The Isomorphism Problem On Classes of Automatic Structures with Transitive Relations

Dietrich Kuske[1], Jiamou Liu[2], and Markus Lohrey[2, ⋆]

[1] TU Ilmenau, Institut für Theoretische Informatik, Germany
[2] Universität Leipzig, Institut für Informatik, Germany
dietrich.kuske@tu-ilmenau.de, liujiamou@gmail.com, lohrey@informatik.uni-leipzig.de

Abstract. Automatic structures are finitely presented structures where the universe and all relations can be recognized by finite automata. It is known that the isomorphism problem for automatic structures is complete for Σ_1^1, the first existential level of the analytical hierarchy. Positive results on ordinals and on Boolean algebras raised hope that the isomorphism problem is simpler for transitive relations. We prove that this is not the case. More precisely, this paper shows:

(i) The isomorphism problem for automatic equivalence relations is complete for Π_1^0 (first universal level of the arithmetical hierarchy).
(ii) The isomorphism problem for automatic trees of height $n \geq 2$ is Π_{2n-3}^0-complete.
(iii) The isomorphism problem for well-founded automatic order trees is recursively equivalent to true first-order arithmetic.
(iv) The isomorphism problem for automatic order trees is Σ_1^1-complete.
(v) The isomorphism problem for automatic linear orders is Σ_1^1-complete.

We also obtain Π_1^0-completeness of the elementary equivalence problem for several classes of automatic structures and Σ_1^1-completeness of the isomorphism problem for trees (resp., linear orders) consisting of a deterministic context-free language together with the prefix order (resp., lexicographic order). This solves several open questions of Ésik, Khoussainov, Nerode, Rubin, and Stephan.

1 Introduction

The idea of an automatic structure goes back to Büchi and Elgot who used finite automata to decide, e.g., Presburger arithmetic [12]. Automaton decidable theories [19] and automatic groups [14] are similar concepts. A systematic study was initiated by Khoussainov and Nerode [24] who also coined the name "*automatic structure*". In essence, a structure is automatic if the elements of the universe can be encoded by strings from a regular language and every relation of the structure can be recognized by a finite state automaton with several heads that proceed synchronously. Automatic structures received increasing interest over the last years [3,6,22,23,25,26,27,30,32,38]. One of the main motivations for investigating automatic structures is that their first-order theories can be decided uniformly (i.e., the input is an automatic presentation and a first-order sentence).

Automatic structures form a subclass of computable (or recursive) structures. A structure is computable, if its domain as well as all relations are computable sets of finite words (or naturals). A typical example of a computable structure is $(\mathbb{N}; +, \times)$. Computable structures are the starting point of the rich field of computable model theory [1,15]. A well-studied problem in this field is the isomorphism problem, which asks whether two given computable structures over the same signature (encoded by Turing-machines for the domain and all relations) are isomorphic. It is well known that the isomorphism problem for computable structures is complete for the first level of the analytical hierarchy Σ_1^1. In fact, Σ_1^1-completeness holds for many subclasses of computable structures, e.g., for linear orders, trees, undirected graphs, Boolean algebras, Abelian p-groups, see [8,16]. These papers also show, as a consequence of the Σ_1^1-completeness of the isomorphism problem, that these classes do not have a "good classification" (more precisely: they do not have a hyperarithmetical Friedberg enumeration).

⋆ The second and third author are supported by the DFG research project GELO.

In [26], it was shown that also for automatic structures the isomorphism problem is Σ^1_1-complete. By a direct interpretation, it follows that for the following classes the isomorphism problem is still Σ^1_1-complete [34]: automatic successor trees, automatic undirected graphs, automatic commutative monoids, automatic partial orders (of height 2), automatic lattices (of height 4), and automatic unary functions. On the other hand (and in contrast to computable structures), the isomorphism problem is decidable for automatic ordinals [27] and automatic Boolean algebras [26]. An intermediate class is the class of all locally-finite automatic graphs, for which the isomorphism problem is complete for Π^0_3 (third level of the arithmetical hierarchy) [37].

Although the interpretation technique from [34] yields undecidability for the isomorphism problem for automatic partial orders, it seems to be difficult to extend this technique to more restricted classes like order trees (i.e., trees seen as particular partial orders) or linear orders. Moreover, the automatic graphs constructed in [26] are based on transition graphs of Turing-machines. But the transitive closure of such a transition graph is in general not automatic (its first-order theory is undecidable in general). Hence, the techniques from [26,34] do not work for automatic order trees, automatic linear orders, and also automatic equivalence relations. Moreover, as mentioned above, for some very restricted classes of automatic structures (ordinals [27], Boolean algebras [26]), the isomorphism problem is decidable. Recent surveys consequently ask whether the isomorphism problem is decidable for automatic equivalence relations and automatic linear orders [25,38]. For automatic equivalence relations, it is conjectured in [25] that the isomorphism problem is decidable. For automatic linear orders, already [27] asked for the decidability status of the isomorphism problem.

Contrary to the hopes expressed in [25,27,38], our main results are:

- The isomorphism problem for automatic equivalence relations is Π^0_1-complete.
- The isomorphism problem for automatic trees of finite height $n \geq 2$ (where the height of a tree is the maximal number of edges along a path) is Π^0_{2n-3}-complete.
- The isomorphism problem for automatic well-founded order trees is recursively equivalent to true arithmetic (the first-order theory of $(\mathbb{N}; +, \times)$).
- The isomorphism problem for automatic order trees is Σ^1_1-complete.
- The isomorphism problem for automatic linear orders is Σ^1_1-complete.

Contrary to the above-mentioned technique using configuration graphs of Turing machines, our proofs are based on the undecidability of Hilbert's 10^{th} problem. Recall that Matiyasevich proved that every recursively enumerable set of natural numbers is Diophantine, i.e., the projection to the first component of the set of zeros of some polynomial $p(\overline{x}) \in \mathbb{Z}[\overline{x}]$ [33]. Honkala inferred that it is undecidable whether the range of a rational power series equals $\mathbb{N}$ [20]. His basic idea was to construct from a given polynomial $p(\overline{x})$ in several variables a finite automaton, such that on a word encoding an input tuple $\overline{c}$ for the polynomial, the automaton has exactly $p(\overline{c})$ many accepting runs, see also [39, Theorem II.11.1]. Using a similar encoding, we show that the isomorphism problem for automatic equivalence relations is Π^0_1-complete, which answers Question 4.2 from [25] negatively. By the same arguments, we obtain that elementary equivalence of automatic equivalence relations (i.e., the problem whether two equivalence relations have the same first-order theory) is Π^0_1-complete as well, thereby partially answering Question 4.8 from [25].

Next, we extend our technique in order to show that the isomorphism problem for automatic successor trees of height $n \geq 2$ is Π^0_{2n-3}-complete. Using the tight correspondence between equivalence structures and trees of height 2, our result for equivalence relations makes up, in some sense, the induction base $n = 2$. Our result for automatic trees of finite height gives an answer to Question 4.6 from [25], which asks for natural classes of automatic structures, for which the isomorphism problem is complete for levels of the arithmetical hierarchy.

For arbitrary automatic trees, we prove that the isomorphism problem has maximal complexity, i.e., that it is Σ^1_1-complete. For automatic successor trees, Σ^1_1-completeness of the isomorphism problem was already shown in [26]. Here, the crucial point is that we consider order trees (for trees of finite height, the distinction between order trees and successor trees is not important). Our proof for Σ^1_1-completeness on automatic order trees is based on a reduction from the isomorphism problem for computable trees. To achieve this reduction, we combine our techniques for automatic trees of finite height with the method from [26].

As a corollary of our construction, we can show that the following problem is Σ_1^1-complete as well: Given two deterministic pushdown automata P_1 and P_2 such that $\varepsilon \in L(P_1) \cap L(P_2)$, are the trees $(L(P_1); \preceq)$ and $(L(P_2); \preceq)$ (where $\preceq$ denotes the prefix relation) isomorphic. Recently, it was shown that the same problem with P_1 and P_2 nondeterministic (resp., deterministic) finite automata is EXPTIME-complete (resp. P-complete) [31].

Finally, using a similar but technically more involved reduction than those for trees, we can show that also the isomorphism problem for automatic linear orders is Σ_1^1-complete. This answers Question 4.3 (and consequently Question 4.7) from [25] negatively. From this proof, we obtain two further results as well:

- Elementary equivalence of automatic linear orders is Π_1^0-complete (giving another partial answer to Question 4.8 from [25]).
- The isomorphism problem for linear orders of the form $(L; \leq_{\mathsf{lex}})$, where L is a deterministic context-free language and $\leq_{\mathsf{lex}}$ is the lexicographic order on strings (these are the algebraic linear orders [4]), is Σ_1^1-complete.

The latter result answers a question of Ésik [13], where he proved undecidability of the isomorphism problem for linear orders of the form $(L; \leq_{\mathsf{lex}})$ with L a context-free language and $\leq_{\mathsf{lex}}$ the lexicographic order. The same problem is decidable for regular languages instead of deterministic context-free languages [41]. A polynomial time algorithm for the case that the regular language is given by a deterministic automaton can be found in [31].

For the important subclass of automatic scattered linear orders, we can provide a reduction of the isomorphism problem to true arithmetic. Although non-arithmetical, true arithmetic belongs to a relatively low level of the hyperarithmetical hierarchy. This shows that the isomorphism problem for scattered linear orders is strictly simpler than for general linear orders. It remains open, whether the isomorphism problem for scattered linear orders is decidable. It seems that our new techniques cannot help to solve this problem: Our proof for linear orders uses shuffle sums, and this construction always yields non-scattered linear orders.

Related work. Beyond the works cited so far, we should mention that Blumensath and Grädel [6] were the first to prove undecidability of the isomorphism problem for automatic structures. The paper [22] studies several methods of constructing automatic equivalence structures with different types of isomorphism invariants. Automatic linear orders were also studied in unpublished work by Delhommé [10], where it was shown that an ordinal is automatic if and only if it is strictly below ω^ω. Extending Delhommé's technique, Khoussainov et al. [27] prove that all automatic linear orders have finite Hausdorff ranks. In the same paper [27], automatic order trees are studied and it is shown that the Cantor-Bendixson rank of any automatic order tree is finite. The paper [23] constructs automatic structures of high ordinal height and Scott rank, again by using transition graphs of Turing-machines. Recently, we extended our techniques to ω-automatic structures (which are represented by Büchi-automata instead of ordinary finite automata) [29], where we proved that the isomorphism problem for ω-automatic structures (even ω-automatic trees of finite height) does not belong to the analytical hierarchy. Lastly, we mention that for equivalence structures, linear orders, and order trees which have automatic presentations over a unary alphabet, the isomorphism problem is decidable in polynomial time [32].

2 Preliminaries

Let $\mathbb{N}_+ = \{1, 2, 3, \ldots\}$. Let $p(x_1, \ldots, x_n) \in \mathbb{N}[x_1, \ldots, x_n]$ be a polynomial with non-negative integer coefficients. We define

$$\mathsf{Img}_+(p) = \{p(y_1, \ldots, y_n) \mid y_1, \ldots, y_n \in \mathbb{N}_+\}\,.$$

If p is not the zero-polynomial, then $\mathsf{Img}_+(p) \subseteq \mathbb{N}_+$.

For any alphabet A, we denote with $\preceq$ the prefix order on the set A^* of finite words over A. For a subset $L \subseteq A^*$ we denote with $\preceq_L$ the restriction of $\preceq$ to L.

2.1 Logic

A *relational structure* $\mathcal{S}$ consists of a *domain* D and finitely many atomic relations on the set D. It will be denoted by $(D; R_1, \dots, R_n)$, where $R_1, \dots, R_n$ are the atomic relations of $\mathcal{S}$. The *signature* of $\mathcal{S}$ is the tuple consisting of the arities of all relations R_i. We will only consider structures with countable domains. Occasionally, we will write $a \in \mathcal{S}$ for $a \in D$. If $\mathcal{S}_1$ and $\mathcal{S}_2$ are two structures with the same signature and with disjoint domains, then we write $\mathcal{S}_1 \uplus \mathcal{S}_2$ for the union of the two structures. Hence, when writing $\mathcal{S}_1 \uplus \mathcal{S}_2$, we implicitly express that the domains of $\mathcal{S}_1$ and $\mathcal{S}_2$ are disjoint. More generally, if $\{\mathcal{S}_i \mid i \in I\}$ is a class of pairwise disjoint structures with the same signature, then we denote with $\biguplus\{\mathcal{S}_i \mid i \in I\}$ the union of these structures.

We assume that the reader has some familiarity with first-order logic (briefly FO) and second-order logic, see e.g. [18] for more details. Recall that in second-order logic there are first-order variables (denoted by lower case letters) and second-order variables of arbitrary arity (denoted by upper case letters), which can be quantified existentially and universally. First-order variables range over elements of the domain of the underlying structure, whereas second-order variables range over relations of the appropriate arity. With $\mathcal{S} \models \varphi$ we denote the fact that the formula φ evaluates to true in the structure $\mathcal{S}$ (which has to have the appropriate signature); if φ has free variables then appropriate values have to get assigned to these variables. If $\varphi(x)$ is an FO-formula, x is a first-order variable, and $m \in \mathbb{N}$, then we will also allow the formulas $\exists^{\geq m} x : \varphi(x)$ and $\exists^{=m} x : \varphi(x)$ with the following meanings: $\mathcal{S} \models \exists^{\geq m} x : \varphi(x)$ (resp., $\mathcal{S} \models \exists^{=m} x : \varphi(x)$) if there exist at least (resp., exactly) m many elements $a \in \mathcal{S}$ with $\mathcal{S} \models \varphi(a)$. Clearly these quantifiers do not increase the expressiveness of FO. Moreover, the quantifier $\exists^{\geq m} x$ can be replaced by a block of m ordinary existential quantifiers.

Definition 1. *Two structures $\mathcal{S}$ and $\mathcal{S}'$ are* elementary equivalent *(denoted $\mathcal{S} \equiv \mathcal{S}'$) if for all first-order formulas φ without free variables, we have:*

$$\mathcal{S} \models \varphi \iff \mathcal{S}' \models \varphi .$$

FSO (for "fragment of second order logic") [30] is a proper extension of FO by the following three formation rules ($\mathcal{S}$ is again a structure with the appropriate signature):

- If $\varphi(x)$ is an FSO-formula and x a first-order variable, then $\exists^{=\infty} x : \varphi$ is also an FSO-formula, which has the following meaning: $\mathcal{S} \models \exists^{=\infty} x : \varphi(x)$ if and only if there are infinitely many $a \in \mathcal{S}$ with $\mathcal{S} \models \varphi(a)$.
- If $\varphi(x)$ is an FSO-formula, x a first-order variable, and $p \in \mathbb{N}_+$, then $\exists^{(p)} x\, \varphi$ is also an FSO-formula, which has the following meaning $\mathcal{S} \models \exists^{(p)} x\, \varphi$ if and only if the number of $a \in \mathcal{S}$ with $\mathcal{S} \models \varphi(a)$ is finite and a multiple of p.
- If X is an n-ary second-order variable that occurs only negatively (i.e., in the range of an odd number of negations) in the FSO-formula $\varphi(X)$, then $\exists X$ infinite : $\varphi(X)$ is also an FSO-formula, which has the following meaning: $\mathcal{S} \models \exists X$ infinite : $\varphi(X)$ if and only if there exists an infinite relation $R \subseteq \mathcal{S}^n$ such that $\mathcal{S} \models \varphi(R)$.[1]

Example 2. FSO allows to express that a graph with edge relation E has an infinite clique by the formula

$$\exists X \text{ infinite} \, \forall x, y : x, y \in X \wedge x \neq y \to (x, y) \in E$$

(X is a unary second-order variable). Note that this formula is equivalent to

$$\exists X \text{ infinite} \, \forall x, y : x \notin X \vee y \notin X \vee x = y \vee (x, y) \in E,$$

i.e., X occurs indeed only negatively.

