A Structure Which Cannot Be Effectively Recovered from Its Back-And-Forth Tree

July 27, 2020

1 Introduction

Two structures $\mathcal{A}$ and $\mathcal{B}$ are back-and-forth equivalent if there is a relation $\sim$ on finite tuples from $\mathcal{A}$ and $\mathcal{B}$ such that $\varnothing \sim \varnothing$ and for every $\bar{a} \sim \bar{b}$:

- $\bar{a}$ and $\bar{b}$ have the same atomic type,
- for all a', there is b' such that $\bar{a}a' \sim \bar{b}b'$,
- for all ab', there is a' such that $\bar{a}a' \sim \bar{b}b'$,

For simplicity, assume for now that the language is finite. It is well-known that two countable structures which are back-and-forth equivalent are actually isomorphic.

One can construct from a structure $\mathcal{A}$ a labeled tree which represents the back-and-forth type of that structure.

Definition 1.1. Fix an infinite set L of labels which we put in an effective bijection with atomic types. Define the back-and-forth tree $\mathcal{T}(\mathcal{A})$ of a structure $\mathcal{A}$ as follows: the nodes of $\mathcal{T}(\mathcal{A})$ are the finite tuples from $\mathcal{A}$; the extension relation is the extension of tuples; each tuple is labeled with (a code for) its atomic type.

One can view this labeled tree as a structure in the language of trees (with a parent-of function) together with a unary relation for each label. Two structures have isomorphic back-and-forth trees if and only if they are back-and-forth equivalent. So the back-and-forth tree of a countable structure encodes the isomorphism type of the structure:

$$\mathcal{A} \cong \mathcal{B} \iff \mathcal{T}(\mathcal{A}) \cong \mathcal{T}(\mathcal{B}).$$

Thus $\mathcal{T}$ witnesses that labeled trees are Borel complete. In fact, this is essentially the same argument that Friedman and Stanley [FS89] used to show that trees (without labels) are Borel complete; the only remaining step is to turn a labeled tree into a tree. In Section 4 we will show in general that every labeled tree is uniformly effectively bi-interpretable with a tree without labels.

From a copy of $\mathcal{A}$, one can compute $\mathcal{T}(\mathcal{A})$, and we know that $\mathcal{T}(\mathcal{A})$ determines the isomorphism type of $\mathcal{A}$. The main question of this paper is whether it is possible to compute a copy of $\mathcal{A}$ given a copy of $\mathcal{T}(\mathcal{A})$. That is, given knowledge about the back-and-forth type of a structure, can we compute a copy of the structure? We will show that the answer is no:

Theorem 1.2. *For any computable ordinal α, there is a structure $\mathcal{A}$ such that $\mathcal{T}(\mathcal{A})$ has a computable copy but $\mathcal{A}$ itself has no Δ^0_α copy.*

So in fact, one cannot recover a copy of $\mathcal{A}$ from $\mathcal{T}(\mathcal{A})$ even with any number of jumps. We will prove this theorem for $\alpha = 1$ in Section 6 and then obtain the full theorem as a corollary in Section 8. The construction is quite difficult and uses new techniques that have promise for further applications.

While answering a question that is interesting in its own right, this theorem also casts light on a number of other interesting questions in computable structure theory. As we will argue below, most constructions in computable structure theory of a structure with some property are of such a flavour that they transfer to the back-and-forth tree, and so the back-and-forth tree has that property as well. Theorem 1.2 says that computable structure theory is much deeper; there can be a complexity in the structure as a whole that cannot be seen from its back-and-forth type.

1.1 Coding information

We say that a structure $\mathcal{A}$ *codes* a set $X \subseteq \omega$ if every copy of $\mathcal{A}$ enumerates X. If $\mathcal{A}$ codes X, then in fact there is a tuple $\bar{a} \in \mathcal{A}$ such that the existential type of $\bar{a}$ can enumerate X [Kni86, Theorem 1.4] (see also [AK00, Theorem 10.17]). This information is also contained in the back-and-forth tree of $\mathcal{A}$: there is a node in $\mathcal{T}(\mathcal{A})$ representing the tuple $\bar{a}$, and the existential type of $\bar{a}$ can be enumerated by searching through extensions of that node. So $\mathcal{A}$ codes X if and only if $\mathcal{T}(\mathcal{A})$ codes X, that is, a structure and its back-and-forth tree code the same sets.

More generally, let $\mathcal{F}$ be a family of subsets of ω. We say that $\mathcal{A}$ *codes* $\mathcal{F}$ if every copy of $\mathcal{A}$ enumerates $\mathcal{F}$ though possibly in a different order. An enormous number of results in computable structure theory have been proved by coding a family of sets into the existential types realized by a structure. For example, Slaman [Sla98] and Wehner [Weh98] independently showed that there is a structure with no computable copy, but with a copy computable in all non-computable degrees; Wehner's construction explicitly builds a family of subsets of ω which can be enumerated exactly by the non-computable degrees and then codes these into the existential types of the structure, while in Slaman's construction such a family can easily be extracted from the existential types.

But once again, the back-and-forth tree can see the existential types realized in the structure, and so if the structure codes a family in such a way, then the back-and-forth tree does as well. As a corollary of Theorem 1.2, we see that there are structures which code families of sets without coding them into the existential types.

Corollary 1.3. *Let $\{\Theta_e : e \in \omega\}$ be the standard computable enumeration of all enumeration-operators. There is a computable structure $\mathcal{A}$ and a family $\mathcal{F}$ of subsets of ω which is enumerated by all copies of $\mathcal{A}$ for which there do not exist a tuple $\bar{a} \in A^{<\omega}$ and a uniformly computable list of Σ_1 formulas $\varphi_\ell(\bar{x}, \bar{y})$ such that*

$$\mathcal{F} = \{\Theta_\ell(\Sigma_1\text{-}tp_{\mathcal{A}}(\bar{a}, \bar{b})) : l \in \omega,\ \bar{b} \in A^{<\omega},\ \mathcal{A} \models \varphi_\ell(\bar{a}, \bar{b})\}.$$

More generally, one could code information into the Σ_α types of a structure for any α. But the back-and-forth tree has access to all of this as well, and so one can view Theorem

1.2 as saying that it is possible to code information into a structure without coding it via types. Instead, the information is coded into the difficulty of amalgamating types, which is a much less straightforward way of coding information.

See Section 7 for more discussion of enumerating families, and for the proof of the above corollary.

1.2 Universality and degree spectra of trees

Informally, we say that a class $\mathcal{C}$ of structures is universal if for any structure of any signature, there is a structure is $\mathcal{C}$ that has the same computability-theoretic properties such as the same degree spectrum, the same computable dimension, etc. More formally, we can ask that the two structures be effectively bi-interpretable, which implies that they have the same computability-theoretic properties. Many classes of structures—graphs, groups, rings, and fields—have been shown to be universal. Other classes of structures—linear orders, boolean algebras, torsion-free abelian groups, real closed fields—are known not to be universal. Usually in these cases, there is some specific computability-theoretic property that cannot be realized in the class. For example, if a boolean algebra has a low copy, then it has a computable copy, and so there is no boolean algebra with Slaman-Wehner degree spectrum.

The case of trees is different however. Trees are not universal in the sense that there are structures which are not bi-interpretable with any tree, but the issue is with the automorphism groups and not with any computability-theoretic property. We will discuss this in more detail in Section 2. In fact, we do not know of any purely computability-theoretic property that can be realized in some structure, but not in a tree. This is because trees can code sets, families, families of families, and so on, into their existential types, and so any example that is constructed in such a way—which, after looking through the literature and many conversations with other experts in the field, seems all of the known examples—can be replicated by a tree. So for example there is a tree with each possible computable dimension, there is a tree with Slaman-Wehner degree spectrum, and there is a tree which is computably categorical but not relatively computably categorical. We will discuss such examples in more detail in Section 3.

This is an undesirable situation, because what we really care about is the informal notion of universality: is it true that for every structure of any signature, there is a tree with the same computability-theoretic properties? The formal notion involving bi-interpretability is just an attempt to capture this informal notion. So when we show that a class of structures is not universal, we really want to exhibit some particular computability-theoretic property that cannot be realized in that class.

For trees, the best approach seems to be by looking at degree spectra.

Question 1.4. Is every degree spectrum the degree spectrum of a tree?

What we are looking for is a way of transforming a structure into a tree, and the back-and-forth tree construction is the obvious thing to try. In Section 4, we show that given a structure $\mathcal{A}$, there is a tree without labels that has the same degree spectrum (in fact is effectively bi-interpretable with) the back-and-forth tree $\mathcal{T}(\mathcal{A})$. So if $\mathcal{A}$ and $\mathcal{T}(\mathcal{A})$ had the same degree spectrum, then this question would have a positive answer.

But Theorem 1.2 says exactly the opposite, and so this attempt to answer the question fails. And as the back-and-forth tree seems like the best transformation of a structure into a tree that one could hope for, we conjecture that the answer is negative: there is a degree spectrum which is not the degree spectrum of a tree. Answering this question would tell use something new and very interesting about degree spectra.

1.3 Linear orders

Frideman and Stanley [FS89] also proved that linear orders are Borel complete. Recall that this means that for each fixed language, there is a Borel operator Φ that takes a structure $\mathcal{A}$ to a linear order $\Phi(\mathcal{A})$ such that

$$\mathcal{A} \cong \mathcal{B} \iff \Phi(\mathcal{A}) \cong \Phi(\mathcal{B}).$$

Their operator Φ is in fact computable and can be defined as the composition of the back-and-forth tree operator $\mathcal{T}$ and the following operator $\mathcal{L}$.

Definition 1.5. Let T be a labeled tree. Define $\mathcal{L}(T)$ recursively as follows: $\mathcal{L}(T)$ is the shuffle sum of $\mathbb{Q} + \ell(\sigma) + 2 + \mathbb{Q} + \mathcal{L}(T_\sigma)$ for all children σ of the root node, where T_σ is the subtree of T below σ and $\ell(\sigma)$ is the (integer code for) the label of σ.

So the Friedman-Stanley operator is $\Phi = \mathcal{L} \circ \mathcal{T}$. We conjecture that:

Conjecture 1.6. *For each computable ordinal α, there is a structure $\mathcal{A}$ such that $\Phi(\mathcal{A})$ is computable but $\mathcal{A}$ has no Δ^0_α copy.*

2 Trees and Universality Under Bi-Interpretability

In this section we will show that trees are not universal under effective bi-interpretability. Effective interpretations were first introduced by Montalbán [Mon13a] but they are essentially the same as the parameterless version of the well-studied notion of Σ-reducibility which was introduced by Ershov [Ers96]. The elementary first-order definitions of a model-theoretic interpretation are now replaced by effective Δ^{c}_1 definitions, and the interpretation is allowed to use tuples of arbitrary sizes.

