AUTOMATIC AND POLYNOMIAL-TIME ALGEBRAIC STRUCTURES

NIKOLAY BAZHENOV, MATTHEW HARRISON-TRAINOR, ISKANDER KALIMULLIN, ALEXANDER MELNIKOV, AND KENG MENG NG

Abstract. A structure is automatic if its domain, functions, and relations are all regular languages. Using the fact that every automatic structure is decidable, in the literature many decision problems have been solved by giving an automatic presentation of a particular structure. Khoussainov and Nerode asked whether there is some way to tell whether a structure has, or does not have, an automatic presentation. We answer this question by showing that the set of Turing machines that represent automata-presentable structures is Σ^1_1-complete. We also use similar methods to show that there is no reasonable characterisation of the structures with a polynomial-time presentation in the sense of Nerode and Remmel.

1. Introduction

Effective mathematics is the study of algorithms (or the lack thereof) on standard mathematical objects such as groups, fields, graphs, or linear orders. To perform computations on a structure, one must have some kind of presentation of the structure that can be manipulated in some model of computation. One of the first examples of such presentations are the "explicitly given" fields of van der Waerden [vdW30] in the 1930's. In the 1950's and early 1960's, in work of Fröhlich and Shepherdson [FS56], Malcev [Mal61], and Rabin [Rab60], the following definition was solidified: a (Turing) computable presentation of a structure $\mathcal{A}$ is an isomorphic copy of $\mathcal{A}$ whose domain is the natural numbers and whose functions, relations, and constants are all Turing computable. For example, a finitely presented group is computable exactly if the word problem is solvable in the group. The study of computable structures has been very fruitful; see the books [AK00, EG00].

A computable presentation of a structure can be computationally highly inefficient. It is thus natural to ask when a computable structure can be transformed into a structure in which the operations can be computed within reasonable resource bounds. In the late 1980's, Nerode and Remmel [NR90] suggested the investigation of polynomial-time computable structures. These are computable structures in which the operations and relations are polynomial-time computable in the length of their input; the formal definition will be given later. For various results on polynomial-time structures see [CR92, Ala16, CDRU09, CR99]. In many natural classes every (Turing) computable algebraic structure has a polynomial-time copy. In the 1990s, Khoussainov and Nerode [KN95, KN94] gave a more restrictive definition of an automatic structure. A structure is automatic if its domain and relations are regular languages, i.e., recognised by a finite automaton. Having an automatic presentation is a very strong condition on a structure; for instance, it implies that the structure is computable in linear time and that the first-order theory of that structure is decidable [KN95, BG00]. One can use these techniques to give decision algorithms for Presburger arithmetic, the real numbers under addition, and atomless Boolean algebras, and there have even been applications to decision problems in chess [BHS12]. The study of automatic structures, and especially of automatic groups, has been particularly fruitful; see, e.g., [Tsa11, BS11, NS09, ECH$^+$92, KKM14, JKS$^+$17, JKS18].

Which algebraic structures admit an automatic presentation? Delhommé [Del04] showed that the automatic ordinals are exactly those below $\omega^{<\omega}$, and Oliver and Thomas [OT05] proved that a finitely

N. Bazhenov is supported by Russian Science Foundation, project No. 18-11-00028. M. Harrison-Trainor is supported by an NSERC Banting fellowship. The research of I. Kalimullin was funded by the subsidy allocated to Kazan Federal University for the state assignment in the sphere of scientific activities, project No. 1.1515.2017/4.6. Also I. Kalimullin is funded by the Russian Ministry of Education and Science (project No. 1.451.2016/1.4) as the federal professor in mathematics. A. Melnikov was supported by Massey University Research Fund and Marsden Fund of New Zealand. K. M. Ng is partially supported by grant MOE2015-T2-2-055.

generated group is automatic if and only if the group is virtually abelian; see [KM10] for more characterisations of this sort. What about polynomial-time presentations? It is not hard to show that *every* (Turing) computable linear ordering admits a polynomial-time presentation [Gri90], and the same can be said about any torsion-free abelian group (essentially [KMN17b], after Downey) and many other structures; see the survey [CR91]. Positive results of this sort suggest that perhaps there could be a general necessary and sufficient condition on a structure to have an automatic or at least a polynomial-time presentation.

In this paper, we use advanced tools of computability theory to show that there is no general reasonable necessary and sufficient condition on a structure to be automata presentable. In a similar vein, we also show that there is no reasonable necessary and sufficient condition on a structure to have a polynomial-time presentation. More formally, we prove that the complexity of the index sets of automatic structures is Σ^1_1-complete, and that the same holds for polynomial-time structures. The former answers Question 4.9 of Khoussainov and Nerode [KN08], and the latter answers Question 3.3 from the survey [Mel17].

Before we formally state and explain our theorems, we first clarify the terminology and discuss how computability theory approaches classification problems in mathematics.

1.1. **Index sets and classification problems.**

Suppose P is a certain property of an algebraic structure. For example, $P(A)$ could be stating that A is a free group, or that it is a well-founded partial order. Using a universal Turing machine, we can interpret every number as a (partial) computable structure and produce an effective listing of all (partial) computable structures. The complexity of the property is reflected in the complexity of its *index set*

$$I_P = \{i \in \omega \mid \text{the } i\text{th computable structure has property } P\}.$$

The latter can be formally measured using various hierarchies such as the arithmetical and the analytical hierarchies [Rog87, Soa87]. Such hierarchies consist of complexity classes which are the analogues in computability theory to classes such as P and NP in complexity theory. The classes correspond to the number and type of quantifiers required to solve the problem. We show some of the complexity classes below.

$$\begin{tikzcd}
 & \Sigma^0_1 \arrow[dr, phantom, sloped, "\subsetneq"] & & \Sigma^0_2 \arrow[dr, phantom, sloped, "\subsetneq"] & & \Sigma^0_3 \arrow[dr, phantom, sloped, "\subsetneq"] & & & & \Sigma^1_1 \arrow[dr, phantom, sloped, "\subsetneq"] & & \Sigma^1_2 \arrow[dr, phantom, sloped, "\subsetneq"] & \\
\Delta^0_1 \arrow[ur, phantom, sloped, "\subsetneq"] \arrow[dr, phantom, sloped, "\subsetneq"] & & \Delta^0_2 \arrow[ur, phantom, sloped, "\subsetneq"] \arrow[dr, phantom, sloped, "\subsetneq"] & & \Delta^0_3 \arrow[ur, phantom, sloped, "\subsetneq"] \arrow[dr, phantom, sloped, "\subsetneq"] & & \Delta^0_4 \arrow[ur, phantom, sloped, "\subsetneq"] \arrow[dr, phantom, sloped, "\subsetneq"] & \cdots & \Delta^1_1 \arrow[ur, phantom, sloped, "\subsetneq"] \arrow[dr, phantom, sloped, "\subsetneq"] & & \Delta^1_2 \arrow[ur, phantom, sloped, "\subsetneq"] \arrow[dr, phantom, sloped, "\subsetneq"] & & \cdots \\
 & \Pi^0_1 \arrow[ur, phantom, sloped, "\subsetneq"] & & \Pi^0_2 \arrow[ur, phantom, sloped, "\subsetneq"] & & \Pi^0_3 \arrow[ur, phantom, sloped, "\subsetneq"] & & & & \Pi^1_1 \arrow[ur, phantom, sloped, "\subsetneq"] & & \Pi^1_2 \arrow[ur, phantom, sloped, "\subsetneq"] &
\end{tikzcd}$$

For instance, if a decision problem falls into the complexity class Δ^1_1 then (essentially) it can be stated using only first-order quantifiers, while those in the classes Π^1_1 and Σ^1_1 may require a second-order quantification [Rog87, Soa87].

We give a few examples of index set results. The Σ^1_1 decision problems are those which can be decided with an existential second order quantifier. Just because a decision problem is in Σ^1_1 does not mean that it *necessarily requires* deciding an existential second order quantifier. For instance, it follows from [CHK$^+$12, MW12] that the index set of free groups I_{free} is merely Σ^0_5, while the brute-force bound gives Σ^1_1. This means that it takes only 5 *first-order* quantifiers over the natural numbers to test whether the ith structure is a free group. This low syntactical complexity reflects the algorithmic nature of freeness; recall that Nielson transformations [LS01] can be used to "calculate" a basis of a group (if the basis exists). In contrast, Riggs [Rig] has shown that the index set $I_{DirDecom}$ of directly decomposable abelian groups is Σ^1_1-*complete*. Recall that a group is directly decomposable if it can be split into the direct sum of its non-trivial subgroups; this brute-force definition is naturally Σ^1_1. From the computability-theoretic standpoint, the result of Riggs illustrates that *there is no reasonable characterisation of directly decomposable abelian groups* because there is no simpler way of checking that a group is decomposable than just testing the naive definition of decomposability. More formally, we will show that if $S \subseteq \mathbb{N}$ is a Σ^1_1 set, then there is a computable reduction f from S to $I_{DirDecom}$;

for some computable function f we have:

$$\text{for all } n: \qquad n \in S \iff f(n) \in I_{DirDecom}.$$

If such a reduction exists, then any decision procedure for $I_{DirDecom}$ would immediately yield one for S; so $I_{DirDecom}$ is at least as hard as S. Once we show that $I_{DirDecom}$ is Σ^1_1-complete, we know that there is no simpler way to check whether a particular computable abelian group $\mathcal{A}$ has a non-trivial direct decomposition rather than to ask the naive question: "Does there exist a pair of non-zero subgroups that split the group?" We give more examples in the table below.

Decision Problem	Complexity	Reference
Well-foundedness of a linear order	Π^1_1	[Spe55]
Atomicity of a Boolean algebra	Π^1_1	[GN02]
Decomposability of a group	Σ^1_1	[Rig]
Complete decomposability of a group	Σ^0_7	[DM14]
Isomorphism between finitely generated abelian groups	Σ^0_3	Folklore
Isomorphism between free groups	Π^0_4	[CHK$^+$12, MW12]
Isomorphism between vector spaces	Π^0_3	[DM08]
Isomorphism between torsion-free abelian groups	Σ^1_1	[DM08]

For instance, the classification by dimension leads to the decision whether two spaces over the same field are isomorphic being merely Π^0_3. Formally, if A_i is the ith computable structure, then the index set for *the isomorphism problem*

$$\{\langle i, j\rangle \in \omega \mid A_i, A_j \text{ are vector spaces over } F \text{ and } A_i \cong A_j\}$$

is Π^0_3-complete, where $\langle i, j\rangle = 2^i 3^j$. In contrast, it follows from [DM08] that there is no better way to check that two torsion-free abelian groups are isomorphic than to search through the uncountably many potential isomorphisms. From the perspective of computability theory, it follows that such groups are unclassifiable up to isomorphism. The abundance of "monstrous" examples of such groups in the literature [Fuc70, Fuc73, Hjo02, Tho03] strongly support this conclusion. Compare this to vector spaces, free groups, or completely decomposable groups which do possess convenient invariants.

As a systematic framework, this strategy of looking at index sets of computable structures originated in [GN02] and has been used in many other applications, see [LS07, Fok07, CFG$^+$07, FGK$^+$15, GBM15a, GBM15b].

1.2. **Results.** We return to the discussion of automatic and polynomial-time structures. As we noted above, Khoussainov and Nerode ([KN08], Question 4.9) asked for the complexity of the index set of automata-presentable structures

$$I_{Aut} = \{i \in \omega \mid \text{the } i\text{th computable structure has an automatic presentation}\}.$$

It is clearly in Σ^1_1: a computable structure $\mathcal{A}$ has an automatic presentation if and only if there exists an automatic structure $\mathcal{B}$ and an isomorphism f between $\mathcal{A}$ and $\mathcal{B}$.

In [KNRS07], it was shown that the isomorphism problem for automatic structures is Σ^1_1-complete. Khoussainov and Minnes [KM09] have also constructed automatic structures with arbitrarily complex invariants in the sense of high Scott rank. These results illustrate that automatic structures possess very complicated invariants, but these results do not give any insight into the the question of Khoussainov and Nerode. Compare this to the situation with groups. We know that the isomorphism problem for groups is Σ^1_1-complete [GN02], but the index set of groups is merely Π^0_2; all we need to check is the three simple axioms. However, we show that I_{Aut} is not simpler than Σ^1_1.

Theorem 1.1. *The index set*

$$I_{Aut} = \{i \in \omega \mid \textit{the } i\textit{th computable structure has an automatic presentation}\}$$

of automata presentable structures is Σ^1_1-complete.

Again, what this means is that being automata presentable is at least as hard as every other Σ^1_1 problem. It follows that there is no hope of finding a convenient characterisation of automatic structures in general, but there could be nice characterisations within large natural classes. For instance, what is the index set of automatic linear orders? What about groups? We leave these questions open.

Interestingly enough, initially the authors were not concerned with automatic structures. They were looking at the index set of *fully primitive recursive structures* [KMN17b]. These are computable structures in which the domain is ω and all operations and relations are primitive recursive. The latter means that unbounded search operators – such as REPEAT ... UNTIL ... – must be forbidden in the algorithm representing the structure. Although fully primitive recursive structures do not have to be computationally feasible, the class serves as an abstraction to study polynomial-time presentations in the following sense. Kalimullin, Melnikov and Ng [KMN17b] have observed that eliminating unbounded search from a Turing computable presentation of a structure is often the key step is showing that the structure has a polynomial-time presentation; see [KMN17b, KMN17a, Mel17] for various examples. On the other hand, to show that a structure does not have a polynomial-time presentation it is often easiest to prove it does not even have a fully primitive recursive presentation, as in [KMN17b, KMN17a, CR91, CR95]. The paper at hand is yet another non-trivial illustration of this important intermediate role of primitive recursion in computable algebra. The authors initially showed that the index set of fully primitive recursive structures is Σ^1_1-complete, and then they realised that the argument can be extended to prove Theorem 1.1 as well as the second main result below:

Theorem 1.2. *The index set*

$$I_{Poly} = \{i \in \omega \mid \textit{the ith computable structure has a polynomial-time presentation}\}$$

of polynomial-time presentable structures is Σ^1_1-complete.

It follows that there is no convenient necessary and sufficient condition for a computable structure to have a polynomial-time presentation. We note that both Theorem 1.2 and 1.1 will be witnessed by structures in some finite language which consists of at most binary predicates and unary functions.

The main results – rather, their proofs – have several pleasant corollaries. In [HT17], the second author proved that the index set of the decidably presentable structures is Σ^1_1-complete; Recall that a structure is *decidable* if its full first-order diagram (i.e., after naming all elements) is computable. From the proof of Theorem 1.2, which incorporates techniques from [HT17], one gets the same result but moreover that the Σ^1_1-outcome is witnessed by an automatic (thus, decidable) structure. Also, as we will discuss later, one technical lemma in the proof of Theorem 1.2 implies that Nurtazin's neat syntactical description of structures categorical with respect to decidable copies [Nur74] cannot be extended to Δ^0_2-categoricity with respect to decidable copies. This is a technical result which will be of interest to experts in computable structure theory; see Corollary 5.6. Also, as a consequence of the proof of Theorem 1.2, the index set of fully primitive recursive structures is Σ^1_1-complete. The latter answers Question 3.3 from [Mel17].

The reader should prepare for technically involved proofs. In Section 3 we will do our best trying to explain the main ideas behind the proofs. However, a solid background in computability theory will perhaps be necessary to understand the arguments in full depth. The proofs of both theorems (1.1 and 1.2) are based on the same general technical fact (Lemma 4.1) which extends the strategy that the second author used in [HT17] in the context of decidable structures. Another important component is a jump inversion theorem (see Theorem 5.1) that improves the key technical result of Khoussainov and Minnes [KM09] and gives a jump-inversion operator from the class of Δ^0_2-structures to automatic structures. On top of these technical tools, we will introduce a diagonalisation method which (amazingly enough!) works for both Theorem 1.1 and 1.2. This general diagonalisation strategy is described in Lemma 4.6. Instead of giving two separate but very similar proofs, we develop a framework. Then both main results will be derived as easy applications of this general apparatus.

The plan of the paper is as follows. Section 2 contains formal definitions and other preliminary material. In Section 3, we describe a labeling technique first used by Selivanov [Sel76] and Goncharov [Gon77], and we informally describe the proof for the case of fully primitive recursive structures. Section 4 contains two general technical facts, Lemma 4.1 and Lemma 4.6, that are used in the proofs of both main results. In Section 5, we give a jump-inversion operator from Δ^0_2-structures to automatic structures. Section 6 introduces another jump-inversion operator acting from Δ^0_2-structures to computable structures. This operator will be used by the diagonalisation strategy from Theorem 1.1. Following this, in Section 7 we prove Theorem 1.1 on automatic structures. Section 8 contains the proof of Theorem 1.2 on polynomial-time structures.

2. Different Kinds of Presentations of Structures

A computable structure (in a computable language $\mathcal{L}$) is one whose domain is the set of natural numbers that we denote ω, and whose basic relations are all uniformly Turing-computable sets of tuples from ω. For other kinds of presentations, we tend to think of the domain of the structure as the set of finite words Σ^* in a finite language Σ. (Using a natural coding, we can identify ω and Σ^*.)

2.1. **Polynomial-time computable structures.**

Let Σ be a finite alphabet. We say that a function $f\colon (\Sigma^*)^n \to \Sigma^*$ is polynomial-time computable if there is a polynomial p and a multitape Turing machine which computes $f(\bar{x})$ in time $p(\ell)$, where ℓ is the length of the input. For a set $S \subseteq (\Sigma^*)^n$, we say that S is polynomial-time computable if its characteristic function is. In this paper we are mainly concerned with structures in a finite language. Let $\mathcal{L} = \{\{R_i\}_{i\in S}, \{f_i\}_{i\in T}, \{c_i\}_{i\in U}\}$ be a finite computable language.

Definition 2.1 ([NR90]). *A structure*

$$\mathcal{A} = (A, \{R_i^{\mathcal{A}}\}_{i\in S}, \{f_i^{\mathcal{A}}\}_{i\in T}, \{c_i^{\mathcal{A}}\}_{i\in U})$$

is polynomial-time computable if

- *the domain A is equal to Σ^*,*
- *each $R_i \subseteq (\Sigma^*)^{s(i)}$ is a polynomial-time computable relation, and*
- *each $f_i\colon (\Sigma^*)^{t(i)} \to \Sigma^*$ is a polynomial-time computable function.*

However, the domain of the polynomial-time structure might be defined as a proper polynomial-time subset of Σ^*, and this leads to a non-equivalent definition [CR91]. For this particular paper there will be no difference which definition we pick, because our result on polynomial-time structures will remain true for both. We chose the definition above only to be consistent with the definition below.

