NON-DENSITY IN PUNCTUAL COMPUTABILITY

NOAM GREENBERG, MATTHEW HARRISON-TRAINOR,
ALEXANDER MELNIKOV, AND DAN TURETSKY

Abstract. In computable structure theory, one considers computable presentations of abstract structures such as graphs or groups, and one thinks of two different computable presentations as being essentially the same if there is a computable isomorphism between them. Because the inverse of a computable function is also computable, the relation of being computably isomorphic is an equivalence relation, and so the only structure on the set of computable presentations is the number of non-equivalent presentations.

Recently there has been increased interest in primitive recursive presentations of structures, and in this setting, the inverse of a primitive recursive function is not necessarily primitive recursive, and so we get a relation of reducibility between structures which induces a partial pre-ordering on the primitive recursive presentations of a structure. Whenever we have a reducibility notion, one of the natural first questions is whether it is dense. We show that it is not dense: There are primitive recursive presentations $\mathcal{A} \cong \mathcal{B}$ of the same abstract structure, such that $\mathcal{A}$ is reducible to $\mathcal{B}$ (there is a primitive recursive isomorphism $\mathcal{A} \to \mathcal{B}$) but $\mathcal{B}$ is not reducible to $\mathcal{A}$ (there is no primitive recursive isomorphism $\mathcal{B} \to \mathcal{A}$), and for any third primitive recursive presentation $\mathcal{M}$ of the same structure, if $\mathcal{A}$ is reducible to $\mathcal{M}$ and $\mathcal{M}$ is reducible to $\mathcal{B}$, then either $\mathcal{M}$ is reducible to $\mathcal{A}$ or $\mathcal{B}$ is reducible to $\mathcal{M}$.

1. Introduction

This paper contributes to the new theory which is focused on eliminating unbounded search from proofs and processes in algebra and infinite combinatorics; see surveys [BDKM19, DMN] for the foundations of this theory. The key motivation here is that unbounded search is often abused in the literature on algorithms performed in infinite algebraic and combinatorial structures. For example, it is well-known that the word problem for finitely generated groups is intrinsic in the sense that if one presentation has decidable word problem then every presentation will have decidable word problem. This is simply because we can match the generators and map words to words to transition between any two given copies. However, this isomorphism relies on the unbounded search as well: we must wait for an element to be spanned by the generators to define its image. Another example is the back-and-forth proof that all countable dense linear orders with no endpoint are isomorphic; see [BDKM19, KMN17, MNb] for more examples. It is natural to ask what happens when we forbid unbounded search, i.e., if we restrict ourselves to *primitive recursive* procedures.

A primitive recursive algorithm of course does not have to be computationally feasible. Nonetheless, somewhat unexpectedly primitive recursive algorithms are useful in the study of more feasible algorithms. As discussed in [KMN17, BDKM19],

Greenberg and Turetsky were supported by a Marsden Fund grant #17-VUW-090.

very often eliminating unbounded search is the crucial step in turning a general Turing computable algebraic procedure into a polynomial time one; for many examples see [Gri90, CDRU09, CR92, CR98, Ala17, Ala18]. Of course, producing a primitive recursive algorithm is typically less challenging than designing a feasible one since we do not have to worry about counting steps explicitly; we only care that there is *some* precomputed bound on all loops and searches. Remarkably, it is not uncommon that the resulting crude primitive recursive algorithm can be modified into a feasible one. For example, the recent solution [BHTK$^+$19] to a problem of Khoussainov and Nerode on the characterisation of automatic structures ([KN08], Question 4.9) relies on a simpler argument that works for primitive recursive structures; with some extra work it is then pushed to automatic structures. Another useful role of primitive recursion is in proving that no feasible procedure is possible at all. Indeed, it is often easiest to argue that a primitive recursive procedure or even a total procedure fails to exist, let alone a polynomial time or automatic one; see, e.g., [CR92, CR98, KMN17]. We conclude that primitive recursion serves as a unifying useful abstraction which connects feasible algebra [CR98, KN08] with the earlier approach to online algorithms in combinatorics via totality [Kie98, KPT94].

In this paper we focus on eliminating unbounded search from proofs and processes in computable structure theory [EG00, AK00]. Following the tradition that goes back to Mal′cev [Mal61] and Rabin [Rab60], computable structure theory studies computably presented countably infinite algebraic structures; these are structures upon the domain $\mathbb{N}$ whose operations and relations are Turing computable. The natural subrecursive analogy of this notion is the following definition:

Definition 1.1 ([KMN17])**.** A countable structure is *fully primitive recursive* or *punctual* if its domain is $\mathbb{N}$ and the operations and predicates of the structure are (uniformly) primitive recursive.

The intuition is that a punctual structure must reveal itself "punctually", i.e., within a precomputed number of steps. We will also fix the convention that all finite structures are also punctual by allowing initial segments of $\mathbb{N}$ to serve as their domains. We will never consider infinite languages in the paper; therefore, we do not need to clarify what uniformity means in Definition 1.1.

Computable structure theory typically studies computable structures up to computable isomorphism, which gives an equivalence relation on computable copies of the same structure. To talk about isomorphisms in the framework of punctual structures, we shall also need to consider *punctual* analogues of computable functions. However, recall that the inverse of a primitive recursive function does not have to be primitive recursive. In contrast with computable structure theory, this leads to a *reduction*:

Definition 1.2 ([KMnN17])**.** A punctual structure $\mathcal{A}$ is *punctually reducible* to a punctual structure $\mathcal{B}$, written $\mathcal{A} \leq_{pr} \mathcal{B}$, if there exists a primitive recursive surjective isomorphism from $\mathcal{A}$ onto $\mathcal{B}$.

This leads to an equivalence relation $\equiv_{pr}$ and *the punctual degree structure* on the classes (the punctual degrees) which will be denoted $\mathbf{PR}(\mathcal{A})$.

The punctual degrees $\mathbf{PR}(\mathcal{A})$ of a structure $\mathcal{A}$ is a rather sensitive invariant which allows us to detect subtle subrecursive differences between two seemingly similar structures. We give an example. The dense linear order is a canonical

example when a computable back-and-forth method works; the other common (algebraically) homogeneous examples include the random graph and the Fraïsse limit of finite abelian p-groups. It seems that for each of these structures the proof requires exactly one unbounded delay at every step. Remarkably, the punctual degrees of the dense linear order, the random graph, and the universal abelian p-group are pairwise non-isomorphic; see [MNb]. Intuitively, the result shows that these delays are actually different in nature in all three cases. Similarly, there exist infinite punctual finitely generated structures having non-isomorphic punctual degrees [KMN17, BKMN]; note that such structures have a unique computable presentation up to a Turing computable isomorphism.

The study of punctual degrees naturally splits into two main themes, one of finding structure and the other of finding examples of non-structure.

In the first theme, one seeks to formulate a general enough property of PR-degrees of $\mathcal{A}$ which is implied by $\mathcal{A}$ having a certain algebraic property, e.g., being finitely generated or homogeneous. It is natural to ask how the algebraic properties of $\mathcal{A}$ are reflected in $\mathbf{PR}(\mathcal{A})$. Very little is known here so far, but there has been some progress in the case of finitely generated structures. For instance, for a finitely generated rigid structure $\mathcal{A}$ with $|\mathcal{A}| > 2$ and a countable lattice $\mathcal{L}$, the following are equivalent: (1) $\mathcal{L}$ can be embedded into $\mathbf{PR}(\mathcal{A})$ preserving sup and inf, and (2) $\mathcal{L}$ is distributive; see [KMZ]. This is a structural result which is yet to be fully understood. Also, it is not hard to show that, for a finitely generated $\mathcal{A}$, $\mathbf{PR}(\mathcal{A})$ has to be dense [BKMN] and is never a Boolean algebra [KMZ]. Work here is ongoing.

The second theme is concerned with finding structures whose punctual degrees have non-trivial and counterintuitive features refuting natural conjectures. While the two themes clearly complement each other both technically and methodologically, the second theme is of some special interest to a computable structure theorist because it resembles the once very popular theory of finite computable dimension [Gon80, GMR89, Hir01, HKSS02, McC02, GLS03] while the techniques are substantially different. For instance, Melnikov and Ng [MNa] have constructed a punctual $\mathcal{A}$ with exactly two punctual degrees. There are also counter-intuitive examples of finitely generated structures [BKMN] and highly technical bizarre examples of unary structures [DGM$^+$]. Each of these counterexamples refutes natural conjecture which intuitively should hold. The main result of this paper contributes to the second theme, since it refutes the following natural conjecture:

Conjecture 1.3. The punctual degrees $\mathbf{PR}(\mathcal{A})$ of any structure $\mathcal{A}$ are always dense.

