ENUMERATING ABELIAN p-GROUPS

ROD DOWNEY, ALEXANDER MELNIKOV, AND KENG MENG NG

ABSTRACT. Given an integer $n > 0$ we give a computable injective listing of the isomorphism types of all computable abelian p-groups of Ulm type $\leq n$. We give a similar result for certain classes of profinite groups.

1. INTRODUCTION

A central problem in many areas of mathematics is the *classification* problem. For a class of structures $\mathcal{K}$, this problem typically asks "Is there a way to understand or classify the structures in $\mathcal{K}$ up to isomorphism?" Often this classification involves determining *invariants* which transform the question of whether $A \cong B$ into whether A and B have the same invariants. Since "isomorphism type" is itself an invariant, we would expect useful invariants to make the classification problem simpler. Examples of classes with a useful invariant include dimension for vector spaces, Baer invariants for completely decomposable groups [Bae37], and Jones polynomials for knots. Mathematical logic provides the tools for understanding the isomorphism problems for various classes of structures. Where classification is possible, we can use logic to calibrate precisely how hard the isomorphism problem is. In this article we use tools of computable structure theory [AK00, EG00] to produce a fine-grained *algorithmic* classification for a broad class of groups.

Computable structure theory offers several general approaches to the classification problem for a given class of structures; see [GK02]. Here we recall that all typical countable structures met in practice are naturally computably given, where a countable algebraic structure is computable if its domain and the operations are Turing computable [Mal61, Rab60]. The standard representation of the additive group of the rationals is computable, and a finitely presented group is computable if its word problem is decidable. The nicest classification of an isomorphism problem would be one where we can decide if two structures are isomorphic; the isomorphism problem is algorithmically decidable. For instance, the isomorphism problem for finite abelian groups is clearly decidable, and furthermore all isomorphism types of such groups can be computably listed without repetition. Although the isomorphism problem for arbitrary finite groups is also decidable, it is still open whether it is computationally *feasibly* decidable; see, e.g., [AT11]. For many infinite structures we have no hope of deciding the isomorphism problem, let alone feasibly deciding it. In these cases we seek to understand how hard the isomorphism problem is using various hierarchies or by asking whether there is a way to injectively list the isomorphism types. We note that a class can have a computable listing of isomorphism types even though the isomorphism problem in the class is not decidable; an

The first two authors were partially supported by the Marsden Fund of New Zealand. The third author is partially supported by MOE2015-T2-2-055 and RG131/17.

elementary example is the class of countable vector spaces over a fixed computable field. After understanding the problem for computable structures, the process of *relativisation* to an arbitrary oracle will allow us to understand the general isomorphism problem for arbitrary countable structures. We now elaborate on these concepts in more detail.

The *isomorphism problem* for a class $\mathcal{K}$ of structures is the set $\{\langle i, j\rangle \mid \mathcal{A}_i \cong \mathcal{A}_j\}$ for some enumeration $\{\mathcal{A}_e \mid e \in \omega\}$ of the computable members of the class we are interested in. The complexity of the isomorphism problem for a class $\mathcal{K}$ can be measured using various hierarchies such as the arithmetical and the analytical hierarchy [Rog87]. Typically such results can be relativised to any oracle, which means that the oracle can be used as a parameter in the result. More formally, the "boldface" [Sac90] versions of such results are not restricted to computable members of the class but can cover the whole class. For example, in the case of torsion-free abelian groups the isomorphism problem is Σ^1_1-complete [DM08]. The result can be relativised to any oracle, showing that the collection of reals that naturally code torsion-free abelian groups is analytic complete [DM08]. Showing the index set is Σ^1_1 complete amounts to showing that listing the isomorphism types of the structure is as hard as listing the isomorphism types of *any* countable structure, and hence there can be *no* invariants which would simplify the isomorphism problem. This gives a computational "proof" that there are no reasonable invariants, since, like dimension for vector spaces, invariants inevitably will simplify the problem (Otherwise what is their point?). Hence logic blended with group theory allows us to answer Fuchs' question of whether there are invariants for general torsion-free abelian groups. In contrast, for computable completely decomposable groups [Bae37] the isomorphism problem is merely Σ^0_7 [DM14]. The result means that the Baer invariants [Bae37] for such groups are somewhat close to being Turing computable. Since Σ^0_7-hardness is an open question, we still have to understand how close to computable they are (precisely). The relativized version of this result can be stated in purely topological terms using the Borel hierarchy, confirming that the class of such groups is (somewhat) algebraically tame.

The nicest algebraic classifications sometimes lead to an injective enumeration of the isomorphism types in the class, that is, a computable enumeration of computable members of the class in which isomorphism types are not repeated. For example, one can easily enumerate all isomorphism types of finitely generated abelian groups without repetition. All such groups naturally have a solvable word problem, therefore looking at the computable members of the class is not really a restriction. The classification of finite simple groups also leads to an injective enumeration of the isomorphism types of the class. For more examples, see [LMS18]. As a consequence of the result of Downey and Melnikov [DM14] discussed above, with the help of a few iterates of the Halting problem we can produce a list of all isomorphism types of computable completely decomposable groups without repetition (recall that the isomorphism problem for this class is merely Σ^0_7). In contrast, it follows from [DM08] that no finite – indeed, no recursive transfinite – iterate of the Halting problem is capable of enumerating all isomorphism types of computable torsion-free abelian groups without repetition.

Based on a similar intuition, Goncharov and Knight [GK02] suggested that a class of computable structures is tame if we have a way of *computably listing* the isomorphism types without repetition.

Definition 1.1. [GK02] We say that a class $\mathcal{K}$ of structures has a *Friedberg enumeration* if there is a computable listing $A_1, A_2, \ldots$ of all isomorphism types of the computable members of $\mathcal{K}$ *without repetition*. That is, for every $i \neq j$, we have $A_i \ncong A_j$.

If a class $\mathcal{K}$ has a Friedberg enumeration, then the position n of the structure A_n in a class $\mathcal{K}$ completely describes its isomorphism type. If a class has a Friedberg enumeration, then we can regard it as "classified" in this well-defined sense, in spite of the actual isomorphism problem being possibly quite complex. The inspiration for the name "Friedberg enumeration" comes from Friedberg's proof [Fri58] that the computably enumerable sets can be listed without repetition. Here "isomorphism" means equality, and Friedberg's proof shows that a class can have an undecidable isomorphism problem (the set $\{\langle i, j\rangle \mid W_i = W_j\}$ is Π^0_2 complete) yet there could be a way to give a computable list of all its members without repetition.

As first noted in [GK02], the approach via enumerations tends to be technically rather challenging. There is no algebraic structure on c.e. sets, yet Friedberg's original proof [Fri58] involves several tricks of which were novel – if not revolutionary – at that time. It is perhaps not surprising that even a very mild algebraic content can potentially make the Friedberg enumeration problem a lot more complicated. In our previous work [DMN17] we solved a problem of Goncharov and Knight [GK02] in the positive by producing a Friedberg enumeration of the class of computable equivalence structures. The [DMN17] result was significantly more complex than in the case of c.e. sets since determining whether two computable equivalence structures were isomorphic was Π^0_4, compared to Π^0_2 for determining equality of c.e. sets.

In constructing a Friedberg enumeration one usually has to *dynamically guess* at whether one computable structure is isomorphic to another. The harder the isomorphism problem, the more difficult the guessing becomes. In fact, Goncharov and Knight [GK02] suggested that there are perhpas no Friedberg enumerations of computable equivalence structures because the isomorphism problem is Π^0_4 which is very complicated to guess dynamically. Our solution [DMN17] required an essential use of an advanced priority argument, akin to the $0'''$ technique. In this paper we advance our techniques even further. Using two separate tree constructions combined into a single proof, we will produce Friedberg enumerations for broad classes of abelian groups in which the isomorphism problem could be at an arbitrarily high finite level of the arithmetical hierarchy.

