ESCAPING TENNENBAUM'S THEOREM AND A STRONG JUMP INVERSION THEOREM

DUARTE MAIA

Abstract. Tennenbaum's theorem states that PA does not admit any nonstandard computable model. In 2022, Pakhomov proved that this theorem is fragile in regards to how PA is expressed, by constructing a theory that is definitionally equivalent to PA (roughly: "it's PA but with a different choice of signature") for which there is a computable nonstandard model. He showed that this fragility does not extend to true arithmetic (any nonstandard model of a theory definitionally equivalent to Th(N) is not computable), but the question of whether this fragility extends to fragments of PA of intermediate strength was left open. We show that it does, by constructing a sequence of theories T^n which are definitionally equivalent to: "PA plus all Π_n truths," all of which admit computable nonstandard models. In the process, we produce a general-purpose theorem for strong jump inversion. Besides applying this theorem to obtain our novel result, we show that several known results from the literature can be seen as direct applications of our theorem.

§1. Introduction.

1.1. Background—Tennenbaum's theorem. It is a well-known result of Tennenbaum (see, e.g., [10]) that PA, the first-order theory of Peano Arithmetic, does not admit any nonstandard computable model. In fact, in every nonstandard model of PA, neither addition nor multiplication is computable. Generalizations of this theorem can be found in the literature, including versions of it for weaker fragments of PA [20], alternative operations [19], and finite set theory [7, 15], to name a few. However, all of these approaches and variations appear to depend heavily on the specificities of the chosen signature. This was observed by Pakhomov, who in 2022 [18] constructed a theory that is definitionally equivalent to PA (see Definition 1.2 below) for which there is a computable nonstandard model. This shows that Tennenbaum's theorem really is reliant on the choice of signature.

In this article, we extend Pakhomov's construction to the stronger theories obtained by adding to PA all true Π_n sentences, for fixed values of n (Theorem 4.5). In the process, we obtain a general theorem for strong jump inversion (Theorem 2.6) and show that this theorem can be used to establish multiple other results already present in the literature (Section 3).

Received April 3, 2026 UTC.

2020 *Mathematics Subject Classification.* Primary 03C57, 03C62, 03H15.

Key words and phrases. logic, computability, computable structure theory, nonstandard, PA, Peano axioms, Peano arithmetic, jump inversion, strong jump inversion.

© The Author(s), 2026. Published by Cambridge University Press on behalf of The Association for Symbolic Logic. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution, and reproduction in any medium, provided the original work is properly cited.

0022-4812/00/0000-0000
DOI:10.1017/jsl.2026.10219

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

DEFINITION 1.1 (Definitional extension). Let T be a theory over a signature $\mathcal{L}$. A *definitional extension* of T is a theory $S \supseteq T$, over a signature $\mathcal{L}' \supseteq \mathcal{L}$, such that

- every theorem of S that only uses symbols from the signature $\mathcal{L}$ is also a theorem of T, and
- every symbol of $\mathcal{L}' \setminus \mathcal{L}$ is S-definable in terms of symbols in $\mathcal{L}$. For example, if P is any predicate symbol in $\mathcal{L}'$, there is an $\mathcal{L}$-formula φ such that

$$S \vdash (\forall \vec{x})(P(\vec{x}) \leftrightarrow \varphi(\vec{x})),$$

and a similar statement holds for function symbols.

DEFINITION 1.2 (Definitional equivalence). Let T_1 and T_2 be two theories, whose signatures $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, are assumed to be disjoint with no loss of generality. We say that T_1 and T_2 are *definitionally equivalent* if there is a theory T which is a definitional extension of both.

REMARK 1.3. The notion of definitional equivalence is related, but not identical, to a more common notion of "equal power of theories" called bi-interpretability: definitional equivalence is strictly stronger, though these two notions coincide for most natural examples. The interested reader will find a thorough comparison of these two notions in [6].

A nontrivial example of two definitionally equivalent theories is that of PA and a specific version of finite set theory, which we will refer to as ZFfin+TC. This theory, and a closely related one, will play an important role in what follows, so we formally introduce them to the reader. A study of this theory can be found in [11].

DEFINITION 1.4. The theory ZF–inf+TC consists of the usual axioms of Zermelo–Fraenkel set theory, with the removal of the axiom of infinity, and with the addition of the so-called "axiom of transitive closure," which for our purposes is equivalent to the statement: $V = \cup_{\alpha \in \mathrm{Ord}} V_\alpha$.

DEFINITION 1.5. The theory ZFfin+TC consists of ZF–inf+TC, together with the negation of the axiom of infinity.

THEOREM 1.6. *The theories* PA *and* ZFfin+TC *are definitionally equivalent*.

PROOF. See [11]. ⊣

1.2. Pakhomov's construction and new results. In [18], Pakhomov constructed a theory that is definitionally equivalent to ZF–inf+TC, and consequently to PA, and has computable nonstandard models. We start by reviewing his results.

DEFINITION 1.7. Let $T(S)$ be the theory defined by Pakhomov, which axiomatizes a single ternary predicate $S(x, y, z)$. The theory $T(S)$ is definitionally equivalent to ZF–inf+TC. We let $T(\in, S)$ be the common definitional extension.

THEOREM 1.8 (Pakhomov [18]). *If T is a consistent c.e. theory extending $T(S)$, then T admits a computable model.*

Pakhomov proves this using an explicit "Henkin construction" type of argument. We were able to reframe his work in terms of the following stronger theorem.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

THEOREM 1.9. *If D is a $0'$-computable model of $T(\in, S)$, the reduct $D\restriction S$ admits a computable copy.*

PROOF. Case $n = 1$ of Theorem 4.1 below. ⊣

In this article, we generalize the methods used by Pakhomov to answer a question posed by him in the affirmative.

QUESTION 1 (Pakhomov [18]). *Fix a value of n. Are there theories definitionally equivalent to "*PA *plus all true Π_n sentences" that have computable non-standard models*?

The motivation for this question is that, as Pakhomov proved in [18], this statement is not true 'in the limit'. More precisely, Pakhomov showed that any nonstandard model of a theory that is definitionally equivalent to true arithmetic cannot be computable. This led to the question of whether a partial result could be recovered.

We answer Pakhomov's question via the following improvement on Pakhomov's construction of $T(S)$.

THEOREM 4.1. *There is a nested sequence of consistent c.e. theories* ZF–inf+TC $\subseteq T^0 \subseteq T^1 \subseteq T^2 \subseteq \cdots$ *satisfying the following properties*:

- *Each T^n is in the signature containing $\in$ and predicates $S^0, \ldots, S^n$, with each S^i being an $(i+2)$-ary predicate symbol.*
- *All of these extensions are conservative over* ZF–inf+TC, *in the sense that they contain no additional theorems in the predicate $\in$.*
- *Each of the predicates $\in$ and S^n is definable from the other within T^n.*
- *Given an X'-computable model D of T^n restricted to the signature containing $S^i, \ldots,$ and S^n, there is an X-computable copy M of $D\restriction(S^{i+1}, \ldots, S^n)$.*

REMARK 1.10. The second and third bullet point imply that, for every n, ZF–inf+TC and $T^n\restriction S^n$ are definitionally equivalent.

From Theorem 4.1, we conclude one of the main results of this article.

THEOREM 4.5. *For every n, there is a theory definitionally equivalent to "*PA *plus all true Π_n sentences" that admits a computable non-standard model.*

PROOF. First, note that the theory T containing PA plus all true Π_n sentences is a $0^{(n)}$-c.e. theory. By the results of Kaye and Wong [11], PA is definitionally equivalent to ZFfin+TC, and therefore T is definitionally equivalent to a $0^{(n)}$-c.e. extension of the theory T^{n+1} from Theorem 4.1. Say $\bar{T}$ is this theory, and let $\bar{T}^+$ be the same with an added constant c and an axiom schema ensuring that c is a nonstandard element. Let D be a $0^{(n+1)}$-computable model of $\bar{T}^+$, in the signature containing the $n+2$ predicates S^0 to S^{n+1}, which exists by the computable completeness theorem (see [8]). Then, apply the relativized version of Theorem 4.1 $n+1$ times, obtaining a computable model of $\bar{T}\restriction S^{n+1}$. This shows that $\bar{T}\restriction S^{n+1}$ is the theory we sought: It is definitionally equivalent to "PA plus all true Π_n sentences," and it admits a computable nonstandard model. ⊣

Pakhomov [18] also posed another question, which has been answered by Lutz and Walsh.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

QUESTION 2 (Pakhomov [18]). *Is there a c.e. theory T such that no definitionally equivalent theory T' admits a computable model*?

THEOREM 1.11 (Lutz–Walsh [13]). *There is a c.e. theory T such that every model of every theory T' definitionally equivalent to T is noncomputable.*

1.3. A strong jump inversion theorem. In the process of answering Pakhomov's question, it happened that certain parts of our construction were not specific to arithmetic, and so we were able to extract a general-purpose theorem for strong jump inversion.[1]

REMARK 1.12. For general background on strong jump inversion theorems and their history, see Section 1.1 of [3]. A general reference for computable structure theory is [16].

REMARK 1.13. We are not the first to seek a general-purpose jump inversion theorem. The 2018 paper [3] also contains a general-purpose theorem (Theorem 2.5) from which some theorems from the literature are recovered. Our theorem seems to follow similar themes as the theorem found in [3], but our assumptions have a different flavor and our conclusion goes in a different direction. Notably, our result requires some amount of creativity in its application—one must, in most cases, construct an intermediate structure to apply our Theorem 2.6 to obtain a jump inversion result, while Theorem 2.5 from [3] is relatively straight-forward in its application to examples.

To present our main result, Theorem 2.6, we need some definitions. From this moment on, all structures are assumed to be relational.

DEFINITION 1.14. For us, an *atomic formula* in the variables $\vec{x}$ is a formula obtained by applying a single *positive* instance of a predicate to some of the variables in the tuple $\vec{x}$ (possibly repeated and reordered). We do not consider the negation of a predicate to be atomic.

DEFINITION 1.15. An *atomic type* in the variables $\vec{x}$ is a set of atomic formulas in $\vec{x}$. For a tuple of elements $\vec{b}$ in a structure, the *atomic type of* $\vec{b}$ consists of the set of atomic formulas satisfied by $\vec{b}$.

