A FRIEDBERG ENUMERATION OF EQUIVALENCE STRUCTURES

RODNEY G. DOWNEY, ALEXANDER G. MELNIKOV, AND KENG MENG NG

Abstract. We solve a problem posed by Goncharov and Knight [Problem 4 in [GK02]]. More specifically, we produce an effective Friedberg (i.e., injective) enumeration of computable equivalence structures, up to isomorphism. We also prove that there exists an effective Friedberg enumeration of all isomorphism types of infinite computable equivalence structures.

1. Introduction

A large part of classical algebra seeks to characterise structures up to isomorphism. To do this, algebraists often develop elaborate theories of invariants. Effective algebra [AK00, EG00] looks at *algorithmically presented* structures, and therefore it is natural to look at these structures using a certain kind of effective classification tool. A typical method would be to use *computable* isomorphisms or some other kind of *effective* isomorphisms. If we use computable isomorphisms, then typically the classical isomorphism type splits into infinitely many computable isomorphism types [EG00]. There is a large body of work seeking to relate syntax to the complexity of the isomorphism types (see for example, Ash and Knight [AK00] for more on this program).

A more general question is the following: "Can we measure the complexity of a *classification problem* in effective algebra?" By a classification problem we mean the following. Suppose $\mathcal{K}$ is a class of "effectively presented" mathematical objects, such as recursively presented groups [Hig61] or computable Banach spaces [PER89]. Such a class $\mathcal{K}$ usually comes with a natural notion of isomorphism. The *classification problem for the class* $\mathcal{K}$ asks:

Can we associate each structure in $\mathcal{K}$ with an invariant so that deciding isomorphism on $\mathcal{K}$ becomes simpler than "brute-force"?

The problem is quite general with many terms being vaguely defined. As a consequence, there have been many non-equivalent ways to approach the problem (see [GK02]). In their seminal paper, Goncharov and Knight [GK02] put together all standard approaches (to date) that are typically used to attack classification problems in effective algebra (see also [FF12]). One such approach is to measure the complexity of various index sets [GK02], or compare classes under effective reductions [FF12]. Such general methods are readily applied to various common algebraic classes such as groups, integral domains, lattices, etc., see [GK02, HKSS02, DM08, FF12]. In fact, often such applications are not restricted to computable members of the classes and can be interpreted as classification results in the most general algebraic sense (see e.g. [DM08, DM13, DM14]).

Goncharov and Knight [GK02] discuss another standard approach to understanding effective classes and their classification. That is: How hard is it to *list* the isomorphism types? In particular, for a class $\mathcal{K}$, is there a uniformly effective list $K_0, K_1, \dots$ in which every isomorphism type from $\mathcal{K}$ is mentioned *exactly once*. Then the position n of K_n in the list fully

Date: July 12, 2017.

The first two authors are partially supported by the Marsden Fund of New Zealand. The third author is partially supported by the grants MOE-RG26/13 and MOE2015-T2-2-055.

describes its isomorphism type. Such an enumeration is called a *Friedberg enumeration.* (Recall that Fridberg [Fri58] proved that there exists a computable one-one enumeration of all c.e. sets without repetition.) If a class has such a Friedberg enumeration, then we can regard it as "classified" in this well-defined sense, in spite of the actual isomorphism problem (viewed for example, as an index set) being perhaps quite complex.

Goncharov and Knight [GK02] gave a general condition on a class which implies the class does not possess a Friedberg enumeration. One model-theoretically simple set of structures not covered by this metatheorem was *equivalence structures.* Because of this, Goncharov and Knight [GK02] (Problem 4) asked whether there exists a Friedberg enumeration of computable equivalence structures. This problem turned out to be rather difficult to resolve.

What is so peculiar about computable equivalence structures? Why did the problem resist solution? See Goncharov and Knight [GK02] and Lange, Miller and Steiner [LSM] for earlier attempts. Why does it seem necessary to use complex priority techniques to attack this problem? First, we shall address these questions, and then we will discuss our positive solution to the problem and its consequences.

1.1. **A simple class with hard properties.**

At first glance computable equivalence structures seem rather primitive. Indeed, countable equivalence structures are elementary from the classical point of view. However, this seemingly elementary class possesses several remarkably deep *effective* properties. This is due to its relationship with the closely related notion of a limitwise monotonic set. Limitwise monotonic sets have proved to be highly useful in different seemingly distant parts of recursion theory [KKM13, KNS97, Har08, DKT11]. Recall that a set S is called *limitwise monotonic* if it is the range ($\subseteq N \cup \{\infty\}$) of a function g, where $g(n) = \sup_s f(n,s)$ for a computable f. For example the sizes of equivalence classes in a computable equivalence structure are exactly the limitwise monotonic sets. This notion was introduced by Khisamiev [Khi98] in the context of abelian groups. Equivalence structures allow us to study the notion in a more computability theoretic setting, where we do not have to worry about the more complex group structure. Nonetheless, even in absence of any algebraic structure, the property of limitwise monotonicity is still not very well understood from the pure computability theoretic point of view (see [DKT11, KKM13]).

Being a computability-theoretic abstraction rather than an actual interesting structure on its own, the class of computable equivalence structures resembles several important standard classes (either locally or globally) that possess "full decompositions", especially abelian p-groups [Fuc70, Khi98], completely decomposable groups [Bae37, DM13], and special classes of linear orders (such as η-presentations and shuffle sums [Dow98]). There are long-standing open problems concerning computable presentations in each of these natural classes, and one would hope that a better understanding of equivalence structures might help in approaching these problems. Indeed, in each of the listed classes combinatorial methods tend to resemble those used in computable equivalence structures. Some of these methods have already found applications in these classes [Har08, KKM13, DMNa, DMNb]. Also, recently there has been an increasing interest in various effective properties of computably enumerable equivalent relations (CEERs). See e.g. Lachlan [Lac87], and see also [ALM$^+$14] for a recent survey on CEERs. Although we will not discuss CEERs further, we note that these results are very closely related to numbering theory [Ers] that has been particularly popular in the Russian/USSR recursion theory tradition.

The class of computable equivalence structures suggests some remarkably challenging problems, we briefly discuss one that has captured our imagination in the recent past. One can easily classify computable equivalence structures that possess a unique computable presentation up to computable isomorphism [CCHM06]. Also, $\emptyset''$ can see whether two computable equivalence structures are isomorphic. What about those which have a unique computable copy up to $\emptyset'$-isomorphism [CCHM06]? Remarkably, the question is still open. In fact, the most recent result towards this problem [DMN15] uses a nonstandard $\emptyset'''$-construction of a combinatorial complexity rarely seen in effective structure theory (let alone computable equivalence relations). The result illustrates how little we understand Δ^0_2 isomorphisms in general. On the other hand, we applied our techniques from [DMN15] to solve two open problems on computable groups (see [DMNa]).

The reader might wonder as to the source of the computability-theoretic complexity. One of the core difficulties in working with equivalence structures is the complexity of the isomorphism relation in the class. Let $E_0, E_1, \dots$ be an effective listing of all computable equivalence structures (allowing repetition). Although seemingly simple, the set $\{i : E_i \cong R\}$ may be Π^0_4-complete for a fixed computable equivalence structure R [CCHM06]. This means that guessing whether two computable presentations are isomorphic is in general Π^0_4. This contrasts with the situation in, say, c.e. sets where equality is merely Π^0_2. Similarly, for effectively closed sets the upper bound is Π^0_2 which makes it possible to produce their Friedberg enumeration [BC08] without resorting to complex priority techniques.

1.2. **A listing without repetition.**

It is clear that a Friedberg enumeration exists in each of the following classes: vector spaces over a given field, algebraically closed fields of a given characteristic, and well-orderings of a type less than any fixed computable ordinal [GK02]. Deeper results on Friedberg enumerations typically require significantly new ideas. As noted in [GK02], such proofs are often indirect and tend to use some global properties of the class rather than a construction. As a consequence, not much is known about Friedberg enumerations for interesting classes. Goncharov and Knight [GK02] isolated a general condition that implies that the classes of computable Boolean algebras, linear orders, and abelian p-groups have no Friedberg enumeration (indeed, not even a hyperarithmetical Friedbeg enumeration). It is also known there is no Friedberg enumeration of the class of torsion-free abelian groups of rank 1 [LSM], but the class of computable algebraic fields possesses such an enumeration [LSM]. See [GK02, LSM] for more results on 1-1 enumerations.

Goncharov and Knight [GK02] showed that the class of all computable equivalence structures with infinitely many infinite classes has a Friedberg enumeration, and noted that perhaps there is no Friedberg enumeration of the class of all computable equivalence structures since the natural invariants for this class look too complex (see [GK02], discussion after Proposition 5.4.) More recently, Lange, R.Miller and Steiner [LSM] showed that there exists a $\emptyset'$-computable Friedberg enumeration of all isomorphism types of computable equivalence structures. However, the general question remained open. At this point the reader should be convinced that the question of Goncharov and Knight looks quite challenging, especially given the Π^0_4-completeness of isomorphism. Nonetheless, we prove:

Theorem 1.1. *There exists a Friedberg enumeration of the class of computable equivalence structures, up to isomorphism.*

We point out that the enumeration provided by Theorem 1.1 includes the isomorphism type of *all* computable equivalence structures, including the finite equivalence structures. More specifically, in Theorem 1.1, we prove that there exists a computable sequence $(U_i, E_i)_{i\in\omega}$ such that for every i, U_i is a c.e. initial segment of ω, and E_i is a partial computable binary equivalence relation defined on all members of U_i, such that every computable equivalence structure is isomorphic to (U_i, E_i) for some i, and that for every $i \neq j$, $(U_i, E_i) \not\cong (U_j, E_j)$. We also remark that Theorem 1.1 does not claim to know for which i is U_i finite. The question of whether we can enumerate all computable equivalence structures, while presenting the finite ones by their canonical indices will be answered by our next main theorem, Theorem 1.2.