Because of the relationship between abelian groups and equivalence structures, the Friedberg enumeration of computable equivalence structures also gives a Friedberg enumeration of all computable abelian p-groups of Ulm type 1; see Section 2. The goal of the present paper is to prove the following result.

Main Theorem. *For each natural number $n > 0$ there is a Friedberg enumeration of all computable abelian p-groups of Ulm type $\leq n$.*

We remark that this gives the first known examples of *algebraically* nontrivial classes with infinite members having a Friedberg enumeration. We emphasize that the groups from the Main Theorem are *not* necessarily reduced. It can be shown

that, for each fixed n, the reduced members of the class do not possess a Friedberg enumeration [GK02]. However, the fact that the groups are not reduced also adds a great deal of complexity to the argument. It is not hard to show that the isomorphism problem for groups in the theorem is Π^0_{2n+2}-complete. We will use a modification of the jump inversion technique from [AKO] to partially reduce the situation to the case of equivalence structures. Our proof will rely on two tree arguments which will be non-trivially combined. The main ideas in this argument are built upon our previous work on equivalence structures, as well as the work of Ash, Knight and Oates [AKO], along with some new devices we introduce here. We leave open the question of whether there is a Friedberg enumeration of the class of abelian p-groups of greater Ulm types, e.g., of all groups of type $< \omega$.

Countable abelian p-groups are exactly the Pontryagin duals of abelian pro-p groups. Pontryagin duality is injective on the isomorphism types of countable abelian and compact abelian groups [Pon34]. Thus, the Ulm invariants of an abelian p-group give rise to pro-Ulm invariants of the respective pro-p dual; see, e.g., [Kie13] for an explicit definition. In [Mel17] the second author showed that the functor uniformly maps computable abelian p-groups into recursive pro-p groups [Mel17], bijectively on isomorphism types. We have:

Corollary 1.2. *For each natural number $n > 0$ there exists a Friedberg enumeration of recursive pro-p abelian groups of pro-Ulm type $\leq n$.*

2. Preliminaries

In this paper all groups are at most countable abelian p-groups for some fixed p. Recall that the p-height $h_p(a)$ of an element a of an abelian p-group is the supremum over all n such that $p^n y = a$ has a solution in the group. Given an abelian p-group A, define $A' = \{a \in A \mid h_p(a) = \infty\}$, $A^{(\delta+1)} = \big(A^{(\delta)}\big)'$, and take the intersection of A_β over $\beta < \alpha$ for a limit ordinal α. For a countable A, the sequence must stabilise at some countable ordinal α called the Ulm type of A; in this case $A^{(\alpha)}$ is equal to the maximal divisible subgroup of A. It is well-known that the maximal divisible subgroup of A detaches as a direct summand of A, and also itself splits into a direct sum of quasi-cyclic groups $\mathbb{Z}_{p^\infty}$. Here $\mathbb{Z}_{p^\infty}$ is the direct limit of the linearly ordered system of all finite cyclic p-groups under the natural inclusion.

If $\alpha \leq 1$, meaning that $A' = A''$, then the old result of Kulikov (see the book [Kur60]) implies that the p-group A splits into the direct sum of its finite cyclic and infinite quasi-cyclic subgroups. Thus, each group of Ulm type ≤ 1 can be naturally associated with an equivalence structure. Under this correspondence, a cyclic or quasi-cyclic summand $\mathbb{Z}_{p^\lambda}$ is replaced by an equivalence class of size λ. Note that this functor is well-behaved on isomorphism types because any two complete decompositions of Ulm type 1 abelian p-groups are isomorphic (as decompositions). The functor is also clearly bijective on isomorphism types.

Remark 2.1. *It is clear that passing from an equivalence structure to the corresponding group is a uniformly effective process. Having this observation in mind, we will often identify an equivalence structure with the respective group. It is far less clear that going from Ulm type 1 groups to equivalence structures is also a uniformly effective procedure* [MN18]*, but we will not need this fact in our proof. It is however obvious that in the case when the abelian p-group is finite, we can find its*

complete decomposition and thus reconstruct the isomorphism type of the respective equivalence structure. We will use this observation without explicit reference.

The *Ulm factors* $A^{(\delta+1)}/A^{(\delta)}$ of A are themselves of Ulm type 1, and therefore they can be described by invariants similar to those for equivalence structures. The ordinal sequence of such invariants indexed by $\delta < \alpha$ gives the Ulm invariant of A; the invariant completely describes the isomorphism type of A [Kap69].

While groups of Ulm type 1 are very similar to equivalence structures, groups of higher Ulm type resemble trees. We will use the technique of p-basic trees to work with abelian p-groups having Ulm type larger than 1.

Definition 2.2 ([Rog77])**.** *A* p-basic tree *is a set* X *together with a binary operation* $p^n \cdot x$ *of the sort* $\{p^n : 0 < n < \omega\} \times X \to X$ *such that:*

(1) *there is a unique element* $0 \in X$ *for which* $p \cdot 0 = 0$,
(2) $p^k \cdot (p^m \cdot g) = p^{k+m} \cdot g$, *for all* $g \in X$ *and* $k, m \in \omega$, *and*
(3) *for every element* $x \in X$, *there is a natural number* n *with* $p^n \cdot x = 0$.

If a prime p is fixed, then one can think of a p-basic tree as a rooted tree with 0 being the root. Given a p-basic tree X, one obtains a p-group $G(X)$ as follows: The set $X \setminus \{0\}$ is treated as the set of generators for $G(X)$, and we add $px = y$ into the collection of relations if $p \cdot x = y$ in X. Every countable abelian p-group is generated by some p-basic tree [Rog77]. Each element of the group $G(X)$ can be uniquely expressed as $\sum_{x \in X} m_x x$, where $m_x \in \{0, 1, \ldots, p-1\}$.

Non-isomorphic trees can produce isomorphic p-groups. Here we will not give a complete description of the congruence relation $\sim$ on rooted trees which is defined by the rule: $T_0 \sim T_1$ iff the groups $G(T_0)$ and $G(T_1)$ are isomorphic. See [Rog77] for a detailed analysis of $\sim$.

Suppose that T is a p-basic tree viewed as a rooted tree. Consider the following procedure:

"Take a simple chain extending $v \in T$, detach it, and
attach this chain to the root of T."

The procedure is called *stripping*. If the tree rank of v does not change after stripping, then the stripped tree T_1 and the original tree T give rise to isomorphic p-groups: $G(T_1) \cong G(T)$. This process can be iterated. Informally speaking, one can replace infinitely many simple chains at once (while preserving tree ranks), and obtain a *fully stripped tree* representing the same group. For example, a fully stripped tree for a reduced p-group of Ulm type 1 is just a collection of finite simple chains attached to 0.

Using this technique, Ash, Knight, and Oates [AKO] proved the following important result. The definition of a limitwise monotonic function can be found in [KKM13, DKT11]. We will not need this notion in the proof of our main result.

Theorem 2.3 (Ash, Knight, and Oates [AKO]; Khisamiev [His81, Khi92])**.** *Suppose that* A *is a countable reduced* p*-group of Ulm type* $N < \omega$. *Then the following conditions are equivalent:*

(1) A *has a computable copy.*
(2) A *has a computable* p*-basic tree representing it.*
(3) (a) *For every* $i < N$, *the character* $\chi(A_i)$ *is a* Σ^0_{2i+2} *set, and*

(b) *for every $i < N$, the set*

$$\#A_i := \{n : (n, 1) \in \chi(A_i)\}$$

is $\mathbf{0}^{(2i)}$-limitwise monotonic.