DEFINITION 1.16 ($0'$-computable c.e.-typed structure). Let $\mathcal{L}'$ be a computable relational signature. A *$0'$-computable c.e.-typed structure* D consists of a $0'$-computable function t, whose domain is the finite power-set of an initial segment of $\mathbb{N}$ (this initial segment we call the *domain* of D), which takes as input the strong index of a finite set of natural numbers $\vec{n} = \{n_0 < n_1 < \cdots < n_{k-1}\}$ and outputs the c.e. index $t(\vec{n})$ for an atomic $\mathcal{L}'$-type $\mathrm{tp}(\vec{n})$ in the variables $(x_{n_0}, \ldots, x_{n_{k-1}})$, with the compatibility condition that if $\vec{n} \subseteq \vec{m}$ then $\mathrm{tp}(\vec{n}) \subseteq \mathrm{tp}(\vec{m})$.

DEFINITION 1.17. We work in the computable infinitary language, as described in [17, Chapter III].

[1] Patrick Lutz was the first person to bring the idea of a broad generalization to my attention. He also provided a working conjecture that was very useful in developing the theory.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

Given a relational signature $\mathcal{L}'$, we define a $\bigvee^{\mathrm{c}}_0$ formula as a formula of the type $\varphi(\vec{x}) \equiv P_1 \wedge \cdots \wedge P_n$, where each P_j is an atomic formula in $\vec{x}$, per Definition 1.14. We also allow for P_j to denote $\top$, the "true" predicate, and $\bot$, the "false" predicate.

We define a $\bigvee^{\mathrm{c}}_1$ formula as a formula (in finitely many variables) of the type

$$\varphi(\vec{x}) \equiv \bigvee_{i<\omega} \varphi_i(\vec{x}),$$

where $\{\varphi_i\}_{i<\omega}$ is a computable sequence of $\bigvee^{\mathrm{c}}_0$ formulas.

Finally, we define the $\bigvee^{\mathrm{c}}_2$ formulas as those formulas in finitely many variables of the type

$$\varphi(\vec{x}) \equiv \bigvee_{i<\omega} \left(\psi_i(\vec{x}) \wedge \neg\varphi_i(\vec{x})\right),$$

where $\{\psi_i\}_{i<\omega}$ and $\{\varphi_i\}_{i<\omega}$ are computable sequences of $\bigvee^{\mathrm{c}}_1$ formulas.

One might imagine extending the definition to higher levels, by defining a $\bigvee^{\mathrm{c}}_{n+1}$ formula as an infinite c.e. disjunction of conjunctions of $\bigvee^{\mathrm{c}}_n$ formulas and negations thereof, but these first levels will be sufficient for our purposes.

DEFINITION 1.18 (QETP). Let $\mathcal{L} \subseteq \mathcal{L}'$ be computable signatures, and let D be a structure over $\mathcal{L}'$. We say that D satisfies the *computable positive quantifier elimination and trash existence property*, abbreviated to *QETP*, if the following properties hold:

(a) $\mathcal{L}$ is finite, closed under negation, and includes the predicate $\neq$,

(b) For every quantifier-free $\mathcal{L}$-type $q(\vec{x}, \vec{y})$, there is a $\bigvee^{\mathrm{c}}_1$ $\mathcal{L}'$-formula $\chi_q(\vec{x})$, computable from q, such that

$$D \vDash (\forall \vec{x}) \left[\chi_q(\vec{x}) \leftrightarrow (\exists \vec{y})\, q(\vec{x}, \vec{y})\right].$$

(c) There is a $0'$-computable *partial* function, denoted by τ, which takes as input a pair $(q(\vec{x}, y, \vec{z}), \varphi(\vec{x}))$, where $q(\vec{x}, y, \vec{z})$ is a quantifier-free $\mathcal{L}$-type[2] and $\varphi(\vec{x})$ is a *finitary* $\mathcal{L}'$-formula, whose output is a $\bigvee^{\mathrm{c}}_2$ $\mathcal{L}'$-formula denoted $\tau_{q\varphi}(\vec{x}, y)$ that satisfies the following two properties:

- For every tuple $(\vec{x}, y, \vec{z})$ of elements in D, of $\mathcal{L}$-type $q(\vec{x}, y, \vec{z})$, there is a formula $\varphi(\vec{x})$ satisfied by $\vec{x}$ such that (q, φ) is in the domain of τ. Equivalently, if $\vec{b}$ is a tuple of elements in D such that $D \vDash \chi_q(\vec{b})$, there is $\varphi(\vec{x})$ such that (q, φ) is in the domain of τ and $D \vDash \varphi(\vec{b})$. Moreover,
- The following formula holds in D:

$$D \vDash (\forall \vec{x}) \left[\chi_q(\vec{x}) \wedge \varphi(\vec{x}) \to (\exists y)\tau_{q\varphi}(\vec{x}, y)\right].$$

(d) Given a pair $(q(\vec{x}, y, \vec{z}), \varphi(\vec{x}))$ in the domain of τ as above, together with a quantifier-free $\mathcal{L}$-type $Q(\vec{x}, y, \vec{z}, \vec{w})$ extending $q(\vec{x}, y, \vec{z})$, there is a $\bigvee^{\mathrm{c}}_1$ $\mathcal{L}'$-formula $\varepsilon\tau_{q\varphi Q}(\vec{x})$, computable from the triple (q, φ, Q), such that

$$D \vDash (\forall \vec{x}) \left[(\exists y) \left[\chi_Q(\vec{x}, y) \wedge \tau_{q\varphi}(\vec{x}, y)\right] \leftrightarrow \varepsilon\tau_{q\varphi Q}(\vec{x}) \right].$$

The function $(q, \varphi, Q) \mapsto \varepsilon\tau_{q\varphi Q}$ may be defined in some cases even if the pair (q, φ) is not in the domain of τ. In this scenario, we do not make any requirement of $\varepsilon\tau_{q\varphi Q}$.

[2]Note the lack of "vector arrow" on the variable y. This denotes a *single* variable, not a tuple of them.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

Note: As a corollary of (c), we obtain $D \vDash (\forall \vec{x}) \left[\chi_q(\vec{x}) \wedge \varphi(\vec{x}) \to \varepsilon\tau_{q\varphi q}(\vec{x})\right]$ for (q, φ) in the domain of τ.

(e) Finally, for all q, φ, Q, we require

$$D \vDash (\forall \vec{x}, y) \left[\varepsilon\tau_{q\varphi Q}(\vec{x}) \wedge \tau_{q\varphi}(\vec{x}, y) \to \chi_Q(\vec{x}, y)\right].$$

We now state the central result of this article.

THEOREM 2.6. *If D is a $0'$-computable c.e.-typed structure over the signature $\mathcal{L}' \supseteq \mathcal{L}$ satisfying the QETP, then the reduct $D{\restriction}\mathcal{L}$ admits a computable copy M. There is a $0'$-computable isomorphism between $D{\restriction}\mathcal{L}$ and M. This result is uniform in the index for D and an index for witnesses that D satisfies the QETP,*[3] *and relativizes uniformly.*

We obtain some known results as corollaries of Theorem 2.6 (see Section 3).

An extended version of this article can be found in [14].

§2. A general jump inversion theorem.

2.1. Preliminary observations. Before the main proof of this article, we make some observations that will be necessary regarding the new types of formulas we introduced in Definition 1.17 and regarding the definition of the QETP (Definition 1.18).

REMARK 2.1. The main property of the $\bigvee_1^c$ formulas is: Given a c.e. index for the atomic $\mathcal{L}'$-type of a tuple of elements $\vec{b}$ in some structure, and a $\bigvee_1^c$ formula $\varphi(\vec{x})$ satisfied by $\vec{b}$, we can computably confirm in finite time that the tuple $\vec{b}$ satisfies the formula φ.

REMARK 2.2. The main property of the $\bigvee_2^c$ formulas is: Given a tuple of elements $\vec{b}$ of a structure and a c.e. index for the atomic type of $\vec{b}$, it is $0'$-c.e. to determine whether a $\bigvee_2^c$ formula holds of $\vec{b}$.

To elaborate: If $\varphi(\vec{x}) \equiv \bigvee_i \left(\psi_i(\vec{x}) \wedge \neg\varphi_i(\vec{x})\right)$ is a $\bigvee_2^c$ formula, and $\vec{b}$ is a tuple of elements of a structure for which we have a c.e. index for the atomic type of $\vec{b}$, we can computably create a sequence of guesses for whether $\varphi(\vec{b})$ holds: Guess 'no' until we find i for which $\psi_i(\vec{b})$ holds. Then guess 'yes' until we find that $\varphi_i(\vec{b})$ holds. Again guess 'no' until we find a new j for which $\psi_j(\vec{b})$ holds, then 'yes' until we find that $\varphi_j(\vec{b})$ holds, and so on. This sequence of guesses will have the following behavior:

- if $\varphi(\vec{b})$ holds, we will guess 'yes' all but finitely many times, and
- if $\varphi(\vec{b})$ does not hold, we will guess 'no' infinitely many times.

REMARK 2.3. Bazhenov et al. introduced in [2] two hierarchies of infinitary formulas, which they named Σ_α^p and Π_α^p for $\alpha < \omega_1$, which are defined with positivity of predicates in mind.[4] These are closely related to our work: Our $\bigvee_n^c$ formulas coincide exactly with the set of quantifier-free Σ_n^p formulas.

[3] This includes: An index for which predicates in $\mathcal{L}$ are the negation of each other, an index for the computable functions χ and $\varepsilon\tau$, and an index for the $0'$-computable function τ.

[4] We would like to thank an anonymous reviewer for bringing this work to our attention.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

REMARK 2.4. We will work with Item (c) of the QETP in the following manner: We envision a computable process that enumerates triples (q, φ, τ), and sometimes deletes them, in such a manner that, if $\tau_{q\varphi}$ is well-defined, the triple $(q, \varphi, \tau_{q\varphi})$ will eventually be enumerated and never removed, and every triple that is not of the form $(q, \varphi, \tau_{q\varphi})$ for (q, φ) in the domain of τ will eventually be removed.

REMARK 2.5. We make an additional technical assumption on the formulas χ_q with no loss of generality. Let $q(\vec{x}, \vec{y}, \vec{z})$ be a quantifier-free $\mathcal{L}$ formula. We can assign a χ-formula to q in (at least) two different ways: By grouping the variables as $(\vec{x}, [\vec{y}, \vec{z}])$, or by grouping them as $([\vec{x}, \vec{y}], \vec{z})$. This leads to two different formulas, which, when necessary to distinguish, we notate, respectively, as $\chi_{q(\vec{x},\cdot,\cdot)}(\vec{x})$ and $\chi_{q(\vec{x},\vec{y},\cdot)}(\vec{x}, \vec{y})$. Now, we assume without loss of generality that, in such a scenario, $\chi_{q(\vec{x},\vec{y},\cdot)}(\vec{x}, \vec{y})$ contains $\chi_{q(\vec{x},\cdot,\cdot)}(\vec{x})$ as a subformula.

