FOUNDATIONS OF ONLINE STRUCTURE THEORY

NIKOLAY BAZHENOV, ROD DOWNEY, ISKANDER KALIMULLIN,
AND ALEXANDER MELNIKOV

Contents

1. Introduction

1.1. **Our goal.** Imagine your job is to receive – perhaps infinitely many – objects of various sizes and pack them into bins, of a fixed size. You receive object n and at time $n+1$ you have to say which bin you should pack it into, and this decision is irrevocable. You are in an *online* situation.

The goal of this paper is to outline the ideas of a new programme in computable mathematics devoted to *online computable structure theory*. This will be a new

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

subfield of computable structure theory, but one with its own highly distinctive character.

First, we will discuss online algorithms for combinatorial structures. Online combinatorics lacks general theory. We will also remind the reader of the related ideas of (Turing) computable mathematics. This subject does have a developed systematic theory; however, it uses the abstract notion of Turing computation which is very far from being online. Then we will introduce our main paradigm which will involve *primitive recursive* structures, and explain why this is the chosen paradigm. In the remaining sections we will explore results already obtained using this paradigm, and articulate some of the many open problems in the area.

1.2. **Turing computable mathematics.** The general area of *computable* or *effective mathematics* is devoted to understanding the algorithmic content of mathematics. The roots of the subject go back to the introduction of non-computable methods into mathematics at the beginning of the 20th century as discussed in Metakides and Nerode [MN82]. Early work concentrated on developing algorithmic mathematics in algebra, e.g. Grete Hermann [Her26], analysis such as Bishop's constructive analysis, (implicitly) using algorithmic methods to understand randomness (Borel [Bor09], von Mises [vM19], Church [Chu40]), understanding effective procedures in finitely presented groups such as Dehn [Deh11], and most notably Hilbert's programme seeking to give a decision procedure for first order logic. We know all of these historical roots led to the development of, for example, computability theory, complexity theory, and algorithmic randomness (see e.g. Downey [Dow17]). The modern version of effective mathematics utilizes the tools developed in these areas, as well as classical tools in algebra, analysis, etc. to calibrate the algorithmic content of many areas of mathematics.

The standard model for such investigations is a (Turing) computable presentation of a structure. By this we mean a coding of the structure with universe $\mathbb{N}$, and the relations and functions coded Turing computably. For example, a computable presentation of a group would be either a finite group, or one where the universe was considered as $\mathbb{N}$ and the group operation was represented as a (Turing) computable function. Note that this framework uses the general notion of a Turing computable function. In particular, we put no resource bound on our computation.

1.3. **Online combinatorics.** A hallmark of the majority of algorithms on finite structures is that the algorithm "knows all about the structure". In other words, the whole structure is given to the algorithm at once. For example, when a complexity theorist talks about the Hamiltonian path problem, they have in mind algorithms that given a description of a finite graph (say, a matrix-presentation of it) outputs such a path. This is sometimes not true for large data sets, and several LOGSPACE algorithms, but we are using this to refer to those students which would learn in a basic algorithm course. What happens to such algorithms if the graph is not given at once, but rather is given to us step-by-step and vertex-by-vertex? This situation is an abstraction to an "online" computation in which the input data is too massive to be given as an input at once. Now of course there are *many* problems in computer science where we can safely assume that universe is infinite and thus we need an online algorithm. For example, a scheduler which assigns users to access shared memory is a classic example.

In the "online" setup the situation becomes quite a bit harder. Consider the following example. Every tree is 2-colourable, but to achieve this colouring you need to know the whole of the tree. Suppose we are given a vast tree one vertex at a time, so that $G = \cup_s G_s$, an *online* presentation of G. When we give you the vertex v we promise to tell you all of the vertices given so far to which v is joined; that is the induced subtree of $v_1, \ldots, v_s$. Your goal is to colour the vertex v_s, before we give you v_{s+1}. We are in an *online* situation. For a tree with n vertices, the sharp lower bound is $O(\log n)$ many colours. It follows that there are online presentations of infinite (computable) trees which cannot be online coloured with any finite number of colours. We see that switching to the online case effects not only the running time, but also the best solution that we can hope for. We remark that online algorithms can be quite complex, and for similar online problems scheduling, the decidability of S2S, and Büchi automata are intimately related.

Beginning in the 1980's there has been quite a lot of work on online infinite combinatorics, particularly by Kierstead, Trotter, Remmel and others ([Kie81, Kie98a, KPT94, LST89, Rem86]). Some results were quite surprising. For example, Dilworth's theorem says that a partial ordering of width k can be decomposed into k chains. Szemeredi and others showed that there is a computable partial ordering that cannot be decomposed into k computable chains. But in 1981, Kierstead proved that there is an online algorithm that will decompose any online presentation of a computable partial ordering into $\frac{5^k-1}{4}$ many (computable) chains. Only in 2009 was this result improved by Bosek and Krawczyk who demonstrated that it can be done with $k^{14 \log k}$ many chains. Work here is ongoing. (See [Kie98b] for a somewhat dated survey.) In the case of finite structures most work comes from comparing offline vs online performance. In this area, the typical setting is to build some kind of function which is measured relative to some size, and the goal of online algorithm design is to improve what is called the *Competitive Performance Ratio* of online divided by offline. For example, first fit gives a competitive ratio of 2 for the classical Bin Packing problem (see Garey and Johnson [GJ79]).

1.4. **Online vs. Turing computable.** The notion of an "online" algorithm in the results mentioned above is rather specific. One may complain that, rather than saying that we must make a decision before the next vertex shows up, it is fine to wait for a bit more of a graph to be shown to us. But *how much more* exactly? Maybe we can wait for 17 more vertices to show up before we make a decision. Perhaps, at stage s we could ask for $\log(s)$ more vertices, etc. It is not hard to see that various answers to this question will lead to a proper hierarchy – rather, a zoo – of "online" computability notions. It is natural to ask:

What is the most general notion of an online algorithm?

Understanding the *online content* of mathematics so far has no general theory, there are only algorithms or proofs that no algorithm exists. Note that the lack of theory for online mathematics stands in stark contrast with the infinite off-line case described by the computable structure theory [AK00, EG00]. However, as we noted above, computable structure theory relies on the most general notion of a computable process that we know today – a Turing computable process. Turing computability provides us with many tools, such as the universal Turing machine and the Recursion Theorem, that are useful in *proving theorems about algorithms.* However, Turing computability in its full generality is not an adequate model in

the online situation, because Turing computable algorithms can use an unbounded search. For instance, recall the example in which we had to online colour a tree. A Turing computable algorithm would just *wait* until a node gets connected to the root of the tree via a path and then will make a decision. There is no *a priori* bound on how long it may take for the path to be revealed, but a Turing computable algorithm does not care. More importantly, Turing computability fails to capture the "impatient" nature of an online algorithm which has to make a decision "now".

1.5. **Our goal, revisited.** Recall that our goal is to give a general abstract foundation for online algorithms. As we will soon see, our approach is based on one natural interpretation of "online" involving *primitive recursive* structures.

We look at infinite algebraic and combinatorial structures such as graphs, linear orderings, groups, unary structures, etc. We will attempt to apply the techniques and intuition coming from the mentioned above (Turing) computable structure theory [AK00, EG00] to our "online" framework. We will succeed in applying some non-trivial techniques and methodology coming from Turing computable structures. However, the intuition in our online case is so different from the Turing computable case that hardly any non-trivial result of computable structure theory can be transformed into one about online algebraic structures. Instead, we will discuss some results that have no analogy in Turing computable structures. Furthermore, many of our results can be seen to hold for polynomial-time structures, and there will be one theorem that settles a conjecture about automatic structures. We also note that many of the results, proofs, and proof sketches that appear in this survey are new.

1.6. **The models.** We will concentrate on infinite structures. Still to do is to develop an appropriate model theory for online finite structures as asked for by Downey and McCartin [DM04]. In its most general formulation, an online algorithm would act on a structure $\mathcal{A}$ given in stages $f(1), f(2), \ldots$, where f is a computable function representing timestamps. At stage $f(n)$ we would enumerate n into the partial structure $A_{f(n)}$ and give complete information about how n relates to $\{0, \ldots, n-1\}$.

Now the question is: *What kinds of structures and time functions should be allowed?* Different choices will result in different theories. Our goal is to give a general setting that also reflects the common online structures encountered. We examine some approaches from the literature:

1.6.1. *Automatic structures.* Khoussainov and Nerode [KN95] initiated a systematic study into automatically presentable algebraic structures; but these seem quite rare. For example, the additive group of the rationals is not automatic [Tsa11]. The approach via finite automata is highly sensitive to how we define what we mean by automatic. For example treating a function as a relation yields quite a different kind of automatic presentation and treating it is as a transducer. See [ECH+92, KKM14] for an alternate approach to automatic groups. Although the theory of automatic structures is a beautiful subject, a finite automaton is definitely not a general enough model for an online algorithm.

1.6.2. *Polynomial time computable structures.* Cenzer and Remmel, Grigorieff, Alaev, and others [CR98, Gri90, Ala17, Ala18] studied polynomial time presentable structures. We omit the formal definitions, but we note that they are sensitive to

how exactly we code the domain. In many common algebraic classes we can show that all Turing computable structures have polynomial-time computable copies. One attractive result is that every computably presentable linear ordering has a copy in linear time and logarithmic space [Gri90]. Similar results hold for broad subclasses of Boolean algebras [CR91], some commutative groups [CR92, CDRU09], and some other structures [CR91].

1.6.3. *Fully primitive recursive structures.* As was noted in [KMN17b], many known proofs from polynomial time structure theory (e.g., [CR91, CR92, CDRU09, Gri90]) are focused on making the operations and relations on the structure *primitive recursive*, and then observing that the presentation that we obtain is in fact polynomial-time.

Definition 1.1 (Essentially Dedekind [Ded60])**.** A function $f : \mathbb{N} \to \mathbb{N}$ is *primitive recursive* if f can be generated from the basic functions $s(x) = x + 1$, $o(x) = 0$, $I^n_m(x_1, \ldots, x_n) = x_m$ by composition and the primitive recursion operator $h = \mathcal{P}(f, g)$:

$$h(x_1, \ldots, x_n, 0) = f(x_1, \ldots, x_n),$$
$$h(x_1, \ldots, x_n, y + 1) = g(x_1, \ldots, x_n, y, h(x_1, \ldots, x_n, y)).$$

The restricted Church-Turing thesis for primitive recursive functions says that a function is primitive recursive iff it can be described by an algorithm that uses only bounded loops. For example, we need to eliminate all instances of WHILE ... DO, REPEAT ... UNTIL, and GOTO in a PASCAL-like language.

As we noted above, primitive recursion plays a rather important intermediate role in transforming (Turing) computable structures into polynomial-time structures. Furthermore, to illustrate that a structure has no polynomial time copy, it is sometimes easiest to argue that it does not even have a copy with primitive recursive operations, see e.g. [CR92]. It is thus natural to systematically investigate into those structures that admit a presentation with primitive recursive operations, as defined below. Kalimullin, Melnikov, and Ng [KMN17b] proposed that an "online" structure must minimally satisfy:

Definition 1.2 ([KMN17b, Mel17])**.** A countable structure is *fully primitive recursive* (fpr) if its domain is $\mathbb{N}$ and the operations and predicates of the structure are (uniformly) primitive recursive.

The main intuition is that we need to define more of the structure "without delay". Here "delay" really means an instance of a truly unbounded search. We informally call fpr structures *punctually computable*. We could also agree that all finite structures are also punctual by allowing initial segments of $\mathbb{N}$ to serve as their domains. Although the definition above is not restricted to finite languages, we will never consider infinite languages in the paper; therefore, we do not clarify what uniformity means in Def. 1.2.

Remark 1.3. The word "fully" in "fully primitive recursive" emphasises that the domain must be the whole of $\mathbb{N}$ and not merely a primitive recursive subset of $\mathbb{N}$; these are provably non-equivalent assumptions. If the domain could be merely a primitive recursive subset of $\mathbb{N}$ then we can delay elements from appearing in the structure; this way one can easily show that each Turing computable graph has a primitive recursive copy; cf. Theorem 2.2 below. We decided that structures in which elements can be delayed are not really online.

Our goal is to give a most general setting that also reflects the common online structures encountered. From a logician's point of view, where do computable structures come from? One of the *fundamental results* of computable structure theory is that:

A decidable theory has a decidable model.