The basis of induction in the proof of the theorem above essentially says that abelian p-groups of Ulm type 1 have the same computability-theoretic and algebraic invariants as computable equivalence structures; this case is rather simple [His81], see also surveys [Mel14, Khi98]. In particular, it follows that the Friedberg enumeration of all computable equivalence structures produced in [DMN17] can be uniformly transformed into a Friedberg enumeration of *all* computable abelian p-groups of Ulm type 1. In this enumeration each group has a computable complete direct decomposition into cyclic and quasi-cyclic summands.

Call an abelian p-group H of Ulm type 1 *proper* if it is reduced (i.e, $H' = 0$) and furthermore the sizes of the finite cyclic summands in its full direct decomposition are unbounded in size. The inductive step in the proof of (3) $\to$ (2) of Theorem 2.3 uses a functor that allows us to uniformly pass from a Δ^0_3 abelian group F represented by a p-basic tree and a proper H to a computable abelian p-group $T_H(F)$ with the properties $(T_H(F))' \cong F$ and $T_H(F)/(T_H(F))' \cong H$. By induction, we will have already proven that F can be represented by a Δ^0_3 p-basic tree. It is well-known that each Δ^0_3-tree can be represented as a Π^0_2-subtree of $\omega^{<\omega}$ uniformly. We will identify F with the respective Π^0_2-tree. We also identify an equivalence structure with the respective p-group; under this correspondence, an equivalence class of size n will represent a cyclic summand of order p^n.

Proposition 2.4 (Ash, Knight and Oates [AKO])**.** *There is a uniform procedure which given a computable copy of a proper H and a Π^0_2 p-basic tree F, outputs a computable p-basic tree $T_H(F)$ with the properties $(T_H(F))' = F$ and $T_H(F)/(T_H(F))' \cong H$.*

Remark 2.5. *Why do we need the proof outline below? We will need a minor modification of the original description of $T_H(F)$. The description of $T_H(F)$ and the verification that it works are not difficult but rather tedious. Although the modification mentioned above is simple, it is perhaps impossible to define it without first giving a general outline of what the original functor does.*

Also, there will be one new and essential case in the verification of the (modified or not) $T_H(F)$. This is the case when H is not proper and has almost every class infinite. Finally, the paper [AKO] *has never been published because the authors did not know Khisamiev had published a similar result a few years before them; an unpublished but ready for publication manuscript* [AKO] *is available upon request. The problem with Khisamiev's published proof* [Khi92] *is that it does not use p-basic trees and is extremely hard to follow, verify, or modify. Although a description of $T_H(F)$ using p-basic trees can be found in* [Mel14, DMN16]*, we decided to give an extended outline of the original Ash-Knight-Oates strategy. See* [DMN16, Mel14, AKO] *for technical details and further discussion.*

Proof outline of Proposition 2.4. The idea behind $T_H(F)$ is as follows. If a node $x \in \omega^{<\omega}$ looks in F (when represented as a Π^0_2 p-basic tree), then we make progress in driving its tree-rank to infinity. This is done by attaching more and more external finite simple chains to x; the sizes of the new simple chains we attach are taken from the sizes of classes or summands in H. (Recall that by Remark 2.1 we can

identify H with the equivalence relation representing its decomposition). If x looks like it is not in F we stop attaching new chains to x, but continue updating the sizes of the already attached simple chains.

Consider the easy but illustrative case in which the set of sizes of classes in H is computable with no repetitions. Then it is not hard to produce $T_H(F)$ by implementing the idea above. However, even in this simple case we must be somewhat careful of the Σ^0_2 outcome in which $x \notin F$. We illustrate this situation in the following example.

Example 2.6. Suppose $px = 0$, i.e., it is an immediate successor of the root node in $\omega^{<\omega}$. Assume that the Π^0_2 predicate has "fired" on x, i.e. a further instance of the Π^0_2 event "$x \in F$" has been observed. Suppose that we have used a simple chain of length 3 on x. This chain corresponds to a class of size 3 in H. However, imagine that we will never get to add further finite simple chains to x because the predicate will never fire again on x. As the result, we will end up with a finite simple chain of length 4. In the group that we will have constructed, it will correspond to a cyclic summand of size p^4. But there could be no equivalence class of size 4 in H, and thus $T_H(F)/(T_H(F))' \ncong H$.

This problem is quite easy to overcome. Instead of adjoining a chain of length 3 to x, attach a chain of length 2. If later x fires again, use some longer class from H, say, of size 17. Attach a chain of length 16 to x and extend that old chain of length 2 attached to x by one extra node. This way we will form a simple chain of length 3 having a longer chain next to it. At this stage, the subgroup generated by all the mentioned nodes will be isomorphic to $\mathbb{Z}_{p^{17}} \oplus \mathbb{Z}_{p^3}$ which is consistent with the sizes in H.

Apart from the subtlety illustrated in the example above, the case when the sizes of classes in H are computable is quite straightforward. However, in general the sizes of classes in H are merely *limitwise monotonic*, meaning that they are Δ^0_2 and furthermore our current guess on the size of a given class will increase with time. Also, we must not forget about repetitions that may occur among the sizes of classes in H. Therefore, the general case needs more care. However, we do know that all the chains that we ever attach will eventually stop growing, because H has no infinite classes. We must avoid problems similar to the one discussed in the example above. This is again done by always choosing a fresh new chain – *larger than all the classes we've used so far* – and attaching it next to our chain. We illustrate the extra effects associated with limitwise monotonicity in the example below.

Example 2.7. We use the notation from the previous example. As before, $px = 0$, i.e., x is an immediate successor of the root in $\omega^{<\omega}$. Assume that we currently have two simple independent chains attached to x, one of length 3 and the other of length 16, as in the end of the previous example. Currently, the subgroup generated by all the mentioned nodes will be isomorphic to $\mathbb{Z}_{p^{17}} \oplus \mathbb{Z}_3$ which is currently consistent with the sizes in H_s. However, assume that the class of size 3 grows. If its size is still below 16 we could simply mimic this in the group by extending the corresponding simple chain. Suppose it reaches the size of 16, so we have two simple chains of length 16 attached to x. Now imagine this same class grows again and becomes size 17. For the sake of maintaining the right isomorphism type, we want to keep the second chain longer than the first one. However, the second chain could be copying

a class in H which is either too small or too slow. In this case we do a *retargeting* by choosing a very large and fresh class in H and associating it with this chain that is currently refusing to grow. While we are searching for an appropriate new class we of course cease any further action. Once such a large class is found, we can grow the second chain.

Remark 2.8. We note that the formal implementation of the retargeting as found in [AKO] hides some of the combinatorics into a careful choice of a *strictly increasing limitwise monotonic function* [Mel14]. Such a limitwise monotonic function can be extracted from H uniformly. The retargeting procedure we have sketched above is equivalent to the uniform use of a strictly increasing limitwise monotonic function.

We will have to reintroduce the sizes (of classes in H) that we abandon due to retargeting. Recall that the size of classes in H might contain repetitions, but this can be sorted out by a careful bookkeeping. The following simple properties of retargeting – equivalently, of using a strictly increasing limitwise monotonic function – will be quite important later:

Property 2.9. Whenever a simple chain obtains a new image in H, that chain grows in size.

Property 2.10. A chain $\mathcal{C}$ is retargeted only if some earlier introduced chain *which copies a class having a smaller index* has grown to a certain size less or equal to the current size of the chain $\mathcal{C}$. (See the example above for one such instance.)

Recall F always contains at least the root because the respective group must contain 0.

Property 2.11. We re-introduce (the size of a) class that has been abandoned due to retargeting, as follows. We attach a new simple chain of the right length to the root and associate it with the class. The new simple chain will have its "priority" equal to the index of the class that it is copying. Thus, only finitely many other chains can potentially upset the chain in the future by growing too long.