[1] The condition that X occurs only negatively in $\varphi(X)$ ensures that if $\mathcal{S} \models \varphi(R)$ then $\mathcal{S} \models \varphi(Q)$ for every subset $Q \subseteq R$.

$$\begin{tikzcd}
 & \Pi^0_1 \arrow[dr, dash] & & \Pi^0_2 \arrow[dr, dash] & & \Pi^0_3 \arrow[dr, dash] & & & \\
\Sigma^0_1 \cap \Pi^0_1 \arrow[ur, dash] \arrow[dr, dash] & & \Sigma^0_2 \cap \Pi^0_2 \arrow[ur, dash] \arrow[dr, dash] & & \Sigma^0_3 \cap \Pi^0_3 \arrow[ur, dash] \arrow[dr, dash] & & \Sigma^0_4 \cap \Pi^0_4 & \cdots & \Sigma^1_1 \\
 & \Sigma^0_1 \arrow[ur, dash] & & \Sigma^0_2 \arrow[ur, dash] & & \Sigma^0_3 \arrow[ur, dash] & & &
\end{tikzcd}$$

Fig. 1. The arithmetical hierarchy and Σ^1_1

2.2 Computability

We assume some familiarity with the basic concepts of computability theory, see e.g. [35,40] for more details. An *index* for a computable function f is a Turing machine (or the Gödel number of a Turing machine) that computes f. With Σ^0_n we denote the n^{th} (existential) level of the arithmetical hierarchy; it is the class of all subsets $A \subseteq \mathbb{N}$ such that there exists a computable predicate $P \subseteq \mathbb{N}^{n+1}$ with

$$A = \{a \in \mathbb{N} \mid \exists x_1 \forall x_2 \cdots Q x_n : (a, x_1, \ldots, x_n) \in P\}, \tag{1}$$

where $Q = \exists$ ($Q = \forall$) for n odd (even). The set of complements of Σ^0_n-sets is denoted by Π^0_n. The *arithmetical hierarchy* is $\bigcup_{n\geq 0} \Sigma^0_n = \bigcup_{n\geq 0} \Pi^0_n$. There is a lot of freedom in the definition of the classes Σ^0_n and Π^0_n. Instead of requiring P in (1) to be a recursive predicate, one can take a quantifier-free FO-formula over the structure $(\mathbb{N}; +, \times)$ (then the last quantifier Qx_n has to be replaced by a block of quantifiers of type Q). Moreover, each of the quantifiers in (1) can be replaced by a block of quantifiers of the same type (since a tuple of naturals can be encoded by a single natural using a computable coding function). In particular, the quantifier $\exists^{\geq m} x$ is allowed in place of an ordinary existential quantifier.[2] Similarly, one may allow first-order quantifiers that range over finite objects of a particular type, e.g., words from a regular language, finite automata, computable mappings (represented by Turing machines), etc..

For lower bounds, we will use the following characterizations of the classes Π^0_m, see [35, Theorem XVIII], where the result is attributed to Kreisel, Shoenfield, and Wang.

Proposition 3. *The class Π^0_{2n} is the class of all sets of the form*

$$\{a \in \mathbb{N} \mid \exists^\infty x_1 \exists^\infty x_2 \cdots \exists^\infty x_n : (a, x_1, \ldots, x_n) \in P\},$$

where $P \subseteq \mathbb{N}^{n+1}$ is a computable predicate. The class Π^0_{2n+1} is the class of all sets of the form

$$\{a \in \mathbb{N} \mid \exists^\infty x_1 \exists^\infty x_2 \cdots \exists^\infty x_n \forall y : (a, x_1, \ldots, x_n, y) \in P\},$$

where $P \subseteq \mathbb{N}^{n+2}$ is a computable predicate.

If quantifiers over arbitrary subsets of natural numbers are allowed, one moves from the arithmetical hierarchy to the so called *analytical hierarchy.* We will only need the first (existential) level Σ^1_1 of this hierarchy. It is the class of all subsets of $\mathbb{N}$ of the form $\{n \in \mathbb{N} \mid \exists A \subseteq \mathbb{N} : (\mathbb{N}; +, \times) \models \varphi(A, n)\}$, where $\varphi(A, n)$ is an FO-formula (the subset A is used as an additional unary predicate). Figure 1 shows an inclusion diagram. By fixing some effective encoding of strings by natural numbers, we can talk about Σ^0_n-sets, Π^0_n-sets, and Σ^1_1-sets of strings over an arbitrary alphabet. Statements about the completeness of a set for one of the above classes will always refer to many-one reductions. A typical example of a Σ^1_1-set, which does not belong to the arithmetical hierarchy and which is not Σ^1_1-complete is *true arithmetic*, i.e., the first-order theory of $(\mathbb{N}; +, \times)$, which we denote by $\mathsf{FOTh}(\mathbb{N}; +, \times)$. In terms of the hyperarithmetical hierarchy,[3] $\mathsf{FOTh}(\mathbb{N}; +, \times)$ is complete for its first non-arithmetical level Δ^0_ω.

[2] On the other hand, the exact counting quantifier $\exists^{=m} x$ introduces an additional quantifier alternation and therefore does not preserve the levels of the arithmetical hierarchy.

[3] The hyperarithmetical hierarchy is a kind of transfinite extension of the arithmetical hierarchy. The class of all hyperarithmetical sets is $\Sigma^1_1 \cap \Pi^1_1$, where Π^1_1 is the set of complements of Σ^1_1-sets. See [35] for more details.

2.3 Automata

We assume basic terminologies and notations from automata theory, see, for example, [21]. As usual, we denote with Σ^+ the set of all non-empty words on the alphabet Σ. For a fixed alphabet Σ, a *(non-deterministic finite) automaton* is a tuple $\mathcal{A} = (S, \Delta, I, F)$ where S is the finite set of states, $\Delta \subseteq S \times \Sigma \times S$ is the transition relation, $I \subseteq S$ is a set of initial states, and $F \subseteq S$ is the set of accepting states. For technical reasons, we want to exclude the empty word from the language accepted by an automaton. Hence, we require $I \cap F = \emptyset$ in the following. A *run* of $\mathcal{A}$ on a word $u = a_1 a_2 \cdots a_n$ ($n > 0$, $a_1, a_2 \ldots, a_n \in \Sigma$) is a word over Δ of the form $r = (q_0, a_1, q_1)(q_1, a_2, q_2) \cdots (q_{n-1}, a_n, q_n)$. It is *accepting* if $q_0 \in I$ and $q_n \in F$. For $u \in \Sigma^+$, we denote by $\mathsf{Run}(\mathcal{A}, u) \subseteq \Delta^+$ the set of all accepting runs of $\mathcal{A}$ on the word u. Since we assume $I \cap F = \emptyset$, we can define the language $L(\mathcal{A})$ accepted by $\mathcal{A}$ as $\{u \in \Sigma^+ \mid \mathsf{Run}(\mathcal{A}, u) \neq \emptyset\}$. For $K \subseteq \Sigma^+$ let $\mathsf{Run}(\mathcal{A}, K) = \bigcup_{u \in K} \mathsf{Run}(\mathcal{A}, u)$, and let $\mathsf{Run}(\mathcal{A}) = \mathsf{Run}(\mathcal{A}, \Sigma^+)$ be the set of all accepting runs of $\mathcal{A}$. This is a regular language: A finite automaton for $\mathsf{Run}(\mathcal{A})$ can be obtained by replacing every transition $(p, a, q) \in \Delta$ by $(p, (p, a, q), q)$. For the accepting run $r \in \mathsf{Run}(\mathcal{A})$, we write $\mathsf{lab}(r)$ for the word accepted by r, i.e., $r \in \mathsf{Run}(\mathcal{A}, \mathsf{lab}(r))$. We state the following fact which is used at several occasions in the paper: for a given automaton $\mathcal{A}$, one can compute effectively the cardinality $|L(\mathcal{A})| \in \mathbb{N} \cup \{\aleph_0\}$ of the language accepted by $\mathcal{A}$.

We use *synchronous n-tape automata* to recognize n-ary relations. Such automata have n input tapes, each of which contains one of the input words. The n tapes are read in parallel until all input words are processed. Formally, let $\Sigma_\diamond = \Sigma \cup \{\diamond\}$ where $\diamond \notin \Sigma$. For words $w_1, w_2, \ldots, w_n \in \Sigma^*$, their *convolution* $w_1 \otimes \cdots \otimes w_n$ or $\otimes(w_1, \ldots, w_n)$ is a word in $(\Sigma_\diamond^n)^*$ of length $\max\{|w_1|, \ldots, |w_n|\}$, and the k^{th} symbol of $w_1 \otimes \cdots \otimes w_n$ is $(\sigma_1, \ldots, \sigma_n)$ where σ_i is the k^{th} symbol of w_i if $k \leq |w_i|$, and $\sigma_i = \diamond$ otherwise. We lift this definition to sets of words in the obvious way, i.e., for languages $L_1, L_2 \subseteq \Sigma^*$ let $L_1 \otimes L_2 = \{w_1 \otimes w_2 \mid w_1 \in L_1, w_2 \in L_2\}$. For an n-ary relation $R \subseteq (\Sigma^*)^n$, let $R^\otimes$ be the set of convolutions of tuples from R. Then R is *FA recognizable* if $R^\otimes$ is a regular language.

2.4 Automatic structures

A structure $\mathcal{S}$ is called *automatic* over Σ if its domain is a regular subset of Σ^* and each of its atomic relations is FA recognizable; any tuple $\mathcal{P}$ of automata that accept the domain and the relations of $\mathcal{S}$ is called an *automatic presentation of* $\mathcal{S}$; in this case, we write $\mathcal{S}(\mathcal{P})$ for $\mathcal{S}$. If an automatic structure $\mathcal{S}$ is isomorphic to a structure $\mathcal{S}'$, then $\mathcal{S}$ is called an *automatic copy* of $\mathcal{S}'$ and $\mathcal{S}'$ is *automatically presentable*. In this paper we sometimes abuse the terminology referring to $\mathcal{S}'$ as simply automatic and calling an automatic presentation of $\mathcal{S}$ also automatic presentation of $\mathcal{S}'$. We also simplify our statements by saying "given/compute an automatic structure $\mathcal{S}$" for "given/compute an automatic presentation $\mathcal{P}$ of a structure $\mathcal{S}(\mathcal{P})$".

Example 4. The following structures are known to be automatic:

- Finite structures
- $(\mathbb{N}; \leq, +)$
- $(\mathbb{Q}; \leq)$
- Every ordinal $< \omega^\omega$
- Transition graphs of Turing machines, i.e., graphs where the set of nodes is the set of all configurations of a fixed Turing machine M and there is an edge from configuration c_1 to configuration c_2 if M can move in one step from c_1 to c_2.

On the other hand, the following structures have no automatic copies:

- $(\mathbb{N}; \times)$ [5]
- Every ordinal $\geq \omega^\omega$ [10]
- Every infinite field [26]
- $(\mathbb{Q}; +)$ [42]

Well-known examples of automatic linear orders are the *lexicographic order* $\leq_{\mathsf{lex}}$ and the *length-lexicographic order* $\leq_{\mathsf{llex}}$ on a regular language D. To define them, we first need a fixed linear order $\leq$ on the alphabet Σ of D. For $w, w' \in D$, we say that w is *lexicographically less* than w', denoted by $w <_{\mathsf{lex}} w'$, if either w is a proper prefix of w' or there exist $x, y, z \in \Sigma^*$ and $\sigma, \tau \in \Sigma$ such that $w = x\sigma y$, $w' = x\tau z$, and $\sigma < \tau$. We write $w \leq_{\mathsf{lex}} w'$ if either $w = w'$ or $w <_{\mathsf{lex}} w'$. Furthermore, $w \leq_{\mathsf{llex}} w'$ if $|w| < |w'|$ or ($|w| = |w'|$ and $w \leq_{\mathsf{lex}} w'$). For convenience, in this paper, we use $\leq_{\mathsf{lex}}$ to denote the lexicographic order regardless of the corresponding alphabets and orders on the alphabets. The precise definition of $\leq_{\mathsf{lex}}$ in different occurrences will be clear from the context. Note that $\leq_{\mathsf{llex}}$ is always a well-order. Hence, every automatic structure can be expanded by a well-order on its domain.

The following theorem from [5,19,24,30,37] lays out the main motivation for investigating automatic structures.

Theorem 5. *From an automatic presentation $\mathcal{P}$ and an FSO-formula $\varphi(x_1, \ldots, x_n)$ in the signature of $\mathcal{S}(\mathcal{P})$, one can compute an automaton for the set $\{(v_1, \ldots, v_n) \in \mathcal{S}(\mathcal{P})^n \mid \mathcal{S}(\mathcal{P}) \models \varphi(v_1, \ldots, v_n)\}^{\otimes}$. In particular:*

- *The FSO theory of any automatic structure $\mathcal{S}$ is (uniformly) decidable.*
- *If $\mathcal{S}$ is automatic and $\mathcal{S}'$ is FSO-interpretable in $\mathcal{S}'$, then $\mathcal{S}'$ is effectively automatic (i.e., an automatic presentation for $\mathcal{S}'$ can be computed from an automatic presentation for $\mathcal{S}$).*

This paper is mainly interested in the complexity of the following two decision problems:

Definition 6. *For a class $\mathcal{K}$ of automatic presentations, we consider the following sets:*

- *The* isomorphism problem $\mathsf{Iso}(\mathcal{K})$ *for $\mathcal{K}$ is the set of pairs $(\mathcal{P}_1, \mathcal{P}_2) \in \mathcal{K} \times \mathcal{K}$ of automatic presentations with $\mathcal{S}(\mathcal{P}_1) \cong \mathcal{S}(\mathcal{P}_2)$.*
- *The* elementary equivalence problem $\mathsf{EE}(\mathcal{K})$ *for $\mathcal{K}$ is the set of pairs $(\mathcal{P}_1, \mathcal{P}_2) \in \mathcal{K} \times \mathcal{K}$ of automatic presentations such that $\mathcal{S}(\mathcal{P}_1) \equiv \mathcal{S}(\mathcal{P}_2)$.*

If $\mathcal{K}$ is the class of all automatic presentations for a class $\mathcal{C}$ of relational structures (e.g. trees or linear orders), then we will briefly speak of the isomorphism problem for (automatic members of) $\mathcal{C}$. The classes $\mathcal{C}$ that will appear in this paper (equivalence relations and various classes of trees and linear orders) have the nice property that they can be axiomatized by a single FSO-formula. Theorem 5 implies that the corresponding classes $\mathcal{K}$ of automatic presentations are decidable. In this case, since the set of all FO-formulas without free variables can be enumerated, Theorem 5 implies that $\mathsf{EE}(\mathcal{K}) \in \Pi^0_1$. On the other hand, the isomorphism problem can be much harder: The isomorphism problem for the class of all automatic structures is complete for Σ^1_1 [26]. However, if one restricts to special subclasses of automatic structures, this complexity bound can be reduced. For example, for the class of automatic ordinals [27] and also the class of automatic Boolean algebras [26], the isomorphism problem is decidable. Another interesting result is that the isomorphism problem for locally finite automatic graphs is Π^0_3-complete [37].

3 Automatic Equivalence Structures

An equivalence structure is of the form $\mathcal{E} = (D; \approx)$ where $\approx$ is an equivalence relation on D. As usual, we denote with $[a]_{\approx}$ (or briefly $[a]$, if $\approx$ is clear from the context), the equivalence class containing $a \in D$. In this section, we prove that the isomorphism problem for automatic equivalence structures is Π^0_1-complete. This result also follows from our handling of automatic trees in Section 4. However, we present it separately here as it is a good starting point for introducing our techniques.

Let $\mathcal{E} = (D; \approx)$ be a countable equivalence structure. Define the function $h_{\mathcal{E}} : \mathbb{N}_+ \cup \{\aleph_0\} \to \mathbb{N} \cup \{\aleph_0\}$ such that for all $n \in \mathbb{N}_+ \cup \{\aleph_0\}$, $h_{\mathcal{E}}(n)$ equals the number of equivalence classes (possibly infinite) in $\mathcal{E}$ of size n. If $\mathcal{E}$ is automatic, $h_{\mathcal{E}}(n)$ can be computed effectively from n: it equals m if

$$\mathcal{E} \models \exists^{=m} x : (\exists^{=n} y : x \approx y \land \forall y : x \approx y \to x \leq_{\mathsf{llex}} y)$$

(here $\exists^{=\aleph_0}$ stands for $\exists^{=\infty}$). Thus, one can check by Theorem 5 whether $h_{\mathcal{E}}(n) = m$ for $m = \aleph_0, 0, 1, 2, \ldots$ until one finds the correct value. Given two automatic equivalence structures $\mathcal{E}_1$ and $\mathcal{E}_2$, deciding if $\mathcal{E}_1 \cong \mathcal{E}_2$ amounts to checking if $h_{\mathcal{E}_1} = h_{\mathcal{E}_2}$, i.e., if $\forall n \in \mathbb{N}_+ \cup \{\aleph_0\} : h_{\mathcal{E}_1}(n) = h_{\mathcal{E}_2}(n)$. Since the function $h_{\mathcal{E}}$ is computable for automatic equivalence structures, the isomorphism problem for automatic equivalence structures is consequently in Π^0_1.

Consider the automatic equivalence structures $(a^+; =)$ and $((bb)^+; =)$ that are isomorphic. There is no FA recognizable isomorphism, but there is a computable one (mapping, e.g., a^n to b^{2n}). This is no coincidence as the following proposition shows.