The proof of this elementary fact is to observe that the Henkin construction is effective, in that if the theory is decidable then the constructed model is decidable as a model. Many standard computable structures come from decidable theories.

Most natural decidable theories are elementary decidable in that the decision procedures are relatively low level. We have to go out of our way to have natural decidable theories whose decision procedures are not primitive recursive. It is not hard to show (see below) that a theory with a primitive recursive decision procedure has a model which is decidable in a primitive recursive sense.

1.6.4. *The upshot.* We have chosen fully primitive recursive structures as our central model. Primitive recursiveness gives a useful *unifying abstraction* to computational processes for structures with computationally bounded presentations. In such investigations we only care that there is *some* bound. Furthermore, these models arise quite naturally through standard decision procedures. Irrelevant counting combinatorics is stripped from such proofs, thus emphasising the effects related to the existence of a bound in principle. These effects are far more significant than it may seem at first glance. Before we describe deeper results into this direction, we must give examples of punctual structures in common classes.

2. The first steps. Examples

2.1. **Existence of punctual presentations.** So what kinds of basic structures have punctual presentations?

Theorem 2.1 (Kalimullin, Melnikov, and Ng [KMN17b])**.** *The following structures all have computable presentations iff they have fully primitive recursive ones.*

(1) *Linear orderings (Grigorieff* [Gri90]*)*
(2) *Boolean algebras*
(3) *Equivalence structures*
(4) *Torsion-free abelian groups*
(5) *Abelian p-groups*
(6) *Locally finite graphs*

Discussion. We outline the case of locally finite graphs. Although this result is a triviality based on the general idea from [CR91], it is actually new.

Suppose we are given an infinite computable locally finite G. Because G is locally finite, for every finite subgraph H of G there exists an infinite set S of nodes in G which are not related to H by an edge and are also pairwise non-related.

We start by quickly enumerating an infinite set S that currently looks independent and we wait for more of G to be computed. We copy the current finite part of G into what we've build so far. When more nodes appear in G we extend the isomorphic embedding ψ of G into our copy, but we always make sure that a bit more of the set S is used in the range of ψ. It is not hard to see that eventually all of the set S will be used. □

Although some clauses of Theorem 2.1 above use non-trivial techniques such as Dobritsa's result [Dob83] in (4), all these proofs exploit a similar idea: Quickly enumerate an infinite nice subset of the structure that we can predict and use it as a delaying gadget. This is of course not really an honest "online" proof. However, we will see that algebraic structures typically have infinitely many primitive recursive isomorphic copies, some of these presentations will be "more online" and the other ones will be "less online" in the sense that will be clarified later. A large portion of our new theory is focused on *comparing* different punctual copies of structures.

In contrast to Theorem 2.1, Kalimullin, Melnikov, and Ng [KMN17b] also showed:

Theorem 2.2 (Kalimullin, Melnikov, and Ng [KMN17b])**.** *In each of the classes below, there are examples of computably presentable structures without fully primitive recursive presentations.*

(1) *Torsion abelian groups.*
(2) *Undirected graphs.*
(3) *Archimedean ordered abelian groups.*

Proof idea. We note that the proofs of the different clauses use substantially different ideas. We outline the proof of (3) which uses rudiments of computable analysis.

We define a subgroup of $\mathbb{R}$ as a $\mathbb{Q}$-vector space over $\{1, \mathbf{r}\}$, where $\mathbf{r}$ is a computable real in the sense of Turing [Tur36, Wei00] such that $\mathbf{r}$ does not have a primitive recursive rapid approximation by basic intervals. Such a real can be easily constructed using a straightforward diagonalisation technique.

Observe that the ordered group has a Turing computable presentation iff $\mathbf{r}$ is a computable real, and a fully primitive recursive presentation iff $\mathbf{r}$ admits a primitive recursive rapid approximation. □

It is natural to ask whether there is a general description of computable structures that admit a punctual presentation. The answer to this question is negative. Let $(M_e)_{e\in\omega}$ be the effective list of all partially computable algebraic structures.

Theorem 2.3. [BHTK$^+$] *The index set* $\{e : M_e$ *has a punctual presentation*$\}$ *is* Σ^1_1*-complete.*

In other words, there is no simpler way to see whether a computable structure has a punctual copy than just stating that there is an isomorphism from the structure to some punctual presentation. This solves a problem left open is the brief conference survey [Mel17]. The proof is too technical to be discussed in this survey, but [BHTK$^+$] contains an extended informal discussion of the proof.

Suprisingly, the same diagonalisation technique with insignificant adjustments allows to prove:

Theorem 2.4. [BHTK$^+$] *The index set* $\{e : M_e$ *has an automatic presentation*$\}$ *is* Σ^1_1*-complete.*

Theorem 2.5. [BHTK$^+$] *The index set* $\{e : M_e$ *has a polynomial-time presentation*$\}$ *is* Σ^1_1*-complete.*

Theorem 2.4 answers a long standing open question by Khoussainov and Nerode [KN95]. We see that the emerging theory of primitive recursive structures has already found significant applications.

2.2. **Primitive recursive decidability.** Recall the discussion after Theorem 2.1. In many cases the easiest way of producing a punctual copy of a structure is to use a predictable part of it to "delay" the enumeration of the non-trivial parts of it. As noted in [KMN17b], the resulting structure is not an honest online structure.

It would be more satisfying if we had a way of pressing specific parts of the structure to be quickly revealed. One way of formalising this idea uses first-order logic. To make the terminology more easily expressible, we will adopt the adjective "punctual" for primitive recursiveness. Thus we have the following.

Definition 2.6. We will call a structure with universe $\mathbb{N}$ *punctually decidable* if it has a primitive recursive Skolem function; that is, the existential witnesses for first-order formulae $\exists a \phi(a, \bar{c})$ with parameters from the structure can be found punctually in their Gödel indices. (If the formula fails then the function returns -1).

Proposition 2.7. *A theory T with a primitive recursive decision procedure has a punctually decidable model.*

Proof Sketch. Observe that the usual Henkin construction works. Recall that we add new constants $C = \{c_i \mid i \in \mathbb{N}\}$ and consider a (primitive recursive) enumeration $\{\tau_j \mid j \in \mathbb{N}\}$ of sentences of the language $L(T)$ of T together with the constants. We construct the model A in stages, and a complete and promptly decidable theory $Q = \{\phi_0, \phi_1, \dots\}$ in $L(T) \cup C$ in stages. We let ϕ_0 denote $(c_0 = c_0)$.

At stage $s = 2e + 1$ if ϕ_e is of the form $\exists x \theta(x)$, as usual find the least i with c_i not occurring in Q_s and let $\phi_s = \theta(c_i)$ so that we realize the formula.

At stage $s = 2e$, let $\bar{c}$ denote the constants in $(\wedge_{\phi_i \in Q_s} \phi_i) \to \tau_e$. Let $\overline{x}$ denote the first sequence of variables of length $|\bar{c}|$ not occurring in $(\wedge_{\phi_i \in Q_s} \phi_i) \to \tau_e$. We punctually check whether T proves $\forall \overline{x}((\wedge_{\phi_i \in Q_s} \phi_i) \to \tau_e)[\overline{x}/\bar{c}]$. If so we let $\phi_s = \tau_e$, and if not, $\phi_s = \neg \tau_e$.

The result is a punctually decidable structure A which models T. □

Recall that the standard method of saying a structure is a decidable one is to say its full diagram is computable. Does the existence of punctual $\exists$-witnesses in a punctual structure $\mathcal{A}$ follow from its full diagram $\{\phi(\bar{a}) : \mathcal{A} \models \phi(\bar{a}), \bar{a} \text{ in } \mathcal{A}\}$ being primitively recursively decidable? The answer is clearly negative, as illustrated by the straightforward example below.

Example 2.8. Consider an equivalence structure E that has infinitely many classes, each of size 2. Construct a punctual copy of E. Start by rapidly enumerating an infinite set consisting of non-equivalent elements. Delay the second representatives of some equivalence classes. We can primitively recursively decide first-order statements about elements in the resulting punctual presentation, however, the Skolem function will not be punctual.

Perhaps, asking for the full diagram to be punctually decidable is too much. Perhaps, 1-decidability is already good enough. Restrict the definition of punctually decidable structure to quantifier-free ϕ to get the natural notion of a punctually 1-decidable structure (in [KMN17b] is was called strongly punctual). More formally, there exists a primitive recursive Φ such that

$$\Phi(\bar{c}, \phi) = \begin{cases} -1, & \text{if } \mathcal{I} \not\models \exists x \phi(\bar{c}, x), \\ y, & \text{such that } \mathcal{I} \models \phi(\bar{c}, y), \end{cases}$$

where $\bar{c} \in \mathcal{I}$ and ϕ is (the Gödel number of) a quantifier-free formula in the language of the structure. We note that this more relaxed approach resembles the earlier notion of an *honest witness* due to Cenzer and Remmel [CR91].

Clearly, there exist punctual structures that have no 1-decidable presentation, and thus no punctually 1-decidable presentation. For instance, there exist computable linear orders in which the successivity relation is intrinsically undecidable [Dow98], and similarly there exists a computable Boolean algebra in which the atom relation is intrinsically undecidable [Gon97]. Now Theorem 2.1 guarantees that in each of these classes there are examples of punctually presented structures that have no 1-decidable copy, let alone punctually 1-decidable copy. However, these examples are unsatisfying since they all give punctual structures that are not even 1-decidable. A rather straightforward example below separates strongly primitive recursive structures from 1-decidable punctual structures.

Proposition 2.9. [KMN17b] *There exists a punctual* 1*-decidable equivalence structure that has no strongly punctual presentation.*

Proof. For any infinite set X, let $E(X)$ denote the equivalence structure having exactly one class of size x for each $x \in X$. Note that for an infinite c.e. set X, the structure $E(X)$ has a computable, hence punctual presentation (by Theorem 2.1(3)). To make $E(X)$ 1-decidable, make the k^{th} class have size exactly x_k, where $X = \{x_0, x_1, x_2, \ldots\}$ is some computable enumeration of X. It is easy to see that deciding an existential formula about $\bar{c}$ boils down to deciding the sizes of the classes that contain $\bar{c}$.

Thus, it remains to build an infinite c.e. set X (in fact, X will be computable) such that $E(X)$ has no strongly punctual presentation. Suppose we have enumerated $\{x_0, \ldots, x_k\}$. Suppose we want to diagonalize against the e^{th} potential strongly punctual structure S_e. When S_e is first processed we use the primitive recursive Skolem function in S_e to primitively recursively decide if there exists a class $[z]$ of size $> s$.

If no then we win because S_e must contain at least two classes of equal sizes, and thus $E(X) \not\cong S_e$. If yes then we can primitively recursively compute a witness z. In this case we say that S_e is pending with witness z. We will ensure that all future elements of X are chosen to be smaller than the current approximation to the size of $[z]$ in S_e.

At stage s of the construction we process each requirement S_e for $e < s$. If S_e is unstarted then we proceed as above, and move to the next requirement. If S_e is already pending with witness z we check if the size of $[z]$ in S_e is larger than s. If yes, the status of S_e remains pending, and we move to the next requirement. If no then the size of $[z]$ must be s. In this case we terminate the actions of stage s at S_e and initialize all lower priority requirements.

It is easy to see that if s is enumerated in X at stage s then this is compatible with the satisfaction of all requirements R_e, $e < s$. Each requirement is initialized finitely often and will be met. Finally X is infinite because only a pending requirement can block the enumeration of s into X at stage s. □

2.3. **Punctual versions of known results.** Another relatively straightforward way to extend the known results on Turing computable models uses the primitive recursive analogy of relativisation. In other words, sometimes a statement of a known result can be modified to a similar statement about punctual structures. If

we are careful enough we might be able to keep almost the same proof. The "punctualisation" principle tends to be a bit more subtle than the standard relativisation principle. To illustrate this principle we shall consider the well-known result of Harrington [Har74] and Goncharov and Nurtazin [GN73] which states that a decidable (complete) theory has a decidable prime model iff the set of principle types of the theory is uniformly computable. To state and prove the primitive recursive version of this result we need several definitions.

If $q(\bar{x}) = q(x_0, x_1, \ldots, x_{n-1})$ is a (complete) n-type of a theory T, then as usual, we identify $q(\bar{x})$ with the set of Gödel numbers of formulae from $q(\bar{x})$. Consider a sequence $(q_i)_{i\in\omega}$ of types of a theory T. We say that the sequence $(q_i)_{i\in\omega}$ is *increasing* if, for each $i \in \omega$, it satisfies the following conditions:

(a) $q_i = q_i(x_0, x_1, \ldots, x_i)$ and $(x_j \neq x_k) \in q_i$ for all $j < k \leq i$;
(b) $q_i \subset q_{i+1}$.