Definition 2.2. [KMN17b] *A structure*

$$\mathcal{A} = (A, \{R_i^{\mathcal{A}}\}_{i\in S}, \{f_i^{\mathcal{A}}\}_{i\in T}, \{c_i^{\mathcal{A}}\}_{i\in U})$$

is fully primitive recursive if

- *the domain A is equal to ω,*
- *$R_i \subseteq (\Sigma^*)^{s(i)}$ are relations with primitive recursive characteristic functions, and*
- *$f_i\colon (\Sigma^*)^{t(i)} \to \Sigma^*$ are primitive recursive functions.*

2.2. **Automatic Structures.**

Definition 2.3. *Let Σ be a finite alphabet. A word automaton is a tuple $\mathcal{M} = (S, \imath, \Delta, F)$ where S is a finite set of states, $\imath$ is the initial state, $\Delta \subseteq S \times \Sigma \times S$ is the transition table, and $F \subseteq S$ is the set of accepting states.*

An automaton can be presented as a finite labeled graph whose nodes are the states S, and there is an edge labeled σ between s and s' if and only if $(s, \sigma, s') \in \Delta$. The input is a finite string $w \in \Sigma^*$. Given $w = \sigma_0\sigma_1 \cdots \sigma_{n-1}$, a run of the automaton is a sequence $q_0, q_1, \cdots, q_n$ where q_0 is the initial state, and $(q_i, \sigma_i, q_{i+1}) \in \Delta$ for all i. The run is accepting if q_n is in the set F of accepting states. The automaton accepts w if there is at least one accepting run. A word w may have no runs, in which case it is not accepted, or more than one run.

Definition 2.4. *A subset $L \subseteq \Sigma^*$ is called a regular language if there is an automaton $\mathcal{M}$ such that L is the set of finite strings accepted by $\mathcal{M}$.*

The collection of regular languages is closed under union, intersection, and complements.

We want to define regular relations $R \subseteq (\Sigma^*)^n$. To do this, we must define the operation of convolution. Given $x_1, \ldots, x_n \in \Sigma^*$, all of the same length, the ith element of their convolution is $\langle x_1(i), \ldots, x_n(i) \rangle$. If $x_1, \ldots, x_n$ are of different lengths, first pad them by appending a new symbol $\diamond$ to make them have the same length. We denote the convolution by $conv(x_1, \ldots, x_n)$; it is a string over the alphabet $(\Sigma \cup \{\diamond\})^n$.

Definition 2.5. *$R \subseteq (\Sigma^*)^n$ is called a regular relation if its convolution*

$$conv(R) = \{conv(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n) \in R\}$$

is regular.

We are now ready to define what it means for a structure to be automatic. An automatic structure is always in a finite relational language.

Definition 2.6 ([KN95])**.** *A structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is automatic if its domain A is a regular language and its relations $R_1, \ldots, R_n$ are regular relations. $\mathcal{A}$ is automatic presentable if it is isomorphic to an automatic structure.*

We note that Hodgson [Hod82] had defined the notions of an automata decidable theory and automatic structure before Khoussainov and Nerode, but his work went largely unnoticed until the paper of Khoussainov and Nerode [KN95] in 1995, when the systematic study of automatic structures took off.

Example 2.7. *Presburger arithmetic is automatic presentable. Its automatic presentation $(\{0,1\}^* \cdot 1, +, \leq)$ uses binary representation, with the least significant bit first. The standard method of adding two numbers in binary uses a single carry bit and is thus automatic.*

Recall from the introduction that one of the reasons for studying automatic presentations is that there is a decision procedure for them.

Theorem 2.8 ([KN95], [BG00])**.** *There is an algorithm that, given an automatic structure $\mathcal{A}$, an elementary first-order formula $\varphi(x_1, \ldots, x_n)$, and a tuple $a_1, \ldots, a_n$, decides whether $\mathcal{A} \models \varphi(a_1, \ldots, a_n)$.*

For more results on automatic structures, the reader is referred to, e.g., [KN95, BG00, KNRS07, KR03, KM10, KLM09].

3. Intuition

3.1. **The label technique.** Perhaps the first implicit use of the technique can be found in Selivanov [Sel76] and Goncharov [Gon77], both in the context of numbering theory. The technique has recently found significant applications in computable structure theory, perhaps most notably in [DKL$^+$15] where it was used to prove Π^1_1-completeness of computable categoricity. In this subsection we briefly outline the main idea of the technique.

We will also use a game-theoretic approach to constructing our structures. In the construction, will have to meet an infinite collection of *requirements*, which are just certain properties which, combined, imply the desired Σ^1_1-completeness of the index set. Each of the requirement will be associated with a set of instructions. This set of instructions can be viewed as winning *strategy* for a Lachlan game [Lac70]. In a Lachlan game both the player and the opponent builds a sequence of c.e. sets, see [Lac70] for a detailed explanation. In our proof sketch the player (us) builds parts of the structure and tries to meet the requirements. The opponent (the adversary) builds the uniform total list of all primitive recursive (or polynomial-time, or automatic) structures. The player also tries to guess on what the opponent builds. Clearly, just guessing would not suffice. The player will implement the above mentioned label technique to "press" the opponent (to be explained below). The formal proof will then boil down to arguing that the player has a uniformly computable winning strategy.

Fix a uniformly computable list $(M_e)_{e \in \omega}$ of all (partial) computable structures, perhaps only of those that possess some specific nice property (for example, primitive recursive or polynomial-time computable). Imagine that we have to build a computable structure M. Among other requirements, we will attempt to meet:

$$M \cong M_e \implies M \cong_{\Delta_1^0} M_e,$$

for every e, where $\cong$ stands for isomorphism and $\cong_{\Delta_1^0}$ for computable isomorphism. Although we may fail to satisfy this requirement, our strategy will typically allow us to push down the complexity of guessing whether $M \cong M_e$ from Σ_1^1 to arithmetical. We usually aim to satisfy the requirement only at some restricted part P_e of M specifically reserved for the index e (or for some other purpose). This P_e will, for example, consist of elements that satisfy some unary predicate (that we may also denote P_e). So we really attempt to meet

$$M \restriction_{P_e} \cong M_e \restriction_{P_e} \implies M \restriction_{P_e} \cong_{\Delta_1^0} M_e \restriction_{P_e},$$

for each e.

The main idea is as follows. Suppose at stage s we have built a finite segment $M[s] \restriction_{P_e}$ of $M \restriction_{P_e}$ and a (partial) isomorphism $f_s : M_e[s] \restriction_{P_e} \to M[s] \restriction_{P_e}$. At the next stage $s+1$ we would like to extend the domain of $M[s] \restriction_{P_e}$ by (say) one extra element a and define the operations and relations on this new element to obtain $M[s+1] \restriction_{P_e}$. We could have just adjoined a to $M[s] \restriction_{P_e}$ without any extra care, but then $M_e \restriction_{P_e}$ might respond by producing an isomorphic copy of $M[s+1] \restriction_{P_e}$ with a new isomorphism $g : M_e[s+1] \restriction_{P_e} \to M[s+1] \restriction_{P_e}$ that does not extend the previously defined f_s. To satisfy the requirement, however, we must prevent M_e from destroying our approximation f_s. This is done as follows. We make sure that our approximation of $M \restriction_{P_e}$ satisfies the peculiar *local rigidity property*:

There exists only one possible isomorphic embedding of $M[s] \restriction_{P_e}$ into $M[s+1] \restriction_{P_e}$.

To enforce this property we use *labels*. A typical label in the literature would be some finite unique configuration attached to (or somehow associated with) every element x of $M[s] \restriction_{P_e}$ (and $M[s+1] \restriction_{P_e}$). For example, we could use a binary relation L and create an L-cycle of some specific size $k[a, s+1]$, and connect it to a. Using a single binary relation will later allow us to keep our language finite. Another simple enough approach would involve using a fresh unary predicate U_a and set $U_a(a) = 1$ and $U_a(x) = 0$ for all x currently in M. Although later we may add more points to M which are labeled by U_a, at this stage the finite part of our structure is rigid.

The opponent controlling M_e must respond by giving us the unique possible extension of f_s that we use to define f_{s+1}, otherwise $M_e \restriction_{P_e} \not\cong M \restriction_{P_e}$. If M_e responds by giving some other configuration (or gives us too many elements) we will keep the configuration $M[s+1] \restriction_{P_e}$ untouched and will perhaps restart the other requirements in some other location of M using another unary predicate P_j. (The exact strategy will depend on what exactly we are trying to do with M.)

It is crucial that the uniqueness of $M[s] \restriction_{P_e} \hookrightarrow M[s+1] \restriction_{P_e}$ for every s does not necessarily imply that $M \restriction_{P_e} = \bigcup_{s\in\omega}(M[s] \restriction_{P_e})$ is rigid, as we may end up with all elements of $M \restriction_{P_e}$ labeled by the same collection of labels in the limit. Although every element of $M[s] \restriction_{P_e}$ participates only in a finite collection of labels unique to this element at stage s, this same element in $M \restriction_{P_e}$ may carry infinitely many labels after the construction is finished. It is perfectly possible that all elements can carry the exact same infinite collection of labels in $M \restriction_{P_e}$, and in fact we can arrange it so that this set of length of loops is computable uniformly in e.

Example 3.1. For simplicity, suppose P_e contains only points with no extra relations on the points. We identify a loop of size $n+1$ (or the n'th unary predicate) with label $\boxed{n}$. We will label each point of P_e by $\boxed{n}$, for *every* n, assuming that $M_e \restriction P_e$ is actually isomorphic to our $M \restriction P_e$ and always faithfully responds to our actions. This is done as follows.

0. Start by introducing x_0 with $\boxed{0}$ on it. Wait for M_e to respond by giving y_0 labeled by $\boxed{0}$.
1. Introduce a new element x_1 and put $\boxed{1}$ on it. Wait for M_e to respond.

Whenever M_e responds by copying us we declare the respective stage *e-expansionary*. Our local goal is to label x_0 by $\boxed{1}$ and x_1 by $\boxed{0}$. However, we can't do it now since M_e might swap its y_0 with y_1 by (secretly) declaring y_1 to be the image of x_0, while we want y_0 to be the actual image of x_0. We can achieve our local goal as follows.

2. Put $\boxed{2}$ onto x_0 and $\boxed{3}$ onto x_1. Wait for M_e to respond by putting $\boxed{2}$ onto y_0 and $\boxed{3}$ onto y_1.
3. Put $\boxed{1}$ onto x_0 and $\boxed{0}$ onto x_1. Wait for M_e to respond by putting $\boxed{1}$ onto y_0 and $\boxed{0}$ onto y_1.

Note that M_e has to follow us; otherwise it is "dead". We could continue in this manner and use $\boxed{m}$ and $\boxed{m+1}$ and put $\boxed{n}$ (for every n) onto both x_0 and x_1. Call this procedure applied to x_0 and x_1 *the limit homogenisation.*

At any later stage we can introduce a new element x_2 (similarly, x_k if we already have $x_0, \dots, x_{k-1}$) as follows.

2. Put x_2 labeled by $\boxed{m}$, where m is fresh and large. Wait for M_e to copy us.
3. Resume the procedure of local homogenisation, but this time apply it to x_0, x_1 and x_2 (not only x_0, x_1).

Note that at every stage s there exist a unique embedding of $M[s] \restriction_{P_e}$ into $M[s+1] \restriction_{P_e}$, as desired. In particular, the opponent's structure M_e cannot reset its embedding at any intermediate stage. Note that all the x_i are automorphic in the limit structure $M \restriction_{P_e}$.

Depending on what exactly we are trying to prove, the label technique can be modified. For example, the uniqueness of $M[s] \restriction_{P_e} \hookrightarrow M[s+1] \restriction_{P_e}$ can be sacrificed by homogenising some parts of the structure at intermediate stages, for the sake of initialisation. But, regardless, the complexity of the guessing on $M_e \restriction_{P_e} \cong M \restriction_{P_e}$ can be significantly simplified.

3.2. **A rough idea that will not work.** We now turn to the discussion of Theorem 1.1 and 1.2. In fact, it is easier to think about the proof of 1.2 but not worry about calculating the time bounds. Thus, for simplicity, let us think about *the index set of fully primitive recursive structures.* The general outline of the strategy described below comes from that used in [HT17].

Let $(L_e)_{e\in\omega}$ be a list of all partially computable linear orders. Fix a Σ^1_1-complete set S, and fix a total computable f such that $L_{f(x)}$ is well-ordered iff $x \notin S$. We can arrange it so that $x \in S$ always implies $L_{f(x)}$ has a fixed isomorphism-type H, the Harrison linear order [Har68].

Even in the polynomial-time proof we will be diagonalising against all primitive recursive structures. Recall that the primitive recursive structures can be uniformly computably listed, but this list $(N_e)_{e\in\omega}$ is itself not primitive recursive.

Remark 3.2. We could replace this list by the list of all polynomial-time structures, or some infinite list of total structures; this would not affect the proof. It will be highly instructive to separate the exact properties of the class that we diagonalise against from the apparatus of the label technique. For the sake of this separation, we invite the reader to assume that $(N_e)_{e\in\omega}$ is a list of all structures that possess some "nice" property $\mathcal{P}$ that is not necessarily "polynomial time"; for example, it could stand for 1-decidable, primitive recursive, or decidable. This approach will help the reader to understand the more involved case of automatic structures. Many ideas below will still apply for such a "nice" property, regardless what exactly it is.

We need to build an M and also satisfy:

$$\Sigma^1_1: \text{ if } x \in S \text{ then } M \text{ has a fully primitive recursive presentation,}$$

and

$$\Pi^1_1: \text{ if } x \notin S \text{ then } (\forall e)\, M \ncong N_e.$$

We will be working with N_e within a separate component of M, call it M^e.

For now, fix some naive diagonalization strategy $\mathcal{D}$ that would be sufficient to diagonalise against a single N_e, in isolation. We assume that we can list isomorphic copies of all the possible outcomes of $\mathcal{D}$ in a "nice" way. Let $(\mathrm{D_n})_{\mathrm{n}\in\omega}$ be such a list.

Example 3.3. For example, it is easy to imagine a strategy that builds a computable M in the language of one functional symbol and guarantees $M \ncong N_e$ for a single e, where N_e is primitive recursive. Simply put a point and wait for N_e to either produce a very long chain or a loop, and we do something that would be different; because N_e must be primitive recursive, and M must simply be computable, we can do this. Clearly, we can uniformly primitively recursively produce a list of all possible isomorphism types of M that can be built by the strategy, even though the particular M we build is not primitive recursive. This strategy will not be sufficient at the end, but it is good enough as an example.

We go back to the Σ^1_1-diagonalization. The very naive, clearly incorrect, but still useful idea can be described as follows.

In M^e, we reserve a bunch of fully primitive recursive (in fact, polynomial-time) copies of the Harrison order H ([Gri90]) that will be put into separate "boxes" (aka locations) in M^e. Denote such boxes by $\boxed{H}$. To each such "square" box $\boxed{H}$ and for each n, attach another special "round" box $\textcircled{\mathrm{D_n}}$ with the outcome structure $\mathrm{D_n}$. This way we can form infinitely many *double-boxes* $\boxed{H}$-$\textcircled{\mathrm{D_n}}$, with infinitely many copies of such a double-box for every fixed n. This collection is uniformly polynomial-time, which we denote by J^e. (J of course, stands for "junk".)

We will also build another, *special* component of the form $\boxed{L_{f(x)}}$-$\textcircled{\mathrm{D}}$, where $L_{f(x)} \cong H$ iff $x \in S$, and where D will be implementing the naive diagonalization strategy $\mathcal{D}$ against a corresponding round box of N_e. If $x \in S$ then, no matter what we do, $J^e \cup \boxed{L_{f(x)}}$-$\textcircled{\mathrm{D}} = J^e \cup \boxed{H}$-$\textcircled{\mathrm{D_n}}$ for some n, and therefore $M^e = J^e \cup \boxed{H}$-$\textcircled{\mathrm{D_n}} \cong J^e$ which is primitive recursive. Therefore, as $M^e \cong J^e$, M^e will have a primitive recursive presentation J^e (in fact, polynomial-time), with all possible uniformity.

Remark 3.4. *Note that the special component itself does not have to be primitive recursive.*

If $x \notin S$ then $L_{f(x)} \cong \alpha$ for some computable ordinal α. In particular, $\boxed{L_{f(x)}}$-$\textcircled{\mathrm{D}} = \boxed{\alpha}$-$\textcircled{\mathrm{D}}$, and since H is not well-ordered, we have that the special component $\boxed{\alpha}$-$\textcircled{\mathrm{D}}$ is stable under automorphisms of M. In other words, $\boxed{\alpha}$ *uniquely determines the position of the diagonalization spot* D *in* M *when* $x \notin S$ (the Π^1_1-outcome). Thus, *if we knew the image of* $\boxed{\alpha}$ *in* N_e we'd be able to diagonalise against N_e using the naive strategy $\mathcal{D}$ applied to the respective round boxes.

Clearly, we don't know which of the components of N_e is isomorphic to $\boxed{\alpha}$ (if there is any at all). The naive diagonalization strategy $\mathcal{D}$ that builds D is a low-level arithmetical strategy and cannot possibly handle such a great potential injury. Thus, the strategy is far from being successful in the Π^1_1 case (when $x \notin S$), but at least we can hopefully handle the Σ^1_1-outcome ($x \in S$).

3.3. **A partial fix.** We now describe another idea that will allow us to partially fix the Π^1_1-outcome. The idea is to mix the rough strategy above with the label technique informally described in Subsection 3.1. We will adopt the idea from Example 3.1 and build the component M^e of M monitoring N_e and carefully placing arrays of labels onto elements of M^e in the right order. It is important that we can add the labels in a primitive-recursive way. For every double-subcomponent $\boxed{L}$-$\textcircled{S}$ (no matter what the contents of the boxes are), only the elements within the square box will be labeled, and it will be done along the lines of the procedure described in Example 3.1. The substructure S within any such round box will never be labeled. We define the notion of an *e-expansionary stage* accordingly (see Example 3.1).

Remark 3.5. In the definition of an e-expansionary stage, the behaviour of N_e within its various round boxes will be ignored. It is crucial that the notion of an e-expansionary stage depends only on whether N_e copies M within its square boxes.

Now, if there are infinitely many e-expansionary stages then all elements in all square boxes – including the square box of the special component $\boxed{L_{f(x)}}$-$\textcircled{\mathrm{D}}$ – end up labeled by *all* labels from an infinite computable set of labels, just as in Example 3.1. In particular, all the analysis from the previous subsection still applies to this new construction *but only if there exist infinitely many e-expansionary stages*. In particular, the Σ^1_1-outcome is still fine.

Also, provided that there exist infinitely many e-expansionary stages, the Π^1_1-outcome looks far less complicated. We can *search* for the least k such that the k'th double-box in N_e copies our special component. (Every double-box can be assigned with a number under some natural effective listing of all double-boxes; the listing depends on the enumeration of N_e.)

Once N_e starts copying the square box of the special component, it can never switch to copying some other square box. This is simply because our computable approximation of the square boxes in $M \upharpoonright_{P_e}$ is locally rigid in the sense of Subsection 3.1, as ensured by the labels. Also, in the Π^1_1 case the square box of the special component contains a labeled copy of an ordinal α, and therefore the respective box in N_e must also contain this same labeled copy of α, unless N_e is not isomorphic. (If $N_e \not\cong M$ then we win by luck). Note that the important point here is that because of the way we label and grow the elements of the square box of the special component, if N_e copies our special component, then the contents of the square box of N_e has to be actually isomorphic to the contents of our square box of the special component. At the end we will put a labeled linear ordering in the square box; this forces the contents of the square box of N_e to be also a copy of the same linear order, even though the linear order itself could be very complicated.

