PUNCTUAL DEGREES AND LATTICE EMBEDDINGS

ISKANDER KALIMULLIN, ALEXANDER MELNIKOV, AND MAXIM ZUBKOV

Abstract. We systematically investigate into the online content of finitely generated structures. We show that a countable lattice L is embeddable into the punctual degrees of a rigid f.g. structure if, and only if, L is distributive. The right-to-left implication is achieved by embedding the atomless Boolean algebra below any non-zero punctual degree of a given f.g. rigid structure. In contrast, we show that a lower cone is never a Boolean algebra, for any (not necessarily rigid!) f.g. structure. We also prove that the punctual degrees of the standard arithmetic are neither a lower semi-lattoce nor an upper semi-lattice.

Keywords: finitely generated structure; isomorphism; primitive recursive function.

1. Introduction

Beginning in the 1980's there has been quite a lot of work on online infinite combinatorics; see [Kie81, Kie98, KPT94, LST89, Rem86]. However, there is no general framework in the literature, there is only a taxonomy of algorithms. This paper contributes to the recently suggested framework [KMN17, BDKM] which aims to develop a general theory behind online computation.

We concentrate on the natural case of finitely generated (f.g.) structures. Decision procedures in such structures are intrinsic, in the following sense. For example, if G is a finitely presented group and it has an algorithmically decidable word or conjugacy problem, then every finite presentation $H \cong G$ will also have the problem decidable. Indeed, fix some generators $\bar{g}$ of G and their isomorphic counterparts $\bar{h}$ in H. Then every element of G is a word in the alphabet of $\bar{g}$, and it can be naturally mapped to the respective word in the alphabet of $\bar{h}$. This observation is heavily exploited in combinatorial group theory [Hig61, LS01], often without explicit reference. In the terminology of computable structure theory [EG00, AK00], finitely generated structures are *relatively computably categorical*. Notice, however, that the observation above uses a rather general notion of a Turing computable process. Imagine that you are given an algorithmic description $\mathcal{C} = \langle c \rangle$ of the infinite cyclic group. Your task is to associate every $x \in \mathcal{C}$ with an integer in the standard representation of $(\mathbb{Z}, +)$. The naive algorithm waits for a late enough stage at which $mc = x$. However, how long will you have to wait? It is easy to diagonalise against all polynomial time, all exponential, or even all hyper-exponential (etc.) algorithms. In other words, this procedure uses an instance of a *truly unbounded search*; this is the same as to say that the algorithm is not *primitive recursive* [Rog87].

2010 *Mathematics Subject Classification.* Primary 03D45, 03C57. Secondary 03D75, 03D80.
The authors were partially supported by Marsden Fund of New Zealand.

It is natural to ask what happens if we forbid unbounded search. Kalimullin, Melnikov, and Ng [KMN17] proposed that an "online" algebraic structure must minimally satisfy the following general definition.

Definition 1.1. A countable structure is *punctually computable* if its domain is $\mathbb{N}$ and the operations and predicates of the structure are (uniformly) primitive recursive[1].

Of course, one naturally seeks to understand the truly efficient algorithms, in the spirit of, e.g., [KN95, BG00, CR98]. Kalimullin, Melnikov and Ng [KMN17] observed that producing a punctual presentation is often sufficient to obtain a polynomial-time one; e.g., [Gri90]. Therefore it makes sense to develop a general theory of punctual algebraic and combinatorial structures; see survey [BDKM] for the foundations of this new emerging theory. See also Alaev [Ala17, Ala18] for an alternative approach.

Recall that the inverse of a primitive recursive function does not have to be primitive recursive. Fix a punctual structure $\mathcal{A}$. The collection of all punctual presentations of $\mathcal{A}$ carries a natural *reduction*, as defined below.

Definition 1.2. Let $\mathcal{A}$ be a punctual structure. Then, for punctual $\mathcal{C}, \mathcal{B}$ isomorphic to $\mathcal{A}$,

$$\mathcal{C} \leq_{pr} \mathcal{B} \text{ if there exists a surjective primitive isomorphism } f : \mathcal{C} \to_{onto} \mathcal{B}.$$

This leads to an equivalence relation $\cong_{fpr}$ and *the punctual degree structure* on the equivalence classes which will be denoted $FPR(\mathcal{A})$, where FPR stands for "fully primitive recursive". What does $FPR(\mathcal{A})$ reflect? If $\mathcal{C} \leq_{pr} \mathcal{B}$ then, in a way, $\mathcal{B}$ has more online content than $\mathcal{C}$ does. For example, the standard copy of $(\mathbb{Q}, <)$ punctually embeds any other punctual copy of the rationals, but some other copies may have slow intervals. The FPR degrees serves as a punctual invariant of a structure. A reader familiar with the terminology of computable structure should compare FPR degrees with degree spectra and categoricity spectra of structures. Note that non-trivial degree or categoricity spectra are typically realised by unnatural structures which have to be specifically constructed; see, e.g., [Mil01, KKM13, FKH$^+$12, FKM10]. In contrast, the FPR degrees of natural and common algebraic structures such as the dense linear order or the random graph seem to possess remarkably non-trivial properties which are yet to be understood. Therefore, it makes sense to study the FPR degrees of specific and natural algebraic structures and compare them with FPR degrees of other structures. For example, the FPR degrees of the dense linear order, the random graph, and the universal countable abelian p-group are pairwise non-isomorphic; see [MN]. It is not known whether the FPR degrees of $(Q, <)$ and the atomless Boolean algebra are isomorphic [BDKM].