We say that a sequence of types $(q_i)_{i\in\omega}$ is *uniformly primitive recursive* if there is a primitive recursive function $f_{type}\colon \omega^2 \to \{0, 1\}$ such that for every i, the function $f_{type}(i, \cdot)$ is the characteristic function of the type q_i.

Suppose that $(q_i)_{i\in\omega}$ is an increasing sequence of types. The sequence $(q_i)_{i\in\omega}$ *has quick witnesses* if there is a primitive recursive function $g_{wit}(x)$ with the following property:

(QW) If k is the Gödel number of a formula $\psi(x_{i_0}, \ldots, x_{i_n}) = \exists y \theta(x_{i_0}, \ldots, x_{i_n}, y)$, where $i_0 < \cdots < i_n$, and $\psi \in q_{i_n}$, then the formula $\theta(x_{i_0}, \ldots, x_{i_n}, x_{g_{wit}(k)})$ belongs to the type $q_{max(i_n, g_{wit(k)})}$.

Theorem 2.10. *Suppose that T is a complete theory with a primitive recursive decision procedure. Then the following conditions are equivalent:*

(i) *T has a punctually decidable prime model;*
(ii) *T has a prime model and there is an increasing, uniformly primitive recursive sequence $(q_i)_{i\in\omega}$ of principal types of T such that $(q_i)_{i\in\omega}$ has quick witnesses.*

Proof. We follow the standard proof of Harrington [Har74] and Goncharov and Nurtazin [GN73] which can be found in [Har98] and see what has to be modified.

(i)⇒(ii). Let $\mathcal{M}$ be a punctually decidable prime model of T, i.e. $\mathcal{M}$ has a primitive recursive Skolem function. Recall that the set of types realized in $\mathcal{M}$ is precisely the set of all principal types of T.

For $i \in \omega$, let $q_i(x_0, x_1, \ldots, x_i)$ be the type realized by the tuple $(0, 1, \ldots, i)$ in $\mathcal{M}$. The desired function f_{type} (from the definition of a uniformly primitive recursive sequence of types) can be defined as follows: for $i \in \omega$,

(1) If $y \in \omega$ is not a Gödel number of a first-order formula of the form $\psi(x_0, \ldots, x_i)$, then set $f(i, y) := 0$.
(2) Otherwise, let $\psi(\bar{x})$ be the formula with the number y. Using the Skolem function, quickly decide whether

$$\mathcal{M} \models \psi(0, 1, \ldots, i). \tag{2.1}$$

If (2.1) is true, then set $f(i, y) := 1$. Otherwise, define $f(i, y) := 0$.

The described procedure shows that the sequence $(q_i)_{i\in\omega}$ is uniformly primitive recursive.

A function g_{wit} giving quick witnesses for $(q_i)_{i\in\omega}$ is recovered in a straightforward way: If k is the Gödel number of a formula $\psi(\bar{x}) = \exists y \theta(x_{i_0}, \ldots, x_{i_n}, y)$, then check whether $\psi(i_0, \ldots, i_n)$ holds, using the Skolem function. If $\mathcal{M} \not\models \psi(i_0, \ldots, i_n)$, then set $g_{wit}(k) := 0$. Otherwise, quickly find an element m with $\mathcal{M} \models \theta(i_0, \ldots, i_n, m)$ and define $g_{wit}(k) := m$.

(ii)⇒(i). Let $(q_i)_{i\in\omega}$ be an increasing, uniformly primitive recursive sequence of principal types from (ii). Fix a function f_{type} witnessing the uniform primitive recursiveness of the sequence $(q_i)_{i\in\omega}$. Let g_{wit} be a function giving quick witnesses for $(q_i)_{i\in\omega}$.

Let $C = \{c_i : i \in \omega\}$ be a set of new constants. We define the complete diagram Ξ of a structure $\mathcal{N}$ as follows. The set C will be the domain of $\mathcal{N}$. Suppose that ψ is a sentence in the language $L(T) \cup C$, and n is the largest number such that the constant c_n occurs in ψ (if no c_n occurs in ψ, then just set $n := 0$). We find the Gödel number k of the formula $\psi(x_0/c_0, x_1/c_1, \ldots, x_n/c_n)$ and compute the value $f_{type}(n, k)$. If $f_{type}(n, k) = 1$, then $\psi \in \Xi$. Otherwise, $\psi \notin \Xi$.

It is not hard to show that the described procedure produces a well-defined complete diagram: Indeed, for any sentence $\psi = \psi(c_0, \ldots, c_n)$, either $\psi(\bar{x}/\bar{c})$ or $\neg\psi(\bar{x}/\bar{c})$ belongs to the type q_n. Hence, either ψ or $\neg\psi$ lies in Ξ; therefore, Ξ is a complete set of sentences. Furthermore, since $q_i \subset q_{i+1}$ for all i, the set Ξ is consistent.

Since $T \subset q_i$ for every i, the structure $\mathcal{N}$ is a model of T. Recall that $(x_j \neq x_k) \in q_i$ for all $j < k \leq i$; thus, $\mathcal{M} \models (c_j \neq c_k)$ for all $j \neq k$. Therefore, one may assume that the domain of $\mathcal{N}$ is equal to ω.

A primitive recursive Skolem function $Sk(x)$ can be defined as follows. Suppose that k is the Gödel number of a sentence

$$\psi(c_{i_0}, \ldots, c_{i_n}) = \exists y \theta(c_{i_0}, \ldots, c_{i_n}, y),$$

where $i_0 < \cdots < i_n$. Using the function $f_{type}(i_n, \cdot)$, we promptly check whether the formula $\psi(\bar{x}/\bar{c})$ belongs to q_{i_n}. If $\psi(\bar{x}/\bar{c}) \notin q_{i_n}$, then set $Sk(k) := -1$. Otherwise, compute the value $g_{wit}(k)$ and define $Sk(k) := c_{g_{wit}(k)}$. Hence, the structure $\mathcal{N}$ is punctually decidable.

Now it is sufficient to show that the model $\mathcal{N}$ is atomic. The definition of $\mathcal{N}$ ensures that for every i, the tuple $\bar{c}^i := (c_0, \ldots, c_i)$ realizes the type q_i and thus, $\bar{c}^i$ satisfies some complete formula $\psi^i(x_0, \ldots, x_i)$. We need to prove that any tuple $(c_{j_0}, \ldots, c_{j_k})$ also satisfies a complete formula. For simplicity, we assume that $(c_{j_0}, \ldots, c_{j_k}) = (c_{m+1}, \ldots, c_{m+r})$ for some m and $r \neq 0$. Then we have

$$\mathcal{M} \models \exists y_0 \ldots \exists y_m \psi^{m+r}(y_0, \ldots, y_m, c_{m+1}, \ldots, c_{m+r}).$$

Note that for any formula $\xi(\bar{x}) = \xi(x_{m+1}, \ldots, x_{m+r})$, we have either $T \vdash (\psi^{m+r} \to \xi)$ or $T \vdash (\psi^{m+r} \to \neg\xi)$, This implies that for any $\xi(\bar{x})$, either $T \vdash (\exists \bar{y} \psi^{m+r}(\bar{y}, \bar{x}) \to \xi(\bar{x}))$ or $T \vdash (\exists \bar{y} \psi^{m+r}(\bar{y}, \bar{x}) \to \neg\xi(\bar{x}))$. Thus, $\exists \bar{y} \psi^{m+r}(\bar{y}, \bar{x})$ is a complete formula of the theory T, and $(c_{m+1}, \ldots, c_{m+r})$ realizes a principal type. Therefore, the model $\mathcal{N}$ is prime. Theorem 2.10 is proved. □

Also, Downey, Harrison-Trainor, Greenberg, and Turetsky have recently observed that it follows from Millar's work [Mil83] that a complete primitive recursive

theory T has a punctually decidable model that omits a given primitive recursive non-principal type p.

Pure primitive recursive model theory is not yet developed aside from the observations included in this section. An interesting project will be to see what develops henceforth.

Problem 2.11. Develop punctual model theory.

3. Uniqueness of presentation

Recall that in the previous section we looked at various examples of punctually presented structures. We also noted that often the easiest way of producing a punctual copy is "dishonest" since we typically construct a copy which is not punctually 1-decidable. However, these same structures will usually have punctual presentations which are much better behaved; see, e.g., Example 2.8. The same structure will usually have more than one punctual presentation, with different presentations having substantially different "online" properties. Consider the following simple but instructive example:

Example 3.1. Let (ω, S) be the unary structure of the natural numbers with the successor. Clearly, the "natural" presentation N of (ω, S) has a number of pleasant online features such as punctual 1-decidability. On the other hand, we can construct a "bad" punctual copy B of (ω, S) which has no punctual Skolem function, as follows. Introduce a new element x and keep it disconnected from 0. Wait for as long as necessary for diagonalisation against the eth potential Skolem function using $\exists y\, S(y) = x$. Then connect x to the origin. Repeat for $e + 1$, etc.

Note that the unique isomorphism from N onto B is primitive recursive *but its inverse is not primitive recursive.*

The example above brings us to the problem of *comparing* different punctual presentations of the same algebraic structure. When two punctual copies of the same structure are identical from the perspective of our framework?

To answer this question we use the intuition coming from computable structure theory. The central classification tool in algebraic structure theory is algebraic isomorphism. Whenever we talk about (Turing) *computable* structure theory, we keep in mind that the central classification tool is a (Turing) *computable isomorphism.* This fundamental principle was implicit in the dawn of the modern incarnation of effective mathematics. One of the fundamental papers from this period is Fröhlich and Shepherdson [FS56]. This paper clearly shows the historical context of the subject, the clear intuition of van der Waerden (which apparently came from Emmy Noether's lecture notes) and the fact that isomorphic computable structures (here fields) can have distinct algorithmic properties, and hence cannot be computably isomorphic. Here we quote from the abstract.

> "Van der Waerden (1930a, pp. 128–131) has discussed the problem of carrying out certain field theoretical procedures effectively, i.e. in a finite number of steps. He defined an 'explicitly given' field as one whose elements are uniquely represented by distinguishable symbols with which one can perform the operations of addition, multiplication, subtraction and division in a finite number of steps. He pointed out that if a field K is explicitly given then any finite extension K' of K can be explicitly given, and that if there is a splitting algorithm for K, i.e. an effective procedure for splitting polynomials with coefficients in K into their irreducible factors in $K[x]$, then (1) there is a splitting algorithm

> for K'. He observed in (1930b), however, that there was no general splitting algorithm applicable to all explicitly given fields K ... We sharpen van der Waerden's result on the non-existence of a general splitting algorithm by constructing (§7) a particular explicitly given field which has no splitting algorithm. We show (§7) that the result on the existence of a splitting algorithm for a finite extension field does not hold for inseparable extensions, i.e. we construct a particular explicitly given field K and an explicitly given inseparable algebraic extension $K(x)$ such that K has a splitting algorithm but $K(x)$ has not."

So in modern terms Fröhlich and Shepherdson [FS56] showed that the halting problem is many-one reducible to the problem of having a splitting algorithm[1]. Subsequently, Mal'tsev [Mal62] gave an example of an abelian group which has two non-computably isomorphic computable copies; in one copy there is an algorithm for linear dependence, and in the other copy there is no such algorithm. Mal'tsev [Mal61] proposed that (Turing) computable presentations must be identified under (Turing) computable isomorphism. He also suggested the notion of computable categoricity (autostability): A structure is computably categorical if it has a unique (Turing) computable copy up to (Turing) computable isomorphism.

We go back to primitive recursion. Here we may be tempted to use primitive recursive isomorphism as the fundamental classification tool. However, the elementary Example 3.1 above provides us with two primitively recursively isomorphic punctual copies of (ω, S) which have substantially different punctual properties. Of course we need to be more careful.

Definition 3.2. We say that punctually computable structures $\mathcal{A}$ and $\mathcal{B}$ are punctually isomorphic iff there is an isomorphism f taking $\mathcal{A}$ to $\mathcal{B}$ with both f and f^{-1} primitive recursive. (Functions f with this property will be called fully primitive recursive or fpr for short.)