The reason why we can make such an assumption is that, given a quantifier-free $\mathcal{L}$-formula $q(\vec{x}, \vec{y})$, we may redefine $\bar{\chi}_q(\vec{x})$ as

$$\bar{\chi}_q(\vec{x}) = \bigwedge_{\vec{x}=(\vec{x}_1,\vec{x}_2)} \chi_{q(\vec{x}_1,\cdot,\cdot)}(\vec{x}_1),$$

where the conjunction ranges over all decompositions of the tuple $\vec{x}$ into two tuples of variables (not necessarily in order).

2.2. Main result.

THEOREM 2.6. *If D is a $0'$-computable c.e.-typed structure over the signature $\mathcal{L}' \supseteq \mathcal{L}$ satisfying the QETP, then the reduct $D\restriction\mathcal{L}$ admits a computable copy M. There is a $0'$-computable isomorphism between $D\restriction\mathcal{L}$ and M. This result is uniform in the index for D and an index for witnesses that D satisfies the QETP,*[5] *and relativizes uniformly.*

PROOF. We build a computable copy of $D\restriction\mathcal{L}$ via a finite injury argument. First, some background assumptions:

- We see the structure D as being given by a computable process, which outputs elements and c.e. indices for the atomic $\mathcal{L}'$-type of the elements output until now, and sometimes erases previously constructed elements. We will be executing this construction in parallel in the background of ours.
- An element of D which is added in this manner and never removed will be referred to as a *real element* of D.
- We will build the $\mathcal{L}$-structure $M \cong D\restriction\mathcal{L}$ via a computable process that outputs a sequence of elements, and with each element, the quantifier-free $\mathcal{L}$-type of the elements output until now.
- We well-order the elements of M and the elements of D by time of addition. This includes "fake" elements of D. When we refer to "the first element of M (or D) that satisfies such-and-such property," we mean the one that was added earliest.

[5] This includes: An index for the function that takes a predicate P in $\mathcal{L}$ to a predicate that is interpreted in D as $\neg P$, an index for the computable functions χ and $\varepsilon\tau$, and an index for the $0'$-computable function τ.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

- We will imagine an assortment of agents, referred to as "agent i" for $i = 0, 1, 2, \ldots$, each of which will be responsible for one step in a back-and-forth construction of an isomorphism between D and the structure M that we are constructing. Roughly speaking, agent i will be responsible for satisfying the requirement: The i-th real element of D is matched to an element of M, and the i-th element of M is matched to an element of D, and the matchings created by agents 0 to i form an isomorphism of finite $\mathcal{L}$-structures. We describe requirement i in more detail below, by describing how it may be injured, after we have introduced some further notation about the construction.
- We assume that only one agent is active at each time. When agent i is done with its task, it will activate agent $i + 1$. When an injury occurs, we divert execution back to a prior agent, say agent j, and delete the state of all agents past agent j.
- The internal state of this construction, that is, the information we will be keeping track of, consists of the following data (plus whatever auxiliary information is necessary to keep track of the agent mechanism):
  - The current status of the structures M and D.
  - One-to-one matchings between some elements of M and some elements of D, the latter of which may or may not be real.
  - To some elements m of M, we match a triple of formulas $(q(\vec{x}, y, \vec{z}), \varphi(\vec{x}), \tau(\vec{x}, y))$ that have been output by the process from Remark 2.4, together with a decomposition of the elements of M at the time that this matching was created into tuples $\vec{x}$, $y = m$, and $\vec{z}$.
  - To each matching of the previous two kinds, we keep track of which agent created it and when.
- The internal state of this construction may be modified via the following operations:
  - We may add a new element to M, in which case we immediately match it to an element of D.
  - We may match a triple of formulas (q, φ, τ) to an unmatched element $y \in M$, together with the decomposition $M_{\text{current}} = (\vec{x}, y, \vec{z})$, where $\vec{x}$ denotes the elements of M that have a match in D at this moment.
  - We may create a matching between an element of M and an element of D, if the element of M has a triple (q, φ, τ) assigned.
  - In the event of injury, we may remove matches of the three kinds above.
- We say that requirement i is injured if any of the following three things happen:
  - An element of D which was matched by agent i to an element of M is deleted.
  - There is an element of M, to whom agent i matched a triple of formulas (q, φ, τ), for which the triple of formulas has been deleted from the enumeration of the process of Remark 2.4.
  - There is an element of M, to whom agent i matched a triple of formulas (q, φ, τ), and a matched element $d \in D$, such that our guess about whether τ holds of this element has changed to 'no'.

  In any of these cases, all matches created after the "faulty match" will be deleted.

Now we list some conditions to be maintained, barring injury. Together with each assumption, we point out the operations that we need to be careful in exercising—

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

the remaining operations will always preserve the given assumption. Moreover, the assumptions will not be broken with the passage of time unless injury happens, in which case the injury-handling part of the algorithm will rectify any issues.

Assumption (A) If $M = (\vec{x}, \vec{y})$, where $\vec{x}$ denotes the elements of M that currently admit matches in D, $\vec{b}$ denotes these matches, and $q(\vec{x}, \vec{y})$ is the type of the elements of M, we demand that $D \vDash \chi_q(\vec{b})$.

We need to be careful to ensure that this is true when we add new elements to M, and when we create matches between elements of M and elements of D. We would also need to be careful in the event that matches are deleted, but this is where Remark 2.5 becomes relevant, as it ensures that this is not a concern.

Assumption (B) If $y \in M$ is matched with a triple of formulas (q, φ, τ) and decomposition $(\vec{x}, y, \vec{z}) \subseteq M$, we demand that $M \vDash q(\vec{x}, y, \vec{z})$ and that all elements of $\vec{x}$ admit matches $\vec{b}$ in D, which satisfy $D \vDash \varphi(\vec{b})$.

We need to be careful to ensure that this is true when we assign a triple to an element. We don't need to be careful when removing matches or elements of D, because our injury process will, in that case, unmatch the formulas assigned to y.

Assumption (C) At most one unmatched element m of M will have an assigned triple at any given time, and in this event, the variables $\vec{x}$ correspond to the elements of M currently matched to elements of D, and the next action to be taken will be to match m to an element of D.

Assumption (D) If y is an element of M with assigned triple $(q(\vec{x}, y, \vec{z}), \varphi(\vec{x}), \tau(\vec{x}, y))$ and y admits a match d in D, if $\vec{b}$ is the tuple of elements matched with $\vec{x}$, it must be the case that $D \vDash \tau(\vec{b}, d)$ *at the current approximation* (in the sense of Remark 2.2).

We need to be careful to ensure that this is true when we create a match between an element of M with an assigned triple (q, φ, τ) and an element of D. We also need to be careful if the current approximation (in the sense of Remark 2.2) ever changes, which is why 'the approximation of the value of $\tau(\vec{b}, d)$ changes' is a cause of injury.

Assumption (E) If m is an element with assigned formulas $(q(\vec{x}, y, \vec{z}), \varphi(\vec{x}), \tau(\vec{x}, y))$, with $\vec{b}$ denoting the matches of the $\vec{x}$, and at this stage M is decomposed as $M = (\vec{x}, y, \vec{z}, \vec{w})$ with quantifier-free $\mathcal{L}$-type $Q(\vec{x}, y, \vec{z}, \vec{w})$, it must be the case that, if the $\vec{b}$ are real elements of D, we have $D \vDash \varepsilon\tau_{q\varphi Q}(\vec{b})$.

We need to be careful to ensure that this is true when new elements are added to M, and when we assign triples (q, φ, τ) to elements of M.

The process starts with the initialization of agent 0.

Now we describe the algorithm that will be followed by agent i. When agent i is initialized:

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

Step I. Find the first element of M, if any, that does not have a match in D. If there is none, skip to Step V. If there is one, call it m.

Step II. If m has an assigned triple of formulas (q, φ, τ), skip to Step IV.[6]

Step III. If m does not have an assigned triple: Let $(\vec{x}, y, \vec{z})$ be the current elements of M, with $\vec{x}$ corresponding to the elements that currently have a match in D—call their matches $\vec{b}$—, y corresponding to m, and $\vec{z}$ corresponding to the remaining elements. Look through the triples (q, φ, τ) enumerated as per Remark 2.4 until you find such a triple that satisfies $M \vDash q(\vec{x}, y, \vec{z})$ and $D \vDash \varphi(\vec{b})$. If all elements of $\vec{b}$ are real, such a triple will be found in finite time by Assumption (A) in conjunction with Item (c) of the QETP. If not all are real, injury will happen. Once such a triple is found, we assign to m the triple (q, φ, τ) with the decomposition $(\vec{x}, y, \vec{z})$ above.

We need to make sure that Assumption (B) and Assumption (E) are preserved.

- Assumption (B)—By design.
- Assumption (E)—We need to guarantee that, if the $\vec{b}$ are all real, we have $D \vDash \varepsilon\tau_{q\varphi q}(\vec{b})$. This is a consequence of the note at the end of Item (d) of the QETP.

Step IV. Let (q, φ, τ) be the formula assigned to m. The formula $q(\vec{x}, y, \vec{z})$ will not, in general, be the quantifier-free $\mathcal{L}$-type of the current elements of M, but it is definitely the quantifier-free $\mathcal{L}$-type of some subset of elements of M. By Assumption (C), the elements corresponding to $\vec{x}$ are exactly the ones that currently admit matches in D. Let $\vec{b}$ denote their matches.

We let $Q(\vec{x}, y, \vec{z}, \vec{w})$ denote the quantifier-free type of the current elements of M, where $\vec{x}$, y, and $\vec{z}$ are the same as above.

Look through the elements of D until you find an element d satisfying $D \vDash \tau(\vec{b}, d)$ to current approximation (in the sense of Remark 2.2).

Technical note: This search must be done in some systematic way so that, if all $\vec{b}$ are real, we eventually find a *real* element d satisfying this formula in such a way that the approximation stays fixed. An example of such a systematic way: Let's say we give the n-th element added to D, say d_n, a penalty of n stages in the time it takes to compute whether $D \vDash \tau(\vec{b}, d_n)$, and select the d such that $D \vDash \tau(\vec{b}, d)$ is verified (and not disproven until now) in the least amount of stages. This will be important when proving finite injury.

We claim that, unless an injury occurs, this search will eventually terminate. By this we mean: Assume that all $\vec{b}$ are real, and that the triple (q, φ, τ) will never be removed from the enumeration from Remark 2.4. Then, by Assumption (A) it must be the case that $D \vDash \chi_Q(\vec{b})$ and by Assumption (E) it must be the case that $D \vDash \varepsilon\tau_{q\varphi Q}(\vec{b})$. Thus, by Item (d) of the QETP, it must be the case that $D \vDash (\exists y)\tau_q(\vec{b}, y)$, and any witness to this statement will serve as the d above.

