ISSN 1995-0802, Lobachevskii Journal of Mathematics, 2021, Vol. 42, No. 4, pp. 735–742. © The Author(s), 2021. This article is an open access publication, corrected publication 2021.

Punctual Categoricity Relative to a Computable Oracle

I. Sh. Kalimullin[1]* and A. G. Melnikov[2]**

(Submitted by M. M. Arslanov)

[1] *N. I. Lobachevskii Institute of Mathematics and Mechanics, Kazan (Volga Region) Federal University, Kazan, 420008 Russia*

[2] *School of Natural and Computational Sciences, Massey University, Auckland, New Zealand*

Received November 4, 2020; revised November 25, 2020; accepted December 15, 2020

Abstract—We are studying the punctual structures, i.e., the primitive recursive structures on the whole set of integers. The punctual categoricity relative to a computable oracle f means that between any two punctual copies of a structure there is an isomorphism which togeteher with its inverse can be derived via primitive recursive schemes augmented with f. We will prove that the punctual categoricity relative to a computable oracle can hold only for finitely generated or locally finite structures. We will show that the punctual categoricity of finitely generated structures is exhaused by the computable oracles with primitive recursive graph. We also present an example of locally finite structure where the punctual categoricity is provided by a primitive recursively bounded computable oracle.

DOI: 10.1134/S1995080221040107

Keywords and phrases: *primitive recursive function, computable function, algebraic structure, finitely generated structure, locally finite structure.*

1. INTRODUCTION

While classical algebra views algebraic structures up to isomorphism, the hallmark of computable structure theory is the study of computable algebraic structures up to computable isomorphism. Such investigations can be traced back to Malcev [1] who noted that already the vector space V_∞ over $\mathbb{Q}$ of infinite dimension has two computable presentations which are not computably isomorphic. In modern terms, V_∞ is not computably categorical. Over the past 50+ years computable structure theory has accumulated a plethora of examples and results supporting the intuition that, in a natural algebraic class, computably categorical structures tend to be algebraically tame. This intuition can be made formal using the closely related notion of relative computable categoricity [2]. Although very complex examples of computably categorical structures are known to exist (e.g., [3, 4]), algebraically interesting computable structures tend to be not computably categorical. This phenomenon led to an extensive study of structures categorical relative to a non-computable oracle; we cite [2, 5] for the classical results and the general framework of Δ^0_n-categorical structures, and we cite [6–10] for several more recent results on this topic.

Similarly to computable structure theory, one of the main topics of feasible structure theory is the study of computationally feasible structures up to feasible isomorphism. Here "feasible" can be interpreted in many different ways, e.g, automatic, polynomial-time, primitive recursive, or punctual (to be defined). The general intuition is that, when compared to computably categorical structures, feasibly categorical structures tend to be even more rare (but see [11, 12] for artificial counterexamples). For instance, a structure is automatically categorical iff it is finite [13]. Similarly, only finite structures can be polynomial-time categorical [14]. An infinite locally finite structure cannot be primitively recursively categorical; this in particular implies that no infinite relational structure can be primitively recursively

*E-mail: Iskander.Kalimullin@kpfu.ru

**E-mail: alexander.g.melnikov@gmail.com

categorical [14]. Nonetheless, there has been essentially no systematic study of feasible structures categorical relative to an oracle; the only exception is the brief note [15]. The main aim of this paper is to initiate a systematic theory of punctual categoricity relative to an oracle; we note that in contrast with computable structure theory, our oracles will tend to be computable. To discuss our results, we need to explain what punctual computability means.

According to [12], a structure in a finite language is punctual if its domain is $\mathbb{N}$, and its operations and relations are primitive recursive. This definition is similar to the notion of a primitive recursive structure suggested by Maltsev [1] and then used by Cenzer and Remmel [16]. The only subtle but important difference is that the domain has to be the whole of $\mathbb{N}$ and not merely a primitive recursive subset of $\mathbb{N}$. This subtle strengthening of the definition led to an unexpectedly rich theory of punctual structures; we cite the survey [17] for a detailed exposition of the new emerging theory.

All results in the previous works on punctual categoricity can be subdivided into two categories:

- The study of punctually categorical structures. This means that between every pair of punctual copies of the structure there is a primitive recursive isomorphism with a primitive recursive inverse.