Definition 2.1. An *effective interpretation* of $\mathcal{A} = (A, P_0^{\mathcal{A}}, P_1^{\mathcal{A}}, \ldots)$ in $\mathcal{B}$ consists of:

- a Δ^{c}_1-definable subset $Dom^{\mathcal{B}}_{\mathcal{A}} \subseteq \mathcal{B}^{<\omega}$,
- a Δ^{c}_1-definable equivalence relation $\sim$ on $Dom^{\mathcal{B}}_{\mathcal{A}}$,
- a sequence of uniformly Δ^{c}_1-definable sets $R_i \subseteq (Dom^{\mathcal{B}}_{\mathcal{A}})^k$, where k is the arity of P_i, which respect $\sim$,
- a surjective map $f^{\mathcal{B}}_{\mathcal{A}} \colon Dom^{\mathcal{B}}_{\mathcal{A}} \to A$ which induces an isomorphism

$$f^{\mathcal{B}}_{\mathcal{A}} \colon (Dom^{\mathcal{B}}_{\mathcal{A}}/\sim; R_0/\sim, R_1/\sim, \ldots) \to \mathcal{A}.$$

Two structures $\mathcal{A}$ and $\mathcal{B}$ are effectively bi-interpretable if they are each effectively interpretable in the other, and moreover, the composition of the interpretations—i.e., the isomorphisms which map $\mathcal{A}$ to the copy of $\mathcal{A}$ inside the copy of $\mathcal{B}$ inside $\mathcal{A}$, and $\mathcal{B}$ to the copy of $\mathcal{B}$ inside the copy of $\mathcal{A}$ inside $\mathcal{B}$—are definable.

Definition 2.2. Two structures $\mathcal{A}$ and $\mathcal{B}$ are *effectively bi-interpretable* if there are effective interpretations of $\mathcal{A}$ in $\mathcal{B}$ and of $\mathcal{B}$ in $\mathcal{A}$ such that the compositions

$$f^{\mathcal{A}}_{\mathcal{B}} \circ \tilde{f}^{\mathcal{B}}_{\mathcal{A}} \colon \mathrm{Dom}_{\mathcal{B}}^{(\mathrm{Dom}^{\mathcal{B}}_{\mathcal{A}})} \to \mathcal{B} \quad \text{and} \quad f^{\mathcal{B}}_{\mathcal{A}} \circ \tilde{f}^{\mathcal{A}}_{\mathcal{B}} \colon \mathrm{Dom}_{\mathcal{A}}^{(\mathrm{Dom}^{\mathcal{A}}_{\mathcal{B}})} \to \mathcal{A}$$

are $\Delta^{\mathfrak{c}}_1$-definable in $\mathcal{B}$ and $\mathcal{A}$ respectively.

Two structures which are effective bi-interpretable share essentially all the same computability-theoretic properties (see [Mon14, Lemma 5.3]). Moreover, two structures which are bi-interpretable have the same automorphism group. We can use this fact to show that there are structures which are not bi-interpretable with any tree, and so trees are not universal.

Proposition 2.3. *The linear order $(\mathbb{Z}, <)$ is not bi-interpretable with any tree.*

Proof. The automorphism group of $(\mathbb{Z}, <)$ is the additive group $\mathbb{Z}$. We will argue that $\mathbb{Z}$ is not the automorphism group of any tree; since structures which are bi-interpretable have the same automorphism groups, this shows that $(\mathbb{Z}, <)$ is not automorphic to any tree.

Suppose to the contrary that T is a tree with automorphism group $\mathbb{Z}$. Fix an automorphism f which generates this automorphism group. Let $\sigma \in T$ be such that $f(\sigma) \neq \sigma$, but such that f fixes the parent τ of σ. Let $\sigma' = f(\sigma)$. The subtree T_σ below σ must be isomorphic to the subtree $T_{\sigma'}$ below σ'. Then T has an automorphism g which maps σ to σ' and σ' to σ (and mapping T_σ to $T_{\sigma'}$ and vice versa using the isomorphism between these subtrees), but which fixes every other node. This map g has order 2, contradicting the assumption that T has automorphism group $\mathbb{Z}$. □

3 Trees and Enumerating Families

Even though there structures that are not bi-interpretable with any tree, in this section we will show how to construct trees realizing many different computability-theoretic properties by coding families of sets into trees.

We begin with a couple of standard definitions about enumerating families.

Definition 3.1. Let $\mathcal{F}$ be a countable family of subsets of ω. A set W is an enumeration of a family $\mathcal{F}$ if $\mathcal{F} = \{W[i] : i \in \omega\}$, where $W[i] = \{j \mid \langle i, j\rangle \in W\}$ is the ith column of W.

We say that X enumerates $\mathcal{F}$ if there is an enumeration W of $\mathcal{F}$ which is c.e. in X. For α a computable ordinal, we say that X Σ^0_α-enumerates $\mathcal{F}$ if there is an enumeration of $\mathcal{F}$ which is Σ^0_α in X.

Definition 3.2. Two enumerations U and W of a family $\mathcal{F}$ are computably equivalent if there is a computable permutation f of ω such that $W[f(i)] = U[i]$.

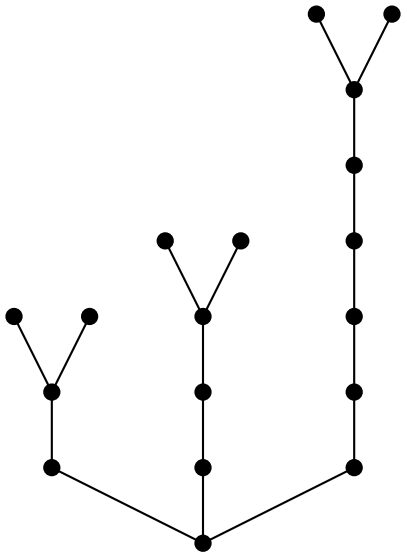
It is not hard to code a family $\mathcal{F}$ into a tree $T(\mathcal{F})$ so that every enumeration of $\mathcal{F}$ computes a copy of $T(\mathcal{F})$, and a copy of $T(\mathcal{F})$ can enumerate $\mathcal{F}$.

Definition 3.3. For a set $W \subseteq \omega$, let $T(W)$ be isomorphic to the following subtree of $\omega^{<\omega}$:

$$\{\varnothing\} \cup \{n0^m : m \leq n \in W\} \cup \{n0^n0 : n \in W\} \cup \{n0^n1 : n \in W\}.$$

Let $T(\mathcal{F})$ by the tree with a root node, whose children are the roots of subtrees $T(W)$ for $W \in \mathcal{F}$.

For example, for $W = \{1, 2, 5\}$, $T(W)$ is the following tree:



We can easily enumerate W from $T(W)$, and compute a copy of $T(W)$ from an enumeration of W (though the domain might not be an intial segment of ω). Thus an enumeration of $\mathcal{F}$ computes a copy of $T(\mathcal{F})$, and a copy of $T(\mathcal{F})$ can enumerate $\mathcal{F}$.

We will now consider a number of construction which proceed by coding a family into a structure. Consider for example the result of Slaman [Sla98] and Wehner [Weh98] that there is a structure whose degree spectrum is exactly the non-computable degrees. The following theorem is explicit in Wehner's proof, and Slaman mentions that it follows easily from his proof:

Theorem 3.4 (Slaman [Sla98] and Wehner [Weh98])**.** *There is a family $\mathcal{F}$ which can be enumerated by exactly the non-computable sets.*

Taking this family $\mathcal{F}$ and building $T(\mathcal{F})$, we immediately obtain:

Corollary 3.5. *There is a tree which has a computably copy in every non-computable degree, but no computable copy.*

So there is a tree with Slaman-Wehner degree spectrum.

We can do the same for a number of other results. We will give two more examples of a computable tree with finite computable dimension greater than 1, and of a computable tree which is computably categorical, but not relatively computably categorical. The original constructions of structures with both of these properties proceeded explicitly through construction a family of sets.

Theorem 3.6 (Goncharov [Gon80])**.** *For each n, there is a family $\mathcal{F}$ that has exactly n enumerations up to computable equivalence.*

Corollary 3.7. *For each n, there is a computable tree of computable dimension n.*

Theorem 3.8 (Badaev [Bad77], Selivanov [Sel76], Goncharov [Gon77])**.** *There is a family $\mathcal{F}$ that has one enumeration up to computable equivalence, but for some X, multiple non-X-equivelent X-enumerations.*

Corollary 3.9. *There is a tree which is computably categorical but not relatively computably categorical.*

We will not give a formal construction, but we can even generalize the ideas of this section to families of families, and so on. So any such construction can be replicated by a tree, and this includes constructions using marker extensions etc., and in fact we do not know of a construction which cannot be emulated by a tree in this way.

4 Back-and-forth Trees

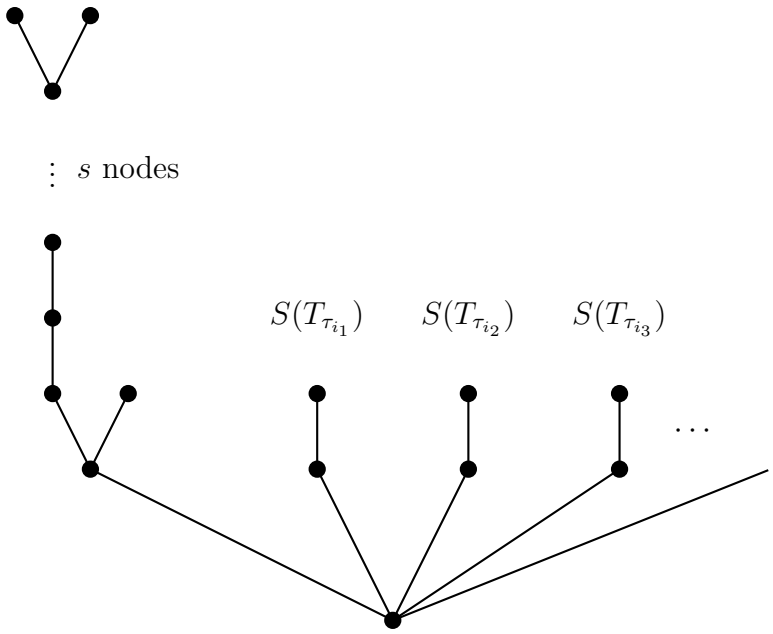
We will now describe the construction of the back-and-forth tree of a structure. Recall that this was implicit in work of Friedman and Stanley [FS89]. The construction is simpler in the case of structures which have only finitely many relations of each arity, but a standard trick allows us to transform any structure into a structure of this type, e.g. by replacing an n-ary relation R by an $m+n$-ary relation S satisfying $S(\bar{x}, \bar{y}) \iff R(\bar{x})$.

Definition 4.1. Let L be a set of infinitely many unary relations ℓ called labels. A labeled tree is a tree T such that each node of T has exactly one label.

The Friedman-Stanley construction did not make the labels explicit, but rather coded them directly into the tree. Any labeled tree can be coded into a tree without labels.

Proposition 4.2. *For every labeled tree T, there is a tree $S = S(T)$ such that S and T are effectively bi-interpretable.*

Proof. Let $(\ell_s)_{s\in\omega}$ be an enumeration of the labels L. Given $\sigma \in T$, let T_σ be the subtree of T with root σ. For each node $\sigma \in T$ with label ℓ_s and children $(\tau_i)_{i\in I}$, we will define a tree $S(T_\sigma)$ using in its construction subtrees $S(T_{\tau_i})$:



Given a copy of T, it is easy to see how to build a copy of $S(T)$. Given a copy S^* of $S(T)$, one builds a copy T^* of T as follows. The nodes of T^* at level n will be the nodes σ of S^* which are at level $2n$ and such that for all $k < n$, $\sigma\restriction_{2k}$ has a child which itself has two children, and this child is not $\sigma\restriction_{2k+1}$. The extension relation will be the same in S^* as it was in T^* (so that σ is the parent of τ in T^* if and only if σ is the parent of the parent of τ in S^*). The labels can be recovered as follows. Given $\sigma \in T^*$, find the child τ of σ in S^* which itself has two children. One of the children ρ of τ itself has a child. Count the number s of children of children of ρ until we find some node below ρ which has two children. Then σ is labeled ℓ_s. □

The back-and-forth tree is defined as follows:

Definition 4.3 (Friedman-Stanley [FS89])**.** Let $\mathcal{A}$ be a structure. Define the labeled tree $\mathcal{T}(\mathcal{A})$ as follows. The nodes of $\mathcal{T}(\mathcal{A})$ are the tuples from $\mathcal{A}$. The extension relation is the extension of tuples. Label each tuple with (a code for) its atomic type.