[6] Technically, this instruction is unnecessary. The only way that the condition could be met is in the case of injury (namely, if the triple (q, φ, τ) assigned to m is removed from the enumeration from Remark 2.4), but in the description of injury-handling we already skip straight to Step IV regardless.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

Once such a d is found, we match m to d.

We need to make sure that Assumption (A) and Assumption (D) hold.

- Assumption (A)—If d is real, this is direct by Item (e) of the QETP.
- Assumption (D)—By construction.

Step V. Next, find the first element of D—call it d—that does not have a match in M. If there is no such element as of right now, proceed to Step VI.

Let $q(\vec{x}, \vec{y})$ be the quantifier-free $\mathcal{L}$-type of the current elements of M, with $\vec{x}$ denoting the elements that currently have matches in D, and let $\vec{b}$ denote their matches. Let $r(\vec{x}, z)$ be the quantifier-free $\mathcal{L}$-type of the tuple $(\vec{b}, d)$.[7] In parallel, look through the quantifier-free $\mathcal{L}$ existential types $Q(\vec{x}, z, \vec{y})$ until you find such a Q extending $q(\vec{x}, \vec{y}) \wedge r(\vec{x}, z)$ such that:

- $D \vDash \chi_{Q(\vec{x},z,\cdot)}(\vec{b}, d)$, and
- for every m with an assigned triple $(q_0(\vec{x}_0, y_0, \vec{z}_0), \varphi(\vec{x}_0), \tau(\vec{x}_0, y_0))$, if $\vec{b}_0$ are the elements of M corresponding to $\vec{x}_0$, $D \vDash \varepsilon\tau_{q_0\varphi Q}(\vec{b}_0)$.

We argue that either such a Q is found in finite time or otherwise an injury will occur. Indeed, suppose that both d and the elements of $\vec{b}$ are all real, and that every matched triple (q_0, φ, τ) will never be injured. Now, since we assume $D \vDash \chi_q(\vec{b})$, there must be a tuple of real elements $\vec{c}$ in D such that $D \vDash q(\vec{b}, \vec{c})$. Now, we claim that the type Q of the tuple $(\vec{b}, d, \vec{c})$ satisfies the desired requisites:

- Since $D \vDash Q(\vec{b}, d, \vec{c})$, we must have $D \vDash (\exists \vec{y}) Q(\vec{b}, d, \vec{y})$ and hence $D \vDash \chi_Q(\vec{b}, d)$.
- If m has the triple (q_0, φ, τ) assigned, with decomposition $(\vec{x}_0, y_0, \vec{z}_0) \subseteq M$, with $\vec{b}_0$ being the elements of D corresponding to $\vec{x}_0$ and d_0 being the element corresponding to m, we have $D \vDash \chi_Q(\vec{b}_0, d_0)$ as a consequence of $D \vDash Q(\vec{b}, d, \vec{c})$, and moreover $D \vDash \tau(\vec{b}_0, d_0)$ by Assumption (D), hence $D \vDash (\exists y) \left[\chi_Q(\vec{b}_0, y) \wedge \tau_{q\varphi}(\vec{b}_0, y)\right]$, and thus $D \vDash \varepsilon\tau_{q_0\varphi Q}(\vec{b}_0)$ by Item (d) of the QETP.

Once such a $Q(\vec{x}, z, \vec{y})$ is found, one of two things occurs. Either it includes $z = y_j$ for some value of j, in which case we match d to the element corresponding to y_j, or it includes the information that all its variables represent distinct elements, in which case we add a new element to M, whose relations to the previous elements are those dictated by Q, and match it to d.

We need to make sure that Assumption (A) and Assumption (E) hold.

- Assumption (A)—By construction, since we ensured $D \vDash \chi_Q(\vec{b}, d)$.
- Assumption (E)—By construction.

[7] Here, we use Item (a) of the QETP to computably obtain (a strong index for) $r(\vec{x}, z)$ from a c.e. index for the atomic $\mathcal{L}'$-type of $(\vec{b}, d)$. More generally, we use the fact that Item (a) of the QETP allows us to computably obtain a strong index for the $\mathcal{L}$-type of a tuple from a c.e. index for its $\mathcal{L}'$-type.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

Step VI. Once all the above steps are done, agent i will pause its execution, and initialize agent $i+1$.

We now describe the process of injury. There are three means of injury:

- Suppose that an element of D is seen to be removed. If this element does not have a match in M, no action is taken. On the other hand, suppose that this element d is matched to an element m in M. In that event, suppose that agent i is the one that made this match, and halt the execution of all agents with index greater than i, erasing matches and assigned formulas created past the moment that m and d were matched. If the match between d and m was created by Step I, restart the execution of agent i. If the match between d and m was created by Step V, initialize agent $i+1$ (and do not restart agent i).
- Suppose that an element $m \in M$ has an assigned triple (q, φ, τ), which is seen to be deleted from the enumeration described in Remark 2.4 at a given stage. Suppose agent i is the one who has assigned this formula. Then, we halt the execution of all agents with index greater than i, and erase all matches and assigned formulas created past the moment that this formula was assigned to m. Then, execute agent i starting in Step III.
- Suppose that an element $m \in M$ has an assigned triple $(q(\vec{x}, y, \vec{z}), \varphi(\vec{x}), \tau(\vec{x}, y))$ and a matched element d. Suppose agent i is the one who has created these matches. Suppose that the elements $\vec{x}$ are matched to the tuple $\vec{b}$, and suppose that the current approximation of $D \vDash \tau(\vec{b}, d)$ (in the sense of Remark 2.2) has changed to 'no'. Then, we halt the execution of all agents with index greater than i, erase all matches and assigned formulas created past this point (including the match between m and d, but we retain the assigned triple (q, φ, τ)), and execute agent i starting in Step IV.

This concludes the construction. It remains to show that the resulting structure M is isomorphic to $D\restriction\mathcal{L}$, and this is done via a finite injury argument. We show that each requirement will be injured finitely many times. This will show that every element in D is eventually matched to an element of M (the i-th element of D (counting both real and fake elements) will be matched by, at worst, the i-th agent past the moment when this element is added to D), and that every element of M is eventually matched to an element of D (given an orphan element of M, inductively consider the first agent activated after all previous elements of M have been matched), and since these matches all preserve the predicates in $\mathcal{L}$, the resulting map will be an isomorphism.

Suppose, for the sake of induction, that all requirements prior to i are injured finitely many times. We show that requirement i itself also suffers finite injury. We do this individually for each of the matches potentially created by agent i:

- First, we show that the matching created by the first orphan element of m at this moment, if there is any, is injured finitely many times. Indeed, per the notation of Remark 2.4 and using Item (c) of the QETP, eventually a triple (q, φ, τ) will be enumerated and never removed, such that the relevant elements of D satisfy $\chi_q \wedge \varphi$. Past this point, after all previously-added false triples have been removed, this triple (or another) will be permanently assigned to m. Afterward, we know (see the technical note in Step IV) that after enough steps we match m with a real element d of D that satisfies $\tau(\vec{b}, d)$ at the current approximation, which will then stay fixed. This match will never cause injury.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

- Now, assuming that we've reached a time where the matching discussed in the previous bullet point or created by previous agents will never be removed, suppose that there is an unmatched element in D for agent i to match with an element of M. If this element is real, there is no injury. If the element is not real, then it is eventually erased. Then execution will move to agent $i+1$, and requirement i is never injured.

This concludes the proof. ⊣

§3. Previously known applications. In this section, we list some known results that can be obtained using Theorem 2.6. We give the proof just for the most interesting one.

3.1. Elementary results.

THEOREM 3.1 [16, Lemma VII.26]. *Suppose that L is a linear order for which every element admits a successor and predecessor, and suppose that the structure $(L, <, S)$ admits a $0'$-computable copy, where $S(x, y)$ is the successor relation. Then, the structure $(L, <)$ admits a computable copy.*

PROOF. Omitted. See [14]. ⊣

PROPOSITION 3.2 (Folklore, [4, Theorem 9.1.4]). *Let $(E, \simeq)$ be an equivalence relation with infinitely many infinite equivalence classes, and assume that the structure $(E, \simeq, \{P_{\geq n}\}_{n\in\mathbb{N}})$ admits a $0'$-computable copy, where $P_{\geq n}(x)$ is the predicate "the equivalence class of x has size at least n."[8] Then, $(E, \simeq)$ admits a computable copy.*

PROOF. Omitted. See [14]. ⊣

DEFINITION 3.3 (Khisamiev, [4, Definition 9.1.3]). A set $X \subseteq \mathbb{N}$ is said to be *limitwise monotonic* if there is a total computable function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that, for every $x \in \mathbb{N}$, the sequence $\{f(x, y)\}_{y\in\mathbb{N}}$ is weakly increasing (i.e., nondecreasing) and bounded (and therefore convergent), and X is the image of the pointwise limit function $g(x) = \lim_y f(x, y)$.

DEFINITION 3.4 [4, Definition 9.1.1]. Let $(E, \simeq)$ be an equivalence relation. The *characteristic set of E* is the set

$$\#E = \{\, n \in \mathbb{N} \mid E \text{ admits at least one equivalence class of size } n \,\}.$$

PROPOSITION 3.5 (Folklore, [4, Theorem 9.1.4]). *Let $(E, \simeq)$ be an equivalence relation with finitely many infinite equivalence classes, and assume that the structure $(E, \simeq, \{P_{\geq n}\}_{n\in\mathbb{N}})$ admits a $0'$-computable copy, where $P_{\geq n}(x)$ is the predicate "the equivalence class of x has size at least n." Assume moreover that the characteristic set of E is limitwise monotonic. Then, $(E, \simeq)$ admits a computable copy.*

PROOF. Omitted. See [14]. ⊣

PROPOSITION 3.6 [3, Proposition 3.8]. *Let T be an infinite subtree of $\omega^{<\omega}$ such that every node of T that is not a leaf has infinitely many successors, of which finitely many are leaves. For the signature, we use the successor relation $S(x, y)$ and a unary relation*

[8] This includes the case where the equivalence class of x is infinite.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

$R_0(x)$ for the root. The other important predicate for a structural jump is $\mathrm{leaf}(x)$ *(x is a leaf).*

If the structure $(T, \mathrm{leaf}, S, R_0)$ admits a $0'$-computable copy, then the structure (T, S, R_0) admits a computable copy.

PROOF. Omitted. See [14]. ⊣

3.2. Khisamiev's theorem. The following theorem is in the literature, see, e.g., Chapter 5 of [4].

In the following, we interpret a c.e. presentation of a group as being a description of the group as a list of generators and relations, such that there is a computer program that enumerates the list of generators and relations. In the present case, this is equivalent to the more general notion of c.e. presentation of a structure in computability theory.

