_i with $\mathrm{ht}(T_2[s_i]) = h_1$, since $T_1 \hookrightarrow T_2 \hookrightarrow T_1$. Continuing in this fashion we complete the argument that r and s must have exactly the same number of immediate successors.

We complete this case by arguing by induction on the number of immediate successors of r, s. First, if r has only one immediate successor r_1, and s has only one immediate successor s_1, then obviously, $T_1[r_1] \hookrightarrow T_2[s_1] \hookrightarrow T_1[r_1]$. By induction on the height on the trees, $T_1[r_1] \cong T_2[s_1]$, so $T_1 \cong T_2$.

Assume that r has immediate successors $r_1, \ldots, r_{m+1}$ and s has immediate successors $s_1, \ldots, s_{m+1}$. By induction on the height of the trees, there must be a successor r_i of r so that

1. $T_1[r_i]$ has height n; and
2. $T_1[r_i]$ is maximal among $T_1[r_1], \ldots, T_1[r_{m+1}]$ with respect to embedding; i.e., if $j \neq i$ and $T_1[r_i] \hookrightarrow T_1[r_j]$, then $T_1[r_i] \cong T_1[r_j]$.

Assume, without loss of generality, that it is r_1.

Now consider $f : T_1 \hookrightarrow T_2$ and $g : T_2 \hookrightarrow T_1$. Since $T_1[r_1]$ has height n, there are $1 \leq j, k \leq m+1$ such that $f : T_1[r_1] \hookrightarrow T_2[s_j]$ and $f : T_1 - T_1[r_1] \hookrightarrow T_2 - T_2[s_j]$; and $g : T_2[s_j] \hookrightarrow T_1[r_k]$ and $g : T_2 - T_2[s_j] \hookrightarrow T_1 - T_1[r_k]$. However, then $T_1[r_1] \hookrightarrow T_1[r_k]$, so by our choice of $T_1[r_1]$, we know that $T_1[r_1] \cong T_1[r_k]$. Consequently, by induction on the height of trees, $T_1[r_1] \cong T_2[s_j]$; and by induction on the number of immediate successors the root node has, $T_1 - T_1[r_1] \cong T_2 - T_2[s_j]$. Therefore, $T_1 \cong T_2$.

Case 2: r is an ω-branch node with immediate successors $r_i, i \geq 1$, and s is an ω-branch node with immediate successors $s_i, i \geq 1$. First, consider the subtree $T_1' \subseteq T_1$ which is defined as follows:

1. $r \in T_1'$;
2. $T_1[r_i] \subseteq T_1'$ iff $T_1[r_i]$ has height n.

Define T_2' similarly. Of course $T_1' \hookrightarrow T_2' \hookrightarrow T_1'$.

Next, consider the subtree $T_1'' \subseteq T_1'$ defined as follows:

1. $r \in T_1''$;
2. $T_1[r_i] \subseteq T_1''$ iff $T_1[r_i] \subseteq T_1'$ and the isomorphism type of $T_1[r_i]$ occurs only finitely often directly above r.

Define T_2'' similarly. We claim that $T_1'' \hookrightarrow T_2'' \hookrightarrow T_1''$. To see this, let $T_1[r_j] \subset T_1''$. Of course, it must be the case that $T_1[r_j] \hookrightarrow T_2[s_k]$ for some $T_2[s_k] \subset T_2'$. If $T_2[s_k] \not\subset T_2''$, then the isomorphism type of $T_2[s_k]$ occurs infinitely often directly above s. Since $\text{ht}(T_2[s_k]) = n$, it must be the case that $T_2[s_k] \hookrightarrow T_1[r_l]$, where the isomorphism type of $T_1[r_l]$ occurs infinitely often directly above r_0. However, then $T_1[r_j] \hookrightarrow T_1[r_l]$, a contradiction to the fact that T_1 has strongly finite type. And so, $T_2[s_k] \subset T_2''$. Consequently, $T_1'' \hookrightarrow T_2''$. A symmetric argument shows that $T_2'' \hookrightarrow T_1''$. By induction on the degree of ω-branching of T_1, $T_1'' \cong T_2''$.

Next, let $T_1[r_j] \subset T_1' - T_1''$. Of course, it must be the case that $T_1[r_j] \hookrightarrow T_2[s_k]$ for some $T_2[s_k] \subset T_2' - T_2''$. Similarly, $T_2[s_k] \hookrightarrow T_2[r_l]$ for some $T_1[r_l] \subset T_1' - T_1''$. But then $T_1[r_j] \hookrightarrow T_1[r_l]$, so $T_1[r_j] \cong T_1[r_l]$, since T_1 has strongly finite type. By induction on the height of the trees, $T_1[r_j] \cong T_1[s_k]$. Similarly, let $T_2[s_p] \subset T_2' - T_2''$. Then $T_2[s_p] \cong T_1[r_t]$ for some $T_1[r_t] \subset T_1' - T_1''$. Therefore, $\{r\} \cup (T_1' - T_1'') \cong \{s\} \cup (T_2' - T_2'')$. Consequently, $T_1' \cong T_2'$.

Finally, we claim that if $f : T_1 \hookrightarrow T_2$, then $f : \{r\} \cup (T_1 - T_1') \hookrightarrow \{s\} \cup (T_2 - T_2')$. Let $T_1[r_j] \not\subseteq T_1'$, and let $f : T_1[r_j] \hookrightarrow T_2[s_k]$. It cannot be the case that $T_2[s_k] \subseteq T_2''$, because, as we have seen, the number of immediate successors y of s with $T_2[y] \subseteq T_2''$ is exactly the same as the number of immediate successors x of r with $T_1[x] \subseteq T_1''$, and each such x must be sent to such an y. Also, it cannot be the case that $T_2[s_k] \subseteq T_2' - T_2''$; otherwise, as we have seen, it would be the case that $T_2[s_k] \cong T_1[r_l]$ where $T_1[r_l]$ occurs infinitely often immediately above r; but $T_1[r_j] \not\hookrightarrow T_1[r_l]$, since T_1 has strongly finite type. Thus $f : T_1[r_j] \hookrightarrow T_1[s_k]$, where $T_1[s_k] \subseteq T_1 - T_1'$. Therefore, $f : \{r\} \cup (T_1 - T_1') \hookrightarrow \{s\} \cup (T_2 - T_2')$. A symmetric argument shows that $\{s\} \cup (T_2 - T_2') \hookrightarrow \{r\} \cup (T_1 - T_1')$. Therefore, by induction on the height of the trees, $\{r\} \cup (T_1 - T_1') \cong \{s\} \cup (T_2 - T_2')$. And so, $T_1 \cong T_2$. ∎

Lemma 2.11 *If T has strongly finite type, then T has a formally Σ_1^0 Scott family of finitary formulas with no parameters.*

Proof. First, note that if T is a tree and T' is a finite tree, then we can say that T' can be embedded in $T[x]$ with a finitary existential formula $\psi(x)$.

We induct on the height of tree T. Let r be the root node of T. If $\mathrm{ht}(T) = 1$, then $T = \{r\}$, so $\{x = x\}$ is the desired Scott family.

Let $\mathrm{ht}(T) = n+1$. Let r have immediate successors $(r_i)_{i \in I}$. First, let $r_1, \dots, r_k, r_{k+1}, \dots, r_l$ be such that

1. $T_1[r_1], \dots, T_1[r_k]$ are all of the direct successor trees of r whose isomorphism types occur finitely often directly above r;
2. $T_1[r_{k+1}], \dots, T_1[r_l]$ are direct successor trees of r whose isomorphism types occur infinitely often directly above r (this list is empty if I is finite);
3. for all j, j' with $k+1 \leq j < j' \leq l$, $T_1[r_j] \ncong T_1[r_{j'}]$; and
4. for all $i \in I$, there is $j \leq l$ so that $T_1[r_i] \cong T_1[r_j]$.

Next, for each $j \in \{1, \dots, l\}$, let T'_j be such that

1. $T'_j \subseteq T[r_j]$;
2. T'_j is finite; and
3. for all $i \in I$, if $T[r_j] \not\hookrightarrow T[r_i]$, then $T'_j \not\hookrightarrow T[r_i]$.

Finally, for each $1 \leq j \leq l$, let Ψ_j be the Turing machine which enumerates the formally Σ_1^0 Scott family for $T[r_j]$.

Given a tuple of variables $\vec{x} = x_1, \dots, x_t$ of length t, we consider all 7-tuples $\rho_t = \langle a, j, s, P, \sigma, \tau, s' \rangle$, where

1. $a \in \{0, 1\}$ and $1 \leq j \leq t$: if $a = 0$, then no member of $\vec{x}$ is going to "represent" r; if $a = 1$, then x_j is going to "represent" the root node r.
2. $s \leq \max\{|I|, t-a\}$, P is a partition of $\{1, \dots, t\}$ ($\{1, \dots, t\} - \{j\}$ if $a = 1$) into s pieces: for the part of $\vec{x}$ remaining, we divide it into subtuples $\vec{y}_1, \dots, \vec{y}_s$ according to P.
3. $\sigma = \langle i_1, \dots, i_s \rangle$ is an s-tuple so that
   a. for each $1 \leq \mu \leq s$, $i_\mu \in \{1, \dots, l\}$;
   b. for each $1 \leq \mu \leq s$, $|\vec{y}_\mu| \leq |T[r_{i_\mu}]|$; and
   c. for each $1 \leq \mu < \nu \leq s$, if $i_\mu = i_\nu$, then $i_\mu \in \{k+1, \dots, l\}$.

4. τ is an s-tuple of natural numbers: for each $1 \leq \mu \leq s$, $\tau(\mu)$ tells us which formula to use from the formally Σ_1^0 Scott family for $T[r_{i_\mu}]$.

5. s' is a natural number.

Now we form the formula $\gamma_{\rho_t}(\vec{x})$:

1. If in 1. above, $a = 1$, then γ_{ρ_t} includes a conjunct which says that x_j is at the bottom of a chain of length $n+1$; otherwise, γ_{ρ_t} includes a conjunct which says that there is v_0 so that $v_0 \prec \vec{x}$.

2. For each tuple $\vec{y}_\mu$ and each tree $T[r_{i_\mu}]$ from 2. and 3. above, γ_{ρ_t} includes conjuncts $\gamma_\mu(\vec{y}_\mu)$ so that

   a. if $j_1, \ldots, j_{w_\mu}$ is a complete list of the elements of $\{1, \ldots, k\}$ so that $T[r_{i_\mu}] \hookrightarrow T[r_{j_1}], \ldots, T[r_{j_{w_\mu}}]$, but $T[r_{i_\mu}] \not\cong T[r_{j_1}], \ldots, T[r_{j_{w_\mu}}]$ (and thus Lemma 2.10 implies $T[r_{j_1}], \ldots, T[r_{j_{w_\mu}}] \not\hookrightarrow T[r_{i_\mu}]$), then $\gamma_\mu(\vec{y}_\mu)$ includes the conjunct which says $\exists v_0 v_1 \cdots v_{w_\mu+1}[v_0$ is at the bottom of a chain of length $n+1$; $v_1, \ldots, v_{w_\mu}, v_{w_\mu+1}$ are incomparable; for each $1 \leq p \leq w_\mu$, $T'_{j_p} \hookrightarrow T[v_p]$, and v_p is at the bottom of a chain of length $\mathrm{ht}(T[r_{j_p}])$; $\vec{y}_\mu \succeq v_{w_\mu+1}$, and $T'_{i_\mu} \hookrightarrow T[v_{w_\mu+1}]]$;

   b. if both of the following are true about Ψ_{i_μ}:

      i. Ψ_{i_μ} halts on input $\tau(\mu)$ (a natural number) in less than s' steps and outputs a formula $\delta(\vec{z})$; and

      ii. $|\vec{y}_\mu|$ is the number of free variables actually appearing in $\delta(\vec{z})$,

      then $\gamma_\mu(\vec{y}_\mu)$ includes the conjunct $\delta(\vec{y}_\mu)$. Otherwise, it includes the conjunct $\perp$ (falsity).

3. For each $1 \leq \mu < \nu \leq s$ so that $i_\mu \neq i_\nu$, but $T[r_{i_\mu}] \cong T[r_{i_\nu}]$, $\gamma_{\rho_t}(\vec{x})$ includes a conjunct which says $\exists v_1 v_2[v_1$ and v_2 are incomparable; each is at the bottom of a chain of length $\mathrm{ht}(T[r_{i_\mu}])$; $\vec{y}_\mu \succeq v_1$ and $\vec{y}_\nu \succeq v_2]$.

Let $\Theta = \{\gamma_{\rho_t}(\vec{x}) : t \in \omega, \vec{x}$ is a t-tuple, and $\vec{x}$, ρ_t are as above $\}$. We claim that Θ is a formally Σ_1^0 Scott family of the desired form. First, since we explicitly describe how to form the formulas γ_{ρ_t}, Θ is certainly c.e. Next, if $\vec{a}$ is a t-tuple of distinct elements of T, then it is obvious to see that there is going to be some $\gamma_{\rho_t}(\vec{x})$ which it satisfies. Finally, assume that $\vec{a} = a_1, \ldots, a_t$ and $\vec{b} = b_1, \ldots, b_t$ are two tuples of distinct elements so that $T \models \gamma_{\rho_t}(\vec{a}) \wedge \gamma_{\rho_t}(\vec{b})$. We must show that $(T, \vec{a}) \cong (T, \vec{b})$.

First, it must be the case that $\vec{a}$ contains the root node r iff $\vec{b}$ does, and that $a_j = r$ iff $b_j = r$. Next, since $T \models \gamma_{\rho_t}(\vec{a})$, $\vec{a}$ (perhaps without a_j) is sorted into $\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_s$ (according to the substitution of $\vec{a}_\mu$ for the subtuple of variables $\vec{y}_\mu$ for each $1 \leq \mu \leq s$) so that for each $1 \leq \mu \leq s$ the following things are determined:

1. all elements in a single tuple $\vec{a}_\mu$ are contained in the same direct successor tree of r;

2. $\vec{a}_\mu$ belongs to a direct successor tree of r of order type that of $T[r_{i_\mu}]$;

3. for all ν with $1 \leq \nu \leq s$ and $\nu \neq \mu$, some element of $\vec{a}_\mu$ and some element of $\vec{a}_\nu$ are not contained in the same direct successor tree of r; and

4. $\vec{a}_\mu$ satisfies the formula $\delta(\vec{z})$, obtained from the formally Σ^0_1 Scott family for $T[r_{i_\mu}]$.

To see 2., notice that clause a. in the formation of $\gamma_{\vec{y}_\mu}$ guarantees that $\vec{a}_\mu$ is contained in a direct successor tree of r which embeds T'_{i_μ}, and hence $T[r_{i_\mu}]$. Therefore, we know that $\vec{a}_\mu$ is either contained in a direct successor tree of order type $T[r_{i_\mu}]$ or one of the *finitely* many non-isomorphic direct successor trees which embed $T[r_{i_\mu}]$. But the rest of clause a. rules out all other possibilities.

To see 3., assume that in 2. we have determined that the type of the direct successor tree to which $\vec{a}_\mu$ belongs is the same as the type of the direct successor tree to which $\vec{a}_\nu$ belongs. Then $\gamma_\rho(\vec{a})$ contains a conjunct which says that $\exists v_1 v_2 [v_1$ and v_2 are incomparable; each of v_1, v_2 is at the bottom of a chain of length $\mathrm{ht}(T[r_{i_\mu}])$; and $v_1 \preceq \vec{a}_\mu$; and $v_2 \preceq \vec{a}_\nu]$.

Of course, the tuple $\vec{b}$ is sorted by γ_{ρ_t} in exactly the same way. Therefore, by the definition of a Scott family, r has immediate successor trees $T_1, \ldots, T_s$, $T'_1, \ldots, T'_s$ so that for all $1 \leq p \neq q \leq s$ the following are true:

1. $T_p \neq T_q$, and $T'_p \neq T'_q$;

2. $\vec{a}_p \in T_p$; $\vec{b}_p \in T'_p$; and

3. $(T_p, \vec{a}_p) \cong (T'_p, \vec{b}_p)$.

And so, $(T, \vec{a}) \cong (T, \vec{b})$. ■

Our next result shows that finite type implies relative computable categoricity in Theorem 1.8.

Theorem 2.12 *If T has finite type, then T has a formally Σ^0_1 Scott family of finitary formulas.*

Proof. We induct on the height of T. If $\mathrm{ht}(T) = 1$, then T is finite, so T has strongly finite type, and the previous result applies. Let $\mathrm{ht}(T) = n + 1$, and let r be the root node of T.

Case 1: r has only finitely many immediate successors $r_1, \ldots, r_m$. Then $T[r_1], T[r_2], \ldots, T[r_m]$ all have finite type, and hence all have formally Σ^0_1 Scott families by induction. For each $1 \leq i \leq m$, let $\vec{c}_i$ be the parameters of $T[r_i]$ which appear in the Scott family for $T[r_i]$. Let the tuple of parameters of our formally Σ^0_1 Scott family be $\vec{c} = r, r_1, \ldots, r_m, \vec{c}_1, \ldots, \vec{c}_m$.

Let $\vec{a}$ be a tuple of distinct elements in T, and let $\vec{x}$ be a corresponding tuple of variables. We construct the formula $\gamma_{\vec{a}}(\vec{x})$ as follows:

1. if any $a_i \in \vec{a}$ is equal to an element c of our parameter set, then we include the conjunct $x_i = c$;

2. let $\vec{a}'$ be the tuple $\vec{a}$ with such elements removed, and let $\vec{x}'$ be the corresponding tuple of variables; we include a conjunct which says that $\vec{x}' \cap \vec{c} = \emptyset$;

3. we divide the tuple $\vec{a}'$ into subtuples $\vec{a}_1, \dots, \vec{a}_m$ so that $\vec{a}_i \in T[r_i]$ (some subtuples might be empty); we divide the tuple of variables $\vec{x}'$ into $\vec{x}_1, \dots, \vec{x}_m$ accordingly; for each $1 \leq i \leq m$, we include the conjunct which says that $\vec{x}_i \succeq r_i$;

4. for each tuple $\vec{a}_i$, we search until we find the first formula δ_i from the formally Σ^0_1 Scott family for $T[r_i]$ which $\vec{a}_i$ satisfies; we include the conjunct $\delta_i(\vec{x}_i)$.

Let $\Theta = \{\gamma_{\vec{a}}(\vec{x}) : \vec{a} \in T \text{ is a tuple of distinct variables}\}$. It is clear that Θ is a formally Σ^0_1 Scott family of finitary formulas.

Case 2: r has infinitely many immediate successors $r_i, i \geq 1$. As usual, let $r_1, \dots, r_k, \dots, r_l$ be such that

1. $T_1[r_1], \dots, T_1[r_k]$ are all of the direct successor trees of r whose isomorphism types occur finitely often directly above r;

2. $T_1[r_{k+1}], \dots, T_1[r_l]$ are direct successor trees of r whose isomorphism types occur infinitely often directly above r;

3. for all j, j' with $k+1 \leq j < j' \leq l$, $T_1[r_j] \not\cong T_1[r_{j'}]$; and

4. for all $i \geq 1$, there is $j \leq l$ so that $T_1[r_i] \cong T_1[r_j]$.

By induction, each of the trees $T[r_1], \dots, T[r_k]$ has a formally Σ^0_1 Scott family of finitary formulas. For each $1 \leq i \leq k$, let $\vec{c}_i$ be the parameters of $T[r_i]$ which appear in the Scott family for $T[r_i]$. Furthermore, notice that that the tree $T - \bigcup_{1 \leq i \leq k} T[r_i]$ has strongly finite type. Therefore, by the previous lemma, this tree has a formally Σ^0_1 Scott family of finitary formulas with no parameters. Let the tuple of parameters of our formally Σ^0_1 Scott family be $r, r_1, \dots, r_k, \vec{c}_1, \dots, \vec{c}_k$.

Let $\vec{a}$ be a tuple of distinct elements in T, and let $\vec{x}$ be a corresponding tuple of variables. We construct the formula $\gamma_{\vec{a}}(\vec{x})$ as follows:

1. if any $a_i \in \vec{a}$ is equal to an element c of our tuple of parameters, then we include the conjunct $x_i = c$;

2. let $\vec{a}'$ be the tuple $\vec{a}$ with such elements removed, and let $\vec{x}'$ be the corresponding tuple of variables; we include a conjunct which says that $\vec{x}' \cap \vec{c} = \emptyset$;

3. we divide the tuple $\vec{a}'$ into subtuples $\vec{a}_1, \ldots, \vec{a}_k, \vec{a}_{k+1}$ so that $\vec{a}_i \in T[r_i]$ for $1 \leq 1 \leq k$ and $\vec{a}_{k+1} \in T - \bigcup_{1 \leq i \leq k} T[r_i]$ (some subtuples might be empty); we divide the tuple of variables $\vec{x}'$ into $\vec{x}_1, \ldots, \vec{x}_k, \vec{x}_{k+1}$ accordingly; for each $1 \leq i \leq k$, we include the conjunct which says that $\vec{x}_i \succeq r_i$; for each element a of $\vec{a}_{k+1}$, we include a conjunct which says that the corresponding variable x is incomparable to $r_1, \ldots, r_k$;

4. for each $1 \leq i \leq k$, we search until we find the first formula δ_i from the formally Σ^0_1 Scott family for $T[r_i]$ which $\vec{a}_i$ satisfies; we include the conjunct $\delta_i(\vec{x}_i)$;

5. we search until we find a formula δ_{k+1} from the formally Σ^0_1 Scott family for $T - \bigcup_{1 \leq i \leq k} T[r_i]$ which $\vec{a}_{k+1}$ satisfies; we include the conjunct $\delta_{k+1}(\vec{x}_{k+1})$.

Let $\Theta = \{\gamma_{\vec{a}}(\vec{x}) : \vec{a} \in T$ is a tuple of distinct variables$\}$. It is clear that Θ is a formally Σ^0_1 Scott family of finitary formulas. ■

3 Computably Non-Categorical Trees

To prepare for the induction that establishes Theorem 1.8, we will prove the following:

Proposition 3.1 *Let T be a computable tree of finite height with root r. If r is not of finite type but every other node in T is of finite type, then T is not computably categorical. Indeed, T has computable dimension ω.*

Our proof of Proposition 3.1 requires several distinct finite-injury constructions for the different possible cases. In each construction we build a computable tree T' isomorphic to T satisfying the requirements

$$\mathcal{R}_e : \quad \varphi_e \text{ one-one and total} \implies [(\exists w_e \in T) T[w_e] \not\cong T'[\varphi_e(w_e)]].$$

We guarantee that T' is isomorphic to T by building a Δ^0_2 function $f : T \to T'$. f will either be an isomorphism from T onto T' or it will be an isomorphism from T onto range(f). In the latter case, $T' \setminus$range(f) will consist of successor trees of the root in T' each of which will have the same isomorphism type as a successor tree of the root in T which occurs infinitely often in T. Therefore, despite the extra successor trees, T and T' will be isomorphic. Goncharov ([10]) proved that if two computable structures are not computably isomorphic but are Δ^0_2 isomorphic, then their computable dimension is ω. So, in the case where f is an isomorphism, we get the infinite dimension part of the theorem for free. We make a separate argument at the end of the section for the case when f does not map onto T'.

The node w_e will be called the *witness node* for requirement $\mathcal{R}_e$, and will be approximated by nodes $w_{e,s}$ at each stage s. At certain stages we will need to find embeddings of $T_s[w_{e,s}]$ into other branches of T, in order to satisfy $\mathcal{R}_e$, and we may redefine f on the nodes in $T_s[w_{e,s}]$.

(We assume that we work with a fixed approximation T_s of T by finite subtrees.) To ensure that $\lim_s f_s(x)$ exists for each $x \in T$, we impose the negative requirements:

$$\mathcal{N}_x : \quad \lim_s f_s(x) \text{ converges.}$$

In addition, for any $y \in T'$, we need to insure that either $\lim_s f_s^{-1}(y)$ exists or y is permanently placed into one of the auxiliary subtrees of T' which are not in the range of f but which occur infinitely often as successor trees of the root in T. In the cases when we use such auxiliary trees, this property will be easy to verify. In the other cases, we explicitly insure this property by meeting the requirements for all $u \in T'$:

$$\mathcal{M}_u : \lim_s f_s^{-1}(u) \text{ converges.}$$

Clearly, satisfying all these requirements will prove the theorem.

By definition, a *successor tree* in T above an ω-branch point x of T is a tree $T[y]$, where y is an immediate successor of x. We use I to stand for an isomorphism type, and say that I *appears finitely often* (resp. *infinitely often*) among the successor trees $\{T[y]\}$ above x if there are only finitely many (resp. infinitely many) immediate successors y of x such that $T[y] \cong I$. In general, when we speak of an isomorphism type I occurring in a tree T, we mean that there is a node $a \in T$ such that $T[a] \cong I$.

The domain of T is always assumed to be ω, and we have a computable approximation to T by:

$$T_s = \{0, 1, \dots s-1\} \cup \{r\},$$

where r is the root of T. We restate Lemma 2.10 because it will be used repeatedly.

Lemma 2.10 *Suppose T and T' are two trees of finite height and strongly finite type. If each of T and T' embeds into the other, then they are isomorphic.*

Lemma 3.2 *Suppose that T is a tree of finite height which is of finite type but not of strongly finite type. Then there exists an ω-branch point $y_0 \in T$ such that all successor trees above y_0 are of strongly finite type, and such that some successor tree which appears only finitely often above y_0 embeds into some successor tree appearing infinitely often above y_0.*

Proof. Let y_0 be maximal in T among nodes which are not of strongly finite type. (This set is non-empty, since it must include the root of T.) Then y_0 must be ω-branching, and every successor tree above y_0 is of strongly finite type. The only way y_0 can fail to be of strongly finite type is for there to be distinct successor trees $T_j \hookrightarrow T_k$ above y_0 such that infinitely many other successor trees above y_0 are isomorphic to T_k. Since y_0 is of finite type, however, the isomorphism type of T_j must appear only finitely often above y_0. ■

Lemma 3.3 *Let $\{T_1, \dots T_n\}$ be any collection of trees of weakly finite type. Then there exist finite trees $S_1, \dots S_n$ such that for all i and j:*

$$S_i \hookrightarrow T_j \iff T_i \hookrightarrow T_j.$$

Proof. To build S_i, consider the set $A_i = \{j \leq n : T_i \not\hookrightarrow T_j\}$. For each $j \in A_i$, there is a finite subtree $S_{i,j} \subseteq T_i$ such that $S_{i,j} \not\hookrightarrow T_j$, by Lemma 2.9. Let S_i be the union of all these subtrees, for all $j \in A_i$. (If A_i is empty, take S_i to be a single node.) ■

We will need a version of Kruskal's Theorem for trees of weakly finite type. In order to prove this theorem, we use labeled finite trees. For our purposes, a *labeled finite tree* is a finite tree S together with a function $l : S \to \{0, 1, \omega\}$. The elements of the set $\{0, 1, \omega\}$ are called labels and the function l is called the labeling function. Let S_1 and S_2 be labeled trees with labeling functions l_1 and l_2. An embedding $f : S_1 \hookrightarrow S_2$ *respects the labels* if for every $x \in S_1$, $l_1(x) \leq l_2(f(x))$. A proof of the following version of Kruskal's Theorem can be found in either [19] or [31]. (In fact, for our purposes, we can assume that there is a uniform finite bound n on the heights of the trees S_i. This assumption leads to a far simpler proof.)