The guessing on such least k is clearly a finitary process. Once this k is found and stable, we can implement the naive diagonalization strategy in the round box of the special component in M by comparing it with the round box of the k'th double-box of N_e (see Example 3.3).

Nonetheless, there is still a *problem* with the Σ^1_1-outcome in the case when N_e eventually never responds, i.e., if there are only finitely many e-expansionary stages. Then we may end up with a *finite* (labeled) special component $\boxed{L_{f(x)}[s]}\!-\!\textcircled{\mathrm{D}[s]}$ that is not isomorphic to any other double-box that we have in J^e. We clearly don't know whether we are in the Π^1_1 or Σ^1_1-case, and in the latter case we want to be able to throw the special component away and guarantee $M^e \cong J^e$, where J^e actually *is* polynomial-time. This problem is easily fixed by introducing more junk components into J^e.

3.4. **A fix to the fix.** At every stage s at which we wait for N_e to respond, we will also introduce infinitely many fresh new copies of $\boxed{L_{f(x)}[s]}\!-\!\textcircled{\mathrm{D}[s]}$ and put them into J^e. This will destroy the local rigidity of $(M^e[s])_{s\in\omega}$, but it will fix the problem with the Σ^1_1-outcome discussed above.

Whenever we further expand the special component at a later stage, we "kill" its previous isomorphic copies (those isomorphic to $\boxed{L_{f(x)}[s]}\!-\!\textcircled{\mathrm{D}[s]}$) in J^e by labeling all their elements with a special label $\mathbf{k}$. This label will be reserved specifically for this purpose. No other elements of M^e (including, in particular, the elements of the special component) will ever be labeled by $\mathbf{k}$.

After the fix, we will actually be searching for the least k such that, for some s, (the square box of) the kth double-box in N_e copies *the isomorphic type* of (the square box of) our special component at every stage $t \geq s$. If $N_e \cong M$ then such a k must exist. However, this will potentially induce finitely many injuries to the diagonalization strategy $\mathcal{D}$. There also exists the possibility in which the strategy $\mathcal{D}$ is initialised infinitely often. In this case $N_e \not\cong M$, but this outcome must also be taken into consideration. Recall that in Example 3.3 we produced a uniform nice list of all potential outcomes of $\mathcal{D}$, for the sake of constructing the "nice" junk collector J^e. As long as we can come up with an appropriate diagonalization strategy that is sufficient to sustain finitely many injuries *and* has highly predictable outcomes, we will construct a primitive recursive J^e with all required uniformity.

The actual solution will require a more subtle local strategy whose infinitary outcome is hard to predict. However, with some care, we will still be able to produce a primitive recursive (in fact, polynomial-time) J^e. We stop the informal discussion here.

The extended sketch above can be pushed to a formal proof, and it is quite remarkable that we get Σ^1_1-completeness of polynomial-time structures almost for free out of it. All we need to do is to decide on our actions within a specified time bound, but this causes no grief. More care will be needed in the automatic case. The problem is that it is not clear which of our actions are automatic and which are not. To overcome this difficulty we will use two different "jump inversion" operators acting independently on round and square boxes, which are operators turning Δ^0_2 copies of a structure into computable structures in a different language. The images of these operators will in fact be "uniformly" regular (automatic). Most of our guessing will be performed at the level of $0'$ with almost no adjustment, this includes the procedure of placing labels, etc.

Because of these similarities between the proofs, it will be most convenient to develop a certain framework and then derive the theorems as applications of this technique. In our formal proof below we isolate the main feature of the Σ^1_1 guessing and the key idea of diagonalisation into two separate lemmas. We now give the formal details.

4. The two key lemmas

In this section, we give two key lemmas that will be used in the proof of the main results. The first one, Lemma 4.1, describes our strategy that deals with square boxes $\boxed{\mathcal{L}}$. In the case of Π^1_1 outcome, the lemma helps us to choose a square box $\boxed{\mathcal{L}}$ inside the opponent's structure. Then the corresponding double box $\boxed{\mathcal{L}}$-$\bigcirc\!\!\!\!\mathcal{S}$ will be used to defeat the opponent (i.e. to ensure that our structure and the opponent's structure are not isomorphic).

The second key lemma, Lemma 4.6, describes our work with the round boxes $\bigcirc\!\!\!\!\mathcal{S}$: this is the groundwork for defeating an opponent after choosing an appropriate double box.

Let $\{\mathcal{A}_i\}_{i\in\omega}$ and $\{\mathcal{B}_j\}_{j\in\omega}$ be sequences of L-structures. We say that the sequences $\{\mathcal{A}_i\}_{i\in\omega}$ and $\{\mathcal{B}_j\}_{j\in\omega}$ are *equal up to isomorphism* if the following conditions hold:

a) For any $i \in \omega$, there is some $j \in \omega$ such that $\mathcal{B}_j \cong \mathcal{A}_i$.
b) For any $j \in \omega$, there is $i \in \omega$ such that $\mathcal{A}_i \cong \mathcal{B}_j$.

it is well-known that graphs are computably universal in the sense that every computable structure can be imitated by a graph (folklore). Although the lemma below is stated for graphs, it will be sufficient for our more general purposes.

Lemma 4.1. *Given computable infinite linear orders $\mathcal{L}$, $\mathcal{H}$, and a computable sequence of (partial) computable graphs $\{\mathcal{B}_i\}_{i\in\omega}$, one can effectively construct a computable function $f(x)$ and a computable sequence of graphs $\{\mathcal{A}_i\}_{i\in\omega}$ with the following properties:*

a) *If $\mathcal{L} \cong \mathcal{H}$, then there is a non-zero i such that $\mathcal{A}_i$ and $\mathcal{A}_0$ are isomorphic.*
b) *Suppose that $\mathcal{L} \not\cong \mathcal{H}$, and the sequences $\{\mathcal{A}_i\}_{i\in\omega}$ and $\{\mathcal{B}_j\}_{j\in\omega}$ are equal up to isomorphism. Then for any $i > 0$, we have $\mathcal{A}_i \not\cong \mathcal{A}_0$. Furthermore, there is a finite limit $k := \lim_s f(s)$, and $\mathcal{A}_0 \cong \mathcal{B}_k$.*

Proof. Without loss of generality, we may assume that for any s, the structures $\mathcal{L}[s+1] \setminus \mathcal{L}[s]$ and $\mathcal{H}[s+1] \setminus \mathcal{H}[s]$ are one-element. We also assume that $\mathcal{L}[0] = \mathcal{H}[0] = \emptyset$.

For any $i \in \omega$, the structure $\mathcal{A}_i$ will be a labeled linear order, i.e. a linear order such that each of its elements is labeled by various labels $\boxed{n}$. We can implement the labels in various ways, for instance use an infinite array of unary predicates. However, we intend o keep our language finite. Thus, let $\mathcal{A}$ be a graph with a unary predicate, inside of which is a linear order, and to identify a loop of size $(n+2)$ with the label $\boxed{n}$. If we remove all the labels from $\mathcal{A}_i$, leaving just the linear order, the resulting structure will be denoted by $R(\mathcal{A}_i)$. Our structures $\mathcal{A}_0$ and $\mathcal{A}_1$ will have $R(\mathcal{A}_0) \cong \mathcal{L}$ and $R(\mathcal{A}_1) \cong \mathcal{H}$.

At a stage s, an element a from $\mathcal{A}_0[s] \cup \mathcal{A}_1[s]$ will have two labels $\boxed{prim(a,s)}$ (the *primary* label) and $\boxed{sec(a,s)}$ (the *secondary* label) which are unique to a. Furthermore, there will be a *bag* of labels: all of the elements of $\mathcal{A}_0[s] \cup \mathcal{A}_1[s]$ will have all labels from the bag.

We fix a computable list of special *killing* labels $\boxed{\mathtt{k}_i}$, $i \in \omega$. If an element of $\mathcal{A}_j[s]$ is labeled by a killing label $\boxed{\mathtt{k}_i}$, then the structure $\mathcal{A}_j$ does not grow after stage s.

At a non-zero stage s, we build a sequence of finite structures $\mathcal{A}_0[s]$, $\mathcal{A}_1[s]$, $\ldots$, $\mathcal{A}_s[s]$ such that $R(\mathcal{A}_0[s]) = \mathcal{L}[s]$ and $R(\mathcal{A}_1[s]) = \mathcal{H}[s]$. At each stage s, some of the finite structures $\mathcal{A}_i[s]$, $i \geq 2$, will be copying $\mathcal{A}_0[s]$, and the others will be labeled by a killing label. We also define a finite non-empty set $scope(s)$. For $i \in scope(s)$, we introduce an auxiliary parameter $flag(i,s) \in \{0, 1, \mathrm{k}\}$. The intuition for the parameter is the following:

(1) If $flag(i,s) \in \{0,1\}$, then we think that $\mathcal{B}_i[s]$ is isomorphic to $\mathcal{A}_{flag(i,s)}[s]$.

(2) If $flag(i,s) = \mathrm{k}$, then $\mathcal{B}_i[s]$ has received a killing label.

We say that a finite function g is a *correct partial isomorphism* from $\mathcal{A}_i[s]$ into $\mathcal{B}_j[t]$ if g satisfies the following conditions:

(1) $dom(g) \subseteq R(\mathcal{A}_i[s])$.
(2) $g \neq \emptyset$ and g is an isomorphic embedding from $dom(g)$ into $R(\mathcal{B}_j[t])$.
(3) Suppose that $a \in dom(g)$. Then a is labeled by a label $\boxed{n}$ in $\mathcal{A}_i[s]$ iff $g(a)$ is labeled by $\boxed{n}$ in $\mathcal{B}_j[t]$.

At a stage s, we define a sequence of finite functions $g_i[s]$, $i \in scope(s)$, such that:

- If $flag(i,s) \in \{0,1\}$, then $g_i[s]$ satisfies the first two conditions from the definition of a correct partial isomorphism from $\mathcal{A}_{flag(i,s)}[s]$ into $\mathcal{B}_i[s]$.
- If $flag(i,s) = \mathrm{k}$, then $g_i[s] = \emptyset$.

The functions $g_i[s]$ witness our convictions about the isomorphism types of $\mathcal{B}_i[s]$.

Construction.

Stage 0. Set $\mathcal{A}_0[0] = \mathcal{A}_1[0] = \emptyset$ and $scope(0) = \varnothing$.

Stage $s > 0$. We will extend $\mathcal{A}_i[s-1]$ to $\mathcal{A}_i[s]$ for $i < s$, and begin defining a new structure $\mathcal{A}_s[s]$. First, we add a new element to the domains of $\mathcal{A}_0$ and $\mathcal{A}_1$ as follows:

(1) We add a fresh element a into $R(\mathcal{A}_0)$ in order to make the orders $R(\mathcal{A}_0[s])$ and $\mathcal{L}[s]$ isomorphic.
(2) We choose two fresh non-killing labels $\boxed{m}$ and $\boxed{n}$ and label a with them. Set $prim(a,s) := m$ and $sec(a,s) := n$. Furthermore, we take every label from the bag and put it onto a.

For $\mathcal{A}_1[s]$, do the same thing except that we use $\mathcal{H}[s]$ in place of $\mathcal{L}[s]$.

What we do next depends on whether we are in an *expansionary* stage or not. We say that stage s is expansionary if it satisfies the following conditions:

(A) Suppose that $i \in scope(s-1)$ is such that $flag(i,s-1) \in \{0,1\}$, and no element from $\mathcal{B}_i[s]$ has a killing label. Let a be the least element from $\mathcal{A}_{flag(i,s-1)}[s-1] \setminus dom(g_i[s-1])$ and b be the least element from $\mathcal{B}_i[s] \setminus ran(g_i[s-1])$. Then there exists a correct partial isomorphism h from $\mathcal{A}_{flag(i,s-1)}[s-1]$ into $\mathcal{B}_i[s]$ such that $g_i[s-1] \subseteq h$, $a \in dom(h)$, and $b \in ran(h)$. Denote such a function h (with the least Gödel number) by $h_i[s]$.
(B) Let i be the least number not in $scope(s-1)$. If $\mathcal{B}_i[s]$ does not contain a killing label, then there exists a correct partial isomorphism g from some $\mathcal{A}_j[s-1]$, $j \in \{0,1\}$, into $\mathcal{B}_i[s]$. Note that by definition $dom(g) \neq \varnothing$.

There is no condition on those $\mathcal{B}_i[s]$ which contain a killing label.

If the stage s is expansionary, then we proceed as follows:

(i) *Renewing labels.* For every $i < s$, $i \notin \{0,1\}$, if $\mathcal{A}_i[s-1]$ does not yet have a killing label, then choose a fresh killing label $\boxed{\mathtt{k}_i}$ and put it onto every element of $\mathcal{A}_i[s-1]$. For every element $a \in \mathcal{A}_0[s-1] \cup \mathcal{A}_1[s-1]$, put its primary label $\boxed{prim(a,s-1)}$ into the bag. Choose a fresh non-killing label $\boxed{fr(a)}$ and declare $prim(a,s) := sec(a,s-1)$ and $sec(a,s) := fr(a)$. If an element $a \in \mathcal{A}_0[s] \cup \mathcal{A}_1[s]$ is not yet labeled by some label $\boxed{m}$ from the bag, then put $\boxed{m}$ onto a.
(ii) *Redefining scope.* Let i be the least number not in $scope(s-1)$. Set $scope(s) := scope(s-1) \cup \{i\}$.
(iii) *Redefining flags.* We define $flag(i,s)$ and $g_i[s]$ for $i \in scope(s)$ as follows.
  - If some element a from $\mathcal{B}_i[s]$ has a killing label, set $flag(i,s) = \mathrm{k}$ and $g_i[s] = \varnothing$.
  - If $flag(i,s-1) \in \{0,1\}$, and $\mathcal{B}_i[s]$ does not contain killing labels, then set $flag(i,s) := flag(i,s-1)$ and $g_i[s] := h_i[s]$.
  - If $i \in scope(s) \setminus scope(s-1)$ (so that i is the least number not in $scope(s-1)$ as in (ii)), and $\mathcal{B}_i[s]$ does not contain killing labels, then because s is an expansionary stage, there is a unique $j \in \{0,1\}$ such that $h_i[s]$ was a correct partial isomorphism from $\mathcal{A}_j[s-1]$ into $\mathcal{B}_i[s]$. Set $flag(i,s) = j$ and $g_i[s] = h_i[s]$.

If s is not expansionary, then do not renew labels and do the following: If $i \notin \{0,1\}$ and $\mathcal{A}_i[s-1]$ does not have a killing label, then $\mathcal{A}_i[s-1]$ is identically equal to $\mathcal{A}_0[s-1]$. Extend $\mathcal{A}_i$ in such a way that $\mathcal{A}_i[s] \cong \mathcal{A}_0[s]$. Also set $\mathit{scope}(s) := \mathit{scope}(s-1)$. Define $\mathit{flag}(i,s)$ and $g_i[s]$ for $i \in \mathit{scope}(s-1)$ as follows.

- If some element a from $\mathcal{B}_i[s]$ has a killing label, set $\mathit{flag}(i,s) = \mathrm{k}$ and $g_i[s] = \varnothing$.
- If $\mathit{flag}(i,s-1) \in \{0,1\}$, and $\mathcal{B}_i[s]$ does not contain killing labels, then set $\mathit{flag}(i,s) := \mathit{flag}(i,s-1)$ and $g_i[s] := g_i[s-1]$.

In all cases, we define $\mathcal{A}_s[s]$ as a fresh isomorphic copy of $\mathcal{A}_0[s]$. We have now built the structures $\mathcal{A}_0[s], \mathcal{A}_1[s], \dots, \mathcal{A}_s[s]$.

Verification.

It is easy to verify the following:

Claim 4.2. *$R(\mathcal{A}_0) \cong \mathcal{L}$ and $R(\mathcal{A}_1) \cong \mathcal{H}$.*

Claim 4.3. *If $\mathcal{L}$ is isomorphic to $\mathcal{H}$, then there is some $i > 0$ such that $\mathcal{A}_i \cong \mathcal{A}_0$.*

Proof. First, assume that our construction has infinitely many expansionary stages. Since labels are renewed infinitely often, for any $a, b \in \mathcal{A}_0 \cup \mathcal{A}_1$, a and b possess the same labels. Since $R(\mathcal{A}_0) \cong \mathcal{L} \cong \mathcal{H} \cong R(\mathcal{A}_1)$, the structures $\mathcal{A}_1$ and $\mathcal{A}_0$ are isomorphic.

Now assume that s^* is the last expansionary stage of our construction. Notice that $\mathcal{A}_{s^*}[s^*]$ and $\mathcal{A}_0[s^*]$ are isomorphic. Furthermore, since every $s > s^*$ is not expansionary, $\mathcal{A}_{s^*}$ never obtains killing labels, and $\mathcal{A}_{s^*}[s] \cong \mathcal{A}_0[s]$. Therefore, $\mathcal{A}_{s^*}$ is isomorphic to $\mathcal{A}_0$. □

Claim 4.4. *Suppose that for any $\mathcal{B}_j$, there is some $\mathcal{A}_m$ isomorphic to $\mathcal{B}_j$. Then there are infinitely many expansionary stages.*

Proof. Assume that s^* is the last expansionary stage. After the stage s^*, the labels are never renewed. Moreover, for any $s \geq s^*$, we have $\mathit{scope}(s) = \mathit{scope}(s^*)$. Note the following: The construction ensures that almost every $\mathcal{A}_i$ is isomorphic to $\mathcal{A}_0$. Furthermore, if $i \notin \{0,1\}$, then either $\mathcal{A}_i \cong \mathcal{A}_0$, or $\mathcal{A}_i$ contains killing labels.

We claim that there is (a least) stage $s_0 > s^*$ such that any $i \in \mathit{scope}(s^*)$ satisfies the following:

(a) Suppose that $\mathit{flag}(i,s^*) \in \{0,1\}$ and $\mathcal{B}_i$ does not have killing labels. Let a be the least element from $\mathcal{A}_{\mathit{flag}(i,s^*)} \setminus \mathit{dom}(g_i[s^*])$ and b be the least element from $\mathcal{B}_i \setminus \mathit{ran}(g_i[s^*])$. Then there exists a correct partial isomorphism h from $\mathcal{A}_{\mathit{flag}(i,s^*)}[s_0-1]$ into $\mathcal{B}_i[s_0]$ such that $g_i[s^*] \subseteq h$, $a \in \mathit{dom}(h)$, and $b \in \mathit{ran}(h)$.

(b) If i is the least number not in $\mathit{scope}(s^*)$, and $\mathcal{B}_i$ does not contain killing labels, then there is a correct partial isomorphism from some $\mathcal{A}_j[s_0-1]$, $j \in \{0,1\}$, into $\mathcal{B}_i[s_0]$.