Some results of computable structure theory lift easily. In this section we focus on such results, while the later sections will be devoted to more technical theorems.

For example, we can easily show that Mal'tsev's example discussed above can be made primitive recursive. We can also look at the Fröhlich-Shepherdson work and observe that every primitive recursively presentable field has a fully primitive recursively presentable algebraic closure.

However, there is much mathematical depth in the new definition. This depth comes from the fact that we must define the isomorphism *and its inverse* "now". That is we specify the domain of $\mathcal{A}$ in stages s once a enters the domain by some primitive recursive time stamp $g(s)$ we need a $b \in \mathrm{dom}\mathcal{B}_{g(s)}$ with $f(a) = b$ and additionally for each c occurring in $\mathrm{dom}\mathcal{B}_s$ by some primitive recursive time stamp $h(s)$ we must specify a $d \in \mathrm{dom}\mathcal{A}_{h(s)}$ with $f(d) = c$. This punctuality means that many classical categoricity arguments *fail* for primitive recursive structures. For instance, (ω, S) clearly has a unique Turing computable copy, but Example 3.1 gives us two non-fpr isomorphic punctual presentations of the structure. Consider also the following example.

Example 3.3. The very first categoricity argument we meet will be Cantor's proof that the countable dense linear ordering without end-points is categorical. This

[1] Metakides and Nerode [MN79] proved this closure was computably unique (i.e. up to computable isomorphism) iff the field has a separable splitting algorithm. (The message here is that the usual method of constructing a closure via adjoining roots essentially using a splitting algorithm is not the only way to construct a closure.)

proof is effective but the proof involves *unbounded search*. That is, at some stage s we have defined $x < y < z$ in the domain of one copy, and have already specified $f(x) < f(y)$. Density guarantees that at some stage some q will enter the other copy of the ordering between $f(x)$ and $f(y)$ (thinking of them as being computably presented), and we can then map $f(z) = q$. *But in the primitive recursive case, why should such a q enter the other copy* punctually*?* In fact, it provably does not have to.

The example above can be extended to prove the following. If a structure has a unique punctual presentation up to fpr isomorphism then we say that it is *punctually categorical*.

Theorem 3.4. [KMN17b]

(1) *An equivalence structure S is punctually categorical iff it is either of the form $F \cup E$, where F is finite and E has only classes of size 1, or S has finitely many classes at most one of which is infinite.*
(2) *A linear order is punctually categorical iff it is finite.*
(3) *A Boolean algebra is punctually categorical iff it is finite.*
(4) *An abelian p-group is punctually categorical iff it has the form $F \oplus \mathbb{V}$, where $p\mathbb{V} = \mathbf{0}$ and F is finite.*
(5) *A torsion-free abelian group is punctually categorical iff it is the trivial group $\mathbf{0}$.*

Discussion. In some clauses the proof follows from Theorem 2.1 and the known description of computable categoricity in the respective class. For instance, for (2), note that each computable linear order is computably isomorphic to a punctually computable one (follows from the proof of Theorem 2.1). It is well-known that a linear order is computably categorical iff it has only finitely many adjacencies. Suppose a punctual L is not like that. Produce a computable copy B of L that is not computably isomorphic to L, and then computably transform it into a punctual B'. Then L and B' are not even computably isomorphic. On the other hand, if L has only finitely many adjacencies, then use the idea from Example 3.3.

The case of Boolean algebras has a less straightforward proof. The problem is that Theorem 2.1 in the case of Boolean algebras does not necessarily give a computably isomorphic punctual copy. Thus, to prove (3) we have to combine the strategies from the respective clause of Theorem 2.1 with a diagonalisation requirement. □

The highly unexpected result below will play a significant role in the last paragraph of this article.

Theorem 3.5 ([DHTK$^+$])**.** Let G be an undirected graph. Then the following are equivalent:

(1) G is punctually categorical.
(2) G becomes a clique or an anti-clique (an independent set) after removing finitely many vertices $\bar{v} = v_0, \dots, v_k$ with each v_i being either adjacent to all $x \in (G - \bar{v})$ or not adjacent to all $x \in (G - \bar{v})$.

Sketch. It is not hard to see that locally finite graphs satisfy the theorem. The proof in this special case goes through several cases. For example, suppose the graph is connected. In this case we use that at every stage there will be vertices

"far-far away" from what we've already built. Build a "bad" punctual copy B in which we have a punctual sequence of pairwise disconnected vertices. We use this sequence to delay the other parts of the graph from being enumerated into the bad copy, and we use them for diagonalisation purposes. Then slowly incorporate these extra vertices into the expanding actual image of our graph within B. The case of several components is similar. This argument can be extended to the case of at most finitely many vertices of infinite degree.

In the harder case of infinitely many vertices of infinite degree, we use the key technical proposition below:

Proposition 3.6. *Suppose G is punctually categorical, and $x \in G$ has degree ∞. Then the set $N(x) = \{y \in G : (x, y) \in E(G)\}$ has to be punctual; that is, there is a primitive recursive lower bound on the speed of its enumeration in G.*

Sketch of Prop. 3.6. We write $|X|$ for the cardinality of X. Imagine that x is the only vertex of G with the property $deg(x) = \infty$, and assume $N(x)$ is very slowly growing, i.e., there is no primitive recursive bound on the stage at which the nth vertex appears in $N(x)$. In this simple case the strategy is straightforward. Build a copy B of G which is essentially identical to G but with $|N_B(x)| = |N_G(x)| - 1$ at every stage. Any isomorphism must match the points of infinite degree, and we know the graph has only one such point. We also know that "most of the time" G puts points into $G \setminus N_G(x)$, and therefore in B we can eventually delay one element from appearing in the 1-neighbourhood of x and still keep B punctual. All we need to do is to wait until $p : G \to B$ is diagonalised on the extra vertex that G has in $N(x)$ when compared with B. Note that we must eventually succeed, for otherwise we would use p to extract a primitive recursive bound on the speed of growth of $N(x)$ in G.

When G has many vertices of infinite degree and is not rigid, we have to consider 2 copies, A and B, of G and look at two vertices in G. This is necessary because $p : B \to A$ does not have to map the natural version of x in B to the natural version of x in A. So suppose A is copying G via ψ and B via ϕ (both are defined by us), and suppose we are trying to diagonalise against a pair (p, q), where $p : B \to A$ and (supposedly) $p^{-1} = q$.

The idea is to either keep $|N_A(p\phi(x))| < |N_B(\phi(x))|$ or $|N_A(p\phi(x))| > |N_B(\phi(x))|$ for as long as possible, and use either p or q to press the opponent to grow the respective neighbourhoods in G. It is also crucial to avoid $|N_A(p\phi(x))| = |N_B(\phi(x))|$ at all costs, because in this situation we cannot "press" the opponent. We informally explain how we "press" below.

For example, suppose at stage s have $|N_A(p\phi(x))| < |N_B(\phi(x))|$. Then evaluate p on $N_B(\phi(x))[s]$; the opponent must grow $N_G(\psi^{-1}p\phi(x))$ in G. The trick here is that we do not make the 1-neighbourhood equal, but rather delay at most one point and press the opponent to grow $N_G(\psi^{-1}p\phi(x))$ 1-point larger than $N_B(\phi(x))[s]$. So one point appears in G and makes the two 1-neighbourhoods look equal; however, we keep this point out of our structure A and wait for another point to appear in $N_G(\psi^{-1}p\phi(x))$. This is fine to delay one point from the diagram of A. Just consider the next point u in G. If $u \notin N_G(\psi^{-1}p\phi(x))$ then copy it into A. If it is in $N_G(\psi^{-1}p\phi(x))$ then this is exactly what we needed. But recall that we challenged the opponent by computing p on the currently larger $N_B(\phi(x))$. The opponent must respond by enumerating more points into $N_G(\psi^{-1}p\phi(x))$, and it must do so within the time bound of p, for otherwise p is not an isomorphism. Thus, unless $N_G(x)$

(thus, $N_B(\phi(x))$) grows within the time needed for $p(N_B(\phi(x))[s])$ to converge, we will be able to make $|N_A(p\phi(x))| > |N_B(\phi(x))|$. But then we can evaluate q on $N_A(p\phi(x))$ and similarly press the opponent to grow $N_B(\phi(x))$ by extending $N_G(x)$ in G; this was our ultimate goal. In the worst case scenario the time bound can be extracted from and $q(N_A(p\phi(x))[t])$, where t depends on the convergence time for $p(N_B(\phi(x))[s])$, making the described above process punctual.

The sketch above describes the worst case scenario, but there will be various cases which also had to be incorporated into the formal argument in [DHTK$^+$]. To maintain the inequality $|N_A(p\phi(x))| \neq |N_B(\phi(x))|$ at every stage we will have to consider the cases when either $N_A(p\phi(x))$ or $N_B(\phi(x))$ starts growing faster than anticipated. Also, $|N_A(p\phi(x))|$ and $|N_B(\phi(x))|$ do not have to differ by only one element at a given stage. In all these cases we have even more advantage over the opponent, however, considering such cases significantly increases the combinatorial complexity of the formal argument. □

Proposition 3.7. *Suppose G is punctually categorical, and $a, b \in G$ both have infinite degree. Then $N(a) =^* N(b)$, i.e., a and b share the same adjacent vertices, up to a finite difference.*

Proof. By Prop. 3.6, both $N(a)$ and $N(a)$ must be punctual in any punctual copy of G. If there is an infinite punctual sequence in $N(b) \setminus N(a)$, then it is easy to produce a punctual copy of G in which $N(a)$ is not punctual, contradicting Prop. **??**. To do so punctually list elements in $N(b) \setminus N(a)$ and use the usual trick to delay other elements, including those in $N(a)$, from appearing in the copy.

Thus, $N(b) \setminus N(a)$ must have no infinite punctual subsequence. In other words, there is no primitive recursive bound on how long we wait between the stages when we see new points in $N(b)$ but *not* in $N(a)$. Informally, this means that $N(b)$ does not press us to give points *not connected* to a. The trick is to think about the graph-theoretic complement $\overline{G}$ of G (i.e., the graph on the same vertices but with the non-edge relation of G). Unless $N(b) \setminus N(a)$ is finite, we can modify G to get a new punctual copy H in which the non-edge 1-neighbourhood of a is not punctual, as follows.

Copy only $N(b)$ into H to diagonalise against the next primitive recursive bound on the speed of growth of $N(a)$ in $\overline{G}$. We must eventually see a disagreement, for otherwise we could use the construction of H to produce a similar bound for $N(b) \setminus N(a)$, contradicting the assumption. As soon as a disagreement is found, copy all the currently skipped points into H and restart from there, this time for the next primitive recursive bound. □

Proposition 3.8. *Suppose G is punctually categorical and $deg(x) = \infty$ for some $x \in G$. Then x is connected to a.e. vertex in G.*

Proof. Artificially adjoin an extra vertex v to G and connect v to all vertices in G by an edge. Then (G, v) is punctually categorical as well. Apply the previous proposition to see that $G = N(v) =^* N(x)$. □

Proposition 3.9. *Suppose G has infinitely many vertices of infinite degree. For each $n \in \omega$, there are only finitely many vertices of degree n. The same holds for co-degree n.*

Proof. Fix any $n+1$ distinct vertices of infinite degree. By Proposition 3.8, almost every vertex in G is adjacent to *all* of them, and thus has degree at least $n+1$. □

Proposition 3.10. *Suppose G has infinitely many vertices of infinite degree and infinitely many vertices of finite degree. For each $v \in G$, consider $(G - v)$ which stands for G without v. Then $G \not\cong (G - v)$.*

Proof. Consider the case when $deg(v) = n < \infty$, the case of co-finite degree is symmetric. Notice that for any x in (G-v), its degree in (G-v) is either the same as its degree in G, or is one less. And the latter can only happen if x is adjacent to v.

Suppose v has degree n. By Proposition 3.9, there are only finitely many vertices of degree n. If G is isomorphic to (G-v), some y_0 in G drops from degree $n+1$ to degree n. But then there must also be some y_1 which drops from degree $n+2$ to degree $n+1$, etc. But each of these y_i is adjacent to v, contrary to the assumption that v has finite degree. □

Suppose G (or its complement) is not essentially locally finite. To diagonalise against a pair (p, q), delay one vertex v in an auxiliary copy that we build. Wait for (p, q) to illustrate a disagreement, and then put v into the copy.