In order for the above to work, it is crucial that H contains only finite classes, and that it has arbitrary large classes. In particular, an inductive argument can be used to show that every search will be successful, and that the retargeting of each chain will eventually stabilise. The "injury" in the construction of Ash, Knight, and Oates is finite. While we have to take care when implementing this for every x in $\omega^{<\omega}$, there are no further surprises; see [Mel14, AKO, DMN16] for the details and a further discussion. □

3. The basic strategy

If $n = 1$ then we have a Friedberg enumeration of all computable equivalence structures, and thus of all abelian p-groups of Ulm type 1. It is clear that the groups in the list are uniformly represented by p-basic trees, generated by the corresponding equivalence structure. Therefore, assume $n > 1$ throughout the rest of this paper.

3.1. **Notation.** Inductively, fix a $0''$-computable Friedberg enumeration $(F_i)_{i\in\omega}$ of all $0''$-computable abelian p-groups of Ulm type $\leq n-1$; furthermore, assume that they are represented by $0''$-computable p-basic trees.

Remark 3.1. The subscripts in $F_0, F_1, F_2, \ldots$ should be understood as the Δ^0_3-index of the respective structures. Equivalently we may think of i as the index of the Turing functional which maps $\emptyset''$ to the characteristic function of the tree F_i. However, it is a well-known fact that each Δ^0_3-tree can be effectively represented as a Π^0_2-subtree of $\omega^{<\omega}$. Therefore we shall identify F_i with the respective Π^0_2-tree that represents it, and think of i as a Π^0_2 index instead.

Definition 3.2. If F has a non-zero terminal node than we say that F is *true*. Note that this is equivalent to saying that the reduced part of the corresponding p-group is non-trivial.

The statement "F_i is true" can be described by the formula:

$$(\exists x)\,[x \in F_i \ \wedge\ x \neq \langle\rangle \ \wedge\ (\forall y)(y \supset x \to y \notin F_i)]$$

has complexity Σ^c_4.

The second main result in [DMN17] says that there is a Friedberg enumeration $(E_i)_{i\in\omega}$ of all infinite equivalence structures, and thus of infinite abelian p-groups of Ulm type 1. We let E_i be the corresponding abelian p-group. Under this correspondence, an equivalence class of size n will represent a cyclic summand of order p^n. By Remark 2.1, E_i can stand for an abelian p-group (represented as a tree) and an equivalence structure for E_i, and these will be used interchangeably.

Definition 3.3. If E_i corresponds to a reduced abelian p-group in which p-heights are unbounded than we say that E_i is *proper*. Equivalently, E_i is proper if it consists only of finite classes and the sizes of its classes are unbounded.

The sentence saying that E_i is proper has complexity Π^c_3; just state that each class is finite (Π^c_3) and that there are arbitrarily large classes (Π^c_2).

Remark 3.4. We briefly explain why we need true groups. We will be using a modification of the Ash-Knight-Oates jump inversion T. If an abelian p-group F is true then the Ulm type of $T_E(F)$ will be *at least* 2. This is because F has a terminal nonzero node. In $(T_H(F))'$ it will become a non-zero terminal node, witnessing that $(T_H(F))'$ is not divisible. This elementary observation will be absolutely crucial in the construction, since all the "junk" produced by various strategies will be of Ulm type at most 1.

We index classes of a computable equivalence structure by natural numbers according to the order at which they appear in the enumeration of the equivalence structure. Typically classes having a smaller index will have a higher "priority".

3.2. **The setup.** Based on the Friedberg enumerations $(F_i)_{i\in\omega}$ and $(E_j)_{j\in\omega}$ described above, fix the effective listing $(F_i, E_j)_{i,j\in\omega}$. Recall that each F_j is a Π^0_2 subtree of $\omega^{<\omega}$ identified with its index, and each E_j is a computable infinite equivalence structure which can be viewed as a p-group in which a complete decomposition is known. The intended role of this enumeration of pairs is informally explained in the remark below.

Remark 3.5. If F_i is true and E_j is proper, then we can *uniformly* produce an abelian p-group $T_{E_j}(F_i)$ of Ulm type at most n such that $(T_{E_j}(F_i))' \cong F_i$ and $T_{E_j}(F_i)/(T_{E_j}(F_i))' \cong E_j$. Since $(F_i)_{i\in\omega}$ and $(E_j)_{j\in\omega}$ were Friedberg, the Ulm classification theorem implies that unequal pairs correspond to non-isomorphic groups *provided that these pairs consist of true and proper members*. Furthermore, the Ulm classification theorem implies that each group of Ulm type k, $1 < k \leq n$, has the form $T_E(F)$ for some true F and proper E. The main idea is to guess on

trueness and properness and enumerate the resulting $T_{E_j}(F_i)$, while the groups of Ulm type 1 will be enumerated elsewhere. The guessing on trueness and properness requires an implementation of a non-uniform $0'''$ technique. Various outcomes of the guessing will be producing "junk" groups that will have finite reduced part, and therefore will be of Ulm type 1. We shall merge two $0'''$ constructions – one from [DMN17] for Ulm type 1 groups and the other for higher Ulm types $\leq n$ – and let them share the "junk". Managing this "junk" is in the hart of the proof.

3.3. **A preliminary discussion of the guessing.** Given (F_i, E_j) we need to test whether F_i is true and E_j is proper. We suppress the subscripts in F_i and E_j and write (F, E) throughout.

We start with the simpler Π^0_3 guessing on properness of E. It is convenient to discuss the outcomes of this guessing early on. We say that an equivalence structure is *eventually bounded* if there is an n such that all classes having indices $> n$ are bounded in size by n. The definition has complexity Σ^c_2. An eventually bounded structure may have a few infinite classes. Note that an equivalence structure is proper iff is not eventually bounded and does not contain infinite classes. The preliminary description of the outcomes of this guessing is:

- pi_j This is a Π^0_2 outcome that says that E is *not* eventually bounded and the jth class in E is infinite.
- Π This is a Π^0_3 outcome that says that E is proper.
- fin This is a Σ^0_2-outcome which says that E is eventually bounded.

It is clear that the outcomes are exclusive and cover all possible cases. The outcomes and the guessing itself will have to be significantly modified.

We must also blend this guessing above with the guessing on trueness of F. Recall that the sentence saying that F is true has complexity Σ^0_4, we represent the respective uniform predicate as $\exists z \Pi^0_3(z)$. *As usual, we assume that the measured predicates satisfy the property of the uniqueness of existential witnesses.*

Each node τ in the tree of strategies for (F, E) (to be defined) will be associated with a *triple* (F, E, z), where each such z is interpreted as a potential existential witness for $\exists z \Pi^0_3(z)$ approximating trueness of F. The outcomes of τ will be the same as above, but the Π^0_3 outcome will also incorporate guessing on $\Pi^0_3(z)$, and the Σ^0_3-counterpart of $\Pi^0_3(z)$ will be spread naturally over the Π^0_2 outcomes pi_j, $j = 0, 1, \ldots$. To describe the details of the guessing and its actual outcomes we must explain the basic strategy associated with (F, E, z) in isolation.

3.4. **The strategy for (F, E, z) in isolation.** The strategy has two tasks that will be performed simultaneously.

3.4.1. *The first task: the description of H.* The strategy dynamically transforms the computable infinite equivalence structure E into a computable equivalence structure H with the properties:

i. If E is not eventually bounded, and one of the two conditions holds
   (i.1) E has infinite classes, or
   (i.2) F looks not true according to $\Sigma^0_3(z)$ (see the previous subsection),
   then H has infinitely many classes with a.e. class infinite.
ii. If E is proper and F is true then $H \cong E$.
iii. If E is eventually bounded then H is finite.

Furthermore, we will ensure that in i. the number of finite classes in H produced by τ is specific to the node τ. We delay the detailed description of H and the

verification of its properties $i.-iii.$ until Section 4. For now, we take these properties as granted.