Proposition 7. *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two isomorphic automatic equivalence structures. Then there exists a computable isomorphism from $\mathcal{E}_1$ to $\mathcal{E}_2$.*

Proof. Let $\mathcal{E}_i = (V_i; \approx_i)$ (w.l.o.g. $V_1 \cap V_2 = \emptyset$) and let $\leq_{\mathsf{llex}}$ denote the length-lexicographic order on $V_1 \cup V_2$. In the following, $\min(U)$ $(U \subseteq V_i)$ denotes the minimal element of U w.r.t. $\leq_{\mathsf{llex}}$. This minimum exists, since $\leq_{\mathsf{llex}}$ is a well-order. Let $\mathsf{Min}_i = \{u \in V_i \mid u = \min([u])\}$. This set is FSO-definable in the structure $(V_i; \approx_i, \leq_{\mathsf{llex}})$. Since $\leq_{\mathsf{llex}}$ is FA recognizable, this structure is automatic. Hence Min_i is a regular language. It contains a unique element from each equivalence class of $\approx_i$. For $u \in V_i$, define:

$$\begin{aligned} m_1(u) &= |\{v \in V_i \mid v <_{\mathsf{llex}} u \wedge u \approx_i v\}| \in \mathbb{N} \\ m_2(u) &= |\{x \in \mathsf{Min}_i \mid x <_{\mathsf{llex}} \min([u]) \wedge |[x]| = |[u]|\}| \in \mathbb{N} \\ m_3(u) &= |[u]| \in \mathbb{N}_+ \cup \{\aleph_0\} \end{aligned}$$

Thus, $m_1(u)$ is the number of equivalent, but smaller words than u and $m_2(u)$ is the number of equivalence classes of size $|[u]|$ whose minimal element is smaller than the minimal element of $[u]$. Clearly, every isomorphism f between $\mathcal{E}_1$ and $\mathcal{E}_2$ must satisfy $m_3(u) = m_3(f(u))$ for all $u \in V_1$. Moreover, there exists a unique isomorphism f such that $m_1(u) = m_1(f(u))$ and $m_2(u) = m_2(f(u))$ for all $u \in V_1$. Below, we show that the mappings m_i $(1 \leq i \leq 3)$ are computable. This implies that the unique isomorphism f with $m_i(u) = m_i(f(u))$ for all $1 \leq i \leq 3$, $u \in V_1$ is computable as well: For a given $u \in V_1$ we enumerate all $v \in V_2$ until we find a v with $m_i(u) = m_i(v)$ for $1 \leq i \leq 3$. Then, we set $f(u) = v$.

Computability of $m_3(u)$ follows from the fact that $[u]$ is FSO-definable in the automatic structure $(V_i; \approx_i, \leq_{\mathsf{llex}}, \{u\})$. Hence $[u]$ is effectively regular (i.e., an automaton for $[u]$ can be computed from u), and we can compute the cardinality of $[u]$. Let $\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$ be this cardinality. The cardinality of the set $\{v \in V_i \mid v <_{\mathsf{llex}} u \wedge u \approx_i v\}$ (i.e., $m_1(u)$) can be computed by the same argument. Finally, using the cardinality $\kappa = |[u]|$, we can find an FSO-definition for the set $\{x \in \mathsf{Min}_i \mid x <_{\mathsf{llex}} \min([u]) \wedge |[x]| = \kappa\}$ in the structure $(V_i; \approx_i, \leq_{\mathsf{llex}}, \mathsf{Min}_i)$, which is again automatic. Hence the set $\{x \in \mathsf{Min}_i \mid x <_{\mathsf{llex}} \min([u]) \wedge |[x]| = \kappa\}$ is regular as well and we can compute its cardinality (which is $m_2(u)$). □

For the Π^0_1 lower bound, we use a reduction from Hilbert's 10th problem: Given a polynomial $p(x_1, \ldots, x_k) \in \mathbb{Z}[x_1, \ldots, x_k]$, decide whether the equation $p(x_1, \ldots, x_k) = 0$ has a solution in $\mathbb{N}_+$ (for technical reasons, it is useful to exclude 0 in solutions). This problem is well-known to be undecidable, see e.g. [33]. More precisely, let $X \subseteq \mathbb{N}_+$ be some Σ^0_1-complete set. Then, Matiyasevich provides two polynomials $p_1(x, x_1, \ldots, x_k), p_2(x, x_1, \ldots, x_k) \in \mathbb{N}[x, x_1, \ldots, x_k]$ such that for all $n \in \mathbb{N}_+$: $n \in X$ if and only if $\exists y_1, \ldots, y_k \in \mathbb{N}_+ : p_1(n, y_1, \ldots, y_k) - p_2(n, y_1, \ldots, y_k) = 0$, i.e., $p_1(n, y_1, \ldots, y_k) = p_2(n, y_1, \ldots, y_k)$. Hence the mapping $n \mapsto (p_1(n, x_1, \ldots, x_k), p_2(n, x_1, \ldots, x_k))$ is a reduction of X to the set

$$\{(p_1, p_2) \in \mathbb{N}[x_1, \ldots, x_k]^2 \mid k \in \mathbb{N}_+, \exists \overline{c} \in \mathbb{N}_+^k : p_1(\overline{c}) = p_2(\overline{c})\}.$$

Since this set belongs to Σ^0_1, it is therefore Σ^0_1-complete. Hence, the set

$$\{(p_1, p_2) \in \mathbb{N}[x_1, \ldots, x_k]^2 \mid k \in \mathbb{N}_+, \forall \overline{c} \in \mathbb{N}_+^k : p_1(\overline{c}) \neq p_2(\overline{c})\}$$

is Π^0_1-complete.

To reduce this problem to the isomorphism problem of automatic equivalence structures, we define for a non-zero polynomial $p \in \mathbb{N}[x_1, \ldots, x_k]$ a countably infinite equivalence structure $\mathcal{E}(p)$ setting

$$h_{\mathcal{E}(p)}(n) = \begin{cases} \aleph_0 & \text{if } n \in \mathsf{Img}_+(p) \\ 0 & \text{otherwise.} \end{cases}$$

In particular, $\mathcal{E}(p)$ does not have infinite equialence classes. Next consider the polynomial function $C : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ with

$$C(x, y) = (x + y)^2 + 3x + y \tag{2}$$

that is injective ($C(x, y)/2$ is the position of $(x+y, x)$ in the lexicographic enumeration of $\mathbb{N}^2$). Now, let $p_1, p_2 \in \mathbb{N}[x_1, \ldots, x_k]$ (with $k \geq 2$) be two non-zero polynomials and define the polynomials

$$S_1(\overline{x}) = C(p_1(\overline{x}), p_2(\overline{x})),\ S_2(\overline{x}) = C(x_1 + x_2, x_1),\ \text{and } S_3(\overline{x}) = C(x_1, x_1 + x_2)\ . \tag{3}$$

Finally, set

$$\mathcal{E}(p_1, p_2) = \mathcal{E}(S_1) \uplus \mathcal{E}(S_2) \uplus \mathcal{E}(S_3) \text{ and } \mathcal{E}_{\mathsf{Good}} = \mathcal{E}(S_2) \uplus \mathcal{E}(S_3)\,.$$

Lemma 8. *For two non-zero polynomials $p_1, p_2 \in \mathbb{N}[x_1, \ldots, x_k]$ (with $k \geq 2$), we have $\mathcal{E}(p_1, p_2) \cong \mathcal{E}_{\mathsf{Good}}$ if and only if $p_1(\overline{c}) \neq p_2(\overline{c})$ for all $\overline{c} \in \mathbb{N}_+^k$.*

Proof. If $p_1(\overline{c}) = p_2(\overline{c})$ for some $\overline{c} \in \mathbb{N}_+^k$, then there is $y \in \mathbb{N}_+$ such that $C(y, y) \in \mathsf{Img}_+(S_1)$. Therefore in $\mathcal{E}(p_1, p_2)$ there is an equivalence class of size $C(y, y)$, but no such equivalence class exists in $\mathcal{E}_{\mathsf{Good}}$. Hence $\mathcal{E}(p_1, p_2) \ncong \mathcal{E}_{\mathsf{Good}}$.

Conversely, suppose that $p_1(\overline{c}) \neq p_2(\overline{c})$ for all $\overline{c} \in \mathbb{N}_+^k$. For all $y, z \in \mathbb{N}_+$, $\mathcal{E}(p_1, p_2)$ contains an equivalence class of size $C(y, z)$ if and only if $C(y, z)$ belongs to $\mathsf{Img}_+(S_1) \cup \mathsf{Img}_+(S_2) \cup \mathsf{Img}_+(S_3)$, if and only if $y \neq z$, if and only if $\mathcal{E}_{\mathsf{Good}}$ contains an equivalence class of size $C(y, z)$. Therefore, for any $s \in \mathbb{N}_+$, $\mathcal{E}(p_1, p_2)$ contains an equivalence class of size s if and only if $\mathcal{E}_{\mathsf{Good}}$ contains an equivalence class of size s. Since moreover, $\mathcal{E}_{\mathsf{Good}}$ has an equivalence class of size s if and only if it has $\aleph_0$ many equivalence classes of size s, and similarly for $\mathcal{E}(p_1, p_2)$, we get $\mathcal{E} \cong \mathcal{E}_{\mathsf{Good}}$. $\square$

To reduce Hilbert's 10^{th} problem to the isomorphism problem of automatic equivalence structures, it remains to construct an automatic presentation of $\mathcal{E}(p)$ from a non-zero polynomial $p \in \mathbb{N}[x_1, \ldots, x_k]$. More precisely, we will construct a nondeterministic automaton $\mathcal{A}$ such that

$$(\mathsf{Run}(\mathcal{A}); \sim) \cong \mathcal{E}(p)$$

where $r \sim r'$ if and only if $\mathsf{lab}(r) = \mathsf{lab}(r')$, i.e., two runs of $\mathcal{A}$ are equivalent if and only if they accept the same word. The necessary connection between polynomials from $\mathbb{N}[\overline{x}]$ and automata is provided by the following lemma; in its general form, we will need it in Section 4.2.2.

Lemma 9. *There exists an algorithm, whose input consists of the following data:*

- *A sequence $s = (a_1, \ldots, a_k)$ of symbols with $k > 0$ and $a_i \neq a_{i+1}$ for $1 \leq i < k$ (we may have $a_i = a_{i+j}$ for $j \geq 2$).*
- *A non-zero polynomial $p(\overline{x}) \in \mathbb{N}[\overline{x}]$ in k variables.*

On such an input, the algorithm constructs an automaton $\mathcal{A} = \mathcal{A}[s, p]$ with $L(\mathcal{A}) = a_1^+ a_2^+ \cdots a_k^+$ such that for all $x_1, \ldots, x_k \in \mathbb{N}_+$ we have $p(x_1, \ldots, x_k) = |\mathsf{Run}(\mathcal{A}, a_1^{x_1} \cdots a_k^{x_k})|$ (the latter is the number of accepting runs of $\mathcal{A}$ on $a_1^{x_1} \cdots a_k^{x_k}$).

Proof. Let us fix $s = (a_1, \ldots, a_k)$ with $a_i \neq a_{i+1}$ for $1 \leq i < k$. We construct the automaton $\mathcal{A}[s, p]$ by induction on the construction of the polynomial p. The base case is provided by the polynomials 1 and x_j.

For $\mathcal{A}[s, 1]$ we take a deterministic automaton with $L(\mathcal{A}[s, 1]) = a_1^+ a_2^+ \cdots a_k^+$. Next, suppose $p(x_1, \ldots, x_k) = x_j$ for some $j \in \{1, \ldots, k\}$. We define

$$\mathcal{A}[s, x_j] = (\{q_0, q_1, \ldots, q_k, p\}, \Delta, \{q_0\}, \{q_k\}),$$

where

$$\Delta = \{(q_i, a_i, q_i) \mid 1 \leq i \leq m\} \cup \{(q_{i-1}, a_i, q_i) \mid 1 \leq i \leq m\} \cup \{(q_{j-1}, a_j, p), (p, a_j, p), (p, a_j, q_j)\}.$$

When the automaton $\mathcal{A}[s, x_j]$ runs on an input word $a_1^{x_1} \cdots a_k^{x_k}$, it has exactly x_j many times the chance to move to state q_j. Therefore there are exactly x_j many accepting runs on $a_1^{x_1} \cdots a_k^{x_k}$.

Next, let $p_1(x_1, \ldots, x_k)$ and $p_2(x_1, \ldots, x_k)$ be polynomials in $\mathbb{N}[\overline{x}]$. Assume as inductive hypothesis that there is, for $i \in \{1, 2\}$, an automaton $\mathcal{A}[s, p_i] = (S_i, \Delta_i, I_i, F_i)$ such that the number of accepting runs of $\mathcal{A}[s, p_i]$ on $a_1^{x_1} \cdots a_k^{x_k}$ equals $p_i(x_1, \ldots, x_k)$.

For the polynomial $p(\overline{x}) = p_1(x_1, \ldots, x_k) + p_2(x_1, \ldots, x_k)$, let $\mathcal{A}[s, p]$ be the disjoint union of the automata $\mathcal{A}[s, p_1]$ and $\mathcal{A}[s, p_2]$. Then, the number of accepting runs of $\mathcal{A}[s, p]$ on $a_1^{x_1} \cdots a_k^{x_k}$ is $p_1(x_1, \ldots, x_k) + p_2(x_1, \ldots, x_k)$.

For the polynomial $p(x_1, \ldots, x_k) = p_1(x_1, \ldots, x_k) \cdot p_2(x_1, \ldots, x_k)$, consider the Cartesian product $\mathcal{A}[s, p] = \mathcal{A}[s, p_1] \times \mathcal{A}[s, p_2]$. This Cartesian product is the automaton $(S_1 \times S_2, \Delta, I_1 \times I_2, F_1 \times F_2)$, where

$$\Delta = \{((p_1, p_2), \sigma, (q_1, q_2)) \mid (p_1, \sigma, q_1) \in \Delta_1, (p_2, \sigma, q_2) \in \Delta_2\}.$$

Then, $\mathsf{Run}(\mathcal{A}[s, f, p], a_1^{x_1} \cdots a_k^{x_k}) = p_1(x_1, \ldots, x_k) \cdot p_2(x_1, \ldots, x_k)$. ☐

Lemma 10. *From two non-zero polynomials $p_1, p_2 \in \mathbb{N}[x_1, \ldots, x_k]$ with $k \geq 2$, one can construct a finite automaton $\mathcal{A}$ such that*

$$\mathcal{E}(p_1, p_2) \cong (\mathsf{Run}(\mathcal{A}); \sim).$$

Proof. Define the polynomials S_1, S_2, and S_3 from (3). We consider S_1, S_2, and S_3 as polynomials in the variables $x_1, \ldots, x_k, x_{k+1}$. Then, using Lemma 9, we can construct a finite automaton $\mathcal{A}$ with $L(\mathcal{A}) = \{1, 2, 3\}a_1^+ a_2^+ \cdots a_{k+1}^+$ such that the number of accepting runs of $\mathcal{A}$ on $d a_1^{x_1} a_2^{x_2} \cdots a_{k+1}^{x_{k+1}}$ equals $S_d(x_1, \ldots, x_{k+1})$. For simplicity, we write $\mathcal{E}$ for the automatic equivalence structure $(\mathsf{Run}(\mathcal{A}); \sim)$. Since $S_d(x_1, \ldots, x_k, x_{k+1}) = S_d(x_1, \ldots, x_k, x_{k+1} + 1)$, we have $h_{\mathcal{E}}(n) > 0$ if and only if $h_{\mathcal{E}}(n) = \aleph_0$ for all $n \in \mathbb{N}_+ \cup \{\aleph_0\}$.

Furthermore, $h_{\mathcal{E}}(n) > 0$ (or, equivalently, $h_{\mathcal{E}}(n) = \aleph_0$) if and only if there exists $w \in L(\mathcal{A})$ that is accepted by precisely n distinct runs. By the construction, this is the case if and only if $n \in \mathsf{Img}_+(S_1) \cup \mathsf{Img}_+(S_2) \cup \mathsf{Img}_+(S_3)$. By the definition of $\mathcal{E}(p_1, p_2)$, this is equivalent to saying $h_{\mathcal{E}(p_1, p_2)}(n) = \aleph_0$. ☐

Lemma 8 and 10 immediately imply that the isomorphism problem for automatic equivalence structures is Π_1^0-complete. Below, we prove a slight strengthening of this. An automatic equivalence structure consists of two parts: a regular set and an FA recognizable equivalence relation. One might think that it is the FA recognizable equivalence relation, which makes the isomorphism problem difficult. But we will prove that actually the regular domain is responsible for undecidablity. More precisely, we can fix the FA recognizable equivalence relation and only put an automaton for the domain into the input and still obtain an undecidable isomorphism problem.

In the following, let Σ_2 denote the alphabet $\{0, 1\} \times \{0, 1\}$. Note that words over Σ_2 are convolutions of two words over $\{0, 1\}$ of the same length. We set $u \otimes v \approx u' \otimes v'$ if and only if $u = u'$. This defines an automatic equivalence structure $(\Sigma_2^*; \approx)$. Note that all equivalence classes of $(\Sigma_2^*; \approx)$ are finite. In the following, we will consider restrictions of this equivalence structure to regular sets $L \subseteq \Sigma_2^*$. To simplify notation, we write $(L; \approx)$ for $(L; \approx \cap L^2)$.