The back-and-forth tree is an isomorphism invariant.

Proposition 4.4. *Two structures which have isomorphic back-and-forth trees are isomorphic.*

Proof. The isomorphism between the back-and-forth trees induces a map $f\colon \mathcal{A}^{<\omega} \to \mathcal{B}^{<\omega}$ such that:

- $f(\varnothing) = \varnothing$,
- for each tuple $\bar{a} \in \mathcal{A}$ and element $a' \in \mathcal{A}$, $f(\bar{a}a') = f(\bar{a})b$ for some $b \in \mathcal{B}$,
- for each $\bar{a} \in \mathcal{A}$, $f(\bar{a})$ is has the same atomic type as $\bar{a}$.

We will define an isomorphism $g\colon \mathcal{A} \to \mathcal{B}$ using a back-and-forth construction. Begin by picking the first element $a_0 \in \mathcal{A}$ and define $g(a_0) = b_0 = f(a_0)$. Then let b_1 be the first element in $\mathcal{B}$ not yet chosen, and let a_1 be such that $f(a_0a_1) = b_0b_1$. Define $g(a_1) = b_1$. Then let a_2 be the first element of $\mathcal{A}$ not yet chosen, and define $g(a_2) = b_2$ where b_2 is such that $f(a_0a_1a_2) = b_0b_1b_2$. Continuing in this way, we define an isomorphism g. □

There is a natural map $f\colon \mathcal{T}(\mathcal{A}) \to \mathcal{A}^{<\omega}$ which takes a node of $\mathcal{T}$ to the corresponding tuple of $\mathcal{A}$. This map has the property that it takes a node to a tuple whose atomic type corresponds to the label of the node. Though it is the canonical such map, it is not the only such map; and moreover, if T is a labeled tree isomorphic to $\mathcal{T}(\mathcal{A})$, then there is no canonical such map $T \to \mathcal{A}^{<\omega}$. We call such a map a pseudo-isomorphism.

Definition 4.5. Let $\mathcal{A}$ be a structure and let T be a tree labeled with atomic types of $\mathcal{A}$. An embedding $f\colon T \to \mathcal{A}^{<\omega}$ is a pseudo-embedding if:

- f maps the root node of T to the empty tuple,
- whenever τ is a child of σ, there is $a \in \mathcal{A}$ such that $f(\tau) = f(\sigma)^\frown a$,
- each $\sigma \in T$ is labeled with the atomic type of $f(sigma)$.

We say that f is a pseudo-isomorphism if it is also surjective.

It is easy to see that if there is a pseudo-isomorphism $f\colon T \to \mathcal{A}^{<\omega}$ then T is isomorphic to the back-and-forth tree of $\mathcal{A}$.

Proposition 4.6. *If T is a tree, then $DgSp(T) = DgSp(\mathcal{T}(T))$.*

Proof. It is clear that $DgSp(T) \subseteq DgSp(\mathcal{T}(T))$ as each copy of T computes a copy of $\mathcal{T}(T)$. We claim that each copy of $\mathcal{T}(T)$ computes a copy of T.

Note that T is isomorphic under a natural identification to the subtree of $\mathcal{T}(T)$ consisting of those tuples $a_1, \ldots, a_n$ where a_1 is a child of the root node, a_2 is a child of a_1, and so on. This subtree is Σ_1-definable in $\mathcal{T}(T)$ because the atomic type of $a_1, \ldots, a_n$ determines whether this is true. It follows that each copy of $\mathcal{T}(T)$ computes a copy of T. □

Later we will use the fact that if we have a disjoint union of structures, we can compute the back-and-forth tree of the disjoint union from the back-and-forth trees of each component.

Lemma 4.7. *Let $\mathcal{A}$ be the disjoint union of $\mathcal{A}_i$. Suppose that $\mathcal{T}(\mathcal{A}_i)$ has a computable copy uniformly in i. Then $\mathcal{T}(\mathcal{A})$ has a computable copy.*

Proof. TODO □

5 Marker extensions

5.1 Δ_α Marker extensions

We use the Marker extensions described in []. Let $\alpha \geq 1$ be a computable successor ordinal. Fix computable structures $\mathcal{B}_0$ and $\mathcal{B}_1$ in the same relational language such that:

1. the pair $\{\mathcal{B}_0, \mathcal{B}_1\}$ is α-friendly,

2. $\mathcal{B}_0$ and $\mathcal{B}_1$ satisfy the same Σ_β sentences for $\beta < \alpha$,

3. each of $\mathcal{B}_0$ and $\mathcal{B}_1$ satisfies a computable Σ_α sentence that the other does not.

Such structures exist; see [].

Given a structure $\mathcal{A}$, define the Σ_α Marker extension $\mathcal{A}_{\Sigma_\alpha}$ of $\mathcal{A}$ as follows. Let $\{R_i : i \in I\}$ be the language of $\mathcal{A}$, with R_i of arity a_i. The domain of $\mathcal{A}_{\Sigma_\alpha}$ is $A \cup \bigcup_{i \in I} U_i$ where A is the domain of $\mathcal{A}$ and the U_i are disjoint infinite sets. $\mathcal{A}_{\Sigma_\alpha}$ has unary relation P_A and P_i picking out A and U_i respectively. For each i there is also an $a_i + 1$-ary relation Q_i assigning to each a_i-tuple $\bar{u}$ an infinite set $U_{i,\bar{u}}$, with $v \in U_{i,\bar{u}}$ if and only if $Q_i(\bar{u}, v)$. The sets $U_{i,\bar{u}}$ are infinite and partition U_i. For each i and $\bar{u} \in A^{a_i}$, $U_{i,\bar{u}}$ is the domain of a structure $\mathcal{U}_{i,\bar{u}}$ which is isomorphic to either $\mathcal{B}_0$ or $\mathcal{B}_1$, with $\mathcal{U}_{i,\bar{u}} \cong \mathcal{B}_0$ if $\mathcal{A} \models R_i(\bar{u})$, and $\mathcal{U}_{i,\bar{u}} \cong \mathcal{B}_1$ otherwise.

Then we have:

Lemma 5.1 (See Lemma 5.5 of []). *Let α be a computable successor ordinal and let $\mathcal{A}$ be a structure. Then $\mathcal{A}$ has a Δ^0_α copy if and only if $\mathcal{A}_{\Delta_\alpha}$ has a computble copy.*

We can also prove a similar fact about the back-and-forth trees of a structure.

Lemma 5.2. *Let $\mathcal{A}$ be a structure and suppose that $\mathcal{T}(\mathcal{A})$ has a Δ_α copy. Then $\mathcal{T}(\mathcal{A}_{\Delta_\alpha})$ has a computable copy.*

Proof. TODO □

5.2 Σ_α Marker extensions

Let $\alpha \geq 1$ be a computable successor ordinal, though we will only need the case $\alpha = 1$. Fix computable structures $\mathcal{B}_0$ and $\mathcal{B}_1$ in the same relational language such that:

1. the pair $\{\mathcal{B}_0, \mathcal{B}_1\}$ is α-friendly,

2. $\mathcal{B}_0 \leq_\alpha \mathcal{B}_1$ ($\mathcal{B}_0$ satisfies every Σ_α formula satisfied by $\mathcal{B}_1$)

3. $\mathcal{B}_0$ satisfies some computable Σ_α sentence that $\mathcal{B}_1$ does not.

Such structures exist; for example, for $\alpha = 1$, we can take $\mathcal{B}_0$ and $\mathcal{B}_1$ to be infinite structures in a language with a single unary relation P, and have $\mathcal{B}_0$ have a single element satisfying P, while $\mathcal{B}_1$ has no elements satisfying P. Define $\mathcal{A}_{\Sigma_\alpha}$ in the same way that we defined $\mathcal{A}_{\Delta_\alpha}$, except using the new $\mathcal{B}_0$ and $\mathcal{B}_1$. We sometimes write $\mathcal{A}_\exists$ for $\mathcal{A}_{\Sigma_1}$. We can prove the corresponding lemmas.

If $\mathcal{A}$ is a structure, a Σ_α copy of $\mathcal{A}$ is one in which the relations are Σ^0_α. Note that such a copy is automatically $\Delta^0_{\alpha+1}$.

Lemma 5.3 (See Lemma 5.5 of []). *Let α be a computable successor ordinal and let $\mathcal{A}$ be a structure. Then $\mathcal{A}$ has a Σ^0_α copy if and only if $\mathcal{A}_{\Sigma\alpha}$ has a computble copy.*

Proof. If $\mathcal{A}_{\Sigma_\alpha}$ has a computable copy, then it is not hard to make a Σ^0_α copy of $\mathcal{A}$; given $\bar{u}$ and R_i, have $\mathcal{A} \models R_i(\bar{u})$ if and only if $\mathcal{U}_{i,\bar{u}} \cong \mathcal{B}_0$, and this is Σ^0_α. (We use here the fact that $\mathcal{B}_0$ satisfies a computable Σ_α sentence that $\mathcal{B}_1$ does not.)

For the other direction, if $\mathcal{A}$ has a Σ^0_α copy, we can use the uniform version of the following fact (see ??) to build a computable copy of $\mathcal{A}_{\Sigma_\alpha}$: For any Σ^0_α set S, there is a uniformly computable sequence $(\mathcal{C}_n)_{n\in\omega}$ such that $\mathcal{C}_n \cong \mathcal{B}_0$ if $n \in S$, and $\mathcal{C}_n \cong \mathcal{B}_1$ otherwise. □

Lemma 5.4. *Let $\mathcal{A}$ be a structure and suppose that $\mathcal{T}(\mathcal{A})$ has a Σ_α copy. Then $\mathcal{T}(\mathcal{A}_{\Sigma_\alpha})$ has a computable copy.*

Proof. TODO □

6 Main Result

In this section, we will prove the main result of this paper.

Theorem 6.1. *There is a structure $\mathcal{A}$ with no computable copy such that $\mathcal{T}(\mathcal{A})$ has a computable copy.*

The proof is quite involved and will extend over the rest of this section. We build $\mathcal{A}$ as the disjoint union of structures $\mathcal{A}_n$, with $\mathcal{A}_n$ satisfying a unary relation R_n. We will make sure that $\mathcal{A}_n$ is not isomorphic to the structure with domain R_n in the nth computable structure. Thus $\mathcal{A}$ will have no computable copy. $\mathcal{T}(\mathcal{A}_n)$ will have a computable copy which is computable uniformly in n, and so by Lemma 4.7, $\mathcal{T}(\mathcal{A})$ will have a computable copy.

Fix n for which we will define $\mathcal{A} = \mathcal{A}_n$ which is not isomorphic to $\mathcal{B}$, the structure with domain R_n in the nth (possibly partial) computable structure. The language of $\mathcal{A}$ will consist of infinitely many unary relations called labels and infinitely many n-ary relations for each n. The relations R will be relations on unordered tuples. We'll consider these relations to all be c.e. relative to a presentation of $\mathcal{A}$. Similarly, we will consider the trees to have c.e. labels. The actual structure will be an existential Marker extensions of these relations; Lemmas 5.3 and 5.4 show that this all works out.