First, let i is the least number not in $\mathit{scope}(s^*)$, and assume that $\mathcal{B}_i$ does not have killing labels. Then $\mathcal{B}_i$ is either isomorphic to $\mathcal{A}_0$ or $\mathcal{A}_1$. Therefore, there exists a stage s_1 such that for any $s \geq s_1$, there is a correct partial isomorphism from one of the structures $\mathcal{A}_0[s_1-1]$ or $\mathcal{A}_1[s_1-1]$ into $\mathcal{B}_i[s_1]$. So (b) is satisfied for any $s \geq s_1$.

Now suppose that $i \in \mathit{scope}(s^*)$ and $\mathit{flag}(i,s^*) \in \{0,1\}$, and assume that $\mathcal{B}_i$ does not have killing labels. Recall that at the expansionary stage s^* we witnessed a correct partial isomorphism from $\mathcal{A}_{\mathit{flag}(i,s^*)}[s^*-1]$ into $\mathcal{B}_i[s^*]$. Since any element a from $\mathcal{A}_0[s^*-1] \cup \mathcal{A}_1[s^*-1]$ has the unique label $\boxed{\mathit{sec}(a, s^*-1)}$ that is never put into the bag, this ensures that $\mathcal{B}_i$ is isomorphic to $\mathcal{A}_{\mathit{flag}(i,s^*)}$ and any isomorphism $f\colon \mathcal{A}_{\mathit{flag}(i,s^*)} \cong \mathcal{B}_i$ extends $g_i[s^*]$. Thus, one can find a large enough s_0 that satisfies the condition (a) above. □

Claim 4.5. *Suppose that for any $\mathcal{B}_j$, there is some $\mathcal{A}_m$ isomorphic to $\mathcal{B}_j$. Then for any $i \in \omega$, there is a stage s^* such that for all $s \geq s^*$, $i \in \mathit{scope}(s)$ and $\mathit{flag}(i,s) = \mathit{flag}(i,s^*)$. Furthermore, if $\mathit{flag}(i,s^*) = \mathrm{k}$, then $\mathcal{B}_i$ has a killing label, and if $\mathit{flag}(i,s^*) \in \{0,1\}$, then $\mathcal{B}_i$ is isomorphic to $\mathcal{A}_{\mathit{flag}(i,s^*)}$.*

Proof. Since there are infinitely many expansionary stages, for any $i \in \omega$, there is a stage s_0 such that $i \in \mathit{scope}(s)$ for all $s \geq s_0$.

First, assume that $\mathcal{B}_i$ contains a killing label $\boxed{\mathtt{k}}$. Choose a stage s such that $i \in \mathit{scope}(s)$ and $\mathcal{B}_i[s]$ contains the label $\boxed{\mathtt{k}}$. It is easy to see that for any $s' \geq s$, we have $\mathit{flag}(i,s') = \mathrm{k}$.

Now suppose that $\mathcal{B}_i$ has no killing labels. Since there are infinitely many expansionary stages, for any $i \geq 2$, $\mathcal{A}_i$ contains a killing label. Thus, $\mathcal{B}_i$ is isomorphic either to $\mathcal{A}_0$ or $\mathcal{A}_1$. Furthermore, there is a least stage $s_1 \geq s_0$ such that for any $s \geq s_1$, we have $flag(i,s) = flag(i,s_1) =: p_i \in \{0,1\}$. The definition of an expansionary stage ensures that the map $g_i := \bigcup_{s \geq s_1} g_i[s]$ is an isomorphism from $R(\mathcal{A}_{p_i})$ onto $R(\mathcal{B}_i)$. Furthermore, any two elements from $\mathcal{A}_{p_i} \cup \mathcal{B}_i$ possess the same labels. Thus, g extends to an isomorphism from $\mathcal{A}_{p_i}$ onto $\mathcal{B}_i$. □

The desired computable function $f(x)$ is defined as follows:

$$f(s) := \max\big\{0, \text{the least } i \in \mathit{scope}(s) \text{ with } \mathit{flag}(i,s) = 0\big\}.$$

Now suppose that $\mathcal{L} \not\cong \mathcal{H}$, and the sequences $\{\mathcal{A}_i\}_{i\in\omega}$ and $\{\mathcal{B}_j\}_{j\in\omega}$ are equal up to isomorphism. Recall that any $\mathcal{A}_i$, $i \geq 2$, obtains a killing label. Moreover, $R(\mathcal{A}_1) \cong \mathcal{H} \not\cong \mathcal{L} \cong R(\mathcal{A}_0)$. Thus, we have $\mathcal{A}_i \not\cong \mathcal{A}_0$ for any non-zero i.

Let k_0 be the least number such that $\mathcal{B}_{k_0} \cong \mathcal{A}_0$. By Claim 4.5, there is a stage s_0 such that for any stage $s \geq s_0$, we have $\{0,1,\dots,k_0\} \subseteq \mathit{scope}(s)$, $\mathit{flag}(k_0,s) = 0$, and $\mathit{flag}(i,s) \neq 0$ for all $i < k_0$. Therefore, $f(s) = k_0$ for any $s \geq s_0$. Lemma 4.1 is proved. □

In the rest of this section, we work with *unary algebras*, i.e. structures in the language $\{g^1\}$. A unary algebra $\mathcal{A}$ is an *injection structure* if the function $g^{\mathcal{A}}$ is 1-1.

We define auxiliary computable injection structures. For a non-zero $n \leq \omega$, the structure $\mathcal{X}_n$ contains infinitely many loops of size m, for every $m < 1+n$. The structure $\mathcal{Y}_n$ is a disjoint union of $\mathcal{X}_n$ and an ω-chain (i.e., a sequence $\{a_i\}_{i\in\omega}$ such that $g(a_i) = a_{i+1}$ for any i). We consider the class

$$K := \{\mathcal{X}_n, \mathcal{Y}_n : 1 \leq n \leq \omega\}.$$

Lemma 4.6. *Given a (total) computable sequence of unary algebras $\{\mathcal{D}_i\}_{i\in\omega}$ and a computable function $f(x)$, one can effectively construct a computable unary algebra $\mathcal{C}$ with the following properties:*

(1) $\mathcal{C}$ is isomorphic to some structure from K, and
(2) if the limit $k = \lim_s f(s)$ exists, then $\mathcal{C}$ is not isomorphic to $\mathcal{D}_k$.

Proof. We build $\mathcal{C}$ in stages. When we say that at a stage s, a component U is a chain of size $(n+1)$, this means the following: U consists of elements $a_0,\dots,a_n$ such that $g(a_i) = a_{i+1}$ for any $i < n$, and $g(a_n)[s]\uparrow$.

A *simple chain extension* of a chain U of size $(n+1)$ works as follows: choose a fresh b and set $g(a_n)[s+1] := b$, $g(b)[s+1]\uparrow$. A *simple loop extension* of a chain U just sets $g(a_n)[s+1] := a_0$.

We keep a special component $T[s]$ in our $\mathcal{C}$: $T[s]$ is either a chain of a finite size, or a loop. By $c[s]$ we denote the cardinality of $T[s]$.

At stage 0, we choose $\mathcal{C}[0] = T[0]$ as a chain of size 1.

Let s be a non-zero stage. First, for each non-zero $m \leq c[s-1]$, we add a fresh loop of size m into $\mathcal{C}$. Consider the following cases:

Case 1. Suppose that $T[s-1]$ is a chain. Then check whether there is an element $a \in \mathcal{D}_{f(s)}[s]$ such that $g^m(a) \neq g^n(a)$ for all $0 \leq m < n \leq c[s-1]$. (Note that the elements $g^m(a)$ might not be in $\mathcal{D}_{f(s)}[s]$, but that since $\mathcal{D}_{f(s)}$ is total, we can find these elements without waiting.) If such an a exists, then define $T[s]$ as a simple loop extension of $T[s-1]$. Otherwise, let $T[s]$ be a simple chain extension of $T[s-1]$.

Case 2. If $T[s-1]$ is a loop and $f(s) \neq f(s-1)$, then choose $T[s]$ as a fresh chain of size $c[s-1]+1$.

Case 3. If $T[s-1]$ is a loop and $f(s) = f(s-1)$, then proceed to the next stage.

This concludes the description of the construction. It is straightforward to check that the constructed $\mathcal{C}$ is computable (uniformly in computable indices of a function f and a sequence $\{\mathcal{D}_i\}_{i\in\omega}$), and $\mathcal{C}$ satisfies the following properties:

a) Suppose that $T[s]$ *stabilizes*, i.e. there is a stage s^* such that $T[s] = T[s^*]$ for all $s \geq s^*$. Then $T[s^*]$ is a loop of size $c[s^*]$, and $\mathcal{C}$ is isomorphic to $\mathcal{X}_{c[s^*]}$.
b) Assume that for any s, there is a stage $s_1 > s$ such that $T[s_1] \cap T[s] = \emptyset$. Then $\mathcal{C}$ is isomorphic to $\mathcal{X}_\omega$.

c) Suppose that $T[s]$ does not stabilize and there is a stage s_0 such that for any $s \geq s_0$, we have $T[s] \cap T[s_0] \neq \emptyset$. Then $\mathcal{C} \cong \mathcal{Y}_\omega$.

Hence $\mathcal{C}$ is isomorphic to a structure from the class K.

Now suppose that there exists a limit $k := \lim_s f(s)$. Without loss of generality, one can assume that $\mathcal{D}_k$ is either isomorphic to some $\mathcal{X}_m$, $1 \leq m \leq \omega$, or isomorphic to $\mathcal{Y}_\omega$.

Let s_0 be the least stage such that $f(s) = k$ for any $s \geq s_0$. Consider the following two cases:

1) Assume that $T[s]$ stabilizes. This implies that $\lim_s T[s]$ is a loop, and there is a stage $s_1 \geq s_0$ and an element $a \in \mathcal{D}_k[s_1]$ such that $g^m(a) \neq g^n(a)$ for all $0 \leq m < n \leq c[s_1 - 1] = c[s_1]$. Furthermore, $T[s] = T[s_1]$ for any $s \geq s_1$. Hence, $\mathcal{C} \cong \mathcal{X}_{c[s_1]}$ and $\mathcal{D}_k$ contains a chain of size $c[s_1] + 1$. Therefore, $\mathcal{C} \ncong \mathcal{D}_k$.
2) Assume that $T[s]$ does not stabilize. Notice that for any $s' \geq s \geq s_0$, we have $T[s] \subseteq T[s']$. Thus, $\mathcal{C}$ is isomorphic to $\mathcal{Y}_\omega$. If $\mathcal{D}_k$ is also isomorphic to $\mathcal{Y}_\omega$, then find the least stage $s_1 \geq s_0$ such that $\mathcal{D}_k[s_1]$ contains an element a from the ω-chain in $\mathcal{D}_k$. The construction ensures that $T[s_1]$ is a loop and $T[s] = T[s_1]$ for any $s \geq s_1$: this contradicts our assumption that $T[s]$ does not stabilize. Therefore, $\mathcal{C} \cong \mathcal{Y}_\omega \ncong \mathcal{D}_k$.

We have shown that $\mathcal{C}$ is not isomorphic to $\mathcal{D}_{\lim_s f(s)}$. □

5. A transformation from Δ^0_2-structures to automatic structures

In this section, we prove the following theorem. For notational convenience, identify structures with their indices.

Theorem 5.1. *There is a computable function Ψ that, given an index for a Δ^0_2-presentation M of a countably infinite graph, outputs an index (as an automatic structure) for an automatic structure $\Psi(M)$. The isomorphism type of $\Psi(M)$ depends only on the isomorphism type of M.*

Moreover, there is a $\mathbf{0}'$-computable function Ψ^{inv} that, given an index for a partial 1-decidable structure N in the language of $\Psi(M)$, produces an index for a partial Δ^0_2 computable graph $\Psi^{inv}(N)$. The isomorphism type of $\Psi^{inv}(N)$ depends only on the isomorphism type of N.

If N is a 1-decidable presentation of $\Psi(M)$, then the corresponding $\Psi^{inv}(N)$ is a Δ^0_2-presentation of M.

5.1. Proof idea.

Khoussainov and Minnes [KM09] produced a similar operator, but it was transforming computable structures into automatic structures, while the decoding was Δ^0_2. We will be based on their idea of using the configuration space of a Turing machine, but we will be a bit more careful with the coding itself. The limit lemma is the key here.

We start by replacing the edge relation on points x, y by a pair of "boxes". Each box will contain a structure that we will define shortly, and whose isomorphism type will depend on whether $E(x, y)$ holds or not. The isomorphism type will be Σ^0_2-recognisable, this is why we need a pair of boxes: one box codes the edge relation, the other one its complement. We describe how to code the edge relation; its negation is coded similarly.

Fix a Turing machine $\mathbb{T}$ that represents the Δ^0_2 edge relation in the following sense. Fix a partial computable function ψ such that

$$E(x, y) \text{ iff } \exists t \psi(x, y, t) \uparrow \text{ iff } \forall^\infty t \psi(x, y, t) \uparrow .$$

Here the quantifier $\forall^\infty t$ means "for cofinitely many t". Choose a machine $\mathbb{T}$ that realizes the function ψ. It is well-known that the configuration space of any Turing machine is automatic. As noted in [KM09], one can uniformly modify the Turing machine to a machine that records the history of computation; this way we obtain a machine whose configuration space contains only non-intersecting paths ("chains") of finite or infinite length and of type ω (i.e. the transitive closure of this path is isomorphic to the ordering of natural numbers). Details will be given in the formal proof.

Fix x and y, and consider the box coding the edge relation between x and y in a Σ^0_2 way. In the box, start with a special point c. For each i, we initiate a chain from the special point that starts with a node whose initial conditions represent the configuration of the tape on input x, y, i, for every i. We shall grow such a chain until the computation halts. Then there exists an infinite chain (in fact, infinitely many of those) attached to the special point iff $\exists t \psi(x, y, t) \uparrow$, i.e., the edge relation holds.

There may be configurations which, when we trace the computation backwards (using the fact that $\mathbb{T}$ is reversable), we find to have begun in an invalid initial configuration. We cannot, in an automatic way, recognize whether a particular configuration in the configuration space of $\mathbb{T}$ represents a valid computation which began with input x, y, i. So to make what we put in the box automatic, we must include all of these configurations as well. So, in fact, we put the whole configuration space of $\mathbb{T}$ into the box, and connect the special point c to each valid initial configuration with input x, y, i. This means that we have added, to what we described in the previous paragraph, some "free-floating" chains, some of which may be finite and others of order type ω.

Finally, we want the isomorphism type of what is in the box to be dependent only on whether or not $E(x, y)$, and in particular, to be independent of whether, for example, there is a computation which halts after exactly three steps. So we will also attach to c infinitely many chains of each finite size, and there also will be infinitely many chains of each finite size and of type ω which are not attached to anything.

Let $\Psi(M)$ be the structure obtained from the given edge relation E and its complement $\overline{E}$. It is clear that the isomorphism type of $\Psi(M)$ does not depend from the choice of corresponding partial functions ψ for E and $\overline{E}$. Also the isomorphism type of $\Psi(M)$ does not depend on the given presentation of M though the choice of the edge relation E.

Now, given a 1-decidable presentation of the output structure $\Psi(M)$, observe that extendability of a node to a longer chain becomes a computable fact. It is clear how to use this observation to say that there exists an infinite chain in a Σ^0_2 fashion, since

$$\text{a chain is infinite} \iff \text{each of its nodes can be extended}$$

which is a Π^0_1 sentence. Since the negation of the edge relation is also uniformly Σ^0_2, we can produce a Δ^0_2-copy of M.

Below we give formal details.

5.2. **Formal proof.**

We fix a special *blank symbol* $\perp$. If we use some alphabet Σ, then we assume that $\perp \notin \Sigma$ (if not specified otherwise). Let $\Sigma_\perp := \Sigma \cup \{\perp\}$. Suppose that $w_1, \dots, w_n \in \Sigma^*$. One can identify a tuple $\bar{w} = w_1, \dots, w_n$ with its *convolution*, i.e. the string $conv(w_1, \dots, w_n)$ of the length $\max_i |w_i|$ over the alphabet $\Sigma^n_\perp$ such that the k-th symbol of $conv(\bar{w})$ is $(\sigma_1, \dots, \sigma_n)$, where

$$\sigma_i = \begin{cases} w_i(k), & \text{if } k < |w_i|, \\ \perp, & \text{otherwise.} \end{cases}$$

5.2.1. *Configuration spaces.*

Suppose that $\mathbf{T}$ is an n-tape Turing machine. The *configuration space* of $\mathbf{T}$ (denoted by $Conf(\mathbf{T})$) is a directed graph whose nodes represent configurations of $\mathbf{T}$. A node of $Conf(\mathbf{T})$ is (the convolution of) an n-tuple $\bar{w}$ such that for $i < n$, the ith coordinate of $\bar{w}$ encodes the contents of the ith tape as follows. If the head of the machine looks at a symbol a on the ith tape, u and v are the strings on the tape before and after this particular occurrence of a, $\mathbf{T}$ is in a state q, then the contents of the tape are encoded as $uqav$. An edge of the graph is a pair (c_1, c_2) such that $\mathbf{T}$ proceeds from the configuration c_1 to c_2 via a one-step computation. Note that $Conf(\mathbf{T})$ is an automatic graph.

A deterministic Turing machine $\mathbf{T}$ is *reversible* if any of weakly connected components of the graph $Conf(\mathbf{T})$ is either a finite chain or an ω-chain. Bennett [Ben73] (see also [KM09]) showed that for any one-tape deterministic Turing machine, one can effectively produce a reversible three-tape Turing machine that accepts the same language.

Let $bin(m)$ denote the binary presentation of a natural number m. For $x_1, \dots, x_k \in \omega$, $cbin(x_1, \dots, x_k)$ denotes the string $conv(bin(x_1), \dots, bin(x_k))$ in the alphabet $\{0, 1, \perp\}^k$. Let λ be the empty string.

We introduce three special conventions:

- If a four-tape Turing machine $\mathbf{T}$ computes a partial computable function $\psi(x_1, \dots, x_k)$, then we assume that it works as follows:
  - if $\psi(x_1, \dots, x_k)\downarrow = y$, then $\mathbf{T}$ starts from a *valid initial configuration*

$$(i^\frown cbin(x_1, \dots, x_k), cbin(x_1, \dots, x_k), \lambda, \lambda)$$

and the halting configuration of this computation is

$$(cbin(x_1,\dots,x_k), cbin(x_1,\dots,x_k), \lambda, q_f\widehat{\ }bin(y)),$$

where q_f is a halting state of $\mathbf{T}$;
  - the machine $\mathbf{T}$ never changes anything on its second tape;
  - if $\psi(x_1,\dots,x_k)\ \uparrow$, then the $\mathbf{T}$-computation starting from the configuration $(i\widehat{\ }cbin(x_1,\dots,x_k), cbin(x_1,\dots,x_k), \lambda, \lambda)$ never halts.