3.4.2. *The second task: the description of $T_H(F)$.* The second task of the strategy is producing $T_H(F)$, which is a modified version of the Ash-Knight-Oates jump inversion. (Strictly speaking, for a fixed H, the functor T_H inverts two jumps.) We will explain the jump inversion in detail later. At this point we need to know the following properties of the jump inversion that will be verified in Section 4. We identify an equivalence structure with the direct sum of cyclic and quasi-cyclic p-groups in which the cyclic summands $\mathbb{Z}_{p^n}$ naturally correspond to equivalence classes of size n.

a. If a.e. class in H is infinite, and there are infinitely many such classes, then $T_H(F) \cong H$, where the latter is viewed as a direct sum of cyclic and quasi-cyclic groups.
b. If H is proper, then $T_H(F)$ will be a computable p-basic tree with the properties $(T_H(F))' = F$ and $T_H(F)/(T_H(F))' \cong H$.
c. If H is finite then so is $T_H(F)$.

We will give the algebraic details of the Ash-Knight-Oates jump inversion in the section below. We will also describe a modification to the inversion that will be necessary in the case of many strategies working together on the tree of strategies. For now, take a., b., and c. as granted.

3.5. **The outcomes.** We have the following description of the outcomes of the basic strategy (in isolation) associated with (E, F, z):

pi_j: This is a Π^0_2 outcome which says that:
- E has arbitrarily large classes of arbitrarily large indices, and
- either the jth class in E is infinite, or F looks not true as witnessed by instance j of the measured Π^0_3-predicate having parameter z.

In this case the strategy will produce a computable $T_H(F) \cong H$ which is a direct sum of at most finitely many cyclic and infinitely many quasi-cyclic groups. Different j will give different isomorphism types of H. Furthermore, we will ensure that different τ on the tree will always produce non-isomorphic H under their Π^0_2 -outcomes (see the next section).

Π: This is a Π^0_3 outcome that says that E is proper and F is true. In this case the strategy outputs a computable basic tree $T_H(F)$ with the properties $(T_H(F))' = F$ and $T_H(F)/(T_H(F))' \cong H \cong E$. Furthermore, since $E \cong H$ is proper and F is true, the Ulm type of $T_H(F)$ is at least 2.

fin: This is a Σ^0_2-outcome which says that almost all classes in E are bounded in size. In this case the strategy will produce a finite abelian p-group $T_H(F)$.

To finalise the description of the basic strategy we must give a detailed description of H and $T_H(F)$ and verify their claimed properties.

4. The description of H and $T_H(F)$

We describe the intended construction of H which is sufficient in presence of only one strategy. Then we explain a modification invented in our previous work [DMN17]. The modification will be highly convenient in the general case of many strategies.

4.1. **The intended construction of H.** We are given an infinite computable equivalence structure E and we must produce a computable equivalence structure H with the properties:

i. If E is not eventually bounded, and one of the two conditions holds
   (i.1) E has infinite classes, or
   (i.2) F looks not true according to $\Pi^0_2(j,z)$ (see the previous subsection),
   then H has infinitely many classes with a.e. class infinite.
ii. If E is proper and F is true then $H \cong E$.
iii. If E is eventually bounded then H is finite.

First, we give a construction that is sufficient to satisfy i., ii., and iii. Then we explain how to modify the construction to ensure that in i we can control the number of finite classes in H. (This will be very convenient when many basic strategies are put into a tree of strategies.)

We write $[m]_L$ for a class of an equivalence structure L with index m. Say that a stage s is expansionary if the parameter $\max\{|[i]|_{E_s}, s' \leq i \leq s\}$ has increased from the previous expansionary stage s'. The parameter measures whether the structure E has arbitrarily large classes with arbitrarily large indices. The module below acts only at expansionary stages.

4.1.1. *Construction.* At every stage, each class in H_s is matched with a class in E_s. Suppose at a stage $[n]_H$ is copying $[i]_E$. If $[i]_E$ grows in E at the next stage or the ith Π^0_2 instance of the Σ^0_3 predicate "F is not true (z)" fires, then perform the following action. *Initialise* (to be clarified) each class $[k]_H$ in H that satisfies

(1) $k > n$, and
(2) $[k]_H$ has been copying a class $[j]_E$ with $j \geq i$.

Each initialised class will be grown by one extra element and assigned to some large enough fresh class in E (if it exists). (Until such large enough classes are found the strategy will cease its action.) Then each currently abandoned classes of E will be assigned to a new fresh class in H. This ends the construction.

4.1.2. *The verification of i., ii, and iii.* To see why iii. holds, recall that the module constructing H acts only at expansionary stages. Since there will be only finitely many such, H will be left finite. To check i. and ii., note that each initialised class must grow. A class can be initialised only due to some larger index class growing or due to some higher priority Π^0_2 instance of the predicate "F is not true (z)" firing; furthermore, in the former case this larger H-index class must be copying a larger E-index class. There are only finitely many such. Thus, if all classes in E are finite and F looks not true according to instance z, then each class can be initialised only finitely often. Also, a class in E has to change its clone in H only if a class with a smaller E-index grows. Therefore, ii. follows by induction. To check i., assume that $[i]$ is the left-most class of E – i.e., the one with the smallest index – that grows to infinity. Since all classes to the left of it are finite, there will be a stage after which the class is stably assigned to a clone in H, call this clone $[k]$. There will be at most finitely many classes of H to the right of $[k]$ that are controlled by classes in E having index less than i. All the rest will be initialised infinitely often. Since E has arbitrary large classes with arbitrary big indices, every search for a new appropriate image for an initialised class will be successful. In particular,

E has infinitely many classes, and therefore so does H. Since each initialised class must grow, co-finitely many classes of H will be infinite.

4.2. **The modification to H.** We adopt a rather convenient modification to the construction of H suggested in [DMN17].

Modification 1 (Informal summary)**.** For each fixed τ, we will reserve as certain infinite collection of finite intervals which will be unique to τ. Ensure that the number of finite classes in the structure below a Π^0_2-outcome will fall into one of these specific intervals.

We give details. Each such τ will be associated with a computable sequence of disjoint and increasing in size finite intervals $I^\tau_0, I^\tau_1, \ldots, I^\tau_n \ldots$, where the intervals are specific to τ and do not intersect the intervals associated with other strategies.

The role of these intervals is as follows. Recall that τ guesses on properness of its equivalence structure E, and it simultaneously builds H. Initially, we agreed that pi_j corresponds to the case when the left-most infinite class of E has index j. In this case we would produce an infinite junk structure H in which the number of finite classes depends on j and on the number of times the outcome pi_j is initialised in the construction. If we keep initialising $\tau\hat{}pi_j$ then this potential number of finite classes will be increasing, due to the image of the class $[j]$ of E being re-located in H. We must find a new definition of the outcome pi_j, so that H built by $\tau\hat{}pi_j$ will have the number of finite classes in one of these intervals I^τ_k, where k depends on j and on the number of times the strategy has been initialised. To succeed we must some up with a way of bounding the number of finite classes in H from below and also of controlling this number from above.

To make sure that number of finite classes in H is sufficiently large we unite classes in E in *clusters*. The idea is as follows. If the first cluster is $([0], [1], [2])$ and the second Π^0_2-outcome is the true outcome, then the structure H will have *at least* 3 finite classes. Write (j, n) for the cluster $\{[j], [j+1], \ldots, [n-1], [n]\}$. The clusters are naturally ordered according to the index of their left-most class j which is also called the index of (j, n). Instead of monitoring the growth of individual classes in E, we ask whether at least one of the classes in the cluster $\{[j], [j+1], \ldots, [n-1], [n]\}$ grows; in this case we initialise the clusters having their index $\geq j$. We explain the initialisation below.