We go back to the finitely generated case. Suppose A is a punctual f.g. structure. What can be said about $FPR(A)$? Obviously, $FPR(A)$ will have the least element $\mathbf{0}$ which is the "natural" term-representation of A. It is known that the FPR degrees

[1]To include finite structures we also allow initial segments of $\mathbb{N}$ to serve as domains. We will never consider infinite languages in the article. Also, Kalimullin, Melnikov and Ng called punctually computable structures "fully primitive recursive". However, this term is a bit too lengthy.

of a f.g. structure can be a singleton [KMN17]. Bazhenov, Kalimullin, Melnikov and Ng [BKMN] have showed that the FPR degrees of any f.g. structure are dense. They also showed that there exists a f.g. rigid A such that $FPR(A)$ is infinite, yet $FPR(A)$ has a greatest element. However, apart from the mentioned above density, no result in [BKMN] gives any information about the algebraic structure of the FPR degrees of common f.g. structures. Say, how do the FPR-degrees of the arithmetic $(\omega, +, \times, <, 0, 1)$ look like?

Our initial goal was to attack the following problem:

Describe the FPR degrees of $\mathbb{N} = (\omega, +, \times, <, 0, 1)$.

Our first result below seemingly killed the whole project.

Theorem 1.3. *Let* $\mathbb{N} = (\omega, +, \times, <, 0, 1)$ *be the ordered semiring of the standard arithmetic. Then* $FPR(\mathbb{N})$ *is neither an upper semi-lattice nor a lower semi-lattice.*

(In fact, the theorem above follows from the stronger Theorem 3.1 saying that that there is a pair of punctual copies of $\mathbb{N}$ having no infimum and no supremum.) Nonetheless, we decided to proceed and look at the FPR degrees of $\mathbb{N}$ from the perspective of lattice embeddings. More specifically, we asked for an algebraic description of finite lattices which are embeddable into $FPR(\mathbb{N})$ preserving sup and inf. This lead to the unexpected main result of the paper.

Theorem 1.4. *Let* L *be a countable lattice and* A *a finitely generated rigid punctual structure. Then the following are equivalent:*

(1) L *is embeddable into* $FPR(A)$*;*
(2) L *is distributive.*

We note that A could be *any* f.g. rigid structure; this clearly covers the case when A is $\mathbb{N}$. In fact, we could weaken the assumption on rigidity by, say, allowing all automorphisms in any copy to be primitive recursive; see also Remark 4.4 for a further discussion. For example, finitely generated *complete groups* have this property. (Recall that a group is complete if each automorphism of the group is inner, i.e., is a conjugation by some fixed element of the group. See [Obr94] for many examples of infinite f.g. complete groups[2].) We do not know if the theorem above holds without any assumption on the automorphism group.

The theorem is proved in two steps. The first and the simpler of the two steps is to show that each lower cone $\check{A} = \{X : X \leq_{pr} A\}$ has to be a distributive lattice if the structure is rigid and f.g.; this is Theorem 2.1. The second step requires more care. More specifically, in Theorem 4.1 we will use a subtle Lemma 4.2 to embed the atomless Boolean algebra $\beta = Int(\eta)$ below any degree $A >_{pr} \mathbf{0}$. These two results combined together clearly imply Theorem 1.4.

We know that each $\check{\mathbf{a}} = \{x : x \leq_{pr} a\}$ is distributive and embeds the atomless Boolean algebra β. It is natural to ask whether $\check{\mathbf{a}} = \{x : x \leq_{pr} a\}$ can be *isomorphic* to β or at least to *some* Boolean algebra. To finish off the paper, we give a definite negative answer to this question.

Theorem 1.5. *Suppose* G *is finitely generated, and assume* $A \in FPR(G)$*. Then* $\check{A} = \{X : X \leq_{pr} A\}/\cong_{pr}$ *is not a Boolean algebra.*

[2]Many thanks to Prof. Marston Conder for this reference.

Note that there is no rigidity assumption on A in the theorem above. The proof is contained in the last section of the paper. It combines the mentioned above Lemma 4.2 with a diagonalisation attempt.

We briefly discuss the techniques of our proofs. Although the injury is only finite, various effects specific to primitive recursive analysis make some of our arguments quite subtle. For example, we invite the reader to try to eliminate Lemma 4.2 from the proof of the main result Theorem 1.4.

Although we have leaned much about the punctual degrees of f.g. structures, there are many questions that remain open. For example, the only case when we have a full understanding of the punctual degrees of a f.g. structure is when it has exactly one degree. We do not know if there the rigidity assumption in Theorem 1.4 can be dropped, but we suspect that there exists a f.g. structure whose punctual degrees have a non-distributive lower cone. Also, we still do not know enough about punctual degrees of common algebraic structures. For example, are the punctual degrees of all f.g. infinite abelian groups isomorphic? What about the n-generated free groups? It seems that new techniques and insights will be required to answer questions of this sort.

2. Distributivity of lower cones

Theorem 2.1. *For every $A \in FPR(G)$ for a f.g. rigid G, the lower cone below A is a distributive lattice under $\leq_{pr}$.*

Proof. Then proof will be split into several lemmas.

Lemma 2.2. *For every $X, Y \leq_{pr} A$, $\sup(X, Y)$ exists.*

Proof. Consider the union Z of the images of X and Y within A. Note that Z itself does not have to be punctual, in the sense that a natural number can be kept outside of the domain of Z for quite long. However, Z contains a natural image of the least copy $\mathbf{0}$ of the structure; this natural image has a rapid enumeration. Using $\mathbf{0}$, it is easy to pass to a punctual copy of Z. This copy has a 1-1 punctual correspondence with Z, and it has domain ω. We identify Z with this copy. It remains to observe that, since G is rigid, any upper bound of X and Y punctually embeds Z as well. □

Lemma 2.3. *For every $X, Y \leq_{pr} A$, $\inf(X, Y)$ exists.*

Proof. Consider the intersection of the images of X and Y within A. Pass to a punctual copy of it (using a copy of $\mathbf{0}$). Using that G is rigid, observe that every lower bound of X and Y punctually embeds the natural punctual copy of the intersection. □

It remains to check distributivity.