We will build $\mathcal{A}$ as a limit of a computable sequence of finite structures on increasing domains. The construction will proceed by stages, each divided into at least three, but possibly more, steps. At stage s, step t, we will define $\mathcal{A}_{s,t}$. At each stage s and step t, after defining $\mathcal{A}_{s,t}$, we wait for an i such that $\mathcal{B}_i$ is isomorphic to $\mathcal{A}_{s,t}$. If there is no such stage, then we just put $\mathcal{A} = \mathcal{A}_{s,t}$ and win; in this case, there are only finitely many stages of the construction. If the construction lasts for infinitely many stages, then there will be a sequence $s_0 \prec s_1 \prec s_2 \prec \cdots$ of true stages, and we will have $\mathcal{A} = \bigcup_{s_i} \mathcal{A}_{s_i,0}$ as a nested union. When describing the construction we will generally assume that for each stage s and step t a value i as above exists, and write $\mathcal{B}_{s,t}$ for $\mathcal{B}_i$; note that the $\mathcal{B}_{s,t}$ are nested, with $\mathcal{B}$ being their union. It is this asymmetry—that the opponent must produce nested structures while we do not—that we will exploint to make $\mathcal{A}$ not isomorphic to $\mathcal{B}$; however, we will still have to show that a copy of $\mathcal{T}(\mathcal{A})$ is computable uniformly, whether there are finitely or infinitely many stages of the construction, and so we are actually quite restricted in what we can do.

To the stage s we associate a number $n(s)$. The true stages are those stages s for which, for every $t \geq s$, $n(t) \geq n(s)$. If $s \leq t$, then t *believes* s if for all $s < s' < t$, $n(s') \geq n(s)$. If t believes s, then if t is a true stage then s must be a true stage. Write $s \preceq t$ if t believes s. It will always be the case that $n(s+1) = n(s)+1$ or $n(s+1) < n(s)$; the value of n can never stay the same from one stage to the next, or increase by more than one.

To show that $\mathcal{T}(\mathcal{A})$ is computable, we will build a computable sequence of labeled trees $T_{s,t}$, each isomorphic to the corresponding $\mathcal{T}(\mathcal{A}_{s,t})$, and such that

$$T_{0,0} \subseteq T_{0,1} \subseteq T_{0,2} \subseteq \cdots \subseteq T_{1,0} \subseteq T_{1,1} \subseteq \cdots \subseteq T_{2,0} \subseteq \cdots .$$

Since the $\mathcal{A}_{s,t}$ are not nested, it is not true that

$$\mathcal{T}(\mathcal{A}_{0,0}) \subseteq \mathcal{T}(\mathcal{A}_{0,1}) \subseteq \mathcal{T}(\mathcal{A}_{0,2}) \subseteq \cdots \subseteq \mathcal{T}(\mathcal{A}_{1,0}) \subseteq \mathcal{T}(\mathcal{A}_{1,1}) \subseteq \cdots \subseteq \mathcal{T}(\mathcal{A}_{2,0}) \subseteq \cdots$$

under the natural inclusion of their domains; so the structure $\mathcal{A}$ will have to be constructed carefully to ensure that the $T_{s,t}$ are nested. The tree T which is the union of the $T_{s,t}$ is computable, but we must argue that it is actually isomorphic to $\mathcal{T}(\mathcal{A})$. To see this, we will also keep track of pseudoisomorphisms $g_{s,t} \colon T_{s,t} \to \mathcal{A}_{s,t}$ and try to ensure that $g = \lim g_{s,0}$ is a pseudoisomorphism $T \to \mathcal{A}$. To make sure that this happens, during the construction we will make sure that if $s \preceq s'$ then $g_{s',0}$ does not change too much from $g_{s,0}$; more precisely:

- if $s \preceq s'$ and $n(s) = n(s')$ then $g_{s,0} \subseteq g_{s',0}$; and
- if $s \preceq s'$ and $n = n(s) < n(s')$ then $g_{s,0}$ and $g_{s',0}$ agree on all nodes of $T_{s,0}$ of height $\leq n$.

Then there are three cases to consider:

1. There are only finitely many stages. If s and t are the last stage and step, then $\mathcal{A} = \mathcal{A}_{s,t}$ and $T = T_{s,t} \cong \mathcal{T}(\mathcal{A}_{s,t})$ so we are done.
2. There is n such that for some s_0, for all true stages $s \geq s_0$, $n(s) = n$. In this case, $g = \bigcup_{\text{true } s \geq s_0} g_{s,0}$ is a nested union.
3. There is no bound on $n(s)$ for s a true stage. Then the $g_{s,0}$ come to a limit g defined as follows. For $\sigma \in T$, $g(\sigma) = g_{s,0}(\sigma)$ where s is any true stage which is sufficiently large that $\sigma \in T_{s,0}$ and $n(s) > |\sigma|$. This g is a pseudoisomorphism $T \to \mathcal{A}$.

See Lemma 6.8 for the details.

At each stage s and step t, every element of $\mathcal{A}_{s,t}$ will have a label ℓ which holds of that element and no other element. We call this ℓ the (s,t)-distinguishing label of a (or the s-distinguishing label is $t = 0$); as the notation implies, the distinguishing label of a particular element of $\mathcal{A}$ will change over time. At each stage s and step t, we will also have certain elements which are designated as *killed*. What this means is that these elements will never again be given a new label, and they will keep the same distinguishing label. Once an element is killed it will remain killed from then on, even if it became killed at a non-true stage.

At stage s, let $n = n(s)$ and let $t_0 \prec t_1 \prec t_2 \prec \cdots \prec t_n = s$ be the previous stages believed by s such that t_i is the least stage with $n(t_i) = i$. The elements of $\mathcal{A}_{s,0}$ will consist of:

- c,
- $a_1, \ldots, a_n$,
- for each $1 \leq i < n$, $a'_{1,t_i}, \ldots, a'_{i,t_i}$ for each i, and
- killed elements.

These elements are all distinct. Over the course of stage s, we will introduce elements c'_s and $a'_{1,s}, \ldots, a'_{n,s}$. If $n(s+1) = n(s) + 1$, so that s is a true stage, then we will have $a_{n+1} = c'_s$.

These values are dependent on the stage. If $s \preceq t$, then t agrees with all of the values defined at stage s; but a stage s with $n = n(s)$ which is not a true stage may define an value for a_n which is later cancelled (that element becoming killed) when we find out that s is not a true stage, and a later stage t with $n = n(t)$ may then redefine a_n as a new element. All true stages agree on the values of these elements.

Suppose that the construction has infinitely many stages; we must argue that $\mathcal{A}$ is not isomorphic to $\mathcal{B}$. The element $c \in \mathcal{A}$ will be special: we will ensure that $\mathcal{B}$ does not have an element isomorphic to it. Because $\mathcal{B}_{s,t}$ is isomorphic to $\mathcal{A}_{s,t}$, and each element of $\mathcal{A}_{s,t}$ has a distinguishing label, there is a unique isomorphism $f_{s,t}$ between $\mathcal{A}_{s,t}$ and $\mathcal{B}_{s,t}$. We may assume that the domain of $\mathcal{B}_{s,t}$ is an initial segment of ω. During the construction we will make sure that at every stage s, $f_{s,0}(a_1), \ldots, f_{s,0}(a_n) < f_{s,0}(c)$ in the standard order on ω; it is the opponent that decides how to build $\mathcal{B}$, so we will have to construct $\mathcal{A}$ in such a way that the opponent is forced to maintain this. Thus if, along the true stages, $\lim_s n(s) \to \infty$, then $\lim_s f_{s,0}(c) \to \infty$, and so $\mathcal{B}$ will have no element isomorphic to $c \in \mathcal{A}$. (To make this true, we also guarantee that if s is a true stage, c will never be given the s-distinguishing label of any other element of $\mathcal{A}_{s,0}$.) The other possibility is that for sufficiently large true stages s, the value of $n(s)$ stays the same, say $n(s) = n$. Whenever we have $s \preceq s'$ and $n(s) = n(s')$, we will have $f_{s,0}(c) < f_{s',0}(c)$; this will be because a new element shows up in $\mathcal{B}$ below the image of c, and then that new element is then killed. So in this case as well, we will have that $\mathcal{B}$ does not contain an element isomorphic to c.

6.1 First few stages of the construction

In an informal way we will go through the first few steps of the construction. In Figures 1 and 2 we show the first two stages of the construction, ending in stage 2 step 0. The two figures show the two possibilities, depending on how $\mathcal{B}$ responds. We always have $n(0) = 0$ and $n(1) = 1$ since we cannot have $n(1) < n(0) = 0$. But then we could either have $n(2) = 2$ in which case 0, 1, and 2 are all true stages (at least so far), or $n(2) = 0$, in which case 0 and 2 are true stages so far, but 1 is not. The figures show $\mathcal{A}_{s,t}$ for each stage, the response $\mathcal{B}_{s,t}$ by our opponent, and the tree $T_{s,t}$. The nodes of the tree are labeled with the images of the pseudoisomorphism $g_{s,t} \colon T_{s,t} \to \mathcal{A}_{s,t}$. Elements of the structures and the tree are represented by black dots; two dots which are in the same position from one diagram to the next represent the same elements of the domain. So, for example, the node on the first level of the tree at $0, 1$ which is labeled c is the same as the node at $0, 2$ labeled c'; what has happened is that the image of this node under the pseudoisomorphism has changed. In

the structures $\mathcal{A}$ and $\mathcal{B}$, the numbers below an element represent the labels given to that element. The elements of $\mathcal{B}$ are labeled with their preimages under $f_{s,t}$ though we drop the subscript to save space.

Begin at stage 0 (and step 0) with $\mathcal{A}_0$ consisting of a single element c with the label '1'. At step 1, we introduce a new relation R_0 and have it hold of c; this is represented in the diagrams by the line attached to the element. At step 2, we introduce a new element c'_0 and change the structure to have R_0 hold of c'_0 instead of c. Both c and c'_0 have the same label '1' that c had at step 1, but they each get a new label that the other does not have: '2' for c and '3' for c'_0. Note that $\mathcal{A}_{0,1} \not\subseteq \mathcal{A}_{0,2}$. Now $\mathcal{B}$ must copy $\mathcal{A}$, except that as it has already put R_0 on the first element of its domain, this element must copy c'_0 rather than c. We must also expand the tree, and adjust the pseudoisomorphism; the node that previously mapped to c now maps to c'_0. Finally, at stage 1 step 0, we put R_0 on all of the elements of $\mathcal{A}$ and promise to put it on any new elements that show up from now on. This essentially means that we can forget about R_0 from now on, and we no longer draw it. We also set $n(1) = 1$ and $a_1 = c'_0$. Note that $f(a_1) < f(c)$, so that we have made one step towards forcing $\mathcal{B}$ to omit an image of the element $c \in \mathcal{A}$.