We conclude that G must be either a clique or an anti-clique after removing finitely many nodes. It is fairly obvious that such a G must also be automorphically trivial; we omit details. □

One naturally seeks to extend the argument above to more complex combinatorial objects:

Problem 3.11. *Is there an algebraic description of punctually categorical structures with finitely many binary relations? What about ternary relations?*

Note that nothing of this nature has been seen in computable structure theory, as all these classes are universal for Turing computability. We will discuss universality later in the paper.

Problem 3.12. *Is there an algebraic description of punctually categorical unary structures?*

It is not hard to see that a structure with only one unary functional symbol can encode an arbitrary family of sets; in particular, there is an example of a computably categorical unary structure which is not relatively computably categorical. But we however strongly suspect that, at least in the case of only one or perhaps two unary operations, there will be an algebraic description of punctual categoricity.

At this point the reader might think that all punctually categorical structures must be either finite or perhaps trivially homogeneous such as a vector space over a finite field, cf. Theorem 3.4(4). It is also natural to conjecture that punctual categoricity always implies computable categoricity, especially after we have seen that graphs fall into this pattern.

Our next goal is to outline the construction of a punctually categorical structure which is not computably categorical. Although logically there is no contradiction in the statement, the reader should agree that it just "does not sound right".

4. The punctual monster

Theorem 4.1 ([KMN17b])**.** There is a punctually categorical structure $\mathcal{A}$ which is *not* computably categorical.

We will not give the proof in full detail. We will however outline the main "pressing" strategy that will be used in some later parts of the paper. For instance, our formal proof of Theorem 7.3 which will appear later in the paper uses the same strategy.

Proof Idea. More specifically, we must have a way of meeting the requirements:

$$\mathcal{A} \cong P_e \implies \mathcal{A} \cong_{fpr} P_e,$$

where $(P_e)_{e\in\omega}$ is the natural uniformly computable listing of all punctual structures. Clearly, the list itself is not primitive recursive, for otherwise we would be able to produce a punctual structure which is not in the list.

Think of $(P_e)_{e\in\omega}$ as of being "increasingly slow in e". However, we will argue that for each fixed e there is a primitive recursive time-function, i.e., a function that bounds the speed of approximation of $P_e = \bigcup_s P_{e,s}$ within the overall uniformly primitive recursive approximation $(P_{e,s})_{e,s\in\omega}$. *For now, take this property for granted.* We delay the formal proof of this fact until §9.

4.1. **Pressing P_0.** The idea is as follows. Start by building an infinite chain using a unary function S:

$$0 \to S(0) \to S^2(0) \to S^3(0) \to \cdots,$$

and use another unary function, say U, to attach a U-loop of some fixed small size to each S^n. To be more specific, suppose we attach 2-loops. Use another unary function r that sends each point back to the origin:

$$\forall x\, r(x) = 0.$$

Do nothing else and wait for the opponent's structure P_0 to respond. (The structure will be rigid.)

The opponent's structure P_0 must give us a few 2-loops, otherwise $P_0 \ncong \mathcal{A}$. However, it is important to see *how exactly* P_0 could fail to be isomorphic to $\mathcal{A}$.

(1) The structure P_0 does not even look right; that is, it is not an S-chain etc. In this case we do nothing.
(2) Otherwise, P_0 could give us an U-loop of a wrong size, say 4. Then we will forever forbid 4 in the construction.
(3) P_0 starts growing a long simple U-chain. It is easiest to drive it to infinity in the construction, as follows. At stage s other strategies will be allowed to use only loops that are shorter than the U-chain as seen in $P_0[s]$.

Assume none of the above cases apply. Then P_0 does respond by giving us a few consequent 2-loops. *Note that, perhaps P_0 has not revealed the position of x in the S-chain.* This point could correspond to one of the 2-loops that we have produced in $\mathcal{A}$ while waiting for the slow P_0 to give us something. But one of our tasks is to demonstrate that the (unique) primitive recursive isomorphism from P_0 to $\mathcal{A}$ is primitive recursive. We must decide promptly where x must be mapped. But we cannot just keep building 2-loops in $\mathcal{A}$, as it will give the opponent an upper hand.

Instead, as soon as P_0 responds by giving a 2-loop, we switch from the pattern

$$2-2-2-2-2-2-2-2-\cdots$$

to the pattern (say)

$$2 - 4 - 2 - 4 - 2 - 4 - 2 - 4 - \cdots,$$

assuming that 4 is currently not forbidden in the construction.

How do we punctually map $x \in P_0$ (see above) to $\mathcal{A}$? Recall that x was a part of a chain of a few consequent 2-loops. In $\mathcal{A}$, the initial segment consisting of adjacent 2-loops has a specific length that we know at the stage, say k. In P_0, calculate r on x to find the origin, and then calculate S of the origin at most k times to figure the position of x.

To compute the unique isomorphism from $\mathcal{A}$ to P_0, simply start from the origin in P_0 and map $\mathcal{A}$ onto P_0 naturally, according to the speed with which P_0 is generated. We use *the primitive recursive time* which measures the speed of its enumeration (see the discussion above). In a way, this will be a non-uniformly primitive recursive proof.

4.1.1. *Pressing P_0 and P_1.* For simplicity, the highest priority structure can be pressed using loops attached to even positions in the S-chain:

$$0, S^2(0), S^4(0), \cdots, S^{2k}(0), \cdots$$

and the lower priority P_1 will be associated with odd positions of the S-chain. Also, P_0 will be using U-loops of even length, and P_1 of odd length.

There are several effects that P_0 could have on P_1. First of all, it could permanently forbid some of the loops from appearing in the structure. This is finitary. Otherwise, P_0 could start growing a long U-chain, and P_1 will be forced to use U-loops which are currently shorter than the length of the chain. But we can assume that P_0 is a lot faster than P_1, so this does not really give us trouble. There are further minor tensions, but all can be sorted using the basic standard priority technique.

We however must note that P_1 could potentially put some restrictions on the actions of P_0. More specifically, if P_1 starts growing a long U-chain late enough in the construction, and grows it long enough, then P_0 will have to live with this. Also, P_1 is "much slower" than P_0, so P_0 will have to act before P_1 either dies or increases its restraint. But this is not really a problem. Recall the informal description of the basic module for P_0 above. The difference is that the loops corresponding to P_0 are located at even positions:

$$2 - \square - 2 - \square - 2 - \square - 2 - \cdots,$$

where the content of the $\square$s does not worry the strategy. The strategy then switches to:

$$\cdots - 2 - \square - 4 - \square - 2 - \square - 4 - \cdots,$$

assuming 4 is small enough and not restrained. However, imagine all the larger loops are currently restrained, but the strategy must act. Simply use a more complex pattern of 2 and 4, such as:

$$\cdots - 2 - \square - 4 - \square - 4 - \square - 2 - \square - 4 - \square - 4 \cdots.$$

We could even get around with using *only* 2 and 4 throughout the construction, for P_0. Then the described above tension with the long U-chain can be hidden and even almost completely erased from the formal proof. It is however important to understand why exactly we often use patterns of loops instead of longer fresh loops in the construction.

Our punctual definition of the isomorphism between P_1 and $\mathcal{A}$ is essentially the same as in the description of one strategy in isolation. We only need to look at a bit larger interval in P_1 around a given point x.

In the general case of many P_e we generalise the ideas described above. We reserve specific locations for loops in $\mathcal{A}$ corresponding to different P_e. At later stages the construction will respect more of the P-structures. Some further minor tensions can be sorted using priority.

4.2. **The idea of the diagonalisation.** In the actual proof of the theorem the pressing strategy will be very similar, but the structure will no longer be just a chain of loops. More generally, it cannot be finitely generated.

It will consist of infinitely many finite (long) chains of loops, that we call *components*. The isomorphism type of the components depends on the construction, but it will strongly resemble the chain of loops described above. The difference with the strategy above is that we can actually *stop* growing the currently active chain and start building a new one.

We will also be constructing a computable $\mathcal{B} \cong \mathcal{A}$. We must meet the diagonalisation requirements:

$$\varphi_e : \mathcal{B} \not\cong \mathcal{A}.$$

We can afford to delay $\mathcal{B}$, but we cannot delay $\mathcal{A}$. This will be crucial.

Here is the outline of the diagonalisation strategy:

(1) Associate φ_e with some special and (currently) unique component C_e in both $\mathcal{A}$ and $\mathcal{B}$.
(2) Wait for φ_e to converge on C_e.
(3) Initiate the enumeration of a new, fresh and unique component Z_e in $\mathcal{A}$, according to the instructions of the pressing strategy. Label C_e in $\mathcal{A}$ using Z_e; this can be done by mapping the origin of Z_e into the origin of C_e using some special unary function.
(4) Freeze the enumeration of $\mathcal{A}$.
(5) When Z_e is finished, introduce a new and fresh component Y_e in $\mathcal{A}$ which is built according to the pressing strategy. Also, introduce an identical copy of C_e and label it using Y_e.
(6) As soon as Y_e is finished, unfreeze $\mathcal{B}$. In $\mathcal{B}$, label its current C_e by Y_e, and introduce its identical copy labeled by Z_e.

It is crucial that we do the actions in the right order. For instance, we must first put Z_e to make sure that the copy of C_e that a structure P_i currently has is the right copy. Then we must put Y_e and use it to press P_i to reveal its version of Y_e. But if P_i shows Y_e then it must also promptly show us the other, new copy of C_e. We omit further details. □

It has recently been shown that computably categorical (c.c.) structures are unclassifiable; more formally, the index set of c.c. structures is Π^1_1-complete [DKL$^+$15]. We have discovered that punctually (fpr) categorical structures do not form a proper subclass of c.c. structures. This brings us to the problem:

Problem 4.2. *Measure the complexity of the index set* $\{e : P_e$ *is punctually categorical*$\}$

The (current) lack of techniques makes the problem above difficult to approach. It may sound silly, but the authors have not agreed on the conjecture for the question below:

Question 1. *Is every punctually categorical structure (relatively) Δ^0_α-categorical for some computable α? (The definition of (relative) Δ^0_α-categoricity can be found in* [AK00]*.)*

Why do we care? The combinatorial games that we play when we study such questions are in the heart of our "online" framework. The universe shows us a pattern and we must recognise it *now*. Punctual categoricity serves as a unifying abstraction for such games, while (relative) Δ^0_α-categoricity is a typical way to measure the complexity of a structure in computable structure theory. One naturally seeks to understand the relationship between the two measures of complexity. In the next section we will further develop these ideas, but this time using back-and-forth analysis rather than categoricity.

One might hope to push the ideas described in this section to a construction of a non-Δ^0_2-categorical but punctually categorical example. However, it seems to necessarily require a new idea. Downey, Melnikov, and Ng have recently announced that α, if it exists, must be > 2.

5. An online back-and-forth invariant

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 5.1. Let A be a punctual structure. Then, for punctual C, B isomorphic to A,

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

This leads to an equivalence relation $\cong_{fpr}$ and the degree structure on the classes which will be denoted $\mathbf{FPR}(A)$.

What does $\mathbf{FPR}(A)$ reflect? If $C \leq_{pr} B$ then, in a way, B has more online content than C does, in the sense that more things happen in B. For example, the standard copy of $(\mathbb{Q}, <)$ punctually embeds any other punctual copy of the rationals; it has a prompt Skolem function, but some other copies may have slow intervals. Also, the standard copy of (ω, S) can be punctually embedded into any other copy; the other copies of (ω, S) will contain points that look non-standard ("infinite", "disconnected") for a very long time, but no such points can be found in the standard copy. In other words, $A \leq_{pr} B$ means that B enumerates itself more impatiently.

This is a new concept unseen in effective mathematics. Thus, we would like to pick some specific algebraic structure and try to understand its FPR-degrees in full depth. We will do so for the dense linear order $(\mathbb{Q}, <)$. Perhaps, the main outcome of our investigation into this direction is that *our proofs will be unexpectedly non-trivial* even for this algebraically elementary structure, and we strongly conjecture that this complexity is unavoidable. Even though we gave it a good thought, we still know very little about the algebraic structure of $\mathbf{FPR}(\mathbb{Q}, <)$, and this investigation is definitely of some technical interest; see § 6 for more details.

We could also take two punctual structures $\mathcal{A}$ and $\mathcal{B}$ and compare $\mathbf{FPR}(\mathcal{A})$ and $\mathbf{FPR}(\mathcal{B})$. If $\mathbf{FPR}(\mathcal{A})$ and $\mathbf{FPR}(\mathcal{B})$ are similar or isomorphic, then the "online content" of the two structures is the same. Also, we could fix some class (or a property) of structures and see whether there is some common feature shared between all $\mathbf{FPR}(\mathcal{A})$ when $\mathcal{A}$ ranges over the class (or over structures having this property, respectively). See § 7 for results and open problems into this direction.