Lemma 2.4. *Suppose $Z \leq \sup(X, Y)$. Then there exist $X_1 \leq_{pr} X$ and $Y_1 \leq Y$ such that $Z = \sup(X_1, Y_1)$.*

Proof. Consider $X_1 = Z \cap im(X)$ and $Y_1 = Z \cap im(Y)$, where $im(X)$ stands for the image of X in the supremum of Y and X under some fixed punctual reduction. (As before, use $\mathbf{0}$ to produce copies upon the domain of ω if necessary.) Clearly, $X_1 \leq_{pr} X$ and $Y_1 \leq_{pr} Y$. Also, by the proof of the lemma for sup, the definition of Z coincides with the definition of the supremum of X_1 and Y_1. □

Then lemma above excludes both M_3 and N_5, and therefore implies the theorem. □

3. No infimum

Theorem 3.1. *There is a pair of punctual presentations of* $\mathbb{N} = (\omega, +, \times, <, 0, 1)$ *wich that neither* $\sup(A_1, A_2)$ *nor* $\inf(A_1, A_2)$ *exists.*

Proof. We will construct a pair of punctual copies A_1 and A_2 of $\mathbb{N}$ without infimum. It follows from Theorem 2.1 that $sup(A_1, A_2)$ does not exist either. Thus, we will establish a bit more that is claimed in the theorem: there is a pair A_1 and A_2 of punctual copies of $\mathbb{N}$ such that neither $\sup(A_1, A_2)$ nor $\inf(A_1, A_2)$ exists.

3.0.1. *The requirements.* We are building A_1 and A_2 of $\mathbb{N}$ and an infinite sequence of $B_i, i = 1, 2, \ldots$, and satisfy the requirements:

P_i: $B_i \leq_{pr} A_1$ and $B_i \leq_{pr} A_2$,
R_i: $B_i <_{pr} B_{i+1}$,
S_e: if P_e is below both A_1 and A_2, then P_e is reducible to B_i, for some i.

(These will also imply that A_1 and A_2 are incomparable.)

3.1. **Intuition.** Let $\mathbf{0}$ denote the least punctual presentation of $\mathbb{N}$ under $\leq_{pr}$. It is easy to describe what $\mathbf{0}$ looks like, up to $=_{pr}$. More specifically, at stage s we have $\mathbf{0}[s] = \{\underline{k} : k \leq s\}$, where

$$\underline{k} = 1 + 1 + \ldots (k \text{ times}) \ldots + 1.$$

What does an $A > \mathbf{0}$ look like? In A, there will be infinitely many stages at which some elements x will be "far-far away" from the origin 0 for a long time. Such elements will look "non-standard" in the sense that $x \neq \underline{k}$ for any $\underline{k}$ generated so far, and they will stay "non-standard" for a long time. More formally, of there was a uniform bound on the stages at which every element is expressed as a term in 0 and 1, then we would have $A =_{pr} \mathbf{0}$. The collection of those elements which already received expressions in terms of 0 and 1 will be called *the standard part of A*. If we construct A, then we can keep x disconnected from the standard part for as long as we like, and in this case we say that we *keep x non-standard*. There will also use variations of this informal terminology throughout the proof; we believe that in each specific instance the meaning of these terms will be self-explanatory.

Now, when we have gained a bit of intuition on temporarily non-standard elements, we are ready to explain the basic diagonalisation strategy. Then we will finally explain the proof idea which, among other things, will be using the strategy below.

3.1.1. *The elementary diagonalisation strategy.* Suppose we are given a punctual $Y \cong \mathbb{N}$ and we would like to build an $X >_{pr} Y$. To diagonalise against $p_e : X \to Y$, do the following:

(1) Copy Y into X.
(2) In X, create a new x and keep it non-standard.
(3) Calculate $p_e(x) \in Y$.
(4) Wait for a stage and a k such that $p_e(x) = \underline{k}$ in Y.
(5) In X, define $x = \underline{m}$, where m is larger than k. (This stage may require a bounded delay on the enumeration of X. The delay may be necessary to compare $\underline{m}$ for all z currently present in X, and thus in Y.)

If Y is the least copy $\mathbf{0}$ of $\mathbb{N}$, then the delay in (5) will not be necessary: just pick m very large. However, in the general case, we must also make sure that x is not set equal to some other z which could currently be already present in Y (thus, in the part of X which copies Y), but its term has not yet been calculated. For that, evaluate $+$ and $\times$ for a sufficient number of steps to make sure that x can be safely incorporated into the standard part.

3.1.2. *An informal description of the construction.* Recall that we are building two copies, A_1 and A_2, of $\mathbb{N}$. At every stage, both A_1 and A_2 will always have *at most one* currently non-standard generator present in them. For example, A_1 will consist of a long enough initial segment of the natural numbers and also a generator x such that every element of A_1 at that stage will be seen as a term in 1 and x. These generators in A_1 and A_2 will be thought as equal, unless prescribed otherwise by some requirement.

We will have infinitely many kinds of such non-standard generators which will appear at some stages in both A_1 and A_2. When the time comes, we will once in a while create fresh non-standard-looking generators in both A_1 and A_2 (one in each) and call them blue. There will be infinitely many stages of the construction at which we create blue generators in both A_1 and A_2. They will be promised to be identical (as numbers) without any surprises. They will stay detached from the standard part for an equal number of steps (long enough), and at a late enough stage they will be incorporated identically into the standard part of both A_1 and A_2. Only after the blue generators are declared standard we will be allowed to put a fresh non-standard generator into A_1 or A_2 (or both) if we need to. (The colour of this fresh generator will not have to be blue.) Then, at some much later stage, we will again create blue generators, etc.