The reader should prepare for a combinatorially involved proof that is contained in Section 2. The good news is that we have to do anything significant *only if* the i^{th} structure E_i in the natural listing with repetitions is not isomorphic to E_k, for any $k < j$, which can be detected in a Σ^0_4 way. The usual priority tree techniques can in principle handle Σ^0_4-guessing. Although our proof is not quite in the same league of complexity as the main result of [DMN15] mentioned above, there are still interesting technical features in this proof. For instance, the level of complexity we have to work with means that a single requirement will be spread over infinitely many nodes of the priority tree, where each such node has to be assigned a Π^0_3-outcome. As we have already mentioned above, the Π^0_4-completeness of isomorphism means that the complexity of our guessing procedure (provably) cannot be simplified. The earlier partial results ([GK02, LSM]) helped us to understand the general case, but the fact that the techniques used by those authors were (quantifiably) simpler perhaps explains why the general question has resisted attack for so long. Indeed, these results also support our strong conjecture that the relatively high combinatorial complexity of our proof is unavoidable.

1.3. **Enumerating only infinite structures.**

There has been a tendency among our colleagues to exclude finite structures from consideration in computable structure theory. Indeed, perhaps one should not expect to get deep results on, say, finite groups using methods of effective algebra. So the reader may ask whether there exists a Friedberg enumeration of *infinite* computable equivalence structures. It may seem at the first reading that the proof of Theorem 1.1 heavily relies on finite structures, and without finite structures the result perhaps fails. Nonetheless, the theorem is still true if we exclude finite structures:

Theorem 1.2. *There exists a Friedberg enumeration of the class of computable infinite equivalence structures, up to isomorphism.*

More specifically, we prove the existence of a computable sequence of equivalence relations $(E_i)_{i\in\omega}$ such that any infinite computable equivalence structure is $\cong (\omega, E_i)$ for some i, and that for every $i \neq j$, $(\omega, E_i) \ncong (\omega, E_j)$. An immediate corollary is a strengthening of our first main theorem; recall that Theorem 1.1 provides a Friedberg enumeration of all computable equivalence structures, without specifying which structures are infinite. Since it is clear that there is a Friedberg enumeration of the class of *finite* equivalence structures up to isomorphism, we can append it to the enumeration of Theorem 1.2 to obtain:

Corollary 1.3. *There is a Friedberg enumeration of the class of all computable equivalence structures, up to isomorphism. Furthermore, we can tell which members of this enumeration are finite and which are infinite, and effectively obtain the canonical indices for the atomic diagram of the finite equivalence structures.*

The proof of the second main result is contained in Section 3. It uses Theorem 1.1 and essentially goes through defining an effective injective functor from a special (large enough) class of computable equivalence structures to the class of all infinite ones. Alternatively, in the proof of Theorem 1.1 we could uniformly replace finite structures by their respective images under the functor, and it would not introduce too much extra tension (we omit details). So it is not necessary to have finite structures either in the main result itself or in its proof.

Before we get to the formal proofs, we note that the main result of the paper readily implies the existence of a Friedberg enumeration of computable abelian p-groups of Ulm type 1, in contrast with the results of [GK02] mentioned above. We suspect that our techniques ([DMNb]) may allow us to produce Friedberg enumerations of larger natural classes of abelian p-groups (i.e., of a finite Ulm type), but we have yet to investigate this.

2. Proof of Theorem 1.1

2.1. **Preliminary analysis.** Our goal is to prove that there exists a Friedberg enumeration of the isomorphism type of computable equivalence structures. More specifically, we shall prove that there exists a computable sequence $(U_i, E_i)_{i \in \omega}$ such that for every i, U_i is a c.e. initial segment of ω, and E_i is a partial computable binary equivalence relation defined on all members of U_i, such that every computable equivalence structure is isomorphic to (U_i, E_i) for some i, and that for every $i \neq j$, $(U_i, E_i) \ncong (U_j, E_j)$.

2.1.1. *Notation and conventions.* First of all, note that any structure $\left(U, \hat{E}\right)$ where U is a c.e. initial segment of ω, and $\hat{E}$ is a partial computable binary equivalence relation defined on all members of U can be effectively identified with the structure $\bigoplus_k \llbracket card\, V^{[k]} \rrbracket$, where $V \subseteq \omega$ is a c.e. set. Here $S^{[k]}$ stands for the k^{th} column of a set $S \subseteq \omega$, and $\llbracket \alpha \rrbracket$ is an equivalence class of size α. Obviously this definition is specially catered to allow for finite structures. It is also clear that we can effectively pass from an index for $\left(U, \hat{E}\right)$ to an index for the corresponding c.e. set V, and vice versa, as well as the index for a (partial) computable isomorphism between the two structures. For our purposes there is no difference between these two approaches; we will adopt whichever presentation that is more convenient.

Now using the approach above we fix the standard effective listing $(E_n)_{n \in \omega}$ of all computable equivalence structures, allowing repetitions. More specifically take $E_n = \bigoplus_k \left\llbracket card\, W_n^{[k]} \right\rrbracket$, where W_n is the n^{th} c.e. set. Our goal is to produce using this sequence, another sequence $(U_n)_{n \in \omega}$ in which isomorphism types are not repeated.

We will use the following notation thoughout the paper. If E is a computable equivalence structure, then $\mathbf{e_j}$ is its j^{th} equivalence class, or equivalently the j^{th} column under the enumeration as above. Since we are interested in isomorphism types, only sizes of the $\mathbf{e_j}$ will matter.

2.1.2. *Some elementary simplifications.* We will divide equivalence structures into five types:

(I) Finite equivalence structures.
(II) Finitely many finite classes and finitely many (with at least one) infinite classes.
(III) Finitely many finite classes and infinitely many infinite classes.
(IV) Infinitely many finite classes which are uniformly bounded in size, and any number of infinite classes.
(V) Infinitely many finite classes with arbitrarily large finite sizes, and any number of infinite classes.

It is clear that any equivalence structure belongs to exactly one of the five types. Note that each isomorphism type in (I), (II), (III) and (IV) is specified by a finite parameter, and it is not difficult to see that the parameters (thus, the respective isomorphism type) satisfying the desired properties can be effectively listed. Thus there exist computable Friedberg enumerations of the isomorphism types in each group (I) to (IV).

The only complicated group is (V), where its members cannot be specified by a finite parameter. We aim to construct a Friedberg enumeration of this class. However due to technical reasons, we shall also allow structures of type (I) and (III) to be enumerated along with (V). The purpose of this is to utilize structures of type (I) and (III) as "junk collectors"; any structure we build during the construction to represent an isomorphism type in (V) may over the course of the construction become abandoned, and in this case we will turn the partially built structure into one of type (I) or (III).

Therefore, our construction will construct a Friedberg enumeration of the isomorphism types in (I), (III) and (V). We can then produce the desired Friedberg enumeration of all possible isomorphism types of computable equivalence structures by adjoining the missing isomorphism types (II) and (IV) to the list.

It will also be convenient to assume that in our list all finite structures are actually non-empty. This assumption does not affect the construction significantly. (We may then simply adjoin the empty structure to our enumeration if necessary).

2.2. **The setup.**

From now on, we will refer to a structure of type (V) as *unbounded*. Recall $(E_n)_{n\in\omega}$ is the effective listing of all computable equivalence structures induced by the standard enumeration of c.e. sets. Note that guessing whether $E_n \cong E_m$ is Π^0_4 in general, and it has been shown by Calvert, Cenzer, Harizanov and Morozov [Theorem 3.13 in [CCHM06]] that the problem can be Π^0_4-complete for a single fixed E_n, where E_n is, and in fact must be unbounded. Thus, in the construction we will have to deal with a Σ^0_4-guessing procedure.

2.2.1. *Requirements and further preliminary remarks.*

In the construction, each computable equivalence structure E_i will have infinitely many strategies τ associated with it. The task of each τ is to produce a computable isomorphic copy $U_\tau \cong E_i$ unless E_i is bounded (not of type (V)) or E_i is isomorphic to some E_j with $j < i$. Thus, each such τ (associated with E_i and building U_τ) works towards meeting the requirement:

$$\bigwedge_{j<i} E_i \ncong E_j \text{ and } E_i \text{ unbounded} \implies U_\tau \cong E_i,$$

or, in other words, makes sure that E_i appears in the list that we construct, and that there are no obvious conflicts with higher priority strategies trying to copy other unbounded structures.

Since the guessing procedure for isomorphism is naturally Σ^0_4, there will be *infinitely many strategies* τ associated with E_i. More specifically, recall that for every i it is Π^0_4 to tell if $E_i \cong E_j$ for some $j < i$. Let this predicate be $\forall k P(i,k)$, where $P(i,k)$ is Σ^0_3. For each k we will have a separate strategy τ guessing whether $P(i,k)$ holds. It is crucial that τ has to produce an isomorphic copy of E_i *only if* it sees that $P(i,k)$ fails, which is a Π^0_3 event for the given i, k.

We will see that the different τs will collectively be able to take care of the *global requirement*, saying that the enumeration must be Friedberg, in particular, that $\forall \tau, \sigma\ [\tau \neq \sigma \rightarrow U_\tau \ncong U_\sigma]$. We will also see that, depending on its true outcome, τ may produce a copy of the associated E_i, as well as several (perhaps, infinitely many) distinct finite structures, or a single structure of type (III); in the latter two cases the structures will be placed into the *junk collector* which will be operating outside the tree of strategies. The junk collector will respect the global Friedberg requirement and will also take care of the *global onto requirement* saying that all isomorphism types structures of the prescribed 'junk' types must eventually appear in the global list that we construct. In fact, the junk collector will be further subdivided into *finite junk collector* and *infinite junk collector*, with infinite junk potentially transformable into finite junk.

2.2.2. *Priority Tree.*

As we noted above, for every i it is Π^0_4 to tell if $E_i \cong E_j$ for some $j < i$, and recall $P(i,k)$ is the Σ^0_3-predicate such that $\forall k P(i,k)$ holds iff $E_i \cong E_j$ for some $j < i$. Each node on the tree will be assigned the task of guessing $P(i,k)$ for some i and k, where i stands for the computable equivalence structure E_i. This is assigned the usual way; a node τ is assigned $P(i,k)$ where $|\tau| = \langle i,k\rangle$. For convenience we write E_τ for the equivalence structure associated with a strategy τ, and we write k_τ for the respective integer k.

The priority tree is a $1+\omega+2$-branching tree. The outcomes ordered from left to right are:

$$\mathtt{init} < \mathtt{pi}_2 0 < \mathtt{pi}_2 1 < \cdots < \mathtt{pi}_3 < \mathtt{wait}$$

The leftmost outcome `init` is a Π^0_2 outcome, the $\mathtt{pi}_2 m$ are each Π^0_2 outcomes that collectively form the Σ^0_3 counterpart of $\mathtt{pi}_3$ which is the Π^0_3 outcome. The waiting outcome `wait` is Σ^0_2.