That is, $\mathcal{A} <_{pr} \mathcal{B}$ should imply that there exists a $\mathcal{C}$ such that $\mathcal{A} <_{pr} \mathcal{C} <_{pr} \mathcal{B}$. The conjecture holds for many natural classes ([KMN17]) as well as for all technical counter-examples known so far [KMZ, DGM$^+$, KMN17, BKMN, KMnN17]. Perhaps most notably, the conjecture holds for arbitrary finitely generated structures [BKMN]. It may seem that if a structure $\mathcal{A}$ is not finitely generated then its punctual degrees should be even more "rich". Such an $\mathcal{A}$ should have more ways of delaying certain configurations from quickly appearing in it, thus there should be even more ways of building an intermediate $\mathcal{C}$. Nonetheless, we prove:

Theorem 1.4. *There exist a punctual structure $\mathcal{A}$ such that $\mathbf{PR}(\mathcal{A})$ are not dense.*

The proof of this theorem is of a special technical interest because it is the first known successful implementation of non-degenerate infinite injury via the tree method in the punctual framework, and as far as we know in feasible algebra in general. The proof contains several novel ideas related to the ways the priority tree method and how it should be understood and used in the punctual context. The fundamental issue with the usual tree method is that we typically cannot simply wait below a Π^0_2 outcome since this wait is clearly unbounded. Nonetheless, guessing at a Π^0_2 outcome seems inevitable in our proof, so we had to adjust the method.

Many questions about punctual degrees remain open. We strongly suspect that the new techniques will be helpful in attacking:

Question 1. Is there a structure $\mathcal{A}$ that has exactly two punctual degrees $\mathbf{b} <_{pr} \mathbf{a}$?

Question 2. Let L be a finite partial order. Is there an $\mathcal{A}$ with $\mathbf{PR}(\mathcal{A}) \cong L$?

In Question 2, finite Boolean algebras and the atomic non-distributive M_5 and N_5 seem most interesting since they will contrast with results from [KMZ].

We also suspect our methods can be applied to natural classes. For instance, we would like to understand which linear orders have dense punctual degrees. We suspect that, for an abelian p-group $\mathcal{A}$, $\mathbf{PR}(\mathcal{A})$ are dense if and only if $|\mathbf{PR}(\mathcal{A})| = \infty$; the latter has a purely algebraic characterisation [KMN17].

2. Proof of Theorem 1.4

Proof. Recall that we need to build punctual $\mathcal{A} <_{pr} \mathcal{B}$ and prove that there is no $\mathcal{M}$ with $\mathcal{A} <_{pr} \mathcal{M} <_{pr} \mathcal{B}$.

2.1. **The language of the structure.** Recall that the domain of every punctual structure has to be ω. The language of $\mathcal{A}$ will be as follows. There will be a unary relation U such that it complement U^c is a pure set in $\mathcal{A}$ in which all n-tuples are automorphic to each other over U. We will be able to "waste time" by adding elements to U^c when necessary. The essential part of the structure of $\mathcal{A}$ will be contained within U. There will also be unary functions f and g. The elements of U will be of two types:

- elements x such that $g(x) = x$ and $f^n(x) = x$ for some $n \geq 3$; or
- elements y such that $f(y) = y$ and $g(y) = x$ for some x of the first type.

We call an element of the first type, together with its images under f, an *n-cycle* where n is least such that $f^n(x) = x$. We call an element of the second type a *tag*. Note that we can immediately tell of which type an element z is by checking whether $g(z) = z$ or $f(z) = z$. An n-cycle will be either tagged or untagged; if it is untagged, then no element of the n-cycle will be the image under g of another element, and if it is tagged, then each element of the n-cycle will be the image under g of a tag, with a different tag for each element of the cycle. So given two tagged n-cycles, any isomorphism of the cycles extends to an isomorphism of the tags. Also, if we see one element of a cycle in a punctual structure, and we know the length of the cycle, we can in a primitive recursive way find all of the elements of the cycle by iterating f; similarly, given a tag, we can find the cycle it is attached to.

2.2. **The requirements.** We build $\mathcal{A}$ and an isomorphic copy $\mathcal{B}$, both punctually computable. At every stage of the construction, we will have partial structures $\mathcal{A}_s$ and $\mathcal{B}_s$ upon initial segments of ω, with $\mathcal{A} = \bigcup \mathcal{A}_s$ and $\mathcal{B} = \bigcup \mathcal{B}_s$.

Remark 2.1. Each $\mathcal{A}_s$ will be a substructure of $\mathcal{B}_s$, and the reduction $\mathcal{A} \leq_{pr} \mathcal{B}$ will be natural in the sense that an element of $\mathcal{A}_s$ will be immediately copied into $\mathcal{B}_s$ but will perhaps receive a different index in the domain ω. Although the reduction witnessing $\mathcal{A} \leq_{pr} \mathcal{B}$ will not be the identity map, it will differ from the identity map up to a punctual g whose range $g(\mathcal{A})$ and whose inverse $g^{-1} : g(\mathcal{A}) \to g$ is also primitive recursive. Having in mind this feature of g, we will sometimes abuse our terminology and call g the identity map.

2.2.1. *The requirements.* Let $(\mathcal{M}_i, \psi_i, \theta_i)_{i \in \omega}$ be a list of the primitive recursive structures and primitive recursive maps $\psi_i : \mathcal{A} \to \mathcal{M}_i$ and $\theta_i : \mathcal{M}_i \to \mathcal{B}$. Let γ_i be a list of the primitive recursive maps $\mathcal{B} \to \mathcal{A}$.

$$\begin{tikzcd}
\mathcal{B} \arrow[dd, bend left, "\gamma_j"] \\
\mathcal{M}_i \arrow[u, "\theta_i"] \\
\mathcal{A} \arrow[u, "\psi_i"]
\end{tikzcd}$$

We must meet the following requirements:

$\mathcal{R}_i$: if ψ_i is an isomorphism from $\mathcal{A}$ onto $\mathcal{M}_i$ and θ_i is an isomorphism from $\mathcal{M}_i$ onto $\mathcal{B}$, then either $\mathcal{M}_i \leq_{pr} \mathcal{A}$ or $\mathcal{B} \leq_{pr} \mathcal{M}_i$.

$\mathcal{S}_i$: γ_i is not a primitive recursive isomorphism from $\mathcal{B}$ onto $\mathcal{A}$.

To make $\mathcal{M}_i \leq_{pr} \mathcal{A}$ or $\mathcal{B} \leq_{pr} \mathcal{M}_i$, we have to define a primitive recursive isomorphism either from $\mathcal{M}_i$ onto $\mathcal{A}$ or from $\mathcal{B}$ onto $\mathcal{M}_i$.

As with any priority argument, it is helpful to first consider a strategy to satisfy requirements individually, and only after this to combine these strategies together. In this argument, combining the strategies is particularly difficult, mainly due to the fact that each of the $\mathcal{M}_i$ is individually primitive recursive, but they are not uniformly primitive recursive.

2.3. **Intuition for one $\mathcal{R}$-strategy.** To begin, we will informally describe the strategy for satisfying a single requirement $\mathcal{R} = \mathcal{R}_i$ while satisfying all of the requirements $\mathcal{S}_i$. Let $\mathcal{M} = \mathcal{M}_i$, $\psi = \psi_i$, $\theta = \theta_i$, where these were defined in the previous subsection.

Up to a punctual delay, we may assume that whenever an element shows up in $\mathcal{M}$, its image in $\mathcal{B}$ under θ is immediately defined, and that its image is an element of the same type; in particular, for $x \in \mathcal{M}$: $x \in U$ if and only if $\theta(x) \in U$; x is part of an n-cycle if and only if $\theta(x)$ is part of an n-cycle; and if x is tagged then $\theta(x)$ is tagged. Note that we cannot say that if $\theta(x)$ is tagged, then x is tagged; this is because the tag y on x yields a tag $\theta(y)$ on $\theta(x)$, but a tag on $\theta(x)$ does not yield a tag on x. Of course if θ is to be an isomorphism, a tag must eventually be added to x, it just does not have to show up within a primitive recursive delay. So we think of $\mathcal{A}$ as a substructure of $\mathcal{M}$, and $\mathcal{M}$ as a substructure of $\mathcal{B}$.

Remark 2.2. It is important to note that this is only the case when we are dealing with only a single requirement $\mathcal{R}$; so this assumption is not strictly necessary but it will make it easier to understand the key points of the construction while dealing with only one $\mathcal{R}$ requirement before adding in complications. In the presence of many requirements we can no longer conclude that this delay described above is primitive recursive *uniformly* in the index i. This is because there is no uniformly primitive recursive enumeration of the θ_i. It will of course be uniformly computable in the index i. Since θ_i is total and thus can be computed in a bounded amount of time, we can keep adding elements to U^c to delay until we see θ_i to halt. It follows that the induced delay is primitive recursive *relative to* θ_i. See [BDKM19] for more examples of subrecursive non-uniformity in punctual structure theory.

Having in mind Remarks 2.1 and 2.2, in presence of only one $\mathcal{R}$-requirement we can identify an element $x \in \mathcal{A}$ with the respective element in $\mathcal{B}$, and also in $\mathcal{M}$:

$$\begin{array}{ccccc} \mathcal{A} & \xrightarrow{\psi} & \mathcal{M} & \xrightarrow{\theta} & \mathcal{B} \\ x & \longrightarrow & \hat{x} & \longrightarrow & x \end{array}$$

If this is ever not the case, we can immediately end the construction with $\mathcal{M} \not\cong \mathcal{A} \cong \mathcal{B}$. (This is by standard techniques but as this will not be necessary in the general case, we will not elaborate on how this can be done.) For any element x of $\mathcal{B}$, we use $\hat{x}$ to denote an element of $\mathcal{M}$ whose image in $\mathcal{B}$ is x. If, for example, x is part of an n-cycle but $\theta(x)$ is not, the requirement $\mathcal{R}$ is automatically satisfied and does not require any further action.