The structure B_1 will be consisting of the standard part and the blue generators (only). This property will make B_1 below both A_1 and A_2. The speed with which we connect the blue generators is dictated by the requirements making B_1 not reducible to the standard "bottom" copy $\mathbf{0}$ of $\mathbb{N}$, according to the simple strategy described at the subsection above. These will be the only strategies working with the blue generators.

Later, when we are ready, we will once in a while introduce a new kind of generators, call them red. (No blue generator can be red.) We also monitor P_0, which is the first punctual structure in the total listing $(P_i)_{i\in\omega}$ of all punctual structures. More specifically, we actually look at the triple (P_0, p_0, q_0), where p_0 and q_0 are potential isomorphisms from P_0 onto A_1 and A_2 respectively.

If no action need to be performed (to be explained), the red generators will be similar to the blue ones in the way they will be handled. They will be created and then, when necessary, connected to the standard part. The structure B_2 will incorporate the standard initial segment, the red and the blue generators, and *nothing else*. We will use the red generators to make $B_2 > B_1$ using the elementary strategy from the previous subsection.

If either p_0 or q_0 is not an isomorphism then it will be eventually discovered. If no point of P_0 is ever mapped into a red (in the general construction, non-blue) generator via q_0 and p_0, then P_0 must be reducible to B_1. Since B_2 is above B_1, and both are below A_1 and A_2, P_0 is not the infimum of A_1 and A_2.

Thus, we assume p_0 does eventually map some $x \in P_0$ into a red (in general, non-blue) generator of A_1. But if it does send x into a red generator of A_1, we

can make $p(x)$ not equal to $q(x)$ when we finally decide on the terms for the red generators.

Note that if both p_0 and q_0 map x into red generators of A_1 and A_2, respectively, then we will have to connect the red generators differently in these two structures. Thus, we will potentially upset the definition of B_2 which assumes that the red generators behave identically in both A_1 and A_2. If this ever happens, we will have to restart B_2. We make a new copy of B_2, and proceed to the next requirements. Use the speed with which the red generators are connected to make B_1 strictly below B_2, and we will never use the red generators again for any other purpose.

The case of all B_i is handled using simple priority; each colour can be upset by only finitely many P_e, thus making the definition of the respective B_j eventually correct.

3.2. **Formal proof.** We describe the basic strategies, and then we describe the construction. In the construction, we will be introducing generators into A_1 and A_2 and assign colours to the generators. The generators will be kept non-standard and then connected to the standard part, according to the strategies below. When a generator becomes standard, it no longer has a colour.

3.2.1. *The strategy for P_i.* The strategy is responsible for building B_i a pair of isomorphisms $v_i : B_i \to A_1$ and $u_i : B_i \to A_2$. At every stage, the isomorphisms will be defined as identical pullback-copying of A_1 (resp., A_2) into B_i on all currently standard elements (which will be the same in A_1 and A_2) and all generators of *colours* j, $j \leq i$. Unless injured by higher priority requirements (to be clarified), the values of the j-coloured generators will be set equal in both A_1 and A_2. Therefore, assuming the strategy is never initialised, the definition of B_1 and the primitive recursive isomorphic reductions u_i and v_i will be consistent and correct.

3.2.2. *The strategy for R_i.* The strategy is responsible for illustrating $B_i <_{pr} B_{i+1}$. The basic strategies for P_i and P_{i+1} will guarantee $B_i \leq_{pr} B_{i+1}$. To ensure $B_{i+1} \not\leq_{pr} B_i$, implement the basic diagonalisation strategy (§3.1.1) on the generators having colour $i+1$ to defeat every potential isomorphism $p_e : B_{i+1} \to B_i$ throughout the construction. (We note that the coloured generators are formally introduced in A_1 and A_2, but they are naturally copied into B_i and B_{i+1}, and thus the respective generators can be identified.) In particular, the strategy will define the stages at which the generators having colour $j+1$ will be assigned with a term, i.e., will become "standard".

3.2.3. *The strategy for S_e.* The strategy monitors P_e. It will be split into infinitely many substrategies $S_{e,i,j}$, each working with the respective triple (P_e, p_i, q_j). Here p_i and q_j are potential reductions onto A_1 and A_2, respectively. The instructions for the substrategy $S_{e,i,j}$ are as follows:

(1) Wait for an $x \in P_e$ such that either $p_i(x)$ or $q_j(x)$ are equal to a linear combination which non-trivially involves a generator of colour $c > \langle e, i, j \rangle$. If such an x is never found, then do nothing. In this case P_e is below B_c.
(2) Suppose $p_i(x) = \underline{m}_0 + \underline{m}_1 z + \ldots + \underline{m}_s z^s$, where z is currently kept nonstandard and has colour $c > \langle e, i, j \rangle$. (The case of $q_j(x)$ is symmetric.)

(a) If $q_j(x)$ is unequal to $p_i(x)$ as a term, then the strategy for R_c will guarantee that value for $q_j(x)$ is not equal to value for $p_i(x)$ in the respective copies of the arithmetic. Thus, in this case we again do nothing.
(b) If both $p_i(x)$ (in A_1) and $q_j(x)$ (in A_2) are equal as terms, then we must ensure these linear combinations will become unequal (as numbers in the respective copies of the arithmetic) when we are ready to agree on the values for z and z'. Note that, initially, the strategy for P_c intended to assign identical terms to both z and z', and therefore P_c will be *injured*.

3.2.4. *Construction.* In the construction, we build A_1 and A_2 punctually so that for every colour k there are infinitely many stages at which a new currently non-standard generator is introduced to both A_1 and A_2.

We then let the P_i-strategies build the respective B_i, and we let the other strategies act according to their instructions. At stage s, we monitor P_i with $i < s$, and we let $S_{e,i,j}$ with $\langle e, i, j\rangle < s$ to act if necessary, with the priority given to the strategy having the smallest index. If P_i is *injured* due to an action of some $S_{e,i,j}$-strategy (and there can be only finitely many such), then it restarts its definition of embeddings u_e and v_e onto A_1 and A_2, respectively, correcting the asymmetry in A_1 and A_2 introduced by the $S_{e,i,j}$-strategy.