We will formally describe the role of each outcome later. At this point we only note that the Π^0_2-outcome `init` corresponds to the situation when E_τ has too few finite classes; in this case we will initialise τ infinitely often. The wait outcome guesses that E_τ is bounded, i.e. does not have large enough finite classes. The other outcomes will guess at the predicate P and some further properties needed to successfully copy E_τ. The $\mathtt{pi}_3$ outcome of τ corresponds to the fact that $P(i,k)$ fails; in this case τ has to copy E_τ into U_τ.

2.3. **A single node in isolation.** Fix τ on the construction tree. We now describe the actions and outcomes of τ, when considered in isolation. Assume τ is working for E_i and there are exactly $k_\tau - 1$ many nodes $\sigma \subset \tau$ such that σ works for E_i and $\sigma * \mathtt{pi}_2 j \subseteq \tau$ for some j. If $\rho * o \subseteq \tau$ for some ρ that works for the same E_i and some other outcome o we will simply assume τ is always inactive and does nothing when visited. The strategy τ will build a structure U_τ (we suppress τ and write U when the context is clear) which will be permanently abandoned when τ is initialised.

2.3.1. *Initialisation.* The strategy τ will be initialised if outcome `init` of τ is played. In isolation, this case corresponds to the scenario when E_τ has no finite classes (note that this can be detected in a Π^0_2 way), or if we move to the left of τ unless we are moving from a $\sigma * \mathtt{pi}_3$ to a $\sigma * \mathtt{pi}_2 k$ outcome for some $\sigma \subset \tau$. In this case the strategy abandons its current U_τ. If U_τ is abandoned then it joins the *finite junk collector* which will be processed by the finite junk collector strategy to ensure there are no repetitions. We then restart with a new U_τ. After each initialisation, all parameters of the strategy are set undefined. When τ is active again (if ever), it will set its U_τ equal to a structure with a single "large" finite class of size larger than any number seen so far in the construction.

2.3.2. *The parameter fin.* The node τ will also monitor the parameter $fin(\tau, j)$. When the context is clear we write $fin(j)$ instead. This parameter will be used to copy E_τ into U_τ and guess whether E_τ is of the unbounded isomorphism type. Given the node τ and a stage s, define the sequence $f_1 < f_2 < \cdots < f_t$ of length at most s inductively by the following. Suppose f_{k-1} has been defined (for $k = 1$ take $f_{k-1} = 0$). Take f_k to be the least such that $f_k > f_{k-1}$ and the class $\mathbf{e}_{\tau,\mathbf{f_k}}$ of E_τ currently has size larger than k and is furthermore the oldest class in E_τ with index larger than f_{k-1} and with size larger than k. The age of a class is the number of stages it has not increased in size; so the oldest class is the one which has not increased in size for the longest time. If every class with index larger than f_{k-1} has size at most k, then f_k is not defined at stage s.

The parameter $fin(j)[s]$ is defined to be equal to the sequence $f_1 < f_2 < \cdots < f_{\langle \tau, j\rangle}$, if all the terms exist at stage s. Otherwise we say that $fin(j)[s]\uparrow$. Note at every stage of the construction $fin(j)$ is a substring of $fin(j+1)$ (if they are both defined), and $fin(j)$ undefined implies that $fin(j+1)$ is undefined. We abuse our terminology and define the *range* of fin at stage s to be the string $f_1 < f_2 < \cdots < f_{\langle \tau, j\rangle}$ for j the largest such that $fin(j)[s]$ is defined.

We explain the use of fin. It is not hard to see that the property of being unbounded is Π^0_3. In fact, the parameter fin is meant to guess whether E_τ is unbounded: The property of E_τ being unbounded is equivalent to the Π^0_3 property that $fin(j)$ holds a stable value for each j. This can be naturally incorporated into the outcomes of τ. The Π^0_2 outcome $\mathtt{pi}_2 j$ stands for the fact that j is the least such that $fin(j)$ does not have a limit, while the outcome $\mathtt{pi}_3$ stands for the fact that $fin(j)$ is stable for every j.

The reason for using $\langle\tau, j\rangle$ instead of $\langle\tau\rangle$ or j is to keep the infinite junk structures produced by the different outcomes of different nodes non-isomorphic. When we copy E_τ into U_τ we make sure that if j is currently largest such that $f(j)$ has a stable definition, then U_τ would have copied somewhere between $\langle\tau, j\rangle$ and $2\langle\tau, j\rangle$ many finite classes of E_τ. As a consequence, the true outcome $\tau * \mathtt{pi}_2 j$ will produce an infinite junk structure with somewhere between $\langle\tau, j\rangle$ and $2\langle\tau, j\rangle$ many finite classes. If furthermore we had that if $\langle\tau, j\rangle < \langle\tau', j'\rangle$ then $2\langle\tau, j\rangle < \langle\tau', j'\rangle$, then in this case the infinite junk structures produced by different strategies or different outcomes will be necessarily non-isomorphic, as they would have a different number of finite classes. It would also guarantee that the "infinite junk collector" has plenty of room to satisfy the global requirements. We will use a modification of the pairing function: We replace the standard pairing function $\langle\tau, j\rangle$ with $3^{\langle\tau,j\rangle}$ throughout the rest of the proof.

2.3.3. *The isomorphism ℓ.* Let $\boldsymbol{e}_{\tau,i}$ and $\boldsymbol{u}_{\tau,i}$ be the i^{th} classes in E_τ and U_τ respectively. To assist us in organising the copying strategy, we will define a potentially Δ^0_2 isomorphism $\ell_\tau : U_\tau \to E_\tau$, which will be total only if $\tau * \mathtt{pi}_3$ is the true outcome of τ. (Note that the totality of a Δ^0_2 function is also Π^0_3). Strictly speaking, ℓ will be a function mapping indices to indices, and we identify $\boldsymbol{u}_i$ with $\boldsymbol{e}_{\ell(i)}$. At the beginning we set $\ell(i)\uparrow$ for all i. For convenience, whenever we wish to change the approximation to $\ell(i)$, we will first set $\ell(i)\uparrow$ before re-defining $\ell(i)$ at a later stage. The final limiting value of $\ell(i)$ is assigned the obvious meaning.

2.3.4. *Action of τ.* During the construction, whenever the node τ is visited, it will first check if there is a least (in the index) unmapped class $\boldsymbol{e}_{\tau,j} \leq$ the largest element in the range of fin (unless we are waiting in step (i)), and if it exists, introduce a new class $\boldsymbol{u}_{\tau,i}$ to match it. This means that we will grow a new $\boldsymbol{u}_{\tau,i}$ to be equal in size to $\boldsymbol{e}_{\tau,j}$ and define $\ell_\tau(i) = j$. If (and only if) a new class is introduced in U_τ, we will also grow all classes $\boldsymbol{u}_{\tau,i}$ for which $\ell_\tau(i)$ is defined to have size equal to $\boldsymbol{e}_{\tau,\ell(i)}$. Next, τ will process the following:

(i) For each nonempty class $\boldsymbol{u}_{\tau,i}$ such that $\ell_\tau(i)\uparrow$, we search for a corresponding $\boldsymbol{e}_{\tau,j}$ not yet mapped via ℓ_τ and with size larger or equal to the size of $\boldsymbol{u}_{\tau,i}$.
   - If $\ell_\tau(i)\downarrow$ for every class $\boldsymbol{u}_{\tau,i}$ in U_τ, go to step (ii).
   - If $\ell_\tau(i)\uparrow$ for some class $\boldsymbol{u}_{\tau,i}$ in U_τ, and a large enough unmapped class in E_τ does not yet exist, play outcome `wait` and go to the next stage.
   - Otherwise for every class $\boldsymbol{u}_{\tau,i}$ with $\ell_\tau(i)\uparrow$ is able to find some image, we take the following actions, for each i. Define $\ell_\tau(i) = j$ (for the corresponding j). Play outcome `wait` and go to the next stage.

(ii) If we are here, it means that every nonempty class $\boldsymbol{u}_{\tau,i}$ has been mapped (i.e. $\ell_\tau(i)\downarrow$). Wait for $dom(fin)$ to increase beyond the previous maximum. If this is the first stage where $dom(fin)$ is longer than any previous stage since the last τ-initialisation, we take outcome $\mathtt{pi}_3$ for this stage and go to (iii). Otherwise, play outcome `wait` at this stage.

(iii) Check if there is some $i < dom(fin)$ such that $fin(i)$ has changed, or the Π^0_2-predicate $Q(\tau, k_\tau, i)$ such that $P(\tau, k_\tau) = \exists i Q(\tau, k_\tau, i)$ has "fired". (A Π^0_2-predicate fires means that the Π^0_2 predicate looks true for one more stage in some computable approximation of the predicate; a Π^0_2-predicate holds if and only if it fires infinitely often). Without loss of generality, we assume that $Q(\tau, k_\tau, 0)$ never fires. Take i to be the least such, if it exists. Take the appropriate action below and go back to step (i).
   - If no such $i < dom(fin)$ exists, do nothing.
   - If $i = 0$, play outcome `init` and initialise τ.
   - If $i > 0$, we play outcome $\mathtt{pi}_2(i-1)$. Preserve each class $\boldsymbol{u}_{\tau,k}$ for all $k < i$ as well as each class $\boldsymbol{u}_{\tau,k}$ such that $\ell_\tau(k)$ is in the tuple $fin(i-1)$ or $\ell_\tau(k) < i$. All

other non-empty classes $\boldsymbol{u}_{\tau,m}$ we grow to size s (or some suitably large number, determined by the junk collector) and set $\ell_\tau(m)\uparrow$.

The strategy we describe here for τ assumes it acts in isolation. During the formal construction (Section 2.6) we will follow a slightly modified form of the strategy described here due to various technical reasons.

2.3.5. *Analysis of the outcomes of τ.* The true outcome of τ could be:

True outcome `wait`: Since we only play outcomes to the left of `wait` finitely often, it is easy to see that in this case we must get stuck waiting forever in (i) or (ii). In either case we will eventually stop adding new classes to U and stop growing existing classes of U. Thus we end up producing a *finite structure U*.