- Examples of punctual structures with punctually unbounded isomorphisms. This means that between each pair of punctual copies of the structure every isomorphism can not be bounded by a primitive recursive function.

Moreover, in the second case the growth of the isomorphism usually is tightly connected with its punctual complexity due the primitive recursiveness of the graphs of such isomorphisms. It is easy to see that if the graph of a function is primitive recursive then either the function is primitive recursive or, otherwise, it has to be primitively recursively unbounded.

For example, if a punctual structure is finitely generated, then there is a "standard" copy of the structure which is the term algebra upon some fixed finite set of generators. This copy can be primitively recursively isomorphically mapped onto every other punctual copy of the structure. However, the inverse of this isomorphism is typically not primitive recursive, although its graph is. This effect can be exploited to code every computable function with a primitive recursive graph to the complexity of isomorphisms of finitely generated structures, in the following sense. For two functions $f, g : \mathbb{N} \to \mathbb{N}$, write $f \leq_{PR} g$ if f can be derived from g and the basic primitive recursive functions using finitely many applications of composition and primitive recursion. We write $f \equiv_{PR} g$ if $f \leq_{PR} g$ and $g \leq_{PR} f$. The theorem below essentially says that every computable function f with a primitive recursive graph can serve as the degree of punctual categoricity of a finitely generated structure, much in the spirit of the well-known analogous results for computable structures; we will discuss this in a bit more detail at the end of the introduction.

Theorem 1 (Kalimullin, Melnikov, Ng [15]). *For every computable function f with a primitive recursive graph there is a finitely generated punctual structure $\mathcal{A}$ such that*

1) there is a punctual copy $\mathcal{B}$ of $\mathcal{A}$ such that for every isomorphism g from $\mathcal{B}$ to $\mathcal{A}$ we have $f \leq_{PR} g$;

2) for every punctual copy $\mathcal{C}$ of $\mathcal{A}$ there is an isomorphism h from $\mathcal{C}$ to $\mathcal{A}$ such that $h, h^{-1} \leq_{PR} f$.

Corollary 1. *For every computable function f with a primitive recursive graph there is a finitely generated punctual structure $\mathcal{A}$ such that for every function g the following two conditions are equivalent:*

a) $f \leq_{PR} g$;

b) for every punctual copy $\mathcal{C}$ of $\mathcal{A}$ there is an isomorphism h from $\mathcal{C}$ to $\mathcal{A}$ such that $h, h^{-1} \leq_{PR} g$.

Our primary technical goal is to understand which punctual structures $\mathcal{A}$ and computable functions f can satisfy the above two properties, and also whether Theorem 1 can be generalized beyond finitely generated structures. Our first result shows that the second condition of Theorem 1 alone puts a strong algebraic restriction on the potential algebraic type of the structure.

Theorem 2. *Let $\mathcal{A}$ be a punctual structure and f be a computable function such that or every punctual copy $\mathcal{C}$ of $\mathcal{A}$ we have an isomorphism $h : \mathcal{C} \to \mathcal{A}$ such that $h, h^{-1} \leq_{PR} f$. Then the structure $\mathcal{A}$ is either finitely generated or is locally finite.*

Considering only finitely generated structures we can easily illustrate that every function f with the properties 1)–2) from Theorem 1 must be $\equiv_{PR}$-equivalent to a function with a primitive recursive graph, as follows. For the term algebra $\mathcal{T}$ of $\mathcal{A}$ upon a fixed finite set of generators, fix primitive recursive isomorphisms $h_1 : \mathcal{T} \to \mathcal{A}$ and $h_2 : \mathcal{T} \to \mathcal{B}$ and consider the isomorphism $g = h_2 \circ h_2^{-1} : \mathcal{B} \to \mathcal{A}$. We have

$$f \leq_{PR} h_1 \circ h_2^{-1} \leq_{PR} h_2^{-1} \leq_{PR} f$$

so that $f \equiv_{PR} h_2^{-1}$ and the graph of h_2^{-1} is obviously primitive recursive. As we observed earlier, a primitively recursively bounded function with a primitive recursive graph must be primitive recursive. Therefore, f either is primitive recursive, or is not primitively recursively bounded. Our second result below illustrates that the same property as above holds if we assume the conditions of Corollary 1.