3.2.5. *Verification.* It is clear that the structures $A_1, A_2, B_i, i = 0, 1, \dots$ are all punctual, and each of them is isomorphic to $\mathbb{N}$.

Note that there are at most finitely many S-substrategies which can possibly upset the definition of the embeddings of B_i onto A_1 and A_2. Furthermore, every such substrategy acts at most once. Thus, every i, P_i is eventually met. It follows at once from the description of the basic diagonalisation strategy (§3.1.1) that R_i is met for every i.

Also, the sub-strategy $S_{e,i,j}$ will either ensure that either p_i and p_j are not isomorphisms form P_e onto A_1 and A_2, respectively, or P_e is reducible to B_c. We claim that S_e is met for every e. Indeed, suppose $P_e \leq_{pr} A_1$ and $P_e \leq_{pr} A_2$ as witnessed by, say, p_i and p_j. Then P_e must be pr-below B_c. However, $B_{c+1} >_{pr} B_c$, and therefore $B_{c+1} >_{pr} B_c \geq_{pr} P_e$. Since B_{c+1} is below both A_1 and A_2, P_e cannot be their infimum. □

4. Embedding $Int(\eta)$

Theorem 4.1. *Suppose X is a finitely generated rigid punctual structure, and suppose $A >_{pr} \mathbf{0}$ is any punctual presentation of it which is not the pr-least presentation. Then the lower cone $\check{A} = \{Z : Z \leq_{pr} A\}$ embeds the atomless Boolean algebra $Int(\eta)$ preserving the supremums and the infimums.*

Proof. The lemma below seems to be crucial for the proof. Before we state the lemma, we need a definition. Every copy punctually embeds a copy of $\mathbf{0}$ which is the term algebra on some fixed tuple of generators. At a stage, there could be elements of A which are not yet been seen as terms of the fixed finitely many generators; we call these elements *currently non-standard*. We will also abuse our terminology and

will call them non-standard for short, but we always keep in mind that this are not the "actual" non-standard elements, and that the notion is dynamic and depends on a stage. At every stage, there will also be a part which consists of terms in the fixed generators of the structure; we call this part *currently standard*, or simply *standard*. We also say that a tuple $\bar{x}$ s-generates an element y if at stage s we see a term t such that $t(\bar{x}) = y$.

Lemma 4.2. *Suppose $A > \mathbf{0}$ is finitely generated (there is no assumption on rigidity). There is a $A^* > \mathbf{0}$ below A with the property: At every stage s we can pick at most one currently non-standard generator in $A[s]$ which, together with the currently standard part of $A[s]$, s-generates $A[s]$.*

Proof. Using a reduction from $\mathbf{0}$ onto A, fix a collection of generators $\bar{c}$ in A which is the isomorphic image of some fixed tuple $\bar{d}$ of generators in $\mathbf{0}$, under the fixed reduction.

We will copy the standard part $\langle \bar{d} \rangle[s]$ into A^* as well as some other elements (to be explained). Let $\bar{d}'$ be the natural image of $\bar{d}$ in A^*. We will view each p_e as a map from A^* to $\mathbf{0}$. For every e, we will wait for $p_e(\bar{d}')$ to generate $\bar{c}$. This non-primitive recursive waiting will be performed at the background and will not be a part of the strategy below. If we ever see that the elements $p_e(\bar{c})$ generate $\bar{d}$, we declare p_e *ready*. At every stage there will be some finite – perhaps, empty – collection of such p_e which have already been declared ready.

We have not yet defined A^*; the construction of A^* will also be performed based on the strategy below. As we have mentioned above, A^* will always be copying the standard part copied from A. The further elements of A^* are picked as follows.

(1) Wait for at least one p_e to be declared ready, and pick e least such.
(2) Choose x with the smallest index outside the standard part $\langle \bar{d} \rangle[s]$ of A, and copy x into A^*.
(3) Keep copying only elements from $\langle \bar{d} \cup x \rangle$ into A^* and monitor $p_e(x)$.
(4) If p_e is discovered to be not an isomorphism, then wait for x to be incorporated into $\langle d' \rangle$,and then go (1) and wait for the next primitive recursive map to be declared ready.
(5) Otherwise, when $p_e(x) \downarrow$, we will see $p_e(x) \in \langle \bar{c} \rangle$ in $\mathbf{0}$. Since p_e had been declared ready, we will also see $p_e(x) \in \langle p_e(d') \rangle$ on that stage. Since p_e insists on being an isomorphism, we will also see $x \in \langle d' \rangle$ in A^*.
(6) Define $q : x \to \mathbf{0}$ on x as follows. Map $\bar{d}$ to the natural pre-image $\bar{u}$ of $\bar{d}'$ in $\mathbf{0}$, map x to the $\bar{u}$-term obtained from the $\bar{d}'$-term witnessing $x \in \langle d' \rangle$ using the straightforward replacement. Since each element $\langle \bar{d}' \cup x \rangle$ is also a term in $\bar{d}'$ and x, extend this map naturally to every such element seen so far. Go to (2).

Note that at every stage s, A^* contains an element x which, together with the currently standard part of A^*, generates the whole of $A^*[s]$. Also, suppose $A^* \leq \mathbf{0}$ as witnessed by p_e, and e is the smallest such. Then p_e will eventually be declared ready, and each p_j, $j < e$, will either show that they are not isomorphisms in finite time or will never be declared ready. (We note here that the outcome when p_j is not onto is simply impossible. If p_j has been declared ready then at has no choice but to copy the structure along the generators or show a local disagreement.)