If we get stuck in (i) then as we eventually stop introducing new classes or grow existing ones, this means that almost every class in E_i has size bounded by some integer, so E_i is not of the unbounded isomorphism type. If we are stuck in (ii) then $fin(i)$ cannot get a suitable current definition, let alone a stable definition for some i. Again this means that almost every size in E_τ is bounded by $\langle\tau,i\rangle$.

True outcome `init`: By convention, $Q(\tau,k_\tau,0)$ never fires, so if `init` is the true outcome then $fin(0)$ fails to hold a stable definition. Hence there are fewer than $\langle\tau,0\rangle$ many finite classes of the different corresponding sizes in E_i, so again E_i is not unbounded. In this case we initialise τ infinitely often, and consequently the strategy produces infinitely many finite structures. We will ensure that these structures will all be non-isomorphic.

True outcome $\mathtt{pi}_2 i$: In this case either $Q(\tau,k_\tau,i+1)$ fires infinitely often or $fin(i+1)$ fails to hold a stable definition. In the former case, as $P(\tau,k_\tau)$ holds, we have more evidence that E_τ is isomorphic to some E of higher priority, so we should not allow τ to copy E_τ into our list. In the latter case there are fewer than $\langle\tau,i+1\rangle$ many finite classes of large enough finite sizes, so again E_i is not unbounded, so in this case we also do not want τ to copy E_τ into our list.

Note that `init` is played finitely often, hence $fin(i)$ must be eventually stable, so that there are at least $\langle\tau,i\rangle$ many finite classes of various sizes in E_τ. After $fin(i)$ is stable we will eventually put all these classes into the range of ℓ_τ. Furthermore ℓ_τ is always preserved on the classes that are mapped to classes in $fin(i)$, on the classes that are mapped to the first i many classes of E_τ, as well as on the first i many classes of U_τ. All other classes in U_τ are increased in size each time we visit outcome $\mathtt{pi}_2 i$. Thus U_τ will be an infinite junk structure with between $\langle\tau,i\rangle$ and $2i+\langle\tau,i\rangle<2\langle\tau,i\rangle$ many finite classes.

True outcome $\mathtt{pi}_3$: This means that all other outcomes to the left of $\mathtt{pi}_3$ are each visited finitely often. Since outcome $\mathtt{pi}_3$ is played infinitely often, this means that we are never forever stuck in (i) or (ii). In fact, $dom(fin)$ grows arbitrarily large. Therefore, if $Q(\tau,k_\tau,i)$ fires infinitely often for some least $i>0$, τ will not be denied the chance to infinitely often play outcome $\mathtt{pi}_2(i-1)$. This implies that $P(\tau,k_\tau)$ does not hold, and every $fin(i)$ will hold a final stable definition. This means that E_τ is not isomorphic to any higher priority E, and E_τ is unbounded, and hence we should let τ copy E_τ into our list. We will argue in the verification that ℓ_τ is eventually total and stable at every input and onto, and consequently witnesses that $U_\tau\cong E_\tau$.

The reader might now wonder if it is necessary to have the outcome `init`; after all, could we not treat the outcome `init` as being part of the sequence of $\mathtt{pi}_2 i$ outcomes? For example, in the case where (outcome `init` is true and) there are fewer than $\langle\tau,0\rangle$ many finite classes in E_τ, could we not simply make U_τ contain exactly, say, τ many finite classes, and avoid doing the self-initialisation? The problem is that we cannot control exactly how many finite classes E_τ contains; it could very well be that E_τ contains *no* finite classes at all. In that case, when we define U_τ, no matter which classes of U_τ copy which classes of E_τ, we are going to have to make $U_\tau \cong I$. If more than one node on the tree does this, then we will be forced to introduce repetitions. Hence, the safe solution is to self-initialise whenever we detect this possibility, and hence the necessity of outcome `init`.

2.4. **Coordination between different τ.** As the outcomes of the requirements are of order-type $1 + \omega + 2$, the true path of the construction will be $\emptyset'''$-computable, rather than being the usual $\emptyset''$-computable. Since it is now possible to visit left of the true path infinitely often, we need to describe the effect each node has on the strategies on its right. If τ plays outcome `init` then all nodes extending $\tau * o$ for an outcome $o \neq$ `init` are initialised. If τ plays outcome $\mathtt{pi}_3$ then all nodes extending $\tau * \mathtt{wait}$ are initialised.

Now suppose τ plays outcome $\tau * \mathtt{pi}_2 i$. We will initialise every node extending $\tau * \mathtt{wait}$ or $\tau * \mathtt{pi}_2 j$ for $j > i$. However we clearly do not wish to initialise a node $\sigma \supseteq \tau * \mathtt{pi}_3$, since σ could be on the true path. We ensure that the next time we visit σ again we will *force* σ to play outcome $\mathtt{pi}_2 i$ (and take the corresponding actions) at least once, even though the basic strategy for σ does not require it to do so. This ensures that if $\tau * \mathtt{pi}_2 i$ is on the true path, then every $\sigma \supseteq \tau * \mathtt{pi}_3$ produces an infinite junk, unless σ is initialised infinitely often due to other reasons, and therefore does not copy any E into our list. On the other hand if σ is on the true path, then for each i, σ is forced to play the outcome $\mathtt{pi}_2 i$ in this way only finitely often, and the true outcome of σ will still correctly reflect the outcomes of its basic strategy.

2.5. **Coordination with junk collectors.** We will now explain how we handle the "junk" arising from the construction. These are structures built by the strategies during the construction which are not of the intended (unbounded) type. We call finite structures which are built by some node τ (or the $INFJUNK$ strategy) but which later get permanently abandoned due to initialisation *finite junk*. Structures of type (III) which are built by some node τ, but which are never abandoned by τ are known as *infinite junk*.

In this subsection we also explain several important modifications to the basic strategy of τ; the modifications are necessary for understanding the rest of the proof. The finite junk collector $FJUNK$ and the infinite junk collector $INFJUNK$ are strategies that act outside of the priority tree, and will get to act at the end of every stage. Since the nodes on the construction tree will produce finite and infinite junk structures, the $FJUNK$ and $INFJUNK$ strategies are there to ensure that all isomorphism types of type (I) and (III) are listed. They will do so by adding to our list the missing structures of types (I) and (III) which are not produced as junk by nodes on the priority tree.

We split these actions into infinitely many substrategies, $FJUNK(F)$ and $INFJUNK(F)$ indexed by the isomorphism type F of a finite equivalence structure, and these substrategies will seek to place a structure of isomorphism type F or $F \oplus I$ respectively, where I is the structure having infinitely many infinite classes.

2.5.1. *Description of $FJUNK(F)$.* Initially, when $FJUNK(F)$ is active for the first time, it checks whether there already exists a finite structure of isomorphism type F in the construction that is either permanently abandoned due to initialisation by some strategy on the tree, or corresponds to outcome `wait` of some strategy and thus may (or may not) be truly abandoned. If none of the above possibilities occur, it introduces a new finite structure of type F. We say that this structure is the witness of $FJUNK(F)$.

$FJUNK(F)$ has to ensure that *exactly* one copy of type F appears in our list. Thus if $FJUNK(F)$ is already holding a witness structure, we need to ensure that *no* node on the construction tree can produce a structure of type F. There are three ways in which this may happen. We explain all three problematic situations and the modifications necessary to resolve the conflicts.

First, a finite structure being built by τ may be abandoned and permanently thrown into the finite junk pool after $FJUNK(F)$ had already chosen its witness. To avoid this conflict we adopt the following modification to the basic strategy of τ:

Modification 1. If τ permanently abandons its structure due to an initialisation, we grow this abandoned structure F into a very large finite structure F'. Formally, given a finite structure F, we *enlarge* F by adding sufficiently many extra classes of size 1 to F to produce a structure F' that is different from any finite equivalence structure considered so far by the construction. Now add the enlarged abandoned structure to the junk pool by assigning it as a witness for $FJUNK(F')$. Note that $FJUNK(F')$ has no current witness and has in fact never acted before in the construction.

Second, a strategy τ on the priority tree may be playing `wait` and might wait forever at construction steps (i) or (ii). Again, the isomorphism type of the structure being built by τ can be the same as F for some $FJUNK(F)$ that already has a witness.

Modification 2. When playing the `wait` outcome, τ will first enlarge its structure U_τ and waits with this enlarged finite structure. Since τ cares about copying E_τ only if it is unbounded, it makes no harm to enlarge U_τ this way. Now the strategy $FJUNK(U_\tau)$ (which has never acted before) is temporarily suspended, since τ is currently holding on to a structure of the same isomorphism type.

If later on τ finishes its wait then U_τ will grow and $FJUNK(U_\tau)$ will then start a new structure as its witness. Since structures are always enlarged by adding a fresh number of new classes, $FJUNK(U_\tau)$ will be blocked in this way at most once.

Third, as we will see from the $INFJUNK$ strategy below that there might be a finite structure used as a witness by an $INFJUNK$ strategy which is permanently abandoned by the $INFJUNK$ strategy and added to the finite junk pool. In this case we also make sure that the abandoned structure F is first enlarged and then added to $FJUNK(F')$ as a witness for the appropriate F'.

Note that once a $FJUNK(F)$ strategy picks its follower, either on its own, or is assigned its follower when a node τ or an $INFJUNK$ strategy abandons its current structure, this follower is permanently tied to $FJUNK(F)$ and no other strategy in the construction will produce a structure of the same type.

2.5.2. *Description of $INFJUNK(F)$.* Here the situation is more complicated. We need to ensure that each strategy $INFJUNK(F)$ will produce in the limit a structure of the form $F \oplus I$, but at every finite stage $INFJUNK(F)$ has a finite part of its intended structure. As in the case of $FJUNK(F)$, each $INFJUNK(F)$ will eventually choose its witness and will start growing it to a structure of isomorphism type $F \oplus I$.

The obvious conflict is that a node τ on the priority tree will also produce an infinite junk structure at the end if $\tau * \texttt{pi}_2 i$ is its true outcome. Hence we need to ensure that the corresponding $INFJUNK$ strategy does not duplicate this structure in our list.

To avoid repetition, $INFJUNK(F)$ must permanently abandon its current witness D every time τ is visited and makes more progress in constructing $F \oplus I$. The structure D will then be enlarged and then placed into the finite junk collector. $INFJUNK(F)$ will then restart again with a new enlarged witness D' and starts growing D' towards $F \oplus I$, until τ is again visited and makes more progress, if ever. In this way, either τ or $INFJUNK(F)$ will successfully construct $F \oplus I$ in our list. Note that given any F, there is at most one pair (τ, i) such that τ constructs $F \oplus I$ under outcome $\tau * \texttt{pi}_2 i$.