We will also see that there is a non-trivial connection between FPR-degrees and the notion of punctual categoricity from the previous section; we discuss this in the paragraph below.

5.1. **Does back and forth imply back-and-forth?** Note that if $\mathbf{FPR}(A)$ contains a single degree this means that for any two punctual copies C, B of A there is a pair of primitive recursive isomorphisms, one going from C onto B, and the other mapping B onto C. *Note that it is not obvious at all that there must be a primitive recursive isomorphism $f : B \to C$ with primitive recursive inverse.*

Question 2. For a punctual A, is $|\mathbf{FPR}(A)| = 1$ equivalent to saying that A is punctually categorical?

The question is open in general, but it can be answered in positive for many standard classes including Boolean algebras, linear orders, and, most notably, graphs [MN] via a rather non-trivial proof.

Theorem 5.2. Suppose $\mathcal{G}$ is a (punctual) graph. Then $|\mathbf{FPR}(\mathcal{G})| = 1$ iff $\mathcal{G}$ is punctually categorical.

Informal discussion. The strategies from the proof of Theorem 3.5 that describes punctually categorical graphs seem to be of little help. Some of the strategies there relied heavily on the isomorphism being fully primitive recursive (i.e., with primitive recursive inverse). In particular, Prop. 3.6 heavily relied on the fact that, for $p : B \to A$, $qp(x') = x'$, where q is the intended inverse of p.

We need a new notion.

(1) $|\mathbf{FPR}(\mathcal{G})| = 1$.
(2) Given any two f.p.r. copies $\mathcal{A} \cong \mathcal{B}$ of $\mathcal{G}$, there exist primitive recursive isomorphisms $f : \mathcal{A} \mapsto \mathcal{B}$ and $g : \mathcal{B} \mapsto \mathcal{A}$, and a primitive recursive function $t : \mathbb{N} \mapsto \mathbb{N}$ such that given $a \in \mathcal{A}$, either $Orb(a) = \{(gf)^n(a) : n \in \omega\}$ has size at most $t(a)$, or every permutation u of $Orb(a)$ can be extended to an automorphism of $\mathcal{G}$.

Note that given (2) we can run a primitive recursive back-and-forth construction to produce a fpr isomorphism between two punctual copies. First, check whether $Orb(a)$ has size $\leq t(a)$. If "yes" then match $Orb(a)$ with $Orb(f(a))$. Otherwise, if $Orb(a)$ has not yet closed after $t(a)$ steps, then do the back-and-forth on $Orb(a)$ and $Orb(f(a))$ essentially ignoring the rest of the structure. Unfortunately, the implication (1) $\to$ (2) is quite non-trivial and will not be presented here. □

5.2. **FPR-degrees as partial orders.** We could approach FPR-degrees from a different perspective. Instead of looking at FPR-degrees of some familiar structures, we could attack the general problem below.

Problem 5.3. *Is there a convenient description of partial orders that can be realised as the FPR-degrees of a (computably categorical) punctual structure?*

See [KMnN17] for several (quite basic) results into this direction. It is well-known that there exist (Turing) computable algebraic structures that have exactly 2 computable copies up to computable isomorphism.

Question 3. *Is there a structure $\mathcal{A}$ such that $1 < |\mathbf{FPR}(\mathcal{A})| < \infty$?*

Melnikov and Ng have recently announced:

Theorem 5.4. *There is a structure with exactly two punctual presentations, up to fpr isomorphism.*

The proof is rather technical, but the authors are currently convinced it works. However, the proof is so complex that at the moment the above theorem should be viewed as a (strong) conjecture. We note that the proof does not even resemble the dimension 2 proof in computable structure theory.

Question 4. *Is there a structure $\mathcal{A}$ such that $|\mathbf{FPR}(\mathcal{A})| > 1$ and $\mathbf{FPR}(\mathcal{A})$ is a linear order under $\leq_{pr}$?*

More generally, we know little about the possible algebraic types of the order $\mathbf{FPR}(\mathcal{A})$, but several basic results can be found in [KMN17a]. We will return to this question in Section 7, where we will refute the conjecture that the FPR-degrees of a finitely generated structure cannot have the greatest element unless the structure is punctually categorical.

6. Homogeneous structures

Recall that a structure $\mathcal{X}$ is homogeneous if every isomorphism $f : F_1 \to F_2$ between any two finitely generated substructures $F_1, F_2 \subseteq \mathcal{X}$ is extendable to an automorphism of $\mathcal{X}$. (Such structures are also called ultrahomogeneous in the literature.) See [Mac11] for a survey on homogeneous structures.

Example 6.1. The following structures are homogeneous:

- $(\mathbb{Q}, <)$, the dense linear order without end-points.
- $\mathcal{R}$, the Random Graph.
- $\mathcal{P} \cong \bigoplus_{i \in \omega} \mathbb{Z}_{p^\infty}$, the universal countable abelian p-group (the infinite direct power of the Prüfer group $\mathbb{Z}_{p^\infty}$).

Each structure in Ex. 6.1 is the Fraïsse limit of finite structures within the respective class. Also, they do share essentially the same back-and-forth proof of their uniqueness up to isomorphism. More specifically, we pick another element and wait for a suitable element on one side, and then we switch sides. The back-and-forth proofs for these structures are identical from the general Turing computability point of view. In particular, *all these proofs are Turing computable* and *all these structures are computably categorical.* Furthermore, there is only one instance of a potentially unbounded search involved in this algorithm. It is natural to conjecture that these proofs are also the same from the standpoint of primitive recursion.

Remarkably, the recent result of Melnikov and Ng [MN] below shows that the back-and-forth proofs for these three structures differ from the perspective of primitive recursion.

Theorem 6.2. The FPR-degree structures of the dense linear order $(\mathbb{Q}, <)$, the random graph $\mathcal{R}$, and the universal divisible abelian p-group $\mathcal{P}$ are pairwise non-isomorphic.

Proof idea. The proof is essentially degree-theoretic in nature. First, we establish that $\mathbf{FPR}(\mathcal{R})$ and $\mathbf{FPR}(\mathcal{P})$ have no greatest element, while $\mathbf{FPR}(\eta)$ does. Also, $\mathbf{FPR}(\mathcal{P})$ has no maximal elements, while $\mathbf{FPR}(\mathcal{R})$ does. Some of these facts require a non-trivial proof. □

We note that Alaev [Ala16] has recently investigated the primitive recursive content of the countable atomless Boolean algebra β. (We note that the use of polynomial time presentations of β in [Ala16] does not really make much difference.) We conjecture that $\mathbf{FPR}(\beta) \cong \mathbf{FPR}(\eta)$, but establishing such an isomorphism could be quite tricky (if it exists); the most naive attempt via the interval algebra presentation seems to fail.

Question 5. *Is* $\mathbf{FPR}(\mathbb{Q}, <) \cong \mathbf{FPR}(\beta)$*, where* β *is the atomless Boolean algebra?*

The reader may find Theorem 6.2 counterintuitive; the authors definitely do. In fact, the theorem disproves our initial conjecture that the FPR-degrees of these structures should be isomorphic. *The discovery of Theorem 6.2 led us to the conclusion that we do not know enough about the FPR-degrees of even the algebraically simplest structures.* It makes sense to pick one familiar and algebraically simple structure and try to understand its FPR-degrees in full depth. The hope is that some of the ideas and techniques can then be applied to some other, perhaps more algebraically interesting, structures.

The dense linear order $(\mathbb{Q}, <)$ is the standard (and perhaps the simplest natural) example when a back-and-forth proof works, it makes sense to investigate $\mathbf{FPR}(\mathbb{Q}, <)$ in some depth, hoping that some of the ideas and techniques will be useful for the general theory. Remarkably, proving the theorem below takes some effort.

Theorem 6.3. [MN] $\mathbf{FPR}(\mathbb{Q}, <)$ is downwards dense.

The proof of the above theorem is non-uniform and quite combinatorially involved; we omit details.

Question 6. *Is* $\mathbf{FPR}(\mathbb{Q}, <)$ *dense?*

Melnikov and Ng have recently conjectured that $\mathbf{FPR}(\mathbb{Q}, <)$ is upwards dense, but their proof contained a serious problem.

We do not know much about the FPR-degrees of common natural structures (such as the atomless Boolean algebra) beyond the results discussed in this section. Such investigations will be of technical interest and will hopefully shed some light on the nature of punctual back-and-forth proofs.

Problem 6.4. Investigate into the FPR-degrees of some elementary natural computably categorical algebraic structures such as:

(1) the dense linear order $(\mathbb{Q}, <)$;
(2) the atomless Boolean algebra (compare with the dense linear order);
(3) (ω, S), $(\mathbb{N}, +)$, and $(\mathbb{N}, +, \times)$ (do they all have isomorphic FPR-degree structures?);
(4) finitely generated abelian groups;
(5) algebraically closed fields of finite transcendence degree (is $\mathbf{FPR}(\overline{\mathbb{Z}_p}) \cong \mathbf{FPR}(\overline{\mathbb{Q}})$, where $\overline{F}$ stands for the algebraic closure of F?);

Even for such algebraically simple classes as above, the raised questions may turn to be very challenging. It seems that the FPR-degrees is an extremely sensitive *computability-theoretic* invariant that can be used to draw conclusions about the *algebraic* properties of the structure, especially in some natural classes. This is some work to do in the future.

7. Finitely generated structures

We have discussed homogeneous structures. Within (relatively) computably categorical structures, the finitely generated structures are the opposite extreme: they become rigid after fixing a finite tuple of generators. One would expect the results on such structures to be somewhat dual to those for the homogeneous ones. However, most of the results discussed in this section were unexpected.

We concentrate on finitely generated (f.g.) structures in a (finite) functional language. Clearly, every such structure is computably categorical in the sense of (Turing) computable structure theory. If A is punctual and f.g. then $\mathbf{FPR}(A)$ has the least degree which is the "naturally generated" term algebra built around the finitely many generators of A. Even the simplest f.g. structures, such as (ω, S), will typically have a pathological copy in which some elements will be kept "disconnected" from the generators long enough to allow for a diagonalisation against primitive recursive isomorphisms.

Example 7.1. $\mathbf{FPR}(\omega, S)$ has no maximal elements and is dense. The strategy described in Example 3.1 can be used to construct a copy strictly $\leq_{pr}$-above any given copy. Density will follow from a more general result below.

One might hope for a general fact that would generalise the example above to all f.g. structures, but this will not work. In our proof sketch of Theorem 4.1 we outlined the construction of a 1-generated punctually categorical infinite rigid structure (this is Prop. 4.2 in [KMN17b]). Thus, we can have $|\mathbf{FPR}(A)| = 1$ for an infinite f.g. A.

Recall that it is open whether $|\mathbf{FPR}(A)| = 1$ implies punctual categoricity of A, but we know it does for graphs. It is not hard to see that, for a f.g. A, $|\mathbf{FPR}(A)| = 1$ is equivalent to punctual categoricity of A [BKMN]; we omit details.

Suppose a f.g. A is not punctually categorical. Can we have $1 < |\mathbf{FPR}(A)| < \infty$?

Theorem 7.2. [BKMN] *Suppose a f.g. A is punctual and $|\mathbf{FPR}(A)| > 1$. Then $\mathbf{FPR}(A)$ is countably infinite and dense.*

Proof. Let $B <_{pr} T$ be two punctual presentations of A. We build a punctual copy X with the property:

$$B <_{pr} X <_{pr} T.$$

The first idea is to switch between copying B and T. The second idea is to use some fixed tuple of generators $\bar{g}$ in both B and T to exclude the unpleasant scenario in which a potential isomorphism from T to X (or from X to B) is not onto. This will allow us to keep the strategies strictly finitary. We will not attempt to diagonalise against, say, $p_e : X \to B$ until we see that $p_e(\bar{g})$ generates the version of $\bar{g}$ in B. Note that if p_e is an onto isomorphism then the p_e-image of the generators of X must span the generators of B. Suppose $p_e : X \to B$ is ready for diagonalisation in the sense above, but X is currently copying B. We can punctually switch to

X copying T by identifying the natural image of B within T (given by $B <_{pr} T$) with X. Then we wait for a *finitary* local disagreement confirming $p_e : X \not\to B$, and then we can switch back to copying B, if necessary. The latter is done by ceasing all actions in X except for evaluating the functional symbols on the already enumerated elements and waiting for (the natural image of) B (in X) to catch up on the extra elements that we may have adjoined to X while copying T. As soon as this happens, we are in the position to diagonalise against some $q_j : T \to X$ which is ready for diagonalisation (in the sense above), etc. The rest is sorted using priority. □

Perhaps, one could strengthen the theorem above and show that $|\mathbf{FPR}(A)|$ are upwards dense for every f.g. structure A (unless $|\mathbf{FPR}(A)| = 1$). Rather surprisingly, this natural hypothesis fails.