Theorem 3. *Let f be a computable function and let $\mathcal{A}$ be a finitely generated punctual structure such that for every computable function g the following two conditions are equivalent:*

a) $f \leq_{PR} g$;

b) for every punctual copy $\mathcal{C}$ of $\mathcal{A}$ there is an isomorphism h from $\mathcal{C}$ to $\mathcal{A}$ such that $h, h^{-1} \leq_{PR} g$.

Then there is a a computable function $f' \equiv_{PR} f$ such that the graph of f' is primitive recursive.

If a structure is not finitely generated then the behavior of its isomorphisms, of course, can potentially be much more complicated. Nonetheless, in all known examples in the literature, a punctual structure is either punctually categorical, or it is built so that all isomorphisms unbounded from or to the structure are not primitively recursively bounded.

All these results and examples raise the question: suppose a punctual $\mathcal{A}$ has the property that for every other punctual copy $\mathcal{B}$ and each isomorphism $f : \mathcal{A} \to \mathcal{B}$, both f and f^{-1} are bounded by a primitive recursive function. Does this imply that $\mathcal{A}$ is punctually categorical? Also, can we extend Theorem 1 to PR-degrees of functions whose graphs are not primitive recursive? Recall that a function with a primitive recursive graph either is primitive recursive or must be primitively recursively unbounded. Also recall that, in the finitely generated case, a primitively recursively bounded isomorphism must be primitive recursive; see the discussion before Theorem 3. What if we drop the condition of being finitely generated? By Theorem 2, such a generalization of Theorem 1 must be witnessed by a locally finite structure. Our third (and main) result says that, remarkably, the answer to both questions is negative.

Theorem 4. *There is a punctual locally finite structure $\mathcal{A}$ which is not punctually categorical such that for every punctually categorical copies $\mathcal{B}, \mathcal{C}$ of $\mathcal{A}$ there is a computable isomorphism h, which together with its inverse is bounded by a primitive recursive function.*

The proof of Theorem 4 relies on the relatively complex "pressing strategy" introduced in [12] and then explained in much detail in [11, 17]. The strategy is combinatorially quite involved, but if the reader is familiar with the strategy they should have no problem understanding the proof. Since the strategy has been explained multiple times in the literature, we decided to keep the paper short and give only a hint to how it works. The reader will have to either take that it works for granted or look it up in the aforementioned papers.

Our results and questions are analogous to the study of degrees of computable categoricity initiated in [18] and the further developed in, e.g., [19–23]. Theorem 3 is similar to the following result on degrees of categoricity for structures with a finite automorphism group [24, 25]: the degree of categoricity of a structure $\mathcal{A}$ with $|Aut(\mathcal{A})| < \infty$ can be directly decomposed into the degrees of isomorphisms between finitely many computable copies of the structure. Turetsky [26] has constructed an example of a structure having infinite automorphism group whose degree of categoricity does not have this decomposability property. These results motivate similar questions for punctual categoricity, but in the punctual case finitely generated and locally finite structures are the most interesting subclasses. In particular, we will see that the proof of Theorem 3 shows that the function f up to $\equiv_{PR}$-equivalence can be decomposed into finitely many isomorphisms $u_0, u_1, \ldots, u_{n-1}$. It is unknown, however, whether we can adapt Theorem 1 to realise such decomposition for arbitrary $n > 1$; we leave this as an open problem.

2. PROOF OF THEOREM 2

Suppose the structure $\mathcal{A}$ is not finitely generated and is not locally finite. Fix a computable listing of all pairs of functions $(h_i, g_i)_{i\in\omega}$ such that $h_i \leq_{PR} f$ and $g_i \leq_{PR} f$. We are building a punctual copy $\mathcal{C} \cong \mathcal{A}$ meeting the requirements

$$R_i: \ g_i \text{is an isomorphism from } \mathcal{A} \text{ to } \mathcal{C} \implies g_i \circ h_i \neq \mathrm{id}$$

for every $i \in \omega$. The requirements will be met one-by-one in a finitary fashion; in particular, there will be no injury.