We describe another possible conflict between infinite junk structures produced by the tree and $INFJUNK$ strategies. Consider the situation when τ is off the true path and is initialised infinitely often due to actions of other strategies. Then it may attempt to make progress in building $F \oplus I$ infinitely often, but in fact it will produce only an infinite collection of finite structures due to it being initialised. However $INFJUNK(F)$ is also prevented from building $F \oplus I$ since its witness is also infinitely often reset due to us (wrongly) assuming that τ was making progress. This means that we will never produce a copy of $F \oplus I$ if we simply follow the instructions as described above. The difficulty goes away if we adopt the following modification to the basic module of τ:

Modification 3. In the definition of the parameter $fin(\tau, j)$ we will search for a longer sequence of integers $f_1 < f_2 < \cdots < f_{\langle \tau, \mathcal{I}_\tau, j\rangle}$, instead of $f_1 < f_2 < \cdots < f_{\langle \tau, j\rangle}$ before. Here $\mathcal{I}_\tau$ is the number of times τ has been initialised through the actions of another node. Here we do not count *self-initialisations* where τ is initialised when playing outcome `init`. This modification causes τ to produce an infinite junk structure with between $\langle \tau, \mathcal{I}_\tau, i\rangle$ and $2i + \langle \tau, \mathcal{I}_\tau, i\rangle$ many finite classes, if τ was initialised exactly $\mathcal{I}_\tau$ many times (by other nodes) and has true outcome $\tau * \texttt{pi}_2 i$. Now after this modification, for each fixed F, the $INFJUNK(F)$ strategy only needs to worry about conflicts with a unique triple $(\tau, \mathcal{I}, i)$. If τ has been initialised fewer or more than $\mathcal{I}$ times, then $INFJUNK(F)$ will work on its witness structure. If τ is initialised exactly $\mathcal{I}$ times then $INFJUNK(F)$ will interact with τ as described above. This ensures that if $INFJUNK(F)$ fails to hold a stable witness, then τ must build a copy of $F \oplus I$.

A final conflict between infinite junk structures produced by the tree and $INFJUNK$ strategies is more subtle. An $INFJUNK(F)$ strategy might have its witness structure reset infinitely many times because a strategy τ plays outcome $\texttt{pi}_2 n$ infinitely often. However the strategy τ might in fact have true outcome $\texttt{pi}_2 m$ for $m < n$, and hence end up constructing an infinite junk structure which is not of type $F \oplus I$. This means that $F \oplus I$ is neither constructed by $INFJUNK(F)$ nor τ.

This problem can be fixed if we allow $INFJUNK(F)$ strategy to pick finitely many witness structures $D_0, D_1, \cdots D_n$ instead of a single witness, where $INFJUNK(F)$ is conflicted with outcome $\texttt{pi}_2 n$ of τ. While $INFJUNK(F)$ detects no conflicts, it will grow D_n towards $F \oplus I$ and keep $D_0, \cdots, D_{n-1}$ finite. Whenever τ plays outcome $\texttt{pi}_2 n$, $INFJUNK(F)$ will abandon D_n and begin with a new D_n (while keeping $D_0, \cdots, D_{n-1}$). When τ plays outcome $\texttt{pi}_2 m$ for $m < n$, we will abandon $D_{m+1}, D_{m+2}, \cdots, D_n$ and restart these with new witness structures, while keeping $D_0, \cdots, D_m$. We also grow D_m for one more step towards making $D_m \cong F \oplus I$.

At the end, if τ has true outcome to the right of $\texttt{pi}_2 n$, then $D_0, \cdots, D_{n-1}$ will finally stabilise at finite structures. We will have $D_n \cong F \oplus I$, but of course $U_\tau \not\cong F \oplus I$. If τ has true outcome $\texttt{pi}_2 n$, then $D_0, \cdots, D_{n-1}$ are stable finite structures, while D_n will be infinitely often abandoned. Here $D_\tau \cong F \oplus I$. Finally if τ has true outcome $\texttt{pi}_2 m$ for some $m < n$ then $D_0, \cdots, D_{m-1}$ are stable finite structures, while $D_{m+1}, \cdots, D_n$ are infinitely often abandoned. We finally make $D_m \cong F \oplus I$ and in this case, U_τ is an infinite junk structure not of type $F \oplus I$. Thus either τ or one of the $INFJUNK(F)$ witnesses (and exactly one) will succeed in building $F \oplus I$.

2.6. **Formal construction.** The basic strategies of τ and the junk collectors have been described above. We put it all together in this section. Due to technical reasons we will make some cosmetic changes to our discussions above. We adopt Modification 3 above in the definition of the paramenter $fin(\tau, j)$. The construction will at stage s define the current approximation δ_s to the true path, where $|\delta_s| = s$. Suppose $\tau \subset \delta_s$ has been defined. We now need to describe the actions of τ and the outcome played at this stage.

2.6.1. *Growing U_τ.* The first thing we do is to check if $\tau * \mathtt{pi}_3$ was played at the previous visit to τ, and if no, we do not grow U_τ and skip to step 2.6.2. Otherwise suppose that $\tau * \mathtt{pi}_3$ was played at the previous visit to τ. Let $s_0 < s$ be the previous stage where U_τ last grew. Take the following actions:

- For each class $\boldsymbol{e}_{\tau,j}$ such that no $\ell_\tau(i)$ maps to it and where $j \leq$ the largest element in the current range of fin, introduce a new class $\boldsymbol{u}_{\tau,i}$ in U_τ to have the same size and set $\ell_\tau(i) = j$.
- For every class $\boldsymbol{u}_{\tau,i}$ in U_τ such that $\ell_\tau(i)$ exists we grow $\boldsymbol{u}_{\tau,i}$ to have the same size as $\boldsymbol{e}_{\tau,\ell_\tau(i)}$.
- Enlarge U_τ by adding sufficiently many new $\boldsymbol{u}_\tau$ classes of size 1, so that the resulting finite structure has never been looked at by the construction.

If this is the first visit to τ since an initialisation (so that s_0 does not exist), we begin building a new structure U_τ by taking U_τ to be the enlargement of the empty structure.

Recall that $s_0 < s$ was the stage where we last grew U_τ; set $s_0 = 0$ if this does not exist. Check if one of the following holds:

- There exists a stage t such that $s_0 < t < s$, and some node α such that $\alpha * \mathtt{pi}_3 \subseteq \tau$ and $\alpha * \mathtt{pi}_2(i-1) \subseteq \delta_t$ for some i, or
- there is some $i < dom(fin)[s_0]$ such that $fin(i)$ has changed or $Q(\tau, k_\tau, i)$ has fired between s_0 and s. (As before we assume that $Q(\tau, k_\tau, 0)$ never fires).

Pick the least i for which one of the above applies:

$i = 0$: Play outcome `init` and initialise τ.

$i > 0$: If the first alternative applies we say that τ *is forced to play outcome* $\tau * \mathtt{pi}_2(i-1)$; if the second alternative applies we say that τ *wants to play outcome* $\tau * \mathtt{pi}_2(i-1)$. In either case we play outcome $\tau * \mathtt{pi}_2(i-1)$ and take the actions described under the basic strategy for τ in Section 2.3.4(iii) when outcome $\mathtt{pi}_2(i-1)$ is played.

No i found: Play outcome `wait`.

In any case go to step 2.6.3.

2.6.2. *Acting for τ.* Suppose that we did not manage to grow U_τ in step 2.6.1. Now take the actions and outcome described under the basic strategy for τ in Section 2.3.4(i) or (ii), whichever applies. Proceed to step 2.6.3.

2.6.3. *Initialising other nodes.* We have described the actions of τ and the outcome of τ at this stage. Now initialise all the nodes mentioned in Section 2.4. When a node τ is initialised we will first enlarge the finite structure U_τ to U' and assign it as a witness to $FJUNK(U')$, reset all parameters associated with the node (except for the counter $\mathcal{I}_\tau$) and increase the value of $\mathcal{I}_\tau$ by 1, unless this is due to a self-initialisation.

This ends the description of the actions and the outcomes of τ. Suppose we have finished with the definition of δ_s of length s. Before we conclude the construction stage s, we shall act for the junk collector strategies. First process the $INFJUNK$ strategies (Section 2.6.4) followed by the $FJUNK$ strategies (Section 2.6.5).

2.6.4. *$INFJUNK$ action.* For each F with code number less than s, we act for $INFJUNK(F)$ as follows. (The actions for different F are independent). First let $\langle \tau, \mathcal{I}, n\rangle$ be the triple such that the number of classes of F is between $\langle \tau, \mathcal{I}, n\rangle$ and $2\langle \tau, \mathcal{I}, n\rangle$. This triple, if it exists, is unique. If this triple is not found, then $INFJUNK(F)$ will have no conflict with any structure built by the construction tree, and in this case, simply enumerate a witness $F \oplus I$ into our list.

Otherwise fix this triple $\langle \tau, \mathcal{I}, n\rangle$. If $\mathcal{I}_\tau$ is currently not equal to $\mathcal{I}$, we will pick a new witness structure D for $INFJUNK(F)$ and begin growing D towards $F \oplus I$. If $\mathcal{I}_\tau$ later becomes equal to $\mathcal{I}$ we will abandon the previous witness structure and proceed as below. If $\mathcal{I}_\tau$ increases from $\mathcal{I}$ to $\mathcal{I}+1$ then we abandon all witness structures and pick a new D.

Otherwise suppose that $\mathcal{I}_\tau = \mathcal{I}$. We begin by picking new witness structures $D_{-1}, D_0, D_1, \cdots, D_n$. At each stage we will determine if $INFJUNK(F)$ has any conflicts with τ. If it is not the case that τ is visited and outcome $\mathtt{pi}_2 m$ for some $m \leq n$ or outcome $\mathtt{init}$ is taken, then there are no conflicts; at this stage $INFJUNK(F)$ simply does nothing with $D_{-1}, D_0, \cdots, D_{n-1}$ and grows D_n for one more step towards making $D_n \cong F \oplus I$.