To bound the number of finite classes in H from above, we adjust the notion of initialisation from 3.4.1 as follows. Similarly to what we had before, if at least one class in the cluster grows then all classes in the clusters having index $\geq j$ will abandon their current images in H. As before, the abandoned will be forced to grow, and then they will be assigned to fresh classes having higher indices. *Additionally,* we allow the cluster (j, n) to initialise some classes in H having their H-index smaller than of those classes in H copying by the cluster. This is done as follows. The set of such classes is defined by induction on the step; we say that these classes to the left are *controlled* by the cluster. These will include all classes currently in H to the left of the right-most H-image of a class in the cluster with the exception of those controlled by higher priority clusters.

There is still one difficulty that needs to be addressed. The problem occurs due to the accumulation of classes controlled by smaller index higher priority clusters. The number of such classes may grow too big and will eventually not allow us to

stay within one of the I^τ_j. This situation happens only if lower index higher priority clusters keep firing.

Example 4.1. Imagine there are too many classes controlled by the first three clusters, say 800, and the true outcome says that the forth cluster is initialised infinitely often. We must ensure that H will end up with somewhere between 100 and 1000 classes; the latter corresponds to the interval I_2. The third cluster could contain merely 300 classes, but it has no control over those 800 classes. In the worst case scenario we may end up with $800 + 300 = 1100$ finite classes. Thus, when we had a chance during initialisation, we should have increased the size of the third cluster to be very large. From the perspective of the 4th Π^0_2-outcome, we should have switched from targeting I_2 to targeting I_3 which itself is larger. We should have made the size of the third cluster equal to, say, 10000. This would allow us to target the interval I_3; targeting this larger interval could be equivalent to saying that H contains somewhere between 10000 and 1000000 of finite classes.

The sizes of the clusters should be adjusted dynamically, as follows. Recall that the number of finite classes produced under a Π^0_2-outcome must be within one of the intervals $I^\tau_0, I^\tau_1, \dots, I^\tau_n \dots$ which increase in size with n increasing. If a cluster is initialised, then we will increase the size of the next cluster. We use initialisation to increase the number of classes in the next cluster. By choosing the number of classes in the cluster large enough we can successfully hit the next, larger interval I^τ_{j+1} in the case if this increased cluster is the left-most cluster that is eventually stable.

In [DMN17] all intervals and clusters have an explicit numerical description. We however strongly suspect that the reader should have no problem in understanding the technique from the explanation above. Note that the use of clusters does not upset the outcomes. More specifically, if E is proper then each cluster will be eventually stable, and every class in H will find a stable pre-image in E. Thus, we will end up with $H \cong E$. If H has at least one infinite class, then the cluster that contains the class and all clusters to the right of it will be initialised infinitely often, but the clusters of smaller indices will eventually become stable, thus guaranteeing that the number of finite classes falls into one of the admissible intervals specific to τ. Finally, if E is eventually bounded then the process of building H will be stuck at some finite stage, and thus H will be finite.

4.3. **The modified Ash-Knight-Oates strategy.** We adopt the following modification to the original strategy of Ash, Knight, and Oates (see the preliminaries):

Modification 2. At every stage at which the Ash-Knight-Oates module initiates a new search or makes a change to the p-basic tree adjoin a fresh and very large simple chain to the root of the p-basic tree. Call this extra simple chain *auxiliary*. If the auxiliary chain has just been introduced, then it does not have to copy any class in H. We also initiate a search for a fresh and large enough class in H that can be matched with the auxiliary chain in the future. The module will not act again until the search is finished. When the module acts again (if ever) the chain is handled by the standard Ash-Knight-Oates module according to its instructions. We also attach a very long auxiliary chain to the root of the p-basic tree if the strategy τ gets initialised. In this case the tree will be forever abandoned.

We abuse notation and write $T_H(F)$ for the *modified* Ash-Knight-Oates jump inversion. We will have to apply the jump inversion in the case when H has infinite classes.

Definition 4.2. Call a coputable equivalence structure *an infinite junk* if it has infinitely many classes almost all of which are infinite.

We identify H and the respective direct sum of cyclic p-groups. We also identify a p-basic tree and the group that it represents. The lemma below justifies the description of the outcomes of our basic strategy.

Lemma 4.3. *There is a uniform procedure which, on input a computable copy of an equivalence structure H and a Π^0_2 p-basic tree F, outputs a computable p-basic tree $T_H(F)$ with the properties:*

(1.) *If H is proper then $(T_H(F))' = F$ and $T_H(F)/(T_H(F))' \cong H$.*
(2.) *If H is an infinite junk, then $T_H(F) \cong H$.*
(3.) *If H is finite then $T_H(F)$ is finite, and furthermore its cardinality can be assumed to be larger than any fixed number.*

(Note that none of the clauses requires F to be true.)

Proof. (1.) It should be clear that the modification does not effect the outcome in the case when H is proper, we elaborate on this below. Each auxiliary chain that we introduce will be treated just as any other chain that is searching for an appropriate image; simple long chains that search for an appropriate class in H appeared in the original module as well [AKO]. Since H is proper, we will eventually succeed in finding a long enough class in H that can be matched to the auxiliary chain. Once this is done, the chain becomes indistinguishable from the other many simple chains that we attach to the root 0 according to the non-modified instructions in [AKO]. It will then be treated accordingly. There are no further interferences of the modification with the rest of the module. It follows that in the case when H is proper, the verification of the new module is almost literally the same as the verification of the original construction in Ash, Knight, and Oates [AKO].

(2.) Here the modification plays no significant role either. However, the analysis of this scenario is new because the case of a non-reduced H has never been considered in the literature. Recall that the first few classes of H could be finite, but the rest of the classes are infinite, and there are infinitely many of them.

First, we claim that almost all simple chains that we will ever attach to nodes will grow infinite. Note that a simple chain may never find a stable pre-image among classes in H. However, at each intermediate step we will always succeed in finding a long enough class in H to switch to. Whenever we switch, the chain itself must grow; see Property 2.9. Thus, we will still grow the length of the chain to infinity, even though it may never find a stable image in H. Now consider those simple chains which do find a stable match in H. We claim that all but finitely many of them will grow infinite as well; indeed, the respective classes in H will be infinite for almost all such chains. Such chains will grow infinite by simply copying the respective stable class in H. The analysis also applies to the auxiliary simple chains from the modification. In particular, since we are never stuck at any intermediate step, there are infinitely many such infinite simple chains to be attached to the root. It follows that the divisible part of $T_H(F)$ will have infinite rank.

There will be at most finitely many exceptional chains that correspond to the finite classes in H. There may also be several finite configurations that become simple chains after "stripping" the tree, i.e., up to group-isomorphism [AKO, **?**]. The latter corresponds to parts of the tree being forever abandoned in a Σ^0_2-outcome of the Π^0_2-approximation. Every individual simple chain, as well as each chain involved into an "abandoned" configuration, must grow whenever its image in H switches (Property 2.9). Thus, a chain or a configuration of chains can be finite only if each chain involved into the configuration finds a stable image in H. There are only finitely many finite classes in H, and thus the reduced part of $T_H(F)$ must be finite. Furthermore, we may be forced to switch the image of a given chain only due to some currently shorter class of a smaller index has grown (Property 2.10).

If a finite class in H is skipped in the construction due to retargeting, then it will be re-introduced again in the form of a simple chain attached to the root (Property 2.11). There are only finitely many classes having a smaller index than the index of the finite class. Therefore, by induction, the finite class will eventually find a stable image in the tree, which will be a simple chain of the right length. It follows that the reduced part of $T_H(F)$ will be isomorphic to the reduced part of H (viewed as a p-group).

(3.) This is obvious from the description of the modification, because the auxiliary chain can be to be taken arbitrarily long. It is crucial that the chain does not have to copy any class in H at the stage when it is first introduced. □

5. The tree of strategies for (E, F)

As we have already mentioned, each fixed pair (E, F) will have its own tree of strategies. Every node on the tree will be associated with a (unique) triple (E, F, z), where z is measuring a Π^0_3-instance of the Σ^0_4-predicate which is being approximated. The basic strategy had to be modified, and this will effect the outcomes. More specifically, the left-most outcome pi_0 will have a special role.