THEOREM 3.7 (Khisamiev). *Every c.e. presented torsion-free abelian group is isomorphic to a computable group. Furthermore, if the group is nontrivial, then this computable copy can be built uniformly in the index of the c.e. presentation.*

PROOF. Given a c.e. presentation of a group G, we first build a $0'$-computable c.e.-typed copy of G over an appropriate signature $\mathcal{L}'$, and we will then show that any nontrivial torsion-free abelian group has the QETP (uniformly) over $\mathcal{L} \subseteq \mathcal{L}'$, where $\mathcal{L}$ is the signature containing only the ternary predicate $S(x, y, z) \equiv$ "$x + y = z$"(and its negation, and equality and its negation).

The signature $\mathcal{L}'$ contains $\mathcal{L}$ plus:

- The countably many predicates $Q_{n,\vec{c}}(\vec{x})$, parametrized over $n \in \mathbb{N}$ and finite tuples $c_1, \ldots, c_k \in \mathbb{Z}$:

$$Q_{n,\vec{c}}(\vec{x}) \equiv \text{“}n \mid c_1x_1 + \cdots + c_kx_k\text{.”}$$

  Note that we allow n to equal zero, in which case (since $0 \mid y$ iff $y = 0$) $Q_{0,\vec{c}}(\vec{x})$ corresponds to the predicate $\vec{c} \cdot \vec{x} = 0$.
- The countably many predicates $\neg Q_{0,\vec{c}}(\vec{x})$, parametrized over finite tuples $c_1, \ldots, c_k \in \mathbb{Z}$:

$$\neg Q_{0,\vec{c}}(\vec{x}) \equiv \text{“}c_1x_1 + \cdots + c_kx_k \neq 0\text{.”}$$

- Countably many predicates $\{n\Delta(x)\}_{n\in\mathbb{Z}}$, whose meaning is as follows: We will carefully pick a single designated nonzero element of G, which we refer to as δ, and $n\Delta(x)$ should be interpreted as "$x = n\delta$."

We need some linear algebra of $\mathbb{Z}$-modules to show that, from a c.e. presentation of G, we can uniformly obtain a $0'$-computable c.e.-typed copy of G. Then, we will explain how we choose the element of G that will be tagged with Δ. To obtain the $0'$-computable c.e.-typed copy of G, here is the basic idea: Consider the computable process that outputs all possible linear combinations of generators. We want to, with each new linear combination of generators, say d, first check whether it is equal to any of the combinations previously produced (this is an easy check with $0'$), and if it is not, we want to output a c.e. index for its type relative to the elements previously output. This is easy for the positive predicates—to check if, for example, $5 \mid 4d + 6c$ (where c is a previously output combination of generators), it suffices to check if there

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

is a further combination of generators u that satisfies the relation $5u - 4d - 6c = 0$. This is a c.e. check. The difficulty lies in the negative predicates, of which $\neg Q_{0,\vec{c}}(\vec{x})$ is the general case. In other words, we need to encode *into a single c.e. index* the full information about what linear combinations of elements-output-so-far equal zero.

Here is a fact from commutative algebra, whose proof can be found in any standard book that covers modules over principal ideal domains (e.g., [5]):

Fact A: Any submodule M of $\mathbb{Z}^k$ admits a finite basis.

Here is the effective version of this fact that we will use:

Fact B: Given a c.e. index for a submodule M of $\mathbb{Z}^k$, we may $0'$-effectively find a basis of M.

Proof idea: Consider the k-width and possibly-infinite-height matrix whose rows are the elements of M. Use $0'$ to find the GCD of the elements of the first column, call it d. If $d = 0$ (i.e., all elements of the first column are zero), skip to the next step. If $d \neq 0$, find a combination of rows of M that has d in its first coordinate, and add this combination to the basis. Then, if $k \geq 1$, recursively apply this algorithm to the $(k-1)$-width and possibly-infinite-height matrix whose elements consist of the rows of the previous matrix that have a 0 as a first entry. Upon finding a combination of rows that has the new GCD as its first coordinates, add the corresponding combination of the rows of the original matrix to the basis. After k steps, we have a basis of M.

Returning to our original problem: We have a collection of combinations of generators, say $d_0, d_1, \ldots, d_k$, and we wish to encode all possible linear combinations of these elements that are equal to zero. We do this by applying Fact B to the c.e. submodule of $\mathbb{Z}^{k+1}$ given by

$$M = \{\, \vec{c} \in \mathbb{Z}^{k+1} \mid \vec{c} \cdot \vec{d} = 0 \,\}.$$

With access to a basis of M, say $b_1, \ldots, b_\ell$ (encoded into the columns of a matrix B), we can tell whether any set of coefficients $\vec{c}$ is in M (and hence, whether we should say yes or no to $Q_{0,\vec{c}}(\vec{d})$) by using standard algorithms to solve the linear equation $B\vec{x} = \vec{c}$ over $\mathbb{Q}$, and conclude $\vec{c} \in M$ if and only if a solution exists and its coefficients are all integers. Coding this basis into a c.e. index, we obtain a c.e. index for the atomic $\mathcal{L}'$-type of a tuple $\vec{d}$ of linear combinations of generators.

Now, let us explain how to choose the element δ, which we will use to decide the predicates $n\Delta$. Of course, this has to have been done in advance, so that we can hardcode the combination of generators corresponding to δ into the c.e. indices of the types of the elements of G in advance.

There is an easy way and a hard way to choose δ. The easy way is simply to pick an arbitrary nonzero element of G. The issue is that to make this choice requires the usage of an oracle for $0'$, which will lead to a non-uniform presentation. Here is a uniform way to choose the element. Let $\{C_i\}_{i \in \mathbb{N}}$ enumerate all possible combinations of generators. To start, we will guess $\delta = C_0$, and look through the relations until (if ever) we see enough relations to conclude $C_0 = 0$. When this happens, we will guess $\delta = N_1 C_1$, where N_1 is a nonzero number divisible by enough integers. This choice is made so that, for every number k for which we'd been able to conclude $k \mid C_0$ before we found that $C_0 = 0$, we have $k \mid N_1$ and so $k \mid N_1 C_1$. We proceed in this manner, enumerating relations until we are able to conclude that $N_1 C_1 = 0$, or equivalently

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

that $C_1 = 0$. If this happens, we change our mind to guess $\delta = N_2 C_2$, where N_2 has enough divisibilities that, again, every divisibility we'd already concluded about $N_1 C_1$ also holds for N_2 and hence $N_2 C_2$. Since, by assumption, G is a nontrivial group, eventually this procedure will run into a nonzero combination of generators, and $0'$ is able to figure out which element is δ in advance. The important fact about this way of choosing δ is that we can *uniformly* find a c.e. index for the type of δ.

Now that we have a $0'$-computable c.e.-typed copy of our group over the signature $\mathcal{L}'$, we will show that every nontrivial torsion-free abelian group with a designated nonzero element with c.e. atomic type satisfies the QETP uniformly.

- First, we construct χ_q. This is an adaptation of the standard proof that the theory of torsion-free abelian groups with divisibility-by-integer predicates admits quantifier elimination, which can be found e.g. in [9, Appendix A.2]. Very briefly, we have the following lemma.

LEMMA 3.8. *Any existential statement is equivalent to a system of the type*

$$(\exists \vec{y}) q(\vec{x}, \vec{y}) \Leftrightarrow \begin{cases} M_0 \vec{x} = 0, \\ n_1 \mid \vec{c}_1 \cdot \vec{x}, \\ \vdots \\ n_\ell \mid \vec{c}_\ell \cdot \vec{x}, \\ K_0 \vec{x} \neq 0, \end{cases} \tag{1}$$

with the system found effectively. We define $\chi_q(\vec{x})$ to be the right-hand side of (1).

The effective transformation goes as follows:

– Express the type $q(\vec{x}, \vec{y})$ as a system of integer coefficient equations and inequations

$$q(\vec{x}, \vec{y}) \equiv \begin{cases} M\vec{x} = N\vec{y}, \\ K\vec{x} \neq L\vec{y}. \end{cases}$$

Important: The notation $K\vec{x} \neq L\vec{y}$ means "every entry of the vector $K\vec{x}$ is distinct from the corresponding entry in $L\vec{y}$."

– Now, write $N = ADB$, where D is a diagonal matrix padded with zeros (i.e. of the form $\begin{bmatrix} D_0 & 0 \\ 0 & 0 \end{bmatrix}$ with D_0 diagonal) and A and B are invertible, all with integer entries [1, Theorem 2.12]. Then, $q(\vec{x}, \vec{y})$ is logically equivalent to a new system of equations and inequations

$$q(\vec{x}, \vec{y}) \Leftrightarrow \begin{cases} M'\vec{x} = DB\vec{y}, \\ K'\vec{x} \neq LB^{-1}B\vec{y}. \end{cases}$$

– The existence of a solution to this system is, since B is invertible, equivalent to the existence of a solution to the following other system:

$$(\exists \vec{y}) q(\vec{x}, \vec{y}) \Leftrightarrow (\exists \vec{z}) \begin{cases} M'\vec{x} = D\vec{z}, \\ K'\vec{x} \neq L'\vec{z}. \end{cases}$$

– The rows of the equation $M'\vec{x} = D\vec{z}$ that have nonzero entries of D yield equations of the type $nz_i = \vec{a} \cdot \vec{x}$. This allows us to remove (while preserving logical equivalence) every instance of z_i from every other

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

(in)equation: Any other (in)equation can be multiplied by n preserving logical equivalence (this uses $n \neq 0$ and the fact that we're working with torsion-free groups), resulting in an instance of (coefficient)$\times nz_i$, which can be substituted for (coefficient) $\times$ $(\sum a_i x_i)$. Thus, we may eliminate all z variables corresponding to nonzero rows of D, and our system now becomes

$$(\exists \vec{y}) q(\vec{x}, \vec{y}) \Leftrightarrow (\exists \vec{z})(\exists \vec{w}) \begin{cases} M_0 \vec{x} = 0, \\ n_1 z_1 = \vec{c}_1 \cdot \vec{x}, \\ \vdots \\ n_\ell z_\ell = \vec{c}_\ell \cdot \vec{x}, \\ K' \vec{x} \neq L' \vec{w}. \end{cases}$$

– The equations including the z variables become divisibility statements, i.e., $(\exists z_i) n z_i = \vec{c} \cdot \vec{x} \Leftrightarrow n \mid \vec{c} \cdot \vec{x} \equiv Q_{n,\vec{c}}(\vec{x})$, so we now need only deal with the inequations $K' \vec{x} \neq L' \vec{w}$.
– Now, let us stratify the equations $K' \vec{x} \neq L' \vec{w}$ as follows: First, consider the equations that use no w-variable. Then, consider the ones that use w_1 and no other. Then, consider the ones that use only w_1 and w_2, and so on. The equations that use no w-variable are important—keep them. These are a requirement on $\vec{x}$ that we cannot get rid of. Let us refer to these equations as $K_0 \vec{x} \neq 0$. On the other hand, at every new stage of the strata, the resulting equations will doubtlessly be satisfied by some w_i. For example, if the first stratum has e equations, this is forbidding at most e possible values of w_1. But the group is nontrivial and hence infinite, whereby there is at least one valid choice for w_1. Then, in the next stratum, contingent on this choice of w_1, there are once again at most (number of equations) forbidden values for w_2. Simply choose w_2 to not be one of those. Continuing in this manner, we see that we can ensure that all equations that include a w variable can be automatically satisfied and are therefore adding no information. This concludes the effective transformation.