Proposition 11. *The set of finite automata $\mathcal{B}$ with $(L(\mathcal{B}); \approx) \cong \mathcal{E}_{\mathsf{Good}}$ is hard for Π_1^0. It equals the set of finite automata $\mathcal{B}$ with $(L(\mathcal{B}); \approx) \equiv \mathcal{E}_{\mathsf{Good}}$.*

Proof. Let $p_1, p_2 \in \mathbb{N}[x_1, \ldots, x_k]$ be non-zero polynomials and let $\mathcal{A}$ be the automaton from Lemma 10 such that $(\mathsf{Run}(\mathcal{A}); \sim) \cong \mathcal{E}(p_1, p_2)$ and therefore by Lemma 8

$$(\mathsf{Run}(\mathcal{A}); \sim) \cong \mathcal{E}_{\mathsf{Good}} \iff \forall \overline{c} \in \mathbb{N}_+^k : p_1(\overline{c}) \neq p_2(\overline{c}).$$

Let $\{b_1, b_2, \ldots, b_m\}$ be the alphabet of $\mathcal{A}$ and let $\Delta = \{t_1, \ldots, t_n\}$ denote the set of transitions of $\mathcal{A}$, where w.l.o.g. $m \leq n$. If $t_i = (p, b_j, q)$, then set $f(t_i) = 0^j 1^{n-j} \otimes 0^i 1^{n-i}$ and extend f to a homomorphism from Δ^* to Σ_2^*. Note that f is injective. For two runs $r, r' \in \mathsf{Run}(\mathcal{A})$, we have $r \sim r'$ if and only if $\mathsf{lab}(r) = \mathsf{lab}(r')$ if and only if $f(r) \approx f(r')$. Hence $(L; \approx) \cong (\mathsf{Run}(\mathcal{A}); \sim)$ with $L = f(\mathsf{Run}(\mathcal{A}))$. Since, from $\mathcal{A}$, we can construct a nondeterministic automaton accepting L, the first claim follows from the Π_1^0-hardness of the set of pairs of polynomials p_1, p_2 over $\mathbb{N}$ with $p_1(\overline{c}) \neq p_2(\overline{c})$ for all $\overline{c} \in \mathbb{N}^k$.

The second statement follows since all equivalence classes of $(L; \approx)$ and $\mathcal{E}_{\mathsf{Good}}$ are finite. This implies that $(L; \approx) \equiv \mathcal{E}_{\mathsf{Good}}$ if and only if they are isomorphic. □

Theorem 12. *The isomorphism problem and the elementary equivalence problem for automatic equivalence structures are Π_1^0-complete.*

Proof. At the beginning of this section, we already argued that the isomorphism problem is in Π_1^0; hardness follows immediately from Proposition 11, since $\mathcal{E}_{\mathsf{Good}}$ is necessarily automatic.

Note that the set of automatic presentations of equivalence structures is decidable. Hence the elementary equivalence problem belongs to Π_1^0 as explained after Definition 1. Hardness follows from the second statement of Proposition 11. □

4 Automatic Trees

A *forest* is a structure $H = (V; \leq)$, where $\leq$ is a partial order on V such that for every $x \in V$, the order $\leq$ restricted to the set $\{y \mid y \leq x\}$ of ancestors of x is a finite linear order. A *tree* is a forest with least element, called *root*, or the empty structure. A maximal element of the forest H is also called a *leaf* of H. Let us fix a forest $H = (V; \leq)$. For a node $v \in V$ we denote with $H(u)$ (the *subtree of* H *rooted at* u) the forest H restricted to the set $\{v \in V \mid u \leq v\}$; this is indeed a tree whose root is u. The *level* of a node $v \in V$ is $|\{x \mid x < v\}| \in \mathbb{N}$. The *height* of H is the supremum of the levels of all nodes in V; it may be infinite even in case H is well-founded. One may also view H as a directed graph $(V; E)$, where there is an edge $(u, v) \in E$ if and only if u is the largest element in $\{x \mid x < v\}$. We call E the *edge relation* of H. The edge relation E is FO-definable in $(V; \leq)$: $(u, v) \in E$ if and only if $u < v$ and $\neg\exists x : u < x < v$. The set of *children* of $v \in V$ is denoted by $E(v) = \{u \in V \mid (v, u) \in E\}$. The forest $H = (V; \leq)$ is *well-founded*, if there does not exist an *infinite branch* in H, which is an infinite sequence of nodes $v_0 < v_1 < v_2 < \cdots$. An example of a well-founded tree is $(X^{<k}; \preceq)$ where $X^{<k}$ denotes the set of all words over the set X of length at most $k-1$. We will denote this tree briefly by $X^{<k}$. We also write $X^{<\omega}$ for the tree whose universe is X^* with the prefix relation. A forest $H_1 = (V_1; \leq_1)$ *embeds* into a forest $H_2 = (V_2; \leq_2)$, briefly $H_1 \hookrightarrow H_2$, if there exists an injective mapping $f : V_1 \to V_2$ such that for all $u, v \in V_1$, $u \leq_1 v$ if and only if $f(u) \leq_2 f(v)$.

Let us emphasize again that in this paper, forests and in particular trees are partial orders. In the literature, a tree viewed as a partial order is also called an *order trees*, whereas the graph consisting of the edge relation of an order tree is also called a *successor tree.* Hence, when talking about trees, we implicitly speak of order trees.

Note that for an automatic forest H and a node $v \in H$ the subtree $H(v)$ is FO-definable in H (using v as a parameter). Hence, the domain of $H(v)$ is a regular subset of the domain of H and $H(v)$ is effectively automatic (i.e., an automatic presentation for $H(v)$ can be computed from an automatic presentation for H and v) as well.

We use $\mathcal{T}_n$ ($n \in \mathbb{N}$) to denote the class of all automatic presentations $\mathcal{P}$ such that $\mathcal{S}(\mathcal{P})$ is an automatic tree of height at most n. Note that one can write down a sentence of FSO that is satisfied by a directed graph G if and only if G is a forest. Hence the set of all automatic presentations of forests is decidable by Theorem 5. The same argument shows decidability of the set of all automatic presentations of trees, of well-founded trees[4], and of trees of height at most n. The same holds for the class of trees of finite height (a generalization of this theorem in the spirit of Theorem 5 can be found in [28]):

[4] This latter result was first shown in [27]. Note that a tree $(V; \leq)$ is well-founded if and only if the undirected graph $(V; \leq \cup \geq)$ does not contain an infinite clique. Hence, one can use the FSO-formula from Example 2.

Theorem 13. *The set of automatic presentations of trees of finite height is decidable.*

Proof. Let $\mathcal{P}$ be an automatic presentation of some tree and let $\mathcal{S}(\mathcal{P}) = (V; \leq)$. Then the relation $\geq$ is rational[5] and a rational transducer $\mathcal{A}$ for $\geq$ can be computed from $\mathcal{P}$. The tree $\mathcal{S}(\mathcal{P})$ has finite height if and only if this transducer is finite-valued, i.e., there exists $n \in \mathbb{N}$ such that $|\{v \in V \mid u \geq v\}| \leq n$ for all words $u \in V$. But this is decidable by [43]. □

4.1 Upper bounds for trees with countably many infinite branches

We will show that the isomorphism problem for automatic trees with countably many infinite branches can be reduced to true arithmetic. Towards this aim, we will parameterize these trees T by their *embeddability rank* $\mathsf{erank}(T)$ (which is defined in Section 4.1.1) and show in Section 4.1.2 the arithmetical upper bound Π^0_{2k-4} for trees of embeddability rank at most k. The claim then follows from the uniformity of our proof and the computability of $\mathsf{erank}(T)$.

4.1.1 The embeddability rank of a tree. Let T be a tree. Its *embeddability rank* or *e-rank* $\mathsf{erank}(T)$ is defined to be

$$\mathsf{erank}(T) = \sup\{k+1 \mid k \in \mathbb{N} \cup \{\omega\}, \mathbb{N}^{<k} \hookrightarrow T\} \in \mathbb{N}_+ \cup \{\omega, \omega+1\}\,.$$

Then the empty tree has e-rank 1 (since only the empty tree $\mathbb{N}^{<0}$ can be embedded into it), any finite and non-empty tree has e-rank 2, the disjoint union of all trees $\mathbb{N}^{<k}$ for $k \in \mathbb{N}$ together with a new root has e-rank ω, and $\mathbb{N}^{<\omega}$ has e-rank $\omega + 1$. A nonempty tree has e-rank 2 if and only if it does not contain an infinite antichain, i.e., if and only if it is finitely branching and has only finitely many branching points, i.e., nodes with at least two children.

By $\mathcal{T}^{\text{er}}_k$, we denote the set of all automatic presentations of trees of e-rank at most $k \in \mathbb{N}_+ \cup \{\omega, \omega+1\}$. Then we have obviously

$$\mathcal{T}^{\text{er}}_1 \subsetneq \mathcal{T}^{\text{er}}_2 \subsetneq \mathcal{T}^{\text{er}}_3 \subsetneq \cdots \subsetneq \bigcup_{i \geq 1} \mathcal{T}^{\text{er}}_i \subseteq \mathcal{T}^{\text{er}}_\omega \subsetneq \mathcal{T}^{\text{er}}_{\omega+1} \tag{4}$$

(strictness in the last inclusion holds since the automatic tree $\mathbb{N}^{<\omega}$ has e-rank $\omega + 1$). The aim of this section is to prove that an automatic tree has only countably many infinite branches if and only if its e-rank is finite. For this, we first prove that the e-rank of a tree is $\omega + 1$ if and only if it has uncountably many infinite branches (Lemma 14) and then, that no automatic tree has e-rank ω (Lemma 17). This latter result implies in particular $\bigcup_{i\geq 1} \mathcal{T}^{\text{er}}_i = \mathcal{T}^{\text{er}}_\omega$. Thus, the inclusion chain (4) can be simplified to

$$\mathcal{T}^{\text{er}}_1 \subsetneq \mathcal{T}^{\text{er}}_2 \subsetneq \mathcal{T}^{\text{er}}_3 \subsetneq \cdots \subsetneq \bigcup_{i \geq 1} \mathcal{T}^{\text{er}}_i = \mathcal{T}^{\text{er}}_\omega \subsetneq \mathcal{T}^{\text{er}}_{\omega+1}\,.$$

We will also show that all the classes $\mathcal{T}^{\text{er}}_k$ ($k \in \mathbb{N}_+ \cup \{\omega, \omega+1\}$) are decidable.

Lemma 14. *Let $T = (L; \leq)$ be a countable tree. The following are equivalent:*

(1) $\mathsf{erank}(T) = \omega + 1$
(2) T has $2^{\aleph_0}$ many infinite branches.
(3) T has uncountably many infinite branches.

Proof. First suppose that $\mathsf{erank}(T) = \omega + 1$. Then the tree $\mathbb{N}^{<\omega}$ embeds into T, hence T has $2^{\aleph_0}$ many infinite branches which proves the implication (1)⇒(2). The implication (2)⇒(3) is trivial.

To prove the remaining implication (3)⇒(1), suppose that T has uncountably many infinite branches. For a node $x \in L$, let $\text{br}(x)$ denote the number of infinite branches of $T(x)$. Let $B =$

[5] A relation $R \subseteq \Gamma^* \times \Gamma^*$ is rational, if it can be accepted by a rational transducer, which is a finite automaton with transition labels from $\Gamma^* \times \Gamma^*$.

$\{x \in L \mid \mathrm{br}(x) > \aleph_0\}$. We first show that B contains two incomparable nodes (w.r.t. the tree order $\leq$). Suppose towards a contradiction that B is linearly ordered. Let X denote the set of nodes $y \in L \setminus B$ whose immediate predecessor (i.e., parent node) belongs to B. Since L is countable, so is X. Hence the number of infinite branches of T equals

$$1 + \sum_{y \in X} \mathrm{br}(y) \leq 1 + \aleph_0 \cdot \aleph_0 = \aleph_0$$

contradicting our assumption that T contains uncountably many many infinite branches. By induction, it follows that the complete binary tree $\{0,1\}^{<\omega}$ can be embedded into T. But then we have $\mathbb{N}^{<\omega} \hookrightarrow \{0,1\}^{<\omega} \hookrightarrow T$, which is statement (1). □

By the following result, the properties from Lemma 14 are decidable.

Proposition 15. *The set of automatic presentations of trees with only countably many infinite branches is decidable.*

Proof. Let $T = (L; \leq)$ be an automatic tree. Let $B \subseteq 2^L$ be the set of its infinite branches, and let in be the set of pairs $(x, a) \in L \times B$ with $x \in a$. In [30], it was shown that the structure $(L \cup B; \leq, B, \mathsf{in})$ is effectively ω-automatic. Hence, by [3], its $(\mathsf{FO} + \exists^{2^{\aleph_0}})$-theory[6] is decidable. In this logic, it is expressible that the set B has size $2^{\aleph_0}$, which means that T has $2^{\aleph_0}$ many infinite branches. By Lemma 14, this is equivalent to the fact that T has uncountably many infinite branches. □

Our next aim is to show that there is no automatic tree of e-rank ω. As a first step, we show that no well-founded automatic tree has e-rank ω.

Lemma 16. *Let $T = (L; \leq)$ be an automatic well-founded tree. Then $\mathsf{erank}(T)$ is finite.*

Proof. Let us fix an arbitrary length-lexicographic order $\leq_{\mathsf{llex}}$ on the set of words L. We define the *Kleene-Brouwer order* $\mathsf{KB}(T) = (L; \sqsubseteq)$, where $u \sqsubset v$ if and only if $v \leq u$ or there exist $w, u_1, v_1 \in L$ such that $u_1 \leq u$, $v_1 \leq v$, w is the parent node of u_1 and of v_1 in T, and $u_1 <_{\mathsf{llex}} v_1$. This order is linear. Moreover, since T is well-founded, $\mathsf{KB}(T)$ is an ordinal.

Note that the expansion of T by $\leq_{\mathsf{llex}}$ is still an automatic structure and that $\sqsubseteq$ is first-order definable in this structure. Hence $\mathsf{KB}(T)$ is an automatic ordinal. Thus, by [11], there exists $k \in \mathbb{N}$ with $\mathsf{KB}(T) < \omega^k$.

By induction on i, we now show $\mathsf{KB}(T(x)) \geq \omega^i$ in case $\mathbb{N}^{<i+1} \hookrightarrow T(x)$ for all nodes x. If $\mathbb{N}^{<2} \hookrightarrow T$ then T is infinite and we get $\mathsf{KB}(T) \geq \omega$. Now assume that $i \geq 2$ and that $\mathbb{N}^{<i+1} \hookrightarrow T$. Then there exists an infinite antichain $\{a_0, a_1, a_2, \ldots\}$ in T such that $\mathbb{N}^{<i} \hookrightarrow T(a_j)$ for all $j \geq 0$. W.l.o.g. assume that $a_0 \sqsubset a_1 \sqsubset a_2 \sqsubset \cdots$. By induction, we have $\mathsf{KB}(T(a_j)) \geq \omega^{i-1}$ for all $j \geq 0$ and therefore

$$\mathsf{KB}(T) \geq \sum_{j \in \mathbb{N}} \mathsf{KB}(T(a_j)) \geq \omega^{i-1} \cdot \omega = \omega^i \,.$$

This finishes the induction.

Since $\mathsf{KB}(T) < \omega^k$, we therefore get $\mathbb{N}^{<k+1} \not\hookrightarrow T$, i.e., the e-rank of T is at most $k+1$ and therefore finite. □

By Lemma 16, no well-founded automatic tree has e-rank ω. To prove this fact for all automatic trees, we will use the notion of Cantor-Bendixon-rank of a tree $T = (L; \leq)$: Let $d(T)$ denote the restriction of T to those nodes that belong to at least two infinite branches of T. This is again a countable tree (possibly empty). By [27], there exists a number $r \in \mathbb{N}$ with $d^r(T) = d^{r+1}(T)$. The least such number is called the *Cantor-Bendixon-rank*. Note that it is very different from the e-rank we defined above: the tree $\mathbb{N}^{<\omega}$ has Cantor-Bendixon-rank 0 and e-rank $\omega + 1$. The following lemma generalizes Lemma 16.

[6] $\mathsf{FO} + \exists^{2^{\aleph_0}}$ is the extension of FO by the quantifier $\exists^{2^{\aleph_0}}$, which expresses that there are $2^{\aleph_0}$ many elements with a given property.

Lemma 17. *Let $T = (L; \leq)$ be an automatic tree with countably many infinite branches. Then* $\mathsf{erank}(T)$ *is finite.*

Proof. The lemma is shown by induction on the Cantor-Bendixon-rank of T. If this rank equals 0, then every node of T belongs to at least two infinite branches, so T is either empty or embeds $\{0,1\}^{<\omega}$. Since T has only countably many infinite branches, we get $T = \emptyset$. Hence, $\mathsf{erank}(T) = 1$.