At some stage, p_e will have the smallest index among the maps which are ready, and thus it will be used by the strategy in the definition of x. If p_e never shows a

local disagreement, then this means that every x that we pick will be incorporated into the standard part of A^*, and this leads to a primitive recursive isomorphism $q : A \to \mathbf{0}$ whose convergence speed can be calculated using p_e. This contradicts the choice of $A >_{pr} \mathbf{0}$. □

Lemma 4.3. *Let $A^* > \mathbf{0}$ satisfy the property from Lemma 4.2, and additionally assume that the finitely generated structure under consideration is rigid. Then there exist incomparable P_0, P_1 such that $\sup(P_0, P_1) = A^*$ and $\inf(P_0, P_1) = \mathbf{0}$.*

Proof. The proof is non-technical, thus there is no point in giving a formal stage-by-stage construction. The difficulties which occur in he proof are related purely to the timing at which the (elementary) actions below will be performed. We give a detailed explanation of what needs to be done, and why the assumptions on A^* make the proof work.

To build P_0, switch back and forth between copying $\mathbf{0}$ and copying A^*. The elementary formal details of the switching procedure can be found in [BKMN]; we will also give more details below. The other structure P_1 will also be switching, but in the opposite way. That is, P_1 will be copying $\mathbf{0}$ when P_0 copies A^*, and vice versa. When P_0 copies A^* and P_1 copies $\mathbf{0}$, we diagonalise against $p_e : P_0 \to P_1$, which will be achieved in finite time. Indeed, if p_e was an isomorphism, we would be able to define a primitive recursive isomorphism of A^* onto $\mathbf{0}$, contradicting the choice of A^*. As in the proof of Lemma 4.2, we do not even attend p_e unless it has mapped the generators to the generators (recall the structure is rigid). Since the generators will be mapped to their natural images, p_e will have no choice but to either copy the structure locally or to show a local disagreement, which will happen in finite time. When we see p_e die, we declare that we are ready to *switch sides*.

When we switch sides, we no longer allow P_0 to copy freshly introduced non-standard elements in A^*, unless they are generated by the unique special non-standard generator of A^* over the standard part. We wait for (the natural image of) $\mathbf{0}$ (in A^*) to ketch up on the unique non-standard generator which may currently be in the structure. Once we see this happen, we declare that P_0 now copies $\mathbf{0}$, and we also declare that P_1 copies A^*. At that stage we instantly switch sides and allow P_1 to copy A^*. *It is crucial that as soon as the least non-standard generator is incorporated into the standard part, everything in P_0 immediately becomes standard.* While we wait for the switch to be performed, there will be no further nonstandard elements in A^* which would not be generated by a specifically chosen special non-standard generator. Thus, while we are waiting, P_0 is still identical to A^*, and P_1 can be kept identical to $\mathbf{0}$. Then we meet another diagonalisation requirement and then switch back similarly, etc.

We claim that $\sup(P_0, P_1) = A^*$. If $H \geq P_0, P_1$, then we define $h : A^* \to H$ as follows. At stage s, see which of the two structures copies A^*. Assume this is P_0. To define $h(x)$, see where x is mapped into P_0, and use the reduction from P_0 to H to match x with the respective image in H. Since the structure is rigid, the composition of the maps coming through the P_0-side will agree with the composition of the maps coming from the P_1-side, thus there will be no problem in extending the isomorphism even after we switch sides.

Remark 4.4. We (strongly) conjecture that rigidity can be replaced by some more relaxed assumption, such as: Every automorphism in a punctual copy of the structure is primitive recursive. We do not know if the lemma holds without any

assumptions on the automorphisms of the structure, and we suspect that in fact it does not.

We claim that $\inf(P_0, P_1) = \mathbf{0}$. Again, if there is some $B \leq_{pr} P_0, P_1$, then we use either the P_0-side or the P_1-side to match every element of B with the respective element in $\mathbf{0}$.

Remark 4.5. It is crucial that, when we are ready to switch sides, there is at most one non-standard generator in T. Otherwise we could run into the scenario when we have connected some non-standard parts in P_0, but some other non-standard parts stay disconnected from the generators for much longer. Then there is a possibility of constructing another infimum (or supremum) which is tighter than the one claimed in the lemma.

□

Note that both P_0 and P_1 satisfy the property of Lemma 4.2. Iterate the process to embed $Int(\eta)$, as follows. Split P_0 and P_1 over $\mathbf{0}$, and then split each of the two, etc. The process can in fact be made simultaneous and effective, but this is not important for establishing the theorem. □

5. Not a Boolean algebra. Proof of Theorem 1.5

Fix $A >_{pr} \mathbf{0}$ of a finitely generated structure. We must show that the lower cone below A is not a Boolean algebra. (If $A =_{pr} \mathbf{0}$ then there is nothing to prove, because every Boolean algebra contains at least two elements.) The plan is to build a copy $A^* \leq_{pr} A$ and construct a $C <_{pr} A^*$ such that C does not have the relative complement below A^*. Use Lemma 4.2 to produce an $A^* > \mathbf{0}$ below A which has at most one special non-standard generator at every stage. (Recall that Lemma 4.2 did not need any assumption on $Aut(A)$.)

We will build a C such that $\mathbf{0} <_{pr} C <_{pr} A^*$ and satisfy the requirements:

$$R_e : P_e \text{ is not the complement of } C \text{ relative to } A^*,$$

where $(P_e)_{e\in\omega}$ is the total enumeration of all punctual structures.

The strategy for making C strictly in-between $\mathbf{0}$ and A^* is exactly the same as in the previous theorem. That is, we will attempt to switch back and forth between copying $\mathbf{0}$ and copying A^*. When C copies $\mathbf{0}$, we diagonalise against $p_e : C \to A^*$, where p_e has already been declared ready (i.e., the generators and their images are bi-spanned), and e is the least such. When C copies A^*, we diagonalise against isomorphisms from $\mathbf{0}$ to C similarly. In any of these cases the diagonalisation process, once initiated, will finish in finite time. Note that the nice property of A^* allows to switch without any extra delay; see Remark 4.5.