- In place of a reversible three-tape Turing machine $\mathbf{T}$, we will often use a reversible four-tape machine $\tilde{\mathbf{T}}$ which works as follows: Starting from a valid initial configuration $(i\widehat{\ }w, w, \lambda, \lambda)$, $\tilde{\mathbf{T}}$ never changes anything on its second tape and uses all the other tapes to perform the computation exactly as the machine $\mathbf{T}$ would have done.
- If a reversible four-tape machine $\mathbf{T}$ computes a partial function $\psi(x_1,\dots,x_k)$ and a configuration of $\mathbf{T}$ does not have the form $(u_1, cbin(y_1,\dots,y_k), u_3, u_4)$, then we omit this configuration from the graph $Conf(\mathbf{T})$. Notice that this does not injure the automatic presentability of $Conf(\mathbf{T})$.

The intuition of the conventions is as follows: We want a four-tape reversible Turing machine $\mathbf{T}$ to not forget its input information (i.e. numbers $x_1,\dots,x_k$ if $\mathbf{T}$ computes $\psi(x_1,\dots,x_k)$), and the second tape is reserved to forever memorize the data.

Recall that each component of the configuration space is a chain of finite type or of type ω. There are three different types of chains:

(1) *Terminating computation chains*: These are finite chains which begin with a valid initial configuration and which represent a computation that terminates after finitely many steps.
(2) *Non-terminating computation chains*: These are chains of type ω which begin with a valid initial configuration but which represent a computation which never terminates.
(3) *Unproductive chains*: these are chains which do not begin with a valid initial configuration.

Note that the set of valid initial configurations of $\mathbf{T}$ is automatic in the graph $Conf(\mathbf{T})$. While one might wish to exclude unproductive chains from the configuration space, it is unfortunately not automatic to decide whether a configuration lies on a computation chain or on an unproductive chain.

5.2.2. *The formal definition of* $\Psi(M)$. Suppose that M is a countable Δ^0_2-computable graph. As usual, we assume that the domain of M is equal to ω. Let ψ and φ be partial computable functions with the following properties: For any $x, y \in \omega$,

$$M \models E(x,y) \ \Leftrightarrow\ \exists t \psi(x,y,t)\uparrow \Leftrightarrow\ \forall^\infty t \psi(x,y,t)\uparrow;$$
$$M \models \neg E(x,y) \ \Leftrightarrow\ \exists t \varphi(x,y,t)\uparrow \Leftrightarrow\ \forall^\infty t \varphi(x,y,t)\uparrow .$$

Notice that given a Δ^0_2-index of M, one can uniformly compute computable indices of ψ and φ.

The language of our structure $\Psi(M)$ consists of the following symbols:

(a) the unary relation *Supp* that is used to encode the domain of M,
(b) binary relations R and Q which are used to encode the edge relation E^M,
(c) binary relations R_{non} and Q_{non} that are used to encode the non-edge $\neg E^M$.

Since our definitions for the edge case and the non-edge case are similar, here we give a detailed description only for encoding the edge relation.

Choose an automatic graph $\mathcal{A}$ such that it consists of infinitely many chains of each finite length and infinitely many ω-chains. Such a graph $\mathcal{A}$ can be constructed as follows:

- the domain of $\mathcal{A}$ is the set $0^*01^* \cup 22^*33^*$;
- $E^{\mathcal{A}}(x,y)$ holds iff x and y satisfy one of the following conditions:
  a) $x, y \in 0^*01^*$, $|x| = |y|$, and y is the least lexicographic successor of x;
  b) $x, y \in 22^*33^*$, $x = 2^k 3^m$, and $y = 2^k 3^{m+1}$ for some k, m.

Also fix an automatic graph $\mathcal{A}_{fin}$ such that $\mathcal{A}_{fin}$ consists of infinitely many chains of each finite length.

We produce a reversible four-tape deterministic Turing machine $\mathbf{T}$ which computes the function ψ and satisfies the conventions formulated above. We build an automatic copy of the directed graph

$Conf(\mathbf{T})$. Without loss of generality, one can assume that the alphabets of the automata for $Conf(\mathbf{T})$, $\mathcal{A}$, and $\mathcal{A}_{fin}$ are pairwise disjoint. Let $\underline{a}$, $\underline{b}$, and $\underline{c}$ be new symbols.

For simplicity, assume that we do not want to encode the non-edge relation $\neg E^M$ into our structure $\Psi(M)$. A discussion about how to correct this assumption will be given later. Then the structure $\Psi(M)$ is defined as follows.

- The domain of $\Psi(M)$ consists of five parts:
  (1) the set $A_0 := \{\underline{a}\widehat{\ }bin(x) : x \in \omega\}$,
  (2) $A_1 := \{\underline{b}\widehat{\ }conv(cbin(x,y), w) : x, y, t \in \omega,\ w = (u_1, cbin(x,y,t), u_3, u_4) \in Conf(\mathbf{T})\}$,
  (3) $A_2 := \{\underline{bb}\widehat{\ }conv(cbin(x,y), u) : x, y \in \omega,\ u \in |\mathcal{A}|\}$,
  (4) $A_3 := \{\underline{c}\widehat{\ }cbin(x,y) : x, y \in \omega\}$,
  (5) $A_4 := \{\underline{cc}\widehat{\ }conv(cbin(x,y), u) : x, y \in \omega,\ u \in |\mathcal{A}_{fin}|\}$.

  The first part is used to encode the domain of M, and the rest is the contents of the boxes (see the proof idea). We introduce the following notation: If $1 \leq i \leq 4$ and $x, y \in \omega$, then $A_i[x,y]$ is a subset of A_i which contains elements of the form $u\widehat{\ }conv(cbin(x,y), v)$, where $|u| \leq 2$.
- $Supp^{\Psi(M)} := A_0$.
- $R^{\Psi(M)}(u,v)$ holds iff there are $x, y \in \omega$ and $i \in \{1,2,3,4\}$ such that u and v satisfy one of the following conditions:
  – $u = \underline{a}\widehat{\ }bin(x)$ and $v \in A_i[x,y]$;
  – $u \in A_i[x,y]$ and $v = \underline{a}\widehat{\ }bin(y)$.

  In other words, $\bigcup_{1\leq i\leq 4} A_i[x,y]$ is the box associated with an ordered pair $(x,y) \in \omega^2$. This association is realized via the predicate $R^{\Psi(M)}$.
- $Q^{\Psi(M)}(u,v)$ holds iff there are $x, y \in \omega$ such that u and v satisfy one of the following:
  – $u = \underline{b}\widehat{\ }conv(cbin(x,y), u_1)$, $v = \underline{b}\widehat{\ }conv(cbin(x,y), v_1)$, and there is an edge from u_1 to v_1 in $Conf(\mathbf{T})$;
  – $u = \underline{bb}\widehat{\ }conv(cbin(x,y), u_1)$, $v = \underline{bb}\widehat{\ }conv(cbin(x,y), v_1)$, and there is an edge from u_1 to v_1 in $\mathcal{A}$;
  – $u = \underline{cc}\widehat{\ }conv(cbin(x,y), u_1)$, $v = \underline{cc}\widehat{\ }conv(cbin(x,y), v_1)$, and there is an edge from u_1 to v_1 in $\mathcal{A}_{fin}$;
  – $u = \underline{c}\widehat{\ }cbin(x,y)$ and $v = \underline{cc}\widehat{\ }conv(cbin(x,y), v_1)$, where v_1 is a node from $\mathcal{A}_{fin}$ such that its in-degree is zero;
  – $u = \underline{c}\widehat{\ }cbin(x,y)$ and $v = \underline{b}\widehat{\ }conv(cbin(x,y), (i\widehat{\ }cbin(x,y,t), cbin(x,y,t), \lambda, \lambda))$ for some $t \in \omega$.

  The predicate $Q^{\Psi(M)}$ establishes the inner structure of our boxes.

It is straightforward to prove that the constructed structure $\Psi(M)$ is automatic.

Now we reiterate our intuitive idea of the construction. The domain of the input graph M is encoded by the nodes $\underline{a}\widehat{\ }bin(x)$, where x is an element from M. These nodes constitute the "external structure" of $\Psi(M)$. The "internal structure" of $\Psi(M)$ contains all the other nodes from $\Psi(M)$, and this structure is intended to encode the edges from M. Consider a pair of points x, y from the input M. We want to encode the edge $E(x,y)$ like this: $x \to \boxed{\text{the box } B[x,y]} \to y$. More formally, we do the following. In $\Psi(M)$, the points x and y correspond to the nodes $\underline{a}\widehat{\ }bin(x)$ and $\underline{a}\widehat{\ }bin(y)$, respectively. These nodes are connected to the box $B[x,y] := \bigcup_{1\leq i\leq 4} A_i[x,y]$ via the relation R. This means that $\underline{a}\widehat{\ }bin(x)$ is R-connected to any element from $B[x,y]$, and any node from the box is R-connected to $\underline{a}\widehat{\ }bin(y)$. The inner structure (i.e. the structure in the language $\{Q\}$) of the box $B[x,y]$ essentially consists of two parts:

(1) The first part contains a special point $c[x,y] := \underline{c}\widehat{\ }cbin(x,y)$ and infinitely many chains of each finite length emanating from $c[x,y]$ (this is guaranteed by adding a copy of $\mathcal{A}_{fin}$ with the help of the set $A_4[x,y]$). Furthermore, the following conditions are equivalent:
   - $M \models E(x,y)$,
   - there is an ω-chain emanating from $c[x,y]$;
   - there is a natural number t such that $\psi(x,y,t)\uparrow$;
   - $(\forall^\infty t)\psi(x,y,t)\uparrow$;
   - there are infinitely many ω-chains emanating from $c[x,y]$.

This equivalence is guaranteed by connecting $c[x,y]$ with the nodes $\underline{b}\widehat{\ }conv(cbin(x,y),(i\widehat{\ }cbin(x,y,t),cbin(x,y,t),\lambda,\lambda))$, $t\in\omega$, which correspond to the valid initial configuration for the $\mathbf{T}$-computation of $\psi(x,y,t)$. The $\mathbf{T}$-computation itself is coded via adding a copy of (a part of) $Conf(\mathbf{T})$ with the help of $A_1[x,y]$.

(2) The second part consists of infinitely many chains of each finite length and infinitely many ω-chains. This is ensured by adding a copy of $\mathcal{A}$ with the help of $A_2[x,y]$ and by the fact that the machine $\mathbf{T}$ is reversible.

The non-edge relation $\neg E^M$ can be coded in a similar way: For any pair of points $x,y\in M$, we introduce their own box $B_{non}(x,y)$. This box is connected to $\underline{a}\widehat{\ }bin(x)$ and $\underline{a}\widehat{\ }bin(y)$ by the R_{non}-relation. The inner structure of $B_{non}(x,y)$ is the structure in the language $\{Q_{non}\}$ and it uses the same encoding as $B(x,y)$ to encode the non-edges.

We omit the technical details about coding non-edges, and from now on we assume that our $\Psi(M)$ is a structure in the language $\{Supp,R,Q,R_{non},Q_{non}\}$. The structure $\Psi(M)$ contains the information about both edge and non-edge relations.

5.2.3. *Verification.* Recall that we already showed that the resulting structure $\Psi(M)$ is automatic. In our proofs, we will use the following claim.

Claim 5.2. *Suppose that M and N are infinite graphs. Then $M\cong N$ iff $\Psi(M)\cong\Psi(N)$.*

Proof. Suppose that f is an isomorphism from M onto N. Then f induces a 1-1 map $\widehat{f}$ from $Supp^{\Psi(M)}$ onto $Supp^{\Psi(N)}$. Our construction ensures that for any nodes $x\neq y$ from M, the box $B[x,y]$ inside $\Psi(M)$ satisfies the following:

- If there is an edge from x to y in M, then $B[x,y]$ contains precisely the following components: the special node $c[x,y]$ with infinitely many ω-chains and infinitely many chains of each finite length emanating from $c[x,y]$, infinitely many "free-roaming" ω-chains, and infinitely many "free-roaming" chains of each finite length.
- If there is no edge from x to y in M, then $B[x,y]$ contains the same free-roaming chains, and $c[x,y]$ emanates infinitely many chains of each finite length and nothing else.

This observation implies that the map $\widehat{f}$ can be extended to an isomorphism from $\Psi(M)$ onto $\Psi(N)$. Moreover, a similar argument shows that given any isomorphism $g\colon\Psi(M)\to\Psi(N)$, the map $g\upharpoonright Supp^{\Psi(M)}$ induces an isomorphism from M onto N. □

Lemma 5.3. *There is a uniform $\mathbf{0}'$-procedure such that:*

- *given a partial 1-decidable structure N in the language of $\Psi(M)$, the procedure outputs a partial Δ^0_2 computable graph X;*
- *if N is a 1-decidable presentation of $\Psi(M)$, then the corresponding X is a Δ^0_2-presentation of M.*

Proof. We define an auxiliary Σ^c_2 formula

$$FinChain(z):=\bigvee\!\!\!\bigvee_{n\in\omega}\exists u_0\ldots\exists u_n\left[(u_0=z)\bigwedge_{i<n}Q(u_i,u_{i+1})\&\neg\exists wQ(u_n,w)\right].$$

The formula says that a node z is an element inside a finite Q-chain. Furthermore, in a 1-decidable structure N, the set $FinChain^N$ is computably enumerable, uniformly in N.

Let the domain of X be equal to $Supp^N$. Then the edge relation $E(x,y)$ in X is defined by a Σ^c_3 formula

$$\exists c\big[R(x,c)\&R(c,y)\&\exists z_0\exists z_1((z_0\neq z_1)\&Q(c,z_0)\&Q(c,z_1))\&\exists z(Q(c,z)\&\neg FinChain(z))\big]. \tag{1}$$

If N is 1-decidable, then the relation defined by this formula (in N) is $\Sigma^0_1(\emptyset')$, uniformly in N. The non-edge relation $\neg E(x,y)$ in X can be given by a Σ^c_3 formula similar to (1), where we replace predicates R and Q with R_{non} and Q_{non}, respectively.

It is not difficult to show that our construction guarantees the following: if N is isomorphic to $\Psi(M)$, then the structure X is isomorphic to M. If N is a 1-decidable structure, then X is Δ^0_2-computable.

Furthermore, since our definition of X uses only computable infinitary formulas, given a computable index of the structure

$$(N; \{x : N \models \exists w Q(x,w)\}),$$

one can compute a Δ^0_2 index of X. $\square$

This concludes the proof of Theorem 5.1.

5.2.4. *Corollaries of Theorem 5.1.* Here we give two useful consequences of the theorem.

First, we give the definition of a *uniformly automatic sequence* of structures, which will be heavily used in the proof of Theorem 1.1. We note that a similar notion of a *uniformly automatic class* was studied by Abu Zaid, Grädel, and Reinhardt [AGR17]. For a finite automaton $\mathbf{A}$, let $L(\mathbf{A})$ denote the language accepted by $\mathbf{A}$.

Definition 5.4. *Let $L = \{P_0^{n_0}, \dots, P_m^{n_m}\}$ be a finite relational language. We say that a sequence of L-structures $\{\mathcal{M}_k\}_{k\in\omega}$ is* uniformly automatic *if it satisfies the following. There is an alphabet Σ and a sequence $\boldsymbol{A}, \boldsymbol{B}_0, \dots, \boldsymbol{B}_m$ of automata such that:*

(1) The alphabet of $\boldsymbol{A}$ is $\{0,1,\bot\} \times \Sigma_\bot$.
(2) For $j \leq m$, the alphabet of $\boldsymbol{B}_j$ is $\{0,1,\bot\} \times \Sigma_\bot^{n_j}$.
(3) For any $k \in \omega$, $\mathcal{M}_k$ has the following properties:
- *the domain of $\mathcal{M}_k$ contains precisely the elements $u \in \Sigma^*$ such that $conv(bin(k), u) \in L(\mathbf{A})$;*
- *for any $j \leq m$, the relation $P_j^{\mathcal{M}_k}$ is equal to the set of all tuples $(w_1, \dots, w_{n_j})$ such that $w_1, \dots, w_{n_j} \in \Sigma^*$ and $conv(bin(k), w_1, \dots, w_{n_j}) \in L(\mathbf{B}_j)$.*

The intuition behind the definition is as follows. If the sequence of graphs $\{\mathcal{N}_k\}_{k\in\omega}$ is uniformly automatic, then one can use only two automata to represent the sequence $\{\mathcal{N}_k\}_{k\in\omega}$. The "domain automaton" $\mathbf{A}$ has two tapes: on the first one we write an index k of a structure, and on the second one we write a string $u \in \Sigma^*$. If the automaton $\mathbf{A}$ works on this input, then it will tell us whether u belongs to the domain of $\mathcal{N}_k$. The "edge automaton" $\mathbf{B}$ has three tapes: Again, on the first one we write $k \in \omega$, the second and the third one are used for inputting strings $u, v \in \Sigma^*$, respectively. If $\mathbf{B}$ works with this input, then it will inform us whether there is an edge from u to v in $\mathcal{N}_k$.

Corollary 5.5. *Suppose that $\{M_k\}_{k\in\omega}$ is a uniformly Δ^0_2-computable sequence of graphs. Then one can effectively uniformly construct a uniformly automatic sequence of structures $\{C_k\}_{k\in\omega}$ such that for any k, $C_k \cong \Psi(M_k)$. Here uniform effectiveness means the following: Given a Δ^0_2-computable index of a sequence $\{M_k\}_{k\in\omega}$, one can effectively produce automata $\mathbf{A}$, $\mathbf{B}_0$, $\dots$, $\mathbf{B}_4$ that witness the uniform automaticity of the sequence $\{C_k\}_{k\in\omega}$.*

Proof Sketch. This essentially repeats the proof of the theorem modulo some minor modifications. First, we choose partial computable functions ψ and φ such that for any $k, x, y \in \omega$,

$$M_k \models E(x,y) \;\Leftrightarrow\; \exists t \psi(k,x,y,t)\uparrow \Leftrightarrow\; \forall^\infty t \psi(k,x,y,t)\uparrow;$$
$$M_k \models \neg E(x,y) \;\Leftrightarrow\; \exists t \varphi(k,x,y,t)\uparrow \Leftrightarrow\; \forall^\infty t \varphi(k,x,y,t)\uparrow.$$

For $k \in \omega$, we define C_k as follows. We proceed with the construction of $\Psi(M_k)$, but:

- we use $\psi(k,x,y,t)$ in place of the "old" $\psi(x,y,t)$;
- the set A_0 contains the strings of the form $conv(bin(k), \underline{a}^\frown cbin(k,x))$, $x \in \omega$;
- the set A_1 consists of the strings $conv(bin(k), \underline{b}^\frown conv(cbin(k,x,y), w))$, where $x, y \in \omega$ and $w = (u_1, cbin(k,x,y,t), u_3, u_4) \in Conf(\mathbf{T})$ for some $t \in \omega$;
- the definitions of A_2, A_3, and A_4 are corrected in a similar way.