Now suppose that the Cantor-Bendixon-rank of $T = (L; \leq)$ is $r+1$. We split L into three sets: L_0 contains all nodes that do not belong to any infinite branch, L_1 consists of those nodes that belong to precisely one infinite branch, and L_2 is the rest (i.e., $(L_2; \leq) \cong d(T)$). The sets L_1 and L_2 (and therefore L_0) are effectively regular [26]. Let T_0 be obtained from the forest $(L_0; \leq)$ by adding a new root. Then T_0 is a well-founded automatic tree that has finite e-rank e_0 by Lemma 16. Also $d(T)$ is an automatic tree with at most $\aleph_0$ many infinite branches. Since its Cantor-Bendixon-rank is properly smaller than that of T, the induction hypothesis guarantees that its e-rank e_2 is finite.

We want to show that the e-rank of T is at most $e_2 + e_0 + 1$. So let $k \in \mathbb{N}_+$ and let $f : \mathbb{N}^{<k} \hookrightarrow T$ be an embedding. We have to prove $k \leq e_2 + e_0$. If the image of f is contained in L_2, then f is an embedding into $d(T)$ implying $k < e_2 \leq e_2 + e_0$. Otherwise let $w_2 \in \mathbb{N}^{<k}$ be a word of minimal length with $f(w_2) \notin L_2$. Then all words of length $< |w_2|$ are mapped into L_2, i.e., the restriction of f to $\mathbb{N}^{<|w_2|}$ is an embedding into $d(L)$ which implies $|w_2| < e_2$. We now distinguish two cases.

(a) Suppose $f(w_2) \in L_0$. Then the mapping $g : \mathbb{N}^{<k-|w_2|} \to T_0$ with $g(x) = f(w_2 x)$ is an embedding implying $k - |w_2| < e_0$ and therefore $k < e_2 + e_0$.

(b) Now suppose $f(w_2) \in L_1$. If $|w_2| = k-1$, then $|w_2| < e_2$ implies $k \leq e_2 \leq e_2 + e_0$. So let $|w_2| < k-1$. Since $(L_1; \leq)$ is a disjoint union of copies of ω, there is some $n \in \mathbb{N}$ with $f(w_2 n) \in L_0$. As in (a), we obtain $k - |w_2 n| < e_0$ which, together with $|w_2 n| = |w_2| + 1 \leq e_2$ implies $k < e_2 + e_0$. □

Corollary 18. *An automatic tree T has countably many infinite branches if and only if* $\mathsf{erank}(T)$ *is finite.*

Proof. If T has countably many infinite branches, then $\mathsf{erank}(T)$ is finite by Lemma 17. If $\mathsf{erank}(T)$ is finite, then it is not $\omega+1$ and so T has only countably many infinite branches by Lemma 14. □

We finish our consideration of the e-rank proving that it can be computed. Consider the following recursively defined formulas of FSO for $k \in \mathbb{N}_+$:

$$\begin{aligned}
\mathsf{rank}_1(x) &= (x = x) \\
\mathsf{rank}_{k+1}(x) &= \exists X \text{ infinite } \forall y, y' \in X : x < y \wedge (y \leq y' \rightarrow y = y') \wedge \mathsf{rank}_k(y)\,.
\end{aligned}$$

By induction on k, one can show:

Lemma 19. *Let T be some tree, v a node of T, and $k \in \mathbb{N}$. Then $T \models \mathsf{rank}_{k+1}(v)$ if and only if $\mathbb{N}^{<k} \hookrightarrow T(v)$ (i.e., $\mathsf{erank}(T(v)) > k$).*

Corollary 20. *For a given automatic tree, one can compute its e-rank.*

Proof. Let $T = (L; \leq)$ be an automatic tree. First, we check whether $\mathsf{erank}(T) = \omega + 1$: By Lemma 14, we have to check whether T contains $2^{\aleph_0}$ many infinite branches, which is decidable by Proposition 15.

Next, assume that it turns out that $\mathsf{erank}(T) < \omega + 1$. Thus, by Lemma 18, $\mathsf{erank}(T)$ is finite. Then, for every $k \in \mathbb{N}$, $\mathsf{erank}(T) \leq k$ if and only if $T \models \neg\exists x : \mathsf{rank}_{k+1}(x)$. Since this is a sentence of FSO that can be computed from k, we can check effectively, whether $\mathsf{erank}(T) \leq k$. By doing this successively for $k = 1, 2, \ldots$, we can compute $\mathsf{erank}(T)$. □

4.1.2 The isomorphism problem. Note that the empty tree is the only tree of e-rank 1. The following definition will be used in our proof of an upper bound for the isomorphism problem for automatic trees of higher e-rank.

Definition 21. *Let $T = (L; \leq)$ be some tree of e-rank $k \geq 2$. Then the* initial segment $I(T) \subseteq L$ *consists of all nodes $x \in L$ with $\mathsf{erank}(T(x)) = k$.*

Note that the root of T always belongs to the initial segment $I(T)$ and that $x \leq y \in I(T)$ implies $x \in I(T)$.

Lemma 22. *Let $T = (V; \leq)$ be some tree with $\mathsf{erank}(T) = k \geq 2$. Then $(I(T); \leq)$ is a tree of e-rank 2.*

Proof. Since $I(T)$ is downwards closed in T, $(I(T); \leq)$ is indeed a tree itself. Since $I(T) \neq \emptyset$ by the above remark, we get $\mathsf{erank}(I(T); \leq) \geq 2$. If $\mathsf{erank}(I(T); \leq) \geq 3$, then $\mathbb{N}^{<2}$ would embed into $(I(T); \leq)$, i.e., $I(T)$ would contain an infinite antichain $B = \{b_i \mid i \in \mathbb{N}\}$. Since the e-rank of $T(b_i)$ is k, we get $\mathbb{N}^{<k-1} \hookrightarrow T(b_i)$ implying $\mathbb{N}^{<k} \hookrightarrow T$. But this contradicts $\mathsf{erank}(T) = k$. ☐

We now study the isomorphism problem $\mathsf{Iso}(\mathcal{T}_2^{\mathrm{er}})$ of automatic trees of e-rank at most 2. Recall that a nonempty tree has e-rank 2 if and only if it is finitely branching and has only finitely many branching points. But this is the case if and only if it is a finite tree where some of the leaves are replaced by infinite branches. In particular, there are only finitely many isomorphisms between two trees of e-rank 2. But these isomorphisms need not be automatic in case the two trees are automatic (e.g., consider the automatic trees $\{a\}^{<\omega}$ and $\{aa\}^{<\omega}$ that are both isomorphic to ω).

Lemma 23. *The following holds:*

(1) Any isomorphism between two automatic trees of e-rank 2 is computable.
(2) From two automatic presentations of trees of e-rank 2, one can compute a list of all (indices of) isomorphisms.
(3) The isomorphism problem $\mathsf{Iso}(\mathcal{T}_2^{\mathrm{er}})$ of automatic trees of e-rank at most 2 is decidable.

Proof. Clearly, (2) implies (3). For (1) and (2), let $\mathcal{P}_1, \mathcal{P}_2 \in \mathcal{T}_2^{\mathrm{er}}$ and let $T_i = \mathcal{S}(\mathcal{P}_i) = (L_i; \leq_i)$. Let $B_i \subseteq L_i$ be the set of nodes $x \in L_i$ such that there exists a leaf y or a branching point y in T_i with $x \leq_i y$. Let $C_i \subseteq L_i$ be the union of B_i and all children of nodes in B_i. Clearly, C_i is downwards closed in T_i, i.e., $(C_i, \leq_i)$ is a tree. Since T_i has no infinite antichains (otherwise, it would embed $\mathbb{N}^{<2}$ and therefore have e-rank at least 3), the sets C_1 and C_2 are finite and computable from $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively.

Any isomorphism $f : T_1 \to T_2$ induces an isomorphism $g : (C_1; \leq_1) \to (C_2; \leq_2)$ with $g(B_1) = B_2$. Note that the nodes from $C_i \setminus B_i$ are the starting points of non-branching infinite branches of T_i. Hence, conversely, any isomorphism $g : (C_1; \leq_1) \to (C_2; \leq_2)$ with $g(B_1) = B_2$ extends uniquely to an isomorphism $f : T_1 \to T_2$. Given the finite set g, the isomorphism f is even computable: for $x \in C_1$, output $g(x)$; for $x \in L_1 \setminus C_1$, compute the unique node $y \in C_1 \setminus B_1$ with $y <_1 x$, compute the distance in T_1 from y to x, and map x to the unique node of the same distance in T_2 from $g(y)$.

A list of all indices of isomorphisms from T_1 to T_2 can be computed by listing all isomorphisms between the finite trees $(C_1; \leq_1)$ and $(C_2; \leq_2)$ that map B_1 to B_2. By the above argument we can compute from an isomorphism $g : (C_1; \leq_1) \to (C_2; \leq_2)$ with $g(B_1) = B_2$ an index for the unique isomorphism $f : T_1 \to T_2$ that extends g. This shows (2). ☐

Lemma 24. *From an automatic presentation $\mathcal{P} \in \mathcal{T}_2^{\mathrm{er}}$ of a tree of e-rank at most 2, one can compute a first-order sentence $\varphi_{\mathcal{P}}$ such that, for all trees T, we have*

$$T \models \varphi_{\mathcal{P}} \iff T \cong \mathcal{S}(\mathcal{P})\,.$$

Proof. Recall that a tree has e-rank at most 2 if and only if it is a finite tree where some of the leaves are replaced by infinite branches. Hence any tree of e-rank at most 2 can be described in first-order logic up to isomorphism. To find $\varphi_{\mathcal{P}}$, simply list all sentences that describe trees of e-rank at most 2 and output the first from this list that holds in $\mathcal{S}(\mathcal{P})$. ☐

Lemma 25. *For $3 \leq n < \omega$, the isomorphism problem $\mathsf{Iso}(\mathcal{T}_n^{\text{er}})$ for automatic trees of e-rank at most n belongs to Π^0_{2n-5}.*

Proof. The proof proceeds by induction on n. So let $n \geq 3$, and $\mathcal{P}_1, \mathcal{P}_2 \in \mathcal{T}_n^{\text{er}}$, $T_i = \mathcal{S}(\mathcal{P}_i) = (L_i; \leq_i)$. Define the automatic forest $H = T_1 \uplus T_2$ and let E be the edge relation of H (it is again automatic). Moreover, let $I = I(T_1) \uplus I(T_2)$.

For $x \in H$ and a tree t, let $\#(x,t) \in \mathbb{N} \cup \{\aleph_0\}$ denote the number of children $y \in E(x)$ of x in H such that $H(y) \cong t$. Given this definition, we have $T_1 \cong T_2$ if and only if there exists an isomorphism $f : (I(T_1), \leq_1) \to (I(T_2), \leq_2)$ such that for all $x \in I(T_1)$:

$$\forall \text{ trees } t \text{ with } \mathsf{erank}(t) \leq n-1 : \#(x,t) = \#(f(x),t)\,.$$

If t is not automatic, then $\#(x,t) = \#(f(x),t) = 0$, i.e., we can restrict quantification to automatic trees of e-rank at most $n-1$. Hence, $T_1 \cong T_2$ if and only if one of the isomorphisms $f : (I(T_1); \leq_1) \to (I(T_2); \leq_2)$ satisfies the following:

$$\forall x \in I(T_1)\ \forall \mathcal{P} \in \mathcal{T}_{n-1}^{\text{er}}\ \forall \ell \geq 1 \left(\begin{array}{rl} & \exists^{\geq \ell} x \in E(x) \setminus I : \quad \mathcal{S}(\mathcal{P}) \cong H(x) \\ \iff & \exists^{\geq \ell} x \in E(f(x)) \setminus I : \mathcal{S}(\mathcal{P}) \cong H(x) \end{array} \right) \tag{5}$$

Recall that by Lemma 22 and 23, a finite list of all isomorphisms $f : (I(T_1), \leq_1) \to (I(T_2), \leq_2)$ (each of these isomorphisms is computable) can be computed. Hence, it suffices to show that the formula (5) is a Π^0_{2n-5}-statement. Note that we have $\mathsf{erank}(\mathcal{S}(\mathcal{P})) < n$ and $\mathsf{erank}(H(x)) < n$ for all $\mathcal{P} \in \mathcal{T}_{n-1}^{\text{er}}$ and for all nodes $x \in (E(x) \setminus I) \cup (E(f(x)) \setminus I)$. We now distinguish the cases $n = 3$ and $n > 3$:

- Let $n = 3$. Then the subformula $\mathcal{S}(\mathcal{P}) \cong H(x)$ in (5) is equivalent to $H(x) \models \varphi_{\mathcal{P}}$, where $\varphi_{\mathcal{P}}$ is the FSO-formula from Lemma 24. By Lemma 19, the set I is FSO-definable and therefore effectively regular. Given $x \in I(T_1)$, the set $E(x)$ is also effectively regular. Since the isomorphism f is computable, the set $E(f(x))$ is effectively regular as well. Hence the equivalence in brackets is an FSO-statement about an automatic structure and therefore decidable. Thus, the whole formula belongs to $\Pi^0_1 = \Pi^0_{2n-5}$.
- Now let $n > 3$. Hence the subformula $\mathcal{S}(\mathcal{P}) \cong H(x)$ is by the induction hypothesis a Π^0_{2n-7} statement. As for the case $n = 3$, one can argue that the sets I, $E(x)$, and $E(f(x))$ are effectively regular. Thus, the whole formula belongs to Π^0_{2n-5}. □

Proposition 26. *The isomorphism problem $\mathsf{Iso}(\mathcal{T}_\omega^{\text{er}})$ of automatic trees with countably many infinite branches is many-one reducible to $\mathsf{FOTh}(\mathbb{N}; +, \times)$.*

Proof. Let $P_1, P_2 \in \mathcal{T}_\omega^{\text{er}}$ be automatic presentations of trees T_1 and T_2 with countably many branches. Then there are $k_1, k_2 \in \mathbb{N}$ such that $\mathsf{erank}(T_i) = k_i$. By Corollary 20, these natural numbers can be computed. If $k_1 \neq k_2$, then the two trees are not isomorphic. Otherwise, they are isomorphic if and only if (P_1, P_2) belongs to the Π_{2k_1-4}-relation Iso_{k_1}. The uniformity of the proof of Lemma 25 implies the result. □

Note that a tree of height n has e-rank at most $n+2$. Hence, we have $\mathcal{T}_n \subseteq \mathcal{T}_{n+2}^{\text{er}}$ for all $n \geq 0$ (recall that $\mathcal{T}_n$ is the class of all automatic presentations of trees of height at most n). Lemma 25 implies that the isomorphism problem $\mathsf{Iso}(\mathcal{T}_n)$ for automatic trees of height at most n belongs to $\Pi^0_{2(n+2)-5} = \Pi^0_{2n-1}$ for all $n \geq 1$. We can improve this upper bound by two levels:

Lemma 27. *The isomorphism problem for the class $\mathcal{T}_n$ of automatic trees of height at most n is*

- *decidable for $n = 1$ and*
- *in Π^0_{2n-3} for all $n \geq 2$.*

Proof. For the first part of the lemma, take $\mathcal{P}_1, \mathcal{P}_2 \in \mathcal{T}_1$. To check if $\mathcal{S}(\mathcal{P}_1) \cong \mathcal{S}(\mathcal{P}_2)$, it suffices to compute the cardinality of the two trees, which can be done as their universes are regular languages.

We show the second part of the lemma (i.e., where $n \geq 2$) by induction on n. Consider automatic presentations $\mathcal{P}_1, \mathcal{P}_2 \in \mathcal{T}_n$. Define the automatic forest $H = \mathcal{S}(\mathcal{P}_1) \uplus \mathcal{S}(\mathcal{P}_2)$ and let E be the edge relation of H (it is again automatic). Let r_i be the root of T_i.

For $n = 2$, the trees T_1 and T_2 have height at most 2. Then $T_1 \cong T_2$ if and only if

$$\forall \kappa \in \mathbb{N} \cup \{\aleph_0\} \; \forall \ell \geq 1 \left(\begin{array}{rl} & \exists^{\geq \ell} x \in E(r_1) : |E(x)| = \kappa \\ \iff & \exists^{\geq \ell} x \in E(r_2) : |E(x)| = \kappa \end{array} \right).$$

In other words: for every $\kappa \in \mathbb{N} \cup \{\aleph_0\}$, r_1 and r_2 have the same number of children with exactly κ children. For fixed $\kappa \in \mathbb{N} \cup \{\aleph_0\}$ and $\ell \geq 1$, the equivalence in brackets is an FSO-sentence and therefore decidable by Theorem 5. Thus, the whole formula is indeed a Π^0_1-sentence (note that $2n - 3 = 1$ for $n = 2$).