Theorem 7.3. [BKMN] *There exists a f.g. A such that $|\mathbf{FPR}(A)| > 1$ and $\mathbf{FPR}(A)$ has a greatest element.*

Idea. Combine the pressing technique described in Section 4 with an (ω, S)-style diagonalisation. (In (ω, S), we would keep some element disconnected from the origin for a long time before we connect it back to 0.)

To build the "top" copy T, we keep a part of the long chain of loops disconnected from the origin. The key idea here is that we could still build a primitive recursive $f : P \to T$ from a given copy, because the pressing strategy still works if we introduce another unary function that sends a point not to the origin but to the beginning of the currently disconnected island. The exact formal details are unpleasant, but there are no surprises in the proof. □

8. Graphs and universality

It is natural to ask whether any structure can be effectively coded into a structure from some specific class preserving all primitive recursive properties of interest, such as punctual categoricity, the isomorphism type of the FPR-degrees, or some other property.

Theorem 8.1. [DHTK$^+$] *The class of structures with only one binary functional symbol is punctually universal.*

We have not formally defined universality, let alone punctual universality. To state the result formally we need a few elementary definitions. For a total function $f : \omega \to \omega$, let $P(f)$ be the least class containing f and all primitive recursive functions closed under composition and primitive recursion. This is done by forbidding the (unbounded) minimisation operator and adding f to the recursive schemata. Similarly, we can define the notion of a primitive recursive functional.

Formally, Theorem 8.1 says: *There exists a primitive recursive functional Γ which, given a structure in a finite signature $\mathcal{A}$ outputs a structure $\Gamma(\mathcal{A})$ in the language of one binary functional symbol such that:*

1. *if $F : \mathcal{A} \to \mathcal{B}$ is an isomorphism then there is an isomorphism $G : \Gamma(\mathcal{A}) \to \Gamma(\mathcal{B})$ for which we have $G \in P(F)$ and $G^{-1} \in P(F^{-1})$;*
2. *if $G : \Gamma(\mathcal{A}) \to \Gamma(\mathcal{B})$ is an isomorphism then there is an isomorphism $F : \mathcal{A} \to \mathcal{B}$ for which we have $F \in P(G)$ and $F^{-1} \in P(G^{-1})$;*
3. *if $\mathcal{U} \cong \Gamma(\mathcal{A})$ then there is a structure $\mathcal{B} \cong \mathcal{A}$ in $P(\mathcal{U})$ and an isomorphism $H : \Gamma(\mathcal{B}) \to \mathcal{U}$ with both H and H^{-1} in $P(U, \mathcal{A})$.*

Furthermore, G and H and their inverses can also be witnessed by primitive recursive functionals with respective inputs.

There is a very strong resemblance between the formal statement above and the general definition of Turing computable universality of a class in [HTMMM17]. The only essential difference is that we use primitive recursive functionals instead of Turing functionals. *Thus, in particular, one should expect every punctually universal class to be Turing universal as well.*

Proof sketch of Thm 8.1. First of all, replace all predicate symbols in $\mathcal{A}$ by functional symbols by adding two extra constants, 0 and 1, and replacing each of the finitely many predicate by a function that map tuples to these constants. Hence, we can assume from the beginning that $\mathcal{A}$ has only functional symbols. We claim that, w.l.o.g., the functions may have different arity. For example, replace $f(x_1, \dots, x_n)$ by $f'(x_1, \dots, x_n, x_{n+1}) = f(x_1, \dots, x_n)$. Suppose the maximal arity among the finitely many functions is m, and the functional symbol of the arity k, $1 \leq k \leq m$, is f_k.

Under these assumptions let $\Gamma(\mathcal{A})$ be the structure with the domain

$$\{[x_1, x_2, \dots, x_k] : 1 \leq k \leq m \ \& \ x_1, x_2, \dots, x_k \in \mathcal{A}\}$$

containing all nonempty strings of the length not greater than m in the alphabet $\mathcal{A}$. The structure $\Gamma(\mathcal{A})$ has the binary symbol for truncated concatenation

$$[x_1, x_2, \dots, x_i] * [y_1, y_2, \dots, y_j] = [x_1, x_2, \dots, x_i, y_1, y_2, \dots, y_{\min(m-i,j)}]$$

and the unary functional symbols

$$g_0([x_1, x_2, \dots, x_k]) = [f_k(x_1, x_2, \dots, x_k)],$$

$$g_i([x_1, x_2, \dots, x_k]) = \begin{cases} [x_i], & \text{if } i \leq k; \\ [x_1], & \text{otherwise.} \end{cases}$$

for $1 \leq i \leq m$.

We will identify each element $x \in \mathcal{A}$ with the one-element string $[x] \in \Gamma(\mathcal{A})$. Then the structure $\mathcal{A}$ is an automorphism base of $\Gamma(A)$ (due to $[x_1, \dots, x_k] = [x_1] * \dots * [x_k]$) which is furthermore definable in $\Gamma(A)$ as the range of all the g_i, $1 \leq k \leq m$. Observe that $\mathcal{A}$ can be punctually interpreted in $\Gamma(\mathcal{A})$ via $f_k(x_1, \dots, x_k) = g_0(x_1 * \dots * x_k)$. The properties (1), (2), and (3) hold for $\Gamma(A)$; we omit details. It remains to emulate the unary functions g_i, $0 \leq i \leq m$, using only one binary function $*$ defined above; this can be done by further enriching the structure by new elements. To do so we add into $\Gamma(A)$ new elements

$$c_0, c_1, \dots, c_m$$

such that

$$c_i * c_j = c_{\min(m, i+j+1)} \text{ for } 0 \leq i, j \leq m,$$

$$x * c_i = c_0, \text{ for } 0 \leq i \leq m, x \neq c_0, c_1, \dots, c_m,$$

$$c_i * x = g_i(x), \text{ for } 0 \leq i \leq m, x \neq c_0, c_1, \dots, c_m.$$

It is clear that each of the new elements $c_0, c_1, \dots, c_m$ can be quickly reconstructed from any copy of the structure; for instance, c_0 is the unique element such that $x * c_0 = c_0$ for at least $m + 2$ different x; $c_1 = c_0 * c_0$; $c_2 = c_1 * c_0$; etc. The verification of (1)-(3) is routine. □

It is a piece of standard knowledge that graphs are universal for Turing computable structures. In contrast, we have:

Graphs are not universal among punctual structures.

Since we have not defined what punctual universality means exactly, we cannot really claim that the above statement is a theorem. However, we claim that graphs cannot be universal for any *reasonable* punctual universality notion; we explain why. Theorem 3.5 says that a punctually categorical graph becomes a clique or an anti-clique after removing finitely many vertices. In particular, every punctually categorical graph is computably categorical. We also know that there exist punctually categorical structures that are not computably categorical, see Theorem 4.1. As we noted above, any reasonable notion of punctual universality must also be a notion of (Turing) computable universality; in particular, it should preserve computable categoricity.

If the reader is not convinced with the above "universality test", here is another one which is based on definability rather than functionals. The construction of Kalimullin, Melnikov and Ng [KMN17b] allows for a coding of a non-computable (c.e.) set into the existential diagram of the structure, and we conjecture that this likely goes all the way up to Σ^c_α for any computable α. That is, we believe that for each computable ordinal α there is a punctually categorical structure which is not Δ^0_α-categorical. Informally, this says that punctually categorical structures can be as far from being computably categorical as it seem possible for them to be. Thus, in a punctually universal class, some punctually categorical representatives must also be highly non-trivial. Graphs however definitely fail this test.

We strongly suspect that our techniques can be extended to a description of punctually categorical directed graphs, thus likely showing that they are not universal.

Problem 8.2. *Is there a punctually universal structure in a predicate language?*

It would be interesting to understand the special case of finitely many binary relations. As we noted at the end of Section 3, these two classes may likely have a nice explicit algebraic description of punctual categoricity.

8.1. **Sub-recursive relativisation.** Note that the formal statement of the universality result for binary functions relies on a certain notion of *subrecursive relativisation*. That is, we can relativise a primitive recursive process by adding a total function f to the primitive recursive schemata; it is perhaps more natural to assume the function f is computable. Informally, this means that we allow the minimisation operator with the bound given by f; and we view everything computed within the time bounds of f as quick enough. We could also define the notion of primitive recursion relative to a class of functions or consider an arbitrary class of functions closed under primitive recursive operators and composition.

Remarkably, most (if not all) of the results surveyed in the article will hold relative to an arbitrary total function f or relative to a class of total computable functions. Sometimes we do not even have to assume that the function (or the class) is computable, all we need is totality. For instance, Theorem 8.1 above is an example of one such result that works in the most general setting of total functions.

Problem 8.3. *Develop a systematic theory of structures punctual relative to a total oracle. Study subrecursive hierarchies of structures.*

So far, the only result into this direction is the theorem below.

Theorem 8.4. [KMnN17] For every $n > 0$ there exists a fully primitive recursive structure which is punctually $0^{(n)}_{PR}$-categorical but not punctually $0^{(n-1)}_{PR}$-categorical.

In the theorem above, $0'_{PR}$ stands for the primitive recursive jump; this is the total function that naturally enumerates all primitive recursive functions: $f(n, x) = p_n(x)$. This process can be naturally iterated to define $0^{(n-1)}_{PR}$; see [KMnN17] for details. The notion of punctual $0^{(n)}_{PR}$-categoricity should be self-explanatory. The proof of the theorem is basically a (punctual) coding of a total function into the diagram of a structure, so it works not only specifically for the primitive recursive jumps; see [KMnN17].

It is widely believed that primitive recursion serves as an adequate model for finitism in proof theory; see, e.g., Tait [Tai81].

Problem 8.5. *Investigate the proof-theoretic content of punctual structures, perhaps in relation with reverse mathematics.*

9. Appendix: Primitive recursive time

We clarify the use of (subrecursively) non-uniform arguments in our proofs. In some proof sketches above we used that a certain process was primitive recursive relative to some primitive recursive function in the total enumeration of all primitive recursive functions. Although the intuition behind this trick is fairly straightforward, it does require a careful thought and a rigorous verification. We intend to use the content of this appendix as a standard reference for our upcoming papers on the subject.

Let $(\varphi_e)_{e\in\omega}$ be the computable listing of all Turing computable functions viewed as general recursive schemata. There is a uniformly computable sub-listing $(\varphi_{l(e)})_{e\in\omega}$ that consists of all primitive recursive schemata. Write p_e for $\varphi_{l(e)}$. There also is a uniformly primitive recursive approximation $(\varphi_{e,s})_{e,s\in\omega}$ where (essentially) $\varphi_{e,s} = \varphi_e[s]$.

More formally, $f(\vec{x})[s]$ is a primitive recursive *approximation* to a computable function $f(\vec{x})$ if

(1) if $f(\vec{x})[s]\downarrow$ then $f(\vec{x})[s] = f(x) \le s$ and $f(\vec{x})[s+1]\downarrow$;
(2) for every $\vec{x}$ there is a stage s such that $f(\vec{x})[s]\downarrow$;
(3) the function

$$f'(\vec{x}, s) = \begin{cases} -1, & \text{if } f(\vec{x})[s]\uparrow; \\ f(\vec{x})[s], & \text{if } f(\vec{x})[s]\downarrow; \end{cases}$$

is primitive recursive.

The function $t(\vec{x}) = \min\{s : f(\vec{x})[s]\downarrow\}$ is called *the time function* for a total computable $f(\vec{x})$.

Lemma 9.1. There is a primitive recursive function $i(n)$ such that for every n the function $p_{i(n)} = \varphi_{l(i(n))}$ is the time function for $p_n = \varphi_{l(n)}$ with respect to the uniformly primitive recursive approximation $(\varphi_{n,s})_{n,s\in\omega}$ as defined above.