The basic strategy for $\mathcal{S}_i$. To diagonalize against a potential primitive recursive isomorphism $\gamma_i\colon \mathcal{B} \to \mathcal{A}$, put an n-cycle C in $\mathcal{B}$, but to keep it out of $\mathcal{A}$ long enough that γ_i must be defined on C in $\mathcal{B}$ but has no reasonable image for C in $\mathcal{A}$.

A naive idea for meeting $\mathcal{R}_i$. Informally, $\mathcal{B}$ will contain more algebraic subcomponents (cycles) than $\mathcal{A}$, according to the basic $\mathcal{S}_i$-strategy above. If $\mathcal{M}_i$ reveals these extra cycles with some fixed primitive recursive delay, then we can match these cycles with those in $\mathcal{B}$ punctually, and $\mathcal{B} \leq_{pr} \mathcal{M}_i$. However, if $\mathcal{M}_i$ keeps revealing these extra cycles slower than expected, we must make sure that $\mathcal{M}_i \leq_{pr} \mathcal{A}$; intuitively, this urges $\mathcal{A}$ to enumerate the cycle and match it with the cycle in $\mathcal{M}_i$.

Of course, we do not have a uniform measure of "speed" with which a cycle C is expected to appear in $\mathcal{M}_i$, and this has to be guessed using the uniformly computable list of all primitive recursive functions. Suppose currently we work with a timestamp function $\tilde{p}_\ell$ which gives us a measure of enumeration speed (to be clarified later).

The obvious problem with the naive idea described above is that $\mathcal{M}_i$ could reveal finitely many cycles $\tilde{p}_\ell$-fast, and therefore we chose to make progress in witnessing $\mathcal{B} \leq_{pr} \mathcal{M}_i$. However, much later we can discover that some fresh cycle C reveals itself slow relative to $\tilde{p}_\ell$, and therefore we should switch to demonstrating that $\mathcal{M}_i \leq_{pr} \mathcal{A}$. But the slow cycles either still have no image in $\mathcal{A}$ or these images appeared too late, and this is inconsistent with $\mathcal{M}_i \leq_{pr} \mathcal{A}$.

Our solution to this problem (to be described in detail below) is based on the following idea. Every time $\mathcal{M}_i$ reveals a cycle C quickly, we immediately put a *tagged* version C° of this cycle into $\mathcal{A}$ and match C with C°. This way we will be

able to catch up in the definition of $\mathcal{M}_i \leq_{pr} \mathcal{A}$ by putting a tag on C as late as we want.

This idea leads to complications in the strategy. As soon as we put C° into $\mathcal{A}$, a similar tagged cycle must also appear in $\mathcal{M}_i$ because $\mathcal{A} \leq_{pr} \mathcal{M}_i$, and since we must guarantee $\mathcal{A} \leq_{pr} \mathcal{B}$ we must also put an extra tagged cycle in $\mathcal{B}$. This means that, keeping in mind the possibility that $\mathcal{M}_i \leq_{pr} \mathcal{A}$ as above, we will be forced to put another extra cycle into $\mathcal{A}$, etc. Thus, at the end we may end up with lots of cycles, most of which are tagged. To make progress in demonstrating $\mathcal{M}_i \leq_{pr} \mathcal{A}$ we will have to *homogenise* the whole component which grew out of the single cycle C.

We now give more details. We begin with clarifying the concept of primitive recursive time of computation, and then we give a detailed description of the main strategies in presence of only one $\mathcal{R}_i$.

2.4. **Primitive recursive time.** Recall that a function is primitive recursive if, and only if, it can be realised on the universal Turing machine with a primitive recursive timestamp function; see the appendix of [BDKM19]. In other words, we have that $p_i(x) = U(s(i); x)[t(x)] \downarrow$ where $t(x)$ is a primitive recursive timestamp function and s is the primitive recursive function from the s-m-n theorem. Let $(p_i)_{i\in\omega}$ be a uniformly computable list of all primitive recursive functions $\omega \to \omega$. Set $\tilde{p}_i(n)$ to be the sum, over $j < i$, of the primitive recursive timestamp functions for $p_j(n)$ (which are $\geq p_j(n)$). So $\tilde{p}_i$ dominates p_j, $j < i$, on all inputs. If we can ensure that a computable function is slower than each of the $\tilde{p}_i$ then the function cannot be primitive recursive. Moreover, $\tilde{p}_i(n)$ has the nice property that it takes time about $\tilde{p}_i(n)$ to compute $\tilde{p}_i(n)$; more formally, $\tilde{p}_i(n)$ is *time-constructible*, in that $\tilde{p}_i(n)$ can be computed in time $O(\tilde{p}_i(n))$, and this is uniform in i. Imagine that we want to wait until after stage $\tilde{p}_i(n)$ to take some action, but that we want to take that action soon after stage $\tilde{p}_i(n)$. The first step would be to compute $\tilde{p}_i(n)$ to see how long we have to wait, and if $\tilde{p}_i(n)$ took a long time to compute, by the time we had computed it we would already have waited too long and missed our chance. But since $\tilde{p}_i$ is constructed using the stopping times, we can compute $\tilde{p}_i(n)$ in only slightly more than $\tilde{p}_i(n)$ steps as follows: for each $j < i$, compute $p_j(n)$, and then add up the total computation time; this is $\tilde{p}_i(n)$.

At every stage of the construction the strategy for $\mathcal{R}$ is associated with an index ℓ and a potential timestamp function $\tilde{p}_\ell$. Before we explain what exactly the strategy does, we note that the requirement $\mathcal{R}$ will have one of two outcomes:

- Σ_2**:** for some ℓ, whenever an untagged n-cycle C enters $\mathcal{B}$, its copy $\hat{\mathsf{C}}$ enters $\mathcal{M}$ by stage $\max_{x\in\mathsf{C}} \tilde{p}_\ell(x)$.
- Π_2**:** for every ℓ, there is an untagged n-cycle C in $\mathcal{B}$ such that $\hat{\mathsf{C}}$ does not enter $\mathcal{M}$ by stage $\max_{x\in\mathsf{C}} \tilde{p}_\ell(x)$.

There will be many tagged n-cycles in all of the structures $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, so it is really the untagged n-cycles that we will have to worry about.

Remark 2.3. Essentially, one should think of the Σ_2 outcome as saying that untagged n-cycles entered $\mathcal{M}$ soon after they entered $\mathcal{B}$, and thus using $\tilde{p}_\ell$ we can demonstrate $\mathcal{B} \leq_{pr} \mathcal{M}_i$. The Π_2 outcome says that for any primitive recursive delay $\tilde{p}_\ell$, there are n-cycles which enter $\mathcal{M}$ much later than they entered $\mathcal{B}$, relative to the delay p_ℓ. In this outcome, the strategy combined with *homogenization* of other components (where we thought we had the Σ_2 outcome) will allow us to punctually map $\mathcal{M}$ onto $\mathcal{A}$.

More on the strategy for $\mathcal{S}_i$. Recall that we also have to diagonalize against maps $\gamma_i\colon \mathcal{B} \to \mathcal{A}$. In the Σ_2 outcome, we can put untagged n-cycles C in $\mathcal{B}$ and hold them out of $\mathcal{A}$ as long as we want; therefore, we simply follow the basic strategy for $\mathcal{S}_i$ as explained above. In the Π_2 outcome, we have to build a map from $\mathcal{M}$ to $\mathcal{A}$. Only when an untagged n-cycle C enters $\mathcal{M}$ we are forced to put it in $\mathcal{A}$. Since we are in the Π_2 outcome, there are many cycles C that take a long time to enter $\mathcal{M}$, and therefore nothing urges us to put them into $\mathcal{A}$. We will use this feature to demonstrate $\gamma_i\colon \mathcal{B} \to \mathcal{A}$ cannot be an isomorphism; more details will be given in §2.5.2. In particular, we do not have to meet $\mathcal{S}_i$ explicitly (via its basic strategy) in this case.

2.5. **Formal details for one $\mathcal{R}$ combined with all $\mathcal{S}_i$.** The structure we build will be divided up into different *components*, each of which is the location of an attempt to satisfy a particular requirement; the n-component consists of all of the n-cycles and their tags. Recall that we use the complement of U to delay when necessary. Also, recall that we identify cycles in $\mathcal{A}$ and $\mathcal{B}$ which indeed agree up to a primitive recursive correspondence; see Remark 2.1

2.5.1. *The description of one n-component.* The n-component works as follows. When it begins to act, it chooses the least ℓ such that the requirement $\mathcal{R}$ might still have the Σ_2 outcome with witness index ℓ. We call this ℓ the *threshold value*. The n-component begins by adding an untagged n-cycle C to $\mathcal{B}$, but not to $\mathcal{A}$, as shown in the diagram below.

$$\begin{array}{ll} \mathcal{B}: & \mathsf{C} \\ \\ \mathcal{M}: & \\ \\ \mathcal{A}: & \end{array}$$

We then wait for one of the following two things to happen, in which case we take the corresponding action. We call one the (current) Π_2 outcome, and the other the (current) Σ_2 outcome. Let s be the current stage.