Now assume that the statement holds for $n - 1$. We have $T_1 \cong T_2$ if and only if

$$\forall v \in E(r_1) \cup E(r_2) \; \forall \ell \geq 1 \left(\begin{array}{rl} & \exists^{\geq \ell} x \in E(r_1) : H(v) \cong H(x) \\ \iff & \exists^{\geq \ell} x \in E(r_2) : H(v) \cong H(x) \end{array} \right).$$

By quantifying over all $v \in E(r_1) \cup E(r_2)$, we quantify over all isomorphism types of trees that occur as a subtree rooted at a child of r_1 or r_2. For each of these isomorphism types τ, we express that r_1 and r_2 have the same number of children x with $H(x)$ of type τ. Note that the automatic trees $H(v)$ and $H(x)$ in the above formula have height $n-1$. Hence, there exists a Π^0_{2n-5}-statement for $H(v) \cong H(x)$. Thus, $T_1 \cong T_2$ is a Π^0_{2n-3}-statement. $\square$

4.2 Arithmetical lower bounds for trees of finite height

In this section, we will show that the upper bounds from Lemmas 25 and 27 are optimal. For the minimal classes $\mathcal{T}_3^{\mathrm{er}}$ and $\mathcal{T}_2$, this is an immediate consequence of Proposition 11:

Corollary 28. *There exists an automatic tree T_{Good} of height 2 and e-rank 3 such that the set of automatic presentations $\mathcal{P}$ with $\mathcal{S}(\mathcal{P}) \cong T_{\mathsf{Good}}$ is Π^0_1-hard. Hence, the isomorphism and the elementary equivalence problems for the classes $\mathcal{T}_2$ and $\mathcal{T}_3^{\mathrm{er}}$ are Π^0_1-hard.*

Proof. Let $\mathcal{E} = (L; \equiv)$ be an automatic equivalence structure without infinite equivalence classes. Now build the tree $T(\mathcal{E})$ as follows:

- The set of nodes is $L \cup \{r\} \cup \{ua \mid u \in L, u \text{ is } \leq_{\mathsf{llex}}\text{-minimal in } [u]_{\equiv}\}$ where r and a are two new letters.
- r is the root, its children are the words ending in a, and the children of ua are the words from $[u]_{\equiv}$.

Then it is clear that $T(\mathcal{E})$ is an automatic tree of height at most 2. Since any node of the form ua has only finitely many successors, it has e-rank 3 (since $\mathbb{N}^{<3}$ does not embed). Furthermore, an automatic presentation for $T(\mathcal{E})$ can be computed from one for $\mathcal{E}$.

Recall the automatic equivalence structure $\mathcal{E}_{\mathsf{Good}}$ (which does not have infinite equivalence classes) from Section 3. Note that $\mathcal{E} \cong \mathcal{E}_{\mathsf{Good}}$ if and only if $T(\mathcal{E}) \cong T(\mathcal{E}_{\mathsf{Good}})$. Hence, we reduced the set of automatic presentations of $\mathcal{E}_{\mathsf{Good}}$ to the set of automatic presentations of $T(\mathcal{E}_{\mathsf{Good}})$. Since the former is Π^0_1-hard by Proposition 11, so is the latter.

For $m, n \in \mathbb{N}$, consider the following FO-formula:

$$\varphi_{m,n} \;=\; \exists r \left(\forall x : (x, r) \notin E \;\wedge\; \exists^{\geq m} y \left((r, y) \in E \;\wedge\; \exists^{=n} z : (y, z) \in E \right) \right).$$

Then, if $\mathcal{E}$ is an equivalence structure without infinite equivalence classes, we have $T(\mathcal{E}) \cong T(\mathcal{E}_{\mathsf{Good}})$ if and only if

$$\forall m, n \in \mathbb{N} : T(\mathcal{E}) \models \varphi_{m,n} \iff T(\mathcal{E}_{\mathsf{Good}}) \models \varphi_{m,n}$$

(here, the relation symbol E from the formula $\varphi_{m,n}$ denotes the edge relation of $T(\mathcal{E})$ and $T(\mathcal{E}_{\mathsf{Good}})$, respectively). Hence, $T(\mathcal{E}) \cong T(\mathcal{E}_{\mathsf{Good}})$ if and only if $T(\mathcal{E}) \equiv T(\mathcal{E}_{\mathsf{Good}})$. Thus, the set of automatic presentations of trees elementary equivalent to $T(\mathcal{E}_{\mathsf{Good}})$ is Π^0_1-hard as well. $\square$

In the rest of this section we will prove a generalization of Corollary 28: The isomorphism problem for the class of automatic trees of height at most $n \geq 2$ is Π^0_{2n-3}-hard. Note that with Lemma 27 it follows that this problem is Π^0_{2n-3}-complete. To prove Π^0_{2n-3}-hardness, we provide a generic reduction from an arbitrary Π^0_{2n-3}-predicate $P_n(x_0)$ to the isomorphism problem for $\mathcal{T}_n$.

In the following, all quantifiers with unspecified range run over $\mathbb{N}_+$. Let $P_n(x_0)$ be a Π^0_{2n-3}-predicate. By Proposition 3, $P_n(x_0)$ is of the form

$$\exists^\infty x_1 \cdots \exists^\infty x_{n-2} \forall y : R(x_0, x_1, \ldots, x_{n-2}, y) ,$$

where R is computable. For $2 \leq i \leq n$ let

$$P_i(x_0, x_1, \ldots, x_{n-i}) = \exists^\infty x_{n-i+1} \cdots \exists^\infty x_{n-2} \forall y : R(x_0, x_1, \ldots, x_{n-2}, y) .$$

Hence, we have

$$\begin{aligned} P_2(x_0, x_1, \ldots, x_{n-2}) &= \forall y : R(x_0, x_1, \ldots, x_{n-2}, y) \quad \text{and} \\ P_{i+1}(x_0, x_1, \ldots, x_{n-i-1}) &= \exists^\infty x_{n-i} : P_i(x_0, x_1, \ldots, x_{n-i-1}, x_{n-i}) . \end{aligned}$$

Let us fix these predicates P_i for the rest of Section 4. W.l.o.g. we can assume

$$\forall \overline{x} \in \mathbb{N}_+^{n-i} \exists x_{n-i} \in \mathbb{N}_+ : P_i(\overline{x}, x_{n-i}) . \tag{6}$$

To ensure this, we can replace the predicate $P_i(\overline{x}, x_{n-i})$ by $P_i(\overline{x}, x_{n-i}) \vee (x_{n-i} = 1)$.[7] By induction on $2 \leq i \leq n$, we will construct the following trees of height i and e-rank $i+1$:

- test trees $T^i_{\overline{c}} \in \mathcal{T}_i$ for $\overline{c} \in \mathbb{N}_+^{n-i+1}$ (which depend on P_i) and
- trees $U^i_\kappa \in \mathcal{T}_i$ for $\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$.

The crucial properties of these trees are the following, where $\overline{c} \in \mathbb{N}_+^{n-i+1}$:

(P1) $P_i(\overline{c})$ holds if and only if $T^i_{\overline{c}} \cong U^i_{\aleph_0}$.
(P2) $P_i(\overline{c})$ does not hold if and only if $T^i_{\overline{c}} \cong U^i_m$ for some $m \in \mathbb{N}_+$.

For $2 \leq i < n$ and $\overline{c} \in \mathbb{N}_+^{n-i}$, the idea is that $T^{i+1}_{\overline{c}} \cong U^{i+1}_m$ if and only if

$$m = \mathsf{card}(\{x_{n-i} \mid P_i(\overline{c}, x_{n-i}) \text{ holds }\}) \in \mathbb{N}_+ \cup \{\aleph_0\} .$$

Note that the case $\mathsf{card}(\{x_{n-i} \mid P_i(\overline{c}, x_{n-i}) \text{ holds }\}) = 0$ is excluded by our assumption (6).

Property (P1) is certainly sufficient for proving Π^0_{2n-3}-hardness (with $i = n$), the second property (P2) and therefore the trees U^i_m for $m < \aleph_0$ are used in the inductive step.

In the following section, we will describe the trees $T^i_{\overline{c}}$ and U^i_κ of height i and prove (P1) and (P2). The subsequent section is then devoted to prove the effective automaticity of these trees.

4.2.1 Construction of trees.

We start with a few definitions concerning forests: Let H_1 and H_2 be two forests. For $\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$, the forest $\kappa \cdot H_1$ is the disjoint union of κ many copies of H_1. Formally, if $H_1 = (V; \leq)$, then $\kappa \cdot H_1 = (V \times \{x \mid 0 \leq x < \kappa\}; \leq')$ with $(v, i) \leq' (w, j)$ if and only if $v \leq w$ and $i = j$. We write $H_1 \sim H_2$ for $\aleph_0 \cdot H_1 \cong \aleph_0 \cdot H_2$. Thus, for countable forests H_1 and H_2, we have $H_1 \sim H_2$ if and only if they are formed, up to isomorphism, by the same set of trees (i.e., any tree is isomorphic to some connected component of H_1 if and only if it is isomorphic to some connected component of H_2). If H is a forest, then we denote with $\langle H \rangle$ the tree that results from adding a new least element to H. The construction in the following two Sections 4.2.1.1 and 4.2.1.2 is similar to a construction from [17] for levels of the hyperarithmetical hierarchy.

[7] The formulas $\exists^\infty x : \varphi(x)$ and $\exists^\infty x : (\varphi(x) \vee (x = 1))$ are equivalent. Moreover if x and y are different variables, then the formulas $(x = 1) \vee (\exists^\infty y : \varphi(x, y))$ and $\exists^\infty y : (\varphi(x, y) \vee (x = 1))$ are equivalent.

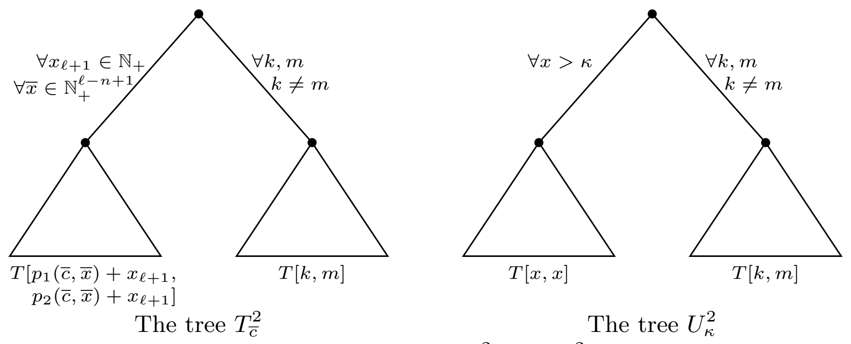

$\forall x_{\ell+1} \in \mathbb{N}_+$
$\forall \overline{x} \in \mathbb{N}_+^{\ell-n+1}$
$\forall k, m$
$k \neq m$
$T[p_1(\overline{c}, \overline{x}) + x_{\ell+1},$
$p_2(\overline{c}, \overline{x}) + x_{\ell+1}]$
$T[k, m]$
The tree $T^2_{\overline{c}}$
$\forall x > \kappa$
$\forall k, m$
$k \neq m$
$T[x, x]$
$T[k, m]$
The tree U^2_κ

Fig. 2. The tree $T^2_{\overline{c}}$ and U^2_κ

4.2.1.1 Induction base: construction of $T^2_{\overline{c}}$ and U^2_κ. Note that P_2 is an $(n-1)$-ary Π^0_1-predicate. By Matiyasevich's theorem, we find two non-zero polynomials $p_1(x_1, \ldots, x_\ell), p_2(x_1, \ldots, x_\ell) \in \mathbb{N}[\overline{x}]$, $\ell > n - 1$, such that for any $\overline{c} \in \mathbb{N}_+^{n-1}$:

$$P_2(\overline{c}) \text{ holds} \iff \forall \overline{x} \in \mathbb{N}_+^{\ell-n+1} : p_1(\overline{c}, \overline{x}) \neq p_2(\overline{c}, \overline{x})\,. \tag{7}$$

For two numbers $k, m \in \mathbb{N}_+$, let $T[k, m]$ denote the tree of height 1 with exactly $C(k, m)$ leaves, where C is the injective polynomial function from (2). Then define the following forests:

$$\begin{aligned}
H^2 &= \biguplus \{T[k, m] \mid k, m \in \mathbb{N}_+, k \neq m\}\,, \\
H^2_{\overline{c}} &= H^2 \uplus \biguplus \{T[p_1(\overline{c}, \overline{x}) + x_{\ell+1}, p_2(\overline{c}, \overline{x}) + x_{\ell+1}] \mid \overline{x} \in \mathbb{N}_+^{\ell-n+1}, x_{\ell+1} \in \mathbb{N}_+\} \text{ and} \\
J^2_\kappa &= H^2 \uplus \biguplus \{T[x, x] \mid x \in \mathbb{N}_+, x > \kappa\} \text{ for } \kappa \in \mathbb{N}_+ \cup \{\aleph_0\}\,.
\end{aligned}$$

Note that $J^2_{\aleph_0} = H^2$. Moreover, the forests J^2_κ ($\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$) are pairwise non-isomorphic, since C is injective.

The trees $T^2_{\overline{c}}$ and U^2_κ, resp., are obtained from $H^2_{\overline{c}}$ and J^2_κ, resp., by taking countably many copies and adding a root (see Figure 2):

$$T^2_{\overline{c}} = \langle \aleph_0 \cdot H^2_{\overline{c}} \rangle \text{ and } U^2_\kappa = \langle \aleph_0 \cdot J^2_\kappa \rangle\,. \tag{8}$$

The following lemma (stating (P1) for the Π^0_1-predicate P_2, i.e., for $i = 2$) is proved in a similar way as Theorem 12.

Lemma 29. *For any $\overline{c} \in \mathbb{N}_+^{n-1}$, we have*

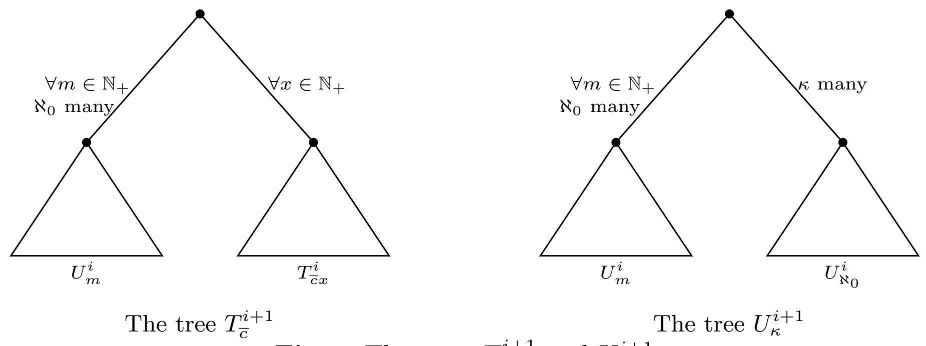
$$P_2(\overline{c}) \text{ holds} \iff H^2_{\overline{c}} \sim J^2_{\aleph_0} \iff T^2_{\overline{c}} \cong U^2_{\aleph_0}\,.$$

Proof. By (8), it suffices to show the first equivalence. First, assume that $P_2(\overline{c})$ holds. We have to prove that the forests $H^2_{\overline{c}}$ and $J^2_{\aleph_0} = H^2$ contain the same trees (up to isomorphism). Clearly, every tree from H^2 is contained in $H^2_{\overline{c}}$. For the other direction, let $\overline{x} \in \mathbb{N}_+^{\ell-n+1}$ and $x_{\ell+1} \in \mathbb{N}_+$. Then the tree $T[p_1(\overline{c}, \overline{x}) + x_{\ell+1}, p_2(\overline{c}, \overline{x}) + x_{\ell+1}]$ occurs in $H^2_{\overline{c}}$. Since $P_2(\overline{c})$ holds, we have $p_1(\overline{c}, \overline{x}) \neq p_2(\overline{c}, \overline{x})$ by (7) and therefore $p_1(\overline{c}, \overline{x}) + x_{\ell+1} \neq p_2(\overline{c}, \overline{x}) + x_{\ell+1}$. Hence this tree also occurs in H^2.

Conversely suppose $H^2_{\overline{c}} \sim H^2$ and let $\overline{x} \in \mathbb{N}_+^{\ell-n+1}$. Then the tree $T[p_1(\overline{c}, \overline{x}) + 1, p_2(\overline{c}, \overline{x}) + 1]$ occurs in $H^2_{\overline{c}}$ and therefore in H^2. Hence $p_1(\overline{c}, \overline{x}) \neq p_2(\overline{c}, \overline{x})$. Since $\overline{x}$ was chosen arbitrarily, this implies $P_2(\overline{c})$. □

Lemma 30. *For every $\overline{c} \in \mathbb{N}_+^{n-1}$ there exists $\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$ such that $T^2_{\overline{c}} \cong U^2_\kappa$.*

Proof. It suffices to show $H^2_{\overline{c}} \sim J^2_\kappa$ for some $\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$. This holds, because if $H^2_{\overline{c}}$ contains a tree of the form $T[m, m]$ for some m (we must have $m \geq 2$), then it contains all trees $T[x, x]$ for $x \geq m$. □



$\forall m \in \mathbb{N}_+$
$\aleph_0$ many
$\forall x \in \mathbb{N}_+$
U^i_m
$T^i_{\overline{c}x}$
The tree $T^{i+1}_{\overline{c}}$
$\forall m \in \mathbb{N}_+$
$\aleph_0$ many
κ many
U^i_m
$U^i_{\aleph_0}$
The tree U^{i+1}_κ

Fig. 3. The trees $T^{i+1}_{\overline{c}}$ and U^{i+1}_κ

Lemma 29 and 30 imply the following lemma, which states (P2) for the Π^0_1-predicate P_2, i.e., for $i = 2$:

Lemma 31. *For any $\overline{c} \in \mathbb{N}^{n-1}_+$, we have:*

$$P_2(\overline{c}) \text{ does not hold} \iff \exists m \in \mathbb{N}_+ : T^2_{\overline{c}} \cong U^2_m\,.$$

This finishes the construction of the trees $T^2_{\overline{c}}$ and U^2_κ for $\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$, and the verification of properties (P1) and (P2).