Theorem 3.4 (Kruskal) *Let $\{S_i : i \in \omega\}$ be an infinite collection of finite trees, each with a labeling l_i. Then there exist $i < j$ in ω and an embedding $f : S_i \hookrightarrow S_j$ (preserving the infimum function) such that for every $x \in S_i$, $l_i(x) \leq l_j(f(x))$.*

Lemma 3.5 (Kruskal's Theorem for weakly finite type) *Fix $n \in \omega$, and let $\{T_i : i \in \omega\}$ be an infinite collection of trees of weakly finite type, with $\mathrm{ht}(T_i) \leq n$ for all i. Then there exist $i < j$ in ω such that T_i can be embedded in T_j.*

Corollary 3.6 *Let $\{T_i : i \in \omega\}$ be an infinite collection of trees of weakly finite type, with $\mathrm{ht}(T_i) \leq n$ for all i. Then there exists $m \in \omega$ such that for every $i > m$, T_i can be embedded in some T_j with $j > i$, and some T_k with $k < i$ can be embedded in T_i.*

Proof. By Lemma 3.5 both $\{\, i \in \omega \mid \forall j > i\, (T_i \not\hookrightarrow T_j)\,\}$ and $\{\, i \in \omega \mid \forall k < i\, (T_k \not\hookrightarrow T_i)\,\}$ must be finite. ■

Proof of Lemma 3.5. We claim that given the collection $\{T_i\}$, we can build a corresponding collection $\{S_i\}$ of labeled finite trees such that if $i < j$ and there is an embedding of S_i into S_j which respects the labels, then T_i also embeds into T_j. To prove this claim, we induct on n. The case $n = 1$ is easy, since there is only one possible tree, containing a single node; we take $S_i = T_i$ for each i, labeling the node of each S_i with 1.

Now assume the claim for n. For each tree T_i given by the lemma, let r_i be the root of T_i, and let $I_{i,1}, \ldots I_{i,m_i}$ be the (finitely many) distinct isomorphism types of successor trees above r_i. Then the inductive hypothesis applies to the set

$$\{I_{i,k} : i \in \omega,\ 1 \leq k \leq m_i\},$$

yielding a set $\{S_{i,k}\}$ of corresponding labeled finite trees. Define S_i inductively as follows:

- S_i has a root s_i, labeled with 1;

- S_i has a chain $u_{i,1} \prec u_{i,2} \prec \cdots \prec u_{i,n}$, each labeled with 0, and with $u_{i,1}$ an immediate successor of s_i

- For each isomorphism type $I_{i,k}$ which appears only finitely often – say p times – among the successor trees above r_i in T_i, add p copies of the corresponding $S_{i,k}$ as successor trees above s_i in S_i, with the root of each of these successor trees labeled with a 1; and

- For each isomorphism type $I_{i,k}$ which appears infinitely often among the successor trees above r_i in T_i, add a copy of the corresponding $S_{i,k}$ as a successor tree above s_i in S_i, but labeling its root with ω, rather than 1.

We have changed the labels on the roots of certain finite successor trees $S_{i,k}$, but only by changing the label of the root from 1 to ω, so we have not introduced any new embeddings among the $S_{i,k}$'s.

Now if f is an embedding of S_i into S_j ($j > i$) which preserves infima and respects labels, then f must map the root s_i to s_j, since both trees have height n. (This was the purpose of the chains $\{u_{i,k}\}$ and $\{u_{j,k}\}$.) Hence each successor tree in S_i maps into a distinct successor tree in S_j, since f preserves infima. It follows that each isomorphism type among the successor trees in T_i maps into some successor tree in T_j. Because of the labeling with 0, 1 and ω on S_i, we know that no $S_{i,k}$ maps into the chain above $u_{j,1}$, and that each infinite-appearing successor tree in T_i maps into an infinite-appearing successor tree in T_j (or possibly into an infinite-appearing subtree of a finite-appearing successor tree in T_j). Finally, each finite-appearing successor tree in T_i appeared just as many times in S_i, and hence there must be sufficiently many successor trees in T_j for each copy of the type to map to.

Applying Theorem 3.4 to our set $\{S_i\}$, we get an f with precisely the properties required by the claim. Hence some T_i embeds into some T_j with $j > i$. ■

We will be interested in the minimal elements (under embedding) of various sets of trees of weakly finite type. Let $\mathcal{T}$ be a set of trees of weakly finite type for which there is a finite bound on the height of the trees appearing in $\mathcal{T}$. Lemma 3.5 says that $\mathcal{T}$ together with the embeddability relation forms a well-quasi-order. Therefore, $\mathcal{T}$ satisfies both the *descending chain condition* that any strictly descending chain in $\mathcal{T}$ under embeddability is finite and the *incomparable chain condition* that any anti-chain under the embeddability relation is finite. (See [19] for more details on these properties. We are using the fact that a quasi-order is a well-quasi-order if and only if it satisfies both of these conditions.) In this context, the appropriate definition of "minimal" is based on equivalence classes under $\equiv$ rather than under $\cong$.

Definition 3.7 $T \in \mathcal{T}$ is *minimal in* $\mathcal{T}$ if for every $T' \in \mathcal{T}$ such that $T' \hookrightarrow T$, we have $T \equiv T'$.

For trees of strongly finite type, this is equivalent to the standard definition of minimal under $\cong$, by Lemma 2.10. However, trees of weakly finite type do not necessarily satisfy this lemma; they form only a quasi-order under $\hookrightarrow$, not a partial order. (The notion of a quasi-order plays no explicit role in the paper after the next corollary.)

Corollary 3.8 *Let $\mathcal{T}$ be an infinite collection of trees of weakly finite type, with* $\mathrm{ht}(T) \leq n$ *for all $T \in \mathcal{T}$. Then there exists a finite set $\mathcal{M} \subseteq \mathcal{T}$ of minimal elements of $\mathcal{T}$ (under the embedding relation) such that for every $T \in \mathcal{T}$ there exists $T' \in \mathcal{M}$ with $T' \hookrightarrow T$.*

Proof. Let $\mathcal{S}$ be the set of all minimal elements of $\mathcal{T}$, and let $\mathcal{M}$ contain exactly one representative from each $\equiv$-equivalence class in $\mathcal{S}$. Then the incomparable chain condition implies that $\mathcal{M}$ must be finite, and the descending chain condition implies that every $T \in \mathcal{T}$ contains a subtree from $\mathcal{M}$. ■

This corollary will frequently be applied with $\mathcal{T}$ being either the set $\{T : T \not\hookrightarrow T_0\}$ (for some fixed T_0) or the set $\{T : T_0 \hookrightarrow T \ \&\ T_0 \not\cong T\}$. We will need one last general fact about embeddings between finite height trees of finite type.

Lemma 3.9 *Let T_0 and T_1 be finite height trees of finite type and let $\mathcal{I}_i$ be the finite set of isomorphism types of successor trees which occur infinitely often immediately above the root in T_i. If $T_0 \equiv T_1$, then $\mathcal{I}_0 = \mathcal{I}_1$.*

Proof. We will write $T_i[x] \in \mathcal{I}_i$ to indicate that the tree $T_i[x]$ is a successor tree of the root in T_i and that its isomorphism type occurs in $\mathcal{I}_i$.

Consider any $T_0[x] \in \mathcal{I}_0$ and suppose that $T_0[x]$ embeds in some $T_1[y] \in \mathcal{I}_1$. Further, suppose that $T_1[y]$ embeds in some $T_0[z] \in \mathcal{I}_0$. Composing these two embeddings gives that $T_0[x] \hookrightarrow T_0[z]$. But, because T_0 has finite type and both $T_0[x]$ and $T_0[z]$ occur infinitely often, it must be that $T_0[x] \cong T_0[z]$. Furthermore, $T_0[x]$, $T_1[y]$ and $T_0[z]$ all have strongly finite type since they occur infinitely often. The two embeddings $T_0[x] \hookrightarrow T_1[y] \hookrightarrow T_0[z] \cong T_0[x]$ show that $T_0[x] \cong T_1[y]$ by Lemma 2.10.

The argument in the previous paragraph establishes the lemma except in the case when there is a type $I \in \mathcal{I}_i$ which does not embed in any type $J \in \mathcal{I}_{1-i}$. Without loss of generality, assume that there is a type $I \in \mathcal{I}_0$ which does not embed in any type in $\mathcal{I}_1$. In this case we will derive a contradiction to the fact that T_0 and T_1 have finite height.

For any $c \in T_j$, we say that c occurs in the *finite part* of T_j if $c \in T_j[a]$ for some a at level 1 of T_j for which $T_j[a]$ is one of the finitely occurring isomorphism types at level 1. Otherwise, we say that c occurs in the *infinite part* of T_j.

We define a notion of rank for our fixed isomorphism type I. A node $a \in T_j$ has rank $\mathrm{rk}(a) \geq 0$ if I embeds in $T_j[a]$. A node a has rank $\mathrm{rk}(a) \geq n+1$ if there are infinitely many nodes c for which $a \prec c$ and $\mathrm{rk}(c) \geq n$. There are a number of simple facts that follow from this definition.

Fact 1. If $\mathrm{rk}(a) \geq n$, then $\mathrm{ht}(T_j[a]) \geq \mathrm{ht}(I) + n$. This fact follows by an induction on n.

Fact 2. If c lies above the root in T_1 and $\mathrm{rk}(c) \geq 0$, then c is in the finite part of T_1. This fact follows since $I \hookrightarrow T_1[c]$ and $\mathcal{I}_1$ contains no isomorphism types into which I embeds.

Fact 3. If c lies above the root in T_0 and $\mathrm{rk}(c) \geq 1$, then c occurs in the finite part of T_0. For a contradiction, assume that c occurs in the infinite part of T_0. Then c must occur in some successor tree $T_0[a]$ of the root whose isomorphism type is in $\mathcal{I}_0$. However, by Fact 1, $\mathrm{ht}(T_0[a]) > \mathrm{ht}(I)$, so $T_0[a] \not\cong I$. This means $I \hookrightarrow T_0[a]$ is an embedding relation between distinct infinitely occurring isomorphism types at level 1 of T_0, which contradicts the fact that T_0 has finite type.

Fact 4. For any $k, n \geq 1$, if $a_1, \ldots, a_k \in T_j$ satisfy $\mathrm{rk}(a_i) \geq n$, then there are $b_1, \ldots, b_k \in T_{1-j}$ which satisfy $\mathrm{rk}(b_i) \geq n$. To establish this fact, fix any embedding $f : T_j \hookrightarrow T_{1-j}$ and let $b_i = f(a_i)$. The result follows by induction on n.

By Fact 1, to derive a contradiction with the fact that T_1 has finite height, it suffices to show that for each $n \geq 1$, T_1 must have a node a with $\mathrm{rk}(a) \geq n$. We establish this by induction on n. Fix embeddings $f : T_0 \hookrightarrow T_1$ and $g : T_1 \hookrightarrow T_0$.

For the case of $n = 1$, we claim that the fact that I does not embed into any element of $\mathcal{I}_1$ implies that there is an $a \in T_1$ for which $\mathrm{rk}(a) \geq 1$. To prove this claim, consider the embedding $f : T_0 \hookrightarrow T_1$. Each copy $T_0[d]$ of I in the infinite part of T_0 must map into the finite part of T_1 under f. Therefore, there must be a node a at level 1 of T_1 (in the finite part) for which infinitely many copies of I embed into $T_1[a]$. For this a, $\mathrm{rk}(a) \geq 1$.

Assume by induction that the fact that I does not embed into any element of $\mathcal{I}_1$ implies that there is an $a \in T_1$ with $\mathrm{rk}(a) \geq n$.

Fix $a_1 \in T_1$ such that $\mathrm{rk}(a_1) \geq n$. We claim that there is an element $a_2 \in T_1$ with $a_2 \neq a_1$ and $\mathrm{rk}(a_2) \geq n$. To prove this claim, notice that by Fact 4, we know that $g(a_1) = b_1 \in T_0$ satisfies $\mathrm{rk}(b_1) \geq n$. Furthermore, setting $c_1 = f(b_1)$, we have that $\mathrm{rk}(c_1) \geq n$.

We split into two cases depending on whether $a_1 = c_1$ or $a_1 \neq c_1$. If $a_1 \neq c_1$, then we let $a_2 = c_1$ and we are done with the claim. Otherwise, if $a_1 = c_1$, then $f(b_1) = a_1$ and $g(a_1) = b_1$. Therefore f maps $T_0[b_1]$ into $T_1[a_1]$ and g maps $T_1[a_1]$ into $T_0[b_1]$, and moreover, restricting f to $T_0 \setminus T_0[b_1]$ and restricting g to $T_1 \setminus T_1[a_1]$ shows that $(T_0 \setminus T_0[b_1]) \equiv (T_1 \setminus T_1[a_1])$. By the induction hypothesis (which applies since we have only removed a portion of the finite parts of T_0 and T_1 by Facts 2 and 3 and therefore not changed the infinite part of either tree), there is an element $a_2 \in T_1 \setminus T_1[a_1]$ for which $\mathrm{rk}(a_2) \geq n$. This establishes the claim.

More generally, for any $k \geq 1$, if $a_1, \ldots, a_k \in T_1$ have $\mathrm{rk}(a_i) \geq n$, then we can fix $g(a_1) = b_1, \ldots, g(a_k) = b_k \in T_0$ with $\mathrm{rk}(b_i) \geq n$. Setting $c_i = f(b_i)$ and splitting into cases as above, we get the existence of $a_{k+1} \in T_1$ with $\mathrm{rk}(a_{k+1}) \geq n$. Therefore, there must be infinitely many nodes in the finite part of T_1 which have rank $\geq n$. We can fix a node a at level 1 of T_1 (and in the finite part) for which infinitely many of these nodes occur in $T_1[a]$. Then, $\mathrm{rk}(a) \geq n + 1$ as required. ■

We now begin the constructions which will ultimately prove Proposition 3.1. Suppose T has finite height, but is not of finite type. In this section we consider the case when the root r is the only node of T which fails to be of finite type. Then T must be ω-branching at r, and we write $x_0, x_1, \ldots$ for the immediate successors of r in T. We present several constructions concerning various ways in which T could fail to have finite type and we prove in each case that T is not computably categorical. After these constructions, we prove Proposition 3.1 by showing that we have considered all possible cases. We present the first proof in the most detail since many of the later arguments will have similar features.

Lemma 3.10 *Let T be a tree of finite height with root r, and suppose that each node above r in T is of finite type. Suppose there is an isomorphism type I_0 which is not of strongly finite type and appears infinitely often as a successor tree above r. If only finitely many other isomorphism types I' appearing above r satisfy $I' \hookrightarrow I_0$, then T is not computably categorical.*

Proof. First, we establish that there is a Δ^0_2 procedure which identifies the immediate successors x of r such that $T[x] \equiv I_0$. Let $\mathcal{F}$ be the set of all isomorphism types I appearing as successor trees above r such that $I \not\hookrightarrow I_0$. Let $\{I_1, \dots I_m\}$ be a set of minimal elements (under $\hookrightarrow$) of $\mathcal{F}$ as given by Corollary 3.8, so that every $I \in \mathcal{F}$ is a supertree of some I_i. By Lemma 3.3, each I_i $(i > 0)$ contains a finite subtree S_i such that $S_i \not\hookrightarrow I_0$. Therefore, a node x at level 1 in T satisfies $T[x] \hookrightarrow I_0$ if and only if $\forall s \forall i \leq m(S_i \not\hookrightarrow T_s[x])$.

Consider the finite number of isomorphism types $I' \hookrightarrow I_0$. For each such I' for which $I_0 \not\hookrightarrow I'$, there is a finite subtree Q' of I_0 such that $Q' \not\hookrightarrow I'$. Taking the union of these finite trees Q' gives a finite tree Q such that $Q \hookrightarrow I_0$ and for all I' as above, $Q \not\hookrightarrow I'$. Therefore, an immediate successor x of r satisfies $T[x] \equiv I_0$ if and only if

$$\forall s \forall i \leq m(S_i \not\hookrightarrow T_s[x]) \wedge \exists s(Q \hookrightarrow T_s[x]).$$

Since we can clearly identify the immediate successors of r in a Δ^0_2 manner, this definition shows that we can identify the immediate successors x of r which satisfy $T[x] \equiv I_0$ with a Δ^0_2 procedure.

During the construction, we try to identify trees $T[x]$ for which x is an immediate successor of r and $T[x] \equiv I_0$. We say that we believe $T[x] \equiv I_0$ at stage s if x is an immediate successor of r in T_s, $Q \hookrightarrow T_s[x]$ and $S_i \not\hookrightarrow T_s[x]$ for all $i \leq m$. Without loss of generality, we assume that the finite tree Q has the same height as I_0. Therefore, if we believe $T[x] \equiv I_0$ at stage s, then any embedding of $T_s[x]$ into I_0 must send x to the root node of I_0.

Since it is of finite type and not of strongly finite type, I_0 must contain an ω-branching node y_0 satisfying Lemma 3.2. Fix such a node y_0 and let J denote the isomorphism type of $I_0[y_0]$. By Lemma 3.2, there exist successor isomorphism types $J_0 \hookrightarrow J_1$ of J with J_0 occurring only finitely often above y_0 and J_1 occurring infinitely often above y_0. Moreover, Lemma 3.2 ensures that both J_0 and J_1 are of strongly finite type, so Lemma 2.10 guarantees that $J_1 \not\hookrightarrow J_0$.

We will build a tree T' and an embedding $f : T \to T'$ such that T' is equal to the range of f together with infinitely many copies of I_0 which are attached to the root of T'. Our strategy to diagonalize against $\varphi_e : T \to T'$ being an isomorphism will roughly be to identify subtrees $T[z]$ which satisfy $T[z] \equiv J_0$. Once we find such a subtree, we make sure that J_1 embeds into our subtree $T'[\varphi_e(z)]$. Because $J_1 \not\hookrightarrow J_0$, it cannot be that $T'[\varphi_e(z)] \cong T[z]$. (In fact, it cannot even be that $T'[\varphi_e(z)] \hookrightarrow T[z]$.)

Consider the finite number of isomorphism types I which occur as successor trees in T and which satisfy $I \equiv I_0$. By Lemma 2.6, each such I contains only finitely many isomorphism types of the form $I[a]$ for $a \in I$. Therefore, we can list the types of all such $I[a]$ where I is as above and $a \in I$ as $K_1, \ldots, K_n$. (This list includes the types of trees $I[a]$ of any level, not just the types of the successor trees of the root of I.) As above, we can use finite trees to distinguish these types up to $\equiv$-equivalence. That is, for K_1 there are finitely many types $K_i \not\hookrightarrow K_1$. List these trees as $K_{i_1}, \ldots, K_{i_l}$. For each $k \leq l$, there is a finite subtree S'_{i_k} of K_{i_k} such that $S'_{i_k} \not\hookrightarrow K_1$. Similarly, there are finitely many K_j such that $K_1 \not\hookrightarrow K_j$. For each such K_j, there is a finite subtree S''_j of K_1 such that $S''_j \not\hookrightarrow K_j$. Taking the union of these trees yields a finite subtree S'' of K_1 such that $S'' \not\hookrightarrow K_j$ for any such j. Therefore, if $T[x] \equiv I_0$ and $x \prec u$, then $T[u] \equiv K_1$ if and only if

$$\forall s \forall k \leq l (S'_{i_k} \not\hookrightarrow T_s[u]) \wedge \exists s (S'' \hookrightarrow T_s[u]).$$

At stage s, we believe that $T_s[u] \equiv K_1$ if and only if the following conditions are satisfied: there is a node $x \prec u$ in T_s such that we believe $T_s[x] \equiv I_0$; $S'' \hookrightarrow T_s[u]$; and for all $k \leq l$, $S'_{i_k} \not\hookrightarrow T_s[u]$.

We can perform similar calculations for the other types K_i. During the construction, if x is an immediate successor of r with $x \prec u$ for some u and we believe $T[x] \equiv I_0$, then we can determine using finite trees which K_j we believe satisfies $T[u] \equiv K_j$ (if any). From the nature of the conditions, it is clear that if we believe infinitely often that $T[x] \equiv I_0$ and $T[u] \equiv K_j$, then in fact these equivalences hold.

Notice that the isomorphism types J, J_0, and J_1 occur among the types $K_1, \ldots, K_n$. Furthermore, by Lemma 3.9, for any node y such that $T[y] \equiv J$, $T[y]$ must have infinitely many successor trees which are isomorphic to J_1.

We begin to describe the construction. Assume that T is approximated in finite stages by T_s. Without loss of generality, we assume that we know the root of T. We build T'_s and a sequence of embeddings $f_s : T_s \to T'_s$. T'_s will consist of the range of f_s (which is a finite tree isomorphic to T_s by f_s) together with finitely many subtrees which are isomorphic to I_0 and which are attached to the root of T'_s.

We say that a tuple $\langle x, y, z \rangle$ is special in T if x is an immediate successor of the root, $T[x] \equiv I_0$, $x \prec y$, $T[y] \equiv J$, z is an immediate successor of y, and $T[z] \equiv J_0$. We say that $\langle x, y, z \rangle$ is special at stage $s+1$ if $x, y, z \in T_s$ and we believe all of these conditions at stage s. Because we have Δ^0_2 approximations to these conditions, we know that $\langle x, y, z \rangle$ is special in T if and only if there is a stage s such that for all $t \geq s$, $\langle x, y, z \rangle$ is special at stage t.

Recall that we have requirements $\mathcal{R}_e$, which attempts to diagonalize against φ_e being an isomorphism, and $\mathcal{N}_u$, which attempts to make the limit of $f_s(u)$ defined so that f is Δ^0_2. The basic strategy for $\mathcal{R}_e$ is to define a witness tuple $\langle x, y, z \rangle$ which we believe is special and wait for $\varphi_e(x) = x'$, $\varphi_e(y) = y'$, and $\varphi_e(z) = z'$ to converge. We next want to determine if we believe that $\langle x', y', z' \rangle$ is going to be special in T'. This amounts to letting $f_s^{-1}(x') = a$, $f_s^{-1}(y') = b$ and $f_s^{-1}(z') = c$, and asking whether we currently believe $\langle a, b, c \rangle$ is special in T. If the answer is no, then it appears that $\mathcal{R}_e$ is not an isomorphism and we do not need to diagonalize. If the answer is yes, then we want to actively diagonalize.

Assume that not only do we believe that $\langle x, y, z \rangle$ and $\langle a, b, c \rangle$ are special in T at stage s of the construction, but they really are special in T. Then, there is an embedding of $T_s[a] \hookrightarrow I_0$

which maps c to the base of a tree of type J_1. (We prove the existence of such an embedding below when we do the formal construction.) Therefore, we can diagonalize by using the embedding of $T_s[a]$ into I_0 to turn $T'_s[x']$ into a copy of I_0 for which $T'_s[z']$ becomes a tree into which J_1 embeds. We now know $J_1 \hookrightarrow T'[z']$ and $T[z] \hookrightarrow J_0$. Therefore, if φ_e is an isomorphism, then $T'[z'] \hookrightarrow T[z]$ and hence $J_1 \hookrightarrow J_0$, which contradicts our choice of J_0.

We also need to redefine f_{s+1} on $T_s[a]$ since we have turned $T'_s[x']$ into I_0 and we do not know that $T_s[a]$ has isomorphism type I_0. Therefore, we add new elements to T'_{s+1} and define the map f_{s+1} to send the tree $T_s[a]$ to these new elements. Notice that once an element $y' \in T'$ has left the range of f, it will never return to the range of f. Therefore, we have that for all $y' \in T'$, either $f_s^{-1}(y')$ converges or y' is permanently part of an auxiliary copy of I_0. Hence we do not need to explicitly discuss the requirements $\mathcal{M}_u$ in this construction.

This action of changing the map so that $f_s(a) \neq f_{s+1}(a)$ conflicts with the requirements of the form $\mathcal{N}_u$ for $u \in T_s[a]$. To fix this problem, we give $\mathcal{R}_e$ $e+1$ many witness tuples with distinct first components and we do not allow $\mathcal{R}_e$ to use the tuple $\langle x, y, z\rangle$ to diagonalize if $f_s^{-1}(\varphi_e(x)) \prec u$ for any $u \leq e$. That is, we give $\mathcal{N}_u$ higher priority than $\mathcal{R}_e$ if $u < e$. Since each $\mathcal{N}_u$ can stop $\mathcal{R}_e$ from using at most one node at level one with which to diagonalize, assigning $e+1$ tuples to $\mathcal{R}_e$ is enough to guarantee that either $\mathcal{R}_e$ will be allowed to diagonalize with one of these tuples or $\mathcal{R}_e$ will be satisfied in a trivial way, such as φ_e not being one-to-one or not respecting the ordering.

There is a second possible worry for the basic strategy for $\mathcal{R}_e$. Assume that $\langle x, y, z\rangle$ is a witness tuple for $\mathcal{R}_e$ and $\langle x', y', z'\rangle$ is as above. Since T'_s consists of the range of f_s together with additional copies of I_0, it is possible that the elements of $\langle x', y', z'\rangle$ sit in one of the copies of I_0. In this case, it makes no sense to look at f_s^{-1} on the values x', y' and z' since these elements are not in the range of f_s. Of course, if it is not the case that $x' \prec y' \prec z'$ in T'_s, then we have beaten φ_e trivially. Otherwise, $\mathcal{R}_e$ can check if x' is the root of the copy of I_0, y' is the root of a tree which is $\equiv J$ and z' is the root of a tree which is $\equiv J_0$. (We will assume that when we add a copy of I_0 to T', we add a "nice" copy in which we know the isomorphism type of each subtree of the form $T'[a]$ for a in this copy of I_0. This is possible since I_0 contains only finitely many such isomorphism types.) If not, then $\mathcal{R}_e$ has already won. If so, then $\mathcal{R}_e$ can win by turning $T'[z']$ into a copy of J_1 (which is possible because $J_0 \hookrightarrow J_1$) and adding a new copy of the old $T'[z']$ above $T'[y']$. Because $T'[y']$ bounds infinitely many copies of J_1, adding an extra copy does not change its isomorphism type. So, we still have $T'[y'] \equiv J$ and $T'[x'] \cong I_0$. This action wins $\mathcal{R}_e$ as above, since $J_1 \hookrightarrow T'[z']$ and $T[z] \hookrightarrow J_0$.