To satisfy R_i, we will use a finite tuple $\mathcal{C}_i \subseteq \mathcal{C}$ isomorphically mapped to $\mathcal{A}_i \subseteq \mathcal{A}$. For $i = 0$ let $\mathcal{A}_0$ be any finite tuple in $\mathcal{A}$ which generates and infinite substructure of $\mathcal{A}$ which is not the whole of $\mathcal{A}$, and let $\mathcal{C}_0$ be the exact copy of $\mathcal{A}_0$.

To meet R_i, assume the finite tuples $\mathcal{C}_i$ and $\mathcal{A}_i$ have already been defined. Keep extending $\mathcal{C}_i$ punctually by adding a few more new elements generated by $\mathcal{C}_i$ and mapping them to the naturally corresponding elements generated by $\mathcal{A}_i$. Meanwhile, do the following.

1. For every $c \in \mathcal{C}_i$, compute the values $h_i(c)$ and $g_i(h_i(c))$.

2. If $g_i(h_i(c)) \neq c$ for some $c \in \mathcal{C}_i$ we are done. Otherwise, if $g_i(h_i(c)) = c$ for every $c \in \mathcal{C}_i$, we start to compute the value $g_i(a)$ for each $a \in \mathcal{A}$ until we find an evidence of non-injectivity of g_i, or an evidence that g_i does not preserve an operation from the language of $\mathcal{A}$.

Note that, since $\mathcal{A}$ is not finitely generated we should have an $a \in \mathcal{A}$ which can not be generated from $\{h_i(c)|c \in \mathcal{C}_i\}$. But the value $g_i(a)$ is forced to be generated from $\mathcal{C}_i$, so that for some term t we have

$$g_i(a) = t(c_1, \ldots, c_n) = t(g_i(h_i(c_1)), \ldots, g_i(h_i(c_n)))$$

for some $c_1, \ldots, c_n \in \mathcal{C}_i$. Thus, g_i either does not respect an operation from the term t, or is not injective since

$$g_i(a) = t(g_i(h_i(c_1)), \ldots, g_i(h_i(c_n))) = g_i(t(h_i(c_1), \ldots, h_i(c_n))) \quad \text{and} \quad a \neq t(h_i(c_1), \ldots, h_i(c_n)).$$

Therefore, the computation in 2. eventually halts. In either case, the requirement is clearly met.

To make sure that $\mathcal{C}$ is isomorphic to $\mathcal{A}$, let $\mathcal{A}_{i+1}$ be the finite part of $\mathcal{A}$ enumerated so far together with the element having the smallest index which has not yet been listed in $\langle A_i \rangle$. Define $\mathcal{C}_{i+1}$ by introducing an isomorphic image of this extra element to $\mathcal{C}$. Go to the next requirement.

3. PROOF OF THEOREM 3

Let f be a computable function f and let $\mathcal{A}$ be a finitely generated punctual structure such that for every computable function g the following two conditions are equivalent:

a) $f \leq_{PR} g$;

b) for every punctual copy $\mathcal{C}$ of $\mathcal{A}$ there is an isomorphism h from $\mathcal{C}$ to $\mathcal{A}$ such that $h, h^{-1} \leq_{PR} g$.

Fix also the punctual term algebra $\mathcal{T}$ of $\mathcal{A}$ over a fixed finite set of generators. There is a series of obvious facts about $\mathcal{T}$ which are collected in the following lemma.

Lemma 1.

1) If $\mathcal{C}$ is a punctual copy of $\mathcal{A}$ then every isomorphism from $\mathcal{T}$ onto $\mathcal{C}$ is primitive recursive. In particular, every automorphism of $\mathcal{T}$ is primitive recursive.

2) If $\mathcal{C}$ is a punctual copy of $\mathcal{A}$ and u, v are isomorphisms from $\mathcal{C}$ onto $\mathcal{T}$ then $u \equiv_{PR} v \leq_{PR} f$.

3) There is a uniformly computable list $\{u_i\}_{i\in\omega}$ of computable functions which contains all isomorphisms from punctual copies of $\mathcal{A}$ onto $\mathcal{T}$ and almost null functions (i.e., $u_i(x) = 0$ beginning with some x). In particular, $u_i \leq_{PR} f$ and the graph of u_i is primitive recursive for every i (but not uniformly).