- Π_2 outcome: A version of C has not showed up in $\mathcal{M}$ within $\tilde{p}_\ell(\mathsf{C}) = \max_{x\in\mathsf{C}} \tilde{p}_\ell(x)$ many steps. Add C to $\mathcal{A}$; when an untagged n-cycle $\hat{\mathsf{C}} = \psi(\mathsf{C})$ shows up in $\mathcal{M}$, we have φ map it to C.

$$\begin{array}{ll} \mathcal{B}: & \mathsf{C} \\ \\ \mathcal{M}: & \hat{\mathsf{C}} \\ & \downarrow \varphi \\ \mathcal{A}: & \mathsf{C} \end{array}$$

- Σ_2 outcome: An n-cycle $\hat{\mathsf{C}}$ enters $\mathcal{M}$ within at most $\tilde{p}_\ell(\mathsf{C}) = \max_{x\in\mathsf{C}} \tilde{p}_\ell(x)$ steps. Add a tagged n-cycle D_1 to $\mathcal{A}$ and $\mathcal{B}$. Set $\varphi(\hat{\mathsf{C}}) = \mathsf{D}_1$.

$$\begin{tikzcd}
\mathcal{B}: & \mathsf{C} & \circ\mathsf{D}_1 \\
\mathcal{M}: & \hat{\mathsf{C}} \arrow[dr, "\varphi"] & \\
\mathcal{A}: & & \circ\mathsf{D}_1
\end{tikzcd}$$

(The circles here denote that the cycle is tagged.) Note that φ is not currently looking like an isomorphism, because $\hat{\mathsf{C}}$ is untagged, but D_1 is tagged.

Then, as soon as the ψ-image $\hat{\mathsf{D}}_1$ of D_1 shows up in $\mathcal{M}$, add a new tagged n-cycle D_2 to $\mathcal{A}$ and, thus, to $\mathcal{B}$.

$$\begin{tikzcd}
\mathcal{B}: & \mathsf{C} & \circ\mathsf{D}_1 & \circ\mathsf{D}_2 \\
\mathcal{M}: & \hat{\mathsf{C}} \arrow[dr, "\varphi"] & \circ\hat{\mathsf{D}}_1 \arrow[dr, "\varphi"] & \\
\mathcal{A}: & & \circ\mathsf{D}_1 & \circ\mathsf{D}_2
\end{tikzcd}$$

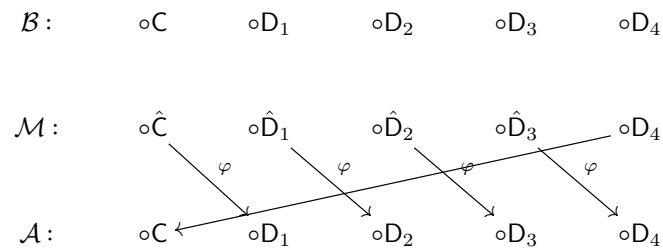
We continue doing this. To meet $\mathcal{S}_1, \ldots, \mathcal{S}_n$, we need to diagonalise against $\gamma_1, \ldots, \gamma_n : \mathcal{B} \to \mathcal{A}$. Eventually, the maps $\gamma_1, \ldots, \gamma_n$ must become defined on $\mathsf{C} \subseteq \mathcal{B}$, but they cannot map C to another untagged n-cycle, because there are no untagged n-cycles in $\mathcal{A}$; the best that such a $\gamma = \gamma_j$ can do is to map C to some D_i as shown in the diagram below. After each of $\gamma_1, \ldots, \gamma_n$ have been computed on C, we add C to $\mathcal{A}$.

$$\begin{tikzcd}
\mathcal{B}: & \mathsf{C} \arrow[ddrr, "\gamma"] & \circ\mathsf{D}_1 & \circ\mathsf{D}_2 & \circ\mathsf{D}_3 & \cdots \\
\mathcal{M}: & \hat{\mathsf{C}} \arrow[dr, "\varphi"] & \circ\hat{\mathsf{D}}_1 \arrow[dr, "\varphi"] & \circ\hat{\mathsf{D}}_1 \arrow[dr, "\varphi"] & \circ\hat{\mathsf{D}}_1 \arrow[dr, "\varphi"] & \cdots \\
\mathcal{A}: & \mathsf{C} & \circ\mathsf{D}_1 & \circ\mathsf{D}_2 & \circ\mathsf{D}_3 & \cdots
\end{tikzcd}$$

Even after this happened, we keep adding new tagged n-cycles to the component.

It is possible that at some point in the construction, the n-component will have to be homogenized. What this means is that we want to make φ—which so far does not look like an isomorphism, because it maps the untagged $\hat{\mathsf{C}}$ to the tagged D_1—into an isomorphism. Add a tag to C, and

extend φ to map the last n-cycle D_k we have built so far to C, e.g.



Now φ looks like an isomorphism on the n-component.

This completes the strategy for a single component.

2.5.2. *Combining several components.* Start the first n-component with $n = 3$ and $\ell = 0$ (recall the minimum length of a cycle is 3). Once the current component is finished acting, start a new n-component with n being the current stage and the threshold value ℓ being the least ℓ such that no component with the same threshold value has had a Π_2 outcome. Whenever we have a new component with a Π_2 outcome, homogenize all of the components with Σ_2 outcomes.

In the end, we will be in one of two cases:

- All of the components have the Π_2 outcome or were homogenized; and for each ℓ there is a component with threshold value ℓ. In this case φ is an isomorphism $\mathcal{M} \to \mathcal{A}$, and so the requirement $\mathcal{R}$ is satisfied.

  We argue that there is no primitive recursive isomorphism $\mathcal{B} \to \mathcal{A}$. If there was, say γ, then let ℓ be such that $\tilde{p}_\ell$ dominates γ. Let n be such that the n-component had this threshold value ℓ and outcome Π_2. There is an untagged n-cycle C in $\mathcal{B}$, but no untagged n-cycle is in $\mathcal{A}$ until after stage $\max_{x\in\mathsf{C}} \tilde{p}_\ell(x)$; since $\tilde{p}_\ell$ dominates γ, γ cannot cannot map C to an untagged n-cycle in $\mathcal{A}$, and hence cannot be an isomorphism.
- All but finitely many components have the Σ_2 outcome with the threshold value ℓ which will never be homogenised; the other components either have the Π_2 outcome or are homogenized.

  In this case there exists a primitive recursive isomorphism $\mathcal{B} \to \mathcal{M}$. Let ℓ be such that all of the non-homogenized components with the Σ_2 outcome have threshold value ℓ. We can define the isomorphism non-uniformly on the finitely many other components. We argue that, on cofinitely many components, the map $\theta^{-1} : \mathcal{B} \to \mathcal{M}$ is punctual. When a cycle C is first added to $\mathcal{B}$, wait for $\max_{x\in\mathsf{C}} \tilde{p}_\ell(x)$-many steps for $\theta^{-1}(\hat{\mathsf{C}})$ to show up in $\mathcal{M}$ and map C to $\hat{\mathsf{C}}$. For an element D_i, when we add D_i to $\mathcal{B}$, we also add it to $\mathcal{A}$, and we can wait for the image $\hat{\mathsf{D}}_i = \psi(\mathsf{D}_i)$ to show up in $\mathcal{M}$; we then map D_i to $\hat{\mathsf{D}}_i$. This process is punctual since p_ℓ and ψ are primitive recursive. So the requirement $\mathcal{R}$ is satisfied.

  We also argue that in this case there is no primitive recursive isomorphism $\mathcal{B} \to \mathcal{A}$; but as described in the construction, an n-component with Σ_2 outcome ensures that $\gamma_1, \ldots, \gamma_n$ are not isomorphisms before adding C to $\mathcal{A}$ unless it is later homogenised. Since there are infinitely many n-components with arbitrarily large n having a Σ_2-outcome and which are never homogenised, all $\mathcal{S}_j$-requirements are met.

In either case, we have satisfied $\mathcal{R}$ and each requirement $\mathcal{S}_j$.

2.6. **The full construction.** Now we will describe how to meet all of the requirements. An n-component will now work with the $\mathcal{R}$-requirements $\mathcal{R}_0, \dots, \mathcal{R}_n$. There is some additional complexity that arises when dealing with many structures $\mathcal{M}_0, \mathcal{M}_1, \dots$.

2.6.1. *The four main issues.* There are four key issues which we will have to circumvent in presence of many $\mathcal{R}$-requirements and which were not visible in the case of only one $\mathcal{R}$-requirement. We informally discuss these issues below; the formal construction is contained in the next subsection.

First, when we add the n-cycle C to $\mathcal{B}$, some of the monitored $\mathcal{M}_i$ might add the corresponding n-cycle $\hat{\mathsf{C}}$ to $\mathcal{M}_i$ very quickly, and others very slowly, where quickly and slowly for $\mathcal{M}_i$ are measured relative to a primitive recursive threshold function $\tilde{p}_{\ell_i}$. When some $\mathcal{M}_i$ has an n-cycle $\hat{\mathsf{C}}$ show up slowly relative to $\tilde{p}_{\ell_i}$ (the Π_2 outcome), we must quickly add C to $\mathcal{A}$ because we want to build a primitive recursive isomorphism $\mathcal{M}_i \to \mathcal{A}$ mapping $\hat{\mathsf{C}}$ in $\mathcal{M}_i$ to C in $\mathcal{A}$). But if $\ell_i < \ell_j$, and hence $\tilde{p}_{\ell_i}$ is dominated by $\tilde{p}_{\ell_j}$, this process might be happening quickly relative to $\tilde{p}_{\ell_j}$, and we have not yet finished waiting for $\mathcal{M}_j$ to respond. Thus, we are adding C to $\mathcal{A}$ sooner than we would like for $\mathcal{M}_j$.