REMARK 3.9. If it is known that not every x_i equals zero, one might in fact choose the w_i variables to be linear combinations of the x_i. This will be important later.

• Next, we construct τ. Intuitively, given $q(\vec{x}, y, \vec{z})$, we consider all possible assignments of y of the form $y = \frac{1}{a} \vec{c} \cdot \vec{x}$ (if such an element exists). We will prove below, when establishing the QETP, that assuming not every element of $\vec{x}$ is null there is such a combination that provably satisfies $\chi_q(\vec{x}, \frac{1}{a} \vec{c} \cdot \vec{x})$. We will choose one such combination, and set $\tau_{q\varphi}(\vec{x}, y) \equiv (ay = \vec{c} \cdot \vec{x})$. If every element of $\vec{x}$ is null, we set $\tau(\vec{x}, y) \equiv N\Delta(y)$, where N is large enough to encompass all divisibilities demanded by the tuple of elements $\vec{z}$ (e.g., if $q(y, z) \equiv (z + z = y)$, we would set $y = 2\delta$).

Let's discuss this process more formally. Given $q(\vec{x}, y, \vec{z})$:
– If q includes the data $y = 0$, set $\tau_{q\top}(\vec{x}, y) \equiv (y = 0)$. Otherwise:
– Set $\varphi_0(\vec{x}) \equiv \wedge_i (x_i = 0)$ and set $\tau_{q\varphi_0}(\vec{x}, y) \equiv \varphi_0(\vec{x}) \wedge N\Delta(y)$, where N is positive and large enough to ensure $(\forall d \neq 0) \chi_q(\vec{0}, Nd)$.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

– For every nonzero integer a and tuple of integers $\vec{c}$, set $\varphi_{a\vec{c}}(\vec{x}) \equiv$ "$\chi_q(\vec{x}, \frac{1}{a}\vec{c}\cdot\vec{x})$" and $\tau_{q\varphi_{a\vec{c}}}(\vec{x}, y) \equiv (ay = \vec{c}\cdot\vec{x})$. We now explain precisely what we mean by $\chi_q(\vec{x}, \frac{1}{a}\vec{c}\cdot\vec{x})$.

Since q is assumed to be a type, we have by definition that $\chi_q(\vec{x}, y)$ is a system of equations as in (1). By multiplying all equations by a, we obtain an equivalent system of equations where every instance of y is being multiplied by a. In other words, we can rewrite $\chi_q(\vec{x}, y) \Leftrightarrow \psi(\vec{x}, ay)$, and as a consequence the formula "$\chi_q(\vec{x}, \frac{1}{a}\vec{c}\cdot\vec{x})$" may be taken to mean

$$\text{“}\chi_q(\vec{x}, \frac{1}{a}\vec{c}\cdot\vec{x})\text{”} \equiv (a \mid \vec{c}\cdot\vec{x}) \wedge \psi(\vec{x}, \vec{c}\cdot\vec{x}).$$

- Finally, we construct $\varepsilon\tau$. Given $q(\vec{x}, y, \vec{z})$ and $\varphi(\vec{x})$ as one of the above cases, as well as $Q(\vec{x}, y, \vec{z}, \vec{w})$ extending q, we wish to find a quantifier-free expression that is equivalent to $(\exists y)\left[\chi_Q(\vec{x}, y) \wedge \tau_{q\varphi}(\vec{x}, y)\right]$. Intuitively, the idea is that in either case our formula τ determines y exactly (as a function of $\vec{x}$), so we can just "plug that y into $\chi_Q(\vec{x}, y)$." To do this more precisely, we split into cases:
  – If $\tau_{q\varphi}(\vec{x}, y) \equiv (ay = \vec{c}\cdot\vec{x})$, we apply the same "multiply every equation by a" trick to write a formula that corresponds to "$(a \mid \vec{c}\cdot\vec{x}) \wedge \chi_Q(\vec{x}, \frac{1}{a}\vec{c}\cdot\vec{x})$." The resulting formula is the desired $\varepsilon\tau$.
  – If $\tau_{q\varphi}(\vec{x}, y) \equiv \varphi_0(\vec{x}) \wedge N\Delta(y)$, then, intuitively, we know that for a tuple to satisfy τ it must be that all x_i are zero, and that $y = N\delta$. Therefore, the existential quantifier may be eliminated, and we wish to determine whether $\chi_Q(\vec{0}, N\delta)$. This is where it is important that we have a c.e. index for the type of δ—and, if we desire a uniform result, that we have this index without recourse to any oracle. Knowing the c.e. index for the type of δ, we can just consult whether $\chi_Q(\vec{0}, N\delta)$ holds—more precisely, there is a computable sequence $\{B_n\}_{n\in\mathbb{N}}$ such that

$$\begin{cases} G \nvDash \chi_Q(\vec{0}, N\delta) & \Rightarrow B_n = \bot \text{ for all } n, \\ G \vDash \chi_Q(\vec{0}, N\delta) & \Rightarrow B_n = \top \text{ for all but finitely many } n. \end{cases}$$

In this scenario, we set

$$\varepsilon\tau_{q\varphi Q}(\vec{x}) \equiv \bigvee_n B_n.$$

Now we verify that the above definitions are witnesses to the QETP.

(a) ($\mathcal{L}$ is finite and closed under negation) Obvious.
(b) ($G \vDash \chi_q(\vec{x}) \leftrightarrow (\exists\vec{y})q(\vec{x}, \vec{y})$) By construction.
(c) There are two bullet points here. In reverse order:
- ($G \vDash \chi_q(\vec{x}) \wedge \varphi(\vec{x}) \to (\exists y)\tau_{q\varphi}(\vec{x}, y)$) Direct from construction.
- (Every tuple $\vec{b}$ of elements of G that satisfies $\chi_q(\vec{b})$ satisfies some $\varphi(\vec{b})$ with (q, φ) in domain of τ)
  – If $\vec{b} = \vec{0}$ this is obvious.
  – If some $b_i \neq 0$: We wish to show that there are $a \in \mathbb{Z}_{\neq 0}$ and $\vec{c} \subseteq \mathbb{Z}$ such that $a \mid \vec{c}\cdot\vec{b}$ and $\chi_q(\vec{b}, \frac{1}{a}\vec{c}\cdot\vec{b})$. To this effect, we use the following related lemma, which obviously implies the desired claim.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

LEMMA 3.10. *If $q(\vec{x}, \vec{y})$ is a quantifier-free type in the smaller signature $\mathcal{L}$, and $\vec{b}$ is a tuple in G such that $G \vDash (\exists \vec{y}) q(\vec{b}, \vec{y})$, and some element of $\vec{b}$ is nonzero, then there exist integer vectors $\vec{c}_1, \ldots, \vec{c}_\ell$ and nonzero integers $a_1, \ldots, a_\ell$ such that, for every i, $a_i \mid \vec{c}_i \cdot \vec{b}$, and $G \vDash q(\vec{b}, \frac{1}{a_1}\vec{c}_1 \cdot \vec{b}, \ldots, \frac{1}{a_\ell}\vec{c}_\ell \cdot \vec{b})$.*

The proof of this lemma uses a change of variables, similar to the one used in the construction of χ_q, followed by noticing Remark 3.9. The details are omitted.

(d) $(G \vDash (\exists y)\left[\chi_Q(\vec{x}, y) \wedge \tau_{q\varphi}(\vec{x}, y)\right] \leftrightarrow \varepsilon\tau_{q\varphi Q}(\vec{x}))$ True by construction.

(e) $(G \vDash \varepsilon\tau_{q\varphi Q}(\vec{x}) \wedge \tau_{q\varphi}(\vec{x}, y) \rightarrow \chi_Q(\vec{x}, y))$ Essentially, this is true because $\tau_{q\varphi}(\vec{x}, y)$ *always* determines y uniquely in terms of $\vec{x}$. As such, if it is the case that there is some y_0 that satisfies $\chi_Q(\vec{x}, y_0) \wedge \tau_{q\varphi}(\vec{x}, y_0)$ (i.e., $\varepsilon\tau_{q\varphi Q}(\vec{x})$ holds), and it is the case that some specific y satisfies $\tau_{q\varphi}(\vec{x}, y)$, then it must be the case that $y = y_0$ and therefore $\chi_Q(\vec{x}, y)$.

We are in condition to apply Theorem 2.6 to the $0'$-computable c.e.-typed copy of G (plus designated element). Since this copy was obtained uniformly from the original presentation, the result is uniform. ⊢

§4. Escaping Tennenbaum's theorem with more truths. In this section, we give a positive answer to the question posed by Pakhomov in [18], about whether there exist theories definitionally equivalent to "PA + all Π_n truths" that admit computable nonstandard models.

As described in the introduction, we will be working primarily with the theory ZF–inf+TC, which consists of the axioms of ZF, with the removal of the axiom of infinity, and with the addition of the axiom of transitive closure, which for our purposes is the axiom stating that every set is in some level V_α of the von Neumann hierarchy.

4.1. Generalizing Pakhomov's construction. This section is dedicated to the proof of Theorem 4.1.

THEOREM 4.1. *There is a nested sequence of consistent c.e. theories* ZF–inf+TC $\subseteq T^0 \subseteq T^1 \subseteq T^2 \subseteq \cdots$ *satisfying the following properties*:

- *Each T^n is in the signature containing $\in$ and predicates $S^0, \ldots, S^n$, with each S^i being an $(i+2)$-ary predicate symbol.*
- *All of these extensions are conservative, in the sense that they contain no additional theorems in the predicate $\in$.*
- *The predicates $\in$ and S^n are definable in terms of the other within T^n.*
- *Given an X'-computable model D of T^n restricted to the signature containing S^i, ..., and S^n, there is an X-computable copy M of $D\restriction(S^{i+1}, \ldots, S^n)$.*

We start the proof by defining the sequence $\{T_n\}_{n\in\mathbb{N}}$ inductively. For the base case, we set T^0 to be ZF–inf+TC, plus the axiom $(\forall x, y)[S^0(x, y) \leftrightarrow (x \notin y)]$. It trivially (and, in the case of the last bullet point, almost vacuously) satisfies all required conditions. Now, we suppose that T^n has already been defined, and set out to define T^{n+1}.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

Within T^n, we define the ordinal-indexed sequence of predicates $S^{n+1}_\alpha(\vec{x}, y)$, where $\vec{x}$ is an $(n+1)$-tuple of variables and α is an ordinal. We define S^{n+1}_α inductively as a compatible sequence of relations on V_α with S^{n+1}_λ having an obvious definition for limit ordinals λ, so it suffices to describe the successor step. To assist us in that regard, we define the following sequence of class functions.