4) For every isomorphism h from one punctual copy of $\mathcal{A}$ onto another punctual copy of $\mathcal{A}$, we have $h \leq_{PR} u_i$ for some i.

Proof. 1) Let w be an isomorphism from $\mathcal{T}$ onto $\mathcal{C}$. Then for each term $t \in \mathcal{T}$ the value $w(t)$ can be computed immediately as the same term from the w-images of the generators. The w-images of the generators can be fixed non-uniformly.

2) The equivalence $u \equiv_{PR} v$ follows from 1) since $u \circ v^{-1}$ and $v \circ u^{-1}$ are primitive recursive automorphisms of $\mathcal{T}$ such that $u = (u \circ v^{-1}) \circ v$ and $v = (v \circ u^{-1}) \circ u$. Also we know from the conditions on f that, for some isomorphism u from $\mathcal{C}$ onto $\mathcal{T}$, we have $u \leq_{PR} f$.

3) The list $\{u_i\}_{i\in\omega}$ can be obtained from a uniformly computable list of all triples $\{\mathcal{A}_i, p_i, \varphi_i^f\}_{i\in\omega}$, where $\mathcal{A}_i$ is a punctual structure in the language of $\mathcal{A}$, p_i is a primitive recursive function, and φ_i^f is the i-th primitive recursive scheme augmented with f. To define u_i, copy the function φ_i^f until it is discovered that either $p_i \circ \varphi_i^f \neq \mathrm{id}$, or $\varphi_i^f \circ p_i \neq \mathrm{id}$, or $p_i(\mathcal{T}) \neq \mathcal{A}_i$. In this case set u_i equal to 0 for almost all inputs.

4) Follows immediately from 1)–3). □

For a sequence of functions $\{v_i\}_{i\in\omega}$ denote the function $v(\langle i, x\rangle) = v_i(x)$ by $v = \oplus_i v_i$. The notation $\oplus_{i<n} v_i$ means that the components v_i in $\oplus_i v_i$ are null functions for $i \geq n$.

Note that if for the sequence $\{u_i\}_{i\in\omega}$ from Lemma 1 and some n we have $f \equiv_{PR} \oplus_{i<n} u_i$ then Theorem 3 is witnessed by the function $f' = \oplus_{i<n} u_i$ whose graph is primitive recursive. Otherwise, if $f \not\leq_{PR} \oplus_{i<n} u_i$ for every n, then we get a computable function g for which the equivalence between a) and b) does not hold:

Lemma 1. *Let f be a computable function and let $\{u_i\}_{i\in\omega}$ be a uniformly computable list of functions such that $f \not\leq_{PR} \oplus_{i<n} u_i$ for every n. Then there is a computable function g such that $f \not\leq_{PR} g$ and $u_i \leq_{PR} g$ for every i.*

Proof. The proof is similar to the proof of well-known Spector's Theorem for Turing reducibility with obvious modifications. We give more details.

We are constructing a computable function g stage-by-stage defining a finite function g_n at each stage n such that $g_n \supset g_{n-1}$. Let $g_{-1} = \emptyset$.

At a stage $n \geq 0$ we ensure that $f \neq \varphi_n^g$ for the n-th primitive recursive scheme φ_n augmented by $g = \cup_n g_n$. To do this we consider the computable function

$$h_n(\langle i, x\rangle) = \begin{cases} g_{n-1}(\langle i, x\rangle), & \text{if } g_{n-1}(\langle i, x\rangle) \text{ is defined;} \\ u_i(x), & \text{if } g_{n-1}(\langle i, x\rangle) \text{ is undefined and } i < n; \\ 0, & \text{otherwise.} \end{cases}$$

It is easy to see that $h_n \leq_{PR} \oplus_{i<n} u_i$, so that we must have $f \not\leq_{PR} h_n$, and hence $f \neq \varphi_n^{h_n}$. Therefore, we can compute a finite part g_n of h_n, $g_{n-1} \subset g_n \subset h_n$, which forces the inequality $f \neq \varphi_n^g$.