Now at stage 1 step 1, we put a new binary relation R_1 on c and a_1; this relation will be on unordered tuples. $\mathcal{B}_{1,1}$ must copy this. At step 2, we introduce new elements c'_1 and $a'_{1,1}$ which have the same labels that c and a_1 had respectively in $\mathcal{A}_{1,1}$. Each of the four elements gets a new label so that they are still distinguished. In $\mathcal{A}_{1,2}$, we put R_1 on the pairs $c, a'_{1,1}$ and c'_1, a_1 but not on c, a_1; so $\mathcal{A}_{1,1} \not\subseteq \mathcal{A}_{1,2}$. Before discussing our opponent's possible responses, let us talk about the tree $T_{1,2}$. Once again the pseudoisomorphism $T_{1,1} \to \mathcal{A}_{1,1}$ cannot be extended to a pseudoisomorphism $T_{1,2} \to \mathcal{A}_{1,2}$. However, we can keep the pseudoisomorphism the same on the first level of the tree, because the tuple c, a_1 satisfies the same atomic formulas in $\mathcal{A}_{1,1}$ as the tuples c'_1, a_1 and $c, a'_{1,1}$; what is going on here is that any existential formula true in $\mathcal{A}_{1,1}$ with one($= n(1)$) parameter is still true of the same parameter in $\mathcal{A}_{1,2}$.

Our opponent must copy $\mathcal{A}_{1,2}$, and they have two choices; they can either turn what used to be the image of c into the image of c'_1, or they can turn what used to be the image of a_1 into the image of $a'_{1,1}$. In Figure 1 we show the former, and in Figure 2 we show the latter. We will begin by discussing the latter. Since $f(c'_1) < f(c)$, at stage 2 step 0 we can set $a_2 = c'_1$, $n(2) = 2$, and put the relaton R_1 on every pair of elements of $\mathcal{A}$ (so that we stop drawing it). Note that we have $f_{2,0}(a_1), f_{2,0}(a_2) < f_{2,0}(c)$ as desired. We have also preserved, from stage 1 to stage 2, the pseudoisomorphism on the first level of the tree.

Let us now consider the other possibility for the opponent's response $\mathcal{B}_{1,1}$, as shown in Figure 2. The key is that the opponent has put the image of a_1 after the image of c. In $\mathcal{A}_{2,0}$, we will do the following. First, we put R_1 on every pair of elements, and stop drawing it. Second, we take all of the labels on $c'_0 = a_1$ in $\mathcal{A}_{1,2}$ and give them to c, and vice versa. Both of these elements receives a new distinguishing label. Finally, we set $n(2) = 0$ and kill all of the elements that were introduced since stage 0, i.e., all of the elements other than c. In the diagrams, we use a circle to represent an element that has been killed.

Note that in the tree $T_{2,0}$, we are able to switch back the swap of c and c'_0 that we made at stage 0 step 2. This is because each of these elements satisfies in $\mathcal{A}_{0,2}$ all of the same existential formulas the other satisfied in $\mathcal{A}_{1,2}$. Thus the pseudoisomorphism $T_{2,0} \to \mathcal{A}_{2,0}$ extends the pseudoisomorphism $T_{0,0} \to \mathcal{A}_{0,0}$ at the previous 2-true stage. The unlabeled elements of the tree are mapped to a killed element, and so will never have to change from

now on.

The opponent is also allowed to switch, from $\mathcal{B}_{1,2}$ to $\mathcal{B}_{2,0}$, the images of c and $c'_0 = a_1$. However, because the opponent made the image of $c'_0 = a_1$ larger than the image of c, this will not allow them to decrease the image of c. So essentially we have returned to where we were at stage 0, except that we have some new elements which are all killed (and hence will not interfere with the construction) and that the image of c in $\mathcal{B}_{2,0}$ has increased compared to what it was in $\mathcal{B}_{0,0}$, so that we have made some progress towards having $\mathcal{A} \not\cong \mathcal{B}$.

An additional consideration. At stage s where $n(s) > 2$, it is possible that we will require more steps. We illustrate this in Figure 3. Due to space constraints, we will no longer keep track of the trees $T_{s,t}$. We begin at stage 2 step 0 where we left off in the case $n(2) = 2$ as in Figure 1. We begin by introducing a ternary relation R_2 in step 1, followed by elements c'_2, $a'_{1,2}$, and $a'_{2,2}$ in step 2. Suppose that the opponent responds with $\mathcal{B}_{2,2}$ as shown. To take advantage of the fact that they have put the image of a_2 above that of c—and so the image of the new element $a'_{2,2}$ is below c—in $\mathcal{A}_{2,3}$ we give c every label that a_2 had in $\mathcal{A}_{2,2}$, and vice versa. We also give a_1 every label that $a'_{1,1}$ had in $\mathcal{A}_{2,2}$. This allows us to undo the injury to the pseudoisomorphism $T_{s,t} \to \mathcal{A}_{s,t}$ that was done at stage 1 step 2 (on the second level of the tree, see Figure 1). Now the opponent, in building $\mathcal{B}_{2,3}$, can swap the images of c and $a_2 = c'_1$, and of a_1 and $a'_{1,1}$. One way that they might do so is shown in Figure 3. The issue here is that we have allowed our opponent to make $f(c) < f(a_1)$, and so we cannot go on to the next stage. But what we can do a similar move again: give c all of the labels that a_1 had in $\mathcal{A}_{2,3}$ and vice versa. This allows us to undo the injury done to the pseudo-isomorphism in stage 0 step 2, and we have $n(3) = 0$.

6.2 Formal Construction of $\mathcal{A}$

The formal construction will look slightly different than the informal picture just given. At a stage s, we may not know how many steps there will be until $\mathcal{B}$ allows us to move on to the next stage. Because of this, if t is the last step at stage s, we will have $\mathcal{A}_{s+1,0} = \mathcal{A}_{s,t}$.

We will also keep track, through the steps t at stage s, of a value $m_{s,t}$. This value will be the current guess at $n(s+1)$. If t is the last stage of step s, then we will have $n(s+1) = m_{s,t}$.

Construction.

- *Stage $s = 0$, step 0.* Begin with $\mathcal{A}_0$ consisting of a single element c labeled with a single label.
- *Stage $s+1$, step 0.* Let t be the last step at stage s. Let $\mathcal{A}_{s+1,0} = \mathcal{A}_{s,t}$. Let $n(s+1) = m_{s,t}$. If $n(s+1) > n(s)$, let $a_{n(s+1)} = c'_s$. Otherwise, cancel everything but the values:
  - c,
  - $a_1, \ldots, a_{n(s+1)}$,
  - $a'_{1,s'}, \ldots, a'_{i,s'}$ for each $1 \leq i < n(s+1)$ and $s' \prec s+1$.
- *Stage s, step 1.* Choose a new $n+1$-ary relation R_s. $\mathcal{A}_{s,1}$ will be $\mathcal{A}_{s,0}$ except that we have
$$R_s(c, a_1, \ldots, a_n).$$

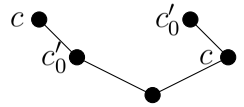
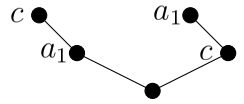
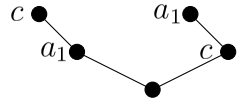
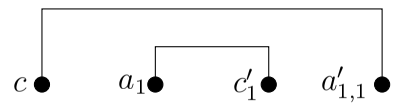
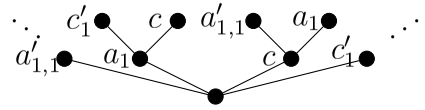
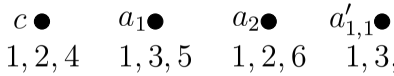
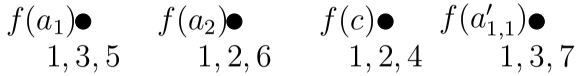
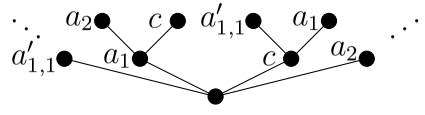
s,i	$\mathcal{A}_{s,i}$	$\mathcal{B}_{s,i}$	$T_{s,i}$
0,0			
0,1			
0,2			
1,0			
1,1			
1,2			
2,0			

Figure 1: The case $n(2) = 2$.

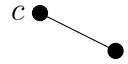
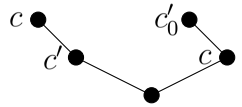
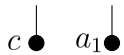
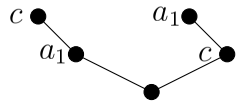
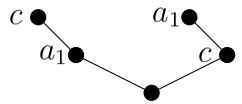
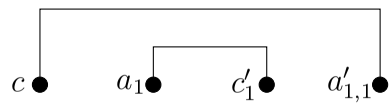
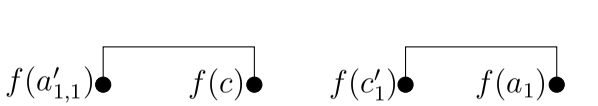
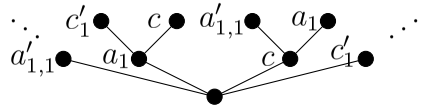
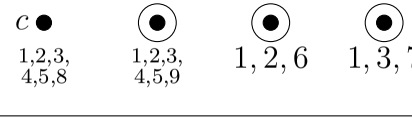
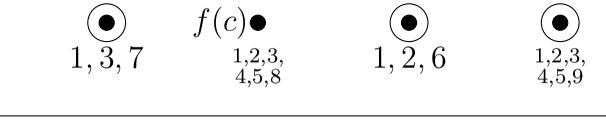
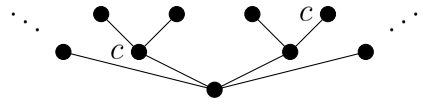
s,i	$\mathcal{A}_{s,i}$	$\mathcal{B}_{s,i}$	$T_{s,i}$
0,0			
0,1			
0,2			
1,0			
1,1			
1,2			
2,0			

Figure 2: The case $n(2) = 0$.