To resolve this, we use the following idea. Since the scenario above happens only when $\mathcal{R}_i$ plays its Π^0_2-outcome, $\mathcal{R}_j$ can assume that $\mathcal{R}_i$ has the true outcome Π^0_2 which means that $\mathcal{M}_i$ is arbitrarily slow. In particular, $\mathcal{R}_i$ will eventually have its threshold lifted arbitrarily large and will play its Π^0_2-outcome again and again arbitrarily late in the construction. Thus, it is safe for $\mathcal{M}_j$ to rely on the threshold of $\mathcal{R}_i$ for its own Π^0_2-outcome, provided that the true outcome of $\mathcal{M}_i$ is indeed Π^0_2.

Of course, we will have to handle more that one requirement at once. In this case, in addition to the outcome, we will maintain the *maximum threshold* value $\ell_{max} := \ell_i$; this keeps track of how long we were able to wait for $\hat{\mathsf{C}}$ to show up. If all of the requirements $\mathcal{R}_1, \dots, \mathcal{R}_k$ we are dealing with have the Σ_2 outcome, then the threshold value is just the maximum of $\ell_1, \dots, \ell_k$ because we were able to wait as long as we wanted before adding C to $\mathcal{A}$. An important aspect of the construction will be that the maximum thresholds eventually keep increasing, in the sense that their limit is ∞, so that they do not limit the construction. This will be further clarified below and then verified in Lemma 2.8.

Second, there is a new issue with homogenization and building a potential isomorphism. Recall that we had to homogenize a component, say the n-component, because it had the outcome Σ_2, and later the m-component with $m > n$ played the outcome Π_2. Recall that (§2.5.1) we would wait for a cycle to appear in the opponent's structure within a certain bound, and only then we would perhaps add a tagged cycle to $\mathcal{A}$. In presence of many $\mathcal{R}$-requirements we cannot afford to wait for many opponent's structures to respond within distinct and increasing bounds. Also, we never had to homogenize a component which played its Π_2 outcome. The strategy will nevertheless be similar. Whereas before we waited until a component had the Σ_2 outcome to introduce tagged n-cycles D, we will now always add tagged n-cycles to each component. Consider the following example.

Example 2.4. Suppose the n-component plays its Σ_2 outcome for $\mathcal{R}_1$, but the Π_2 outcome for $\mathcal{R}_2$. In particular, it attempts to build an isomorphism $\mathcal{M}_2 \to \mathcal{A}$.

Suppose also that at a later stage the n-component must be homogenized because some later component has the Π_2 outcome for $\mathcal{R}_1$. The question is whether we would like to add a tagged cycle to $\mathcal{A}$, and if yes then how soon. The basic strategy for $\mathcal{R}_2$ says that a tagged cycle is to be added only when $\mathcal{R}_2$ finishes its computation with its much slower threshold. However, $\mathcal{R}_1$ cannot afford to wait for this computation to finish since it has to act now.

The solution will be to add a tagged cycle to $\mathcal{A}$ without waiting for $\mathcal{R}_2$ to finish its computation. This action will force a tagged cycle to appear in $\mathcal{M}_2$, and this can potentially happen very quickly. This is of course consistent with the basic strategy for $\mathcal{R}_2$, because it really only cares about untagged cycles. Also, it could then be the case that the global true outcome of $\mathcal{R}_2$ is Σ_2 (i.e., cofinitely many components will play Σ_2 with the same threshold) and so we actually need to build an isomorphism $\mathcal{B} \to \mathcal{M}_2$. Thus, we must also think of the potential image of the new cycle of $\mathcal{M}_2$ in $\mathcal{B}$, but this will be provided by the basic strategy for $\mathcal{R}_1$.

To summarise, to make sure that we never get stuck in the definitions of our maps and that we have enough tagged cycles, just add as many as necessary and as quickly as possible. This will not upset the main strategy in its essence.

Third, when dealing with a single requirement $\mathcal{R}$, we were able to assume that the composition $\theta \circ \psi \colon \mathcal{A} \to \mathcal{M} \to \mathcal{B}$ was essentially the identity map in the sense of Remark 2.1. The strategy then was to only add a single n-cycle at a time, and to wait for its images to appear in $\mathcal{M}$ before adding a new n-cycle. Now, when we add an n-cycle to $\mathcal{A}$, we might need to add more n-cycles to $\mathcal{A}$ for the sake of $\mathcal{M}_i$ (which must act relatively quickly) before the image of the n-cycle has appeared in $\mathcal{M}_j$ (which might be acting much more slowly). This means in particular that, for instance, θ_j will have many potential images for a given (tagged) cycle, and its choice does not have to naturally line up with what θ_i does. Nevertheless this is not really an issue because there is little interaction between θ_i and θ_j, and similarly for ψ_i and ψ_j. All we need is that $\mathcal{B}$ and $\mathcal{A}$ have enough (tagged) cycles to be matched with those in $\mathcal{M}_j$, and this is up to the opponent to ensure that his maps make sound choices.

There is one time at which we will need a little more control, and that is when a component is homogenized. When the n-component is homogenized, we will stop adding new n-cycles, which will mean that each of $\mathcal{A}$, $\mathcal{M}_i$, and $\mathcal{B}$ have exactly the same number of n-cycles, all of which will be tagged; thus ψ_i and θ_i will naturally have to be surjective on the n-cycles, and this will be enough for us.

Fourth, we cannot have a single threshold value for each requirement, but rather for each requirement $\mathcal{R}_i$ and each guess σ at the outcomes of the requirements $\mathcal{R}_0, \ldots, \mathcal{R}_{i-1}$, we must have a threshold value ℓ_i^σ. Whether $\hat{\mathsf{C}}$ enters $\mathcal{M}_i$ quickly or slowly is dependent on the threshold value; but we have a number of different possible threshold values ℓ_i^σ for each σ, and which one we use depends on the current outcomes of $\mathcal{R}_0, \ldots, \mathcal{R}_{i-1}$. This combinatorial complexity is sorted using the tree of strategies.

2.6.2. *The tree of strategies.* The n-component will have an *outcome* for each requirement $\mathcal{R}$ which is either Σ_2 or Π_2. We think of these outcomes as being organized on a tree, with the Π_2 outcomes to the left of the Σ_2 outcomes. Formally, we

define the tree of (possible) outcomes $\mathcal{T} = \{\Sigma_2, \Pi_2\}^{<\omega}$ which consists of the finite maps from indices i of requirements $\mathcal{R}_i$ to outcomes $\{\Sigma_2, \Pi_2\}$.

Whenever a component plays an outcome σ, it injures every previous component that had an outcome to the right of σ on the tree (which will mean homogenizing the components which had those outcomes). The true path will, as usual, be the leftmost path visited infinitely often.

For each requirement i and string $\sigma = \langle \sigma_0, \dots, \sigma_{i-1} \rangle$ of outcomes for the higher priority requirements, we maintain a threshold value ℓ^σ. These values get updated after the action of each component, and we denote the threshold value after the n-component by $\ell^\sigma[n]$. Sometimes we write ℓ_i^σ for ℓ^σ as a reminder of which requirement $\mathcal{R}_i$ the threshold value is for, though of course $i = |\sigma|$. The threshold value represents our current best guess at the witness for the Σ_2 outcome of $\mathcal{R}_i$ under the assumption that the higher priority requirements have the outcomes listed in σ.

2.6.3. *The formal construction.* We will describe the action of each component, leaving the construction of the isomorphisms to the verification. At any point in time only one component will be active. On completion, the n-component will define an *outcome* $\mathbf{\Gamma} = (\Gamma_1[n], \dots, \Gamma_k[n])$ and a *maximum threshold* $\ell_{max}[n]$ as well as defining new threshold values $\ell^\sigma[n]$. We begin with the 3-component, and when it is finished continue with the 4-component, the 5-component, etc.

Action of the n-component: Begin by adding an untagged n-cycle C to $\mathcal{B}$, and at each stage add a new tagged n-cycle $\mathsf{D}_1, \mathsf{D}_2, \dots$ to $\mathcal{A}$ and $\mathcal{B}$:

$$\begin{array}{lccccc} \mathcal{B}: & \mathsf{C} & \circ\mathsf{D}_1 & \circ\mathsf{D}_2 & \circ\mathsf{D}_3 & \cdots \\ \\ \mathcal{A}: & & \circ\mathsf{D}_1 & \circ\mathsf{D}_2 & \circ\mathsf{D}_3 & \cdots \end{array}$$

We keep adding new tagged n-cycles D until the component is homogenized. Then, until the loop is ended, at each consequent stage s do the following:

- For each $i \le n$ for which $\mathcal{R}_i$ has not yet been determined to have the Σ_2 outcome, check whether an n-cycle $\hat{\mathsf{C}}$ enters $\mathcal{M}_i[s]$ with $\hat{\mathsf{C}}$ mapped to C in $\mathcal{B}$ by $\theta_i : \mathcal{M}_i \to \mathcal{B}$. If not, then do nothing; if such an n-cycle $\hat{\mathsf{C}}$ has entered $\mathcal{M}_i$, then declare $\mathcal{R}_i$ to have the outcome $\Gamma_i[n] := \Sigma_2$.
- For each $i \le n$ for which $\mathcal{R}_i$ has not yet been declared to have the Σ_2 outcome, check whether there is $\sigma = (\sigma_0, \dots, \sigma_{i-1})$ such that:
  - for $j < i$, if σ_j is Σ_2 then $\mathcal{R}_j$ has already been declared to have the Σ_2 outcome in this component;
  - with $\ell = \ell_i^\sigma$, the current stage s has $s \ge \tilde{p}_\ell(x)$ for each $x \in \mathsf{C} \subseteq \mathcal{B}$.