By the construction above, if $g_i(\langle i, x\rangle)$ is undefined then $u_i(x) = g(\langle i, x\rangle)$, so that for each i there are only finite many x for which we can have $u_i(x) \neq g(\langle i, x\rangle)$. Hence, $u_i \leq_{PR} g$ for every i. □

4. PROOF OF THEOREM 4

We build a rigid punctual structure $\mathcal{A}$ which is not punctually categorical but such that, between any pair of punctual presentations of $\mathcal{A}$, the unique isomorphism between these two copies is punctually bounded.

The pressing strategy. We explain only the idea. For details, see [11, 12, 17]. The main purpose of this strategy is to "press" the opponent's punctual structure to reveal the needed segment of his structure within a precomputed number of stages. The basic idea is as follows. Fix a language consisting of three unary functions s, u, p. Begin building $\mathcal{A}$ with a sequence of u-cycles of length 2 arranged into an omega-chain using s. We call the first element in the chain the origin of the chain. We also map every 2-cycle to the origin of the chain using p.

Keep padding the structure with 2-cycles until the first primitive recursive structure P_0 gives a sequence of two consequent 2-cycles. If P_0 shows some other configuration which does not even look like a chain of cycles, then P_0 is not isomorphic to $\mathcal{A}$. If P_0 reveals (say) a 3-cycle instead, we can restrain 3 and agree to never use it in the construction.

Thus, if $P_0 \cong \mathcal{A}$, then we must eventually see two or more consequent 2-cycles connected by a unary function in P_0. As soon as we the two consequent 2-cycles are discovered, we switch the gears and

continue building the ω-chain in $\mathcal{A}$ using a sequence of alternating 2-cycles and 4-cycles. Thus, at some later stage we will have the follwing configuration in $\mathcal{A}$:

$$2{-}2{-}2{-}\ldots(n_0\,\text{times}){-}2{-}2{-}4{-}2{-}4{-}2{-}4{-}2,$$

where n_0 depends on the stage at which we switched gears.

Every cycle in the sequence is connected to the left-most point in the omega-chain using another unary function. Thus, given a 2-cycle in P_0 we can instantly compute the origin with the speed of the enumeration of P_0; just evaluate the unary function on any point of the 2-cycle. Since n_0 depends on ly on the speed of the enumeration of P_0, we know that after at most n_0 many applications of the unary function which was used to form the ω-chain to the origin of the chain, we will generate the sequence of two consequent 2-cycles in P_0. If this does not happen, $P_0 \ncong \mathcal{A}$. This gives a way to match elements of P_0 with the corresponding elements in $\mathcal{A}$ with a primitive recursive delay that depends on P_0.

At a later stage we use the sequence of 2-cycles and 4-cycles to press two structures at once, namely P_0 and P_1. We will wait until both of them respond by giving the sequence $2{-}4{-}2$ or prove that they are not isomorphic to $\mathcal{A}$. Then we switch gears again and will, for example, use the pattern $2{-}4{-}4$ to press three structures P_0, P_1, P_2, etc. This finishes the informal description of the idea. See [12] for further details.

Terminating a chain. At the stage at which we decide to switch gears, we can choose to stop building the ω-like chain by (say) mapping its current end-point to itself. Then we can initiate a new chain of cycles with a new origin. Every cycle in the new chain will be mapped to the new origin. It will be completely disconnected from the previous chain. Then we can finish building the chain and initiate a new one if necessary, etc.

Note that each such finite chain of cycles will have a unique isomorphism type. In $\mathcal{A}$, let x_i denote a finite chain that is being constructed (i.e., has not been finished yet); we say that x_i is currently open. If at some stage the chain has been terminated, i.e., we have already finished building the chain and started a new one, then we say that x_i is closed.

Proof idea. The language of $\mathcal{A}$ consists the three unary functions s, u, p required to implement the pressing strategy, one extra unary function v, and a unary predicate D.

The structure $\mathcal{A}$ will consist of pairs of finite chains, in the sense of the previous subsection. Every element of such finite chain, perhaps with the exception of the last element, will satisfy D. At every stage we will keep exactly two finite chains, say x_i and y_i, open. When we close x_i and y_i, which will happen simultaneously, for exactly one of the two chains the end-point will satisfy D. Before the chains are closed, these two chains will be kept identical. Furthermore the respective elements in x_i and y_i will be mapped to each other using the unary function v. We say that these two chains are conjugate (via v) in $\mathcal{A}$.