However, notice that if we performed this action infinitely often with the same copy of I_0, then we might move the same subtree of this copy of I_0 infinitely often and not have a copy of I_0 in the limit. Therefore, we need to restrict this action from happening infinitely often. When a requirement $\mathcal{R}_e$ creates a copy of I_0, it marks it with the same priority as $\mathcal{R}_e$. We only allow requirements $\mathcal{R}_i$ with $i < e$ to diagonalize using this copy of I_0 as described above. This causes only finitely additions to the tree after it is defined, so it really does have type I_0 in the limit.

Because $\mathcal{R}_e$ is not allowed to use a witness tuple $\langle x, y, z\rangle$ when $\varphi_e(x)$ converges to the root of a copy of I_0 which is marked by a requirement of higher priority, we have to allow $\mathcal{R}_e$ extra witness tuples. Each time $\mathcal{R}_i$ with $i < e$ marks a new copy of I_0, $\mathcal{R}_e$ is given an extra witness tuple. This action will only occur finitely often, so in the end, $\mathcal{R}_e$ has $e+n+1$ many witness tuples, where n is the number of copies of I_0 marked by requirements of higher priority. We cannot fix the number n at the beginning of the construction since finitely often a requirement of higher priority may create a copy of I_0 thinking that it is diagonalizing against a particular witness tuple only to discover later that this witness tuple was not actually special.

We turn to a more formal description of the construction. At each stage s, we define a finite list of tuples which are special at that stage and which have distinct first components. More formally, let $\langle x_0, y_0, z_0\rangle$ be the $\leq_{\mathbb{N}}$-least tuple (under a fixed coding of $\mathbb{N}^3$) that is special at stage s. Let $\langle x_{i+1}, y_{i+1}, z_{i+1}\rangle$ be the $\leq_{\mathbb{N}}$-least tuple greater than $\langle x_i, y_i, z_i\rangle$ which is special at stage s and such that x_{i+1} is not equal to x_j for any $j \leq i$. List these tuples at stage s by

$$\langle x_{0,s}, y_{0,s}, z_{0,s}\rangle, \ldots, \langle x_{p_s,s}, y_{p_s,s}, z_{p_s,s}\rangle.$$

At stage s, we assign $e+n+1$ many of these tuples to the requirement $\mathcal{R}_e$, where n is the number of copies of I_0 created by requirements of higher priority by stage s. If there are not enough tuples for $\mathcal{R}_e$ to get its full set of tuples, then it is not assigned any tuples. $\mathcal{R}_e$ may be later declared to be satisfied by one of the tuples it has been assigned. If that tuple ever changes, then $\mathcal{R}_e$ is said to be injured and it is no longer satisfied.

Because there are infinitely many copies of I_0 attached to the root of T, there is an infinite set of special tuples in T which have pairwise distinct first components. Therefore, each tuple of the form $\langle x_{m,s}, y_{m,s}, z_{m,s}\rangle$ is eventually defined and reaches a limit $\langle x_m, y_m, z_m\rangle$, which is a special tuple in T. So, each requirement $\mathcal{R}_e$ is eventually assigned a complete set of special tuples.

At stage $s+1$ we extend the isomorphism f_s to $f_{s+1} : T_{s+1} \to T'_{s+1}$ by adding fresh elements to T'_{s+1}, unless there exists an $\mathcal{R}_e$ requirement with $e \leq s$ which requires attention. If some $\mathcal{R}_e$ requires attention, we let the highest priority such requirement act. Below we define when $\mathcal{R}_e$ requires attention and what action $\mathcal{R}_e$ takes when it is allowed to act.

Assume that $\mathcal{R}_e$ is assigned the tuple $\langle x, y, z\rangle$ during the construction. $\mathcal{R}_e$ waits for $\varphi_e(x)$, $\varphi_e(y)$ and $\varphi_e(z)$ to converge to some x', y', and z' respectively. Once these computations converge (say at stage $s+1$), $\mathcal{R}_e$ checks for two possible easy wins. First, if any of the elements is not in T'_s, then $\mathcal{R}_e$ wins by making sure that these elements are all placed in T'_{s+1} and they do not satisfy $x' \prec y' \prec z'$. Second, if either φ_e is not one-to-one or x' is equal to the root in T' or it is not the case that $x' \prec y' \prec z'$, then φ_e cannot be an isomorphism and $\mathcal{R}_e$ wins without performing any action.

Assume that $\mathcal{R}_e$ does not win trivially and that it has not been declared satisfied by one of its tuples. The action for $\mathcal{R}_e$ splits into two cases. Either x', y' and z' are all in the range of f_s or else one of these elements falls outside the range of f_s. We first consider the case when all the elements are in the range of f_s. In this case, $\mathcal{R}_e$ checks if $\langle f_s^{-1}(x'), f_s^{-1}(y'), f_s^{-1}(z')\rangle$ is special. If not, then $\mathcal{R}_e$ does not require attention. If this tuple is special, then $\mathcal{R}_e$ checks if there is an element $u \in T$ with $u < e$ and $f_s^{-1}(x') \prec u$. If so, then the requirement $\mathcal{N}_u$ takes precedence over $\mathcal{R}_e$ and prevents $\mathcal{R}_e$ from acting. Otherwise, $\mathcal{R}_e$ requires attention.

When $\mathcal{R}_e$ is allowed to act, it begins a concurrent search for one of the following:

1. an embedding $T'_s[x'] \hookrightarrow I_0$ which sends x' to the root of I_0 and sends z' to the base of a tree into which the isomorphism type J_1 embeds; or
2. a stage $t \geq s$ at which we no longer believe $\langle f_s^{-1}(x'), f_s^{-1}(y'), f_s^{-1}(z')\rangle$ is special.

One comment is necessary to explain condition (1). Because I_0 has finite type and is computable, we can assume that we have a nice copy of I_0 in which we know the isomorphism type of each subtree $I_0[a]$. Therefore, we know which subtrees J_1 embeds into, so we can search for an embedding as in (1) in an effective manner. In the sublemmas verifying this construction, we show that this search procedure must terminate.

If we see (2) happen first, then we no longer think that $\mathcal{R}_e$ requires attention with this tuple. We extend our tree to T'_{s+1} as if no requirement had required attention and go to the next stage. If we see (1) happen first, then we perform the following actions:

- turn $T'_s[x']$ into a copy of I_0 with x' as the root and z' as the root of a tree into which J_1 embeds; and
- add extra elements to T' to be the new images of the elements above $f_s^{-1}(x')$ in T under f_{s+1}; and
- leave $f_{s+1} = f_s$ on all other elements from T; and
- add new elements to T' to correspond to the elements in $T_{s+1} \setminus T_s$ and define f_{s+1} in the obvious way;
- declare $\mathcal{R}_e$ satisfied with this tuple.

In this case, we say that $\mathcal{R}_e$ acts at this stage. Notice that the elements in the new copy of I_0 in T' are outside of the range of f_{s+1}. Also, as in the comments explaining the search in (1) above, we know exactly how this copy of I_0 is constructed in the sense that we know the isomorphism type of each subtree. Furthermore, if $\mathcal{R}_e$ is never injured after this stage, then $\langle x, y, z\rangle$ is one of the final tuples assigned to $\mathcal{R}_e$. In this case, $T[z] \equiv J_0$ but $J_1 \hookrightarrow T'[z']$. Since $\varphi_e(z) = z'$ and $J_1 \not\hookrightarrow J_0$, φ_e cannot be an isomorphism.

We still need to see what action to take if one of the elements x', y' or z' is not in the range of f_s. Because we did not get an easy win for $\mathcal{R}_e$, it must be the case that $x' \prec y' \prec z'$ all sit in some successor tree of the root in T'_s. Since one of these elements is not in the range of f_s, this successor tree must be one of the trees of type I_0 added to T'_s. If this copy of I_0 was created by a requirement of higher priority than $\mathcal{R}_e$, then $\mathcal{R}_e$ does not act at this stage and it ignores this particular witness tuple in future calculations (since it already knows that it is not allowed to diagonalize with this tuple). Otherwise, because we know how such a copy of I_0 was constructed, we can check whether x' is the root of this copy of I_0, whether y' has infinitely many successor trees of type J_1, whether z' is an immediate successor of y', and whether z' is the base of a subtree which is $\equiv J_0$. If any of these conditions fail, then we say that $\mathcal{R}_e$ is satisfied by this tuple since φ_e does not appear to be an isomorphism. If all of these conditions hold, then we add a new subtree above y' of the same type as the subtree above z' and we add elements to the subtree above z' to turn it into a copy of J_1. (In this case, we say that $\mathcal{R}_e$ acts at this stage.) Because y' must bound infinitely many copies of J_1, we still have a tree of type I_0 and now we have diagonalized against φ_e being an isomorphism. We declare $\mathcal{R}_e$ satisfied by this tuple.

This completes the description of the construction. We verify that it succeeds in the following sublemmas.

Sublemma 3.11 *In the case when x', y' and z' are in the range of f_s, the concurrent search procedure between (1) and (2) terminates.*

Proof. Assume that (2) does not occur. Then, $f_s^{-1}(x')$ is an immediate successor of r in T, $f_s^{-1}(z')$ is an immediate successor of $f_s^{-1}(y')$, $T[f_s^{-1}(x')] \equiv I_0$, $T[f_s^{-1}(y')] \equiv J$, and $T[f_s^{-1}(z')] \equiv J_0$. Because $T[f_s^{-1}(x')] \equiv I_0$, we can fix an embedding $\psi : T[f_s^{-1}(x')] \hookrightarrow I_0$. We have already observed that $\psi(f_s^{-1}(x'))$ is the root of I_0. Because $T[f_s^{-1}(y')] \equiv J$, we know that $f_s^{-1}(y')$ has infinitely many successor trees of type J_1. Since $T[f_s^{-1}(z')] \equiv J_0$, we know that $T[f_s^{-1}(z')] \hookrightarrow J_1$.

Consider the restriction of ψ to the finite tree $T_s[f_s^{-1}(x')]$. Let a be a node at level 1 in $T[f_s^{-1}(y')]$ which has type J_1 and is not in $T_s[f_s^{-1}(y')]$. Fix an embedding ξ of $T_s[f_s^{-1}(z')]$ into $T[a]$ which sends $f_s^{-1}(z')$ to a and let $\psi(a) = b \in I$. Notice that $J_1 \hookrightarrow I[b]$ and $\psi\xi$ maps $T_s[f_s^{-1}(z')]$ into $I[b]$ with $\psi(\xi(f_s^{-1}(z'))) = b$. Define ψ' on $T_s[f_s^{-1}(x')]$ by making it equal to ψ for all nodes that are not in $T_s[f_s^{-1}(z')]$ and equal to $\psi\xi$ on all nodes in $T_s[f_s^{-1}(z')]$. Because f_s is an isomorphism between the finite trees $T_s'[x']$ and $T_s[f_s^{-1}(x')]$, we can abuse notation and view ψ' as an embedding from $T_s'[x']$ into I_0. Notice that ψ' has exactly the properties required for condition (1). ∎

Sublemma 3.12 *Each $\mathcal{R}_e$ requirement only acts finitely often.*

Proof. Let s be a stage after which all R_i for $i < e$ do not act. In particular, they do not create new copies of I_0, so the number of witness tuples required by $\mathcal{R}_e$ is fixed at this stage. Let $t \geq s$ be a stage at which $\mathcal{R}_e$ has a full set of witness tuples and each such tuple is actually special in T. Suppose $\mathcal{R}_e$ acts with the tuple $\langle x, y, z\rangle$ after stage t. Then $\mathcal{R}_e$ declares itself satisfied with this tuple and remains satisfied forever since $\langle x, y, z\rangle$ is never taken away. Therefore, $\mathcal{R}_e$ acts at most once after stage t. ∎

Sublemma 3.13 *For each $u \in T$, $f_s(u)$ reaches a limit as $s \to \infty$.*

Proof. The value of $f_s(u)$ can only change if some requirement $\mathcal{R}_e$ diagonalizes using a witness x such that $f_s^{-1}(\varphi_e(x)) \prec u$. However, only requirements $\mathcal{R}_e$ with $e \leq u$ can act in this way. Therefore, once these requirements have stopped acting, the value of $f_s(u)$ cannot change. ∎

Sublemma 3.14 *Each requirement $\mathcal{R}_e$ is eventually satisfied.*

Proof. Let n be the number of copies of I_0 created during the construction by requirements of higher priority. Let s be a stage at which $\mathcal{R}_e$ has been assigned its final set of tuples, $\langle x_i, y_i, z_i \rangle$ for $i < e+n+1$, and all requirements of higher priority have stopped acting. For a contradiction, assume that φ_e is an isomorphism from T to T'. Let $t \geq s$ be a stage at which φ_e has converged on all entries in the tuples assigned to $\mathcal{R}_e$. Let $\langle x'_i, y'_i, z'_i \rangle$ denote these image tuples. Since $\mathcal{R}_e$ did not get an easy win, we can assume that the values in these image tuples are either in the range of f_t or in a copy of I_0 constructed by stage t.

For any tuple $\langle x_i, y_i, z_i \rangle$ which is mapped by φ_e into a copy of I_0, the image tuple $\langle x'_i, y'_i, z'_i \rangle$ is special in T' since φ_e is an isomorphism. Also, since the witness tuples for $\mathcal{R}_e$ have distinct first components, if two witness tuples are mapped to copies of I_0, then these copies are distinct (or else we win trivially). Therefore, if at least $n+1$ many tuples are mapped to copies of I_0, then at least one of these copies of I_0 was not created by a requirement of higher priority. In this case, we immediately diagonalize with such a tuple and $\mathcal{R}_e$ is won permanently.

Otherwise, there are at least $e+1$ many witness tuples whose images lie in the range of f_t. It is possible that some requirements of lower priority will act in a manner which causes some of these witness tuples $\langle x'_i, y'_i, z'_i \rangle$ to be contained in a copy of I_0 in T' at a later stage. If ever we reach a point where $n+1$ of the witness tuples are in copies of I_0, then $\mathcal{R}_e$ wins as in the previous paragraph.

If this does not happen, then for at least $e+1$ many tuples the values of f^{-1} on each entry in $\langle x'_i, y'_i, z'_i \rangle$ is not changed by any requirement of lower priority. Each of these tuples sits in a distinct cone immediately above the root of T'. Since the requirements $\mathcal{N}_u$ for $u < e$ can only protect e many of these cones, there must be an unprotected image tuple with which $\mathcal{R}_e$ enters the concurrent search of conditions (1) and (2). Because φ_e is an isomorphism, by the Δ^0_2 approximation to special tuples, $\mathcal{R}_e$ must eventually see that $\langle f_t^{-1}(x'_i), f_t^{-1}(y'_i), f_t^{-1}(z'_i) \rangle$ is special. From here, $\mathcal{R}_e$ will begin the concurrent search procedure and must discover an embedding as in condition (1). At this point, $\mathcal{R}_e$ diagonalizes, contradicting the fact that φ_e is an isomorphism. This finishes the proof of Lemma 3.10. ■

Lemma 3.15 *Let T be a tree of finite height, ω-branching at its root r, such that all nodes above r are of finite type. Let $x_0, x_1, \ldots$ be the immediate successors of r in T. If I_0 is an isomorphism type such that infinitely many i satisfy $T[x_i] \hookrightarrow I_0$ and infinitely many j satisfy both $I_0 \hookrightarrow T[x_j]$ and $T[x_j] \not\hookrightarrow I_0$, then T is not computably categorical.*

Proof. Let $\mathcal{T}$ be the set of all isomorphism types of successor trees above r in T, let $\mathcal{F} \subset \mathcal{T}$ be the set of types which do not embed into I_0 and let $\mathcal{E}$ contain those types in $\mathcal{F}$ into which I_0 embeds. By Corollary 3.8, $\mathcal{E}$ and $\mathcal{F} - \mathcal{E}$ each has a finite set of minimal elements, which we denote by $\mathcal{E}_0$ and $\mathcal{F}_0$, with every element of $\mathcal{E}$ lying above an element of $\mathcal{E}_0$ (in the embeddability order), and every element of $\mathcal{F} - \mathcal{E}$ lying above an element of $\mathcal{F}_0$. (Notice that no element of $\mathcal{E}$ can lie below an element of $\mathcal{F} - \mathcal{E}$, but it is possible for an element of $\mathcal{F} - \mathcal{E}$ to lie below an element of $\mathcal{E}$.) Lemma 3.3 yields a finite collection of finite subtrees, one S_i in each $J_i \in \mathcal{F}_0$ and one R_i in each $K_i \in \mathcal{E}_0$, such that no S_i embeds into any other S_j or into

I_0, and no R_i embeds into any other R_j, S_j or I_0. The important facts about these relations are that for all $I \in \mathcal{T}$,

$$I \in \mathcal{F} \Leftrightarrow \exists S_i(S_i \hookrightarrow I) \,\text{or}\, \exists R_i(R_i \hookrightarrow I)$$
$$\exists R_i(R_i \hookrightarrow I) \Leftrightarrow I \in \mathcal{E}.$$

At each stage s, we define the witness elements $w_{0,s} < \cdots < w_{p_s,s}$ at that stage to be those nodes $x \in T_s$ which we currently think are at the base of a successor tree of the root in T whose isomorphism type is not in $\mathcal{F}$. More specifically, we look for x satisfying:

- x is an immediate successor of r in T_s; and
- No $S_i \hookrightarrow T_s[x]$; and
- No $R_i \hookrightarrow T_s[x]$.

Then for each e, $w_e = \lim_s w_{e,s}$ exists, since infinitely many successor trees embed into I_0. We assign $e+1$ many witnesses to the requirement $\mathcal{R}_e$. (If there are not $e+1$ many witnesses available for $\mathcal{R}_e$, then it is not assigned any witnesses.) Because the limit of $w_{e,s}$ exists for each e, each requirement will eventually be assigned a full set of witnesses which never change. We need $e+1$ many witnesses since at most one witness may be forbidden by each of the requirements $\mathcal{N}_u$ for $u < e$.

We build T' in stages as T'_s, and we build a Δ^0_2-isomorphism $f : T \to T'$ by finite approximations $f_s : T_s \to T'_s$. At stage $s+1$, we extend f_s to f_{s+1} by adding fresh elements to T'_{s+1}, unless the following conditions hold for some requirement $\mathcal{R}_e$ and one of its witnesses $w_{e,s}$. (Here we abuse notation slightly by letting $w_{e,s}$ stand for an arbitrary witness node for $\mathcal{R}_e$ at stage s. This conflicts with our indexing of the witness nodes above, but it makes the connection between $w_{e,s}$ and the requirement $\mathcal{R}_e$ clearer.)

- $\varphi_{e,s}(w_{e,s})\downarrow \in T'_s$; and
- $\varphi_{e,s}(w_{e,s})$ is an immediate successor of $f_s(r)$ in T'_s; and
- No $S_i \hookrightarrow T'_s[\varphi_{e,s}(w_{e,s})]$; and
- No $R_i \hookrightarrow T'_s[\varphi_{e,s}(w_{e,s})]$; and
- it is not the case that $f_s^{-1}(\varphi_{e,s}(w_{e,s})) \prec u$ for any $u \in T$ with $u \leq e$ (this represents the restraint placed on $\mathcal{R}_e$ by $\mathcal{N}_u$ for $u < e$).

If these conditions hold for some $e \leq s$, then let e be the highest priority requirement for which these conditions hold. We attempt to diagonalize to meet $\mathcal{R}_e$ by searching for a stage $t > s$ such that either

1. $f_s^{-1}(\varphi_e(w_{e,s}))$ is not an immediate successor of r in T_t; or

2. Some $S_i \hookrightarrow T_t[f_s^{-1}(\varphi_e(w_{e,s}))]$; or

3. Some $R_i \hookrightarrow T_t[f_s^{-1}(\varphi_e(w_{e,s}))]$; or

4. There is an immediate successor x of r in T_t such that $T_t[x] \cap T_s = \emptyset$ and $T'_s[\varphi_e(w_{e,s})] \hookrightarrow T_t[x]$ and also some $R_i \hookrightarrow T_t[x]$.

If any of the first three conditions hold, then we define f_{s+1} and T'_{s+1} as if no requirement needed attention. In this case, it appears that φ_e is not an isomorphism since either $\varphi_e(w_{e,s})$ is not the base of a successor tree in T' or $\varphi_e(w_{e,s})$ is the base of a successor tree which does not embed into I_0. However, if $w_e = w_{e,s}$, then w_e is the base of a successor tree in T which does embed into I_0.

If the fourth condition holds, then we add $T_t[x]$ to our current copy of T, and define f_{s+1} to map $T_t[x]$ onto $T'_s[\varphi_e(w_{e,s})]$, adding fresh elements to T' above $\varphi_e(w_{e,s})$ to form a copy of $T_t[x]$. We also add more fresh elements to T' to be the new image of $T_s[f_s^{-1}(\varphi_e(w_e))]$ under f_{s+1}. Thus f_{s+1} is still an isomorphism, but hereafter $T'[\varphi_e(w_{e,s})]$ will grow as a copy of some successor tree containing R_i. By definition of R_i, this successor tree cannot be embedded into I_0. On the other hand, if $w_e = w_{e,s}$, then $T[w_e]$ can be embedded into I_0. Hence $\mathcal{R}_e$ will be satisfied.

If none of conditions 1-3 hold for any t, then $T[f_s^{-1}(\varphi_e(w_{e,s}))]$ does not lie above any minimal element of $\mathcal{F}$, so it must embed into I_0. Since infinitely many successor trees above r are supertrees of I_0 and do not embed into I_0, we see that condition 4 must then apply for some $t > s$. Therefore, this search procedure terminates.

If at some later stage s' we have $w_{e,s'+1} \neq w_{e,s'}$, then $\mathcal{R}_e$ and all lower priority requirements are injured at that stage. This happens if $w_{e,s'}$ is no longer an immediate successor of r, or if some R_i or S_i embeds into $T_{s'}[w_{e,s'}]$. However, such injuries can only happen finitely often for each e, since $w_{e,s}$ converges. Therefore, each requirement $\mathcal{R}_e$ only acts finitely often.

Since $\mathcal{R}_e$ has $e+1$ many witnesses and the requirements $\mathcal{N}_u$ for $u < e$ protect at most e many successor trees of the root in T from having the value of f_s changed on them, $\mathcal{R}_e$ must have some witness for which it is allowed to redefine f_s if it needs to in order to diagonalize. (As in the previous construction, if φ_e is not one-to-one or does not respect the ordering, then we win trivially and we cease trying to diagonalize.) Therefore, every requirement of the form $\mathcal{R}_e$ is eventually satisfied. Finally, notice that $f_s^{-1}(y)$ only changes if $y \in T'_s[\varphi_{e,s}(w_{e,s})]$ and $\mathcal{R}_e$ acts at stage s. In this case, some $R_i \hookrightarrow T'_{s+1}[\varphi_{e,s}(w_{e,s})]$ and therefore no $\mathcal{R}_k$ strategy ever redefines f_t^{-1} for $t > s$ on this subtree in an attempt to diagonalize again. Therefore, $f_s^{-1}(y)$ reaches a limit for each $y \in T'$.

The tree T' built by this process is computable, and isomorphic to T, since at each stage we have a homomorphism f_s from T_s into T'_s, with $f_s(r) = r$, whose range is all of T'_s. Our construction makes clear that $f = \lim_s f_s$ exists, since each $\mathcal{R}_e$ requirement must respect the $\mathcal{N}_u$ requirements for $u < e$. This finishes the proof of Lemma 3.15. ■

Lemma 3.16 *Let T be a tree of finite height with root r, such that all nodes above r are of finite type. Suppose there exist distinct isomorphism types $I_0, I_1, \ldots$ and I_ω appearing as* *successor trees above r. Suppose further that for every i, $I_i \hookrightarrow I_\omega$, and that I_ω appears infinitely often as a successor tree above r. Then T is not computably categorical.*

Notice that we do not require that $I_0, I_1, \ldots I_\omega$ be the only isomorphism types appearing as successor trees above r.

Proof. First we apply Corollary 3.8 to the set of successor trees above r which do not embed into I_ω, and use Lemma 3.3 to choose finite subtrees $R_1, \ldots R_m$ of the minimal elements of this set, such that no R_j embeds into I_ω.

Case 1. If there are only finitely many i such that $I_i \hookrightarrow I_\omega \not\hookrightarrow I_i$, then we will appeal to Lemma 3.19. Since there are finitely many such I_i, there is a finite subtree $S \subseteq I_\omega$ such that $S \not\hookrightarrow I_i$ for any such I_i. Hence the immediate successors x of r such that $T[x] \equiv I_\omega$ are precisely those x satisfying:

- $(\forall s)(\forall j) R_j \not\hookrightarrow T_s[x]$; and
- $(\exists s) S \hookrightarrow T_s[x]$.

This set is Δ^0_2 and infinite, and for all x_0 and x_1 in the set we have $T[x_0] \hookrightarrow I_\omega \hookrightarrow T[x_1]$, so indeed Lemma 3.19 will apply. (Also, the proof of Lemma 3.19 will not depend on Lemma 3.16 at all.)

Case 2. Now suppose that there are infinitely many i such that $I_i \hookrightarrow I_\omega \not\hookrightarrow I_i$. In this case, we build T' and an embedding $f : T \to T'$ such that T' is equal to the range of f plus extra copies of I_ω added as immediate successors of the root. Because I_ω occurs infinitely often as a successor tree of the root in T, we have that T and T' are isomorphic. As before, we build T' and f in stages such that at each stage s, T'_s is equal to the range of f_s plus finitely many copies of I_ω.