The strategy for R_e is less straightforward. We split it into sub-strategies, each monitoring a triple (P_e, q_e, l_e) where $q_e : P_e \to A^*$ and $l_e : \mathbf{0} \to P_e$ are potential isomorphisms. The sub-strategy computes $q_e(l_e(\bar{c}))$, where $\bar{c}$ is some fixed tuple of generators in $\mathbf{0}$, and waits for the image of $\bar{c}$ in A^* and $q_e(l_e(\bar{c}))$ to generate each other. If this ever happens, then the substrategy will be declared ready.

5.0.1. *The definition of $N(C, P_e)$.* Let $p : C \to A^*$ be the primitive recursive isomorphism which will be constructed by stages. At every stage its value will be well-defined. Let $N(C, P_e)$ be the punctual re-enumeration of $\langle p(C) \cup q_e(P_e)\rangle \subseteq A^*$. The re-enumeration uses that the fixed tuple of generators of A^* will be contained

in $p(C) \cup q_e(P_e)$, and therefore we can punctually list the "standard" substructure of A^* which will also be contained in $\langle p(C) \cup q_e(P_e)\rangle$; the other elements can be incorporated with a bounded delay. Although $N(C, P_e)$ is not really a subset of A^*, but rather its retract via a punctual bijection, we abuse our notation and at every stage identify $N(C, P_e)$ with a subset of A^*. Note that, since q_e was declared ready, the standard part of $q_e(P_e)$ can be essentially identified with the standard part of A^* and of $p(C)$.

If q_e and l_e are indeed isomorphisms, we will have that $N(C, P_e)$ is a punctual copy of A^* with the property $N(C, P_e) \leq_{pr} A^*$; the reduction is essentially the identity. (Strictly speaking, the reduction will be the bijection defined for the sake of making the domain of $N(C, P_e)$ equal to ω). Also, under the same assumptions we will have $C \leq_{pr} N(C, P_e)$ and $P_e \leq N(C, P_e)$.

In other words, we have constructed *some* upper bound $N(C, P_e)$ of P_e and C, but only under the assumption that all the maps under consideration are indeed isomorphisms.

Remark 5.1. Since we have assumed that the generators have been seen to span each other, we do not have to worry about the unpleasant case of a proper isomorphic embedding. In other words, if something goes wrong with the potential isomorphisms, we will see a confirmation at a finite stage.

5.0.2. *Pressing P_e.* If A^* is indeed the supremum of C and P_e, then so is $N(C, P_e) \leq_{pr} A^*$. In particular, there should exist some for some isomorphism $v_e : A^* \to N(C, P_e)$. For every e, we shall test whether $v_e : A^* \to N(C, P_e)$ is an isomorphism. We further split the sub requirements, each monitoring its own primitive recursive v_e. As usual, we wait for $v_e(\bar{c})$ and $\bar{c}$ to generate each other. Again, if this never happens then we never attend the sub-requirement monitoring v_e.

The value of v_e is that it will allow us to press P_e to show us non-standard elements when they appear in A^*, by intentionally keeping them out of C.

5.0.3. *The definition of $L(C, P_e)$.* The subrequirement working with (P_e, q_e, l_e, v_e) will attempt to define some lower bound $L(C, P_e)$ below C and P_e such that $L(C, P_e) >_{pr} \mathbf{0}$. The definition of $L(C, P_e)$ will be dynamic, but it will be similar to $N(C, P_e)$, in the sense that it will be "living" inside A^*, but up to a punctual bijection whose only role is to make the domain equal to ω.

We split the subrequirement even further, for the last time. Each individual one will be trying to diagonalise against $u_e : L(C, P_e) \to \mathbf{0}$, and again will act only if it is declared ready.

When the subrequirement working with $(P_e, q_e, l_e, v_e, u_e)$ is ready to act, it goes over the following stages. At every stage it waits for a disagreement showing that one of the maps is not an isomorphism. If it discovers such a disagreement, it instantly stops. Recall that at every stage A^* has at most one special nonstandard generator at every stage.

(1) Let C copy $\mathbf{0}$.
(2) Use $v_e : A^* \to N(C, P_e)$ to compute the image of a special generator in P_e, when it is introduced in A^*.
(3) Copy the special generator into C and define a (perhaps, not optimal) lower bound $L(C, P_e)$ which contains the standard part as well as the special generator, and matches these with their natural images in P_e and C in accordance with p, q_e, v_e. More specifically, we copy A^* into $L(C, P_e)$

naturally, using p. We define the image of the special generator (thus, of the rest) in $N(C, P_e)$ using v_e, and then we use the definition of $N(C, P_e)$ to punctually find the respective pre-images under q_e. (Later in the construction, while no sub strategy of the strategy for $(P_e, q_e, l_e, v_e,)$ is active, we will copy C into $L(C, P_e)$ and will not wait for any diagonalisation to occur.)

(4) Wait for either $u_e : L(C, P_e) \to \mathbf{0}$ to die or $v_e : A^* \to N(C, P_e)$ to die. Repeat 1-4 until ether we diagonalise u_e or one of the monitored maps proves to be not an isomorphism.

The strategy is finitary in the following sense. Either at least one of the maps observed by the substrategy must prove itself wrong, or we will eventually diagonalise against u_e. It is crucial that the substrategy acts only when all the maps observed by it have proved themselves ready (in the sense of bi-spanning of the generators and their images under the map). If all the maps except for u_e are never wrong, then we will eventually diagonalise against u_e, for otherwise we would be able to use u_e to build a reduction from A^* onto $\mathbf{0}$.