Since we use the configuration space $Conf(\mathbf{T})$ for *only one* Turing machine $\mathbf{T}$ (this $\mathbf{T}$ computes $\psi(k,x,y,t)$), it is not difficult to build automata witnessing the uniform automaticity of the constructed sequence $\{C_k\}_{k\in\omega}$. Notice that C_k is computably isomorphic to $\Psi(M_k)$. $\square$

From now on, we always assume that we use the corollary above when we apply Ψ to a Δ^0_2-computable sequence of graphs.

Corollary 5.6. *The index set of decidable structures that are Δ^0_2-categorical w.r.t. decidable presentations is Π^1_1 complete.*

In particular, for any computable non-zero ordinal α, there exists a decidable structure $\mathcal{D}_\alpha$ which is Δ^0_2-categorical w.r.t. decidable presentations, but not relatively Δ^0_α-categorical w.r.t. decidable presentations. Hence, the criterion of Nurtazin [Nur74] does not have a reasonable generalisation to the case of Δ^0_2-isomorphisms.

Proof Sketch. Fix a Π^1_1 complete set X. A relativisation of the result of Downey, Kach, Lempp, Lewis-Pie, Montalbán, and Turetsky [DKL+15] gives a uniformly Δ^0_2-computable sequence of graphs $\{M_k\}_{k\in\omega}$ such that for any k, the structure M_k is Δ^0_2-categorical w.r.t. Δ^0_2-computable presentations if and only if $k \in X$. We consider the sequence $\{\Psi(M_k)\}_{k\in\omega}$.

(1) Suppose that the structure $\Psi(M_k)$ is Δ^0_2-categorical w.r.t. decidable presentations. Let N_0 and N_1 be Δ^0_2-computable copies of M_k. Then $\Psi(N_0)$ and $\Psi(N_1)$ are automatic copies of $\Psi(M_k)$ and hence, there is a Δ^0_2-computable isomorphism f from $\Psi(N_0)$ onto $\Psi(N_1)$. The map $f \upharpoonright Supp^{\Psi(N_0)}$ induces a Δ^0_2-isomorphism from N_0 onto N_1. Therefore, M_k is Δ^0_2-categorical w.r.t. Δ^0_2-computable presentations.

(2) Suppose that the structure M_k is Δ^0_2-categorical w.r.t. Δ^0_2-computable presentations. Let C_0 and C_1 be decidable copies of $\Psi(M_k)$. We apply Lemma 5.3 and produce Δ^0_2-computable structures D_0 and D_1 using C_0 and C_1, respectively, as an input. Since $D_0 \cong D_1 \cong M_k$, there is a Δ^0_2-isomorphism g from D_0 onto D_1.

One can use the decidability of C_0 and C_1 to extend this g to a Δ^0_2-isomorphism acting from C_0 onto C_1. We give a sketch of the construction. Suppose that we want to map a box $B(x,y)$ from C_0 onto the box $B(g(x),g(y))$ from C_1. Given an element z from $B(x,y)$, we use the oracle $\mathbf{0}'$ to learn the answers to the following questions:

(1) Is z connected to the special node $c[x,y]$?
(2) What is the length of the unique Q^{C_0}-chain going through z?

If we know these answers for all $z \in B(x,y)$, then we can recover the desired isomorphism between two boxes. Thus, N_0 and N_1 are Δ^0_2-isomorphic, and $\Psi(M_k)$ is Δ^0_2-categorical w.r.t. decidable presentations.

Therefore, for any $k \in \omega$, the following conditions are equivalent:

(1) $k \in X$,
(2) M_k is Δ^0_2-categorical w.r.t. Δ^0_2-computable presentations,
(3) $\Psi(M_k)$ is Δ^0_2-categorical w.r.t. decidable presentations.

So the index set under consideration is Π^1_1 complete. □

6. A transformation that uses equivalence structures

Theorem 6.1. *There is a computable function Φ that, given an index for Δ^0_2 presentation M of a unary algebra, outputs an index for a computable structure $\Phi(M)$. The isomorphism type of $\Phi(M)$ depends only on that of M. If M belongs to the class K defined before Lemma 4.6, then the structure $\Phi(M)$ has an automatic copy.*

Moreover, there is a $\mathbf{0}'$-computable function Φ^{inv} that, given an index for a partial 1-decidable structure N, produces an index for a total Δ^0_2-computable unary algebra $\Phi^{inv}(N)$. The isomorphism type of $\Phi^{inv}(N)$ depends only on that of N.

If N is a copy of some $\Phi(M)$, then $\Phi^{inv}(N)$ is isomorphic to M.

6.1. **Proof idea.** Suppose that M is a unary algebra in the language $\{f^1\}$, and x, y are elements from M. Then $\Phi(M)$ is built as follows. We delete the function f^M and add a box $B[x,y]$ such that its inner structure is an equivalence relation $E[x,y]$ with infinitely many classes:

- if $f^M(x) \neq y$, then every class of $E[x,y]$ has size 2;
- if $f^M(x) = y$, then $E[x,y]$ consists of infinitely many classes of size 1 and infinitely many classes of size 2.

Notice that $f^M(x) = y$ iff the box $B[x, y]$ contains an equivalence class of size 1. This condition is definable by a finitary Σ_2 formula.

6.2. **Formal proof.** Let $\mathcal{E}[2]$ denote a computable infinite equivalence structure such that each of its classes has size 2. By $\mathcal{E}[1, 2]$ we denote a computable equivalence structure such that it consists of infinitely many 1-classes and infinitely many 2-classes. It is easy to verify the following

Claim 6.2. *Given a Σ^0_2 set S, one can effectively uniformly produce a computable sequence of equivalence structures $\{\mathcal{E}_k\}_{k\in\omega}$ such that for any k,*

$$\mathcal{E}_k \cong \begin{cases} \mathcal{E}[1,2], & \text{if } x \in S, \\ \mathcal{E}[2], & \text{if } x \notin S. \end{cases}$$

A more formal definition of $\Phi(M)$ can be stated as follows. Suppose that M is a Δ^0_2-computable unary algebra. Then $\Phi(M)$ is a structure in a finite relational language $\{Supp^1, R^2, Q^2\}$ such that:

- $Supp^{\Phi(M)}$ is the domain of M. We may assume that $Supp^{\Phi(M)}$ is the set of all even numbers, and odd numbers are used to construct infinite boxes $B[x, y]$.
- For any $x, y \in Supp^{\Phi(M)}$, the relation $R^{\Phi(M)}$ connects x with every element from $B[x, y]$ and $R^{\Phi(M)}$ connects every element from $B[x, y]$ with y.
- For any $x, y \in Supp^{\Phi(M)}$, the box $B[x, y]$ contains a computable equivalence structure (in the language $\{Q\}$) isomorphic to

$$\begin{cases} \mathcal{E}[1,2], & \text{if } f^M(x) = y, \\ \mathcal{E}[2], & \text{if } f^M(x) \neq y. \end{cases}$$

The claim above implies that given a Δ^0_2-index of M, one can compute a computable index of $\Phi(M)$. Using an argument similar to that of Claim 5.2, it is not hard to prove the following.

Claim 6.3. *Suppose that A and B are infinite unary algebras. Then $A \cong B$ iff $\Phi(A) \cong \Phi(B)$.*

Claim 6.4. *Given a partial 1-decidable structure N, one can effectively in $\mathbf{0}'$ (uniformly) construct a total Δ^0_2-computable unary algebra X. Furthermore, if N is a copy of $\Phi(M)$ for some M, then X is isomorphic to M.*

Proof Sketch. The argument is similar to Lemma 5.3. The main difference is that we need to construct a *total* unary algebra X. We use a finitary Σ_2 formula

$$\xi(x, y) := \exists c[R(x, c) \& R(c, y) \& \forall w(w \neq c \to \neg Q(c, w))].$$

First, we use the oracle $\mathbf{0}'$ to determine whether the set $Supp^N$ is empty. If $Supp^N = \emptyset$, then let X be a computable copy of $\mathcal{X}_1$ from the class K.

If $Supp^N \neq \emptyset$, then the domain of our X is $Supp^N$. If we want to define the value $f^X(x)$ for an element $x \in Supp^N$, then we use the oracle $\mathbf{0}'$ and ask whether $N \models \exists y \xi(x, y)$ (this can be done, since for a partial 1-decidable structure, its 1-diagram is represented by a c.e. set of $\exists$- and $\forall$-formulas). If such a y exists, then we choose some such y and set $f^X(x) := y$. If there is no such y, then define $f^X(x) := x$.

Given a computable index of a structure

$$(N; \{u : N \models \forall w(w \neq u \to \neg Q(u, w))\}),$$

one can compute a Δ^0_2-computable index of X. □

Claim 6.5. *If M is a unary algebra from the class K used in Lemma 4.6, then the structure $\Phi(M)$ has an automatic copy.*

This is not necessarily uniform.

Proof Sketch. We sketch a proof for the structure $\mathcal{Y}_2$ consisting of infinitely many 1-loops, infinitely many 2-loops, and one ω-chain.

Let $\underline{a}$, $\underline{b}$, $\underline{c}$, $\underline{d}$, $\underline{e}$ be new symbols. An automatic copy $\mathcal{A}$ of $\Phi(\mathcal{Y}_2)$ is defined as follows.

- The domain of $\mathcal{A}$ consists of three parts:

(1) $A_0 := \{v\hat{\ }bin(x) : v \in \{\underline{a}, \underline{b}, \underline{c}, \underline{d}\},\ x \in \omega\}$;
(2) $A_1 := \{uv\hat{\ }cbin(x,y,z), uv\underline{e}\hat{\ }cbin(x,y,z) : u, v \in \{\underline{a}, \underline{b}, \underline{c}, \underline{d}\},\ x, y, z \in \omega\}$;
(3) $A_2 := \{\underline{a}^3\hat{\ }cbin(x,z), \underline{b^2 c}\hat{\ }cbin(x,z), \underline{c^2 b}\hat{\ }cbin(x,z), \underline{d}^3\hat{\ }cbin(x, x+1, z) : x, z \in \omega\}$.

- $Supp^{\mathcal{A}} := A_0$.
- For $v, w \in \mathcal{A}$, $R^{\mathcal{A}}(v,w)$ holds iff there are $x, y, z \in \omega$ with one of the following properties:
  - $v = p\hat{\ }bin(x)$ and $w = pqr\hat{\ }cbin(x,y,z)$ for some $p, q \in \{\underline{a}, \underline{b}, \underline{c}, \underline{d}\}$ and $r \in \{\underline{e}, \lambda\}$;
  - $v = pqr\hat{\ }cbin(x,y,z)$ and $w = q\hat{\ }bin(y)$ for some $p, q \in \{\underline{a}, \underline{b}, \underline{c}, \underline{d}\}$ and $r \in \{\underline{e}, \lambda\}$;
  - $\{v, w\} = \{\underline{a}\hat{\ }bin(x), \underline{a}^3\hat{\ }cbin(x,z)\}$;
  - $v = \underline{b}\hat{\ }bin(x)$ and $w = \underline{b^2 c}\hat{\ }cbin(x,z)$;
  - $v = \underline{b^2 c}\hat{\ }cbin(x,z)$ and $w = \underline{c}\hat{\ }bin(x)$;
  - $v = \underline{c}\hat{\ }bin(x)$ and $w = \underline{c^2 b}\hat{\ }cbin(x,z)$;
  - $v = \underline{c^2 b}\hat{\ }cbin(x,z)$ and $w = \underline{b}\hat{\ }bin(x)$;
  - $v = \underline{d}\hat{\ }bin(x)$ and $w = \underline{d}^3\hat{\ }cbin(x, x+1, z)$;
  - $v = \underline{d}^3\hat{\ }cbin(x, x+1, z)$ and $w = \underline{d}\hat{\ }bin(x+1)$.
- $Q^{\mathcal{A}}(v,w)$ holds iff either $v = w$ and $v \in A_1 \cup A_2$, or the set $\{v, w\}$ is equal to $\{pq\hat{\ }cbin(x,y,z), pq\underline{e}\hat{\ }cbin(x,y,z)\}$ for some $p, q \in \{\underline{a}, \underline{b}, \underline{c}, \underline{d}\}$ and $x, y, z \in \omega$.

It is straightforward to verify that the structure $\mathcal{A}$ is automatic. The idea behind the construction of $\mathcal{A}$ is the following:

- 1-loops in $\mathcal{Y}_2$ correspond to elements $\underline{a}\hat{\ }bin(x)$, $x \in \omega$;
- a 2-loop in $\mathcal{Y}_2$ is encoded by a pair $(\underline{b}\hat{\ }bin(x), \underline{c}\hat{\ }bin(x))$;
- the ω-chain from $\mathcal{Y}_2$ is associated with the sequence $\{\underline{d}\hat{\ }bin(k)\}_{k\in\omega}$.

Hence, $\mathcal{A}$ is an automatic copy of $\Phi(\mathcal{Y}_2)$. An automatic presentation for an arbitrary algebra $\mathcal{Y} \in K$ can be constructed in a similar fashion. □

This concludes the proof of Theorem 6.1.

Corollary 6.6. *There exists a uniformly automatic sequence of structures $\{\mathcal{F}_k\}_{k\in\omega}$ such that $\mathcal{F}_0 \cong \Phi(\mathcal{X}_\omega)$, $\mathcal{F}_1 \cong \Phi(\mathcal{Y}_\omega)$, $\mathcal{F}_{2k} \cong \Phi(\mathcal{X}_k)$ and $\mathcal{F}_{2k+1} \cong \Phi(\mathcal{Y}_k)$ for any non-zero k.*

Proof Sketch. The proof of Claim 6.5 can be easily modified in such a way that in place of special new symbols $\underline{a}$, $\underline{b}$, $\underline{c}$, ..., one can use binary strings $bin(1)$, $bin(2)$, $bin(3)$, ... This modification allows us to construct a desired sequence $\{\mathcal{A}_k\}_{k\in\omega}$ and automata $\mathbf{A}$, $\mathbf{B}_0$, $\mathbf{B}_1$, $\mathbf{B}_2$ witnessing the uniform automaticity of the sequence. □

7. Proof of Theorem 1.1

7.1. **Formal proof.** Fix a Π^1_1 complete set O and a computable sequence of linear orders $\{\mathcal{L}_n\}_{n\in\omega}$ with the following properties ([Har68]):

- if $n \in O$, then $\mathcal{L}_n$ is well-ordered;
- if $n \notin O$, then $\mathcal{L}_n$ is isomorphic to Harrison linear order $\mathcal{H} = \omega_1^{CK} \cdot (1 + \eta)$.

We will build a uniformly computable sequence of structures $\{\mathcal{S}_n\}_{n\in\omega}$ such that:

- if $n \in O$, then $\mathcal{S}_n$ has no 1-decidable presentations;
- if $n \notin O$, then $\mathcal{S}_n$ has an automatic copy.

We fix a number n and describe how to construct (uniformly in n) the structure $\mathcal{S} := \mathcal{S}_n$. We also denote $\mathcal{L} := \mathcal{L}_n$.

First, we need to describe the language of our structure $\mathcal{S}$. We assume that the operator Ψ outputs structures in the language $L_\Box := \{Supp_\Box, R_\Box, Q_\Box, R^{non}_\Box, Q^{non}_\Box\}$. Here the subscript $\Box$ means that we want Ψ to be applied only to square boxes $\boxed{\mathcal{M}}$. The operator Φ is applied only to round boxes $\bigcirc\!\!\!\!\!\mathcal{N}$ and it outputs structures in the language $L_\circ := \{Supp_\circ, R_\circ, Q_\circ\}$.

A double box $\boxed{\Psi(\mathcal{M})}$-$\bigcirc\!\!\!\!\!\!\!\!\Phi(\mathcal{N})$ is organized as follows. We use two binary relations E_{box} and R_{box} such that:

- E_{box} is an equivalence relation, and every double box is an E_{box}-class;
- $R_{box}(x, y)$ connects any x from a square box $\boxed{\Psi(\mathcal{M})}$ to any y from the corresponding round box $\bigcirc\!\!\!\!\!\Phi(\mathcal{N})$.

So, our working language L_0 is equal to

$$\{Supp_\square, R_\square, Q_\square, R_\square^{non}, Q_\square^{non}\} \cup \{Supp_\circ, R_\circ, Q_\circ\} \cup \{E_{box}, R_{box}\}.$$

We fix a computable list of all partial 1-decidable L_0-structures $\{\mathcal{M}_e\}_{e\in\omega}$.

For a structure $\mathcal{M}_e$, we produce two lists of uniformly partial 1-decidable structures $\{\mathcal{U}_k\}_{k\in\omega}$ and $\{\mathcal{V}_k\}_{k\in\omega}$. Notice the following: Formally, it would be more correct to write $\mathcal{U}_k^e$ and $\mathcal{V}_k^e$, since these structures depend on an index e. Nevertheless, for convenience, we will omit the superscript e in this and similar situations.

The idea behind the lists is the following: We want to treat our $\mathcal{M}_e$ as a disjoint union of double boxes $\boxed{\mathcal{U}_k}$-$\bigcirc\!\!\!\!\!\mathcal{V}_k$, $k \in \omega$.

The construction of the structures proceeds as follows. For $m \in \omega$, we define:

- The domain of $\mathcal{U}_m$ is equal to the set $\{x : E_{box}(m, x) \& \exists y R_{box}(x, y)\}$, and $\mathcal{U}_m$ is an $L_\square$-substructure of $\mathcal{M}_e$.
- The domain of $\mathcal{V}_m$ is the set $\{x : E_{box}(m, x) \& \exists y R_{box}(y, x)\}$, and $\mathcal{V}_m$ is an $L_\circ$-substructure of $\mathcal{M}_e$.

Since $\mathcal{M}_e$ is partial 1-decidable and the definitions of $\mathcal{U}_m, \mathcal{V}_m$ are given by $\exists$-formulas, the constructed structures $\mathcal{U}_k$ and $\mathcal{V}_k$, $k \in \omega$, are uniformly partial 1-decidable. Note that there is some subtlety here: Two different numbers, say, 0 and 1 can produce the same double boxes, e.g. $\mathcal{U}_0 = \mathcal{U}_1$ and $\mathcal{V}_0 = \mathcal{V}_1$. Nevertheless, this will not be an obstacle in our construction. We will always ensure that either the constructed $\mathcal{S}$ has an automatic presentation, or $\mathcal{S}$ is not isomorphic to $\mathcal{M}_e$.