  If there is such an i, choose the least. The maximum threshold is $\ell_{max}[n] := \ell_i^\sigma[n-1]$ for the witness i. Declare each $\mathcal{R}_j$ which does not already have the Σ_2 outcome to have the outcome $\Gamma_j[n] := \Pi_2$. Add C to $\mathcal{A}$. End the loop.
- If all of $\mathcal{R}_1, \dots, \mathcal{R}_n$ have been declared to have the Σ_2 outcome, declare each $\mathcal{R}_j$, $j > n$, to have the outcome $\Gamma_j[n] := \Pi_2$. Eventually the maps $\gamma_1, \dots, \gamma_n : \mathcal{B} \to \mathcal{A}$ must become defined on $\mathsf{C} \subseteq \mathcal{B}$, but they cannot map C to another untagged n-cycle, because there are no untagged n-cycles in $\mathcal{A}$;

the best that such a γ can do is to map C to some tagged n-cycle D. After each of $\gamma_1, \dots, \gamma_n$ have been defined on C, we can add C to $\mathcal{A}$.

Once this has happened we will end the loop. The maximum threshold for the component is $\ell_{max}[n] \coloneqq \ell_{max}[n-1]+1$.

Let $\mathbf{\Gamma}[n] = (\Gamma_1[n], \Gamma_2[n], \dots)$ be the outcome of the n-component. For each m-component, $m < n$, with outcome $\mathbf{\Gamma}[m]$ to the right of $\mathbf{\Gamma}[n]$—i.e. with, for some i, $\Gamma_1[m] = \Gamma_1[n], \dots, \Gamma_{i-1}[m] = \Gamma_{i-1}[n]$, and $\Gamma_i[m] = \Sigma_2$ but $\Gamma_i[n] = \Pi_2$—we homogenize the m-component by adding a tag to C. We stop adding new tagged m-cycles to $\mathcal{A}$ and $\mathcal{B}$.

We update the thresholds as follows:

- For each $\sigma \prec \mathbf{\Gamma}$, with $i = |\sigma|$, define:
  - $\ell^\sigma[n] \coloneqq \ell^\sigma[n-1]$ if $\Gamma_i = \Sigma_2$ (i.e., if $\mathcal{R}_i$ had the Σ_2 outcome in this component); and
  - $\ell^\sigma[n] \coloneqq \ell[n-1]+1$ if $\Gamma_i = \Pi_2$ (i.e., if $\mathcal{R}_i$ had the Π_2 outcome in this component).

  (This defines ℓ on initial segments of $\mathbf{\Gamma}$.)
- For each $\sigma\Pi_2 \prec \mathbf{\Gamma}$, with $i = |\sigma|$, and each $\tau \in \{\Sigma_2, \Pi_2\}^{<\omega}$, let $\ell^{\sigma\Sigma_2\tau}[n] \coloneqq \max(\ell^\sigma[n-1]+1, \ell^{\sigma\Sigma_2\tau}[n-1])$. (This defines ℓ everywhere to the right of $\mathbf{\Gamma}$.)
- For each other σ, let $\ell^\sigma[n] \coloneqq \ell^\sigma[n-1]$. (This defines ℓ everywhere to the left of $\mathbf{\Gamma}$.)

This ends the construction.

2.7. **Verification.** Let $n_1, n_2, n_3, \dots$ be the n-components which are never homogenized. The standard type of argument in priority constructions proves:

Lemma 2.5. *For each requirement $\mathcal{R}_i$, one of the following is the case:*

(1) For every sufficiently large j, $\Gamma_i[n_j] = \Sigma_2$, or
(2) For every sufficiently large j, $\Gamma_i[n_j] = \Pi_2$.

Proof. We argue by induction on i. We may assume that for every sufficiently large $j \geq J$, for each $i' < i$, $\Gamma_{i'}[n_j]$ takes on a fixed value. Now if $j' > j > J$ and $\Gamma_i[n_{j'}] = \Pi_2$ but $\Gamma_i[n_j] = \Sigma_2$, then $\Gamma_1[n_{j'}], \dots, \Gamma_i[n_{j'}]$ would be to the left of $\Gamma_1[n_j], \dots, \Gamma_i[n_j]$, contradicting the fact that the n_j-component is never homogenized. So for sufficiently large j, $\Gamma_i[n_j]$ takes on the same value (either Σ_2 or Π_2). □

In the first case, we say that the requirement $\mathcal{R}_i$ has *(true) outcome* Σ_2; in this case, $\hat{\mathsf{C}}$ always appears relatively quickly in $\mathcal{M}_i$, where quickly is measured relative to some primitive recursive function $\tilde{p}_\ell$. Lemma 2.6 shows that in this case the threshold values stabilize on a single value of ℓ. In the second case, we say that $\mathcal{R}_i$ has *(true) outcome* Π_2. Then $\hat{\mathsf{C}}$ often appears slowly (or not at all) in $\mathcal{M}_i$, relative to any primitive recursive function. Lemma 2.7 shows that in this case the threshold values increase unbounded.

Lemma 2.6. *Let σ be the sequence of true outcomes of $\mathcal{R}_0, \dots, \mathcal{R}_{i-1}$. If the true outcome of $\mathcal{R}_i$ is Σ_2, then there is ℓ such that for sufficiently large j, $\ell_i^\sigma[n_j] = \ell$.*

Proof. Let K be sufficiently large that for $k \geq K$ and $j \leq i$, $\Gamma_j[n_k]$ is the true outcome of $\mathcal{R}_j$. Thus, for $m \geq n_K$, the outcome $\mathbf{\Gamma}[m]$ of the m-component is never

to the left of $\sigma^\frown\langle\Sigma_2\rangle$, as if it was, the n_K-component would be injured. So for each such m, either $\mathbf{\Gamma}[m]$ is to the right of σ, or it extends $\sigma^\frown\{\Sigma_2\}$. A component never changes the thresholds ℓ^σ where σ is to the left of its outcome, so if $\mathbf{\Gamma}[m]$ is to the right of σ, then we have $\ell_i^\sigma[m] = \ell_i^\sigma[m-1]$. If $\mathbf{\Gamma}[m]$ extends $\sigma^\frown\{\Sigma_2\}$, then we also have $\ell_i^\sigma[m] = \ell_i^\sigma[m-1]$. Thus for all $m \geq K$, $\ell_i^\sigma[m] = \ell_i^\sigma[n_K]$. □

Lemma 2.7. *Let σ be the sequence of true outcomes of $\mathcal{R}_0, \ldots, \mathcal{R}_{i-1}$. If the true outcome of $\mathcal{R}_i$ is Π_2, then $\lim_{j\to\infty} \ell_i^\sigma[n_j] = \infty$.*

Proof. Let K be sufficiently large that for $k \geq K$ and $j \leq i$, $\Gamma_j[n_k]$ is the true outcome of $\mathcal{R}_j$. Thus, for $m \geq n_K$, the outcome $\mathbf{\Gamma}[m]$ of the m-component is never to the left of $\sigma^\frown\langle\Pi_2\rangle$, as if it was, the n_K-component would be injured. So for each such m, either $\mathbf{\Gamma}[m]$ is to the right of σ, it extends $\sigma^\frown\{\Sigma_2\}$, or it extends $\sigma^\frown\{\Pi_2\}$. A component never changes the thresholds to the left of its outcome, so if $\mathbf{\Gamma}[m]$ is to the right of σ, then we have $\ell_i^\sigma[m] = \ell_i^\sigma[m-1]$. If $\mathbf{\Gamma}[m]$ extends $\sigma^\frown\{\Sigma_2\}$, then we also have $\ell_i^\sigma[m] = \ell_i^\sigma[m-1]$. And if $\mathbf{\Gamma}[m]$ extends $\sigma^\frown\{\Pi_2\}$, then we also have $\ell_i^\sigma[m] = \ell_i^\sigma[m-1]+1$. The latter is the case whenever $m = n_k$, $k \geq K$, and so it follows that $\lim_{j\to\infty} \ell_i^\sigma[n_j] = \infty$. □

Suppose that σ is the outcome of the requirements $\mathcal{R}_0, \ldots, \mathcal{R}_{i-1}$ in the n-component, and that $\mathcal{R}_i$ has outcome Π_2 in the n-component. The idea of the construction was that we want to wait for $\hat{\mathsf{C}}$ to show up in $\mathcal{M}_i$ until we can compute $\tilde{p}_{\ell_i^\sigma[n-1]}(x)$. But we might not wait this long, because before this happens we might find that some other requirement has the Π_2 outcome. The maximum threshold $\ell_{max}[n]$ measures how long we actually waited. The next lemma will show that this threshold value increases to infinity, which will mean that this poses no issue.

Lemma 2.8. $\lim_{m\to\infty} \ell_{max}[m] = \infty$.

Proof. To each n-component, we will assign a sequence $\mathbf{t}[n] = (t_1[n], t_2[n], t_3[n], t_4[n], \ldots)$. Here, $t_r[n]$ will be the number of σ with $\ell^\sigma[n] = r$. Note that each entry of $\mathbf{t}[n]$ is always finite (because we have $\ell^\sigma[n] \geq |\sigma|$).