We pick one immediate successor y_0 of r in T such that $T[y_0] \cong I_\omega$, and use this finite information to identify our witness nodes. Our goal is to identify witness nodes $x \in T$ such that $T[x] \hookrightarrow I_\omega \not\hookrightarrow T[x]$ and then to diagonalize by making $T'[\varphi_e(x)] \cong I_\omega$. Because $T[x] \hookrightarrow I_\omega$ if and only if for each R_i, $R_i \not\hookrightarrow T[x]$, we can measure that x is an immediate successor of the root such that $T[x] \hookrightarrow I_\omega$ in a Δ^0_2 way. To measure whether $I_\omega \not\hookrightarrow T[x]$, at stage s, for each x which is an immediate successor of r in T_s, we define

$$t(x,s) = \mu t[t \geq x \ \&\ T_t[y_0] \not\hookrightarrow T_s[x]].$$

Then we choose the witness nodes $w_{0,s}, \ldots w_{p_s,s}$ to be those x which are successors of the root in T_s (and hence for which $t(x,s)$ is defined) and no R_j embeds into $T_s[x]$. We index these witnesses so that

$$t(w_{0,s}, s) \leq t(w_{1,s}, s) \leq \cdots \leq t(w_{p_s,s}, s)$$

with $w_{e,s} <_{\mathbb{N}} w_{e+1,s}$ for any e such that $t(w_{e,s}, s) = t(w_{e+1,s}, s)$.

Clearly, if x appears as a witness node at infinitely many stages s, then x must be an immediate successor of r and $T[x] \hookrightarrow I_\omega$. Moreover, for an immediate successor x such that

$I_\omega \hookrightarrow T[x]$, we must have $\lim_s t(x,s) = \infty$. On the other hand, if $T[x]$ is of one of the infinitely many types I_i for which $I_\omega \not\hookrightarrow I_i$, then $I_\omega \not\hookrightarrow T[x]$, and there is some t and some finite tree S_i such that $S_i \subseteq T_t[y_0]$ and $S_i \not\hookrightarrow T[x]$. Therefore, $\lim_s t(x,s)\downarrow \leq t$. Hence for each of these latter values of x, there must be an e with $\lim_s w_{e,s}\downarrow = x$. We write w_e for this x, and note that since there are infinitely many such x, the limit w_e must exist for all e. This gives us our witness nodes. We assign $e+1$ many witness to the requirement $\mathcal{R}_e$. (As before, we abuse notation when we describe the action of requirement $\mathcal{R}_e$ at stage s by denoting its witness by $w_{e,s}$.)

At stage $s+1$ of the construction, we extend our previous map f_s to T_{s+1}. We do this by adding fresh elements to the image T'_{s+1}, unless some requirement $\mathcal{R}_e$ requires attention. We say that $\mathcal{R}_e$ requires attention if there exists a witness $w_{e,s}$ for $\mathcal{R}_e$ such that $\varphi_{e,s}(w_{e,s})\downarrow$ (say $\varphi_{e,s}(w_{e,s}) = y$), y is an immediate successor of the root in T'_s, y is in the range of f_s, and it is not the case that $f_s^{-1}(y) \prec u$ for some $u < e$. Notice that if $\mathcal{R}_e$ does not require attention because y is not an immediate successor of the root in T', then we win $\mathcal{R}_e$ trivially as long as $w_{e,s} = w_e$. If y is a successor of the root of T'_s but $\mathcal{R}_e$ does not require attention because y is not in the range of f_s, then y is the root of a tree of type I_ω in T'. Again, if $w_{e,s} = w_e$, then we have won $\mathcal{R}_e$ trivially. And if $y \prec u$ for some $u < e$, then we do not allow $\mathcal{R}_e$ to act on this witness $w_{e,s}$ since the action could damage the negative requirement $\mathcal{N}_u$.

If $\mathcal{R}_e$ is the highest priority requirement needing attention, then we check if some R_j embeds into $T'_s[y]$. If so, then we know $T'_s[y] \not\hookrightarrow I_\omega$. Assuming $w_{e,s}$ turns out to be a true witness, $T[w_{e,s}] \ncong T'[y]$ and we have won $\mathcal{R}_e$. If no R_j embeds in $T'_s[y]$, then we attempt to diagonalize. Search concurrently until we find

1. some $R_j \hookrightarrow T_t[f_s^{-1}(y)]$ for $t \geq s$; or

2. some witness $w_{0,t}, \ldots, w_{e,t}$ changes; or

3. an embedding of $T'_s[y]$ into I_ω appears.

By the definition of R_j, we know that this search procedure must terminate.

If the search in (1) or (2) is successful, then we do not need to do anything for $\mathcal{R}_e$. Either we do not really believe that $w_{e,s}$ is the correct witness, or we believe that when we get to stage t we will win $\mathcal{R}_e$ easily because $R_j \hookrightarrow T'_t[y]$.

If the search in (3) is successful, then we add fresh elements to T'_{s+1} above y to make $T'_{s+1}[y] \cong I_\omega$. (Notice that since I_ω is of finite type, we can execute this step all at once, using a nice copy of I_ω constructed from only finitely much information.) Also, add more fresh elements to T'_{s+1} to be the new image of $T_s[f_s^{-1}(y)]$ under f_{s+1}. Thus, we have redefined f on $T_s[f_s^{-1}(y)]$, but as in the previous arguments, because $\mathcal{R}_e$ must respect $\mathcal{N}_u$ for $u < e$, this can happen only finitely often. Moreover, if $w_e = w_{e,s}$, then $T[w_e]$ will not be isomorphic to I_ω, hence not isomorphic to $T[\varphi_e(w_e)]$, satisfying requirement $\mathcal{R}_e$. Finally, notice that for $y \in T'$, we only change $f_s^{-1}(y)$ when we remove y from the range of f. However, when this happens, y permanently becomes part of an auxiliary copy of I_ω. ■

The next lemma is not a separate case of our overall proof of Proposition 3.1, but it will be used later in the proof of Lemma 3.18.

Lemma 3.17 *Let T be a tree of finite height with root r. Let $x_0, x_1, \ldots$ be the immediate successors of r in T, and assume that every x_i is of finite type. Moreover, assume that there exists an infinite Δ^0_2 set $G \subseteq \{x_i : i \in \omega\}$ such that*

1. *if $x_i \in G$ and $T[x_i] \cong T[x_j]$, then $x_j \in G$,*
2. *every $T[x_i]$ with $x_i \in G$ embeds into infinitely many $T[x_j]$ with $x_j \in G$, and*
3. *for each $x_i \in G$, $\{x_j : T[x_j] \cong T[x_i]\}$ is a finite subset of G.*

Then T is not computably categorical.

Proof. We construct a computable tree T' isomorphic to T, such that for every e, if φ_e were an isomorphism from T to T', then one of the $\cong$-classes described in the lemma would be infinite. Let G_s be a computable approximation to G, with every G_s finite. At each stage s we define a finite subtree $D_s \subset T$ with $D_s \subseteq D_{s+1}$ and an isomorphism $f_s : D_s \to T'_s$, such that f_s converges to a Δ^0_2-isomorphism $f : T \to T'$. We will choose infinitely many witness elements w^j_i for each (total one-to-one) function φ_i.

We begin by motivating our strategy for a single $\mathcal{R}_e$ requirement. For simplicity of notation, we assume that G is computable, as adding the Δ^0_2 approximation to G is straightforward. $\mathcal{R}_e$ begins with a single witness $w^0_e \in G$ and waits for a stage s such that $\varphi_e(w^0_e)$ converges. If $\varphi_e(w^0_e) \in T'_s$ and $f^{-1}_s(\varphi_e(w^0_e)) \notin G$, then $\mathcal{R}_e$ is satisfied and we do not perform any action. (Notice that by condition (1) on G, if some $x \in G$ we have $f^{-1}(\varphi_e(x)) \notin G$, then φ_e cannot be an isomorphism from T to T'.) Otherwise, if either $\varphi_e(w^0_e) \notin T'_s$ or $f^{-1}_s(\varphi_e(w^0_e)) \in G$, we begin our action for $\mathcal{R}_e$.

Search for a $t > s$ and an $x \in T$ such that $x \in G$, $T'_s[\varphi_e(w^0_e)] \hookrightarrow T_t[x]$, and $T_t[x]$ is disjoint from D_s. If $f^{-1}_s(\varphi_e(w^0_e))$ is in G, then we must find such an x since $T[f^{-1}_s(\varphi_e(w^0_e))]$ embeds into $T[y]$ for infinitely many $y \in G$. (If $\varphi_e(w^0_e) \notin T'_s$, then the embedding condition is trivial and we merely look for $T_t[x]$ which is disjoint from D_s.) We now define the map f_{s+1} by changing the map f_s on $T_s[f^{-1}_s(\varphi_e(w^0_e))]$. Use the embedding of $T'_s[\varphi_e(w^0_e)]$ into $T_t[x]$ and add extra elements to T'_{s+1} to make f_{s+1} map $T_t[x]$ onto $T'_{s+1}[\varphi_e(w^0_e)]$. Add more new elements to T'_{s+1} to serve as the new image of $T_s[f^{-1}_s(\varphi_e(w^0_e))]$ under f_{s+1}. For all other points in D_s, let $f_{s+1} = f_s$. We now have defined f_{s+1} on D_{s+1}, which is equal to D_s plus $T_t[x]$. Finally, we define $w^1_e = x$. (Notice that we can speed up the approximation of T to assume that $D_{s+1} \subseteq T_{s+1}$.)

Repeat the above procedure, but working with w^1_e instead of w^0_e. Notice that if we keep extending our map f_{s+1} to copy the successor trees it is currently defined on, we will get that f is an isomorphism between $T'[\varphi_e(w^0_e)]$ and $T[w^1_e]$. Therefore, if φ_e is an isomorphism,

$$T[w^0_e] \cong T'[\varphi_e(w^0_e)] \cong T[w^1_e].$$

By repeating this process (and assuming that φ_e continues to converge on all of our witnesses w_e^n and is well behaved – see below), we get a sequence of witnesses w_e^n such that if φ_e is an isomorphism, then $T[w_e^n] \cong T[w_e^{n+1}]$ for all n. This contradicts the fact that the isomorphism types given by nodes in G occur finitely often. Notice that we change the map f as we go from f_s to f_{s+1} on $T_s[f_s^{-1}(\varphi_e(w_e^n))]$ and we also change the map from f_s^{-1} to f_{s+1}^{-1} on $T_s'[\varphi_e(w_e^n)]$. Since we are not turning $T_s'[\varphi_e(w_e^n)]$ into an auxiliary tree, we will need to explicitly address the requirements $\mathcal{M}_u$ for the first time.

By well behaved, we mean that φ_e is one-to-one, that it maps the root to the root, that it maps comparable nodes to comparable nodes, and that it maps incomparable nodes to incomparable nodes. If we ever see any of these conditions violated, then we know φ_e is not an isomorphism and we can stop working on $\mathcal{R}_e$. In all of the work below, we assume that we stop work on $\mathcal{R}_e$ if we get an easy win because it is not well behaved in this sense.

Combining the basic strategy for $\mathcal{R}_e$ with the $\mathcal{N}_i$ strategies is a little more subtle than in previous constructions because one $\mathcal{R}_e$ requirement can cause infinitely many changes in the map f_s. Before redefining f_s on $T_s[f_s^{-1}(\varphi_e(w_e^n))]$, we check if $f_s^{-1}(\varphi_e(w_e^n)) \prec m$ for any m from $0, \ldots, \langle e, n\rangle$ in T. If not, then we act as above. If it is below any such m, then we cannot redefine f_s on this subtree. We also employ a similar strategy to deal with the $\mathcal{M}_i$ strategies. That is, if $\varphi_e(w_e^n) \leq \langle e, n\rangle$, then $\mathcal{R}_e$ cannot use w_e^n as a witness since this would involve redefining $f_s^{-1}(\varphi_e(w_e^n))$. In either case, $\mathcal{R}_e$ must repick its witness w_e^n. If $n > 0$, we declare that w_e^n is undefined. This forces us to repeat the cycle above for w_e^{n-1} and gives us a new (large) witness w_e^n. Assuming that φ_e is well behaved, this new witness gives us a different value for $f_s^{-1}(\varphi_e(w_e^n))$ (which we assume is in G). Since each point in G is an immediate successor of the root in T, we will have to repeat this process at most $2\langle e, n\rangle + 1$ many times before we are guaranteed to be allowed to redefine f_s. If $n = 0$, then we need to choose a new initial witness w_e^0. To do this, we declare that the old w_e^0 is disallowed for $\mathcal{R}_e$ and we let the new w_e^0 be the least element of G which has not been disallowed for $\mathcal{R}_e$. If φ_e is well behaved, then we will have to redefine our initial witness in this fashion at most $2\langle e, 0\rangle + 1$ many times. Therefore, we eventually get our infinite sequence of witnesses and win.

We also need to see how different $\mathcal{R}$ strategies work together. Each $\mathcal{R}_e$ will have some finite (possibly empty) list of witnesses $w_e^0, \ldots, w_e^n$ at stage s. We say that w_j^i has higher priority than w_q^p if $\langle i, j\rangle < \langle p, q\rangle$. Consider an $\mathcal{R}_e$ strategy working with other $\mathcal{R}$ strategies. If $\mathcal{R}_e$ has a largest witness w_e^n and $\varphi_e(w_e^n)$ converges with $f_s^{-1}(\varphi_e(w_e^n)) \in G$, then in addition to checking whether $f_s^{-1}(\varphi_e(w_e^n))$ is below any of the numbers $0, \ldots, \langle e, n\rangle$ in T, $\mathcal{R}_e$ also checks whether $f_s^{-1}(\varphi_e(w_e^n))$ is equal to any other w_i^m. If so, then changing f_s on $T_s[f_s^{-1}(\varphi_e(w_e^n))]$ could damage the requirement $\mathcal{R}_i$. Therefore, $\mathcal{R}_e$ checks if the node w_i^m has higher priority than w_e^n. If so, then $\mathcal{R}_e$ cannot change the map on this cone, so it acts as when it was restricted by an $\mathcal{N}$ or $\mathcal{M}$ requirement. If not, then it causes all w_i^m of lower priority to become undefined and goes ahead with its action as above.

Notice that this action may allow $\mathcal{R}_i$ to injure $\mathcal{R}_e$ even though $e < i$. However, only finitely many witnesses w_i^m can injure a given w_e^n, and therefore, w_e^n will eventually reach a limit which $\mathcal{R}_e$ can use.

We now present the full construction, which is nothing more than the above description with the Δ^0_2 guessing for elements of G. We start by setting $D_0 = \{r\}$ and $T'_0 = \{0\}$, with $f_0(r) = 0$.

At stage $s+1$, we make a preliminary definition of our witnesses by induction on i from $0 \leq i \leq s$. If $w^0_{i,s} \in G_s$, then let $w^0_{i,s+1} = w^0_{i,s}$ and $w^j_{i,s+1} = w^j_{i,s}$ for all $j > 0$ such that $w^j_{i,s}$ is defined. If $w^0_{i,s} \notin G_s$ or $w^0_{i,s}$ is not defined, then $w^0_{i,s+1}$ and all lower priority witnesses $w^n_{e,s+1}$ are undefined (even if some of these were defined earlier in the induction). Next, we check if some new initial witness $w^0_{k,s+1}$ can be defined. If there is a $k \leq s$ and an element $x \in G_s$ such that $w^0_{k,s+1}$ is undefined, $x \notin \{w^j_{i,s+1} : \langle i,j\rangle < \langle k,0\rangle\}$, and x is not disallowed for w^0_k, then we let $w^0_{k,s+1}$ be the least such x. Make all lower priority witnesses undefined.

We then find the least pair $\langle i,j\rangle$ such that $\varphi_{i,s}$ is well behaved, $w^j_{i,s+1}$ is defined, $w^{j+1}_{i,s+1}$ is not defined, $\varphi_{i,s}(w^j_{i,s+1})\downarrow$, and either $f_s^{-1}(\varphi_{i,s}(w^j_{i,s+1})) \in G_s$ or $\varphi_{i,s}(w^j_{i,s+1}) \notin T'_s$. (If there is no such pair, we end the stage, let $D_{s+1} = D_s \cup \{s\}$ and $f_{s+1} = f_s$ plus add one fresh element $f_{s+1}(s)$ to T'_{s+1} if needed.) If $\varphi_{i,s}(w^j_{i,s+1}) \in T'_s$, then check the following two conditions for compatibility with the appropriate $\mathcal{N}$ and $\mathcal{R}$ requirements. (In the case when $\varphi_{i,s}(w^j_{i,s+1}) \notin T'_s$, we can skip these checks.)

First, check for compatibility with the $\mathcal{N}$ and $\mathcal{M}$ requirements. If $\varphi_i(w^j_{i,s+1}) > \langle i,j\rangle$ and there is no $k \leq \langle i,j\rangle$ such that $f_s^{-1}(\varphi_i(w^j_{i,s+1})) \prec k$ in T_s, then go to the check in the next paragraph. Otherwise, if $j > 0$, declare $w^j_{i,s+1}$ undefined and begin the next stage. If $j = 0$, then declare $w^0_{i,s+1}$ disallowed for w^0_i, make $w^0_{i,s+1}$ undefined and go to the next stage. (In both of these cases, we extend D_s to $D_{s+1} = D_s \cup \{s\}$, add an extra element to T'_{s+1}, and define f_{s+1} on this new element if necessary.)

Second, check for compatibility with the $\mathcal{R}$ requirements. If there is no higher priority $w^n_{e,s+1}$ such that $\varphi_i(w^j_{i,s+1}) = f_s(w^n_{e,s+1})$, then go to the next paragraph. Otherwise, if there is such a $w^n_{e,s+1}$ and $j > 0$, declare $w^j_{i,s+1}$ undefined and begin the next stage. If $j = 0$, then declare $w^0_{i,s+1}$ disallowed for w^0_i, make $w^0_{i,s+1}$ undefined and go to the next stage. (Handle D_{s+1}, T'_{s+1} and f_{s+1} as in the previous paragraph.)

If both of these checks are successful, search for the least stage $t > s+1$ such that either

- $\exists x \in G_t(T_t[x] \cap D_s = \emptyset \;\&\; T'_s[\varphi_i(w^j_{i,s+1})] \hookrightarrow T_t[x])$; or
- $\varphi_{i,s}(w^j_{i,s+1}) \in T'_s$ and $f_s^{-1}(\varphi_{i,s}(w^j_{i,s+1})) \notin G_t$.

In the latter case we repeat the above process for the next pair $\langle i,j\rangle \leq s$ which appears to need attention. In the former case, we set $w^{j+1}_{i,s+1} = x$, add elements to T'_{s+1} above (and possibly including) the node $\varphi_i(w^j_{i,s+1})$ to make a copy of $T_t[x]$, and define f_{s+1} to map $T_t[x]$ onto these elements, according to the embedding we found. If this requires redefining f_{s+1} on elements of D_s which had mapped into $T'_s[\varphi_i(w^j_{i,s+1})]$ under f_s, we add fresh elements to T'_{s+1} to be their images. For all other elements of D_s, f_{s+1} takes the same value as f_s. We add the elements of $T_t[x]$ to D_{s+1} and we enumerate s into D_{s+1}, adding a fresh element as its image in T'_{s+1} if necessary. Thus T'_{s+1} is the bijective image of D_{s+1} under f_{s+1}.

We claim that the search for stage t must eventually terminate. If $f_s^{-1}(\varphi_i(w^j_{i,s+1})) \notin G$, this is clear. If $f_s^{-1}(\varphi_i(w^j_{i,s+1})) \in G$, then there are infinitely many other nodes $x \in G$ such that $T[f_s^{-1}(\varphi_i(w^j_{i,s+1}))] \hookrightarrow T[x]$. Eventually we find a node x (in G_t but not necessarily in G) with such an embedding, such that $T[x] \cap D_s = \emptyset$, and we use it. Finally, if $\varphi_i(w^j_{i,s+1}) \notin T'_s$, then $T'_s[\varphi_i(w^j_{i,s+1})]$ is considered to be empty, hence embeds trivially into $T_t[x]$ for the first $x \notin D_s$ to appear in any later $T_t \cap G_t$.

The verification that the construction succeeds is essentially as described in the informal setting. The witness $w^0_{0,s}$ can only be changed if it leaves G_s or if $\varphi_0(w^0_{0,s})$ converges either to 0 or such that $f_s^{-1}(\varphi_0(w^0_{0,s})) \prec 0$ in T. Therefore, this witness is only injured finitely often due to the Δ^0_2 nature of G and is injured at most once by each of the requirements $\mathcal{N}_0$ and $\mathcal{M}_0$. Since no other requirement can injure $w^0_{0,s}$, this witness eventually reaches its final value. Similarly, for each $w^j_{i,s}$, once the higher priority witnesses have reached their final values (which may include never being defined again), this witness suffers finite injury due to the fact that G is Δ^0_2, finite injury due to the restraints of $\mathcal{N}$ and $\mathcal{M}$, and finite injury due to the restraints of the higher priority witnesses for $\mathcal{R}$ requirements. Therefore, for every witness $w^j_{i,s}$, there is a stage t such that either $w^j_{i,s}$ has stabilized by stage t or $w^j_{i,s}$ is never defined after stage t.

Since each witness stabilizes, $\mathcal{R}_i$ only changes f_s finitely many times for each potential witness $w^j_{i,s}$. Therefore, because of the restraint imposed by the $\mathcal{N}$ requirements, $f(x) = \lim_s f_s(x)$ exists for all $x \in T$ and because of the restraint of the $\mathcal{M}$ requirements, $f^{-1}(y) = \lim_s f_s^{-1}(y)$ exists for all $y \in T'$. Thus f gives a Δ^0_2-isomorphism from T to T'.

If φ_i is indeed well behaved and total, then we define a growing sequence of nodes $w^0_{i,s}, \ldots, w^j_{i,s}$ which eventually settle down to $w^0_i, \ldots, w^j_i$. If $f^{-1}(\varphi_i(w^j_i)) \notin G$, then φ_i cannot be an isomorphism, since the lemma assumes that if $T[x] \cong T[y]$, then $(x \in G \iff y \in G)$. If $f^{-1}(\varphi_i(w^j_i)) \in G$, then eventually the second clause in the search for stage t will never again apply, and we will find a t and an x which we define to be w^{j+1}_i.

Once w^{j+1}_i has converged, we define $f_s(w^{j+1}_i) = \varphi_i(w^j_i)$. This action may be injured finitely many times, but eventually it settles on a final w^{j+1}_i with $f(w^{j+1}_i) = \varphi_i(w^j_i)$. We know that f is an isomorphism from T to T'. If φ_i were an isomorphism as well, then we would have

$$T[w^j_i] \cong T'[\varphi_i(w^j_i)] \cong T[w^{j+1}_i]$$

for every j, the first isomorphism being φ_i and the second being f^{-1}. But $w^0_i \in G$ (since we check this immediately at every stage), and $w^k_i \neq w^j_i$ for all $k \neq j$, since each new w^k_i is always chosen as a node in T not yet in the domain D_s of f_s. This contradicts the assumptions of the lemma, so T and T' cannot be computably isomorphic. ∎

Lemma 3.18 *Let T be a tree of finite height, such that the root r has infinitely many immediate successors $x_0, x_1, \ldots$. Assume that all nodes above r are of finite type, and that there are infinitely many isomorphism types in the set $\{T[x_i]\}$. Suppose that only finitely many of these isomorphism types appear infinitely often as successor trees above r, and that for each* *such type I, only finitely many other types appearing above r embed into I. Then T is not computably categorical.*

Proof. Let $\mathcal{T}$ be the set of isomorphism types of successor trees in T and let $\mathcal{I}$ be the set of types in $\mathcal{T}$ which embed in any of the infinite-occurring types (including the infinitely-occurring types themselves). By assumption $\mathcal{I}$ is finite. We let $\{S_i\}$ be a finite collection of finite trees such that no S_i embeds into any element of $\mathcal{I}$, yet every $T[x_i] \notin \mathcal{I}$ has some S_i as a subtree. (Here we use Corollary 3.8 and Lemma 2.9.) The elements of $\mathcal{I}$ are precisely those types into which no S_i embeds. Therefore, there is a Δ^0_2 guessing process to identify all successor trees above r whose type is in $\mathcal{I}$.

Notice further that by Corollary 3.6, we infer that of the elements of $\mathcal{T} - \mathcal{I}$, all but finitely many embed into infinitely many other elements of $\mathcal{T} - \mathcal{I}$. Let

$$U = \{x_i : T[x_i] \in \mathcal{T} - \mathcal{I} \;\&\; (\exists m)(\forall j \geq m) T[x_i] \not\hookrightarrow T[x_j]\}$$

be this finite set. We will concern ourselves with the set F of immediate successors x of r in T such that $T[x] \in \mathcal{T} - \mathcal{I}$ and $T[x]$ embeds into infinitely many other elements of $\mathcal{T} - \mathcal{I}$. We have a Δ^0_2-approximation F_s for F, and we may assume that each $F_s \cap U = \emptyset$.

Case 1. Suppose there are only finitely many equivalence classes $\mathcal{E}_0, \dots \mathcal{E}_p$ under $\equiv$ among $\{T[x] : x \in F\}$. At least one must be infinite, so assume that $\mathcal{E}_0, \dots \mathcal{E}_q$ are the infinite classes and $\mathcal{E}_{q+1}, \dots, \mathcal{E}_p$ are the finite ones. Since each isomorphism type occurs only finitely often in each $\mathcal{E}_i$, we have that the set

$$X = \{x \in F : T[x] \in \mathcal{E}_{q+1} \cup \dots \cup \mathcal{E}_p\}$$

is finite. Let $G = F - X$. Since F is Δ^0_2 and X is finite, G is Δ^0_2. G is exactly the kind of set to which we can apply Lemma 3.17. Therefore, T is not computably categorical.

Case 2. Suppose there are infinitely many equivalence classes under $\equiv$ among $\{T[x] : x \in F\}$. We will write $\mathcal{E}_j \hookrightarrow \mathcal{E}_k$ to indicate that some (hence all) elements of $\mathcal{E}_j$ embed into some (hence all) elements of $\mathcal{E}_k$. For each class $\mathcal{E}_k$, pick one representative $T[x_{i_k}]$, and apply Corollary 3.6 to $\{T[x_{i_k}] : k \in \omega\}$. (We do not need this procedure of picking elements to be computable since we only use it to obtain a finite amount of information detailed below.) This gives us a K such that for all $k \geq K$, there are infinitely many $j > k$ such that $\mathcal{E}_k \hookrightarrow \mathcal{E}_j$.

Consider the equivalence classes $\mathcal{E}_0, \dots, \mathcal{E}_{K-1}$. Divide these classes and renumber them so that $\mathcal{E}_0, \dots, \mathcal{E}_q$ are the finite ones. Then, just as above in Case 1, the set

$$X = \{x \in F : T[x] \in \mathcal{E}_0 \cup \dots \cup \mathcal{E}_q\}$$

is finite. Therefore, $G = F - X$ is a Δ^0_2 set to which we can apply Lemma 3.17. Therefore, T is not computably categorical. ■

Oddly, the remaining case turns out to be the hardest. This is the situation in which we have infinitely many isomorphism types appearing above r, all of finite type, with every such isomorphism type embedding into every other one. One would think that with so many embeddings at hand the proof would be easy. Alternatively, Lemma 2.10 shows that infinitely many of these types must fail to be of strongly finite type, hence must have embeddings available within them to satisfy all the requirements. Curiously, the presence of so many types and embeddings interferes with the availability of non-strongly-finite types, and vice versa, so that in the end we must use a completely different approach. The following lemma will not be used directly in the proof of Proposition 3.1, but it is necessary to finish the proof of Case 1 in Lemma 3.16.