4.2.1.2 Induction step: construction of $T^{i+1}_{\overline{c}}$ and U^{i+1}_κ. Note that P_{i+1} is a $(n-i)$-ary predicate and P_i a $(n-i+1)$-ary one.

We now apply the induction hypothesis. For any $\overline{c} \in \mathbb{N}^{n-i}_+$, $x \in \mathbb{N}_+$, and $\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$, let $T^i_{\overline{c}x}$ and U^i_κ be trees of height i such that:

- $P_i(\overline{c}, x)$ holds if and only if $T^i_{\overline{c}x} \cong U^i_{\aleph_0}$.
- $P_i(\overline{c}, x)$ does not hold if and only if $T^i_{\overline{c}x} \cong U^i_m$ for some $m \in \mathbb{N}_+$.

Let us define the forest

$$H^{i+1} = \biguplus\{\aleph_0 \cdot U^i_m \mid m \in \mathbb{N}_+\}\,.$$

The trees $T^{i+1}_{\overline{c}}$ and U^{i+1}_κ ($\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$) are defined as follows:

$$T^{i+1}_{\overline{c}} = \langle H^{i+1} \uplus \biguplus\{T^i_{\overline{c}x} \mid x \in \mathbb{N}_+\}\rangle \quad \text{and} \quad U^{i+1}_\kappa = \langle H^{i+1} \uplus \kappa \cdot U^i_{\aleph_0}\rangle\,. \tag{9}$$

Note that the height of any of these trees is one more than the height of the forests defining them and therefore at most $i + 1$.

The following lemma shows (P1) for the Π^0_{2i-1}-predicate P_{i+1}.

Lemma 32. *For any $\overline{c} \in \mathbb{N}^{n-i}_+$, we have*

$$P_{i+1}(\overline{c}) \text{ holds} \iff T^{i+1}_{\overline{c}} \cong U^{i+1}_{\aleph_0}\,.$$

Proof. We have $T^{i+1}_{\overline{c}} \cong U^{i+1}_{\aleph_0}$ if and only if $T^{i+1}_{\overline{c}}$ contains infinitely many copies of the tree $U^i_{\aleph_0}$. Since $U^i_{\aleph_0} \not\cong U^i_m$ for all $m \in \mathbb{N}_+$, this is equivalent to the existence of infinitely many $x \in \mathbb{N}_+$ with $T^i_{\overline{c}x} \cong U^i_{\aleph_0}$. By induction, this holds, if and only if there exist infinitely many $x \in \mathbb{N}_+$ with $P_i(\overline{c}, x)$, i.e., if and only if $P_{i+1}(\overline{c})$ holds. □

The following lemma states (P2) for the Π^0_{2i-1}-predicate P_{i+1}:

Lemma 33. *For any $\overline{c} \in \mathbb{N}^k_+$, we have*

$$P_{i+1}(\overline{c}) \text{ does not hold} \iff \exists m \in \mathbb{N}_+ : T^{i+1}_{\overline{c}} \cong U^{i+1}_m\,.$$

Proof. Properties (P1) and (P2) of the trees $T^i_{\overline{c}x}$ ($x \in \mathbb{N}_+$) imply that every tree $T^i_{\overline{c}x}$ ($x \in \mathbb{N}_+$) is isomorphic to one of the trees U^i_κ for some $\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$. This implies that for every $\overline{c} \in \mathbb{N}^k_+$, there exists $\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$ such that $T^{i+1}_{\overline{c}} \cong U^{i+1}_\kappa$. With Lemma 32 we get

$$P_{i+1}(\overline{c}) \text{ does not hold } \iff T^{i+1}_{\overline{c}} \not\cong U^{i+1}_{\aleph_0} \iff \exists m \in \mathbb{N}_+ : T^{i+1}_{\overline{c}} \cong U^{i+1}_m\,.$$

□

In summary, we obtained the following:

Proposition 34. *For the Π^0_{2n-3}-predicate $P_n(x_0)$ we have for all $c \in \mathbb{N}_+$:*

$$P_n(c) \textit{ holds } \iff T^n_c \cong U^n_{\aleph_0}\,.$$

To infer the Π^0_{2n-3}-hardness of the isomorphism problems for $\mathcal{T}_n$ and $\mathcal{T}^{\mathrm{er}}_{n+1}$ from this proposition, it remains to be shown that the trees T^n_c and $U^n_{\aleph_0}$ are effectively automatic – this is the topic of the next section.

4.2.2 Automaticity.

Automatic presentations of the trees $T^i_{\overline{x}}$ and U^i_m for $m \in \mathbb{N}_+ \cup \{\aleph_0\}$ will be constructed inductively. Note that the construction of $T^{i+1}_{\overline{x}}$ involves all the trees $T^i_{\overline{x}\,x}$ for $x \in \mathbb{N}_+$. Hence we need *one single automatic presentation* for the forest consisting of all these trees. Therefore, we will deal with forests.

Let us take symbols a, $\#$, and β; later we need further symbols b, c, d, and \$. For $n_1, \dots, n_j \in \mathbb{N}_+$ and a word $w \in a^{n_1}\#^+ \cdots a^{n_{j-1}}\#^+ a^{n_j}$ we define the tuple $t(w) = (n_1, \dots, n_j)$. For $2 \le i \le n$ let R_i be the regular language

$$R_i = a^+(\#a^+)^{n-i}.$$

Technically, this section proves by induction over i the following statement:

Proposition 35. *For $2 \le i \le n$, one can compute an automatic forest $\mathcal{F}_i$ with the following properties:*

- *the set of roots of $\mathcal{F}_i$ is $(R_i \cup \beta^*)\#$.*
- *$\mathcal{F}_i(v\#) \cong T^i_{t(v)}$ for all $v \in R_i$.*
- *$\mathcal{F}_i(\beta^m\#) \cong U^i_m$ for all $m \in \mathbb{N}_+$.*
- *$\mathcal{F}_i(\#) \cong U^i_{\aleph_0}$.*

This will give the desired result since T^n_x is then isomorphic to the connected component of $\mathcal{F}_n$ that contains the word $a^x\#$ (and similarly for U^n_κ). Note that this connected component is automatic by Theorem 5 since it consists of all nodes above $a^x\#$. Moreover, an automatic presentation for the connected component containing $a^x\#$ can be computed from x.

4.2.2.1 Induction base: the automatic forest $\mathcal{F}_2$. The forest $\mathcal{F}_2$ will be formed by the language and certain accepting runs of a finite automaton $\mathcal{A}$; this fact will be used later in Section 4.4. More precisely, let $\mathcal{A}$ be a finite automaton. We define the forest $\mathsf{forest}(\mathcal{A})$ as follows: Clearly, $(L(\mathcal{A}); \preceq_{L(\mathcal{A})})$ (recall that $\preceq_{L(\mathcal{A})}$ denotes the prefix relation restricted to $L(\mathcal{A})$) is a forest. The set of all leaves of the forest $(L(\mathcal{A}); \preceq)$ is again a regular language; let us denote this language with $\mathsf{leaves}(\mathcal{A})$. Then, we define the forest

$$\mathsf{forest}(\mathcal{A}) = (L(\mathcal{A}) \uplus \mathsf{Run}(\mathcal{A}, \mathsf{leaves}(\mathcal{A})); \le),$$

where

$$\le \ = \ \preceq_{L(\mathcal{A})} \cup \{(u, r) \mid u \in L(\mathcal{A}), r \in \mathsf{Run}(\mathcal{A}, \mathsf{leaves}(\mathcal{A})), u \preceq \mathsf{lab}(r)\}.$$

Clearly, $\mathsf{forest}(\mathcal{A})$ is automatic. Intuitively, we take the forest resulting from the prefix order on the regular language $L(\mathcal{A})$ and append to each leaf v of $(L(\mathcal{A}); \preceq_{L(\mathcal{A})})$ all runs of $\mathcal{A}$ on v as children.

All these children are leaves in the forest $\mathsf{forest}(\mathcal{A})$. In case $\mathsf{forest}(\mathcal{A})$ is a tree, we denote this tree with $\mathsf{tree}(\mathcal{A})$.

From now on, we use the notations from Section 4.2.1.1. The following construction is visualized in Figure 4–6.

Besides the symbols a, $\#$, and β, we need the auxiliary symbols b, c, and d. The symbol $\#$ has two purposes in the following construction: (i) it separates consecutive blocks of a's, b's, and c's and (ii) it ensures that some subtrees appears $\aleph_0$ many times. Let us start with the regular language

$$\begin{aligned} L_2 \;&= R_2\Big(\{\varepsilon\} \,\cup\, (\#^+a^+)^{\ell-n+2} \,\cup\, \#^+b^+\#b^+ \,\cup\, \#^+c^+\#c^+\Big)\# \;\cup \\ &\beta^*\Big(\{\varepsilon\} \,\cup\, \#^+b^+\#b^+ \,\cup\, \#^+c^+\#c^+ \,\cup\, \beta\#^+d^+\Big)\# \end{aligned}$$

Using Lemma 9, we can construct finite automata $\mathcal{A}_1, \ldots, \mathcal{A}_4$ with the following properties:

– $L(\mathcal{A}_1) = R_2(\#^+a^+)^{\ell-n+2}\#$ and on every word $v\# \in L(\mathcal{A}_1)$ the automaton $\mathcal{A}_1$ has exactly $C(p_1(\overline{x}) + x_{\ell+1}, p_2(\overline{x}) + x_{\ell+1})$ many accepting runs, where $(\overline{x}, x_{\ell+1}) = t(v)$,

– $L(\mathcal{A}_2) = (R_2 \cup \beta^*)\#^+b^+\#b^+\#$ and on every word from $(R_2 \cup \beta^*)\#^+b^x\#b^y\#$, the automaton $\mathcal{A}_2$ has exactly $C(x, x + y)$ many accepting runs.

– $L(\mathcal{A}_3) = (R_2 \cup \beta^*)\#^+c^+\#c^+\#$ and on every word from $(R_2 \cup \beta^*)\#^+c^x\#c^y\#$, the automaton $\mathcal{A}_3$ has exactly $C(x + y, x)$ many accepting runs.

– $L(\mathcal{A}_4) = \beta^+\#^+d^+\#$ and on every word from $\beta^m\#^+d^x\#$, the automaton $\mathcal{A}_4$ has exactly $C(m + x, m + x)$ many accepting runs.

Note that $L(\mathcal{A}_i) \subseteq L_2$ for $1 \le i \le 4$ and that the languages $L(\mathcal{A}_1), \ldots, L(\mathcal{A}_4)$ are pairwise disjoint. Moreover, their union is exactly the set of leaves of the forest $(L_2; \preceq)$. From the automata $\mathcal{A}_1, \ldots, \mathcal{A}_4$ we can easily construct an automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L_2$ and for all $1 \le i \le 4$ and $w \in L(\mathcal{A}_i)$, we have $|\mathsf{Run}(\mathcal{A}, w)| = |\mathsf{Run}(\mathcal{A}_i, w)|$. Then we set

$$\mathcal{F}_2 = \mathsf{forest}(\mathcal{A}).$$

We claim that with this definition of $\mathcal{F}_2$ all properties of Proposition 35 hold for $i = 2$. Clearly, the set of roots of $\mathcal{F}_2$ is indeed $(R_2 \cup \beta^*)\#$. Here, the final $\#$ in all words from L_2 is important in order to make the set $(R_2 \cup \beta^*)\#$ prefix-free.

Now consider a root $v\#$ with $v \in R_2$. The set of children of $v\#$ is the prefix-free set

$$v\big((\#^+a^+)^{\ell-n+2} \,\cup\, \#^+b^+\#b^+ \,\cup\, \#^+c^+\#c^+\big)\#.$$

The automaton $\mathcal{A}_1$ ensures that the subtree rooted in a word $vw\#$ ($w \in (\#^+a^+)^{\ell-n+2}$) is $T[p_1(\overline{x}) + x_{\ell+1}, p_2(\overline{x}) + x_{\ell+1}]$, where $t(vw) = (\overline{x}, x_{\ell+1})$. Moreover the automaton $\mathcal{A}_2$ (resp. $\mathcal{A}_3$) ensures that the subtree rooted in a word $v\#^m b^x\#b^y\#$ (resp. $v\#^m c^x\#c^y\#$) is $T[x, x + y]$ (resp. $T[x + y, x]$). Finally, since each of these subtrees appears $\aleph_0$ many times (due to the factors $\#^+$), the subtree rooted in $v\#$ is indeed isomorphic to $T^2_{t(v)}$.

A similar argument shows that the subtree rooted in a root $\beta^m\#$ (resp., $\#$) is U^2_m (resp., $U^2_{\aleph_0}$) for all $m \in \mathbb{N}_+$. For U^2_m note that the subtree rooted in a child $\beta^m\#^n d^x\#$ is $T[m + x, m + x]$.

4.2.2.2 Induction step: the automatic forest $\mathcal{F}_{i+1}$. Suppose $\mathcal{F}_i = (V_i; \le_i)$ is as described in Proposition 35. The following construction is visualized in Figure 7–9. We note that the forest $\mathcal{F}_{i+1}$ (for $i \ge 2$) will *not* be realized as $\mathsf{forest}(\mathcal{A})$ for a finite automaton $\mathcal{A}$.

For a regular subset $K \subseteq V_i$ (K will be either $R_i\#$, $\beta^+\#$, or $\{\#\}$ below) let us write $\mathcal{F}_i(K)$ for the set of all nodes $v \in V_i$ such that $u \le_i v$ for some $u \in K$. Since $\mathcal{F}_i$ has bounded height, the set $\mathcal{F}_i(K)$ is regular if K is regular. Take a new symbol \$ and let the regular language

$$V_{i+1} \;=\; (R_{i+1} \cup \beta^*)\# \;\cup\; \mathcal{F}_i(R_i\#) \;\cup\; (R_{i+1} \cup \beta^*)\#^+\mathcal{F}_i(\beta^+\#) \;\cup\; (\beta^* \cup \#)\$^+\mathcal{F}_i(\#).$$

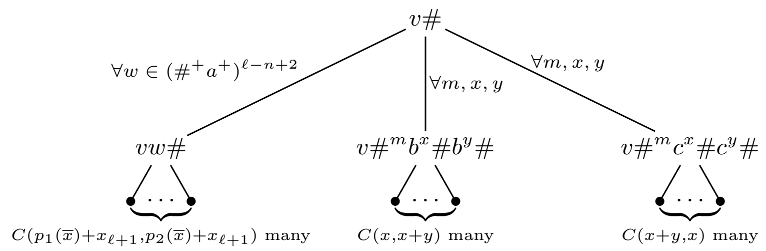


Fig. 4. Automatic presentation of $T^2_{t(v)}$ for $v \in R_2$: Let $t(vw) = (\overline{x}, x_{\ell+1})$.

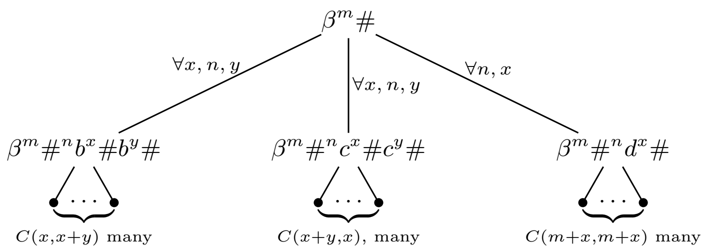


Fig. 5. Automatic presentation of U^2_m ($m \in \mathbb{N}_+$)

be the universe of the forest $\mathcal{F}_{i+1}$. The partial order $\leq_{i+1}$ of $\mathcal{F}_{i+1}$ is the reflexive and transitive closure of the following relation $\sqsubseteq$:

- $u \sqsubseteq v$ for all $u \in (R_{i+1} \cup \beta^*)\#$ and $v \in V_{i+1}$ with $u \preceq v$.
- $u \sqsubseteq v$ for all $u, v \in \mathcal{F}_i(R_i\#)$ with $u \leq_i v$.
- $wu \sqsubseteq wv$ for all $w \in (R_{i+1} \cup \beta^*)\#^+$, and $u, v \in \mathcal{F}_i(\beta^+\#)$ with $u \leq_i v$.
- $wu \sqsubseteq wv$ for all $w \in (\beta^* \cup \#)\$^+$ and $u, v \in \mathcal{F}_i(\#)$ with $u \leq_i v$.
- $\beta^m\# \sqsubseteq \beta^{m-x}\$^x\#$ for all $m \in \mathbb{N}_+$ and $1 \leq x \leq m$ (these edges are the reason for $\mathcal{F}_{i+1}$ not be being of the form $\mathsf{forest}(\mathcal{A})$ for a finite automaton $\mathcal{A}$).