It follows that, if $P_i \cong \mathcal{A}$, then it must reveal its versions of the two chains with a precomputed delay. This will be used to show that the structure satisfies the primitive recursive boundedness stated in the theorem. Nonetheless, we have the freedom to make the two chains non-isomorphic when we close them. We use this for a straightforward diagonalization as follows.

Build the second punctual copy $\mathcal{B}$ of $\mathcal{A}$ and meet

$$p_i : \mathcal{B} \to \mathcal{A} \text{ is not an isomorphism}$$

using x_i and y_i as follows.

- Wait for a primitive recursive $p_i : \mathcal{A} \to \mathcal{B}$ to converge on x_i and y_i. Let ξ_i be the end-point of x_i
- When x_i and y_i are ready to be closed, make sure that $D(\xi_i) \neq D(p_i(\xi_i))$.

The formal construction. We simultaneously build $\mathcal{A} \cong \mathcal{B}$. To meet the i-th diagonalization requirement, do the following substeps.

1. Initiate the enumeration of two conjugate chains $x_i^{\mathcal{A}}$ and $y_i^{\mathcal{A}}$ in $\mathcal{A}$ and also identical conjugate chains $x_i^{\mathcal{B}}$ and $y_i^{\mathcal{B}}$ in $\mathcal{B}$. Declare $D(\eta)$ for every element of the chains, unless specified otherwise. The isomorphism type of the chains is determined according to the pressing strategy.

2. Wait for $p_i : \mathcal{A} \to \mathcal{B}$ to converge on the origins of $x_i^{\mathcal{A}}$ and $y_i^{\mathcal{A}}$.

3. If the p_i-images of the origins of $x_i^{\mathcal{A}}$ and $y_i^{\mathcal{A}}$ are not the (distinct) origins of $x_i^{\mathcal{B}}$ and $y_i^{\mathcal{B}}$, then:

(a) Wait for a stage at which the chains can be declared closed according to the pressing strategy.

(b) Close the chains in both $\mathcal{A}$ and $\mathcal{B}$, and proceed to 1. for the next requirement.

4. Otherwise, if the origins are mapped to the origins (but perhaps in a different order), wait for a stage at which the chains can be declared closed according to the pressing strategy.

5. When the chains are declared closed, let ξ_i and ρ_i be the end-points of $x_i^{\mathcal{A}}$ and $y_i^{\mathcal{A}}$, respectively.

6. Declare $D(\xi_i)$ in $\mathcal{A}$ and $\neg D(p_i(\xi_i))$ in $\mathcal{B}$.

7. Declare $\neg D(\rho_i)$ in $\mathcal{A}$ and $D(p_i(\rho_i))$ in $\mathcal{B}$.

8. Proceed to the next diagonalization requirement.

The verification. It is clear that every diagonalization requirement is met, and thus $\mathcal{A}$ and $\mathcal{B}$ are not punctually isomorphic. Note that $\mathcal{A}$ is rigid. To see why $\mathcal{A}$ has the property of primitive recursive boundedness of the isomorphisms, suppose $P_i \cong P_j \cong \mathcal{A}$. The pressing strategy ensures that every point in P_i can be primitively recursively matched with a point in one of the two chains which are currently open, and thus the isomorphism is bounded by the function that takes the maximum index of the two potential images; the same can be said about the inverse of the isomorphism inverse. Since the same argument can be applied to P_j, we can derive a primitive recursive bound on the isomorphism between P_i and P_j as well as on its inverse. This finishes the proof of the theorem.

FUNDING

The work was supported by the Russian Science Foundation, project no. 18-11-00028. The second author was supported by Rutherford Discovery Fellowship and the Marsden Fund of New Zealand.