Lemma 3.19 *Let T be a tree of finite height with root r, such that every successor tree above r is of finite type. Suppose there is an infinite set $X = \{x_0, x_1, \ldots\}$ of immediate successors of r satisfying:*

1. *X is Δ^0_2; and*
2. *For all $x_i, x_j \in X$, $T[x_i] \equiv T[x_j]$; and*
3. *$\{T[x_i]\}$ includes infinitely many distinct isomorphism types.*

Then T is not computably categorical.

Proof. To simplify the proof, we will assume that X contains every immediate successor of r in T. The finite-injury construction we present can readily be adapted to the more general case, using Δ^0_2-approximations to X. We begin by presenting the particular case in which $\mathrm{ht}(T) = 4$. As no tree of height < 4 satisfies the hypotheses of the lemma, this will serve as the base case for an induction on the height of T. Suppose $\mathrm{ht}(T) = 4$ and every successor tree above r embeds into every other such successor tree, and there are infinitely many isomorphism types occurring among these successor trees and they are all of finite type. Each time a node becomes established at level 1 in T, we know that it is the root of a successor tree, so we define it to be the next x_i. Since the successor trees all embed into each other, they must all have the same height – namely 3, since $\mathrm{ht}(T) = 4$ – so every node at level 1 of T eventually is identified as x_i for some i. Also, it is not hard to see that since each $T[x_i]$ has finite type, each $T[x_i]$ must be ω-branching at its root for the conditions of the lemma to hold.

If y is an immediate successor of any x_i, then the isomorphism type of $T[y]$ is determined by the number of immediate successors y has. By Lemma 3.9, since $T[x_i] \equiv T[x_j]$ are finite type trees, they have exactly the same infinitely occurring successor trees. Therefore, there is a finite list $n_1 < n_2 < \cdots < n_k$ such that $n_j \leq \omega$ for each $1 \leq j \leq k$, and for every i, the successor trees in $T[x_i]$ which occur infinitely often are the n_j-branching trees of height 2 for $1 \leq j \leq k$. We next show that $k = 1$. Suppose that $k > 1$. Under this assumption, $T[x_i]$ has an infinitely occurring successor tree which embeds into a nonisomorphic infinitely occurring successor tree. Since this violates the definition of finite type, we must have $k = 1$. Let $\gamma = n_1$ be such that the unique infinitely occurring successor tree in every $T[x_i]$ is γ-branching.

Furthermore, we claim that for any m such that $\gamma < m \leq \omega$, each $T[x_i]$ has exactly the same number of m-branching successor trees. To see this fact, first suppose $\gamma < \omega$,

$T[x_i]$ has u_i many ω-branching trees, $T[x_j]$ has u_j many ω-branching trees and $u_i < u_j$. We know $T[x_j] \hookrightarrow T[x_i]$, so each ω-branching successor tree in $T[x_j]$ must map into an ω-branching successor tree in $T[x_i]$, and because of the heights of the trees, two different ω-branching successor trees in $T[x_j]$ cannot map into the same ω-branching successor tree in $T[x_i]$. Therefore, we have an immediate contradiction. Second, fix the maximal $m < \omega$ such that $\gamma < m$ and $T[x_i]$ has v_i many m-branching successor trees for some $v_i > 0$. (Because $T[x_i]$ has finite type, it has only finitely many different isomorphism types among its successor trees. Therefore, either there is no m with $\gamma < m < \omega$ such that $T[x_i]$ has an m-branching successor tree, in which case we have established our claim, or there is a maximal such m.) Each of these trees must map into a successor tree in $T[x_j]$ which is at least m-branching (but not ω-branching since those successor trees are already mapped to by the ω-branching successor trees in $T[x_i]$) and no two such m-branching successor tree in $T[x_i]$ can map into the same successor tree in $T[x_j]$. Therefore, $T[x_j]$ must have at least v_i many successor trees which are at least m-branching but not ω-branching. However, if $T[x_j]$ has a successor tree that is more than m-branching but less than ω-branching, then it has no place to map to under $T[x_j] \hookrightarrow T[x_i]$. Therefore, $T[x_j]$ has at least v_i many successor trees which are exactly m-branching. By switching the roles of $T[x_i]$ and $T[x_j]$, we see that $T[x_j]$ must have exactly v_i many successor trees which are m-branching. We can obviously continue this process with the next largest number which is less than m, greater than γ and such that $T[x_i]$ has at least one successor tree with that number of branches.

We now know that any $T[x_i]$ and $T[x_j]$ must look identical with respect to their successor trees which are more than γ-branching. However, there must be infinitely many different isomorphism types among the $T[x_i]$ trees. These differences in isomorphism type must be due to the successor trees which are less than γ-branching. Therefore, infinitely many $T[x_i]$ contain a node w at level 1 in $T[x_i]$ which has fewer than γ immediate successors. We will use as our witness nodes those nodes w with $< \gamma$ immediate successors, with at most one witness node in each successor tree. When necessary to diagonalize, we add more successors to $\varphi_e(w)$ in T' so that it has exactly γ successors.

We identify the witness nodes as follows. At any given stage, the witness node in $T[x_i]$ should be that node $x \in T[x_i]$ with $< \gamma$ successors which has $\text{level}_{T_s}(x) = 2$ and which has gone the longest without acquiring any new successors. That is, for each x at level 1 in $T[x_i]$, let

$$t_x = (\mu t \geq x)[\text{all successors of } x \text{ in } T_s \text{ are in } T_t],$$

and choose as *the witness node in* $T[x_i]$ *at stage* s the smallest x such that t_x is minimal.

However, it is possible that $T[x_i]$ contains no nodes with $< \gamma$ successors, so we must search among different successor trees. At first, we choose $w_{0,s}$ to be the witness node in $T[x_0]$. If this witness node changes at some subsequent stage s_1, then we choose w_{0,s_1+1} to be the witness node in $T[x_1]$. If at a subsequent stage s_2 the witness node in $T[x_1]$ changes, then we change w_{0,s_2+1} back to the (current) witness node in $T[x_0]$, then $T[x_1]$ again, then $T[x_2]$, then back to $T[x_0]$, and so on. In general, let s_k be the next stage (if any) after s_{k-1} at which $w_{0,s}$ changes, and choose $w_{0,1+s_k}$ to be the witness node in $T[x_i]$ at stage $1 + s_k$, where $k = \langle i, j \rangle$.

The properties proved above guarantee that there must be infinitely many $T[x_i]$ containing nodes x such that x is an immediate successor of x_i and such that x has $< \gamma$ successors, so eventually $w_{0,s}$ converges to some w_0. At the same time, we do the same for the witness node $w_{1,s}$ for $\mathcal{R}_1$, looking only at witness nodes in successor trees $T[x_j]$ in which $w_{0,s}$ has never yet been located, and so on by a standard finite-injury process.

Sublemma 3.20 *For every e, $w_e = \lim_s w_{e,s}$ exists and has $< \gamma$ successors in T.*

Proof. Assume by induction that the lemma holds for all $i < e$. Then each of $w_0, \dots w_{e-1}$ lies above one of $x_0, \dots x_k$, for some k. By our assumptions about T, there must be a node y in some $T[x_j]$ with $j > k$ such that $\text{level}_T(y) = 2$ and y has $< \gamma$ successors. Assume that this y is chosen to have minimal t_y among all such nodes in $T[x_j]$. (Hence y acquires no new successors after stage t_y. If there is more than one y with minimal t_y, we take y to be the smallest of them.)

Pick a stage s_0 by which $x_0, \dots x_j$ are all established, so that $T[x_j]$ will be available to us when we define $w_{e,s}$ at all $s \geq s_0$. (Hence $w_{e,s}$ will never be undefined after s_0.) If $w_{e,s}$ fails to converge to a limit, then it must be in $T[x_j]$ at infinitely many stages s, according to our instructions for choosing $w_{e,s}$. Since t_y is minimal, we must have $w_{e,s} = y$ at cofinitely many of the stages such that $w_{e,s} \in T[x_j]$. Hence $w_{e,s} = y$ at some stage $s > t_y$. But then $w_{e,s} = y$ for all subsequent stages s as well, proving the sublemma. (Possibly $w_{e,s}$ converged to some other limit in some other $T[x_k]$ instead of converging to y, of course.) However, in any case, the construction guarantees that the limit must have $< \gamma$ successors in T. ∎

We build T' by copying T at each stage s, with the following provision. Find each $e \leq s$ such that $w_{e,s}$ is defined and $\varphi_{e,s}(w_{e,s})\downarrow$ (say $y = \varphi_{e,s}(w_{e,s})$) and $f_s^{-1}(y)$ lies at level 2 in T_s and has fewer than γ successors in T_s. If there is no such e, simply extend f_s to f_{s+1} by adding new elements to T'_{s+1}. If there is, then for the least such e, add new elements to T' so that y has exactly γ successors in T'. We also add new elements to T' to be the image of $T_s[f_s^{-1}(y)]$ under f_{s+1}. The elements of $T'_{s+1}[y]$ will not lie in the image of the limit f, but T and T' will still be isomorphic, since every x_i has infinitely many immediate successors with exactly γ successors of their own. Notice that as in previous constructions with auxiliary trees, if a node $y \in T'$ is removed from the range of f, then it permanently becomes part of an auxiliary γ-branching subtree of T'. We have simply added one more such node above $f(x_i)$ in T' during this redefinition of f. The only injuries to $\mathcal{R}_e$ occur when $w_{i,s+1} \neq w_{i,s}$ for some $i < e$. Thus we have ensured that $y = \varphi_{e,s}(w_{e,s})$ has γ successors in T', while $w_{e,s}$ has $< \gamma$ successors in T_s. If $w_{e,s} = w_e$, then $w_{e,s}$ acquired no new successors in T after stage s, leaving $\mathcal{R}_e$ satisfied.

Two minor modifications to this strategy are required for $\mathcal{R}$ and $\mathcal{N}$ strategies to work together. First, as we have done before, we assign $e+1$ many witnesses to $\mathcal{R}_e$ and we check whether $f_s^{-1}(y) \prec u$ for $u < e$ before allowing $\mathcal{R}_e$ to act. This modification insures that $f = \lim_s f_s$ exists and that $T \cong T'$. Second, since there are parts of T'_s which are not in the range of f_s, it is possible that y lies at level 2 in T'_s but it is not in the range of f_s. In this case, y already has γ successors because of the action of some $\mathcal{R}$ requirement. Therefore, if $w_{e,s} = w_e$, then $\mathcal{R}_e$ has already won without needing to act.

We now assume by induction that for all trees T with $\text{ht}(T) < n$ satisfying the hypotheses of the lemma, we have a construction of a tree T' which is isomorphic to T but not computably isomorphic to it. Let $\text{ht}(T) = n$, and suppose that every successor tree above r embeds into every other such successor tree, and that there are infinitely many isomorphism types occurring among these successor trees and that they are all of finite type. Each time a node becomes established at level 1 in T, we know that it is the root of a successor tree, so we define it to be the next x_i. Since the successor trees all embed into each other, they must all have the same height – namely $n-1$, since $\text{ht}(T) = n$ – so every node at level 1 of T eventually is identified as x_i for some i.

Notice that for x_i and x_j at level 1 in T, we have that $T[x_i] \equiv T[x_j]$ and that both of these trees are of finite type. Therefore, by Lemma 3.9, the set of isomorphism types which occur infinitely often among the successor trees of x_i in $T[x_i]$ is exactly the same as the set of isomorphism types which occur infinitely often among the successor trees of x_j in $T[x_j]$. Let these types be $I_1, \dots I_p$. We will consider two cases.

Case 1. Suppose there are infinitely many i such that some finite-appearing successor tree in $T[x_i]$ embeds into one of $I_1, \dots I_p$. Then (without loss of generality) there must be infinitely many i such that some finite-appearing successor tree in $T[x_i]$ embeds into I_1. The construction in this case will be much the same as the construction in the case where $\text{ht}(T) = 4$. The witness nodes will be roots of finite-appearing successor trees in various $T[x_i]$, and we will embed those finite-appearing trees into successor trees in $T[x_i]$ isomorphic to I_1 when necessary to satisfy the requirements. In this general case, however, it is more difficult to locate the witness nodes.

Since I_1 has strongly finite type, we can use finitely much information to construct a nice copy of I_1. By a nice copy, we mean both that we know the isomorphism type of every subtree of the form $I_1[a]$ and also that $x \prec y$ implies that $x < y$. We use this copy of I_1, along with the notion of a basic embedding, to pick out witness nodes in T.

Definition 3.21 Two nodes x and y in a tree S are *siblings* if they have the same immediate predecessor in S. (This includes the case $x = y$.)

Definition 3.22 An embedding $\psi : S \hookrightarrow T$ is *basic* if it maps the root of S to the root of T and for every pair of siblings $y_0 < y_1$ in T, if $T[y_0] \cong T[y_1]$ and $y_1 \in \text{range}(\psi)$, then also $y_0 \in \text{range}(\psi)$ and $\psi^{-1}(y_0) < \psi^{-1}(y_1)$. (Here, of course, $<$ refers to the standard ordering on ω, not to the tree structure of T or S.)

The intuition for building a (not necessarily computable) basic embedding is that, having mapped x to $\psi(x)$, we consider the immediate successors $x_0, x_1, \dots$ of x in numerical order (so $x_i < x_{i+1}$ for all i). Having defined ψ on $x_0, \dots x_i$, we choose an isomorphism type above $\psi(x)$ into which to map $S[x_{i+1}]$, and let $\psi(x_{i+1})$ be the least root y of a successor tree of that isomorphism type above $\psi(x)$ such that y is not already in the range of ψ. The only problem with this algorithm is that several successor trees from S may have to map into the same finitely occurring successor tree in T. We show how to handle this problem below. Notice that for our nice copy of I_1, it is computable for finite trees S, uniformly in S, whether a basic embedding of S into I_1 exists, and also whether any specific map $\psi : S \to I_1$ is a basic embedding or not.

We prove the following sublemmas about basic embeddings. Although they apply to any trees of strongly finite type, we will apply them to our nice copy of I_1.

Sublemma 3.23 *Let S be a tree with finite type and U be a tree with strongly finite type. If there is an embedding $f : S \hookrightarrow U$, then there is a basic embedding $g : S \hookrightarrow U$.*

Proof. We proceed by induction on the height of U. If U has height 1, then S must have height 1 and they both consist only of a root. The basic embedding g sends the root of S to the root of U. Assume U has height greater than 1 and that we know the theorem by induction for all trees of lower heights.

Fix the embedding f and define g to send the root of S to the root of U. We describe how g behaves on all successor trees $S[x]$ of the root in S by splitting into two cases. Let $y_0, y_1, \ldots$ be all the nodes at level 1 in U numbered so that $i < j$ implies $y_i < y_j$. Assume that the successor trees $U[y_i]$ for $i \leq n$ are exactly the successor trees whose isomorphism types occur only finitely often in U. (The proof below does not depend on the fact that the finitely occurring successor trees have roots which are less than the roots of the infinitely occurring successor trees. It does, however, simplify the notation.) We first consider those successor trees of S that f embeds into some $U[y_i]$ for $i > n$ and second we consider those successor trees of S which f embeds into some y_i for $i \leq n$.

Let $x_{i_0} < x_{i_1} < \cdots$ be the nodes at level 1 in S such that f embeds $S[x_{i_k}]$ into one of the infinitely occurring isomorphism types of successor trees in U. Fix j_k to be the index of the node y_{j_k} at level 1 in U such that $f : S[x_{i_k}] \hookrightarrow U[y_{j_k}]$. Define g on the trees $S[x_{i_k}]$ by recursion on k. Let $g(x_{i_k}) = y$ where y is the $\leq_{\mathbb{N}}$–least node at level 1 in U such that $U[y] \cong U[y_{j_k}]$ and y is not in the range of g yet. Since $S[x_{i_k}] \hookrightarrow U[y]$, by induction there is a basic embedding of these trees. Let g be defined on $S[x_{i_k}]$ to be such an embedding.

Next, consider the successor trees $S[x]$ in S such that f maps $S[x]$ into one of the finitely occurring isomorphism types of successor trees in U. Notice that if f maps two successor trees $S[x_1]$ and $S[x_2]$ into the same successor tree $U[y]$, then y is not in the range of f. We consider each of the finitely occurring isomorphism types separately. Fix one of these types and assume without loss of generality that $U[y_0], \ldots, U[y_m]$ are the successor trees with this isomorphism type and that $y_0 < \cdots < y_m$. For $i \leq m$, let Y_i be the set of nodes x at level 1 in S such that f maps $S[x]$ into $S[y_i]$. We consider first the sets Y_i which have size 1 and then the sets which have size at least 2. (Of course, it is possible that some Y_i are empty and we ignore these sets.)

For each Y_i with size 1, fix the unique successor tree in S which maps into $U[y_i]$. Let $x_{i_0} < \cdots < x_{i_k}$ be the root nodes of these successor trees. Define $g(x_{i_l}) = y_l$ for $l \leq k$. Since we know $S[x_{i_l}] \hookrightarrow U[y_l]$, we can extend our definition of g (by the induction hypothesis) to be a basic embedding between these subtrees.

We consider the remaining Y_i with size at least 2 individually. Fix such a Y_i and let $x_0, x_1, \ldots \in Y_i$ be the nodes of level 1 in S such that f embeds $S[x_k]$ into $U[y_i]$. (This list may be either finite or infinite.) Consider an auxiliary tree S_i formed by taking a root node and attaching the trees $S[x_k]$ immediately above the root. Let $u \leq m$ be the least index such that we have not defined g mapping into $U[y_u]$ yet. By our assumptions, we know that S_i embeds in $U[y_u]$. (Notice that this is where we use the fact that Y_i has size at least 2. In this case, the node y_i was not in the image of f because f mapped more than one successor tree into $U[y_i]$.) By the induction hypothesis, there is a basic embedding of S_i into $U[y_u]$. Let g be the restriction of such a basic embedding to all nodes in S_i except the root node. This definition of g maps all of the $S[x_k]$ trees into U in a basic way. (That is, any violation of the requirement on the images of siblings in the restricted version of g would have been a violation of the restriction on the basicness of the embedding of S_i.) Performing the action of the last two paragraphs for each isomorphism type of a finitely occurring successor tree in U completes the description of g. ∎

Sublemma 3.24 *Let U be a finite height tree of strongly finite type and let $f : U \hookrightarrow U$. Then for all x at level 1, we have that $f(x)$ is at level 1 and $U[x] \cong U[f(x)]$. (We are not claiming that f is an isomorphism, which it need not be, but only that these successor trees are isomorphic.)*

Proof. Fix a node x at level 1 in U such that $f(x) \neq x$. We split into the cases when $U[x]$ is a finitely occurring isomorphism type and when $U[x]$ is an infinitely occurring isomorphism type.

Suppose $U[x]$ is a finitely occurring isomorphism type and there is some $m < n$ such that $f^m(x) = f^n(x)$. Now f, being an embedding, has a one-to-one inverse g, with $\text{dom}(g) = \text{range}(f)$. So $f^{n-m}(x) = g^m(f^n(x)) = g^m(f^m(x)) = x$, forcing

$$U[x] \hookrightarrow U[f(x)] \hookrightarrow U[f^{n-m}(x)] = U[x].$$

Since these are strongly finite trees, $U[x] \cong U[f(x)]$ by Lemma 2.10, and moreover, $1 \leq \text{level}(f(x)) \leq \text{level}(f^{n-m}(x)) = 1$.

Now suppose $U[x]$ is a finitely occurring isomorphism type but there is no $m < n$ such that $f^n(x) = f^m(x)$. We first show that there must be a n_1 such that $\text{level}(f^{n_1}(x)) > 1$. Because $U[x]$ is a finitely occurring isomorphism type and U has strongly finite type, if $U[x] \hookrightarrow U[y]$ and $\text{level}(y) = 1$, then $U[y]$ is a finitely occurring successor tree. Fix y such that $\text{level}(y) = 1$ and f embeds $U[x]$ into $U[y]$. There are two possibilities, either $y \prec f(x)$ (in which case $\text{level}(f(x)) > 1$ and we are done) or $y = f(x)$ (in which case $\text{level}(f(x)) = 1$). If $f(x) = y$, then we repeat the above process to gain information about $f^2(x)$. Since $U[y] = U[f(x)]$ is a finitely occurring successor tree, f must embed $U[f(x)]$ into a finitely occurring successor tree $U[z]$ with $\text{level}(z) = 1$. Again, either we have $z \prec f^2(x)$ (in which case we are done) or $z = f^2(x)$. In the latter case, we repeat the process again. Each time we repeat this process, we either find that $\text{level}(f^n(x)) > 1$ (and we are finished) or $f^n(x)$ is the root of another finitely occurring successor tree. Since $f^n(x) \neq f^m(x)$ for all $n \neq m$, we can never repeat the root of a particular finitely occurring successor tree during this process. Because there are only finitely many finitely occurring successor trees, this process must stop at some value n_1 with level$(f^{n_1}(x)) > 1$.

Let x_1 be the node at level 1 such that $x_1 \prec f^{n_1}(x)$. We next show that it is not the case that $f^k(x_1) = x_1$ for some k. For a contradiction, assume that $f^k(x_1) = x_1$ for some k. Then, because $f^{p+k}(x_1) = f^p(x_1)$ for all p, there must be a node y_1 such that $f^{n_1}(y_1) = x_1$. However, then we have $f^{n_1}(y_1) = x_1 \prec f^{n_1}(x)$, so $y_1 \prec x$ which implies that y_1 is the root. Since f must take the root to the root, this gives the desired contradiction.

Repeating the argument above for x_1, there must be an n_2 such that level$(f^{n_2}(x_1)) > 1$. Therefore,

$$\text{level}(f^{n_1+n_2}(x)) > \text{level}(f^{n_2}(x_1)) > 1$$

and hence level$(f^{n_1+n_2}(x)) > 2$. If we now let x_2 be the node of level 1 such that $x_2 \prec f^{n_2}(x_1)$, we can repeat the argument to show there is an n_3 such that level$(f^{n_1+n_2+n_3}(x)) > 3$. Repeating this process contradicts the fact that U has finite height. Therefore, it cannot be the case that $f^k(x) \neq x$ for all k. We have now shown that f must permute the successor trees $U[x]$ which have finitely occurring isomorphism types.

It remains to consider x such that $U[x]$ has an infinitely occurring isomorphism type. By the argument above, f cannot map $U[x]$ into a finitely occurring isomorphism type because it must permute those types. Therefore, f must embed $U[x]$ into some $U[z]$ which has an infinitely occurring isomorphism type. However, this means that $U[x] \cong U[z]$ since U has strongly finite type and hence we must have $f(x) = z$. ■

Sublemma 3.25 *If $T_1 \cong T_2$ have strongly finite type and $f : T_1 \hookrightarrow T_2$, then for all x at level 1 in T_1, $f(x)$ is at level 1 in T_2 and $T_1[x] \cong T_2[f(x)]$.*

Proof. Let g be any isomorphism from T_2 to T_1. This sublemma follows from Sublemma 3.24 by considering $gf : T_1 \hookrightarrow T_1$. Notice that as in the proof of Sublemma 3.24, if $T_1[x]$ has finitely occurring isomorphism type, then $T_2[f(x)]$ has finitely occurring isomorphism type. Furthermore, if y is at level 1 in T_2 and $T_2[y]$ has finitely occurring isomorphism type, then y is in the range of f. ■

Sublemma 3.26 *Let U be a finite height tree of strongly finite type, $f : U \hookrightarrow U$, and $k < ht(U)$. For all nodes x at level k, $f(x)$ has level k and $U[x]$ is isomorphic to $U[f(x)]$.*

Proof. This follows by induction on k using Sublemma 3.24. ■

Sublemma 3.27 *Let $T_1 \cong T_2$ be finite height trees of strongly finite type. There is exactly one basic embedding $f : T_1 \hookrightarrow T_2$ and f is an isomorphism.*

Proof. We proceed by induction on the height of T_1. The case for height 1 is trivial. Assume the sublemma holds for all trees of height less than the height of T_1. By Sublemma 3.23, we know that there is a basic embedding $f : T_1 \hookrightarrow T_2$. We need to show that f is onto (and therefore is an isomorphism) and is unique.

We know that f sends the root of T_1 to the root of T_2. Consider any node x at level 1 in T_2. We have already seen that if x has an infinitely occurring isomorphism type then x cannot be equal to $f(y)$ where $T_1[y]$ has finitely occurring type. Also, if x is not in the range of f, then we have an immediate contradiction since some $z > x$ with $T_2[z] \cong T_2[x]$ must be in the range of f because T_1 has infinitely many nodes u at level 1 with $T_1[u] \cong T_2[x]$. This contradicts the fact that f is basic. Therefore, x must be in the range of f. Furthermore, if x is the k-th node at level 1 in T_2 with its isomorphism type (where we measure k-th using the $\leq_{\mathbb{N}}$–ordering), it must be the case that $f(u) = x$ where u is the k-th node in T_1 with this type. Therefore, for nodes at level 1 in T_1 with infinitely occurring isomorphism types, the map f is uniquely determined. By induction, the values of f above these nodes are uniquely determined and give an isomorphism between the successor trees.

If $T_2[x]$ is a finitely occurring type, then we already know that $x = f(y)$ for some $y \in T_1$. By an argument similar to the one above, the value of y is uniquely determined, and by the induction hypothesis, f is a uniquely determined isomorphism from $T_1[y]$ to $T_2[x]$. ■

Sublemma 3.28 *For each basic embedding $\psi : S \hookrightarrow I_1$, the restriction of ψ to $S \cap \{r, 0, \dots s\}$ is also a basic embedding. (Here r is the root of S.)*

This is clear from the definition of basic embedding. ■

Sublemma 3.29 *Let S and I_1 be finite height trees such that S is of finite type and I_1 is of strongly finite type. Suppose every basic embedding $S \hookrightarrow I_1$ includes the node y of I_1 in its image. Then there is an s such that every basic embedding of $S \cap \{r, 0, 1, \dots, s\}$ into I_1 includes y in its image.*

It then follows from Sublemma 3.28 that every basic embedding of every $S \cap \{0, 1, \dots, t\}$ into I_1 with $t \geq s$ includes y in its image.