5.1. **The actual outcomes.** The general strategy associated with (E_i, F_j, z) has the following outcomes:

pi_0: This is a Π^0_2 outcome which says that one of the first few classes of E is infinite. The number of these few classes measured will depend on the strategy and on its position on the tree. It is equal to the size of the highest priority cluster in E_i. Every time one of these classes in E grows the strategy is initialised by restarting its construction of H and of $T_H(F)$ and abandoning their previous versions.

pi_j, $j > 0$: This is a Π^0_2 outcome which says that:
- E has arbitrarily large classes of arbitrarily large indices, and
- either the jth class in E is infinite, or F looks not true as witnessed by instance j of the measured Π^0_3-predicate having parameter z.

In this case the strategy will produce a computable $T_H(F) \cong H$ which is a direct sum of at most finitely many cyclic and infinitely many quasi-cyclic groups. Because of the modification that we adopted for H, different pi_j will give different isomorphism types of H. Furthermore, we will ensure that different τ on the tree will always produce non-isomorphic H under their Π^0_2 -outcomes (see the next section). The isomorphism type of H

can be reconstructed based on the stage of the construction and under the assumption that pi_j is the true outcome.

Π: This Π^0_3 outcome says that E is proper and F is true. In this case the strategy outputs a computable basic tree $T_H(F)$ with the properties $(T_H(F))' = F$ and $T_H(F)/(T_H(F))' \cong H \cong E$. Furthermore, since $E \cong H$ is proper and F is true, the Ulm type of $T_H(F)$ is at least 2.

fin: This is a Σ^0_2-outcome which says that almost all classes in E are bounded in size. In this case the strategy will produce a finite abelian p-group $T_H(F)$.

The ordering of the outcomes of each such τ is

$$pi_0 < pi_1 < \ldots < \Pi < fin.$$

Every node ending with Π or fin will be terminal in the tree, and there will be no strategy associated with this node. The root of the tree is associated with $(E, F, 0)$. If τ is associated with (E, F, z), then $\tau^\frown pi_j$ will be associated with $(E, F, z+1)$.

5.2. **The current true path. Initialisation.** The definition of the current true path δ_s at stage s will be standard to such constructions. The nodes below the Π^0_3-outcome will be visited between the stages at which the Π^0_2-outcomes are played.

Initialisation will be standard: A strategy is initialised by setting all its parameters undefined and starting to build a new version of $T_H(F)$. The previous version of the group will be left finite forever; it will be put into the *junk collector* (to be explained shortly).

In the construction, we initialise all strategies to the right of the current true path δ_s, with the exception of those which are below the Π^0_3-outcome of the lowest priority strategy that belongs to both δ_s and δ_{s-1}. In other words, nodes below a Π^0_3-outcome cannot be initialised by nodes to the left of it unless a higher priority strategy above it is itself initialised. Also, if τ plays π_0 then it additionally initialises itself.

5.3. **Putting many trees together.** In the construction, there will be infinitely many such trees, one for each pair (E_j, F_i). Also, there will be another tree which will be responsible for enumerating all groups of Ulm type 1. This tree will be taken from [DMN17] without any further modification. All these trees and strategies will be fairly independent, but they will share the same global strategy that manages the junk; see the next section for details.

6. Managing the junk

We call a structure *a junk structure* if it is produced by a τ either due to initialisation or below a Π^0_2- or a Σ^0_2-outcome. Such structures will fall into one of the two special classes below.

(1) Finite junk. These are finite abelian p-groups that are either produced due to initialisation, or are build if the Σ^0_2-outcome fin is the true outcome. Because of the Modification, the cardinalities of such finite groups may be assumed to be fresh and unseen at the stage when they are first introduced; see Lemma 4.3(3). W.l.o.g. we view these groups as finite equivalence structures.

(2) Infinite junk (c.f. Def. 4.2). These are abelian groups of Ulm type 1 whose divisible part has infinite rank. A group of this sort will be produced if the

true outcome of τ is of the form pi_j, $j > 0$; see Lemma 4.3(2). We will view such groups as infinite junk equivalence structures; see Definition 4.2.

The modification to the Ash-Knight-Oates functor implies that different strategies produce different finite junk. The modification to the definition of H combined with Lemma 4.3(2) allows us to make infinite junk specific under each Π^0_2 outcome of each strategy, which is not the left-most outcome. (Recall that the left-most outcome initialises the strategy and thus contributes into producing finite junk.)

6.1. **The infinite junk collector.** The task of this global strategy is to ensure that each isomorphism type of infinite junk structure H is represented in the enumeration. Here the isomorphism type of an infinite junk structure is identified with the isomorphism type of the respective group of Ulm type 1. Note that under each Π^0_2-outcome, which is not the left-most outcome, we have a specific guess on the isomorphism type of the infinite junk $T_H(F)$ produced under the outcome (if it is played infinitely often). Furthermore, by the modification to H, this isomorphism type is specific to the node and the outcome. Call this isomorphism type L. If the outcome is played again at stage s, then we say that L is *active* at s. Otherwise, we say that it is not active.

Initially, we start an enumeration of all isomorphism types of infinite junk structures in the infinite junk collector, independently from the tree. If L becomes active, then we are in the danger of having repetitions, for the following reason. Suppose an enumeration of $L' \cong L$ has already been initiated by the infinite junk collector. When L becomes active, we stop building L' and permanently put the currently finite L' into the finite junk collector (to be explained). We also artificially adjoin a very large cyclic summand to L' to make it look different from all the other finite structures that we have ever seen in the construction. While L is no longer active, we re-introduce its isomorphism type to the junk collector by using a fresh $L'' \cong L$.

6.2. **The finite junk collector.** This global strategy must ensure that every isomorphism type of a finite abelian p-group is represented in the enumeration. As usual, we can identify such groups with finite equivalence relations. Recall that we adopted a modification to $T_H(F)$ which makes it very large and fresh at every stage when more stages are done in its approximation.

Just as with the infinite junk collector, we initially make a few steps in enumerating all finite abelian p-groups currently independently from the rest of the construction. However, new finite abelian groups will appear in the construction. Such groups could be of three different kinds:

(1) Finite abelian groups that are permanently abandoned due to initialisation. According to the modification we adopted in the definition of $T_H(F)$, if a strategy is initialised then its structure gets a very large and fresh simple chain. This will make the isomorphism type of the abandoned finite group unique at the stage.
(2) Finite abelian groups that appear in the collector due to an infinite junk structure being permanently "killed" by the infinite junk collector. The isomorphism type of this abandoned finite group will again be unique at the stage, for essentially the same reason as in the case above. See the description of the infinite junk collector.
(3) Structures that are finite approximations to a $T_H(F)$ of some τ at a finite stage. According to the modification to $T_H(F)$ that we adopted, at every

stage the isomorphism type of the finite $T_H(F)[s]$ will be fresh and unique. This is achieved by artificially adjoining a fresh and long simple chain to the root of the Π^0_2 p-basic tree every time the strategy has to temporarily stop enumerating its $T_H(F)$.

If a group A of a sort (1), (2), or (3) is permanently put into the finite junk collector, then there could not have already been another finite group in the collector isomorphic to A. This is because the isomorphism type of the new group was artificially made "fresh" in each of the three cases. However, in case (3) the group may resume growing, and therefore we will have to simply reintroduce the finite isomorphism type that it had at the previous stage (before it grew again). This is done using some new finite group. We put this new fresh group into the finite junk collector. This group will never become isomorphic to any other finite group in the construction, because all groups produced by τ will have cyclic summands that are too big, and because the finite junk collector itself will never duplicate the finite group at any later stage.