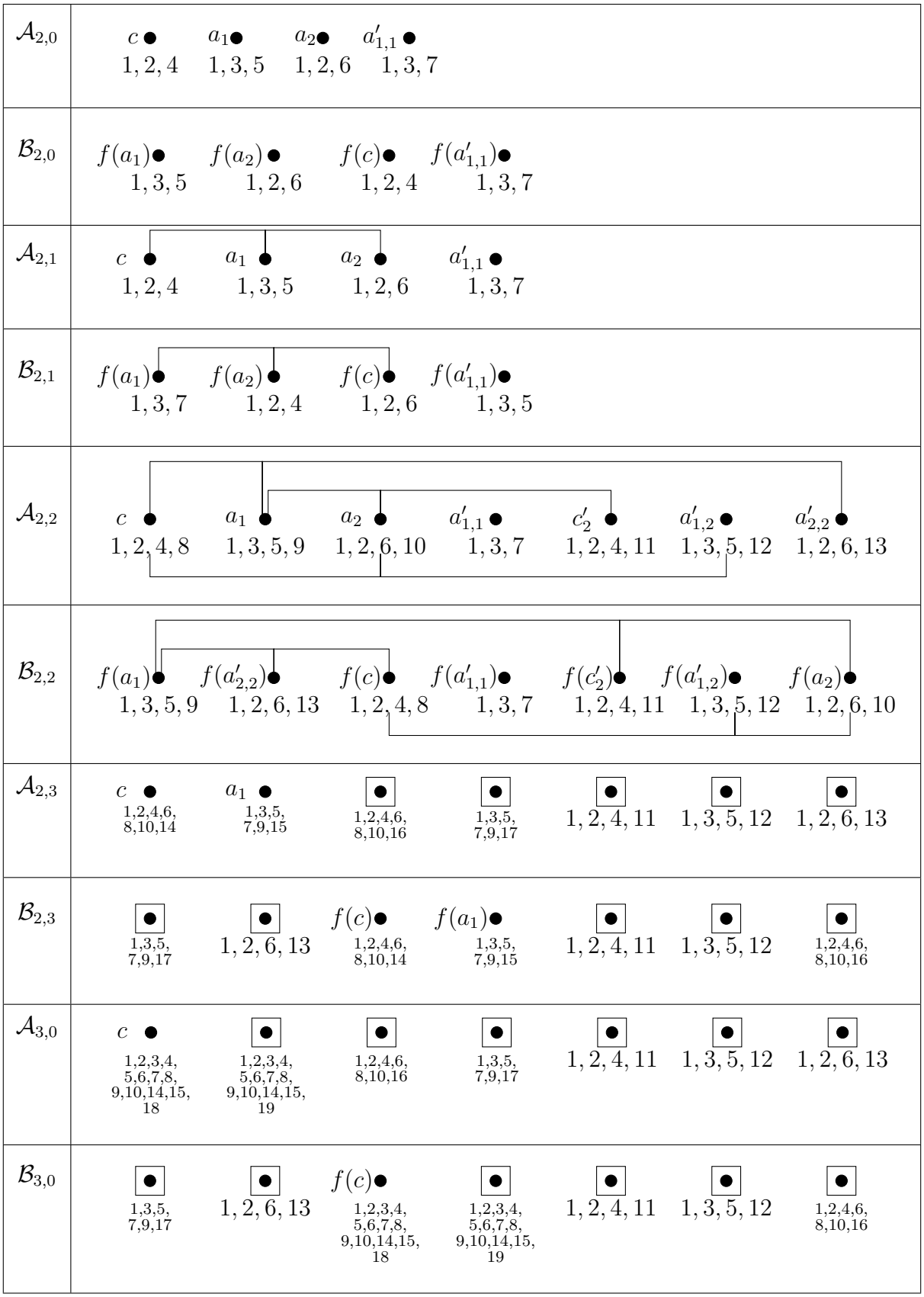


Figure 3: One possibility for stage 2.

Recall that R_s is a relation on unordered tuples.

- *Stage s, step* 2. Since we did not add any new elements or labels in $\mathcal{A}_{s,1}$, we have that $f_{s,1} = f_{s,0}$. So we have $\mathcal{B}_{s,1} \models R_s(f_{s,0}(c), f_{s,0}(a_1), \ldots, f_{s,0}(a_n))$. (Recall that R_s is a relation on unordered tuples.)

  The structure $\mathcal{A}_{s,2}$ will look like $\mathcal{A}_{s,0}$ but with new elements, labels, and relations. Introduce, in $\mathcal{A}_{s,2}$, new elements $c'_s, a'_{1,s}, \ldots, a'_{n,s}$. These elements have the same labels that the corresponding elements $c, a_1, \ldots, a_n$ had in $\mathcal{A}_{s,0}$, but we give each element a new unique label. So, for example, c and c'_s have all the same labels that c had in $\mathcal{A}_{s,0}$, but each of them has one more label that the other does not. We have relations

  $$R_s(c'_s, a_1, \ldots, a_n) \qquad \text{and, for each } i, \qquad R_s(c, a_1, \ldots, a'_{i,s}, \ldots, a_n).$$

  We do not have $R_s(c, a_1, \ldots, a_n)$. In general, R_s holds of any such tuple where there is exactly one $'$.

- *Stage s, step* 3. Whether we proceed onto stage $s+1$, or add another step to stage s, depends on what $\mathcal{B}$ does.

  Recall that $f_{s,1} = f_{s,0}$ was the isomorphism $\mathcal{A}_{s,1} \to \mathcal{B}_{s,1}$, and so it gives a map $\mathcal{A}_{s,2} \to \mathcal{B}_{s,2}$, but this map is not neccesarily an isomorphism since $\mathcal{A}_{s,1}$ is not a substructure of $\mathcal{A}_{s,2}$ (but $\mathcal{B}_{s,1}$ is a substructure of $\mathcal{B}_{s,2}$). There are two possibilities for $f_{s,2}(c)$; either it is equal to $f_{s,0}(c)$, or it is one of the new elements in $\mathcal{B}_{s,2} - \mathcal{B}_{s,1}$. This is because all of the other elements of $\mathcal{B}_{s,1}$ have a label which c does not. Similarly, for each a_i, either $f_{s,2}(a_i)$ is $f_{s,0}(a_i)$ or it is one of the new elements in $\mathcal{B}_{s,2} - \mathcal{B}_{s,1}$.

  Since $\mathcal{B}_{s,1}$ is a substructure of $\mathcal{B}_{s,2}$, we must have that

  $$\mathcal{B}_{s,2} \models R_s(f_{s,0}(c), f_{s,0}(a_1), \ldots, f_{s,0}(a_n))$$

  since this was true of $\mathcal{B}_{s,1}$. However, it is not true of $\mathcal{A}_{s,2}$ that $\mathcal{A}_{s,2} \models R_s(c, a_1, \ldots, a_n)$. So the isomorphism $f_{s,2} \colon \mathcal{A}_{s,2} \to \mathcal{B}_{s,2}$ cannot extend $f_{s,0}$. There are $n+1$ different $n+1$-tuples which satisfy R_s in $\mathcal{A}_{s,2}$, and it must be one of these that maps to $f_{s,0}(c), f_{s,0}(a_1), \ldots, f_{s,0}(a_n)$ in $\mathcal{B}_{s,2}$. These tuples are $c'_s, a_1, \ldots, a_n$ and, for each i, $c, a_1, \ldots, a'_{i,s}, \ldots, a_n$; moreover, these must map to $f_{s,0}(c), f_{s,0}(a_1), \ldots, f_{s,0}(a_n)$ in order. So we have $n+1$ possibilities divided into two cases:

  Case 1. $f_{s,2}(c) \in \mathcal{B}_{s,2} - \mathcal{B}_{s,1}$. For each $1 \leq i \leq n$, $f_{s,2}(a_i) = f_{s,0}(a_i) \in \mathcal{B}_{s,1}$.

  Case 2. There is j such that $f_{s,2}(a_j) \in \mathcal{B}_{s,2} - \mathcal{B}_{s,1}$. For each $1 \leq i \leq n$, $i \neq j$, $f_{s,2}(a_i) = f_{s,0}(a_i) \in \mathcal{B}_{s,1}$; and $f_{s,2}(c) = f_{s,0}(c) \in \mathcal{B}_{s,1}$.

  In Case 1: This is the last step of stage s. We define $\mathcal{A}_{s,3} \supseteq \mathcal{A}_{s,2}$. Put R_s on every $n+1$-tuple of elements, and on every $n+1$-tuple of elements which is added to $\mathcal{A}$ at any later stage of the construction. Set $m_{s,3} = n(s) + 1$.

  In Case 2: We do not yet move onto stage $s+1$. We define $\mathcal{A}_{s,3} \supseteq \mathcal{A}_{s,2}$. Put R_s on every $n+1$-tuple of elements, and on every $n+1$-tuple of elements which is added to $\mathcal{A}$ at any later stage of the construction.

Let $s' < s$ be the previous stage with $n(s') = j - 1$. Give c all of the labels $a_j = c'_{s'}$ had in $\mathcal{A}_{s,2}$, and give $a_j = c'_{s'}$ all of the labels c had in $\mathcal{A}_{s,2}$. Similarly, for $1 \leq i < j$, give a_i all of the labels $a'_{i,s'}$ had in $\mathcal{A}_{s,2}$, and give $a'_{i,s'}$ all of the labels a_i had in $\mathcal{A}_{s,2}$. Each of these elements gets a new $(s, 3)$-distinguishing label.

Set $m_{s,3} = j - 1$. Continue the stage s with step 4 of stage s.

- *Stage s, step $t + 1 > 3$.* Let $m = m_{s,t}$. We have two cases.

  Case 3. $f_{s,t}(a_1), \ldots, f_{s,t}(a_m) < f_{s,t}(c)$.

  Case 4. For some j, $f_{s,t}(a_j) > f_{s,t}(c)$.

  In Case 3: Step $t + 1$ will be the last step of stage s. We define $\mathcal{A}_{s,t+1} = \mathcal{A}_{s,t}$. Set $m_{s,t+1} = m_{s,t}$.

  In Case 4: Let j be least such that $f_{s,t}(a_j) > f_{s,t}(c)$. Let $s' < s$ be the previous stage with $n(s') = j - 1$. Give c all of the labels $a_j = c'_{s'}$ had in $\mathcal{A}_{s,t}$, and give $c'_{s'}$ all of the labels c had in $\mathcal{A}_{s,t}$. Similarly, for $i < j$, give a_i all of the labels $a'_{i,s'}$ had in $\mathcal{A}_{s,t}$, and give $a'_{i,s'}$ all of the labels a_i had in $\mathcal{A}_{s,t}$. Each of these elements gets a new $(s, t + 1)$-distinguishing label.

  Set $m_{t+1} = j - 1$. Continue the construction with step $t + 2$ of stage s.

End of construction

6.3 Verification

We will begin by proving various lemmas about the construction. First, we show that elements which are introduced at non-true stages are killed when we find out that that stage was not a true stage.

Lemma 6.2. *Suppose that $s \not\preceq s + 1$, and let $s' \leq s$ be the previous $s + 1$-true stage. Then each element of $\mathcal{A}_s - \mathcal{A}_{s'}$ is killed by stage $s + 1$.*

Proof. At stage $s + 1$ step 0, we kill all of the elements other than:

- c,
- $a_1, \ldots, a_{n(s+1)}$,
- $a'_{1,t}, \ldots, a'_{i,t}$ for each $1 \leq i < n(s + 1)$ and $t \prec s + 1$.

These elements were all introduced at stage s' or earlier. □

Lemma 6.3. *Suppose that $s \leq t$ and $x \in \mathcal{A}$ is killed at stage s. Then $f_s(x) = f_t(x)$.*

Proof. Looking at the construction, we see that no new labels are added to a killed element, and so the s-distinguishing label of x is still its t-distinguishing label. □

Lemma 6.4. *Suppose that $s \prec s+1 \prec t$. Then, for each $1 \leq i \leq n(s)$:*

$$f_t(a'_{i,s}) = f_{s+1}(a'_{i,s}) > f_s(c).$$

Proof. Since $s \prec s+1$, we must be in Case 1 at stage s step 3. For each i, $1 \leq i \leq n(s)$, there are only two elements of $\mathcal{A}_{s,2}$ satisfying the s-distinguishing label of a_i—a_i and $a'_{i,s}$—and of these only a_i was in $\mathcal{A}_{s,0}$. So there is only one element of $\mathcal{B}_{s,0}$ satisfying this label, and this element is $f_{s,0}(a_i)$. Since we are in Case 1, $f_{s,2}(a_i) = f_{s,0}(a_i)$ and so $f_{s,2}(a'_{i,s}) \in \mathcal{B}_{s,2} - \mathcal{B}_{s,0}$; thus $f_{s,2}(a'_{i,s}) > f_{s,0}(c)$ since the latter is in $\mathcal{B}_{s,0}$. We also have $f_{s+1}(a'_{i,s}) = f_{s,2}(a'_{i,s})$ since we make no further changes to $\mathcal{A}$ at stage s. So we have shown that $f_{s+1}(a'_{i,s}) > f_{s(c)}$.

Now we claim that $a'_{i,s}$ has no more labels at stage t than it had at stage $s+1$. There are only two times when we give an element a new label at a stage u:

1. When we introduce the elements c'_u and $a'_{1,u}, \ldots, a'_{n(u),u}$, we give a new label to c and $a_1, \ldots, a_{n(u)}$.
2. In Cases 2 and 4 when we give c and $a_j = c'_{s'}$ each other's labels, and a_i and $a'_{i,s'}$ each other's labels, for some $s' < u$.