First, we always have $\ell^\sigma[n] \geq \ell^\sigma[m]$ for $n > m$, which means that the sequence of $\mathbf{t}[n]$ is non-increasing as the components increase: $\mathbf{t}[3] \geq \mathbf{t}[4] \geq \cdots$.

Now we will argue that if $\ell_{max}[n+1] \leq \ell_{max}[n]$, then $\mathbf{t}[n+1] < \mathbf{t}[n]$ and, in particular, for some $i \leq \ell_{max}[n]$, $t_i[n+1] < t_i[n]$. This can only happen finitely many times for each i, so we can conclude that $\limsup_{m\to\infty} \ell_{max}[m] = \infty$. There are two possibilities from one stage to the next, only in the second of which can we have $\ell_{max}[n+1] \leq \ell_{max}[n]$:

- If each of $\mathcal{R}_1, \ldots, \mathcal{R}_{n+1}$ in the $(n+1)$-component have the Σ_2 outcome, then we define
$$\ell_{max}[n+1] = \ell_{max}[n] + 1.$$
- If some $\mathcal{R}_i$ had the Π_2 outcome and ended the loop for the $(n+1)$-component, then we set $\ell_{max}[n+1] = \ell_i^\sigma[n]$ where $\sigma = (\Gamma_0[n], \ldots, \Gamma_{i-1}[n])$. We have $\ell^\sigma[n+1] = \ell^\sigma[n] + 1$. Together with the fact that for each τ, $\ell^\tau[n+1] \geq \ell^\tau[n]$, this implies that $\mathbf{t}[n+1] < \mathbf{t}[n]$, and in particular for some $j \leq \ell^\sigma[n] = \ell_{max}[n]$, $t_j[n+1] < t_j[n]$.

This proves the lemma. □

Let σ be a list of outcomes for $\mathcal{R}_0, \dots, \mathcal{R}_{i-1}$. Whenever we have a Σ_2 outcome for $\mathcal{R}_i$ extending σ for the first time, we reset the thresholds ℓ^τ for τ extending $\sigma\Sigma_2$ so that $\ell^\tau \geq \ell^\sigma$. This results in:

Lemma 2.9. *Let σ, τ be a sequence of outcomes with $\sigma^\frown\langle\Sigma_2\rangle \preceq \tau$. Then for every n, $\ell^\sigma[n] \leq \ell^\tau[n]$.*

Proof. Let $i = |\sigma|$. This is true at the beginning (for the 3-component), and we argue that it remains true from one stage to the next.

First suppose that $\Gamma_i[n+1] = \Sigma_2$. Then we have $\ell^\sigma[n+1] = \ell^\sigma[n]$. For each $\tau \in \{\Sigma_2, \Pi_2\}^{<\omega}$, we have $\ell^{\sigma\Sigma_2\tau}[n+1] := \ell^{\sigma\Sigma_2\tau}[n]$. Since $\ell^\sigma[n] \leq \ell^{\sigma\Sigma_2\tau}[n]$, the same remains true for $n+1$.

Second, suppose that $\Gamma_i[n+1] = \Sigma_2$. Then we have $\ell^\sigma[n+1] = \ell^\sigma[n]+1$. For each $\tau \in \{\Sigma_2, \Pi_2\}^{<\omega}$, we have $\ell^{\sigma\Sigma_2\tau}[n+1] := \max(\ell^\sigma[n]+1, \ell^{\sigma\Sigma_2\tau}[n]) \geq \ell^\sigma[n+1]$. □

The next two lemmas say that the outcome of $\mathcal{R}_i$ in the n-component actually determines how quickly or slowly $\hat{\mathsf{C}}$ entered $\mathcal{M}_i$.

Lemma 2.10. *Let σ be the sequence of outcomes for $\mathcal{R}_1, \dots, \mathcal{R}_{i-1}$ in the n-component. Suppose that the outcome $\Gamma_i[n]$ of $\mathcal{R}_i$ on the n-component was Σ_2. Let s be the first stage at which $\tilde{p}_{\ell^\sigma[n-1]}(x)$ is computed for every $x \in \mathsf{C} \subseteq \mathcal{B}$. Then $\hat{\mathsf{C}}$ entered $\mathcal{M}_i$ by the stage s.*

Proof. We argue by induction on i. Inductively, and using Lemma 2.9, by this stage s, for every $j < i$ with $\sigma(j) = \Sigma_2$, $\mathcal{R}_j$ has already been declared to have the Σ_2 outcome. Then if $\hat{\mathsf{C}}$ had not entered $\mathcal{M}_i$ by stage s, the n-component would declare $\mathcal{R}_i$ to have the Π_2 outcome. Since this is not the case, $\hat{\mathsf{C}}$ must have entered $\mathcal{M}_i$ by stage s. □

Lemma 2.11. *Suppose that the outcome $\Gamma_i[n]$ of $\mathcal{R}_i$ on the n-component was Π_2. Let s be the first stage at which $\tilde{p}_{\ell_{max}[n]}(x)$ is computed, for $x \in \mathsf{C} \subseteq \mathcal{B}$. Then $\hat{\mathsf{C}}$ has not entered $\mathcal{M}_i$ by stage s.*

Proof. If $\hat{\mathsf{C}}$ had entered $\mathcal{M}_i$, $\mathcal{R}_i$ would have been declared to have outcome Σ_2 by the n-component. □

Next we will show that for each i, either $\mathcal{B} \leq_{pr} \mathcal{M}_i$ (if $\mathcal{R}_i$ had the Σ_2 outcome) or $\mathcal{M}_i \leq_{pr} \mathcal{A}$ (if $\mathcal{R}_i$ had the Π_2 outcome).

Lemma 2.12. *Suppose that ψ_i is an isomorphism from $\mathcal{A}$ onto $\mathcal{M}_i$ and θ_i is an isomorphism from $\mathcal{M}_i$ onto $\mathcal{B}$, and that the outcome of $\mathcal{R}_i$ is Σ_2. Then $\mathcal{B} \leq_{pr} \mathcal{M}_i$.*

Proof. We have to argue that there is a primitive recursive isomorphism $\mathcal{B} \to \mathcal{M}_i$. Let σ be the sequence of true outcomes of $\mathcal{R}_0, \dots, \mathcal{R}_{i-1}$. Fix K and ℓ such that for $k \geq K$ and $j < i$, $\Gamma_j[n_k] = \sigma(j)$, $\Gamma_i[n_k] = \Sigma_2$, and $\ell_i^\sigma[n_j] = \ell$.

We define an isomorphism $\psi\colon \mathcal{B} \to \mathcal{M}_i$ component-by-component. On the nth component, we have a number of different cases:

- If $n < n_K$, then we construct the isomorphism by non-uniformly mapping $\hat{\mathsf{C}}$ to C. For each tagged n-cycle D in $\mathcal{B}$, we map D to a tagged n-cycle in $\mathcal{M}_i$; we can find such a tagged n-cycle in a primitive recursive way by finding the image, under $\psi_i\colon \mathcal{A} \to \mathcal{M}_i$, of a tagged n-cycle in $\mathcal{A}$. While we do this, we must also make sure that each tagged n-cycle in $\mathcal{M}_i$ is the image of some tagged n-cycle in $\mathcal{B}$.

- If $n \geq n_K$ and $\ell_i \leq \ell$, then after we put C into $\mathcal{B}$, wait for one of the following to happen:
  - An n-cycle $\hat{\mathsf{C}}$ with $\theta_i(\hat{\mathsf{C}}) = \mathsf{C} \in \mathcal{B}$ enters $\mathcal{M}_i$.

    Map C to $\hat{\mathsf{C}}$. We map tagged n-cycles in $\mathcal{B}$ to tagged n-cycles in $\mathcal{M}_i$ as described above. If the n-component is ever homogenized, then we stop adding new tagged n-cycles to $\mathcal{A}$ and $\mathcal{B}$. In both $\mathcal{A}$ and $\mathcal{B}$, every n-cycle is tagged, and $\mathcal{A}$ and $\mathcal{B}$ have the same finite number of n-cycles. Thus after a primitive recursive delay, $\hat{\mathsf{C}}$ must be the image under θ_i of an n-cycle in $\mathcal{A}$ (which may not be $\mathsf{C} \subseteq \mathcal{A}$), and hence will receive a tag; we can then map the tag of $\mathsf{C} \subseteq \mathcal{B}$ to the tag of $\hat{\mathsf{C}}$.
  - For some $j < i$, $\sigma(j) = \Pi_2$ but an n-cycle $\hat{\mathsf{C}}$ with $\theta_j(\hat{\mathsf{C}})$ enters $\mathcal{M}_j$.

    We know that the n-component will be homogenized because its outcome will be to the right of the true outcome (its outcome cannot be to the left of the true outcome, or the n_K-component would be homogenized). So C will at some point receive a tag. So we can map each of C and each other tagged n-cycle D in $\mathcal{B}$ to any other tagged n-cycle in $\mathcal{M}_i$, in such a way that every tagged n-cycle in $\mathcal{M}_i$ is the image of one in $\mathcal{B}$. We can do this primitively recursively because we have, via the map $\psi_i \colon \mathcal{A} \to \mathcal{M}_i$, a primitive recursive list of tagged n-cycles in $\mathcal{M}_i$.