7.1.1. *Construction.* Our structure $\mathcal{S}$ is built as a disjoint union of a sequence of L_0-structures $\{\mathcal{R}_e\}_{e\in\omega}$. For $e \in \omega$, $\mathcal{R}_e$ is used to diagonalize against $\mathcal{M}_e$:

$$\text{If } \mathcal{L} \not\cong \mathcal{H}, \text{ then } \mathcal{R}_e \text{ witnesses that } \mathcal{S} \not\cong \mathcal{M}_e.$$

We describe the construction of $\mathcal{R}_e$. Given $\mathcal{M}_e$, we produce two sequences of uniformly partial 1-decidable structures $\{\mathcal{U}_i\}_{i\in\omega}$ and $\{\mathcal{V}_i\}_{i\in\omega}$. Then, by Theorem 5.1 and Theorem 6.1, we obtain two Δ_2^0-computable sequences of partial Δ_2^0-computable structures $\{\tilde{\mathcal{B}}_i\}_{i\in\omega}$ and $\{\tilde{\mathcal{D}}_i\}_{i\in\omega}$ with the following properties:

- $\tilde{\mathcal{B}}_i$ is a graph. If $\mathcal{U}_i \cong \Psi(M)$ for some M, then $\tilde{\mathcal{B}}_i \cong M$.
- $\tilde{\mathcal{D}}_i$ is a total unary algebra. If $\mathcal{V}_i \cong \Phi(N)$ for some N, then $\tilde{\mathcal{D}}_i \cong N$.

Now we want to apply Lemma 4.1. In order to do this, we introduce the following conventions: We simultaneously change the sequences $\{\tilde{\mathcal{B}}_i\}_{i\in\omega}$ and $\{\tilde{\mathcal{D}}_i\}_{i\in\omega}$ in order to produce new sequences $\{\mathcal{B}_j\}_{j\in\omega}$ and $\{\mathcal{D}_j\}_{j\in\omega}$ such that:

- Suppose that some of $\tilde{\mathcal{B}}_i$ contains a label $\boxed{\langle e, 0\rangle}$. Then the sequence $\{\mathcal{B}_j\}_{j\in\omega}$ contains precisely those $\tilde{\mathcal{B}}_i$ (up to isomorphism) which have a label $\boxed{\langle e, 0\rangle}$ inside. Furthermore, $\mathcal{B}_j$ was $\tilde{\mathcal{B}}_i$ in the original sequence, then $\mathcal{D}_j$ is isomorphic to the disjoint union of $\tilde{\mathcal{D}}_i$ and $\mathcal{X}_1$ (from the class K). Notice that if $\tilde{\mathcal{D}}_i$ also belongs to K, then $\mathcal{D}_j \cong \tilde{\mathcal{D}}_i$.
- Suppose that no $\tilde{\mathcal{B}}_i$ contains a label $\boxed{\langle e, 0\rangle}$. Then the sequence $\{\mathcal{B}_j\}_{j\in\omega}$ consists of empty structures, and any $\mathcal{D}_j$ is isomorphic to $\mathcal{X}_1$.

We assume that the construction of Lemma 4.1 puts the label $\boxed{\langle e, 0\rangle}$ on all elements of any $\mathcal{A}_i$. Our conventions ensure that different $\mathcal{R}_e$ do not interfere with each other.

By a relativisation of Lemma 4.1, we construct a Δ_2^0-computable function $f(x)$ and a Δ_2^0-computable sequence of (total) graphs $\{\mathcal{A}_i\}_{i\in\omega}$ satisfying the following:

a) If $\mathcal{L} \cong \mathcal{H}$, then there is a non-zero i with $\mathcal{A}_i \cong \mathcal{A}_0$.

b) Suppose that $\mathcal{L} \ncong \mathcal{H}$, and the sequences $\{\mathcal{A}_i\}_{i\in\omega}$ and $\{\mathcal{B}_j\}_{j\in\omega}$ are equal up to isomorphism. Then for any $i > 0$, $\mathcal{A}_i \ncong \mathcal{A}_0$. Moreover, there is a limit $k := \lim_s f(s)$ and $\mathcal{A}_0 \cong \mathcal{B}_k$.

Notice that the construction of Lemma 4.1 can be easily modified in such a way that any $\mathcal{A}_i$ is infinite.

After that we apply a relativisation of Lemma 4.6 and obtain a Δ^0_2-computable unary algebra $\mathcal{C}$ such that:

i) $\mathcal{C}$ is isomorphic to a structure from the class K;
ii) If $\mathcal{L} \ncong \mathcal{H}$ and the sequences $\{\mathcal{A}_i\}_{i\in\omega}$ and $\{\mathcal{B}_j\}_{j\in\omega}$ are equal up to isomorphism, then $\mathcal{C} \ncong \mathcal{D}_{\lim_s f(s)}$.

Fix a uniformly automatic sequence $\{\mathcal{F}_i\}_{i\in\omega}$ from Corollary 6.6. Informally speaking, the sequence lists all structures $\Phi(\mathcal{Y})$, $\mathcal{Y} \in K$, without repetitions.

Then the desired structure $\mathcal{R}_e$ is a disjoint union of a *special* double box $\boxed{\Psi(\mathcal{A}_0)}\text{-}\textcircled{$\Phi(\mathcal{C})$}$ and infinitely many copies of every double box $\boxed{\Psi(\mathcal{A}_{i+1})}\text{-}\textcircled{$\mathcal{F}_j$}$, where $i, j \in \omega$.

7.1.2. *Verification.* It is not difficult to show that the structure $\mathcal{S}$ is computable.

Lemma 7.1. *If $n \in O$ (i.e. the outcome is Π^1_1), then $\mathcal{S}$ has no 1-decidable copies.*

Proof. Recall that $\mathcal{L} \ncong \mathcal{H}$. Assume that $\mathcal{S}$ is isomorphic to some (total) 1-decidable $\mathcal{M}_e$. Consider the special double box $\boxed{\Psi(\mathcal{A}_0)}\text{-}\textcircled{$\Phi(\mathcal{C})$}$ from $\mathcal{R}_e$.

Our construction is organized in such a way that for any double box $\boxed{\Psi(\mathcal{A})}\text{-}\textcircled{$\mathcal{G}$}$ from $\mathcal{S}$, the following conditions are equivalent:

- this double box belongs to $\mathcal{R}_e$;
- there is an element from $\mathcal{A}$ labeled by $\boxed{\langle e, 0\rangle}$;
- any element from $\mathcal{A}$ is labeled by $\boxed{\langle e, 0\rangle}$.

Therefore, since $\mathcal{S} \cong \mathcal{M}_e$, the sequences $\{\mathcal{A}_i\}_{i\in\omega}$ and $\{\mathcal{B}_j\}_{j\in\omega}$ (from the construction of $\mathcal{R}_e$) are equal up to isomorphism. So by Lemma 4.1 we get that $\mathcal{A}_{i+1} \ncong \mathcal{A}_0$ for any i, and $\mathcal{A}_0 \cong \mathcal{B}_k$, where $k := \lim_s f(s)$.

Consider a 1-decidable double box $\boxed{\mathcal{U}}\text{-}\textcircled{$\mathcal{V}$}$ from $\mathcal{M}_e$ which corresponds to the Δ^0_2-computable box $\boxed{\mathcal{B}_k}\text{-}\textcircled{$\mathcal{D}_k$}$. Then $\boxed{\mathcal{U}}\text{-}\textcircled{$\mathcal{V}$}$ is the unique double box from $\mathcal{M}_e$ such that its square box is isomorphic to $\Psi(\mathcal{B}_k) \cong \Psi(\mathcal{A}_0)$. Again, since $\mathcal{S} \cong \mathcal{M}_e$, we deduce that the round boxes $\mathcal{V}$ and $\Phi(\mathcal{C})$ must be isomorphic. On the other hand, we know that $\mathcal{C} \ncong \mathcal{D}_k$. Thus, we have $\mathcal{V} \cong \Phi(\mathcal{D}_k) \ncong \Phi(\mathcal{C})$; contradiction. Therefore, the structure $\mathcal{S}$ has no 1-decidable copies. $\square$

Now we want to deal with the Σ^1_1 outcome. First, we prove two lemmas about automatic structures.

Lemma 7.2. *Suppose that $\mathcal{U}$ is an automatic $L_\Box$-structure and $\mathcal{V}$ is an automatic $L_\circ$-structure. Then the double box $\boxed{\mathcal{U}}\text{-}\textcircled{$\mathcal{V}$}$ has an automatic presentation.*

Proof. Without loss of generality, we assume that the alphabets of automata for $\mathcal{U}$ and $\mathcal{V}$ are disjoint. Then an automatic copy $\mathcal{A}$ of $\boxed{\mathcal{U}}\text{-}\textcircled{$\mathcal{V}$}$ is defined as follows.

The domain of $\mathcal{A}$ is the union of domains of $\mathcal{U}$ and $\mathcal{V}$. For any $P \in L_\Box$, let $P^{\mathcal{A}} := P^{\mathcal{U}}$. For $P \in L_\circ$, we set $P^{\mathcal{A}} := P^{\mathcal{V}}$. We define $R^{\mathcal{A}}_{box} := \{(u, v) : u \in \mathcal{U},\ v \in \mathcal{V}\}$ and $E^{\mathcal{A}}_{box} := dom(\mathcal{A})^2$. $\square$

Lemma 7.3. *Suppose that $\{\mathcal{U}_k\}_{k\in\omega}$ is a uniformly automatic sequence of $L_\Box$-structures and $\{\mathcal{V}_m\}_{m\in\omega}$ is a uniformly automatic sequence of $L_\circ$-structures. Then the structure consisting of infinitely many copies of $\bigsqcup_{k,m\in\omega}\boxed{\mathcal{U}_k}\text{-}\textcircled{$\mathcal{V}_m$}$ (where $\bigsqcup$ denotes the disjoint union of double boxes) has an automatic presentation $\mathcal{W}$. Furthermore, given automata witnessing the uniform automaticity of sequences $\{\mathcal{U}_k\}_{k\in\omega}$ and $\{\mathcal{V}_m\}_{m\in\omega}$, one can effectively produce an automaton witnessing the automaticity of $\mathcal{W}$.*

Proof Sketch. The proof is based on the idea from the previous lemma. Suppose that automata $\mathbf{A}$, $\mathbf{B}_0, \dots, \mathbf{B}_4$ witness the uniform automaticity of the sequence $\{\mathcal{U}_k\}_{k\in\omega}$, and automata $\mathbf{C}$, $\mathbf{D}_0$, $\mathbf{D}_1$, $\mathbf{D}_2$ witness the uniform automaticity of the sequence $\{\mathcal{V}_m\}_{m\in\omega}$.

We may assume that the alphabets of $\mathbf{A}$ and $\mathbf{C}$ are disjoint. We sketch how to construct a desired automatic presentation $\mathcal{W}$. The idea behind the construction is the following: Any double box $\boxed{\mathcal{I}}\text{-}\text{\textcircled{$\mathcal{J}$}}$ from our presentation $\mathcal{W}$ is described by three parameters $k, m, p \in \omega$. The parameter k says that the square box $\mathcal{I}$ is isomorphic to $\mathcal{U}_k$, m says that the round box $\mathcal{J}$ is isomorphic to $\mathcal{V}_m$, and p says that this double box is the pth copy of $\boxed{\mathcal{U}_k}\text{-}\text{\textcircled{$\mathcal{V}_m$}}$.

The domain of $\mathcal{W}$ is equal to

$$\{conv(bin(k), conv(bin(k), u), bin(m), bin(p)) : k, m, p \in \omega,\ conv(bin(k), u) \in L(\mathbf{A}) \cup L(\mathbf{C})\}.$$

If $conv(bin(k), u) \in L(\mathbf{A})$, then the corresponding elements from the domain lie in square boxes isomorphic to $\mathcal{U}_k$. Otherwise, these elements belong to round boxes isomorphic to $\mathcal{V}_k$. Here the number m describes the isomorphism type of the complementing box: e.g., if $conv(bin(k), u) \in L(\mathbf{A})$, then an element $conv(bin(k), conv(bin(k), u), bin(m), bin(p))$ belongs to a double box isomorphic to $\boxed{\mathcal{U}_k}\text{-}\text{\textcircled{$\mathcal{V}_m$}}$.

Recall that the automaton $\mathbf{B}_1$ is responsible for defining the relation $R_\Box$ in all $\mathcal{U}_k$, $k \in \omega$. We set

$$R_\Box^{\mathcal{W}} := \{(conv(bin(k), conv(bin(k), u), bin(m), bin(p)), conv(bin(k), conv(bin(k), v), bin(m), bin(p))) : \\ k, m, p \in \omega,\ conv(bin(k), u, v) \in L(\mathbf{B_1})\}.$$

Informally speaking, we essentially use the same automaton $\mathbf{B}_1$ to define the relation $R_\Box$ in our square boxes. Other relations from $L_\Box \cup L_\circ$ are defined in a similar way.

We set

$$R_{box}^{\mathcal{W}} := \{(conv(bin(k), conv(bin(k), u), bin(m), bin(p)), conv(bin(m), conv(bin(m), v), bin(k), bin(p))) : \\ k, m, p \in \omega,\ conv(bin(k), u) \in L(\mathbf{A}),\ conv(bin(m), v) \in L(\mathbf{C})\}.$$

The equivalence relation $E_{box}^{\mathcal{W}}$ is such that any its class has the form

$$\{conv(bin(k), conv(bin(k), u), bin(m), bin(p)),\ conv(bin(m), conv(bin(m), v), bin(k), bin(p)) : \\ conv(bin(k), u) \in L(\mathbf{A}),\ conv(bin(m), v) \in L(\mathbf{C})\}$$

for some $k, m, p \in \omega$. □

The next lemma describes the Σ_1^1 outcome.

Lemma 7.4. *If $n \notin O$, then $\mathcal{S}$ has an automatic presentation.*

Proof. Recall that $\mathcal{L}$ is isomorphic to $\mathcal{H}$. Consider any $\mathcal{R}_e$. By Lemma 4.1, (in this $\mathcal{R}_e$) there is an index i such that the structures $\mathcal{A}_{i+1}$ and $\mathcal{A}_0$ are isomorphic. Moreover, $\mathcal{C}$ is isomorphic to a structure from K. Therefore, the special double box $\boxed{\Psi(\mathcal{A}_0)}\text{-}\text{\textcircled{$\Phi(\mathcal{C})$}}$ is isomorphic to some non-special double box $\boxed{\Psi(\mathcal{A}_{i+1})}\text{-}\text{\textcircled{$\mathcal{F}_j$}}$.

Note that any non-special double box has infinitely many copies inside $\mathcal{S}$. Thus, our structure $\mathcal{S}$ is isomorphic to a computable structure $\mathcal{S}_{omit}$ which is obtained from $\mathcal{S}$ by deleting all special double boxes. Now it is sufficient to show that $\mathcal{S}_{omit}$ has an automatic presentation.

This time we need to distinguish the structures $\mathcal{A}_i$ arising from different $\mathcal{R}_e$, $e \in \omega$. Hence, let $\mathcal{A}_i^e$ denote the structure $\mathcal{A}_i$ which was built during the construction of $\mathcal{R}_e$. The proof of Lemma 4.1 shows that the sequence $\{\mathcal{A}_i^e\}_{e,i\in\omega}$ is uniformly Δ_2^0-computable. By Corollary 5.5, there is a uniformly automatic sequence of structures $\{\mathcal{G}_k\}_{k\in\omega}$ such that for any $e, i \in \omega$, we have $\mathcal{G}_{\langle e,i\rangle} \cong \Psi(\mathcal{A}_{i+1}^e)$.

Notice that the sequence $\{\mathcal{F}_m\}_{m\in\omega}$, which was used in the construction, is uniformly automatic. Moreover, $\mathcal{F}_0 \cong \Phi(\mathcal{X}_\omega)$, $\mathcal{F}_1 \cong \Phi(\mathcal{Y}_\omega)$, $\mathcal{F}_{2m} \cong \Phi(\mathcal{X}_m)$, and $\mathcal{F}_{2m+1} \cong \Phi(\mathcal{Y}_m)$ for any non-zero $m \in \omega$.

By Lemma 7.3, there is an automatic presentation $\mathcal{W}$ of the structure consisting of infinitely many copies of $\bigsqcup_{k,m\in\omega}\boxed{\mathcal{G}_k}\text{-}\text{\textcircled{$\mathcal{F}_m$}}$. It is easy to see that $\mathcal{W}$ is isomorphic to $\mathcal{S}_{omit}$. Therefore, $\mathcal{S}$ has an automatic presentation. □

This concludes the proof of Theorem 1.1.

8. Proof of Theorem 1.2

8.1. **Construction.** For starters, we reintroduce our preliminary arrangements as in Theorem 1.1. Let O be a Π^1_1 complete set, and choose a computable sequence of linear orders $\{\mathcal{L}_n\}_{n\in\omega}$ such that:

- if $n \in O$, then $\mathcal{L}_n$ is an ordinal;
- if $n \notin O$, then $\mathcal{L}_n$ is a copy of the Harrison linear order.

We will build a computable sequence of structures $\{\mathcal{S}_n\}_{n\in\omega}$ such that:

- if $n \in O$, then $\mathcal{S}_n$ has no primitive recursive copies;
- if $n \notin O$, then $\mathcal{S}_n$ has a polynomial-time copy on the domain $Bin(\omega)$.

As in Theorem 1.1, we fix a number n and describe the construction of the structure $\mathcal{S} := \mathcal{S}_n$. Again, $\mathcal{L}$ is an abbreviation for $\mathcal{L}_n$.

The language L_0 of the structure $\mathcal{S}$ contains one binary relation R and two unary functions f and g. A double box $\boxed{\mathcal{U}}$-$\textcircled{\mathcal{V}}$ is arranged as follows:

- First, we add two additional elements $c[\mathcal{U},\mathcal{V}]$ and $d[\mathcal{U},\mathcal{V}]$. The elements are used to "distinguish" a square box and a round box. For $x \in \{c[\mathcal{U},\mathcal{V}], d[\mathcal{U},\mathcal{V}]\}$, we set $f(x) = g(x) = d[\mathcal{U},\mathcal{V}]$.
- Roughly speaking, the square box $\boxed{\mathcal{U}}$ is a graph in the language $\{R\}$:
  - the condition $R(x, y)$ implies that x and y lie in the same square box;
  - for any x from $\boxed{\mathcal{U}}$, $f(x) := x$ and $g(x) := c[\mathcal{U},\mathcal{V}]$.
- The round box $\textcircled{\mathcal{V}}$ is a unary algebra in the language $\{f\}$: For any $x \in \mathcal{V}$, we have $f(x) \in \mathcal{V}$ and $g(x) := d[\mathcal{U},\mathcal{V}]$

We fix a computable list $\{\mathcal{M}_e\}_{e\in\omega}$ of all primitive recursive structures in the language L_0. Given a primitive recursive structure $\mathcal{M}_e$, we produce two computable sequences $\{\mathcal{U}_k\}_{k\in\omega}$ and $\{\mathcal{V}_k\}_{k\in\omega}$. As in Theorem 1.1, we want to treat our $\mathcal{M}_e$ as a disjoint union of infinitely many double boxes. The sequences satisfy the following properties:

- Any $\mathcal{U}_k$ is a total graph, and any $\mathcal{V}_k$ is a total unary algebra.
- If $\mathcal{M}_e$ has no elements x and y such that $x \neq y$, $f(x) = g(x) = f(y) = g(y) = y$, then every $\mathcal{U}_k$ is a singleton and every $\mathcal{V}_k$ is a copy of $\mathcal{X}_1$.
- If x and y are different elements from $\mathcal{M}_e$ with $f(x) = g(x) = f(y) = g(y) = y$, then there are infinitely many k such that:
  - If the set $U[x] := \{z : g(z) = x\}$ is empty, then $\mathcal{U}_k$ is a singleton. Otherwise, $\mathcal{U}_k$ is isomorphic to the $\{R\}$-substructure of $\mathcal{M}_k$ on the domain $U[x]$.
  - If the set $V[x] := \{z : z \neq x,\ z \neq y,\ g(z) = y\}$ is empty, then $\mathcal{V}_k \cong \mathcal{X}_1$. Otherwise, $\mathcal{V}_k$ is isomorphic to the disjoint union of $\mathcal{X}_1$ and the $\{f\}$-subalgebra of $\mathcal{M}_k$ on the domain $V[x]$.