Suppose that τ is visited and outcome $\mathtt{pi}_2 m$ for $m < n$ is taken. Then $INFJUNK(F)$ does nothing with $D_{-1}, D_0, \cdots, D_{m-1}$, and abandons and picks new witness structures for $D_{m+1}, \cdots, D_n$. It also grows D_m by one more step towards making $D_m \cong F \oplus I$.

Suppose that τ is visited and outcome $\mathtt{init}$ is taken. Then $INFJUNK(F)$ abandons and picks new witness structures for $D_0, \cdots, D_n$. It also grows D_{-1} by one more step towards making $D_{-1} \cong F \oplus I$.

Finally suppose that τ is visited and outcome $\mathtt{pi}_2 n$ is taken. The action of τ at this stage was to restrain a set of classes G of U_τ and grow all the remaining ones. If $F \subseteq G$ and such that every class of $G - F$ is larger than the largest class in F, and has grown since the last time we considered this step, then a conflict occurs at this stage. $INFJUNK(F)$ will abandon its current witness and pick a new witness structure D_n and keeps $D_{-1}, \cdots, D_{n-1}$. Otherwise if there are no conflicts then simply grow D_n.

In the strategy above we need to ensure several things. First, every time we pick a new witness structure, we need to take it to be a suitable enlargement of the empty structure. Second, every time we grow an existing structure, we need to increase the size of one of its classes, as well as take a suitable enlargement of it. Third, if any witness structure is abandoned, $INFJUNK(F)$ will first enlarge it to a suitable finite structure D' and then assign it to $FJUNK(D')$ as a follower.

2.6.5. *$FJUNK$ action.* For each F with code number less than s, we act for $FJUNK(F)$ as follows. If $FJUNK(F)$ has no current witness, and if there is no structure X currently in the list such that $X \cong F$, then $FJUNK(F)$ will introduce a witness structure of isomorphism type F. Otherwise, do nothing.

This ends the description of the construction.

2.7. **Verification.**

We define the true path of the construction inductively, each time selecting the leftmost outcome which is visited infinitely often. It is easy to check that each node on the true path is initialised by another node only finitely often. Infinite self-initialisation is still possible.

We will need to argue two things: First, all structures of type (I), (III) and (V) are represented in our list, and second, that our list does not contain repetitions.

2.7.1. *All isomorphism types in (I), (III) and (V) are represented.* We begin with an important lemma.

Lemma 2.1. *Let τ be a node on the construction tree, which is visited infinitely often by the construction, initialised by other nodes $\mathcal{I} < \infty$ times, and where $\tau * \mathtt{pi}_2 i$ is the leftmost outcome of τ visited infinitely often. Then the final structure U_τ built by τ is an infinite junk structure of type (III) with between $\langle \tau, \mathcal{I}, i\rangle$ and $2\langle \tau, \mathcal{I}, i\rangle$ many finite classes.*

Proof. Since $\mathtt{pi}_2 i$ is played only when we grow U_τ, this means that we grow U_τ infinitely often. This means that $\tau * \mathtt{pi}_3$ is also played infinitely often and so $dom(fin)$ grows arbitrarily large. Since τ is visited infinitely often but outcomes to the left of $\mathtt{pi}_2 i$ are each played finitely often, this means that $fin(i)$ will eventually be defined and hold a stable value. After $fin(i)$ is stable, for each class $\boldsymbol{e}_{\tau,f}$ of $fin(i)$, we will be able to define $\ell_\tau^{-1}(f)$ and never again redefine $\ell_\tau^{-1}(f)$ (notice that these cannot be redefined by an outcome to the right of $\tau * \mathtt{pi}_2 i$). Since all classes in $fin(i)$ are finite, this means that there are at least $\langle \tau, \mathcal{I}, i\rangle$ many finite classes in U_τ.

Each time we play outcome $\mathtt{pi}_2 i$ we will preserve a fixed set of U_τ classes: these are $\boldsymbol{u}_{\tau,j}$ for $j < i+1$ and $\boldsymbol{u}_{\tau,\ell_\tau^{-1}(f)}$ for $f \in fin(i)$ or $f < i+1$. There are at most $2i+2+\langle \tau, \mathcal{I}, i\rangle < 2\langle \tau, \mathcal{I}, i\rangle$ many preserved classes. All other classes are increased in size. Thus U_τ will have at most $2\langle \tau, \mathcal{I}, i\rangle$ many finite classes.

Finally each time we grow U_τ we enlarge it, so U_τ must contain infinitely many classes, and hence infinitely many infinite classes. Hence U_τ is an infinite junk structure of type (III). Note that τ *does not* have to be on the true path in the statement of the lemma. □

Lemma 2.2. *Suppose τ is on the true path, and the true outcome of τ is $\mathtt{pi}_3$. Then $E_\tau \cong U_\tau$ is unbounded.*

Proof. Since $\tau * \mathtt{pi}_3$ is played infinitely often, hence $dom(fin)$ must grow arbitrarily long. Furthermore $\tau * \mathtt{pi}_2 i$ is played finitely often for each i, this means that $fin(i)$ must eventually be defined and stable. Therefore E_τ must be unbounded. Also the unboundedness of E_τ implies that whenever ℓ_τ is set undefined for some class $\boldsymbol{u}_i$ in U_τ and later seeks a new image in E_τ, it will be able to find a suitable image (since E_τ contains arbitrarily large classes). Similarly, since U_τ is grown at infinitely many stages and $dom(fin)$ grows arbitrarily long, whenever $\ell_\tau^{-1}(j)$ is set undefined for some class $\boldsymbol{e}_j$ in E_τ, we will always get a chance to redefine $\ell_\tau^{-1}(j)$. Furthermore each $\ell_\tau(i)$ and $\ell_\tau^{-1}(j)$ will not be made undefined once $fin(i)$ or $fin(j)$ is stable; notice these cannot be made undefined by an outcome or node to the right of $\tau * \mathtt{pi}_2 i$ or $\tau * \mathtt{pi}_2 j$. Thus ℓ_τ is total, and clearly bijective.

It is also easy to check that at every stage where $\ell_\tau(i)$ is defined, the current size of $\boldsymbol{u}_i$ is at most the current size of $\boldsymbol{e}_{\ell(i)}$: This is certainly true at the point when $\ell_\tau(i)$ receives a new definition, and after that we only grow $\boldsymbol{u}_i$ to match the size of $\boldsymbol{e}_{\ell(i)}$ (unless under a $\mathtt{pi}_2 n$ outcome where we also make $\ell(i)$ undefined). Since there are infinitely many stages where we grow U_τ, we have in fact that the sizes of $\boldsymbol{u}_i$ and $\boldsymbol{e}_{\ell(i)}$ are equal. Therefore $E_\tau \cong U_\tau$, and in fact $E_\tau \cong_{\Delta^0_2} U_\tau$. □

Lemma 2.3. *All structures in our list are of type either (I), (III) or (V).*

Proof. Every time $FJUNK(F)$ receives or picks a witness structure, it is of type F and no further changes are made to this witness structure.

If $INFJUNK(F)$ picks a witness structure D, and if D is either later abandoned or is never grown again, then D ends up being finite. On the other hand if D is never abandoned and is grown infinitely often, then we will be able to make it of type $F \oplus I$; the only issue is that we always enlarge a structure when growing, however, this action only adds classes of size 1, so is compatible with extending it to $F \oplus I$.

Finally if a structure U is introduced by a node τ, and is either later abandoned, or is grown finitely often, then U ends up being finite. Otherwise suppose U is grown infinitely often and never abandoned. This means that τ is visited infinitely often but initialised only finitely often, and the leftmost outcome of τ visited infinitely often has to be either $\mathtt{pi}_3$ or $\mathtt{pi}_2 n$ for some n. If it is $\mathtt{pi}_2 n$ then we apply Lemma 2.1. Otherwise if it is $\mathtt{pi}_3$ then we note that τ has to be on the true path (otherwise τ will be forced to play a $\tau * \mathtt{pi}_2 n$ outcome infinitely often for some n) and so we apply Lemma 2.2. □

Any activity on a structure X can be classified according to: X is first introduced to the list, X is grown, or X is abandoned. These are the only three ways in which X can be modified. It is easy to check the following fact:

Fact 2.4. Any activity on a structure X must also be followed by an enlargement of the structure at the same time. The only exception is when a $FJUNK$ strategy picks a new witness structure.

Lemma 2.5. *Every isomorphism type in (I), (III) and (V) are represented.*

Proof. Consider a finite type F, and let s be the stage where $FJUNK(F)$ is first active. If $FJUNK(F)$ ever picks a follower structure, it will be stable of type F, so let's assume it never gets to pick a follower. This means that at stage s there is at least one structure X currently in the list such that $X \cong F$. (In fact there is at most one such X at any time, but so far we have not addressed uniqueness, and this will be done in Section 2.7.2). By Fact 2.4 no activity which occurs after stage s can produce a structure of type F. Thus in order that $FJUNK(F)$ remains blocked at every stage, it must be that one of these structures X already present at stage s is stable, and of type F. So all structures in type (I) are represented.

Now fix F and consider $INFJUNK(F)$. If $INFJUNK(F)$ eventually detects no conflicts then it will hold a stable witness structure of type $F \oplus I$. (Again enlarging the structure while growing is compatible with making it of type $F \oplus I$). Otherwise $INFJUNK(F)$ will infinitely often detect a conflict. This conflict must each time be with a node τ playing some outcome $m \leq n$ or $\mathtt{init}$ and during the time when $\mathcal{I}_\tau = \mathcal{I}$, for a unique triple $\langle \tau, \mathcal{I}, n \rangle$.

Let m be the least where the conflict happens infinitely often. If $m < n$ then D_m will hold a stable witness structure and we will make $D_m \cong F \oplus I$. Same for the case $\mathtt{init}$ where $D_{-1} \cong F \oplus I$. On the other hand if $m = n$ and a conflict is detected infinitely often then $G - F$ must contain only infinite classes, and in fact U_τ must contain only F as its set of finite classes. Hence $U_\tau \cong F \oplus I$. (Note that U_τ is never abandoned unless $\mathcal{I}_\tau$ is increased or outcome $\mathtt{init}$ is played). So all structures in type (III) are represented.