  We claim that one of these must happen by stage $\max_{x \in \mathsf{C} \subseteq \mathcal{B}} \tilde{p}_\ell(x)$. Suppose that up to stage $\max_{x \in \mathsf{C} \subseteq \mathcal{B}} \tilde{p}_\ell(x)$, there is no $j < i$ with $\sigma(j) = \Pi_2$ such that an n-cycle $\hat{\mathsf{C}}$ with $\theta_j(\hat{\mathsf{C}})$ enters $\mathcal{M}_j$; this means that no such $\mathcal{R}_j$ is declared to have the Σ_2 outcome in the n-component. For each $j < i$ with $\sigma(j) = \Sigma_2$, by Lemma 2.9 we know that $\ell_j^\sigma[n-1] \leq \ell = \ell_i^\sigma$. If there is some $j \leq i$ with $\sigma(j) = \Sigma_2$ such that by stage $\max_{x \in \mathsf{C} \subseteq \mathcal{B}} \tilde{p}_\ell(x)$ no n-cycle $\hat{\mathsf{C}}$ with $\theta_j(\hat{\mathsf{C}})$ has entered $\mathcal{M}_j$, then (by the contrapositive of Lemma 2.10) for the least such j, $\mathcal{R}_0, \ldots, \mathcal{R}_j$ would be declared to have outcome $\sigma{\restriction}_j \Pi_2$ in the n-component. This is to the left of the true outcome, which is the outcome of n_K, and so the n_K-component would be injured. Thus we can conclude that for each $j \leq i$ with $\sigma(j) = \Sigma_2$, by stage $\max_{x \in \mathsf{C} \subseteq \mathcal{B}} \tilde{p}_\ell(x)$, an n-cycle $\hat{\mathsf{C}}$ with $\theta_j(\hat{\mathsf{C}})$ has entered $\mathcal{M}_j$. In particular, this is true for $j = i$. So we have shown that one of the two possibilities above must occur by stage $\max_{x \in \mathsf{C} \subseteq \mathcal{B}} \tilde{p}_\ell(x)$.

Putting together these isomorphisms on each component, which are uniformly primitive recursive, we get a primitive recursive isomorphism $\mathcal{B} \to \mathcal{M}_i$. □

Lemma 2.13. *Suppose that ψ_i is an isomorphism from $\mathcal{A}$ onto $\mathcal{M}_i$ and θ_i is an isomorphism from $\mathcal{M}_i$ onto $\mathcal{B}$, and that the outcome of $\mathcal{R}_i$ is Π_2. Then $\mathcal{M}_i \leq_{pr} \mathcal{A}$.*

Proof. We have to argue that there is a primitive recursive isomorphism $\mathcal{M}_i \to \mathcal{A}$. Fix K such that for $k \geq K$ and $j < i$, $\Gamma_j[n_k] = \sigma(j)$, $\Gamma_i[n_k] = \Pi_2$.

We define an isomorphism $\mathcal{M}_i \to \mathcal{A}$ component-by-component. On the nth component, we have two different cases:

- If $n < n_K$, then we construct the isomorphism by non-uniformly mapping $\hat{\mathsf{C}}$ to C. (As in the previous lemma, we map tagged n-cycles in $\mathcal{M}_i$ to tagged n-cycles in $\mathcal{A}$.)
- If $n \geq n_K$, we wait for an element $\hat{\mathsf{C}}$ with $\theta_i(\hat{\mathsf{C}}) = \mathsf{C}$ to appear in $\mathcal{M}_i$. While we wait, we have to map tagged n-cycles in $\mathcal{M}_i$ to tagged n-cycles in $\mathcal{A}$;

we can do this because we have a primitive recursive list of tagged n-cycle D in $\mathcal{A}$. When $\hat{\mathsf{C}}$ enters $\mathcal{M}_i$, we have two possibilities.

First, if C is already in $\mathcal{A}$, then we map $\hat{\mathsf{C}}$ to C. $\hat{\mathsf{C}}$ cannot receive a tag unless its image $\mathsf{C} = \theta_i(\hat{\mathsf{C}})$ in $\mathcal{B}$ does, which only happens if the n-component is homogenized; in this case, C also receives a tag in $\mathcal{A}$, and we can map the tag of $\hat{\mathsf{C}}$ to this tag.

Otherwise, if C is not already in $\mathcal{A}$, the outcome of the n-component for $\mathcal{R}_i$ is Σ_2, and so we know that the n-component will later be homogenized, and $\hat{\mathsf{C}}$ will receive a tag. So we just map $\hat{\mathsf{C}}$ to a tagged n-cycle in $\mathcal{A}$. When $\hat{\mathsf{C}}$ receives a tag, we map it to the tag of its image in $\mathcal{B}$.

When the n-component is homogenized, $\mathsf{C} \subseteq \mathcal{A}$ receives a tag, and we can find a new tagged n-cycle $\hat{\mathsf{D}}$ in $\mathcal{M}_i$ and map it to C in $\mathcal{A}$.

Putting together these isomorphisms on each component, which are uniformly primitive recursive, we get a primitive recursive isomorphism $\mathcal{M}_i \to \mathcal{A}$. □

Lemma 2.14. *Each requirement S_i is satisfied: γ_i is not an isomorphism from $\mathcal{B}$ to $\mathcal{A}$.*

Proof. If there is some n_j-component, $n_j \geq i$, such that each of the requirements $\mathcal{R}_1, \ldots, \mathcal{R}_{n_j}$ has the Σ_2 outcome, then $\mathsf{C} \in \mathcal{B}$ is an untagged n-cycle (since the n_j-component is never homogenized), and we do not add C to $\mathcal{A}$ until after γ_n has already become defined on C; so the image of C under γ_n cannot be an untagged n-cycle, and γ_n cannot be an isomorphism.

Otherwise, let i' be such that γ_i is dominated by $\tilde{p}_{i'}$. For every n_j-component, $n_j \geq i'$, some requirement $\mathcal{R}_k$ has the outcome Π_2. For sufficiently large j, $\ell_{max}[n_j] > i'$. This means that in the n_j-component, C is not added to $\mathcal{A}$ until after stage $\max_{x \in \mathsf{C} \subseteq \mathcal{B}} \tilde{p}_{i'}(x)$ (see Lemma 2.11), and so no element of C is among the first $\max_{x \in \mathsf{C} \subseteq \mathcal{B}} \tilde{p}_{i'}(x)$ elements of $\mathcal{A}$. Since $\tilde{p}_{i'}$ dominates γ_i, for $x \in \mathsf{C} \subseteq \mathcal{B}$, $\gamma_i(x)$ cannot be in $\mathsf{C} \subseteq \mathcal{A}$; and as the n_j-component is never homogenized C is the only untagged n_j-cycle. Thus γ_i is not an isomorphism. □

By Lemmas 2.12 and 2.13, each $\mathcal{R}$ requirement is satisfied. By Lemma 2.14, each $\mathcal{S}$ requirement is satisfied. This completes the proof of the theorem. □

References

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

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

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

[BDKM19] Nikolay Bazhenov, Rod Downey, Iskander Kalimullin, and Alexander Melnikov. Foundations of online structure theory. *Bull. Symb. Log.*, 25(2):141–181, 2019.

[BHTK+19] Nikolay Bazhenov, Matthew Harrison-Trainor, Iskander Kalimullin, Alexander Melnikov, and Keng Meng Ng. Automatic and polynomial-time algebraic structures. *The Journal of Symbolic Logic*, pages 1–32, 04 2019.

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

[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.

[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.

[DGM+] R. Downey, N. Greenberg, A. Melnikov, K.M. Ng, and D. Turetsky. Punctual categoricity and universality. *to appear in the Journal of Symbolic Logic.*

[DMN] Rod Downey, Alexander Melnikov, and Keng Meng Ng. Foundations of online structure theory ii: the operator approach. *Preprint.*

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

[GLS03] S. Goncharov, S. Lempp, and R. Solomon. The computable dimension of ordered abelian groups. *Adv. Math.*, 175(1):102–143, 2003.

[GMR89] S. S. Goncharov, A. V. Molokov, and N. S. Romanovskiĭ. Nilpotent groups of finite algorithmic dimension. *Sibirsk. Mat. Zh.*, 30(1):82–88, 1989.

[Gon80] S. Goncharov. The problem of the number of nonautoequivalent constructivizations. *Algebra i Logika*, 19(6):621–639, 745, 1980.

[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.

[Hir01] Denis R. Hirschfeldt. Degree spectra of intrinsically c.e. relations. *J. Symbolic Logic*, 66(2):441–469, 2001.

[HKSS02] D. Hirschfeldt, B. Khoussainov, R. Shore, and A. Slinko. Degree spectra and computable dimensions in algebraic structures. *Ann. Pure Appl. Logic*, 115(1-3):71–113, 2002.

[Kie98] 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.

[KMN17] 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.

[KMZ] I. Kalimullin, A. Melnikov, and M Zubkov. Punctual degrees and lattice embeddings. *to appear in proceedings of Aspects of Computation (World-Scientific).*

[KN08] Bakhadyr Khoussainov and Anil Nerode. Open questions in the theory of automatic structures. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, (94):181–204, 2008.

[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.

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

[McC02] Charles F. D. McCoy. Finite computable dimension does not relativize. *Arch. Math. Logic*, 41(4):309–320, 2002.

[MNa] A. Melnikov and K.M. Ng. A structure of punctual dimension two. *to appear in Proceedings of the American mathematical Society.*

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

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