5.0.4. *Construction.* We list all subrequirements $(P_e, q_e, l_e, v_e, u_e)$ in some priority order and wait for them to be declared ready. Meanwhile, we construct C switching back and forth between copying $\mathbf{0}$ and A^*, according to its basic strategy. As soon as a fresh diagonalisation subrequirement is declared ready, we pick the highest priority such. We then allow C to switch to copying $\mathbf{0}$ and act according to the instructions of the substrategy. After the action of the substrategy is finished, we allow C, to switch at least twice back and forth between $\mathbf{0}$ and A^*. Then repeat.

5.0.5. *Verification.* A large portion of the verification was incorporated into the description of the basic strategies. We put everything together.

The construction is finitary, and there is no explicit injury or initialisation involved. It is crucial that, as soon as a map is declared ready, we no longer have to worry about the Π^0_2-outcome in which the map is a proper isomorphic embedding. In the construction, we simply never get to work with any map which is never declared ready.

Once a map is declared ready, all the diagrams will be maintained commutatively until we see some disagreement. Either a disagreement must occur, or we succeed in diagonalising against u_e. For either A^* is not the real supremum, and then q_e or v_e must prove to be wrong, or we will punctually copy A^* into the lower bound $L(C, P_e)$ and rely on the fact that $A^* > \mathbf{0}$. Indeed, if we keep successfully copying A^* into $L(C, P_e)$, then $u_e : L(C, P_e) \to \mathbf{0}$ cannot be an isomorphism.

Assume q_e, l_e, v_e are all actual isomorphisms and A^* is a supremum. Then each u_e will either never be declared ready, or will eventually become ready. In the former case it cannot be an onto isomorphism $u_e : L(C, P_e) \to \mathbf{0}$, and in the latter case we will eventually diagonalise against it, and furthermore will see a finitary disagreement witnessing that $u_e : L(C, P_e) \to \mathbf{0}$ is not an isomorphism. Either way, we will succeed in constructing $L(C, P_e)$ strictly above $\mathbf{0}$.

References

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

[Ala17] P. E. Alaev. Structures computable in polynomial time. I. *Algebra Logic*, 55(6):421–435, 2017.

[Ala18] P. E. Alaev. Structures computable in polynomial time. II. *Algebra Logic*, 56(6):429–442, 2018.

[BDKM] N. Bazhenov, R. Downey, I. Kalimullin, and A. Melnikov. Foundations of online structure theory. Submitted.

[BG00] Achim Blumensath and Erich Grädel. Automatic structures. In *15th Annual IEEE Symposium on Logic in Computer Science (Santa Barbara, CA, 2000)*, pages 51–62. IEEE Comput. Soc. Press, Los Alamitos, CA, 2000.

[BKMN] N. Bazhenov, I. Kalimullin, A. Melnikov, and K. M. Ng. Punctual presentations of finitely generated structures. Submitted.

[CR98] D. Cenzer and J. B. Remmel. Complexity theoretic model theory and algebra. In Yu. L. Ershov, S. S. Goncharov, A. Nerode, and J. B. Remmel, editors, *Handbook of recursive mathematics, Vol. 1*, volume 138 of *Stud. Logic Found. Math.*, pages 381–513. North-Holland, Amsterdam, 1998.

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

[FKH$^+$12] A. Frolov, I. Kalimullin, V. Harizanov, O. Kudinov, and R. Miller. Spectra of high$_n$ and non-low$_n$ degrees. *J. Logic Comput.*, 22(4):755–777, 2012.

[FKM10] Ekaterina B. Fokina, Iskander Sh. Kalimullin, and Russell Miller. Degrees of categoricity of computable structures. *Arch. Math. Log.*, 49(1):51–67, 2010.

[Gri90] Serge Grigorieff. Every recursive linear ordering has a copy in $DTIME\text{-}SPACE(n, log(n))$. *J. Symb. Log.*, 55(1):260–276, 1990.

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

[Kie81] H. A. Kierstead. An effective version of Dilworth's theorem. *Trans. Am. Math. Soc.*, 268:63–77, 1981.

[Kie98] H. A. Kierstead. On line coloring k-colorable graphs. *Israel J. Math.*, 105(1):93–104, 1998.

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

[KMN17] Iskander Kalimullin, Alexander Melnikov, and Keng Meng Ng. Algebraic structures computable without delay. *Theoret. Comput. Sci.*, 674:73–98, 2017.

[KN95] Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. In *Logic and Computational Complexity (Indianapolis, IN, 1994)*, volume 960 of *Lecture Notes in Comput. Sci.*, pages 367–392. Springer, Berlin, 1995.

[KPT94] H. A. Kierstead, S. G. Penrice, and W. T. Trotter Jr. On-line coloring and recursive graph theory. *SIAM J. Discrete Math.*, 7:72–89, 1994.

[LS01] Roger C. Lyndon and Paul E. Schupp. *Combinatorial group theory*. Classics in Mathematics. Springer-Verlag, Berlin, 2001. Reprint of the 1977 edition.

[LST89] L. Lovász, M. Saks, and W. T. Trotter Jr. An on-line graph coloring algorithm with sublinear performance ratio. *Discrete Math.*, 75:319–325, 1989.

[Mil01] R. Miller. The Δ^0_2-spectrum of a linear order. *J. Symbolic Logic*, 66(2):470–486, 2001.

[MN] A. G. Melnikov and K. M. Ng. The back-and-forth method and computability without delay. Preprint.

[Obr94] Viatcheslav N. Obraztsov. On infinite complete groups. *Comm. Algebra*, 22(14):5875–5887, 1994.

[Rem86] J. B. Remmel. Graph colorings and recursively bounded Π^0_1-classes. *Ann. Pure Appl. Logic*, 32:185–194, 1986.

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

Kazan Federal University
Email address: ikalimul@gmail.com

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

Kazan Federal University
Email address: maxim.zubkov@kpfu.ru