Now finally we fix an unbounded E of type (V), and argue that it is represented. Let i be the least such that $E_i \cong E$; since i is least, there is some k such that $P(i,k)$ has Π^0_3-outcome. Let τ be along the true path assigned E_i and guessing $P(i,k)$. We claim that τ must have true outcome $\mathtt{pi}_3$. Since E_τ is unbounded, and τ being on the true path is initialised by another node only finitely often, this means that $\mathcal{I}_\tau$ is stable and hence $fin(\tau, n)$ eventually holds a stable value for each n. We cannot be stuck waiting under Section 2.3.4, step (i) for τ, as E_τ is unbounded, and so this means that outcome $\mathtt{pi}_3$ of τ is played infinitely often.

Suppose some outcome to the left of $\mathtt{pi}_3$ is played infinitely often. This cannot be $\mathtt{init}$ because $fin(0)$ holds a stable value. On the other hand this cannot be outcome $\mathtt{pi}_2(n-1)$; τ cannot be forced to infinitely often play $\mathtt{pi}_2(n-1)$, otherwise the true path is to the left of τ, and τ cannot want to infinitely often play $\mathtt{pi}_2(n-1)$ since $fin(n)$ is eventually stable and $Q(\tau, k, n)$ eventually stops firing. Hence $\tau * \mathtt{pi}_3$ is along the true path. By Lemma 2.2 we have $U_\tau \cong E_i$. Hence all structures of type (V) are represented. □

2.7.2. *Our list does not contain repetitions.* Now it remains to verify that no two structures in our list are of the same type. Fix two structures X and Y in our list. We shall argue that $X \not\cong Y$.

Suppose X and Y are both finite. Then they each have a final activity before being stable, assume that the final activity for X is after the final activity for Y. The following are all the different possibilities:

Both X and Y are picked by $FJUNK$ strategies: Since each $FJUNK$ strategy picks at most one witness structure, this means that $X \ncong Y$.

X is not picked by a $FJUNK$ strategy: By Fact 2.4, X is enlarged during its final activity and so $X \ncong Y$.

X is picked by a $FJUNK$ strategy: Then as Y is already stable when X is picked, $X \ncong Y$.

Now suppose that X and Y are both infinite, in particular neither of them are ever abandoned. Again the following are all the different possibilities:

Both X and Y infinitely often grown by $INFJUNK$ strategies: Since each single $INFJUNK(F)$ strategy produces at most one infinite structure at the end (which is necessarily of the type $F \oplus I$), this means that X and Y are witnesses of different $INFJUNK$ strategies, hence, $X \ncong Y$.

Both X and Y infinitely often grown by strategies on priority tree: Then there are nodes τ_x and τ_y responsible for X and Y respectively. Each node works on a single structure U_τ at any one time, so $\tau_x \neq \tau_y$. This means that τ_x and τ_y are initialised finitely often, visited infinitely often and play a leftmost outcome $\mathtt{pi}_3$ or $\mathtt{pi}_2 n$ for some n infinitely often. If τ_x plays leftmost outcome $\mathtt{pi}_2 n$ then apply Lemma 2.1 to see that X is an infinite junk structure with between $\langle \tau_x, \mathcal{I}_{\tau_x}, n\rangle$ and $2\langle \tau_x, \mathcal{I}_{\tau_x}, n\rangle$ many finite classes. If τ_x plays leftmost outcome $\mathtt{pi}_3$ then apply Lemma 2.2 to see that $X \cong E_{\tau_x}$ is unbounded. The same applies to τ_y. Hence it is clear that $X \ncong Y$ except in the case where both τ_x and τ_y have leftmost outcome $\mathtt{pi}_3$.

If this is the case then both τ_x and τ_y must lie along the true path, so assume that $\tau_x * \mathtt{pi}_3 \subseteq \tau_y$. As τ_y is not deactivated, it means that $i_x \neq i_y$, where τ_x is assigned E_{i_x} and τ_y is assigned E_{i_y}. Since *both* nodes τ_x and τ_y have true outcome $\mathtt{pi}_3$, it must be that $E_{i_y} \ncong E_{i_x}$ (note that it could be that $i_x > i_y$). Hence $X \ncong Y$.

X infinitely often grown by τ and Y infinitely often grown by $INFJUNK(F)$: As in the analysis above, in order for τ to produce an infinite junk structure, it must be initialised finitely often, visited infinitely often and play a leftmost outcome $\mathtt{pi}_2 n$ for some n infinitely often. Suppose that $X \cong Y \cong F \oplus I$. The number of finite classes in F must be between $\langle \tau, \mathcal{I}_\tau, n\rangle$ and $2\langle \tau, \mathcal{I}_\tau, n\rangle$. Since τ plays leftmost outcome $\mathtt{pi}_2 n$, this means that $INFJUNK(F)$ will have stable finite witness structures $D_{-1}, \cdots, D_{n-1}$, while D_n gets abandoned infinitely often. This means that $INFJUNK(F)$ does not in fact produce an infinite structure, a contradiction.

This ends the proof of Theorem 1.1.

3. A Friedberg enumeration of infinite equivalence structures.

In this section we prove the second result of the paper, that is, we produce a Friedberg enumeration of all isomorphism types of infinite computable equivalence structures.

We slightly abuse our terminology and identify a finite equivalence structure with its isomorphism type. Every non-empty finite equivalence structure F is uniquely described up to isomorphism by the finite sequence $0 < s_0 \leq s_1 \leq \ldots \leq s_k$ of sizes of its classes. Each such tuple is also uniquely described by the number $N_F = p_0^{s_0} p_1^{s_1} \ldots p_k^{s_k}$, where $p_0 = 2, p_1 = 3, \ldots$ is the standard effective list of prime numbers. (Here the subscript F in N_F denotes the isomorphism type of the corresponding finite equivalence structure.)

By the main result of the paper, Theorem 1.1, we know that there exists a Friedberg enumeration of all isomorphism types of equivalence structures which are either finite, unbounded, or of type (III). Suppose $(F_n)_{n\in\omega}$ is such an enumeration. As was noted earlier in the paper, we may assume that all structures in this list are non-empty. Notice that in the enumeration provided by Theorem 1.1, we cannot effectively tell which structures are finite and which are infinite, since some structures are built by nodes which may never again grow the structure. Thus we need a separate, non-trivial argument to show that we can eliminate finite structures from this list.

Definition of V_n. Given any F_n as above, define the structure

$$V_n = F_n \oplus C_n,$$

where C_n is infinite and attempts to "code" F_n as follows. If F_n is finite, then C_n is the structure with infinitely many classes of size 1 and exactly N_{F_n}-$K_{F_n,2}$-many classes of size 2, where $K_{F_n,2}$ is the number of classes of size 2 in F_n. If F_n is infinite, then C_n is empty.

Note that $K_{F,2}$ is always smaller than N_{F_n} for a trivial counting reason (assuming F_n is non-empty), so the definition makes sense. Clearly, each V_n is infinite. More can be said.

Lemma 3.1. *Any Friedberg enumeration of the sequence $(V_n)_{n\in\omega}$ can be effectively turned into a Friedberg enumeration of all infinite equivalent structures (up to isomorphism).*

Proof. Which infinite isomorphism types are missing in the list $(V_n)_{n\in\omega}$? All structures of type (III) and (V) are infinite structures in the list $(F_n)_{n\in\omega}$ and hence will appear in the list $(V_n)_{n\in\omega}$. We are missing the structures in (II), as well as the structures in (IV) which are not of the form $F \oplus C$, where F is finite and C has infinitely many classes of size 1 and exactly N_F-K_F-many classes of size 2. We can evidently list all such structures which are not of this kind. By adjoining the missing isomorphism types, we produce a listing of all infinite isomorphism types which is easily checked to be Friedberg. □

The reader might now wonder if it is possible for finite structures $F \neq F'$ but the associated $F \oplus C \cong F' \oplus C'$. This is in fact not possible, which is the reason why we chose to define C in that way. This will be explained in the next lemma. Now it remains to prove that there exists a Friedberg enumeration of the sequence $(V_n)_{n\in\omega}$:

Lemma 3.2. *Given an enumeration of $(F_n)_{n\in\omega}$ we can produce an enumeration of $(V_n)_{n\in\omega}$.*

Proof. We write F and V instead of F_n and V_n. Fix an effective approximation $(F[s])_{s\in\omega}$ of F by finite substructures, $F[s] \subseteq F[s+1]$. Without loss of generality, each extension $F[s] \subseteq F[s+1]$ may be assumed to be at most a one-point extension.

Before showing the construction, we give the reason for our choice of C. We wish to append infinitely many elements to each finite structure F, since we need $V = F \oplus C$ to be an infinite structure, so C should be padded with infinitely many elements. The easiest way to do this is to add infinitely many classes of size 1 to C; the reason for choosing infinitely many classes of size 1 over, say, a single infinite class is that it is easier to move these classes of size 1 from C into F whenever F grows; remember we have to prepare for the possibility that F itself is infinite.

However, letting C contain only classes of size 1 will not be enough. Remember we have to produce a list without repetitions, so for example, if F and F' are finite structures such that $F' - F$ are all size 1 classes, then we get $F \oplus C \cong F' \oplus C'$. Thus, our definition of C must distinguish between these two cases. To do this, we let C contain, in addition to infinitely many classes of size 1, also $N - K$ many classes of size 2. This ensures that $F \oplus C$

contains exactly N many classes of size 2 (remember N codes the atomic diagram of F), and so $F \oplus C \not\cong F' \oplus C'$.

Now we describe the construction. Suppose we have already defined $V[s]$ which is of the form $F'[s] \oplus T_1[s] \oplus T_2[s]$, where $F'[s] \cong F[s]$, the component $T_1[s]$ contains at least 2^s disjoint singleton classes, and $T_2[s]$ consists of exactly $N_{F[s]}$-$K_{F[s],2}$-many classes of size 2. We will define $V[s]$ in stages. We will also implicitly maintain the isomorphism from $F[s]$ onto $F'[s]$ in a rather straightforward way, so we do not introduce any notation for the isomorphism.

Case 1. If $F[s] = F[s+1]$, then we grow $V[s]$ to $V[s+1]$ according to the definition of $V[s+1]$, i.e., by growing the T_1-component.

Note if $F[s] = F$, then this way we'll end up producing V. However, if F is actually infinite, then we need to make sure that all classes previously forming T_1 and T_2 eventually become parts of F'. For this purpose we assign *priority* to elements of $T_1[s] \oplus T_2[s]$, depending on when the class was first introduced in the construction (the earlier the class is introduced the higher priority it receives).