No new labels are given to $a'_{i,s}$ for the sake of (1). At each stage u, $s+1 \leq u < t$, since s is a u-true stage, (2) can only happen for $j > n(s)$. Thus a new label cannot be added to $a'_{i,s}$ for the sake of (2) before stage t. So $a'_{i,s}$ is still the only element at stage t satisfying its $s+1$-distinguishing label. This means that $f_t(a'_{i,s}) = f_{s+1}(a'_{i,s})$. □

Lemma 6.5. *If $s \prec t$, $f_t(c) > f_s(c)$.*

Proof. We argue inductively. It suffices to show this where s is greatest such that $s \prec t$. It will also be more convenient to adjust the value of t by one: given $s \prec t+1$, show that $f_{t+1}(c) > f_s(c)$.

First, suppose that $s = t$. Then we must be in Case 1 at stage s step 3, as in Case 2 we end up with $n(s+1) < n(s)$. Then $f_{s+1}(c) \in \mathcal{B}_{s,2} - \mathcal{B}_{s,1}$ and so $f_{s+1}(c) > f_s(c)$ as $f_s(c) \in \mathcal{B}_{s,1}$ and this is an initial segment of $\mathcal{B}_{s,2}$.

Second, suppose that $t \not\prec t+1$. We have $s \prec t$ (since $n(s) \leq n(t+1) < n(t)$) and so $f_s(c) < f_t(c)$ by the inductive hypothesis. So if we can show that $f_{t+1}(c) \geq f_t(c)$, then we are done. Since $t \not\prec t+1$, we are in Case 2 at stage t step 3. Note that we then have $f_{t,2}(c) = f_{t,0}(c) = f_t(c)$. We also have $f_{t,2}(c'_t) \in \mathcal{B}_{s,2} - \mathcal{B}_{s,1}$ and so $f_{t,2}(c'_t) > f_{t,2}(c)$. Looking at what we did in the construction in Case 2, we see that either $f_{t,3}(c) = f_{t,2}(c)$ or $f_{t,3}(c) = f_{t,2}(c'_t) > f_{t,2}(c)$ as there is a label that at this point has been given only to c and c' in $\mathcal{A}$, and $f_{t,2}(c)$ and $f_{t,2}(c')$ in $\mathcal{B}$. In either case, $f_{t,3}(c) \geq f_{t,2}(c) = f_t(c)$.

The construction now proceeds to step 4. If, at step 4, we are in Case 3, then we make no further changes to $\mathcal{A}$, and so $f_{t+1}(c) = f_{t,3}(c) \geq f_t(c)$ as desired. On the other hand, suppose that at step 4 we are in Case 4. We will argue inductively on steps $r \geq 3$ that $f_{t,r}(c) \geq f_s(c)$, and then if r is the last step of stage t, $f_{t+1}(c) = f_{t,r}(c) \geq f_s(c)$. We already have the base case $r = 3$. Suppose that we know that $f_{t,r}(c) \geq f_s(c)$. At step $r+1$, if we are in Case 3, then $r+1$ is the last step of stage t and we do nothing to $\mathcal{A}$, so that $f_{t+1}(c) = f_{t,r+1}(c) = f_{t,r}(c) \geq f_s(c)$. So suppose that we are in Case 4. Let j be as in Case 4, with $m_{r+1} = j-1$. We have $f_{t,r}(a_j) > f_{t,r}(c)$. Looking at the construction, we have either $f_{t,r+1}(c) = f_{t,r}(c)$ or $f_{t,r+1}(c) = f_{t,r}(a_j) > f_{t,r}(c)$, as these are the only elements of $\mathcal{A}_{t,r+1}$ with the (t,r)-distinguishing label of c. In either case, $f_{t,r+1}(c) \geq f_{t,r}(c)$ as desired. □

Lemma 6.6. *Suppose that s is a true stage, $x, y \in \mathcal{A}_{s,0}$ with $x \neq y$. Then no element of $\mathcal{A}$ has both the s-distinguishing label of x and the s-distinguishing label of y.*

Proof. The elements of $\mathcal{A}_{s,0}$ are:

- c,
- $a_1, \ldots, a_n$,
- for each $1 \leq i < n$, $a'_{1,t_i}, \ldots, a'_{i,t_i}$ for each i, and
- killed elements

where $n = n(s)$ and $t_0 \prec t_1 \prec t_2 \prec \cdots \prec t_n = s$ are the previous stages believed by s such that t_i is the least stage with $n(t_i) = i$.

Killed elements keep the same distinguishing labels forever, so we may assume that x and y are not killed. There are two times when we give a label already applied to one element to another element at a stage t:

1. When we introduce the elements c'_t and $a'_{1,t}, \ldots, a'_{n(t),t}$ which have the distinguishing labels of c and $a_1, \ldots, a_n$ respectively.
2. In Cases 2 and 4 when we give c and $a_j = c'_{s'}$ each other's labels, and a_i and $a'_{i,s'}$ each other's labels, for some $s' < t$.

After stage s, since s is a true stage, (2) can only happen for $j > n(s)$ and $s' \geq s$. Thus the elements $a'_{1,t_i}, \ldots, a'_{i,t_i}$ for $1 \leq i < n$ keep the same distinguishing labels from stage s on.

We claim that the only elements which can receive the s-distinguishing label of a_i, $1 \leq i \leq n(s)$, are elements of the form $a'_{i,s'}$ for $s' \geq s$. Indeed we see that in both (1) and (2), whenever one of these elements (a_i or $a'_{i,s'}$ for $s' \geq s$) receives a label which had already been given to another element, that other element is also one of these elements. Thus no element can ever be given the s-distinguishing labels of both a_i and a_j, $i \neq j$, or of a_i and c. This proves the lemma. □

Lemma 6.7. *$\mathcal{A}$ is not isomorphic to $\mathcal{B}$.*

Proof. Let $s_0 \prec s_1 \prec s_2 \prec s_3 \prec \cdots$ be the true stages. Given $x \in \mathcal{B}$, we claim that x is not the isomorphic image of c. Using Lemma 6.5, let i be such that $x \in \mathcal{B}_{s_i,0}$ and $f_{s_i}(c) > x$. Then $x = f_{s_i}(y)$ for some $y \in \mathcal{A}_{s_i,0}$, $y \neq c$. So in $\mathcal{B}_{s_i,0}$, x has the s_i-distinguishing label of y. By Lemma 6.6, no element of $\mathcal{A}$ has both the s_i-distinguishing label of y and the s_i-distinguishing label of c. So x cannot be the isomorphic image of c. Thus $\mathcal{B}$ is not isomorphic to $\mathcal{A}$. □

Lemma 6.8. *$\mathcal{T}(\mathcal{A})$ has a computable copy.*

Proof. For each stage s, let i_s be the last step of stage s. We will computably build

$$T_{0,0} \subseteq T_{0,1} \subseteq \cdots \subseteq T_{0,i_0} = T_{1,0} \subseteq \cdots \subseteq T_{1,i_1} = T_{2,0} \subseteq \cdots$$

such that $T_{s,i}$ is isomorphic to $\mathcal{T}(\mathcal{A}_{s,i})$. Moreover, for each stage s, we have a pseudo-isomorphism $f_s \colon T_{s,0} \to \mathcal{A}_{s,0}$. We will argue that $T = \bigcup T_{s,i}$ is isomorphic to $\mathcal{A}$.

Note that we define $T_{s,i}$ at every stage s and *step* i, whereas f_s is only defined for every *stage* s. It is of course possible that there are only finitely many stages, because $\mathcal{B}$ stops copying $\mathcal{A}$. Then, if s, i is the last stage and step, we have $T = T_{s,i}$ is isomorphic to $\mathcal{T}(\mathcal{A}_{s,i}) = \mathcal{T}(\mathcal{A})$. So T is isomorphic to $\mathcal{T}(\mathcal{A})$ even though we did not necessarily build an isomorphism between the two.

If there are infinitely many stages of the construction, then $T = \bigcup T_{s,0}$ will be an infinite union, so we must give an argument that T is isomorphic to $\mathcal{T}(\mathcal{A})$. We will have that $f = \lim f_s$ is a pseudo-isomorphism from $T \to \mathcal{A}$. This limit will be a Δ^0_2 limit along the true stages.

We make a new convention: At stage s, an *active element* of $\mathcal{A}_s$ is one of the elements $c, a_1, \ldots, a_{n(s)}$. We also write $\overline{f}(\sigma)$ for the tuple $f(\sigma\restriction_1), \ldots, f(\sigma\restriction_{|\sigma|})$. The other elements are all inactive. Given stages $s \preceq t$, the maps f_s and f_t will satisfy a certain agreement condition:

($*$) If $n(s) = n(t)$ then $f_s \subseteq f_t$. If $n(t) > k = n(s)$ then whenever $\sigma \in T_{s,0}$ is such that $\overline{f}_s(\sigma)$ contains at most k active elements (but possibly other inactive or killed elements), then $f_s(\sigma) = f_t(\sigma)$.

A second property that we will use to argue that $f = \lim f_s$ stabilizes is as follows:

($\dagger$) Suppose $s \preceq s+1$, so that $n(s) = k$ and $n(s+1) = k+1$. Given $\sigma \in T_{s,0}$, $\overline{f}_{s+1}(\sigma)$ contains at most $k+1$ active elements.

Using ($*$) and ($\dagger$), we can argue that $f = \lim_s f_s$ is a pseudo-isomorphism $T \to \mathcal{A}$. Let $s_0 \preceq s_1 \preceq s_2 \preceq \cdots$ be the sequence of true stages. If there are n and I such that for all $i \geq I$ $n = n(s_i)$, then $f = \bigcup_{i \geq I} f_{s_i}$ and these are nested. So we may assume that $\lim_{i\to\infty} n(s_i) = \infty$.

- Given $\sigma \in T$, we must argue that $f_s(\sigma)$ comes to a limit. If $\sigma \in T_{s_i,0}$, with s_i greatest with that value of $n(s_i) = k$, then $s_{i+1} = s_i + 1$ and $n(s_{i+1}) = n(s_i) + 1 = k+1$. By ($\dagger$), $\overline{f}_{s_{i+1}}(\sigma)$ contains at most $k+1$ active elements. Then we can argue inductively on j, using ($*$), that $f_{s_{i+j+1}}(\sigma) = f_{s_{i+j}}(\sigma)$. So $f(\sigma)$ comes to a limit.

- Given $\bar{a} \in \mathcal{A}^{<\omega}$, we must argue that there is $\sigma \in T$ such that $f(\sigma) = \bar{a}$. Let s_i be a true stage such that $n(s_i) > |\bar{a}|$. Then $\bar{a}$ can have at most $|\bar{a}| < n(s_i)$ live elements. There is σ such that $f_{s_i}(\sigma) = \bar{a}$, and by ($*$), $f(\sigma) = \bar{a}$.

We must now give the construction of T and f and then verify that they satisfy ($*$) and ($\dagger$). We will also build intermediate maps $g_{s,t} \colon T_{s,t} \to \mathcal{A}_{s,t}$, with $g_s = g_{s,0}$.