OPEN ACCESS

REFERENCES

1. A. Malcev, "Constructive algebras. I," Usp. Mat. Nauk **16** (3), 3–60 (1961).
2. C. Ash and J. Knight, *Computable Structures and the Hyperarithmetical Hierarchy,* Vol. 144 of *Studies in Logic and the Foundations of Mathematics* (North-Holland, Amsterdam, 2000).
3. R. G. Downey, A. M. Kach, S. Lempp, A. E. M. Lewis-Pye, A. Montalbán, and D. D. Turetsky, "The complexity of computable categoricity," Adv. Math. **268**, 423–466 (2015).
4. A. G. Melnikov and K. M. Ng, "Computable torsion abelian groups," Adv. Math. **325**, 864–907 (2018).
5. C. Ash, "Recursive labeling systems and stability of recursive structures in hyperarithmetical degrees," Trans. Am. Math. Soc. **298**, 497–514 (1986).
6. R. Downey, A. G. Melnikov, and K. M. Ng, "Abelian p-groups and the Halting problem," Ann. Pure Appl. Logic **167**, 1123–1138 (2016).
7. E. J. Barker, "Back and forth relations for reduced abelian p-groups," Ann. Pure Appl. Logic **75**, 223–249 (1995).
8. R. Downey, A. G. Melnikov, and K. M. Ng, "On δ_2^0-categoricity of equivalence relations," Ann. Pure Appl. Logic **166**, 851–880 (2015).
9. A. G. Melnikov, "Torsion-free abelian groups with optimal Scott families," J. Math. Logic **18**, 1850002 (2018).
10. V. O. González, "Computability in the class of real closed fields," Ph.D. Thesis (Notre Dame Univ., 2014).
11. R. Downey, N. Greenberg, A. G. Melnikov, K. M. Ng, and D. Turetsky, "Punctual categoricity and universality," J. Symbol. Logic (in press).

12. I. S. Kalimullin, A. G. Melnikov, and K. M. Ng, "Algebraic structures computable without delay," Theor. Comput. Sci. **674**, 73–98 (2017).
13. B. Khoussainov and A. Nerode, "Automatic presentations of structures," in *Logical and Computational Complexity, Proceedings of the International Workshop LCC'94, Indianapolis, Indiana, USA, October 13–16, 1994* (1994), pp. 367–392.
14. P. E. Alaev, "Existence and uniqueness of structures computable in polynomial time," Algebra Logic **55**, 72–76 (2016).
15. I. S. Kalimullin, A. G. Melnikov, and K. M. Ng, "The diversity of categoricity without delay," Algebra Logic **56**, 171–177 (2017).
16. D. Cenzer and J. Remmel, "Polynomial–time abelian groups," Ann. Pure Appl. Logic **56**, 313–363 (1992).
17. N. Bazhenov, R. Downey, I. Kalimullin, and A. Melnikov, "Foundations of online structure theory," Bull. Symbol. Logic **25**, 141–181 (2019).
18. E. B. Fokina, I. Kalimullin, and R. Miller, "Degrees of categoricity of computable structures," Arch. Math. Logic **49**, 51–67 (2010).
19. B. F. Csima, J. N. Y. Franklin, and R. A. Shore, "Degrees of categoricity and the hyperarithmetic hierarchy," Notre Dame J. Formal Logic **54**, 215–231 (2013).
20. E. Fokina, V. Harizanov, and D. Turetsky, "Computability-theoretic categoricity and Scott families," Ann. Pure Appl. Logic **170**, 699–717 (2019).
21. N. A. Bazhenov, "Degrees of autostability relative to strong constructivizations for Boolean algebras," Algebra Logic **55**, 87–102 (2016).
22. N. A. Bazhenov and M. M. Yamaleev, "Degrees of categoricity of rigid structures," Lect. Notes Comput. Sci. **10307**, 152–161 (2017).
23. l N. A. Bazhenov, I. S. Kalimullin, and M. M. Yamaleev, "Strong degrees of categoricity and weak density," Lobachevskii J. Math. **41**, 1630–1639 (2020).
24. N. A. Bazhenov, I. S. Kalimullin, and M. M. Yamaleev, "Degrees of categoricity vs. strong degrees of categoricity," Algebra Logic **55**, 173–2177 (2016).
25. N. A. Bazhenov, I. Sh. Kalimullin, and M. M. Yamaleev, "Degrees of categoricity and spectral dimension," J. Symbol. Logic **83**, 103–116 (2018).
26. D. Turetsky, "Coding in the automorphism group of a computably categorical structure," J. Math. Logic **20** (3) (2020, in press).