Proof. Our argument is purely classical and we do not claim (or need) any effectiveness in this sublemma. We proceed by induction on the height of I_1. The case when I_1 has height 1 (and hence consists of only the root) is trivial. Assume that the height of I_1 is greater than 1 and that the sublemma holds for all trees of shorter height. We split the argument into two cases: when y is contained in one of the finitely occurring successor trees in I_1 and when y is contained in one of the infinitely occurring successor trees in I_1. It suffices to show that if y is contained in the range of all basic embeddings, then there is a finite subtree U of S for which all basic embeddings of U into I_1 hit y.

First, consider the case when y is contained in one of the finitely occurring successor trees of I_1. Let S_1 be the subtree of S consisting of the root plus all the successor trees which do not embed into any infinitely occurring successor tree in I_1. Let J_1 be the subtree of I_1 containing the root and all the finitely occurring successor trees in I_1. We denote the successor trees in J_1 by $J_1[z_0], \dots, J_1[z_l]$. Notice that not only does $S_1 \hookrightarrow J_1$, but there is a basic embedding $S \hookrightarrow I_1$ such that $S \setminus S_1$ is mapped into $I_1 \setminus J_1$. This basic embedding can be obtained by fixing a basic embedding $S_1 \hookrightarrow J_1$ and then mapping each successor tree $S[x]$ not in S_1 (by induction on x) by a basic embedding into $I_1[z]$ where z is the $<$-least node at level 1 in $I_1 \setminus J_1$ which has not yet been mapped into.

We denote the finitely occurring successor trees in S_1 by $S_1[x_0], \ldots, S_1[x_k]$ with the assumption that $x_0 < \cdots < x_k$. For simplicity of notation, we assume that there is only one isomorphism type for an infinitely occurring successor tree in S_1 (the general case when there are finitely many such isomorphism types will be clear from the argument below) and we denote these successor trees $S_1[y_0], S_1[y_1], \ldots$ with the assumption that $y_0 < y_1 < \cdots$. We make no assumptions about the $<$-ordering between elements x_i and y_j.

We claim that for any $z \in J_1$ at level 1, either infinitely many copies of $S_1[y_0]$ can be embedded into $J_1[z]$ or else there is a finite upper bound m_z on the number of copies of $S_1[y_0]$ that can be embedded into $J_1[z]$. To see this fact, consider an auxiliary tree U formed by taking a root with infinitely many copies of $S_1[y_0]$ as successor trees. (Notice that since $S_1[y_0]$ has finite type, so does U.) Because U has finite type, we know that it embeds into $J_1[z]$ if and only if all of its finite subtrees embed into $J_1[z]$. Therefore, if $U \not\hookrightarrow J_1[z]$, then we obtain a finite bound m_z as above. Since there are only finitely many successor trees in J_1, we let m' be a finite number such that for all $z \in J_1$ at level 1, if m' copies of $S_1[y_0]$ embed into $J_1[z]$, then infinitely many copies of $S_1[y_0]$ embed into $J_1[z]$. Finally, since there are $l+1$ many successor trees in J_1, then we let $m = m'(l+1)$. The point of m is that we know that if m many copies of $S_1[y_0]$ are embedded into J_1, then at least m' many must have been embedded into some $J_1[z_j]$ and hence infinitely many copies of $S_1[y_0]$ could have been embedded into that successor tree.

We next define a finite tree $S_1' \subset S_1$ whose basic embeddings into J_1 will encode (in a way made precise below) the possible successor trees $J_1[v]$ that a successor tree $S_1[u]$ could be sent to by a basic embedding of S into I_1. S_1' consists of the root plus successor trees $S_1'[x_i] \subset S_1[x_i]$ for $i \leq k$ and $S_1'[y_0], \ldots, S_1'[y_m]$. Each of the $S_1'[y_i]$ trees are isomorphic with $S_1'[y_0] \subset S_1[y_0]$. We pick these finite trees to have the following embedding properties, where z ranges over all nodes at level 1 in J_1.

1. $S_1'[x_i] \not\hookrightarrow I_1[v]$ for any $v \in I_1 \setminus J_1$ (and the same for $S_1'[y_i]$).

2. $S_1'[x_i] \hookrightarrow J_1[z]$ if and only if $S_1[x_i] \hookrightarrow J_1[z]$ (and similarly for $S_1'[y_i]$).

3. If $S_1[x_i] \hookrightarrow J_1[z]$ only by sending x_i to z, then the same property holds for $S_1'[x_i]$ (and similarly for $S_1'[y_i]$).

4. Consider all possible choices of nonrepeating sequences $u_0, \ldots, u_q$ with each u_a equal either to some x_i (with $i \leq k$) or some y_j (with $j \leq m$). Let U denote the tree formed by taking a root and successor trees $S_1[u_0], \ldots, S_1[u_q]$ and let U' denote the finite tree formed by taking a root and successor trees $S_1'[u_0], \ldots, S_1'[u_q]$. Then, $U \hookrightarrow J_1[z]$ if and only if $U' \hookrightarrow J_1[z]$.

The fact that we can pick finite subtrees with properties (1), (2) and (4) is clear from Lemma 2.9. To see that we can get property (3), suppose that all finite subtrees of $S_1[x_i]$ containing

x_i can be embedded into $J_1[z]$ without sending x_i to z. Then for one of the finitely many isomorphism types of the successor trees in $J_1[z]$, arbitrarily large subtrees of $S_1[x_i]$ can be embedded into this type. But, then $S_1[x_i]$ can be embedded into a successor tree of $J_1[z]$ with this type, and hence there is an embedding of $S_1[x_i]$ into $J_1[z]$ which does not send x_i to z.

Property (1) says that any embedding $S_1' \hookrightarrow I_1$ must actually send S_1' into J_1. Properties (2) and (4) together say that any embedding $S_1' \hookrightarrow J_1$ can be extended to an embedding $S_1 \hookrightarrow J_1$ and hence to an embedding $S \hookrightarrow I_1$. Property (3) says that if this extension requires that a node at level 1 in S_1 map to a node at level 1 in J_1, then this requirement was already present for the embedding of S_1'.

For any basic embeddings $f, g : S_1' \hookrightarrow J_1$, we say $f \sim g$ if and only if the following two conditions hold.

$$\forall i \leq k \, \forall j \leq l \, \big((f(x_i) \in J_1[z_j] \leftrightarrow g(x_i) \in J_1[z_j]) \wedge (f(x_i) = z_j \leftrightarrow g(x_i) = z_j) \big)$$
$$\forall i \leq m \, \forall j \leq l \, \big((f(y_i) \in J_1[z_j] \leftrightarrow g(y_i) \in J_1[z_j]) \wedge (f(y_i) = z_j \leftrightarrow g(y_i) = z_j) \big)$$

It is clear that $\sim$ is an equivalence relation and that up to $\sim$ equivalence, there are only finitely many basic embeddings $S_1' \hookrightarrow J_1$. Let $f_0, \ldots, f_q$ be a list containing one element from each equivalence class. For a basic embedding $g : S \hookrightarrow I_1$, we say that $g \sim f_i$ if the restriction of g to S_1' is equivalent to f_i.

Our definition of m and the properties (1)–(4) above insure that for every basic embedding $g : S \hookrightarrow I_1$, there is an f_i such that $g \sim f_i$ and that for each f_i, there is a basic embedding $g : S \hookrightarrow I_1$ such that $g \sim f_i$. It is in this sense that the embeddings $f_0, \ldots, f_q$ encode information about the possible basic embeddings of S into I_1 when restricted to S_1.

We use the f_i embeddings to prove the sublemma in the case when y is a node in J_1. First, by the properties of the previous paragraph, a node z_j at level 1 in J_1 is in the range of all basic embeddings $S \hookrightarrow I_1$ if and only if z_j is in the range of f_i for all $i \leq q$. Therefore, the finite tree S_1' is large enough to determine if the roots of the finitely occurring successor trees in I_1 are in the range of all basic embeddings.

Second, suppose that $y \in J_1[z_j]$, but $y \neq z_j$. For each $i \leq q$, we define a tree U_i corresponding to f_i. Fix i and consider all $u \in \{x_0, \ldots, x_k, y_0, \ldots, y_m\}$ such that f_i maps $S_1'[u]$ into $J_1[z_j]$. We split the definition of U_i into two cases. If f_i maps less than m' copies of trees of the form $S_1'[y_a]$ into $J_1[z_j]$, then let U_i consist of a root plus the successor trees $S_1[u]$ for which f_i maps $S_1'[u]$ into $J_1[z_j]$. If f_i maps at least m' many $S_1'[y_a]$ trees into $J_1[z_j]$, then let U_i consist of a root plus the successor trees $S_1[x_b]$ for which f_i maps $S_1'[x_b]$ into $J_1[z_j]$ plus infinitely many successor trees isomorphic to $S_1[y_0]$. In either case, we know that U_i embeds into $J_1[z_j] = I_1[z_j]$. Since the height of $I_1[z_j]$ is strictly less that the height of I_1, we can apply the induction hypothesis on height from the beginning of the sublemma. There is a finite subtree U_i' such that every basic embedding of U_i' into $I_1[z_j]$ has y in its range. We expand each finite tree $S_1'[u]$ to include the finite tree $U_i' \cap S_1[u]$. Once we have expanded S_1' by performing this action for each $i \leq q$, we have that each basic embedding $S_1' \hookrightarrow J_1$ must include y in its range.

We now consider the case when y is an element of one of the infinitely occurring successor trees in I_1. We begin by establishing facts about about when an infinitely occurring successor tree $I_1[x]$ is not in the range of every basic embedding $S \hookrightarrow I_1$. (Think of x as the node at level 1 in I_1 such that $y \in I_1[x]$.) First, if there is a basic embedding which sends two successor trees of S into $I_1[x]$, then x is obviously not in the range of this embedding. Second, if there is some basic embedding f such that x is not in the range of f, then there is a basic embedding g such that range$(g) \cap I_1[x] = \emptyset$. To see this second fact, let $x = x_0 < x_1 < \cdots$ be the nodes at level 1 in I_1 with $x_i \geq x$ and $I_1[x] \cong I_1[x_i]$. Since f is basic and $x \notin$ range(f), we know that for all i, $x_i \notin$ range(f). Fix isomorphisms $h_i : I_1[x_i] \to I_1[x_{i+1}]$ which are also basic embeddings. (Such maps exist by Sublemma 3.27.) Define $g(u)$ as follows. If $f(u) \notin I_1[x_i]$ for any i, then $g(u) = f(u)$. If $f(u) \in I_1[x_i]$, then let $g(u) = h_i(f(u))$. Thus, g shifts the image of f on $I_1[x_i]$ to $I_1[x_{i+1}]$. Because $x_i \notin$ range(f), g is a basic embedding.

From the previous paragraph, we see that if x is the node at level 1 in I_1 such that $y \in I_1[x]$, then x must be in the range of all basic embeddings and each basic embedding maps a single successor tree $S[u]$ into $I_1[x]$. We next show there is a bound on how big u can be. For a contradiction, assume that there is no such bound. Let $J_1, \dots, J_m$ be the isomorphism types of successor trees in S that can be embedded into $I_1[x]$ by a basic embedding. Assume that $J_1, \dots, J_l$ are types which occur finitely often as successor trees in S and $J_{l+1}, \dots, J_m$ are types which occur infinitely often. Because there is no bound on the u such that $S[u]$ is embedded into $I_1[x]$ by a basic embedding, there must be basic embeddings which map all occurrences of the types $J_1, \dots, J_l$ in S as well as arbitrarily many copies of each of the types $J_{l+1}, \dots, J_m$ into the successor trees $I_1[v]$ with $v < x$. However, as we saw above, if arbitrarily many finite copies of some J_i can be embedded into some $I_1[v]$, then infinitely many copies can be embedded into $I_1[v]$. Therefore, there must be a basic embedding which sends all copies of the types $J_1, \dots, J_m$ from S into the successor trees $I_1[v]$ for $v < x$. This means that there is a basic embedding which does not map into $I_1[x]$ at all, contradicting our assumption that y is in the range of all basic embeddings.

We conclude that if $I_1[x] \cap$ range$(f) \neq \emptyset$ for all basic f (and hence $x \in$ range(f)), then there is a bound on the elements u at level 1 such that $S[u]$ is mapped into $I_1[x]$ by a basic embedding. We can therefore limit the "level 1 width" of S which we need to consider when looking at how much of S is required to force the basic embeddings to intersect $I_1[x]$. This bound means that by an argument similar to the one when y was assumed to be from a finitely occurring successor tree in I_1, we isolate a finite tree $U \subset S$ such that every basic embedding $U \hookrightarrow I_1$ must hit x. From here, we again consider the finitely many successor trees in S which could map into $I_1[x]$ by a basic embedding and apply the inductive hypothesis on the height to handle the nodes $y \in I_1[x]$ with $y \neq x$. ■

We now begin our description of the construction for Case 1 of Lemma 3.19. The *witness node in $T[x_i]$ at stage s* will be a node z at level 1 in $T_s[x_i]$ such that $T_s[z] \hookrightarrow I_1$ but for which we believe $T[z] \ncong I_1$. The first condition is easy to check, using our canonical copy of I_1. To satisfy the second condition, we want there to be a basic embedding of $T[z]$ into I_1 which is not surjective. Since I_1 is of strongly finite type, Sublemma 3.27 will then ensure that $T[z] \ncong I_1$, and we will use this fact to diagonalize.

At stage s, we define the witness node in $T_s[x_i]$ by finding the least pair $\langle z, y\rangle$ such that z is at level 1 in $T_s[x_i]$ and there is a basic embedding of $T_s[z]$ into I_1 whose image does not contain y. We set $v_{i,s} = \langle z, y\rangle$, since it appears at this stage that $T[z]$ is the successor tree we want in $T[x_i]$, and define this z to be the witness node in $T[x_i]$ at stage s.

Sublemma 3.30 *$\lim_s v_{i,s}$ converges if and only if there exists z at level 1 in $T[x_i]$ such that $T[z]$ embeds into I_1 but is not isomorphic to I_1.*

Proof. Assume $\lim_s v_{i,s} = v_i = \langle z, y\rangle$. Then $T[z]$ must embed into I_1, by Lemma 2.9. On the other hand, there must be a basic embedding of $T[z]$ into I_1 which omits y from its image, by Sublemmas 3.23 and 3.29, and then Sublemma 3.27 ensures that $T[z] \ncong I_1$.

Conversely, suppose that for some z with $\text{level}_{T[x_i]}(z) = 1$ we have $T[z] \hookrightarrow I_1$ but $T[z] \ncong I_1$. By Sublemma 3.23, there is a basic embedding of $T[z]$ into I_1, which cannot be surjective because $T[z] \ncong I_1$. Thus there must be a least pair $\langle z, y\rangle$ such that some basic embedding of $T[z]$ into I_1 omits y. By Sublemma 3.28, we will have $v_{i,s} \leq \langle z, y\rangle$ for all sufficiently large s. Consider any pair $\langle z', y'\rangle < \langle z, y\rangle$. By our choice of $\langle z, y\rangle$, no basic embedding of $T[z']$ into I_1 omits y', so Sublemma 3.29 ensures that $v_{i,t} \neq \langle z', y'\rangle$ for sufficiently large t. Thus $\lim_s v_{i,s} = \langle z, y\rangle$. ∎

However, it is possible that $T[x_i]$ does not contain any finite-appearing successor tree which embeds into I_1, so we must search among different successor trees. At first, we choose $w_{0,0}$ to be the witness node in $T[x_0]$. If $v_{0,s_1} \neq v_{0,0}$ at some subsequent stage s_1, then we choose w_{0,s_1} to be the witness node in $T[x_1]$ at stage s_1. If at a subsequent stage s_2 we have $v_{1,s_2} \neq v_{1,s_1}$, then we change w_{0,s_2} back to the witness node in $T[x_0]$ at stage s_2, then $T[x_1]$ again, then $T[x_2]$, then back to $T[x_0]$, and so on, just as in the construction for trees of height 4. By the assumption of Case 1 (which began on page 47), $\lim_s v_{i,s}$ converges for infinitely many i, so $w_{0,s}$ must eventually converge to some w_0. At the same time, we do the same for the witness node $w_{1,s}$ for $\mathcal{R}_1$, looking only at witness nodes in successor trees $T[x_j]$ in which $w_{0,s}$ has never yet been located, and so on by a standard finite-injury process. The following sublemma is now clear:

Sublemma 3.31 *For every e, $w_e = \lim_s w_{e,s}$ exists and $T[w_e] \hookrightarrow I_1$ and $T[w_e] \ncong I_1$.* ∎

We build T' by copying T at each stage, with the following provision. Find each $e \leq s$ such that $w_{e,s}$ is defined and $\varphi_{e,s}(w_{e,s})\downarrow$ (say $y = \varphi_{e,s}(w_{e,s})$) and $f_s^{-1}(y)$ lies at level 2 in T_s. If there is no such e, simply extend f_s to f_{s+1} by adding new elements to T'_{s+1}. If there is such an e, then for each such e, we check whether there exists a basic embedding of $T_s[f_s^{-1}(y)]$ into I_1 (recall that this is a computable condition using our nice copy of I_1). If no such embedding exists, then we are assured that $T[f_s^{-1}(y)] \not\hookrightarrow I_1$, so if $w_{e,s} = w_e$, then $\mathcal{R}_e$ will be satisfied. If such an embedding does exist, then we add elements to $T'_{s+1}[y]$ to make it a copy of I_1, and add more new elements to T'_{s+1} to be the new image of $T_s[f_s^{-1}(y)]$ under f_{s+1}. Hence $T'[\varphi_e(w_{e,s})] \cong I_1$, so if $w_{e,s} = w_e$, we have satisfied $\mathcal{R}_e$. Notice that f_{s+1} is no longer onto

T'_{s+1} since we have added a copy of I_1. However, we know that I_1 occurs infinitely often as a successor tree above x_i in T, so T and T' are still isomorphic. $\mathcal{R}_i$ is injured if $w_{e,s+1} \neq w_{e,s}$ for some $e < i$.

As with the case for trees of height 4, there are two minor modifications necessary for this strategy. First, we give $2e+1$ many witnesses to each $\mathcal{R}_e$ strategy and force this requirement to respect $\mathcal{N}_u$ and $\mathcal{M}_u$ for $u < e$. That is, $\mathcal{R}_e$ is forbidden to use $w_{e,s}$ if $u \prec f_s^{-1}(\varphi_{e,s}(w_{e,s}))$ for some $u < e$ or if $\varphi_{e,s}(w_{e,s}) < e$. The reason for explicitly adding the $\mathcal{M}_u$ requirements will become clear below when we discuss the general case where our set of nodes X from the statement of Lemma 3.19 is Δ^0_2 rather than computable. Second, it is possible that $\varphi_e(w_{e,s}) = y$ lies in a copy of I_1 in T_1. In this case, if $w_e = w_{e,s}$, then $\mathcal{R}_e$ is satisfied without any further action.

We end this case with some comments on how to combine the Δ^0_2 approximation for the x_i elements with the strategy just described. Suppose we have an element x which we think is an x_i element and we diagonalize at stage s using a successor of x. This means that we want to create a copy of I_1 as a successor tree of $T'[f_s(x)]$. The worry is that if we put I_1 down all at once, it may turn out that x is not one of the x_i elements, and even worse, that x has no successor tree of type I_1. Such an outcome could destroy our isomorphism. Therefore, we fix an approximation $I_{1,s}$ to I_1 by finite subtrees. Instead of putting down all of I_1 at once as a successor tree of $T'[f_s(x)]$, we build up I_1 by putting down $I_{1,t}$ at stage $t \geq s$. Furthermore, before putting down $I_{1,t}$, we check for either

1. evidence that x is not an x_i element; or
2. a successor y of x such that y is bigger than any number seen so far in the construction and $I_{1,t}$ embeds into $T[y]$.

This search procedure must terminate since if x is an x_i element, then x has infinitely many successor trees of type I_1. The reason for including clause (2) in the search is that if we find evidence that x is not an x_i element at stage $t+1$, then we can use the successor tree $T[y]$ found at stage t to map to the copy of $I_{1,t}$ currently sitting as a successor tree of $T'_t[f_t(x)]$. Since we can correct any mistakes caused by the Δ^0_2 approximation, it is straightforward to add it in as a formal part of the above construction. Finally, notice that when we take into account this Δ^0_2 approximation procedure, we can have elements $y \in T'$ which leave the range of f and later return to the range of f. This is the reason why we need to explicitly add the $\mathcal{M}_u$ requirements into this construction.

Case 2. If Case 1 (which began on page 47) does not apply, then there are only finitely many $T[x_i]$ in which some finite-appearing successor tree embeds into any of $I_1, \ldots, I_p$. We assume finitely much information, namely the roots of those finitely many successor trees, and ignore them in our construction, defining the elements x_i to be those nodes at level 1 in T which are *not* roots of these finitely many successor trees.

Consider any embedding $T[x_i] \hookrightarrow T[x_j]$ among these nodes at level 1 in T. Since no successor tree which occurs finitely often in $T[x_i]$ can embed into any of $I_1, \ldots, I_p$, we know that they must embed into the successor trees which occur finitely often in $T[x_j]$. Of course, the same relation holds for embeddings $T[x_j] \hookrightarrow T[x_i]$. Therefore, if we let T_i (respectively T_j) be the tree formed by taking a root and adjoining all of the finitely occurring successor trees in $T[x_i]$ (in $T[x_j]$ respectively), then we have $T_i \equiv T_j$.

Sublemma 3.32 *Fix $f : T_i \hookrightarrow T_j$ and $g : T_j \hookrightarrow T_i$. Then for any $u \in T_i$ at level 1, there is a $v \in T_j$ at level 1 such that $T_i[u] \equiv T_j[v]$, and vice versa.*

Proof. Fix any $u_1 \in T_i$ at level 1 and let $v_1 \in T_j$ be such that v_1 is at level 1 and f maps $T_i[u_1] \hookrightarrow T_j[v_1]$. If g maps $T_j[v_1] \hookrightarrow T_i[u_1]$, then we are done, so assume this does not happen. Let $u_2 \neq u_1$ be at level 1 in T_i such that g maps $T_j[v_1] \hookrightarrow T_i[u_2]$.

Fix $v_2 \in T_j$ at level 1 such that f maps $T_i[u_2] \hookrightarrow T_j[v_2]$. We claim that $v_2 \neq v_1$. For a contradiction, suppose that $v_2 = v_1$. Then f maps $T_i[u_2] \hookrightarrow T_j[v_2] = T_j[v_1]$ and g maps $T_j[v_1] \hookrightarrow T_i[u_2]$. Therefore, $T_i[u_2] \equiv T_j[v_1]$ which means $f(u_2) = v_2 = v_1$. But, f also maps $T_i[u_1] \hookrightarrow T_j[v_1]$, so $v_1 \prec f(u_1)$. Together, these statements imply that $f(u_2) \prec f(u_1)$, which contradicts the fact that u_1 and u_2 are incomparable nodes at level 1 in T_i.

We now check whether g maps $T_j[v_2] \hookrightarrow T_i[u_1]$. If so, then we have

$$T_i[u_1] \hookrightarrow T_j[v_1] \hookrightarrow T_i[u_2] \hookrightarrow T_j[v_2] \hookrightarrow T_i[u_1].$$

In this case, $T_i[u_1] \equiv T_i[u_2] \equiv T_j[v_1] \equiv T_j[v_2]$ with $f(u_1) = v_1$, $f(u_2) = v_2$, $g(v_1) = u_2$ and $g(v_2) = u_1$, which means we are done. Otherwise, fix $u_3 \in T_i$ at level 1 such that $u_3 \neq u_1$ and g maps $T_j[v_2] \hookrightarrow T_i[u_3]$.

We claim that $u_3 \neq u_2$. For a contradiction, suppose that $u_3 = u_2$. Then g maps $T_j[v_2] \hookrightarrow T_i[u_3] = T_i[u_2]$ and f maps $T_i[u_2] \hookrightarrow T_j[v_2]$. Therefore, $T_j[v_2] \equiv T_i[u_2]$ and so $g(v_2) = u_2$. But, g also maps $T_j[v_1] \hookrightarrow T_i[u_2]$ which means $u_2 \prec g(v_1)$. Therefore, $g(v_2) \prec g(v_1)$. This contradicts the fact that v_1 and v_2 are incomparable nodes at level 1 in T_j.

We next let v_3 be a node at level 1 in T_j such that f maps $T_i[u_3] \hookrightarrow T_j[v_3]$. We claim that v_3 is not equal to either v_1 or v_2. For a contradiction, suppose $v_3 = v_1$. Then f maps $T_i[u_3] \hookrightarrow T_j[v_3] = T_j[v_1]$ and the composition gfg maps $T_j[v_1] \hookrightarrow T_i[u_3]$. Therefore, $T_i[u_3] \equiv T_j[v_1]$ and $f(u_3) = v_3 = v_1$. But, we also know that f maps $T_i[u_1] \hookrightarrow T_j[v_1]$ which implies $v_1 \prec f(u_1)$. Together, these statements say that $f(u_3) \prec f(u_1)$ which contradicts the fact that u_1 and u_3 are incomparable nodes at level 1 in T_i. The argument that $v_3 \neq v_2$ is similar.

We continue by induction. Suppose $u_1, \ldots, u_n$ are pairwise distinct nodes at level 1 in T_i and $v_1, \ldots, v_n$ are pairwise distinct nodes at level 1 in T_j such that f maps $T_i[u_k] \hookrightarrow T_j[v_k]$ (for $k \leq n$) and g maps $T_j[v_k] \hookrightarrow T_i[u_{k+1}]$ (for $k < n$). We check if g maps $T_j[v_n] \hookrightarrow T_i[u_1]$. If so, then

$$T_i[u_1] \equiv \cdots \equiv T_i[u_n] \equiv T_j[v_1] \equiv \cdots \equiv T_j[v_n]$$

and $f(u_k) = v_k$ (for $k \leq n$), $g(v_k) = u_{k+1}$ (for $k < n$) and $g(v_n) = u_1$. In this case, we are done.

Otherwise, we fix $u_{n+1} \neq u_1$ at level 1 in T_i such that g maps $T_j[v_n] \hookrightarrow T_i[u_{n+1}]$. We argue as above that $u_{n+1} \neq u_k$ for all $k \leq n$. We let v_{n+1} be a node at level 1 in T_j such that f

maps $T_i[u_{n+1}] \hookrightarrow T_j[v_{n+1}]$. We argue as above that $v_{n+1} \neq v_k$ for all $k \leq n$. We are now in position to continue the induction.