Case 2. Now assume $F[s+1]$ is a one-point extension of $F[s]$ by element x.

- Suppose x forms a new singleton class. Then look for the highest priority singleton class $[y]$ currently in $T_1[s]$. Then adjoin it to the copy of $F'[s]$ in $V[s+1]$ and declare this class to be the image of x in the new definition of $F'[s+1]$:
$$F'[s+1] = F'[s] \oplus [y].$$
(The class keeps its priority even after it has been adjoined.)
- Assume x extends the already existing class $\mathbf{f}$ that currently has size ≥ 2. To define $F'[s+1]$ find the respective class $\mathbf{f'}$ in $F'[s]$ and extend it by one new element.
- Suppose x extends the already existing singleton class $\mathbf{f}$ that naturally corresponds to $\mathbf{f'}$ in $F'[s]$. Then $\mathbf{f'}$ was introduced to $F'[t]$ ($t \leq s$) by adjoining a T_1-class to the F'-component. See whether there exists a two-element class currently in $T_2[s]$ that has higher priority than that of $\mathbf{f'}$ in the construction.
  - If there is no such class, then extend $\mathbf{f'}$ by one point to define $F'[s+1]$ and extend the natural isomorphism from $F[s+1]$ onto $F'[s+1]$ in the most obvious way.
  - If there is such class, then let $\mathbf{t_2}$ be such a class of highest priority. In this case remove $\mathbf{f'}$ from $F'[s]$, return it back to $T_1[s]$ to form $T_1[s+1]$. (The class still keeps its priority.) Then adjoin $\mathbf{t_2}$ to what is left of $F'[s]$ to define $F'[s+1]$. Define the natural isomorphism from $F[s+1]$ onto $F'[s+1]$ by matching $\mathbf{f}$ with $\mathbf{t_2}$ (instead of $\mathbf{f'}$).

To finish actions in Case 2, extend the T_1- and T_2-components accordingly to satisfy the definition of $V[s+1]$

Verification. The key to the verification is checking that the structures can be consistently grown from stage to stage, i.e., that $V[s+1] \supseteq V[s]$. For this purpose we need:

Claim 3.3. *Let $F \subsetneq G$ be non-empty finite equivalence structures. Then $N_F - K_F < N_G - K_G$.*

Proof. It is sufficient to check that the lemma is true in the case of a one-point extension, i.e., when $G \setminus F$ is a singleton $\{x\}$. If x forms a singleton class of G, then without loss of generality we may assume that it contributes p_0^1 into $N_G = p_0^1 \cdots$. In this case we have that any factor of the form p_i^k in N_F corresponds to p_{i+1}^k in N_G. But $p_i < p_{i+1}$ and $p_0 = 2$ while $K_{G,2} = K_{F,2}$. Thus the lemma holds in this case.

Now assume x extends one of the already existing equivalence classes of F. Then $K_{G,2} \leq K_{F,2} + 1$. Suppose x extends the j^{th} largest class in F, i.e. of size s_j in the notation preceding the lemma. If s_j was the largest size of an equivalence class in F then it stays largest in G, and thus the lemma follows from $2p_j^{s_j} \leq p_j^{s_j+1}$. Otherwise, suppose we have some $s_{j+1} > s_j$. But then we may assume that the class holds its position as the j^{th} largest class in G (after a suitable re-arrangement) and the lemma holds for the same reason as above. □

Now, regardless of which case we are in at a stage s, the claim above ensures that we can always make the action of adjusting the size of $T_2[s]$ (growing is necessary) since it never gets smaller.

The rest of the verification is rather standard. Indeed, we have already noted that if F is finite then $V = \bigcup_s V[s]$ is of the desired form. Thus assume F is infinite. Since F_n is of type (III) or (V) and hence also contains infinitely many large enough classes (finite or infinite), we conclude that there will be infinitely many stages that are 1-point extensions at which a class of size 1 grows in $F[s]$, and also infinitely many stages at which a new singleton class is introduced to $F[s]$

A class from the T_2-component, once put into the F'-component, will stay there permanently. Thus, every class that has ever been put into the T_1-component can be extracted from it and then put back at most finitely many times.

Therefore, if F is infinite then we see that every class that has ever been put into either the T_1- or the T_2-component will be eventually enumerated into the F'-component of V and will never leave it. Equivalently, $V = \bigcup_{s\in\omega} F'[s]$. It remains to show that the sequence of natural isomorphisms from $F[s]$ onto $F'[s]$, as defined in the construction, induces an isomorphism of F onto V. But this is a consequence of priority: The natural isomorphism has to be redefined on a class at most finitely often, and once stable it is stable for the whole class (regardless of whether the class is finite or not). Therefore the sequence of natural isomorphisms induces a Δ_2^0 isomorphism from F onto V, as desired. □

It remains to verify that the list $(V_n)_{n\in\omega}$ contains no repetitions, i.e., that $V_n \not\cong V_m$ for each $n \neq m$. Recall that $(F_n)_{n\in\omega}$ is a Friedberg enumeration of all computable isomorphism types of equivalence structures of type (I), (III) or (V). Suppose first that F_n is finite. In this case V_n is bounded and contains no infinite classes. Thus if F_m is infinite then $V_n \not\cong V_m \cong F_m$. Now if F_m is finite, then the number of classes of size 2 in V_m is different from that in V_n, since V_n will have exactly N_{F_n} many classes of size 2. On the other hand, if both F_m, F_n are infinite then $F_m \cong V_m$ and $F_n \cong V_n$.

References

[AK00] C. 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.

[ALM$^+$14] Uri Andrews, Steffen Lempp, Joseph S. Miller, Keng Meng Ng, Luca San Mauro, and Andrea Sorbi. Universal computably enumerable equivalence relations. *J. Symb. Log.*, 79(1):60–88, 2014.

[Bae37] R. Baer. Abelian groups without elements of finite order. *Duke Math. J.*, 3(1):68–122, 1937.

[BC08] Paul Brodhead and Douglas Cenzer. Effectively closed sets and enumerations. *Arch. Math. Logic*, 46(7-8):565–582, 2008.

[CCHM06] W. Calvert, D. Cenzer, V. Harizanov, and A. Morozov. Effective categoricity of equivalence structures. *Ann. Pure Appl. Logic*, 141(1-2):61–78, 2006.

[DKT11] R. Downey, A. Kach, and D. Turetsky. Limitwise monotonic functions and applications. In *Proceedings of STACS 2012*, pages 56–85, 2011.

[DM08] R. Downey and A. Montalbán. The isomorphism problem for torsion-free abelian groups is analytic complete. *J. Algebra*, 320(6):2291–2300, 2008.

[DM13] R. Downey and A. Melnikov. Effectively categorical abelian groups. *J. Algebra*, 373:223–248, 2013.

[DM14] Rodney Downey and Alexander G. Melnikov. Computable completely decomposable groups. *Trans. Amer. Math. Soc.*, 366(8):4243–4266, 2014.

[DMNa] R. Downey, A. Melnikov, and K. Ng. Abelian p-groups and the halting problem. Preprint.

[DMNb] R. Downey, A. Melnikov, and K. Ng. Iterated effective embeddings of abelian p-groups. *To appear in International Journal of Algebra and Computation.*

[DMN15] Rod Downey, Alexander G. Melnikov, and Keng Meng Ng. On Δ_2^0-categoricity of equivalence relations. *Ann. Pure Appl. Logic*, 166(9):851–880, 2015.

[Dow98] R. Downey. Computability theory and linear orderings. In *Handbook of recursive mathematics, Vol. 2*, volume 139 of *Stud. Logic Found. Math.*, pages 823–976. North-Holland, Amsterdam, 1998.

[EG00] Y. Ershov and S. Goncharov. *Constructive models.* Siberian School of Algebra and Logic. Consultants Bureau, New York, 2000.

[Ers] Y.L. Ershov. Theory of numberings. *In: Griffor, E.R. (ed.) Handbook of Computability Theory, volume 140 of Studies in Logic and the Foundations of Mathematics, pp. 473–506. Elsevier Science, Amsterdam.*

[FF12] Ekaterina B. Fokina and Sy-David Friedman. On Σ_1^1 equivalence relations over the natural numbers. *MLQ Math. Log. Q.*, 58(1-2):113–124, 2012.

[Fri58] Richard M. Friedberg. Three theorems on recursive enumeration. I. Decomposition. II. Maximal set. III. Enumeration without duplication. *J. Symb. Logic*, 23:309–316, 1958.

[Fuc70] L. Fuchs. *Infinite abelian groups. Vol. I.* Pure and Applied Mathematics, Vol. 36. Academic Press, New York, 1970.

[GK02] S. Goncharov and J. Knight. Computable structure and antistructure theorems. *Algebra Logika*, 41(6):639–681, 757, 2002.

[Har08] K. Harris. η-representation of sets and degrees. *J. Symbolic Logic*, 73(4):1097–1121, 2008.

[Hig61] G. Higman. Subgroups of finitely presented groups. *Proc. Roy. Soc. Ser. A*, 262:455–475, 1961.

[HKSS02] D. Hirschfeldt, B. Khoussainov, R. Shore, and A. Slinko. Degree spectra and computable dimensions in algebraic structures. *Ann. Pure Appl. Logic*, 115(1-3):71–113, 2002.

[Khi98] N. Khisamiev. Constructive abelian groups. In *Handbook of recursive mathematics, Vol. 2*, volume 139 of *Stud. Logic Found. Math.*, pages 1177–1231. North-Holland, Amsterdam, 1998.

[KKM13] I. Kalimullin, B. Khoussainov, and A. Melnikov. Limitwise monotonic sequences and degree spectra of structures. *Proc. Amer. Math. Soc.*, 141(9):3275–3289, 2013.

[KNS97] B. Khoussainov, A. Nies, and R. Shore. Computable models of theories with few models. *Notre Dame J. Formal Logic*, 38(2):165–178, 1997.

[Lac87] A. H. Lachlan. A note on positive equivalence relations. *Z. Math. Logik Grundlag. Math.*, 33(1):43–46, 1987.

[LSM] K. Lange, R. Steiner, and R. Miller. Effective classification of computable structures. *The Notre Dame Journal of Formal Logic.*

[PER89] Marian B. Pour-El and J. Ian Richards. *Computability in analysis and physics.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1989.