Construction. Begin with $T_{0,0} = \mathcal{T}(\mathcal{A}_{0,0})$ and f_0 the natural pseudo-isomorphism $T_{0,0} \to \mathcal{A}_{0,0}$. Suppose that we have defined

$$T_{0,0} \subseteq T_{0,1} \subseteq \cdots \subseteq T_{0,i_0} = T_{1,0} \subseteq \cdots \subseteq T_{1,i_1} = T_{2,0} \subseteq \cdots \subseteq T_{s,0}$$

and $f_0, \ldots, f_s$. We must define $T_{s,1}, T_{s,2}, \ldots, T_{s,i_s}, T_{s+1,0}$ and f_{s+1}. In the process, we will also define the $g_{s,t}$.

Step 1: $\mathcal{A}_{s,1} \supseteq \mathcal{A}_{s,0}$, and they have the same domains, so we can define $T_{s,1} \supseteq T_{s,0}$ with $g_{s,1} = g_{s,0} = g_s$ still a pseudo-isomorphism $T_{s,1} \to \mathcal{A}_{s,1}$.

Step 2: For each $\sigma \in T_{s,1}$, define $g_{s,2}(\sigma)$ as follows. If $g_s(\sigma)$ is inactive or killed, then $g_{s,2}(\sigma) = g_s(\sigma)$. If, among $g_s(\tau)$, for $\tau \prec \sigma$, there are less than n active elements, then also set $g_{s,2}(\sigma) = g_s(\sigma)$. Finally, if there are n active elements among $g_s(\tau)$, for $\tau \prec \sigma$, then set $g_{s,2}(\sigma)$ as follows: if $g_s(\sigma) = c$, then $g_{s,2}(\sigma) = c'_s$; if $g_s(\sigma) = a_i$, then $g_{s,2}(\sigma) = a'_{i,s}$.

We can see from the construction of $\mathcal{A}_{s,2}$ that this is a pseudoembedding. Now extend $T_{s,1}$ to $T_{s,2}$ and extend $g_{s,2}$ to a pseudoisomorphism $T_{s,2} \to \mathcal{A}_{s,2}$.

Step 3: If $s \preceq s+1$, then this is the last step of stage 3 and $\mathcal{A}_{s,3} = \mathcal{A}_{s,2}$. So set $g_{s,3} = g_{s,2}$.

If $s \not\preceq s+1$, let t be maximal with $n(t) = m_{s,3}$. Then we have $m_{s,3} = n(t) < n(t+1) \leq n(s)$ and $t \preceq s$. We claim that $g_t \colon T_{t,0} \subseteq T_{s,2} \to \mathcal{A}_{s,3}$ is still a pseudoembedding. Consider $\sigma \in T_{t,0}$. Then by (†), $\overline{f}_{t+1}(\sigma)$ has at most $n(t)+1 = n(t+1)$ active elements. By (∗), $f_s(\sigma) = f_{t+1}(\sigma)$; also, $g_{s,2}(\sigma) = f_s(\sigma)$ because $\overline{f}_s(\sigma)$ has at most $n(t) < n(s)$ active elements. So f_{t+1} is a pseudoembedding $T_{t+1,0} \to \mathcal{A}_{s,3}$.

We want to argue that f_t is also a pseudoembedding $T_{t,0} \to \mathcal{A}_{s,3}$. Given $\sigma \in T_{t,0}$, since $t \preceq t+1$ the only change we made from f_t to f_{t+1} is that sometimes when $f_t(\sigma) = c$ we set $f_{t+1}(\sigma) = c'_t = a_{n(t+1)}$, and sometimes when $f_t(\sigma) = a_i$ we set $f_t(\sigma) = a'_{i,t}$. But in Case 2 of step 3 at stage $s+1$, we give c all of the labels c'_t had and vice versa, and similarly for a_i and $a'_{i,t}$. So f_t is a pseudoembedding $T_{t,0} \to \mathcal{A}_{s,3}$.

Define $T_{s,3}$ and $g_{s,3} \colon T_{s,3} \to \mathcal{A}_{s,3}$ such that $g_{s,3} \supseteq g_t$ is a pseudoisomorphism.

Step t: This is similar to step 3.

Verification. We must now check (∗) and (†). For (∗), if $s \preceq s+1 = t$, then this is clear from the construction—step 2 above is the only place where f_{s+1} is made to differ from f_s. Otherwise, if $t < s$ is maximal such that $n(t) = n(s)$, then in step 3 / step t we define f_{s+1} be having it extend f_t. The other cases of (∗) follow from these two cases.

For (†), if $s \preceq s+1$ then we do nothing in step 3. In step 2, it is not hard to see that if $\sigma \in T_{s,0}$ then $\overline{f}_{s+1}(\sigma)$ can have at most $n(s)+1 = n(s+1)$ active elements, as for $\tau \preceq \sigma$, $f_{s+1}(\tau)$ can only be active if $f_s(\tau)$ was, and there are only $n(s)+1$ active elements of $\mathcal{A}_s$. □

6.4 Conclusion of the proof

We have built $\mathcal{A}_n$ (which, by abuse of notation, we were calling $\mathcal{A}$). By Lemma 6.7 is not isomorphic to $\mathcal{B}$, the structure with domain R_n in the nth (possible partial) computable structure. Moreover, by Lemma 6.8, $\mathcal{T}(\mathcal{A}_n)$ has a computable copy uniformly in n.

Then let $\mathcal{A}$ be the structure which is the disjoint union of the $\mathcal{A}_n$, each of which satisfies the unary relation R_n. Then $\mathcal{A}$ has no computable presentation. By Lemma 4.7, $\mathcal{T}(\mathcal{A})$ has a computable copy.

7 More on Enumerations of Families

The second author showed that whenever a structure codes a family of sets, the family of sets can be recovered from the existential types of certain tuples, but the set of tuples to look at is only Σ_3-definable.

Let $\{\Theta_e : e \in \omega\}$ be the standard computable enumeration of all enumeration-operators.

Lemma 7.1 (Lemma 3.5 of [Mon13b])**.** *Let $\mathcal{A}$ be a structure. If $\mathcal{F}$ is a family of subsets of ω which is enumerated by all copies of $\mathcal{A}$, then there is a tuple $\bar{a}$ and a uniformly computable list of Σ_3 formulas $\varphi_\ell(\bar{x}, \bar{y})$ such that*

$$\mathcal{F} = \{\Theta_\ell(\Sigma_1\text{-}tp_{\mathcal{A}}(\bar{a}, \bar{b})) : l \in \omega,\ \bar{b} \in A^{<\omega},\ \mathcal{A} \models \varphi_\ell(\bar{a}, \bar{b})\}.$$

If the family is simple enough—like the Slaman-Wehner family, where all of the sets in the family are c.e.—then the Σ_3 formulas are already complicated enough to code the family without having to actually look at the structure at all. The Σ_3 formulas can just say to enumerate the sets given by certain operators Θ_ℓ which just enumerate a c.e. set in the family without looking at the Σ_1-type. On the other hand, the Σ_3 formulas cannot be improved to Σ_1 formulas.

Corollary 1.3. *There is a computable structure $\mathcal{A}$ and a family $\mathcal{F}$ of subsets of ω which is enumerated by all copies of $\mathcal{A}$ for which there do not exist a tuple $\bar{a} \in A^{<\omega}$ and a uniformly computable list of Σ_1 formulas $\varphi_\ell(\bar{x}, \bar{y})$ such that*

$$\mathcal{F} = \{\Theta_\ell(\Sigma_1\text{-}tp_{\mathcal{A}}(\bar{a}, \bar{b})) : l \in \omega,\ \bar{b} \in A^{<\omega},\ \mathcal{A} \models \varphi_\ell(\bar{a}, \bar{b})\}.$$

Proof. Let $\mathcal{A}$ be the structure from Theorem 6.1. Let $\mathcal{F}$ be the Slaman-Wehner family which is enumerated by every non-computable degree. Then every copy of $\mathcal{A}$ is non-computable, and hence enumerates $\mathcal{F}$.

Suppose that there were $\bar{a} \in A^{<\omega}$ and a uniformly computable list of Σ_1 formulas $\varphi_\ell(\bar{x}, \bar{y})$ such that

$$\mathcal{F} = \{\Theta_\ell(\Sigma_1\text{-tp}_{\mathcal{A}}(\bar{a}, \bar{b})) : l \in \omega,\ \bar{b} \in A^{<\omega},\ \mathcal{A} \models \varphi_\ell(\bar{a}, \bar{b})\}.$$

Then every copy of $\mathcal{T}(\mathcal{A})$ could enumerate $\mathcal{F}$. But $\mathcal{T}(\mathcal{A})$ has a computable copy, a contradiction. □

8 Higher Up

We can use Marker extensions to obtain Theorem 1.2 from Theorem 6.1.

Theorem 1.2. *For any computable ordinal α, there is a structure $\mathcal{A}$ such that $\mathcal{T}(\mathcal{A})$ has a computable copy but $\mathcal{A}$ itself has no Δ^0_α copy.*

Proof. We may assume that α is a successor ordinal. Relativize Theorem 6.1 to $0^{(\alpha)}$ to obtain a structure $\mathcal{A}$ be such that $\mathcal{T}(\mathcal{A})$ has a $0^{(\alpha)}$-computable copy but $\mathcal{A}$ has no $0^{(\alpha)}$-computable copy. Let $\mathcal{B}$ be the Δ_α-Marker extension of $\mathcal{A}$. Then by Lemma 5.2 $\mathcal{T}(\mathcal{B})$ has a computable copy, but by Lemma 5.1 $\mathcal{B}$ has no $0^{(\alpha)}$-computable copy. □

References

[AK00] C. J. Ash and J. Knight. *Computable structures and the hyperarithmetical hierarchy*, volume 144 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 2000.

[Bad77] S. A. Badaev. Computable enumerations of families of general recursive functions. *Algebra i Logika*, 16(2):129–148, 249, 1977.

[Ers96] Y. L. Ershov. *Definability and computability*. Siberian School of Algebra and Logic. Consultants Bureau, New York, 1996.

[FS89] Harvey Friedman and Lee Stanley. A Borel reducibility theory for classes of countable structures. *J. Symbolic Logic*, 54(3):894–914, 1989.

[Gon77] S. S. Goncharov. The number of nonautoequivalent constructivizations. *Algebra i Logika*, 16(3):257–282, 377, 1977.

[Gon80] S. S. Goncharov. Problem of the number of non-self-equivalent constructivizations. *Algebra and Logic*, 19(6):401–414, 1980.

[Kni86] Julia F. Knight. Degrees coded in jumps of orderings. *J. Symbolic Logic*, 51(4):1034–1042, 1986.

[Mon13a] A. Montalbán. A fixed point for the jump operator on structures. *Journal of Symbolic Logic*, 78(2):425–438, 2013.

[Mon13b] Antonio Montalbán. A fixed point for the jump operator on structures. *J. Symbolic Logic*, 78(2):425–438, 2013.

[Mon14] A. Montalbán. Computability theoretic classifications for classes of structures. In S. Y. Jang, Y. R. Kim, D.-W. Lee, and I. Yie, editors, *Proceedings of the International Congress of Mathematicians (Seoul 2014). Vol. II*, pages 79–101. Kyung Moon Sa Co., Seoul, 2014.

[Sel76] V. L. Selivanov. The numerations of families of general recursive functions. *Algebra i Logika*, 15(2):205–226, 246, 1976.

[Sla98] Theodore A. Slaman. Relative to any nonrecursive set. *Proc. Amer. Math. Soc.*, 126(7):2117–2122, 1998.

[Weh98] Stephan Wehner. Enumerations, countable structures and Turing degrees. *Proc. Amer. Math. Soc.*, 126(7):2131–2139, 1998.