In particular, *a function f is primitive recursive iff it can be emulated on the universal Turing machine with a primitive recursive time-bound on the steps of approximation*. Furthermore, the index of the primitive recursive time function for f can be found uniformly and primitively recursively in any given index of f. We advise to skip the technical and notationally heavy proof of the lemma below at the first reading.

Proof. We will prove a more general result which can be used to define the primitive recursive jump and "relativize" results in the subrecursive hierarchy. It is also quite instructive because it clarifies what it means to be primitive recursive relative to some total computable function. Its proof, however, is not really any harder than the proof of the stated less general lemma.

For a total function $f(\vec{x})$ define the f-primitive recursive schemas p^f by induction:

(1) The functions $o(x) = x, s(x) = x + 1$, $I^n_m(x_1, \dots, x_n) = x_m$ and $f(\vec{x})$ are f-primitive recursive schemas.
(2) (Composition). If $g^k_1(\vec{x})$, $0 \le k \le n$, and $g_2(y_0, \dots, y_n)$ are f-primitive recursive schemas then the function $g_3(\vec{x})$:
$$g_3(\vec{x}) = g_2(g^0_1(\vec{x}), \dots, g^n_1(\vec{x}))$$
is a f-primitive recursive schema.
(3) (Primitive recursion). If $g_1(\vec{x})$ and $g_2(\vec{x}, y, z)$ are f-primitive recursive schemas then the function $g_3(\vec{x}, y)$:
$$g_3(\vec{x}, 0) = g_1(\vec{x}) \text{ and } g_3(\vec{x}, y + 1) = g_2(\vec{x}, y, g_3(\vec{x}, y))$$
is a f-primitive recursive schema.

Let $f(\vec{x})[s]$ be a primitive recursive approximation for $f(\vec{x})$ with the corresponding time function $t(\vec{x})$. Define inductively the primitive recursive approximation $p^f[s]$ for each f-primitively recursive schema p^f:

(1) The function $f(\vec{x})$ already has the primitive recursive approximation $f(\vec{x})[s]$. The primitive recursive approximation for basic primitive recursive functions $o(x)$, $s(x)$ and $I^n_m(\vec{x})$ are defined as
$$t(\vec{x})[s]\downarrow \iff (\exists y \le s)[(\vec{x}, y) \in \text{ graph } u],$$
where u is one of these basic functions.
(2) If $g^k_1(\vec{x})[s]$, $0 \le k \le n$, and $g_2(y_0, \dots, y_n)[s]$ are primitive recursive approximations to $g^k_1(\vec{x})$, $0 \le k \le n$, and $g_2(y_0, \dots, y_n)$ with the corresponding time functions $t^k_1(\vec{x})$, $0 \le k \le n$, and $t_2(y_0, \dots, y_n)$, respectively, then for the function $g_3(\vec{x})$:
$$g_3(\vec{x}) = g_2(g^0_1(\vec{x}), \dots, g^n_1(\vec{x}))$$
we define the primitive recursive approximation $g_3(\vec{x})[s]$ such that
$$g_3(\vec{x})[s]\downarrow \iff (\forall k \le n)[g^k_1(\vec{x})[s]\downarrow] \ \&\ g_2(g^0_1(\vec{x}), \dots, g^n_1(\vec{x}))[s]\downarrow.$$
It is easy to see that the function
$$t_3(\vec{x}) = \max(\{t^k_1(\vec{x}) : 0 \le k \le n\} \cup \{t_2(g^0_1(\vec{x}), \dots, g^n_1(\vec{x}))\})$$
is the corresponding time function for $g_3(\vec{x})$.

(3) If $g_1(\vec{x})[s]$ and $g_2(\vec{x}, y, z)[s]$ are primitive recursive approximations to $g_1(\vec{x})$ and $g_2(\vec{x}, y, z)$ with the corresponding time functions $t_1(\vec{x})$ and $t_2(\vec{x}, y, z)$, respectively, then for the function $g_3(\vec{x}, y)$:

$$g_3(\vec{x}, 0) = g_1(\vec{x}) \text{ and } g_3(\vec{x}, y+1) = g_2(\vec{x}, y, g_3(\vec{x}, y))$$

we define the primitive recursive approximation $g_3(\vec{x}, y)[s]$ such that

$$g_3(\vec{x}, y)[s]\downarrow \iff (\exists u_0, \ldots, u_y \leq s)[g_1(\vec{x})[s]\downarrow = u_0 \ \&\ (\forall i < y)[g_2(\vec{x}, i, u_i)[s]\downarrow = u_{i+1}]].$$

It is easy to see that the function

$$t_3(\vec{x}, y) = \max(\{t_1(\vec{x})\} \cup \{t_2(\vec{x}, i, g_3(\vec{x}, i)) : i < y\}).$$

is the corresponding time function for $g_3(\vec{x}, \vec{z})$.

It follows from the inductive definitions above that we have proven more: *There is a primitive recursive function $i(n)$ such that for every n the function $\Phi^t_{i(n)}(x)$ is the time function for $p^f_n(x)$ with respect to the uniformly primitive recursive approximation $p^f_n(x)[s]$.* □

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.

[Ala16] P. E. Alaev. Atomless Boolean algebras computable in polynomial time. *Siberian Electronic Mathematical Reports*, 13:1035–1039, 2016.

[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.

[BHTK+] N. Bazhenov, M. Harrison-Trainor, I. Kalimullin, A. Melnikov, and K. M. Ng. Automatic and polynomial-time algebraic structures. Preprint.

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

[Bor09] É. Borel. Les probabilités dénombrables et leurs applications arithmétiques. *Rend. Circ. Mat. Palermo*, 27(1):247–271, 1909.

[CDRU09] Douglas Cenzer, Rodney G. Downey, Jeffrey B. Remmel, and Zia Uddin. Space complexity of abelian groups. *Arch. Math. Log.*, 48(1):115–140, 2009.

[Chu40] A. Church. On the concept of a random sequence. *Bull. Am. Math. Soc.*, 46(2):130–135, 1940.

[CR91] Douglas A. Cenzer and Jeffrey B. Remmel. Polynomial-time versus recursive models. *Ann. Pure Appl. Logic*, 54(1):17–58, 1991.

[CR92] Douglas A. Cenzer and Jeffrey B. Remmel. Polynomial-time abelian groups. *Ann. Pure Appl. Logic*, 56(1-3):313–363, 1992.

[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.

[Ded60] R. Dedekind. *Was sind und was sollen die Zahlen? 8te unveränderte Aufl.* Friedr. Vieweg & Sohn, Braunschweig, 1960.

[Deh11] M. Dehn. Über unendliche diskontinuierliche Gruppen. *Math. Ann.*, 71(1):116–144, 1911.

[DHTK+] R. Downey, M. Harrison-Trainor, I. Kalimullin, A. Melnikov, and D. Turetsky. Graphs are not universal for online computablility. Preprint.

[DKL+15] Rodney G. Downey, Asher M. Kach, Steffen Lempp, Andrew E. M. Lewis-Pye, Antonio Montalbán, and Daniel D. Turetsky. The complexity of computable categoricity. *Adv. Math.*, 268:423–466, 2015.

[DM04] R. G. Downey and C. McCartin. Some new directions and questions in parameterized complexity. In C. S. Calude, E. Calude, and M. J. Dinneen, editors, *Developments in Language Theory*, volume 3340 of *Lect. Notes Comput. Sci.*, pages 12–26. Springer, Berlin, 2004.

[Dob83] V. P. Dobritsa. Some constructivizations of abelian groups. *Sib. Math. J.*, 24(2):167–173, 1983.

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

[Dow17] R. Downey. Turing and randomness. In B. J. Copeland, J. P. Bowen, M. Sprevak, and R. Wilson, editors, *The Turing guide*, pages 427–436. Oxford University Press, Oxford, 2017.

[ECH+92] David B. A. Epstein, James W. Cannon, Derek F. Holt, Silvio V. F. Levy, Michael S. Paterson, and William P. Thurston. *Word processing in groups*. Jones and Bartlett Publishers, Boston, MA, 1992.

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

[FS56] A. Fröhlich and J. Shepherdson. Effective procedures in field theory. *Philos. Trans. Roy. Soc. London. Ser. A.*, 248:407–432, 1956.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and intractability*. W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[GN73] S. S. Goncharov and A. T. Nurtazin. Constructive models of complete solvable theories. *Algebra Logic*, 12(2):67–77, 1973.

[Gon97] S. Goncharov. *Countable Boolean algebras and decidability*. Siberian School of Algebra and Logic. Consultants Bureau, New York, 1997.

[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.

[Har74] L. Harrington. Recursively presentable prime models. *J. Symb. Log.*, 39:305–309, 1974.

[Har98] V. S. Harizanov. Pure computable model theory. 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 3–114. North-Holland, Amsterdam, 1998.

[Her26] Grete Hermann. Die Frage der endlich vielen Schritte in der Theorie der Polynomideale. *Math. Ann.*, 95(1):736–788, 1926.

[HTMMM17] Matthew Harrison-Trainor, Alexander Melnikov, Russell Miller, and Antonio Montalbán. Computable functors and effective interpretability. *J. Symb. Log.*, 82(1):77–97, 2017.

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

[Kie98a] H. A. Kierstead. On line coloring k-colorable graphs. *Israel J. Math.*, 105(1):93–104, 1998.

[Kie98b] H. A. Kierstead. Recursive and on-line graph coloring. In Yu. L. Ershov, S. S. Goncharov, A. Nerode, and J. B. Remmel, editors, *Handbook of recursive mathematics, Vol. 2*, volume 139 of *Stud. Logic Found. Math.*, pages 1233–1269. North-Holland, Amsterdam, 1998.

[KKM14] O. Kharlampovich, B. Khoussainov, and A. Miasnikov. From automatic structures to automatic groups. *Groups Geom. Dyn.*, 8:157–198, 2014.

[KMN17a] I. Sh. Kalimullin, A. G. Melnikov, and K. M. Ng. The diversity of categoricity without delay. *Algebra Logic*, 56(2):171–177, 2017.

[KMN17b] Iskander Kalimullin, Alexander Melnikov, and Keng Meng Ng. Algebraic structures computable without delay. *Theoret. Comput. Sci.*, 674:73–98, 2017.

[KMnN17] I. Sh. Kalimullin, A. G. Mel′ nikov, and K. M. Ng. Different versions of categoricity without delays. *Algebra Logika*, 56(2):256–266, 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.

[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.

[Mac11] Dugald Macpherson. A survey of homogeneous structures. *Discrete Math.*, 311(15):1599–1634, 2011.

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

[Mal62] A. Mal′cev. On recursive Abelian groups. *Dokl. Akad. Nauk SSSR*, 146:1009–1012, 1962.

[Mel17] Alexander G. Melnikov. Eliminating unbounded search in computable algebra. In *Unveiling dynamics and complexity*, volume 10307 of *Lecture Notes in Comput. Sci.*, pages 77–87. Springer, Cham, 2017.

[Mil83] Terrence Millar. Omitting types, type spectrums, and decidability. *J. Symbolic Logic*, 48(1):171–181, 1983.

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

[MN79] G. Metakides and A. Nerode. Effective content of field theory. *Ann. Math. Logic*, 17(3):289–320, 1979.

[MN82] G. Metakides and A. Nerode. The introduction of nonrecursive methods into mathematics. In *The L. E. J. Brouwer Centenary Symposium (Noordwijkerhout, 1981)*, volume 110 of *Stud. Logic Found. Math.*, pages 319–335. North-Holland, Amsterdam, 1982.

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

[Tai81] W.W. Tait. Finitism. *The Journal of Philosophy*, 78:524–546, 1981.

[Tsa11] T. Tsankov. The additive group of the rationals does not have an automatic presentation. *J. Symbolic Logic*, 76(4):1341–1351, 2011.

[Tur36] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[vM19] R. von Mises. Grundlagen der Wahrscheinlichkeitsrechnung. *Math. Z.*, 5(1–2):52–99, 1919.

[Wei00] Klaus Weihrauch. *Computable analysis*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2000.

Sobolev Institute of Mathematics, Novosibirsk, Russia
E-mail address: bazhenov@math.nsc.ru

Victoria University of Wellington
E-mail address: Rod.Downey@msor.vuw.ac.nz

Kazan Federal University, Kazan, Russia
E-mail address: ikalimul@gmail.com

Massey University
E-mail address: alexander.g.melnikov@gmail.com