Given sets A, mA, an ordinal α, and sets $S^iA \subseteq (A \cup \{\alpha\})^{i+2}$ for $i = 0, \dots, n+1$,[9] define an element $w^n(\alpha, mA, A, SA, \dots, S^nA)$ inductively in n as follows:

- $w^0(\alpha, mA, A) = (\alpha, (A, mA))$, and
- $w^{n+1}(\alpha, mA, A, S^0A, S^1A, \dots, S^nA, S^{n+1}A) = w^n(\alpha, (S^{n+1}A, mA), A, S^0A, \dots, S^nA)$.

The main defining features of this sequence of class functions are:

- for every n, $(\alpha, mA, A, S^0A, \dots, S^nA) \mapsto w^n(\alpha, mA, A, S^0A, \dots, S^nA)$ is an injective definable class function, and
- $w^n(\alpha, mA, A, S^0A, \dots, S^nA)$ is *not* in V_α.

Let us go back to defining S^{n+1}_α inductively, assuming that we have already defined S^0 up to S^n, for $n \geq 0$. Assuming that S^{n+1}_α is already defined on tuples of elements of V_α, we define a "good sequence" $(\alpha, mA, A, S^0A, \dots, S^{n+1}A)$ as one that satisfies the following properties:

- A is a subset of V_α,
- for $i = 0, \dots, n+1$, the set S^iA is a subset of $(A \cup \{\alpha\})^{i+2}$,
- for $i = 0, \dots, n$, the set $S^iA \cap A^{i+2}$ coincides with the relation S^i on A,
- the set $S^{n+1}A \cap A^{n+3}$ coincides with the relation S^{n+1}_α on A,
- the set S^0A contains all pairs (a, α), (α, a), and (α, α), with $a \in A$, and
- for $i = 0, \dots, n$, the following condition holds: For every $(i+3)$-tuple $(\vec{a}, b)$ of elements of $A \cup \{\alpha\}$, if $\vec{a} \notin S^iA$, then $(\vec{a}, b) \in S^{i+1}A$.

We are now ready to define $S^{n+1}_{\alpha+1}$. Given an $(n+3)$-tuple $\vec{x}$ of elements in $V_{\alpha+1}$:

- If all elements of the tuple are in V_α, set $S^{n+1}_{\alpha+1}(\vec{x}) \equiv S^{n+1}_\alpha(\vec{x})$.
- If there is a good sequence $(\beta, mA, A, S^0A, \dots, S^{n+1}A)$, for some $\beta \leq \alpha$, such that the element $w = w^{n+1}(\beta, mA, A, S^0A, \dots, S^{n+1}A)$ is in $V_{\alpha+1}$, and every entry of $\vec{x}$ is an element of $A \cup \{w\}$, set $S^{n+1}_{\alpha+1}(\vec{x})$ to hold true if, and only if, upon replacing every instance of w in the tuple $\vec{x}$ by β, the resulting tuple is in $S^{n+1}A$.

  In the sequence, when such a choice of good sequence is clear from context, denote by $\vec{x}^*$ the operation of replacing every entry of $\vec{x}$ that equals β by w, and let $\vec{x}_*$ denote the inverse operation. Thus, the definition of $S^{n+1}_{\alpha+1}$ in this case can be reworded as: $S^{n+1}_{\alpha+1}(\vec{x}) \equiv [\vec{x}^* \in S^{n+1}A]$.

  Remark: Note that, by definition, if $(\beta, mA, A, S^0A, \dots, S^{n+1}A)$ is a good sequence and $n \geq 0$, we also have that $(\beta, (S^{n+1}A, mA), A, S^0A, \dots, S^nA)$ is also a good sequence (for a smaller value of n), and thus the element $w = w^{n+1}(\beta, mA, A, S^0A, \dots, S^{n+1}) = w^n(\beta, (S^{n+1}A, mA), A, S^0A, \dots, S^nA)$

[9] Here, the expression S^iA is merely notation (where i is a finite ordinal and A is a set), intended to evoke the idea that S^iA is defining the relation S^i between the elements of A and a new element. Note that S^0A is not the same as A.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

will also satisfy: $S^n(\vec{x}) \equiv [\vec{x}^* \in S^n A]$ for all tuples of elements of $A \cup \{w\}$, and inductively for lower indices. The case of S^0 is an edge case, but the definition of good sequence still ensures that $S^0(x, y) \equiv [(x, y)^* \in S^0 A]$.

- For every other triple of elements from $V_{\alpha+1}$, set $S^{n+1}_{\alpha+1}(\vec{x})$ to hold true.

Note that the first and second item in the definition do not contradict each other: If a tuple $\vec{x}$ fits both bullet points, then either all its elements are in A, in which case the definition of good sequence ensures that there is agreement between both possible definitions of $S^{n+1}_{\alpha+1}(\vec{x})$, or one of its elements is $w = w^{n+1}(\beta, mA, A, \dots, S^{n+1}A)$, in which case $S^{n+1}_{\gamma}(\vec{x})$ would have been defined to agree with the second bullet point for γ the smallest ordinal such that $w \in V_\gamma$.

This induces a well-defined relation S^{n+1}, whose main defining properties are as follows.

LEMMA 4.2. *Provably in* ZF–inf+TC, $\neg S^n(\vec{x})$ *iff* $(\forall y) S^{n+1}(\vec{x}, y)$.

PROOF. By construction. ⊣

LEMMA 4.3. *Let M be a model of* ZF–inf+TC. *Let A be a finite subset of M, and suppose that we wish to find an element w that relates to all elements of A with regard to the predicates $S^0, \dots, S^{n+1}$ in a prescribed manner. The following is a sufficient condition for there to exist an element w of M satisfying this prescription*: *For every $0 \leq i \leq n$, the prescription satisfies the rules*:

- *The prescription agrees with the pre-existing relations on tuples that don't contain any instance of w.*
- *For all $x \in A \cup \{w\}$, it is prescribed that $S^0(x, w)$ and $S^0(w, x)$.*
- *For all $(\vec{x}, y) \in (A \cup \{w\})^{i+2}$ such that $\neg S^i(\vec{x})$ is prescribed, it is also prescribed that $S^{i+1}(\vec{x}, y)$.*

PROOF. By construction. ⊣

These two properties are enough to establish the following theorem.

THEOREM 4.4. *Let D be an X'-computable model of T^n in the predicates $S^i, \dots, S^n$, with $i < n$. Then, the reduct $D\restriction(S^{i+1}, \dots, S^n)$ admits an X-computable copy.*

PROOF. We will apply Theorem 2.6 to D which, as an X'-computable model over a finite signature, is also an X'-computable X-c.e.-typed model. We need only establish that D has the QETP over $\mathcal{L} = (S^{i+1}, \dots, S^n)$ (plus negations) and $\mathcal{L}' = (S^i, \dots, S^n)$ (plus negations).

We construct χ, τ, and $\varepsilon\tau$:

- First, we construct χ_q for $q(\vec{x}, \vec{y})$ a quantifier-free $\mathcal{L}$-type with all variables distinct.

  We start by considering the case, where $q(\vec{x}, y)$ only has one y-variable. In this case, the formula $(\exists y) q(\vec{x}, y)$ is true if, and only if, there exists an element y satisfying a certain prescription of predicates $S^{i+1}, \dots, S^n$. We show that the following three statements are equivalent:
  – $P(\vec{x}) \equiv (\exists y) q(\vec{x}, y)$
  – $Q(\vec{x})$: For every instance of $\neg S^{j+1}(\vec{z}_0, z_1)$ in $q(\vec{x}, y)$:

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

* if $j > i$, we require that $S^j(\vec{z}_0)$ be in q, and[10]
* if $j = i$, $z_1 \equiv y$, and $\vec{z}_0$ is a subtuple of $\vec{x}$, we require $S^i(\vec{z}_0)$.

Moreover, we also demand $(q\restriction\vec{x})(\vec{x})$. In other words, whatever demand q makes of the tuple $\vec{x}$, we also demand it.

– $R(\vec{x}) \equiv$ There is a y such that $q(\vec{x}, y)$ and the relations S^0 to S^i hold for all tuples including elements from $(\vec{x}, y)$ that include at least one instance of y.

Indeed, $P(\vec{x}) \to Q(\vec{x})$ follows from the fact that $(\forall\vec{z}_0)((\exists z_1)\neg S^{j+1}(\vec{z}_0, z_1)) \to S^j(\vec{z}_0)$ (and the assumption that q is a type), $Q(\vec{x}) \to R(\vec{x})$ follows by Lemma 4.3, and $R(\vec{x})$ obviously implies $P(\vec{x})$.

Now, still in the case where there is only one y-variable, we set $\chi_q(\vec{x}) \equiv Q(\vec{x})$ as above. We will now handle the case where there are two y-variables, say y_1 and y_2, and an obvious induction will handle the general case. We will use the formulas P, Q, and R constructed above for different formulas q. We will use a subscript to indicate which formula q is being used.

Let $q(\vec{x}, y_1, y_2)$ be a quantifier-free $\mathcal{L}$-type. Then, let us look at $\chi_q(\vec{x}, y_1)$. As above, it's either $\bot$ (if the first bullet point of the definition of $Q_q(\vec{x}, y_1)$ fails), in which case the quantifier elimination is trivial, or otherwise it consists of $q\restriction(\vec{x}, y_1)$, together with a few *positive* requirements of type $S^i(\vec{z}_0)$ with $\vec{z}_0$ a subtuple of $(\vec{x}, y_1)$. Let $q'(\vec{x}, y_1)$ be $q\restriction(\vec{x}, y_1)$ together with those requirements that include y_1, and $q^*(\vec{x})$ be the remainder of these requirements. Note that

$$(\exists y_1)(\exists y_2)q(\vec{x}, y_1, y_2) \leftrightarrow (\exists y_1)\chi_q(\vec{x}, y_1) \leftrightarrow q^*(\vec{x}) \wedge (\exists y_1)q'(\vec{x}, y_1),$$

and crucially that, since all S^i-requirements made by q' are positive, $(\exists y_1)q'(\vec{x}, y_1)$ is logically between $P_{q'}(\vec{x})$ and $R_{q'}(\vec{x})$, and as such is equivalent to both, and as such is equivalent to $Q_{q'}(\vec{x})$. Then, we conclude $(\exists\vec{y})q(\vec{x}, \vec{y}) \leftrightarrow q^*(\vec{x}) \wedge Q_{q'}(\vec{x})$, and so we define the latter as χ_q in this scenario. The induction continues in a similar fashion, in the event that there are more y-variables, with the essential point being that every S^i-demand being made at any step is positive.