Since there are only finitely many nodes at level 1 in T_i and T_j, this process must come to an end. Therefore, we get our result in the end. Notice that this proof shows that the number of nodes at level 1 in T_i is less that or equal to the number of nodes at level 1 in T_j. If we switch the roles of T_i and T_j, we get that for any $u \in T_j$ at level 1, there is a $v \in T_i$ at level 1 such that $T_j[u] \equiv T_i[v]$. Therefore T_i and T_j have the same number of nodes at level 1. ■

This sublemma tells us that each $T[x_i]$ has the same number of finitely occurring successor trees. Let $y_1, \ldots, y_q$ be the roots of the finitely occurring successor trees in $T[x_0]$. We define an equivalence relation by $y_i \sim y_j$ if $T[y_i] \equiv T[y_j]$. The embedding relation $\hookrightarrow$ between these classes is well defined. Furthermore, we know by Sublemma 3.32 that the classes defined in a similar way for the finitely occurring successor trees of any other $T[x_i]$ are exactly the same and each equivalence class has exactly the same size.

For each y_n, fix a finite tree $S_n \subset T[y_n]$ such that S_n does not embed into any of the infinite types $I_1, \ldots, I_p$. Furthermore, for each $m \leq q$ such that $T[y_n] \not\hookrightarrow T[y_m]$, we extend S_n to a finite tree such that $S_n \not\hookrightarrow T[y_m]$. Of course, these finite trees S_n have the same properties relative to the finitely occurring successor trees above any other $T[x_i]$. Therefore, we can use these trees to identify the finitely many finitely occurring successor trees above each node x_i. That is, for each x_i we look for the appropriate number of successors y such that $T[y]$ contains one of these finite trees. Furthermore, given a Δ^0_2 procedure to identify the nodes x_i, there is a Δ^0_2 procedure to identify the successors y of x_i for which $T[y]$ is a finitely occurring successor tree and to determine which equivalence class $T[y]$ belongs to.

There are infinitely many different isomorphism types among the trees $\{T[x_i] : i \in \omega\}$. Since they all have the same infinite-occurring isomorphism types, we may fix an equivalence class $\mathcal{E}$ such that $\{T[y] : y \in T \wedge y \in \mathcal{E}\}$ contains infinitely many different isomorphism types. (Here we are interpreting $\mathcal{E}$ as a class including successor nodes from each of the trees $T[x_i]$. By the comments above, we have a Δ^0_2 procedure to identify $y \in \mathcal{E}$.) Moreover, every $T[y]$ with $y \in \mathcal{E}$ is of height at most $n-2$, since $\text{level}_T(y) = 2$. Our construction of T' therefore uses induction on height, for which we regard the height 4 case given above as the base case. We identify elements of $\mathcal{E}$ above the various x_i and do the same construction on them that we did for a tree T with $\text{ht}(T) \leq n-1$.

To be more specific, suppose that $\mathcal{E} = \{z_i | i \in \omega\}$. We begin the proof of Lemma 3.19 again, using the z_i elements in place of the x_i elements. The only change in the hypothesis of the lemma is that each z_i is at level 2 rather than at level 1. However, this change does not affect the argument at all. That is, the fact that $T[z_i] \equiv T[z_j]$ implies that each $T[z_i]$ has the same infinitely occurring successor trees. We denote the isomorphism types of these trees by $J_1, \ldots, J_b$. If there are infinitely many i for which $T[z_i]$ has a finitely occurring successor tree which embeds in one of $J_1, \ldots, J_b$, then we may assume without loss of generality that there are infinitely many i which work with J_1. We run exactly the same argument as in Case 1, looking for appropriate successors of the z_i with which to diagonalize. Otherwise, if there are not infinitely many such i, we are back in Case 2 and we repeat this process over again with nodes at level 3. Since T has finite height, this process must stop. This completes the case of a height n tree, and also completes the proof of Lemma 3.19. ∎

Proof of Proposition 3.1. Let T be as in the statement of Proposition 3.1. Let r be the root of T, and let $x_0, x_1, \ldots$ be the immediate successors of r in T. Then every node above r in T is of finite type. Since r is not of finite type, there must be infinitely many of these successor trees. We consider the three ways in which r could fail to be of finite type as in Definition 1.7.

First, suppose there is an isomorphism type I which occurs infinitely often as a successor tree of r and which does not have strongly finite type. We split into two cases. If there are only finitely many isomorphism types of successor trees of r which embed into I, then Lemma 3.10 shows that T is not computably categorical. If there are infinitely many isomorphism types of successor trees of r which embed into I, then Lemma 3.16 implies that T is not computably categorical. Therefore, we can assume that any isomorphism type which occurs infinitely as a successor tree of r has strongly finite type.

Second, suppose there exist distinct isomorphism types I_0 and I_1 such that each occurs infinitely often as a successor tree to r and $I_0 \hookrightarrow I_1$. Since we can assume I_0 and I_1 have strongly finite type and they are not isomorphic, we must have $I_1 \not\hookrightarrow I_0$ by Lemma 2.10. We can now apply Lemma 3.15. Let the indices i be those for which $T[x_i] \cong I_0$ and let the indices j be those for which $T[x_j] \cong I_1$. There are infinitely many such indices i and j, $T[x_i] \hookrightarrow I_0$, $I_0 \hookrightarrow T[x_j]$ and $T[x_j] \not\hookrightarrow I_0$. Therefore, Lemma 3.15 proves that T is not computably categorical. We can now assume that there is no embedding between isomorphism types which occur infinitely often as successor trees of r. By Lemma 3.5, we know that there can only be finitely many isomorphism types which occur infinitely often as successor trees of r.

Finally, let $\mathcal{T}$ be the set of isomorphism types which occur among the successor trees of r. It could be that $\mathcal{T}$ is infinite. We split into two cases. If there is an infinitely occurring isomorphism type I for which $I' \hookrightarrow I$ for infinitely many $I' \in \mathcal{T}$, then we can apply Lemma 3.16. Otherwise, the finitely many isomorphism types which occur infinitely often each have only finitely many isomorphism types from $\mathcal{T}$ which embed into them. This situation is exactly the hypothesis for Lemma 3.18. Thus we have proved Proposition 3.1. ∎

In the cases above in which we constructed a Δ^0_2 isomorphism f between T and T', the computable dimension of T must be ω by Goncharov [10]. However, we can see this more directly (and prove it in the remaining cases) simply be rewriting the positive requirements:

$$\mathcal{R}_{\langle e,i\rangle} : \quad \varphi_e \text{ one-one and total } \implies [(\exists w_e \in T_i) T_i[w_e] \not\cong T'[\varphi_e(w_e)]].$$

Here $\{T_i\}$ is assumed to be a finite sequence of computable trees isomorphic to T, and the T' which we construct to satisfy these requirements will be of a different computable isomorphism class from each of them. In the original construction, T actually served a dual purpose, as both the template for T' and the computable isomorphism type to be avoided. Here we always use T_0 as the template, but diagonalize simultaneously against all the T_i.

4 Induction

In this section, we prove the second half of Theorem 1.8, that trees which are not of finite type cannot be computably categorical, and indeed must have computable dimension ω. Section 2 established the converse of this statement, and Section 3 enables us to use induction to prove the following proposition.

Proposition 4.1 *Let T be a tree of finite height but not of finite type. Then T is not computably categorical.*

Proof. The proof uses induction on the height n of T. The base case $n = 2$ is trivial, since every tree of height ≤ 2 is of finite type. Let r be the root of T, with immediate successors $x_0, x_1, \ldots$. If every node x_i is of finite type, then Proposition 3.1 shows that T has infinite computable dimension. So we may suppose that some isomorphism type I_0 appearing above r is not of finite type. (Without loss of generality we assume that $T[x_0] \cong I_0$.) By the inductive hypothesis, I_0 must not be computably categorical, so there is a computable tree U which is isomorphic to $T[x_0]$ but not computably isomorphic to it, and we may take the domain of U to be the computable set $T[x_0]$. Let V be identical to T, only with U in place of $T[x_0]$. Then V is computable and isomorphic to T.

Now we assume for a contradiction that T *is* computably categorical. Then there must exist a computable isomorphism φ from V to T, which must map U to some other successor tree $T[x_j]$ computably isomorphic to U. (Hence $j \neq 0$.) Moreover, φ would then have to map $T[x_j]$ (which is also a successor tree in V) to yet another successor tree $T[x_k]$ computably isomorphic to U, and so on. Therefore, the isomorphism type I_0 must appear infinitely often above r in T.

Moreover, since I_0 appears infinitely often above r, we can build another computable tree isomorphic to T, simply by adding any computable copy of I_0 as a new successor tree above r. Since this copy can be of any computable isomorphism type for I_0, the same argument as above shows that every computable isomorphism type of I_0 must appear infinitely often as a successor tree above r. Indeed, under the assumption that T is computably categorical, we see that for each such computable isomorphism type this process would yield an infinite c.e. set of roots of successor trees of that computable isomorphism type.

Since I_0 is not computably categorical, we have at least two of these c.e. sets, say C_1 and C_2. The idea is to use elements of $C_1 = \{w_0, w_1, \ldots\}$ as witness elements when we build T'. We wait until $\varphi_{e,s}(w_e)$ converges, and then redefine the Δ^0_2-isomorphism $f : T \to T'$ so that from stage s on, $T'[\varphi_e(w_e)]$ is built computably isomorphic to $T[y_e]$, where $C_2 = \{y_0, y_1 \ldots\}$. (Namely, define $f_{s+1}(f_s^{-1}(\varphi_e(w_e))) = y_e$.) The difficulty is that at the stage s at which $\varphi_e(w_e)$ converges, we do not know if $T'_s[\varphi_e(w_e)]$ embeds into I_0 or not, since $f_s^{-1}(\varphi_e(w_e))$ may or may not lie in a successor tree in T isomorphic to I_0. To handle this difficulty, we appeal to a corollary of Kruskal's Theorem.

Corollary 4.2 *Let $\{S_i : i \in \omega\}$ be an infinite set of finite trees. Then there exists $m \in \omega$ such that for every j there is an $i \leq m$ such that $S_i \hookrightarrow S_j$.*

Proof. If the set $\{S_j : (\forall i < j)[S_i \not\hookrightarrow S_j]\}$ were infinite, it would contradict Kruskal's Theorem. Hence we may take m to be the greatest index in this set. The corollary follows by an easy induction on the indices $> m$. ∎

Let J be the set of all finite trees S which do not embed into I_0. Then Corollary 4.2 yields a finite subset $\mathcal{S} \subseteq J$ such that for every $S \in J$ there is some $S' \in \mathcal{S}$ such that $S' \hookrightarrow S$. Moreover, no $S' \in \mathcal{S}$ embeds into I_0.

The witness elements for our construction will be the nodes w_e described above. Since the set C_1 is infinite and computably enumerable, we need not use Δ_2^0 guessing, either for them or for the corresponding nodes $y_e \in C_2$. (Technically, we will use Δ_2^0-guessing, but with a simple method of renaming the elements of C_1 and C_2.)

A requirement $\mathcal{R}_e$ requires attention at stage s if s is the least stage such that $\varphi_{e,s}(w_e)$ converges to some $w'_e \in T'_s$. At each stage s, we simply extend f_s to f_{s+1} mapping T_{s+1} to T'_{s+1} by adding fresh elements to T'_{s+1} as needed, except on those successor trees $T_s[w_e]$ such that $\mathcal{R}_e$ requires attention. For those e, we search for the least $t \geq s$ such that one of the following holds:

1. $\text{level}_{T_t}(f_s^{-1}(w'_e)) \neq 1$; or
2. some $S' \in \mathcal{S}$ embeds into $T_t[f_s^{-1}(w'_e)]$; or
3. $T'_s[w'_e]$ embeds into $T_t[y_e]$.

If either (1) or (2) holds, then again we simply extend f_s to f_{s+1} on $T_{s+1}[w_e]$ by adding fresh elements to T'_{s+1}, without redefining f_{s+1} on any nodes. However, if (3) holds, then we may need to redefine f.

The idea of the redefinition of f is as follows. Let $a = f_s^{-1}(w'_e)$. Currently, using f_s, $T'[w'_e]$ is being built by copying $T[a]$ and $T'[f_s(y_e)]$ is being built by copying $T[y_e]$. We use the embedding in (3) to define f_{s+1} so that $T'[w'_e]$ begins copying $T[y_e]$ and $T'[f_s(y_e)]$ begins copying $T[a]$. This successfully diagonalizes because $T[w_e]$ and $T[y_e]$ are not computably isomorphic. Before performing this switch, we need to check that no higher priority requirements will be injured. We ask first whether w'_e lies in the set

$$P_{e,s} = \{f_s(y_0), \ldots f_s(y_{e-1})\} \cup \{w'_i : i < e \ \&\ \varphi_{i,s}(w_i)\downarrow = w'_i\}.$$

If it does, then we cannot redefine f without possibly injuring some $\mathcal{R}_i$ of higher priority, so instead we eliminate w_e from our enumeration of C_1 and pick w_{s+1} to be the witness node for $\mathcal{R}_e$. (At future stages, we will refer to this node as w_e.) However, if this elimination has already happened $2e$ times for different values of w_e at previous stages, and the elements of $P_{e,s}$ have not changed since those stages, then we need not perform the elimination again (nor redefine f at all), since in this case φ_e must map all $2e+1$ of those different values of w_e into $P_{e,s}$ and hence cannot be one-to-one.

If $w'_e \notin P_{e,s}$, then we may proceed without injuring any higher-priority requirement. (If $w'_e = f_u(y_j)$ for some $j > e$ at some later stage u, we will simply ignore that y_j and renumber

C_2 with the element y_{j+1} as y_j instead, thereby possibly injuring a lower-priority requirement once.) Let g be the embedding of $T'_s[w'_e]$ into $T_t[y_e]$. (We may assume $g(w'_e) = y_e$.) Define $f_{s+1}(x) = g^{-1}(x)$ for all x in the image of g, and add fresh elements to T'_{s+1} to be the range of all of $T[y_e]$ under f_{s+1}. For all sufficiently large elements $x \in T[y_e]$, we may take $f_{s+1}(x) = x$. Thus f_{s+1} now maps $T[y_e]$ to $T'[w'_e]$.

We also must redefine f_{s+1} on the set $A = f_s^{-1}(T'_s[w'_e])$. Now since f_s is an isomorphism from T_s to T'_s, $A \cong T'_s[w'_e]$ must embed into I_0 via a lifting of the same g, so we may find an embedding h of A into $T[y_e]$. We add enough elements of $T[y_e]$ to T_{s+1} so that $A \hookrightarrow T_{s+1}[y_e]$, then combine this embedding with $f_s : T_s[y_e] \hookrightarrow T'_s[f_s(y_e)]$, adding fresh elements to T_{s+1} as needed, to define f_{s+1} on A. Thus f_{s+1} maps $T[f_s^{-1}(w'_e)]$ to $T'[f_s(y_e)]$. This completes the construction.

With the redefinition, we see that now $T[y_e]$ is not only isomorphic to $T'[w'_e]$, but actually computably isomorphic to it via f, since f is the identity map on cofinitely much of $T[y_e]$. If φ_e were an isomorphism from T to T', then $f^{-1} \circ \varphi_e$ would be a computable isomorphism from $T[w_e]$ onto $T[y_e]$, which is impossible, by our choice of w_e and y_e. Hence $\mathcal{R}_e$ is satisfied.

Moreover, if either (1) or (2) holds for w'_e, then $\mathcal{R}_e$ must again be satisfied. This is clear for (1), since $\text{level}_T(w_e) = 1$. If (2) holds, then some $S \in \mathcal{S}$ embeds into $T'[w'_e]$, but not into $T[w_e]$ (by our choice of $\mathcal{S}$). Hence clearly $\mathcal{R}_e$ is satisfied.

We must show that when we search for a stage t in the construction, we do eventually find one. Suppose $\text{level}_{T'}(w'_e) = 1$, and suppose that no $S \in \mathcal{S}$ embeds into $T[f_s^{-1}(w'_e)]$. By our choice of $\mathcal{S}$, this guarantees that every finite subtree of $T[f_s^{-1}(w'_e)]$ embeds into I_0. But $T'_s[w'_e]$ is isomorphic to $T_s[f_s^{-1}(w'_e)]$, hence must embed into I_0. Since $T[y_e] \cong I_0$, we will eventually find a stage t and an embedding satisfying (3).

The redefinition process does no injury to any other $\mathcal{R}_e$. The only possibility for injury among the requirements occurs when elements of C_1 or C_2 must be ignored or renamed, as described above, and when this happens each requirement respects the higher-priority requirements, so ultimately each $\mathcal{R}_e$ is satisfied.

Moreover, redefinition of f can only occur finitely often on any $T[x]$, and redefinition of f^{-1} can only occur finitely often on any $T'[x']$, since each requirement is injured only finitely often. (Our care in making $f_s(w_e) \notin P_{e,s}$ ensured this for f^{-1}.) Hence f is a bijection between T and T'. Since our redefinitions always respected the partial order we were building on T', T' is computable and f is an isomorphism. But since each $\mathcal{R}_e$ holds, there is no computable isomorphism from T to T', contradicting our assumption that T was computably categorical. ■

Corollary 4.3 *Every computable tree T of finite height but not of finite type must have infinite computable dimension.*

Proof. Because we proved Proposition 4.1 by contradiction, we do not know if there are two computable copies of T which are Δ^0_2-isomorphic but not computably isomorphic. Therefore, we cannot apply Goncharov's result that a pair of computable structures which are Δ^0_2-isomorphic but not computably isomorphic must have computable dimension ω. Instead, the proof proceeds by induction on the height of T. Assume T is a tree of finite height which does not have finite type. If every successor tree in T has finite type, then we are done by the results in Section 3. Otherwise, we fix $y \in T$ at level 1 such that $T[y]$ does not have finite type. By the induction hypothesis, $T[y]$ must have infinite computable dimension.

For a contradiction, assume that T has finite computable dimension m. Fix representatives $T^0, \ldots, T^{m-1}$ of the computable isomorphism classes of T and fix nodes $y^i \in T^i$ at level 1 such that $T[y] \cong T^i[y^i]$. To run a diagonalization argument as above, we need to find appropriate c.e. sets C^i in T^i and C in T. We define these sets and specify their exact properties below.

First, we define C^i for a fixed $i < m$. Let x^i_e denote the nodes at level 1 in T^i and assume that $y^i = x^i_0$. Let $U_0, \ldots, U_{m-1}$ be computable copies of $T^i[x^i_0]$ defined on the same numbers as $T^i[x^i_0]$ which are pairwise not computably isomorphic and are not computably isomorphic to $T^i[x^i_0]$. Let T^i_j (for $j < m$) be the computable tree formed by taking T^i and replacing $T^i[x^i_0]$ by U_j. Since the computable dimension of T^i is m, one of the T^i_j trees must be computably isomorphic to T^i. Without loss of generality, assume it is T^i_0 and fix an isomorphism $f : T^i_0 \to T^i$. f must send U_0 to some successor tree $T^i[x^i_{j_0}]$ with $x^i_{j_0} \neq x^i_0$. Thus, $T^i[x^i_{j_0}]$ is a successor tree in T^i_0 and f must send this tree to some $T^i[x^i_{j_1}]$. Repeating this process, we get a c.e. set of nodes $x^i_{j_k}$ for successor trees in T^i which are computably isomorphic to U_0. We denote these nodes by w^i_e and we let C^i be the c.e. set of these nodes.

Second, we define C. Let x_e, for $e \in \omega$, denote the nodes at level 1 in T and assume that $y = x_0$. Let $V_0, \ldots, V_{m-1}$ be computable copies of $T[x_0]$ which are not computably isomorphic to any of the trees $U_0, \ldots, U_{m-1}$ used in the definition of C^i for any i. (It does not matter if we reuse computable isomorphism types when defining C^i and C^j for $i \neq j$, but we need to have different computable isomorphism types when we define C. There are enough computable isomorphism types to accomplish these requirements because $T[x_0]$ has infinite computable dimension.) Let T_j (for $j < m$) be the computable tree formed by taking T and replacing $T[x_0]$ by V_j. By the same argument as in the last paragraph, we obtain a c.e. set C of nodes u_e at level 1 in T such that $T[u_e]$ is computably isomorphic to (without loss of generality) V_0. We can sum up the important properties of these c.e. sets by:

- $T^i[w^i_e] \cong T[y]$ for $e \in \omega$ and $i < m$;
- $T[u_e] \cong T[y]$ for $e \in \omega$;
- $T[u_k] \cong T^i[w^i_e]$ for $e, k \in \omega$ and $i < m$, but not by a computable isomorphism.

We build $T' \cong T$ which is not computably isomorphic to any T^i by an argument very similar to the one given above. We build T' in stages together with a Δ^0_2-isomorphism $f : T \to T'$. We index the witnesses in C as $u_{\langle e,i \rangle}$ with $e \in \omega$ and $i < m$ and we use the nodes w^i_e to diagonalize against φ_e being an isomorphism from T^i to T'. The strategy to defeat φ_e and T^i is to wait for $\varphi_e(w^i_e)$ to converge to some $v^i_e \in T'$ at stage s. Let $a = f^{-1}_s(v^i_e)$. We have that f_s maps $T_s[a]$ to $T'_s[v^i_e]$ and maps $T_s[u_{\langle e,i \rangle}]$ to $T'_s[f_s(u_{\langle e,i \rangle})]$. As above, we either find evidence that we have an easy win or else we find an embedding of $T'_s[v^i_e]$ into $T[u_{\langle e,i \rangle}]$. In the latter case, we use this embedding to define f_{s+1} so that it swaps the action of f_s on the successor trees, by making $T'[v_e^i]$ start to copy $T[u_{\langle e,i\rangle}]$ and making $T'[f_s(u_{\langle e,i\rangle})]$ start to copy $T[a]$. As above, this successfully diagonalizes since we know that $T^i[w_e^i]$ is not computably isomorphic to $T[u_{\langle e,i\rangle}]$. The formal details of this argument are essentially as above. ■

We note that for trees T in which nodes at levels ≥ 1 are not of finite type, these proofs only establish the existence of infinitely many computable isomorphism classes of copies of T, without giving us any actual idea how to construct copies in such classes. To construct copies in new classes would require a direct proof in the style of the Lemmas of Section 3, instead of the less-edifying proofs by contradiction in Proposition 4.1 and Corollary 4.3.

5 Δ_n^0-categoricity

The goal of this section is to prove the following theorem.

Theorem 5.1 *For each $n \geq 1$, there is a computable finite height tree T such that T is Δ_{n+1}^0-categorical but not Δ_n^0-categorical.*

We actually prove a slightly stronger statement by considering a more restrictive definition of trees. In this section, we define a tree to be a set $T \subseteq \omega^{<\omega}$ which is closed under initial segments. Such trees are obviously trees in the earlier sense, but they have the additional feature that the successor relation is computable. Therefore, we really establish Theorem 5.1 for computable finite height trees which have a computable successor relation.

Proof. The strategy for this proof is to show by induction on $n \geq 1$ that for any infinite and coinfinite Σ_n^0 (if n is odd) or Π_n^0 (if n is even) relation $P(x)$, there are computable trees T_P and S_P such that T_P and S_P are Δ_{n+1}^0-isomorphic but any isomorphism between them computes $P(x)$. For the purposes of presenting a general outline, suppose n is odd, so T_P needs to code a Σ_n^0 relation $P(x)$.

T_P will be ω-branching at the root, with the property that for any node τ at level 1, $T_P[\tau]$ has one of two distinct isomorphism types. Trees of one type are called coded Σ_n^0 trees and trees of the other type are called uncoded Σ_n^0 trees. There will be a computable sequence of nodes $\tau_x \in T_P$ such that each τ_x is at level 1 and

$$P(x) \Leftrightarrow T_P[\tau_x] \text{ is a coded } \Sigma_n^0 \text{ tree.}$$

We will be able to say exactly what the isomorphism type of T_P is. It ω-branches at the root and has infinitely many nodes τ at level 1 for which $T_P[\tau]$ is a coded Σ_n^0 tree and infinitely many nodes at level 1 for which $T_P[\tau]$ is an uncoded Σ_n^0 tree. This description of the isomorphism type of T_P will allow us to build a computable tree S_P which is isomorphic to T_P, but for which we know exactly which nodes at level 1 correspond to coded Σ_n^0 trees and which do not. Therefore, we will be able to compute $P(x)$ from any isomorphism between S_P and T_P.

To fill in the details of this outline, we first show how to code a Σ^0_1 relation. Next, we show how to pass from the coding of a Σ^0_1 relation to a coding for a Π^0_2 relation, and how to pass from the coding of a Π^0_2 relation to the coding of a Σ^0_3 relation. Finally, we outline the general procedure for coding a Σ^0_{n+1} relation from the coding of a Π^0_n relation, and the procedure for coding a Π^0_{n+1} relation from the coding of a Σ^0_n relation.

First, we show how to code an infinite and coinfinite Σ^0_1 relation $P(x)$. Assume that

$$\forall x(P(x) \Leftrightarrow \exists d R(x,d))$$

where R is computable. Let $T_P \subseteq \omega^{<\omega}$ be the computable set given by the closure of the following conditions under initial segments.

1. For all n, m, $\langle n, m\rangle \in T_P$.
2. For all n, m, $\langle n, m, 0\rangle \in T_P$ if and only if m is the least number such that $R(n,m)$ holds.

T_P has height 4 and is ω-branching at the root. If $P(n)$ holds, then $T_P[\langle n\rangle]$ is a tree of height 3 which ω-branches at the root and has a unique node at level 2 (in the restricted tree). We refer to any tree with this isomorphism type as a coded Σ^0_1 tree. If $P(n)$ does not hold, then $T_P[\langle n\rangle]$ is ω-branching at the root and has no nodes at level 2. We refer to any tree with this isomorphism type as an uncoded Σ^0_1 tree. Notice that the coded and uncoded Σ^0_1 trees are not isomorphic.

To give a slightly more general perspective, we need to distinguish coding a Σ^0_1 relation and coding a Σ^0_1 sentence. To code the Σ^0_1 relation P, we build a tree T_P which is ω-branching and for each $\langle n\rangle \in T_P$, we let $T_P[\langle n\rangle]$ code the Σ^0_1 sentence $P(n)$. That is, we effectively generate a tree $T_P[\langle n\rangle]$ which is a coded Σ^0_1 tree if the Σ^0_1 sentence $P(n)$ is true and is an uncoded Σ^0_1 tree if the Σ^0_1 sentence $P(n)$ is false.