Using arguments similar to those of Theorem 1.1, we may assume that for any k, either $\mathcal{U}_k$ is a singleton, or $\mathcal{U}_k$ contains an element labeled with $\boxed{\langle e, 0\rangle}$.

The desired structure $\mathcal{S}$ is built as a disjoint union of L_0-structures $\mathcal{R}_e$, $e \in \omega$. Again, if the orders $\mathcal{L}$ and $\mathcal{H}$ are not isomorphic, then $\mathcal{R}_e$ ensures that $\mathcal{S} \not\cong \mathcal{M}_e$.

The construction of $\mathcal{R}_e$ proceeds as follows. By Lemma 4.1, we construct a computable sequence of graphs $\{\mathcal{A}_i\}_{i\in\omega}$ and a computable function $f(x)$ such that:

- If $\mathcal{L} \cong \mathcal{H}$, then there is a non-zero i with $\mathcal{A}_i \cong \mathcal{A}_0$.
- If $\mathcal{L} \not\cong \mathcal{H}$ and the sequences $\{\mathcal{A}_i\}_{i\in\omega}$ and $\{\mathcal{U}_j\}_{j\in\omega}$ are equal up to isomorphism, then $\mathcal{A}_0$ is not isomorphic to any $\mathcal{A}_i$ with $i \neq 0$. Moreover, there is a limit $k := \lim_s f(s)$ and $\mathcal{A}_0 \cong \mathcal{U}_k$.

After that, we apply Lemma 4.6 and obtain a computable unary algebra $\mathcal{C}$ such that:

- $\mathcal{C}$ is isomorphic to a structure from the class K, and
- if the limit $k = \lim_s f(s)$ exists, then $\mathcal{C} \not\cong \mathcal{V}_k$.

The structure $\mathcal{R}_e$ is organized (in a natural way) as a disjoint union of a *special double box* $\boxed{\mathcal{A}_0}\text{-}\enclose{circle}{\mathcal{C}}$ and infinitely many copies of each of the following non-special double boxes: $\boxed{\mathcal{A}_{i+1}}\text{-}\enclose{circle}{\mathcal{X}_j}$, $\boxed{\mathcal{A}_{i+1}}\text{-}\enclose{circle}{\mathcal{Y}_j}$, where $i \in \omega$, $1 \leq j \leq \omega$.

8.2. **Verification.**

Lemma 8.1. *If $\mathcal{L} \ncong \mathcal{H}$, then $\mathcal{S}$ has no primitive recursive copies.*

Proof. Assume that $\mathcal{S}$ is isomorphic to $\mathcal{M}_e$. Consider the special double box $\boxed{\mathcal{A}_0}\text{-}\enclose{circle}{\mathcal{C}}$ from $\mathcal{R}_e$. By Lemma 4.1, the box is a unique double box in $\mathcal{S}$ such that its square box is isomorphic to $\mathcal{A}_0$. Moreover, there exists a limit $k = \lim_s f(s)$ such that $\mathcal{A}_0 \cong \mathcal{U}_k$. The box $\boxed{\mathcal{U}_k}\text{-}\enclose{circle}{\mathcal{V}_k}$ is a unique double box in $\mathcal{M}_e$ with $\mathcal{U}_k \cong \mathcal{A}_0$. This contradicts with $\mathcal{C} \ncong \mathcal{V}_k$. □

Lemma 8.2. *If $\mathcal{L} \cong \mathcal{H}$, then $\mathcal{S}$ has a polynomial-time copy on the domain $Bin(\omega)$.*

Proof. As in Lemma 7.4, it is easy to show that $\mathcal{S}$ is isomorphic to the structure $\mathcal{S}_{omit}$ which is obtained from $\mathcal{S}$ by omitting all special double boxes. For $e, i \in \omega$, let $\mathcal{A}_{e,i+1}$ denote the structure $\mathcal{A}_{i+1}$ which was built during the construction of $\mathcal{R}_e$.

Claim 8.3. *The disjoint union of $\mathcal{A}_{e,i+1}$, $i \in \omega$, has a polynomial-time copy $\mathcal{P}_e$ on the domain $Bin(\omega)$. Furthermore, there is a polynomial-time algorithm that given an element $x \in \mathcal{P}_e$ outputs a number j such that x belongs to the copy of $\mathcal{A}_{e,j}$ inside $\mathcal{P}_e$. This is uniform in $e \in \omega$.*

Proof Sketch. In order to build the desired $\mathcal{P}_e$, we modify the construction of Lemma 4.1. First, we will assume that the order $\mathcal{H}$ from the construction is a polynomial-time copy of Harrison linear order. Notice that the structure $\mathcal{A}_{e,1}$ is infinite: if we remove all its labels, we obtain a copy of $\mathcal{H}$.

The informal idea of the proof is as follows. If we cannot execute some procedure in a polynomial time, then we replace it with the following action: While waiting for the computation of the procedure to halt, we just build $\mathcal{A}_{e,1}$ as a copy of $\mathcal{H}$.

We give a list of time-consuming procedures (at a non-zero stage s) from the original construction and sketch how to make them polynomial-time:

a) *Building $\mathcal{A}_0[s;0]$.* Here we want to ensure that the structures $\mathcal{L}[s]$ and $R(\mathcal{A}_0[s;0])$ are isomorphic. Since a given order $\mathcal{L}$ may be not polynomial-time, this can be a hindrance to our construction. Nevertheless, this can be resolved: In place of $\mathcal{L}$, we will work with a polynomial copy of $\omega + \mathcal{L}$.
b) *Checking whether a stage s is expansionary.* This may be very time-consuming: in particular, we may need to look through *all* possible correct partial isomorphisms from, say, $\mathcal{A}_1[s-1]$ into $\mathcal{B}_0[s]$, where $\mathcal{B}_0$ is not even polynomial-time. In order to deal with it, we proceed as follows. We first go through s many stages of the checking. If the computation did not halt, then we declare that the time is *s-frozen* and go to the next stage. At a stage $t > s$, if the time is s-frozen, then the construction goes like this:
   - Add a fresh element to (our copy of) $R(\mathcal{A}_{e,1})$. This can be done in a polynomial time, since $\mathcal{H}$ is polynomial-time.
   - Do the first t stages of computation for checking whether s is an expansionary stage.
   - If the computation halts, then proceed as in stage s of the original construction of Lemma 4.1. After that, declare the time $(s+1)$-frozen.
   - If the computation does not halt, then the time is again s-frozen. Do not change anything in our structure and proceed to the next stage.
- *Redefining $flag(i,s)$ and $g_i[s]$.* Again, here we need to search for correct partial isomorphisms. This search can be modified similar to the previous procedure.

Furthermore, we may assume that at any stage s, we always put the binary string $bin(s)$ into our $\mathcal{P}_e$. Also, at a stage $\langle i, t\rangle$, one puts a fresh element only into either a copy of $\mathcal{A}_{e,1}$ or a copy of $\mathcal{A}_{e,i+1}$. Hence, the domain of the constructed $\mathcal{P}_e$ is equal to $Bin(\omega)$, and for $x \in Bin(\omega)$, one can compute in a polynomial time the index j such that x belongs to the copy of $\mathcal{A}_{e,j}$. □

Recall that any structure from the class K has an automatic presentation. Hence, any structure from K has a polynomial-time copy. Furthermore, this is uniform.

Let $\{\mathcal{P}_{e,q,r} : e, q, r \in \omega\}$ be a sequence of polynomial-time graphs such that for any $e, q, r \in \omega$, $\mathcal{P}_{e,q,r}$ is a polynomial-time copy of the disjoint union of $\mathcal{A}_{e,i+1}$, $i \in \omega$, on the domain $\{bin(4\langle e, q, r, x\rangle) : x \in \omega\}$. We also choose a sequence of polynomial-time unary algebras $\{\mathcal{F}_{e,k,q,r} : e, k, q, r \in \omega\}$ such that $\mathcal{F}_{e,k,q,r}$ has domain $\{bin(4\langle e, k, q, r, x\rangle + 1) : x \in \omega\}$, and for any e, k, q, the sequence $\{\mathcal{F}_{e,k,q,r}\}_{r\in\omega}$ lists all structures from K without repetition.

Then the polynomial-time copy of $\mathcal{S}_{omit}$ on the domain $Bin(\omega)$ is organized as follows. Let $e, k, q, r \in \omega$. We form a double box $\boxed{\mathcal{U}}$-$\textcircled{\mathcal{V}}$: $\mathcal{U}$ is a copy of $\mathcal{A}_{e,k+1}$ from $\mathcal{P}_{e,q,r}$; $\mathcal{V}$ is the structure $\mathcal{F}_{e,k,q,r}$; and we set $c[\mathcal{U}, \mathcal{V}] := bin(4\langle e, k, q, r\rangle + 2)$ and $d[\mathcal{U}, \mathcal{V}] := bin(4\langle e, k, q, r\rangle + 3)$. □

This concludes the proof of Theorem 1.2.

References

[AGR17] Faried Abu Zaid, Erich Grädel, and Frederic Reinhardt. Advice Automatic Structures and Uniformly Automatic Classes. In Valentin Goranko and Mads Dam, editors, *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*, volume 82 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:20, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[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. Existence and uniqueness of structures computable in polynomial time. *Algebra and Logic*, 55(1):72–76, 2016.

[Ben73] C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Develop.*, 17:525–532, 1973.

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

[BHS12] Dan Brumleve, Joel David Hamkins, and Philipp Schlicht. The mate-in-n problem of infinite chess is decidable. In *How the world computes*, volume 7318 of *Lecture Notes in Comput. Sci.*, pages 78–88. Springer, Heidelberg, 2012.

[BS11] G. Braun and L. Strüngmann. Breaking up finite automata presentable torsion-free abelian groups. *Internat. J. Algebra Comput.*, 21(8):1463–1472, 2011.

[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.

[CFG$^+$07] W. Calvert, E. Fokina, S. S. Goncharov, J. F. F. Knight, O. Kudinov, A. S. Morozov, and V. Puzarenko. Index sets for classes of high rank structures. *J. Symbolic Logic*, 72(4):1418–1432, 2007.

[CHK$^+$12] J. Carson, V. Harizanov, J. Knight, K. Lange, C. McCoy, A. Morozov, S. Quinn, C. Safranski, and J. Wallbaum. Describing free groups. *Trans. Amer. Math. Soc.*, 364(11):5715–5728, 2012.

[CR91] Douglas Cenzer and Jeffrey 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.

[CR95] D. Cenzer and J. Remmel. Feasibly categorical abelian groups. In *Feasible mathematics, II (Ithaca, NY, 1992)*, volume 13 of *Progr. Comput. Sci. Appl. Logic*, pages 91–153. Birkhäuser Boston, Boston, MA, 1995.

[CR99] D. Cenzer and J.B. Remmel. Polynomial time versus computable boolean algebras. *Recursion Theory and Complexity, Proceedings 1997 Kazan Workshop (M. Arslanov and S. Lempp eds.), de Gruyter*, pages 15–53, 1999.

[Del04] Christian Delhommé. Automaticité des ordinaux et des graphes homogènes. *C. R. Math. Acad. Sci. Paris*, 339(1):5–10, 2004.

[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.

[DM08] R. G. Downey and A. Montalbán. The isomorphism problem for torsion-free abelian groups is analytic complete. *J. Algebra*, 320(6):2291–2300, 2008.

[DM14] Rodney Downey and Alexander G. Melnikov. Computable completely decomposable groups. *Trans. Amer. Math. Soc.*, 366(8):4243–4266, 2014.

[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.

[FGK$^+$15] E. B. Fokina, S. S. Goncharov, V. Kharizanova, O. V. Kudinov, and D. Turetski. Index sets of n-decidable structures that are categorical with respect to m-decidable representations. *Algebra Logika*, 54(4):520–528, 544–545, 547–548, 2015.

[Fok07] E. B. Fokina. Index sets of decidable models. *Sibirsk. Mat. Zh.*, 48(5):1167–1179, 2007.

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

[Fuc70] L. Fuchs. *Infinite abelian groups. Vol. I.* Pure and Applied Mathematics, Vol. 36. Academic Press, New York, 1970.

[Fuc73] L. Fuchs. *Infinite abelian groups. Vol. II.* Academic Press, New York, 1973. Pure and Applied Mathematics. Vol. 36-II.

[GBM15a] S. S. Goncharov, N. A. Bazhenov, and M. I. Marchuk. The index set of Boolean algebras that are autostable relative to strong constructivizations. *Sibirsk. Mat. Zh.*, 56(3):498–512, 2015.

[GBM15b] S. S. Goncharov, N. A. Bazhenov, and M. I. Marchuk. Index sets of constructive models of natural classes that are autostable with respect to strong constructivizations. *Dokl. Akad. Nauk*, 464(1):12–14, 2015.

[GN02] S. S. Goncharov and Dzh. Naĭt. Computable structure and antistructure theorems. *Algebra Logika*, 41(6):639–681, 757, 2002.

[Gon77] S. S. Gončarov. The number of nonautoequivalent constructivizations. *Algebra i Logika*, 16(3):257–282, 377, 1977.

[Gri90] Serge Grigorieff. Every recursive linear ordering has a copy in dtime-space(n, log(n)). *J. Symb. Log.*, 55(1):260–276, 1990.

[Har68] J. Harrison. Recursive pseudo-well-orderings. *Trans. Amer. Math. Soc.*, 131:526–543, 1968.

[Hjo02] Greg Hjorth. The isomorphism relation on countable torsion free abelian groups. *Fund. Math.*, 175(3):241–257, 2002.

[Hod82] Bernard R. Hodgson. On direct products of automaton decidable theories. *Theoret. Comput. Sci.*, 19(3):331–335, 1982.

[HT17] M. Harrison-Trainor. There is no classification of the decidably presentable structures. Preprint, 2017.

[JKS$^+$17] S. Jain, B. Khoussainov, F. Stephan, D. Teng, and S. Zou. Semiautomatic structures. *Theory Comput. Syst.*, 61(4):1254–1287, 2017.

[JKS18] S. Jain, B. Khoussainov, and F. Stephan. Finitely generated semiautomatic groups. *Computability*, 7(2–3):273–287, 2018.

[KKM14] O. Kharlampovich, B. Khoussainov, and A. Myasnikov. From automatic structures to automatic groups. *Groups, Geometry, and Dynamics*, 8(1):157–198, 2014.

[KLM09] B. Khoussainov, J. Liu, and M. Minnes. Unary automatic graphs: an algorithmic perspective. *Math. Struct. Comput. Sci.*, 19(1):133–152, 2009.

[KM09] Bakhadyr Khoussainov and Mia Minnes. Model-theoretic complexity of automatic structures. *Ann. Pure Appl. Logic*, 161(3):416–426, 2009.

[KM10] Bakhadyr Khoussainov and Mia Minnes. Three lectures on automatic structures. In *Logic Colloquium 2007*, volume 35 of *Lect. Notes Log.*, pages 132–176. Assoc. Symbol. Logic, La Jolla, CA, 2010.

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

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

[KN94] Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. In *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13-16 October 1994*, pages 367–392, 1994.

[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.

[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.

[KNRS07] Bakhadyr Khoussainov, André Nies, Sasha Rubin, and Frank Stephan. Automatic structures: richness and limitations. *Log. Methods Comput. Sci.*, 3(2):2:2, 18, 2007.

[KR03] B. Khoussainov and S. Rubin. Automatic structures: Overview and future directions. *J. Autom. Lang. Comb.*, 8(2):287–301, 2003.

[Lac70] A. H. Lachlan. On some games which are relevant to the theory of recursively enumerable sets. *Ann. of Math. (2)*, 91:291–310, 1970.

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

[LS07] S. Lempp and T. A. Slaman. The complexity of the index sets of $\aleph_0$-categorical theories and of Ehrenfeucht theories. In *Advances in logic*, volume 425 of *Contemp. Math.*, pages 43–47. Amer. Math. Soc., Providence, RI, 2007.

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

[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.

[MW12] Charles McCoy and John Wallbaum. Describing free groups, Part II: Π^0_4 hardness and no Σ^0_2 basis. *Trans. Amer. Math. Soc.*, 364(11):5729–5734, 2012.

[NR90] A. Nerode and J. B. Remmel. Polynomial time equivalence types. In *Logic and computation (Pittsburgh, PA, 1987)*, volume 106 of *Contemp. Math.*, pages 221–249. Amer. Math. Soc., Providence, RI, 1990.

[NS09] André Nies and Pavel Semukhin. Finite automata presentable abelian groups. *Ann. Pure Appl. Logic*, 161(3):458–467, 2009.

[Nur74] A. T. Nurtazin. Strong and weak constructivization and computable families. *Algebra and Logic*, 13(3):177–184, 1974.

[OT05] Graham P. Oliver and Richard M. Thomas. Automatic presentations for finitely generated groups. In *STACS 2005*, volume 3404 of *Lecture Notes in Comput. Sci.*, pages 693–704. Springer, Berlin, 2005.

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

[Rig] K. Riggs. The decomposability problem for torsion-free abelian groups is analytic complete. Proceedings of the American Mathematical Society (to appear).

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

[Sel76] V. L. Selivanov. Enumerations of families of general recursive functions. *Algebra and Logic*, 15(2):128–141, 1976.

[Soa87] R. Soare. *Recursively enumerable sets and degrees.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1987. A study of computable functions and computably generated sets.

[Spe55] Clifford Spector. Recursive well-orderings. *J. Symb. Logic*, 20:151–163, 1955.

[Tho03] S. Thomas. The classification problem for torsion-free abelian groups of finite rank. *J. Amer. Math. Soc.*, 16(1):233–258, 2003.

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

[vdW30] B. van der Waerden. Eine Bemerkung über die Unzerlegbarkeit von Polynomen. *Math. Ann.*, 102(1):738–739, 1930.

Sobolev Institute of Mathematics and Novosibirsk State University, Novosibirsk, Russia
E-mail address: nickbazh@yandex.ru
URL: http://bazhenov.droppages.com

Department of Pure Mathematics, University of Waterloo, ON, Canada N2L 3G1
E-mail address: maharris@uwaterloo.ca
URL: http://www.math.uwaterloo.ca/∼maharris/

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

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

Nanyang Technological University
E-mail address: kmng@ntu.edu.sg