7. The construction

We put all the components together into one construction. The construction will have the following modules:

(1) The finite junk collector which is shared between all other modules. It is responsible for managing the finite junk produced by all other modules. It also is responsible for listing all finite abelian p-groups without repetition (up to isomorphism).
(2) The infinite junk collector, also shared between all other modules. It is responsible for managing the infinite junk produced by all other modules. It is also responsible for making sure that all infinite junk groups/equivalence structures appear in the list without repetitions (up to isomorphism).
(3) For every i, j, there will be a module $T_{i,j}$ which implements the tree of strategies associated with the pair (F_i, E_j).
(4) A module that implements a Friedberg enumeration $(D_k)_{k\in\omega}$ of all equivalence structures, as described in detail in [DMN17]. Each equivalence structure from the list will be identified with the abelian p-group having cyclic and quasi-cyclic summands of the respective sizes. This correspondence is fully uniform. The module is a $0'''$-construction that itself consists of:
    (a) A tree of strategies responsible for producing a Friedberg enumeration of equivalence structures having arbitrarily large finite classes (and any number of infinite classes). As a biproduct of this process some nodes of the tree will be producing infinite junk structures and finite structures. The isomorphism types of infinite junk structures will be again specific to each node and outcome. Each finite structure will be fresh and unique at the stage when it is first introduced by a strategy. Both finite and infinite junk structures will be put into the respective junk collectors.
    (b) An external enumeration of abelian p-groups of Ulm type 1 of the following kinds:
        (i) Groups with finitely many finite summands and finitely many (with at least one) quasi-cyclic summands.

(ii) Groups with infinitely many finite summands which are uniformly bounded in size, and any number of quasi-cyclic summands.

(Recall that the "infinite junk" ones and the finite ones will be listed elsewhere.)

In the construction, we let each module act according to its instructions.

7.1. **The verification.** The most significant components of the verification is contained in the description of the strategy and in the description of the junk collectors; see the respective sections. We summarise these below.

Recall the notation and terminology; see 3.1. Recall also that the main purpose of the tree associated with a pair (F_i, E_j) – where F_j is a Π^0_2 p-basic tree and E_j is an computable infinite equivalence structure – was to guess whether F_i is true and E_j is proper. This was a Σ^0_4-guessing overall, so it had to be split into Π^0_3-attempts. If one of the Π^0_3-outcomes is the true outcome of the guessing, then the tree of strategies terminates there. There will perhaps be infinitely many nodes to the left of the terminal node which will be visited infinitely often, but every single node to the left can be visited only finitely many times. There will be finitely many above that are visited infinitely often. These will produced at most finitely many infinite junk structures and perhaps infinitely many finite junk structures due to initialisation. Under the true Π^0_3-outcome we end up with $T_{E_j}(F_i)$ having Ulm type > 1 and at most n, with the properties $(T_{E_j}(F_i))' \cong F_i$ and $T_{E_j}(F_i)/(T_{E_j}(F_i))' \cong E_j$. This was carefully checked in the description of the strategy and its verification.

If none of the Π^0_3 outcomes of the tree is the true outcome, then perhaps one of the finitary Σ^0_2 outcomes fin is the true outcome of some (thus, every) node. This corresponds to E being eventually bounded. The tree terminates under this outcome. Furthermore, eventually the first node will also be playing the outcome fin, therefore all actions n the tree will eventually be stopped forever. Thus, overall the tree will end up producing finitely many finite groups, all of which will be put into the finite junk collector.

It could be also the case that the true path is infinite; this corresponds to the situation when all the true outcomes of all nodes are the Π^0_2-outcomes. In this case the tree will produce infinitely many infinite junk groups and perhaps infinitely many finite groups (due to initialisation). These will be handled by the respective global collectors.

Recall that $(F_i)_{i\in\omega}$ and $(E_j)_{j\in\omega}$ were Friedberg; see 3.1. As we have already discussed above, all groups of Ulm type between 2 and n will be listed this way if we allow i and j to range over ω. The trees corresponding to various i and j will also produce infinite and finite junk, but those will be put into the respective collectors.

It is crucial that no other isomorphism type of groups having Ulm type 1 other than the junk structures will be produced by the trees associated with (F_i, E_j), $i, j \in \omega$. We outsource the task of producing all Ulm type 1 groups which are *not* finite and *not* infinite junk to the module from [DMN17]. The remaining isomorphism types are listed by the infinite and finite junk collectors, as described and verified in Section 6.

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.

[AKO] C. Ash, J. Knight, and S. Oates. Recursive abelian p-groups of small length. Unpublished. An annotated manuscript is available upon request.

[AT11] Vikraman Arvind and Jacobo Torán. Solvable group isomorphism is (almost) in NP $\cap$ conp. *TOCT*, 2(2):4:1–4:22, 2011.

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

[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.

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

[DMN16] Rodney Downey, Alexander G. Melnikov, and Keng Meng Ng. Abelian p-groups and the halting problem. *Ann. Pure Appl. Logic*, 167(11):1123–1138, 2016.

[DMN17] Rodney G. Downey, Alexander G. Melnikov, and Keng Meng Ng. A Friedberg enumeration of equivalence structures. *J. Math. Log.*, 17(2):1750008, 28, 2017.

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

[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.

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

[His81] N. Hisamiev. Criterion for constructivizability of a direct sum of cyclic p-groups. *Izv. Akad. Nauk Kazakh. SSR Ser. Fiz.-Mat.*, (1):51–55, 86, 1981.

[Kap69] I. Kaplansky. *Infinite abelian groups*. Revised edition. The University of Michigan Press, Ann Arbor, Mich., 1969.

[Khi92] N. Khisamiev. Constructive abelian p-groups. *Siberian Adv. Math.*, 2(2):68–113, 1992.

[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.

[Kie13] Jonathan A. Kiehlmann. Classifications of countably-based abelian profinite groups. *J. Group Theory*, 16(1):141–157, 2013.

[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.

[Kur60] A. Kurosh. *The theory of groups*. Chelsea Publishing Co., New York, 1960. Translated from the Russian and edited by K. A. Hirsch. 2nd English ed. 2 volumes.

[LMS18] Karen Lange, Russell Miller, and Rebecca M. Steiner. Classifications of computable structures. *Notre Dame J. Form. Log.*, 59(1):35–59, 2018.

[Mal61] A. Mal′cev. Constructive algebras. I. *Uspehi Mat. Nauk*, 16(3 (99)):3–60, 1961.

[Mel14] Alexander G. Melnikov. Computable abelian groups. *Bull. Symb. Log.*, 20(3):315–356, 2014.

[Mel17] Alexander Melnikov. Computable topological groups and pontryagin duality. page 1, 08 2017.

[MN18] Alexander G. Melnikov and Keng Meng Ng. Computable torsion abelian groups. *Adv. Math.*, 325:864–907, 2018.

[Pon34] L. Pontrjagin. The theory of topological commutative groups. *Ann. of Math. (2)*, 35(2):361–388, 1934.

[Rab60] M. Rabin. Computable algebra, general theory and theory of computable fields. *Trans. Amer. Math. Soc.*, 95:341–360, 1960.

[Rog77] L. Rogers. Ulm's theorem for partially ordered structures related to simply presented abelian p-groups. *Trans. Amer. Math. Soc.*, 227:333–343, 1977.

[Rog87] H. Rogers. *Theory of recursive functions and effective computability*. MIT Press, Cambridge, MA, second edition, 1987.

[Sac90] Gerald E. Sacks. *Higher recursion theory*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1990.

Victoria University of Wellington
Email address: Rod.Downey@msor.vuw.ac.nz

Massey University
Email address: alexander.g.melnikov@gmail.com

Nanyang Technological University
Email address: kmng@ntu.edu.sg