The isomorphism type of T_P is uniquely determined by the following facts: the root of T_P is ω-branching, and for every $\tau \in T$ at level 1, $T_P[\tau]$ is either a coded Σ^0_1 tree or an uncoded Σ^0_1 tree, with infinitely many of each type. Furthermore, T_P has the property that

$$P(n) \;\Leftrightarrow\; T_P[\langle n\rangle] \, \text{is a coded} \, \Sigma^0_1 \, \text{tree.}$$

To see that T_P is Δ^0_2-categorical, suppose that T is computable and isomorphic to T_P. For any $\tau \in T$ at level 1, $T[\tau]$ is a coded Σ^0_1 tree if and only if $\exists \sigma_0, \sigma_1(\tau \prec \sigma_0 \prec \sigma_1)$. $0'$ can determine which nodes $\tau \in T$ are at level 1 and can tell whether $T[\tau]$ is a coded or uncoded Σ^0_1 tree. If $T[\tau]$ is a coded Σ^0_1 tree, then $0'$ can determine the unique node at level 2 in $T[\tau]$. Of course, $0'$ can also determine this information in T_P, so we can easily build the Δ^0_2 isomorphism.

To see that T_P need not be Δ^0_1-categorical, assume $P(x)$ is noncomputable. Let $S_P \subseteq \omega^{<\omega}$ be the closure of the following conditions under initial segments.

1. For all n, m, $\langle n, m\rangle \in S_P$.
2. If n is even, then $\langle n, 0, 0\rangle \in S_P$.

S_P is a computable tree and by the description of the isomorphism type of T_P, $S_P \cong T_P$. In addition,

$$S_P[\langle n\rangle] \text{ is a coded } \Sigma^0_1 \text{ tree } \Leftrightarrow n \text{ is even.}$$

For any isomorphism $f : T_P \to S_P$, $P(n)$ holds if and only if $f(\langle n\rangle) = \langle m\rangle$ for some even number m. Therefore, the fact that P is noncomputable implies that f cannot be Δ^0_1.

We turn to coding an infinite and coinfinite Π^0_2 relation $P(x)$ such that

$$P(x) \Leftrightarrow \forall c \exists d R(x, c, d),$$

where R is computable. Let $T_P \subseteq \omega^{<\omega}$ be the closure of the following conditions under initial segments. (We give both an informal and a formal description of these conditions. Later, we will trust the reader to fill in the formal descriptions.)

1. T_P is ω-branching at the root. Formally, let $\langle 1\rangle \in T_P$ and $\langle p^n\rangle \in T_P$ for all primes p and all $n \geq 1$. To clarify the coding below, view $\langle 1\rangle$ as $\langle 2^0\rangle$.

2. For all n and m, $T_P[\langle 2^n, m\rangle]$ is the tree defined above for coding the Σ^0_1 sentence $\forall c \leq m \exists d R(n, c, d)$. Formally, for all i, $\langle 2^n, m, i\rangle \in T_P$ and $\langle 2^n, m, i, 0\rangle \in T_P$ if and only if i is the least number such that $\forall c \leq m \exists d \leq i R(n, c, d)$.

3. For all odd primes p and all $n \geq 1$, $T_P[\langle p^n\rangle]$ consists of $n - 1$ copies of the coded Σ^0_1 tree and infinitely many copies of the uncoded Σ^0_1 tree. Formally, for all m and i, $\langle p^n, m, i\rangle \in T_P$, and $\langle p^n, m, i, 0\rangle \in T_P$ if and only if $i < n - 1$.

If $P(n)$ holds, then $T_P[\langle 2^n\rangle]$ is ω-branching at the root, and every node at level 1 (in this restricted tree) is the base of a coded Σ^0_1 tree. We call any tree with this isomorphism type a coded Π^0_2 tree.

For an odd prime p and $n \geq 1$, the tree $T_P[\langle p^n\rangle]$ is ω-branching at the root and at level 1 has exactly $n - 1$ many coded Σ^0_1 trees and infinitely many uncoded Σ^0_1 trees. We call any tree with this isomorphism type an $(n - 1)$-uncoded Π^0_2 tree. Notice that if $P(n)$ does not hold, then $T_P[\langle 2^n\rangle]$ is an m-uncoded Π^0_2 tree for some m.

Just as in the Σ^0_1 case, we have given a procedure for effectively constructing a tree $T_P[\langle 2^n\rangle]$ from the Π^0_2 sentence $P(n)$. This tree is a coded Π^0_2 tree if the Π^0_2 sentence $P(n)$ holds and it is an m-uncoded Π^0_2 tree (for some m) if the Π^0_2 sentence $P(n)$ is false. The other trees of the form $T_P[\langle p^n\rangle]$ are added so that the isomorphism type of T_P will be independent of the choice of P, as long as P is infinite.

We can now describe the isomorphism type of T_P precisely. T_P is ω-branching at the root and consists of infinitely many coded Π^0_2 trees and infinitely many m-uncoded Π^0_2 trees for each m. Furthermore, we have

$$P(n) \Leftrightarrow T_P[\langle 2^n\rangle] \text{ is a coded } \Pi^0_2 \text{ tree.}$$

To see that T_P is Δ^0_3-categorical, fix a computable tree T which is isomorphic to T_P. For any $\tau \in T$ at level 1, $T[\tau]$ is a coded Π^0_2 tree if and only if

$$\forall \sigma \big((\tau \prec \sigma \wedge \neg \exists \delta (\tau \prec \delta \prec \sigma)) \to T[\sigma] \text{ is a coded } \Sigma^0_1 \text{ tree} \big).$$

Since the property of being a Σ^0_1 tree can be expressed in a Σ^0_1 manner, this predicate is Π^0_2. Similarly, the predicate which says $T[\tau]$ is an n-uncoded Π^0_2 tree (for a fixed value of n) is Σ^0_2. Formally, $T[\tau]$ is an n-uncoded Π^0_2 tree if and only if there are disjoint $\tau_0, \dots, \tau_{n-1}$ such that $\tau \prec \tau_i$ and $T[\tau_i]$ is a coded Σ^0_1 tree and for all disjoint $\tau_0, \dots, \tau_n$ such that $\tau \prec \tau_i$, at least one τ_i is not the root of a coded Σ^0_1 tree. Since expressing $T[\sigma]$ is a coded Σ^0_1 tree is a Σ^0_1 statement, this entire expression is the conjunction of a Σ^0_1 and a Π^0_1 statement, and hence is Σ^0_2.

Assume that T is a computable tree which is isomorphic to T_P. By the comments above, $0''$ can determine which nodes at level 1 in T_P and T are the base of coded Π^0_2 trees and which are the base of n-uncoded Π^0_2 trees. Once we match these nodes up, we can use $0'$ to build the isomorphism above level 1, since we are essentially back in the case of Σ^0_1 trees.

Because we can describe the isomorphism type of T_P precisely, we can build a computable tree $S_P \subseteq \omega^{<\omega}$ which is isomorphic to T_P and for which

$$A = \{x | S_P[\langle x \rangle] \text{ is a coded } \Pi^0_2 \text{ tree} \}$$

is computable. To see that T_P need not be Δ^0_2-categorical, consider the case when $P(x)$ is Π^0_2-complete. If $f : T_P \to S_P$ is an isomorphism, then $P(x) \Leftrightarrow f(x) \in A$. Therefore, f cannot be Δ^0_2 without contradicting the Π^0_2-completeness of P.

The last example we consider before the general case is how to code the Σ^0_3 relation

$$P(x) \Leftrightarrow \exists b \forall c \exists d R(x, b, c, d).$$

We first code $P(x)$ into a computable tree $T_P \subseteq \omega^{<\omega}$ as follows.

1. T_P is ω-branching at the root. In this case, we let $\langle n \rangle \in T_P$ for all n.

2. For each n: $T_P[\langle n \rangle]$ is ω-branching at the root; for each m, $T_P[\langle n \rangle]$ has infinitely many nodes at level 1 each of which is the root of the tree for the Π^0_2 sentence $\forall c \exists d R(n, m, c, d)$; and for each m, $T_P[\langle n \rangle]$ has infinitely many nodes at level 1 each of which is the root of the m-uncoded Π^0_2 tree.

If $P(n)$ holds, then $T_P[\langle n \rangle]$ consists of infinitely many coded Π^0_2 trees as well as infinitely many m-uncoded Π^0_2 trees for each m. We refer to any tree with this isomorphism type as a coded Σ^0_3 tree. If $P(n)$ does not hold, then $T_P[\langle n \rangle]$ consists of infinitely many m-uncoded Π^0_2 trees for each m, and nothing else. We refer to any tree with this isomorphism type as an uncoded Σ^0_3 tree. As above, we are coding the Σ^0_3 sentences $P(n)$ by effectively constructing a tree $T_P[\langle n \rangle]$ which is a coded Σ^0_3 tree if the Σ^0_3 sentence $P(n)$ is true and is an uncoded Σ^0_3 tree if the Σ^0_3 sentence $P(n)$ is false.

We can describe the isomorphism type of T_P precisely as follows. T_P is ω-branching at the root and consists of infinitely many coded Σ^0_3 trees and infinitely many uncoded Σ^0_3 trees.

To see that T_P is Δ^0_4-categorical, let T be any computable tree which is isomorphic to T_P. For any $\tau \in T$ at level 1, $T[\tau]$ is a coded Σ^0_3 tree if and only if

$$\exists \tau_2 (\tau < \tau_2 \wedge \text{level}(\tau_2) = 2 \wedge T[\tau_2] \text{ is a coded } \Pi^0_2 \text{ tree}).$$

Since determining if $T[\tau_2]$ is a coded Π^0_2 tree is Π^0_2, this predicate is Σ^0_3. Therefore, $0'''$ can determine which nodes at level 1 in T and T_P are the base of a coded Σ^0_3 tree and which are the base of an uncoded Σ^0_3 tree. Once these nodes are matched up correctly, $0''$ can build the rest of the isomorphism as in the previous case. Therefore, T_P is Δ^0_4-categorical.

Since we can describe the isomorphism type of T_P exactly, we can build a computable tree $S_P \subseteq \omega^{<\omega}$ which is isomorphic to T_P and such that

$$A = \{n | S_P[\langle n \rangle] \text{ is a coded } \Sigma^0_3 \text{ tree}\}$$

is computable. As above, if $P(x)$ is Σ^0_3-complete, there cannot be a Δ^0_3 isomorphism between T_P and S_P.

We now present two general constructions. First, we use a construction similar to the Π^0_2 coding to pass from a Σ^0_n coding to a Π^0_{n+1} coding. Let $P(x)$ be an infinite and coinfinite Π^0_{n+1} relation such that

$$P(x) \Leftrightarrow \forall a R(x,a)$$

where R is Σ^0_n. Assume that we have defined the isomorphism types for a coded Σ^0_n tree and an uncoded Σ^0_n tree and that such trees are not isomorphic. Assume that these trees are defined in such a way that given any Σ^0_n sentence we can effectively construct a tree which is a coded Σ^0_n tree if the sentence is true and is an uncoded Σ^0_n tree if the sentence is false. Furthermore, assume that there is a Σ^0_n sentence which is true in the coded Σ^0_n tree and false in the uncoded Σ^0_n tree. We define T_P as the closure of the following conditions under initial segments.

1. T_P is ω-branching at the root. Formally, we put $\langle 1 \rangle \in T_P$ and $\langle p^u \rangle \in T_P$ for all primes p and $u \geq 1$.

2. For all u and m, $T_P[\langle 2^u, m \rangle]$ is constructed to code the Σ^0_n sentence $\forall a \leq m R(u,a)$.

3. For all odd primes p and $u \geq 1$, $T_P[\langle p^u \rangle]$ consists of $(u-1)$ copies of the coded Σ^0_n tree and infinitely many copies of the uncoded Σ^0_n tree.

If $P(u)$ holds, then the root of $T_P[\langle 2^u \rangle]$ is ω-branching and the nodes at level 1 in this tree are all roots of coded Σ^0_n trees. We refer to any tree with this isomorphism type as a coded Π^0_{n+1} tree.

If $P(u)$ does not hold, then the root of $T_P[\langle 2^u \rangle]$ is ω-branching, and the trees above the nodes at level 1 contain infinitely many copies of the uncoded Σ^0_n tree and for some m, exactly m copies of the coded Σ^0_n tree. We refer to any tree with this isomorphism type as an m-uncoded Π^0_{n+1} tree.

The isomorphism type of T_P is uniquely determined by the fact that the root is ω-branching and T_P consists of infinitely many copies of the coded Π^0_{n+1} tree and infinitely many copies of the m-uncoded Π^0_{n+1} tree for each m.

For any computable tree T isomorphic to T_P and any $\tau \in T$ at level 1, $T[\tau]$ is a coded Π^0_{n+1} tree if for all m, there are distinct nodes $\tau_0, \ldots, \tau_m$ such that for all $i < m$,

$$\tau < \tau_i \wedge \text{level}(\tau_i) = 2 \wedge T_P[\tau_i] \text{ is a coded } \Sigma^0_n \text{ tree.} \qquad (1)$$

This condition is Π^0_{n+1} by our assumption on the complexity of determining if a tree is a coded Σ^0_n tree. Furthermore, $T[\tau]$ is an m-uncoded Π^0_{n+1} tree if there exist distinct nodes $\tau_1, \dots, \tau_m$ such that for all $1 \leq i \leq m$, equation (1) holds (for $m = 0$ this check is vacuous), but for all choices of nodes $\tau_1, \dots, \tau_{m+1}$, there is some $1 \leq i \leq m+1$ such that

$$(\tau < \tau_i \wedge \text{level}(\tau_i) = 2) \rightarrow T_P[\tau_i] \text{ is an uncoded } \Sigma^0_n \text{ tree.}$$

Altogether, this condition is the conjunction of a Σ^0_n statement and a Π^0_n statement.

To see that T_P is Δ^0_{n+2}-categorical, notice that each of the conditions in the previous paragraph can be determined by $0^{(n+1)}$. Therefore, given any computable tree T isomorphic to T_P, $0^{(n+1)}$ can match up the nodes at level 1 correctly. The fact that the rest of the isomorphism can be built follows by induction.

To see that T_p is not Δ^0_{n+1}-categorical, consider the case when $P(x)$ is Π^0_{n+1}-complete. Because we know the isomorphism type of T_P exactly, we can build a computable tree S_P such that

$$A = \{m | S_P[\langle m \rangle] \text{ is a coded } \Pi^0_{n+1} \text{ tree} \}$$

is computable. If $f : T_P \rightarrow S_P$ is an isomorphism, then $P(x)$ holds if and only if $f(\langle 2^x \rangle) = \langle m \rangle$ for some $m \in A$. Therefore, f cannot be Δ^0_n without contradicting the fact that P is Π^0_{n+1}-complete.

It remains to show how to pass from a Π^0_n coding to a Σ^0_{n+1} coding. Let $P(x)$ be an infinite and coinfinite Σ^0_{n+1}-complete relation given by

$$P(x) \Leftrightarrow \exists a R(x, a)$$

where R is Π^0_n. Assume that we have determined the isomorphism types for the coded and m-uncoded Π^0_n trees with the corresponding complexity results as above. We define the computable tree T_P as the downward closure of the following conditions.

1. T_P is ω-branching at the root. Formally, for all u, $\langle u \rangle \in T_P$.

2. For each u: $T_P[\langle u \rangle]$ is ω-branching at the root; for each m, $T_P[\langle u \rangle]$ has infinitely many nodes at level 1 each of which is the root of tree coding the Π^0_n sentence $R(u, m)$; and for each m, $T_P[\langle u \rangle]$ has infinitely many nodes at level 1 each of which is the root of an m-uncoded Π^0_n tree.

If $P(u)$ holds, then $T_P[\langle u \rangle]$ is ω-branching at the root and contains infinitely many copies of the coded Π^0_n tree and infinitely many copies of the m-uncoded Π^0_n tree for each m. We refer to any tree with this isomorphism type as a coded Σ^0_{n+1} tree.

If $P(u)$ does not hold, then $T_P[\langle u \rangle]$ is ω-branching at the root and consists of infinitely many copies of the m-uncoded Π^0_n tree for each m. In particular, there are no coded Π^0_n trees in $T_P[\langle u \rangle]$. We refer to any tree with this isomorphism type as an uncoded Σ^0_{n+1} tree.

Let T be a computable tree isomorphic to T_P and suppose $\tau \in T$ is a level 1 node. $T[\tau]$ is a Σ^0_{n+1} coded tree if and only if there is a $\tau_2 \in T$ such that

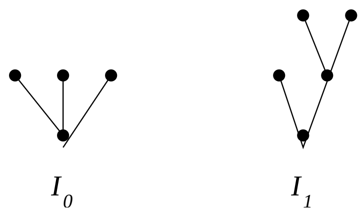
$$\tau < \tau_2 \wedge \text{level}(\tau_2) = 2 \wedge T[\tau_2] \text{ is a coded } \Pi^0_n \text{ tree.}$$

By assumption on the complexity of Π^0_n trees, this condition is Σ^0_{n+1}. As above, this condition implies that T_P is Δ^0_{n+2}-categorical. To show that T_P is not Δ^0_{n+1}-categorical, we define a tree S_P and argue as above.

6 Trees Under the Infimum Function

We end this paper with a brief discussion about trees defined using the infimum function and a conjecture about when they are computably categorical. We have already mentioned (see Section 1) that if $(T, \wedge)$ is computable, then the corresponding $(T, \prec)$ is also computable. However, it is simple to build a tree T in which $\prec$ is computable and $\wedge$ is not. Start with 0 as the root, and make every even number a successor of 0 at level 1. To diagonalize against the possibility that φ_e computes $\wedge$, we wait until $\varphi_e(\langle 4e+2, 4e+4\rangle)\downarrow= 0$, and if this ever happens, we add the next available odd number x to T at level 1 with $x \prec 4e+2$ and $x \prec 4e+4$. T will have domain ω and height 3, and $\prec$ will be computable, but our diagonalization ensures that $\wedge$ is not computable.

The notion of an embedding depends strongly on whether we define trees using $\prec$ or $\wedge$. Consider the following two trees:



It is easy to embed I_0 into I_1 with respect to $\prec$, but there is no embedding of I_0 into I_1 respecting $\wedge$. (The infimum of any pair of distinct nodes in I_0 is the root, whereas no possible image of I_0 in I_1 has the same property.) For the remainder of this section, therefore, we will speak of $\prec$-embeddings and $\wedge$-embeddings, to distinguish these two types of embeddings. In the rest of the paper, of course, "embedding" always means $\prec$-embedding. Notice that for an isomorphism, it does not matter which notion we use. That is, any isomorphism between trees under $\prec$ is also an isomorphism of the same trees under $\wedge$ (and conversely).

The simple example above creates problems when one investigates computable categoricity. Consider the tree T which consists of a root with infinitely many copies of I_0 and infinitely many copies of I_1 (and nothing else) as successor trees above the root. This tree has height 4, and $(T, \prec)$ is not of finite type, hence not computably categorical, by Theorem 1.8. However, $(T, \wedge)$ *is* computably categorical. Clearly we can build a computable copy of $(T, \wedge)$, and given any two computable copies, we can find the root of each, then identify successor trees of each type in each copy and match them up. In particular, every successor tree contains three pairwise-incomparable nodes a, b, and c, and once those nodes have appeared, we simply compute $a \wedge b$, $a \wedge c$, and $b \wedge c$. The successor tree is of type I_0 iff these three infima are equal.

We do have the following result.

Lemma 6.1 *Any computably categorical tree $(T, \prec)$ will still be computably categorical when re-interpreted as $(T, \wedge)$, assuming only that the function $\wedge$ so defined is computable.*

Proof. If a computable structure $(T', \wedge')$ is isomorphic to $(T, \wedge)$, then the corresponding $(T', \prec')$ is also computable, hence isomorphic to $(T, \prec)$ via some computable φ. As noted above, the isomorphism φ must also preserve infima, so it is an isomorphism of $(T, \wedge)$ onto $(T', \wedge')$ as required. ■

To determine computable categoricity for trees under the infimum function, therefore, we need to consider $\wedge$-embeddings rather than $\prec$-embeddings. Fortunately, one of our main tools, Kruskal's Theorem (stated as Theorem 3.4), yields not only $\prec$-embeddings but also $\wedge$-embeddings. The first step, therefore, is to refine the notion of being of finite type by referring to $\wedge$ instead of $\prec$.

Definition 6.2

1. A tree T is *of strongly finite $\wedge$-type* if it satisfies Definition 1.6 when the word "embedding" is replaced everywhere by "$\wedge$-embedding."
2. A tree T is *of finite $\wedge$-type* if it satisfies Definition 1.7 when the word "embedding" is replaced everywhere by "$\wedge$-embedding" and "strongly finite type" is replaced everywhere by "strongly finite $\wedge$-type."

Notice that in our example from the previous page, $(T, \wedge)$ has finite $\wedge$-type but $(T, \prec)$ does not have finite $\prec$-type.

We conjecture that with these definitions, the proofs from Sections 2, 3, and 4 will go through with relatively few modifications, as long as one always refers to $\wedge$-embeddings and (strongly) finite $\wedge$-type. Thus we would have the analogue of Theorem 1.8:

Conjecture 6.3 *For a computable tree $(T, \wedge)$ of finite height, the following are equivalent:*

1. *T is of finite $\wedge$-type;*
2. *$(T, \wedge)$ is computably categorical;*
3. *$(T, \wedge)$ has finite computable dimension;*
4. *$(T, \wedge)$ is relatively computably categorical.* ■

In [24], Miller proved the corresponding result for computable trees $(T, \wedge)$ of infinite height: the computable dimension of $(T, \wedge)$ must be ω. Together with Conjecture 6.3, this would answer the question of computable categoricity for all trees under the infimum function.

References

[1] C.J. Ash; Categoricity in hyperarithmetical degrees, *Annals of Pure and Applied Logic* **34** (1987), 1-14.

[2] C.J. Ash & J.F. Knight; *Computable Structures and the Hyperarithmetic Hierarchy* (Amsterdam: Elsivier Science 2000.)

[3] C. J. Ash, J. F. Knight, M. Mannasse, & T. Slaman; Generic copies of countable structures, *Annals of Pure and Applied Logic* **42** (1989), 195-205.

[4] J. Chisholm; On intrisically 1-computable trees, unpublished manuscript.

[5] J.N. Crossley, A.B. Manaster, & M.F. Moses; Recursive categoricity and recursive stability, *Annals of Pure and Applied Logic* **31** (1986), 191-204.

[6] R.G. Downey; On presentations of algebraic structures, in *Complexity, Logic, and Recursion Theory*, ed. A. Sorbi (New York: Dekker, 1997), 157-205.

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

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

[9] S.S. Goncharov; Autostable models and algorithmic dimensions, *Handbook of Recursive Mathematics*, vol. 1 (Amsterdam: Elsevier, 1998), 261-287.

[10] S.S. Goncharov; Nonequivalent constructivizations, *Proc. Math. Inst. Sib. Branch Acad. Sci.* (Novosibirsk: Nauka, 1982).

[11] S.S. Goncharov; The problem of the number of non-self-equivalent constructivizations, *Algebra and Logic* **19** (1980), 401-414 (English translation).

[12] S.S. Goncharov; Groups with a finite number of constructivizations, *Soviet Math. Dokl.* **19** (1981) 58-61.

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

[14] S.S. Goncharov, S. Lempp & R. Solomon; The computable dimension of ordered abelian groups, to appear in *Advances in Mathematics*.

[15] S.S. Goncharov, A.V. Molokov & N.S. Romanovskii; Nilpotent groups of finite algorithmic dimension, *Siberian Math Journal* **30** (1989), 63-68.

[16] D.R. Hirschfeldt, B. Khoussainov, R.A. Shore, & A.M. Slinko; Degree spectra and computable dimension in algebraic structures, *Annals of Pure and Applied Logic* **115** (2002), 71-113.

[17] B. Khoussainov & R.A. Shore; Computable isomorphisms, degree spectra of relations, and Scott families, *Annals of Pure and Applied Logic* **93** (1998), 153-193.

[18] B. Khoussainov & R.A. Shore; Effective model theory: the number of models and their complexity, *Models And Computability: Invited Papers from Logic Colloquium '97*, ed. S.B. Cooper & J.K. Truss, LMSLNS **259** (Cambridge: Cambridge University Press, 1999), 193-240.

[19] J.B. Kruskal; Well quasi-ordering, the tree theorem, and Vázsonyi's conjecture, *Transactions of the American Mathematical Society* **95** (1960), 210-225.

[20] 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).

[21] O.V. Kudinov; An integral domain with finite algorithmic dimension, unpublished manuscript.

[22] P. LaRoche, Recursively presented Boolean algebras, *Notices Amer. Math. Soc.* **24** (1977), A-552, research announcement.

[23] G. Metakides & A. Nerode; Effective content of field theory, *Ann. Math. Logic* **17** (1979), 289-320.

[24] R.G. Miller; The computable dimension of trees of infinite height, to appear in *Journal of Symbolic Logic.*

[25] R.G. Miller; The Δ_2^0 spectrum of a linear order, *Journal of Symbolic Logic* **66** (2001), 470-486.

[26] C.St.J.A. Nash-Williams; On well-quasi-ordering finite trees, *Proc. Cambridge Phil. Soc.* **59** (1963), 833-835.

[27] A.T. Nurtazin; Computable classes and algebraic criteria of autostability, thesis from Math. Inst. Siberian Branch of SSSR Acad. Sci., Novosibirsk, 1974 (Russian).

[28] A.T. Nurtazin; Strong and weak constructivizations and enumerable families, *Algebra and Logic* **13** (1974), 177-184.

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

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

[31] S.G. Simpson; Nonprovability of certain combinatorial properties of finite trees, *Harvey Friedman's Research on the Foundations of Mathematics*, ed. L. A. Harrington, M. D. Morley, A. Scedrov & S. G. Simpson (Amsterdam: North-Holland, 1985), 87-117.

[32] T.A. Slaman; Relative to any nonrecursive set, *Proceedings of the American Mathematical Society* **126** (1998), 2117-2122.

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

[34] S. Wehner; Enumerations, countable structures, and Turing degrees, *Proceedings of the American Mathematical Society* **126** (1998), 2131-2139.

[35] W. White, On the complexity of categoricity in computable structures, *Mathematical Logic Quarterly* **49** 6 (2003), 603-614.

Department of Mathematics
University of Wisconsin
480 Lincoln Drive
Madison, WI 53706-1388
E-mail: lempp@math.wisc.edu

E-mail: cmccoy1@nd.edu

Department of Mathematics
Queens College – C.U.N.Y.
65-30 Kissena Blvd.
Flushing, New York 11367
E-mail: rmiller@forbin.qc.edu

196 Auditorium Road
University of Connecticut U-3009
Department of Mathematics
Storrs, CT 06269-3009
E-mail: solomon@math.uconn.edu