- Now, we construct τ. Given $q(\vec{x}, y, \vec{z})$, we set $\tau_{q\top}(\vec{x}, y)$ to be the demand that every S^i-relation including y at least once holds, and that $\chi_q(\vec{x})$ holds.
- Finally, $\varepsilon\tau_{q\top Q}(\vec{x})$ is simply $\chi_Q(\vec{x})$.

The proof that these formulas are witnesses to the QETP is simple and follows the same P, Q, and R reasoning used to establish that $\chi_q(\vec{x}) \leftrightarrow (\exists y)q(\vec{x}, y)$. ⊣

Theorem 4.1 immediately follows.

As explained in the introduction, we obtain as a corollary.

THEOREM 4.5. *For every n, there is a theory definitionally equivalent to "*PA *plus all true Π_n sentences" that has a computable nonstandard model.*

§5. Further questions. It is a standard theorem (or definition) that a PA degree is the same as one which computes a nonstandard model of PA. On the other hand, we saw that there is a computable nonstandard model of the definitionally equivalent

[10]Note: This demand is happening outside of first-order logic. Rather, it's a check that we metatheoretically make of $q(\vec{x}, y)$, and if this check is failed, we set $Q(\vec{x}) \equiv \bot$.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

theory T_0 constructed by Pakhomov—and so, by a theorem of Knight [12], there exists one in every degree. Let us define, for a theory T that is definitionally equivalent to PA, its *nonstandard spectrum* as the set of degrees of nonstandard models of T.

QUESTION 1. *If T is a c.e. theory that is definitionally equivalent to* PA, *we can see that its nonstandard spectrum is at most the set of all Turing degrees, and at least the set of all* PA *degrees. Does there exist such a theory whose nonstandard spectrum is strictly in-between these two cases? More generally, what can the nonstandard spectrum of a c.e. theory definitionally equivalent to* PA *look like?*

The methods used in this article are too coarse to address this question; they would have to be cleverly modified to stop the nonstandard spectrum from including 0.

We mentioned in the introduction that Lutz and Walsh [13] produced a c.e. consistent theory such that no definitionally equivalent theory has a computable model. In fact, they defined, for each infinite computable binary branching tree R with no "guessable" path, a consistent theory $T_{\mathrm{LW}}(R)$ such that no theory definitionally equivalent to $T_{\mathrm{LW}}(R)$ has a computable model. We wonder how close such models are to being computable.

QUESTION 2. *Define the Lutz–Walsh spectrum of an infinite computable binary tree R none of whose paths are guessable, say* $\mathrm{LWSpec}(R)$, *as the set of Turing degrees that compute a model of a theory definitionally equivalent to $T_{\mathrm{LW}}(R)$. It is clear that this* $\mathrm{LWSpec}(R)$ *always contains all* PA *degrees, and in* [13] *it is proven that* $\mathrm{LWSpec}(R)$ *does not contain* 0 (*if, again, none of the paths of R are guessable*). *Does* $\mathrm{LWSpec}(R)$ *ever contain any degrees that are not* PA? *If so, how far from* PA *can we get? Is there such a tree R for which* $\mathrm{LWSpec}(R)$ *consists of all nonzero degrees? Is there any nonzero degree that can never be in* $\mathrm{LWSpec}(R)$?

In the statement of Theorem 4.1, the arity of the predicate used to replace set inclusion increased by 1 for every jump in complexity. We wonder if this is a logical necessity or merely an artifact of our construction.

QUESTION 3. *For a given value of n, does there exist a theory T^n_Q, axiomatizing a single ternary predicate $Q(x, y, z)$, such that T^n_Q is definitionally equivalent to "*PA *plus all Π_n truths" which admits a computable nonstandard model? What if we require the predicate to be binary, instead?*

While Theorem 2.6 guarantees the existence of a $0'$-computable isomorphism between the original structure and its computable copy, most results in the literature can only guarantee (out of necessity) a $0''$ or $0'''$-computable isomorphism. This is explained by the fact that, in our proofs, we must first create a $0'$-computable c.e.-typed copy which is isomorphic to the original structure, though this isomorphism may itself need to be complex. Hirschfeldt asked whether one could obtain a more direct result, for example, by weakening the assumptions of the QETP.

QUESTION 4 (Hirschfeldt). *Does there exist a version of Theorem 2.6 that requires weaker assumptions, possibly using an infinite injury method instead of finite injury, from which theorems, such as Theorem 3.1 (jump inversion for linear orders) or Proposition 3.2 (jump inversion for equivalence relations with infinitely many infinite classes) may be obtained more easily?*

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

Many general results in computable structure theory necessitate the use of a finite relational signature, oftentimes for reasons similar to those that led us to the notion of $0'$-computable c.e.-typed structure. We wonder if our definition would be applicable to other scenarios.

QUESTION 5. *Are there results in the literature, applicable to finite relational signatures, which could be further (and usefully) generalized using the notion of* $0'$*-computable c.e.-typed structure*?

In Section 2.1, we found it useful to work with $\bigvee^c$ formulas, which are related to but not exactly the same as Σ^c_1 formulas. Ash, Knight, Manasse, and Slaman showed that the definability of a relation by a Σ^c_1 formula is related to computability-theoretic properties, namely, the notion of being a r.i.c.e. (relatively intrinsically c.e.) relation (see [16, Theorem II.16] for details). We wonder if $\bigvee^c$ formulas admit a similar treatment.

QUESTION 6. *Is there a computability-theoretic characterization of those relations in a structure that are definable by a* $\bigvee^c$ *formula*?

Acknowledgments. I would like to thank Denis Hirschfeldt, Miles Kretschmer, and Patrick Lutz for their valuable feedback, help, and advice, without which this work would not have been possible.

REFERENCES

[1] R. B. BAPAT, ***Graphs and Matrices***, second ed., Universitext, Springer, Hindustan Book Agency, London, 2014, pp. xii+193.

[2] N. BAZHENOV, E. FOKINA, D. ROSSEGGER, A. A. SOSKOVA, and S. V. VATEV, *A Lopez-Escobar theorem for continuous domains*. ***Journal of Symbolic Logic***, vol. 90 (2025), no. 2, pp. 854–871.

[3] W. CALVERT, A. FROLOV, V. HARIZANOV, J. KNIGHT, C. MCCOY, A. SOSKOVA, and S. VATEV, *Strong jump inversion*. ***Journal of Logic and Computation***, vol. 28 (2018), no. 7, pp. 1499–1522.

[4] R. G. DOWNEY and A. MELNIKOV, ***Computable Structure Theory: A Unified Approach***, Theory and Applications of Computability, Springer, Cham, 2026, pp. xii+540.

[5] D. S. DUMMIT and R. M. FOOTE, ***Abstract Algebra,*** vol. 3, Wiley, Hoboken, 2004.

[6] H. M. FRIEDMAN and A. VISSER, *When Bi-interpretability implies synonymy*, ***The Review of Symbolic Logic***, Vol 18, Cambridge University Press, Cambridge, 2025, no. 4, pp. 971–990.

[7] M. T. GODZISZEWSKI and J. D. HAMKINS, *Computable quotient presentations of models of arithmetic and set theory*, ***Logic, Language, Information, and Computation***, Lecture Notes in Computer Science, 10388, Springer, Berlin, 2017, pp. 140–152. https://doi.org/10.1007/978-3-662-55386-2_10.

[8] V. S. HARIZANOV, ***Pure Computable Model Theory***, Studies in Logic and the Foundations of Mathematics, 138, Elsevier, Amsterdam, 1998, pp. 3–114.

[9] W. HODGES, ***Model Theory***, Encyclopedia of Mathematics and its Applications, 42, Cambridge University Press, Cambridge, 1993, pp. xiv+772.

[10] R. KAYE, ***Models of Peano Arithmetic***, Oxford Logic Guides, 15, Oxford Science Publications. The Clarendon Press, Oxford University Press, New York, 1991, pp. x+292.

[11] R. KAYE and T. L. WONG, *On interpretations of arithmetic and set theory*. ***Notre Dame Journal of Formal Logic***, vol. 48 (2007), no. 4, pp. 497–510.

[12] J. F. KNIGHT, *Degrees coded in jumps of orderings*. ***The Journal of Symbolic Logic***, vol. 51 (1986), no. 4, pp. 1034–1042.

[13] P. LUTZ and J. WALSH, *A theory satisfying a strong version of Tennenbaum's theorem*, preprint, 2023, arXiv:2309.11598 [math.LO]. https://arxiv.org/abs/2309.11598.

[14] D. MAIA, *Escaping Tennenbaum's theorem and a strong jump inversion theorem*, preprint, 2026, arXiv:2603.04599 [math.LO]. https://arxiv.org/abs/2603.04599.

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press

[15] A. Mancini and D. Zambella, *A note on recursive models of set theories*. ***Notre Dame Journal of Formal Logic***, vol. 42 (2001), no. 2, pp. 109–115.

[16] A. Montalbán, ***Computable Structure Theory: Within the Arithmetic***, Perspectives in Logic, Cambridge University Press, Cambridge, 2021.

[17] ———, ***Computable Structure Theory: Beyond the Arithmetic***, Perspectives in Logic, Cambridge University Press, Cambridge, 2026.

[18] F. Pakhomov, *How to escape Tennenbaum's theorem*, preprint, 2022. arXiv:2209.00967 [math.LO]. https://arxiv.org/abs/2209.00967.

[19] J. H. Schmerl, *Tennenbaum's theorem and recursive reducts*, ***Set Theory, Arithmetic, and Foundations of Mathematics: Theorems, Philosophies*** (J. Kennedy and R. Kossak, editors), Lecture Notes Series, 36, Association for Symbolic Logic, La Jolla, 2011, pp. 112–149.

[20] G. Wilmers, *Bounded existential induction*. ***The Journal of Symbolic Logic***, vol. 50 (1985), no. 1, pp. 72–90.

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF CHICAGO
USA
E-mail: dmaia@uchicago.edu

https://doi.org/10.1017/jsl.2026.10219 Published online by Cambridge University Press