We claim that with this definition of $\mathcal{F}_{i+1}$ all properties of Proposition 35 hold.

First of all, the above definition of $\sqsubseteq$ implies that $\mathcal{F}_{i+1}$ is indeed an automatic forest. Moreover, the definition of $\sqsubseteq$ implies that the set of roots of $\mathcal{F}_{i+1}$ is $(R_{i+1} \cup \beta^*)\#$. Now consider $v\# \in R_{i+1}\#$. The set of children of $v\#$ is $v\#a^+\# \;\cup\; v\#^+\beta^+\#$. Note that $v\#a^+\# \subseteq R_i\#$. By induction, the subtree rooted in the child $v\#a^x\#$ is $T^i_{t(v)x}$. Moreover, by induction, the subtree rooted in the child $v\#^x\beta^m\#$ is U^i_m. Hence, the tree rooted in $v\# \in R_{i+1}\#$ consists of the root $v\#$ together with all trees $T^i_{t(v)x}$ (for $x \in \mathbb{N}_+$) and $\aleph_0$ many copies of U^i_m (for all $m \in \mathbb{N}_+$). Hence, the tree rooted in $v\# \in R_{i+1}\#$ is indeed $T^{i+1}_{t(v)}$.

The children of the root $\beta^m\#$ ($m \in \mathbb{N}_+$) are the nodes $\beta^{m-x}\$^x\#$ (for all $1 \leq x \leq m$) and $\beta^m\#^x\beta^k\#$ (for all $k, x \in \mathbb{N}_+$). Moreover, the subtree rooted in the child $\beta^{m-x}\$^x\#$ (resp., $\beta^m\#^x\beta^k\#$) is the tree $U^i_{\aleph_0}$ (resp. U^i_k). It follows that the tree rooted in $\beta^m\#$ is indeed U^{i+1}_m. Finally, similar arguments show that $U^{i+1}_{\aleph_0}$ is the tree rooted in $\#$.

Theorem 36. *The following holds:*

(a) For any $n \geq 2$, the isomorphism problem $\mathsf{Iso}(\mathcal{T}_n)$ for automatic trees of height at most n is Π^0_{2n-3}-complete.
(b) For any $n \geq 3$, the isomorphism problem $\mathsf{Iso}(\mathcal{T}^{\mathrm{er}}_n)$ for automatic trees of e-rank at most n is Π^0_{2n-5}-complete.

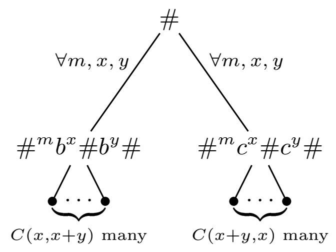


Fig. 6. Automatic presentation of $U^2_{\aleph_0}$

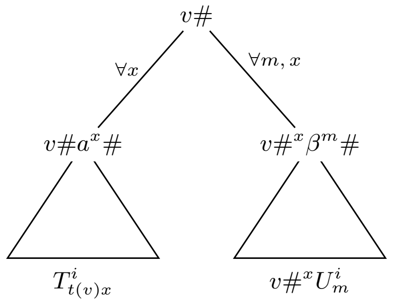


Fig. 7. Automatic presentation of $T^{i+1}_{t(v)}$ for $v \in R_{i+1}$

(c) The isomorphism problems for the following classes of automatic structures are recursively equivalent to $\mathsf{FOTh}(\mathbb{N};+,\times)$*: (i) automatic trees of finite height, (ii) well-founded automatic trees, (iii) automatic trees with only countably many infinite branches.*

Proof. We first prove (a). Containment in Π^0_{2n-3} was shown in Lemma 27. For the hardness, let $P_n \subseteq \mathbb{N}_+$ be any Π^0_{2n-3}-predicate and let $x \in \mathbb{N}_+$. Then, as above, we construct the automatic forest $\mathcal{F}_n$ of height n. The trees T^n_x and $U^n_{\aleph_0}$ are first-order definable in $\mathcal{F}_n$ since they are (isomorphic to) the trees rooted at $a^x\#$ and $\#$, resp. Hence these two trees are automatic. By Proposition 34, they are isomorphic if and only if $P_n(c)$ holds.

The upper bound in (b) is stated in Lemma 25. The lower bound follows as in (a) from the fact that $\mathsf{erank}(T^n_c) = \mathsf{erank}(U^n_{\aleph_0}) = n+1$ (one can embed into these trees $\mathbb{N}^{<n}$ but not $\mathbb{N}^{<n+1}$).

The upper bound in (c) for the largest class (automatic trees with only countably many infinite branches) is stated in Proposition 26. The lower bound for the smallest class (automatic trees of finite height) follows from our proof for (a), since the construction is uniform in the predicate P. $\square$

Concerning the lower bound, we actually proved a slightly stronger statement: For every $n \geq 2$, there exists a fixed Π^0_{2n-3}-complete set $P_{2n-3} \subseteq \mathbb{N}_+$. If we apply our construction, we obtain a *fixed automatic forest* $\mathcal{F}_n$ of height n with the following properties: It is Π^0_{2n-3}-complete to determine, whether for a given $x \in \mathbb{N}_+$, the tree rooted at $a^x\#$ in $\mathcal{F}_n$ is isomorphic to the tree rooted at $\#$ in $\mathcal{F}_n$.

4.3 Computable trees of finite height

In this section, we briefly discuss the isomorphism problem for computable trees of finite height. In an automatic tree, one can compute the root by Theorem 5 which is not the case for computable trees. A similar remark concerns the edge relation E: in an automatic tree, it is FA recognizable, but in a computable tree, it need not be computable. But these two concepts (root and edge relation) are foundational for our proof of the upper bounds of the isomorphism problem. Therefore, here, we consider rooted successor trees, i.e., structures of the form $(V;E,r)$ where E is the edge relation of some tree $(V;\leq)$ with root r.

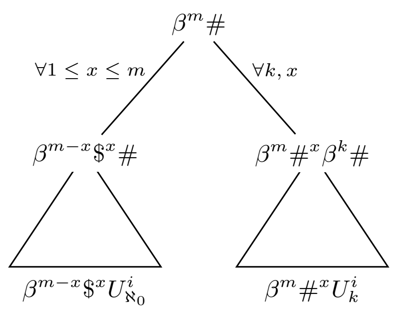


Fig. 8. Automatic presentation of U^{i+1}_m ($m \in \mathbb{N}_+$)

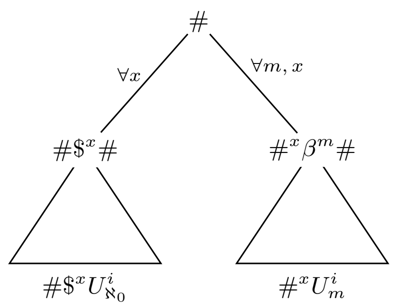


Fig. 9. Automatic presentation of $U^{i+1}_{\aleph_0}$

Theorem 37. *For every $n \geq 1$, the isomorphism problem for computable rooted successor trees of height at most n is Π^0_{2n}-complete.*

Proof. For the upper bound, let us first assume that $n = 1$. Two computable trees T_1 and T_2 of height 1 are isomorphic if and only if: for every $k \geq 0$, there exist at least k nodes in T_1 if and only if there exist at least k nodes in T_2. This is a Π^0_2-statement. For the inductive step, we can use arguments similar to those from the proof of Lemma 27.

For the lower bound, we first note that the isomorphism problem for computable rooted trees of height 1 is Π^0_2-complete. The problem whether a given recursively enumerable set is infinite is Π^0_2-complete [35]. For a given deterministic Turing-machine M, we construct a computable tree $T(M)$ of height 1 as follows: the set of leaves of $T(M)$ is the set of all accepting computations of M. We add a root to the tree and connect the root to all leaves. If $L(M)$ is infinite, then $T(M)$ is isomorphic to the height-1 tree with infinitely many leaves. If $L(M)$ is finite, then there exists $m \in \mathbb{N}$ such that $T(M)$ is isomorphic to the height-1 tree with m leaves. We can use this construction as the base case for our construction in Section 4.2.1.2. This yields the lower bound for all $n \geq 1$. □

Remark 38. Among other result, in [7] it is shown that the isomorphism problem of computable equivalence structures is Π^0_4-complete. This result also follows from Theorem 37. This is because given any computable equivalence structure $\mathcal{E} = (D; \approx)$ one may effectively construct a computable rooted successor tree $T_{\mathcal{E}} = (V; E, r)$ of height 2 such that for any two computable equivalence structures $\mathcal{E}_1$ and $\mathcal{E}_2$, we have $\mathcal{E}_1 \cong \mathcal{E}_2$ if and only if $T_{\mathcal{E}_1} \cong T_{\mathcal{E}_2}$: Let $V = \{r\} \cup C \cup D$ where C is the set of children of the root r and D is the set of leaves. Two nodes $u, v \in D$ have the same parent node if and only if $u \approx v$. The converse construction transforms any computable rooted successor tree T of height 2 to a computable equivalence structure $\mathcal{E}$ such that $T_{\mathcal{E}} \cong T$.

4.4 Σ^1_1-hardness for trees

In this section, we prove that the isomorphism problem for the class of all automatic trees is hard (and hence complete) for Σ^1_1. In [26], the authors prove Σ^1_1-completeness for the isomorphism problem for automatic graphs. Their proof, as pointed out in [38], can be easily adapted to show the same lower bound for automatic *successor trees*. So, it is important to note that we work (as in all of this paper) with *order trees*, i.e., trees viewed as partial orders. We will prove Σ_1^1-hardness for trees of the form $\mathsf{tree}(\mathcal{A})$ for a finite automaton $\mathcal{A}$; see Section 4.2.2 (this fact will be needed in Section 4.5).

For a word $u = a_0 a_1 \cdots a_n \in \{0,1\}^*1$ with $a_i \in \{0,1\}$, let $\mathsf{num}(u) = \sum_{i=0}^{n} 2^i a_i$, i.e., u is the binary expansion of $\mathsf{num}(u)$ (least significant bit first).

Lemma 39. *There exists an automaton $\mathcal{A}_{\mathsf{num}}$ on the alphabet $\{0,1\}$ with $L(\mathcal{A}_{\mathsf{num}}) = \{0,1\}^*1$ such that $\mathsf{num}(w) = |\mathsf{Run}(\mathcal{A}_{\mathsf{num}}, w)|$ for all $w \in \{0,1\}^*1$.*

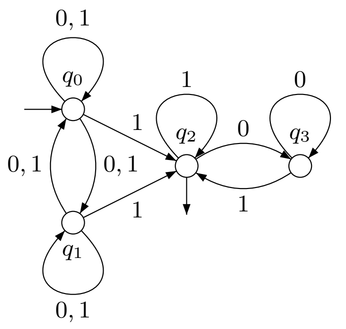


Fig. 10. The automaton $\mathcal{A}_{\mathsf{num}}$

Proof. Let $\mathcal{A}_{\mathsf{num}}$ be the automaton from Figure 10. Note that for each 1, the automaton can move from q_0 and q_1 resp. to the final state q_2, then the rest of the input is processed deterministically. If $\mathcal{A}_{\mathsf{num}}$ moves at input position k to the final state q_2, then there are 2^{k-1} possible runs until reaching input position k. Hence, if $p_1 < p_2 < \cdots < p_n$ are the 1-positions in an input $w \in \{0,1\}^*1$ ($p_1 \geq 1$, $p_n = |w|$) then $\mathcal{A}_{\mathsf{num}}$ has $\sum_{i=1}^{n} 2^{p_i - 1} = \mathsf{num}(w)$ accepting runs on w. $\square$

The following lemma is analogous to Lemma 9. The only difference is that the first argument to the polynomial $p(x, \overline{y})$ is encoded in binary. To prove the lemma, we use the same construction as for Lemma 9 except the induction base $p(x) = x$ is handled by Lemma 39.

Lemma 40. *There exists an algorithm that, given a non-zero polynomial $p(x, \overline{y}) \in \mathbb{N}[x, \overline{y}]$ in $k+1$ variables, constructs an automaton $\mathcal{A}[p(x, \overline{y})]$ on the alphabet $\{0, 1, a, \#\}$ with $L(\mathcal{A}[p(x, \overline{y})]) = \{0,1\}^*1(\#^+ a^+)^k \#$ such that*

$$p(\mathsf{num}(w), n_1, \ldots, n_k) = |\mathsf{Run}(\mathcal{A}[p(x, \overline{y})], w\#^{m_1} a^{n_1} \cdots \#^{m_k} a^{n_k} \#)|$$

for all $w \in \{0,1\}^*1$ and $m_1, n_1, \ldots, m_k, n_k \in \mathbb{N}_+$.

The following lemma will be only used for the case that the set A is decidable. We state the lemma for arbitrary Σ_2^0-sets, since the proof is exactly the same.

Lemma 41. *There exist two trees U_0 and U_1 of height 3 ($U_0 \not\cong U_1$) with the following property: For a given index of a Σ_2^0-set $A \subseteq \{0,1\}^*1$ one can effectively construct a finite automaton $\mathcal{B}$ such that $\mathcal{F} = \mathsf{forest}(\mathcal{B})$ is a forest of height 3 with the following properties:*

- *The set of roots of $\mathcal{F}$ is $\{0,1\}^*1\#$.*
- *For every $w \in \{0,1\}^*1$, $\mathcal{F}(w\#) \cong U_1$ if $w \in A$, and $\mathcal{F}(w\#) \cong U_0$ if $w \notin A$.*

Proof. Recall the definition of the trees U^2_κ ($\kappa \in \mathbb{N}_+ \cup \{\aleph_0\}$) from Section 4.2.1.1. We define the trees U_0 and U_1 as follows:

$$U_0 = \left\langle \biguplus \{\aleph_0 \cdot U^2_m \mid m \in \mathbb{N}_+\} \right\rangle$$
$$U_1 = \left\langle \biguplus \{\aleph_0 \cdot U^2_\kappa \mid \kappa \in \mathbb{N}_+ \cup \{\aleph_0\}\} \right\rangle$$

Fix a Σ^0_2-set $A \subseteq \{0,1\}^*1$. Using the effectiveness of Matiyasevich's theorem, we can compute from an index of A two polynomials $p_1(x, y, \overline{z})$ and $p_2(x, y, \overline{z})$ with coefficients in $\mathbb{N}$ such that

$$A = \{w \in \{0,1\}^*1 \mid \exists y \in \mathbb{N}_+ \ \forall \overline{z} \in \mathbb{N}^\ell_+ : p_1(\mathsf{num}(w), y, \overline{z}) \neq p_2(\mathsf{num}(w), y, \overline{z})\} .$$

We first repeat the construction from Section 4.2.2.1 with Lemma 40 instead of Lemma 9, i.e., the first argument to the polynomials p_1 and p_2 is binary encoded. We obtain a finite automaton $\mathcal{B}'$ such that the forest $\mathcal{F}' = \mathsf{forest}(\mathcal{B}')$ satisfies the following properties, which are analogous to Proposition 35:

- The set of roots of $\mathcal{F}'$ is $\{0,1\}^*1\#^+a^+\# \cup \alpha^+\#$ (we do not need the root $\#$ of $\mathcal{F}_2$).
- $\mathcal{F}'(w\#^n a^y \#) \cong T^2_{\mathsf{num}(w)\,y}$ for all $w \in \{0,1\}^*1$ and $n, y \in \mathbb{N}_+$.
- $\mathcal{F}'(\alpha^m\#) \cong U^2_m$ for all $m \in \mathbb{N}_+$.

Together with Lemmas 29 and 31 we get the following properties for all $w \in \{0,1\}^*1$ and $n, y \in \mathbb{N}_+$:

$$\forall \overline{z} \in \mathbb{N}^\ell_+ : p_1(\mathsf{num}(w), y, \overline{z}) \neq p_2(\mathsf{num}(w), y, \overline{z}) \text{ holds} \iff \mathcal{F}'(w\#^n a^y\#) \cong U^2_{\aleph_0}$$
$$\exists \overline{z} \in \mathbb{N}^\ell_+ : p_1(\mathsf{num}(w), y, \overline{z}) = p_2(\mathsf{num}(w), y, \overline{z}) \text{ holds} \iff \exists m \in \mathbb{N}_+ : \mathcal{F}'(w\#^n a^y\#) \cong U^2_m$$

The domain of the final forest $\mathcal{F}$ is the set

$$V = \{0,1\}^*1\# \ \cup \ \mathcal{F}'(\{0,1\}^*1\#^+a^+\#) \ \cup \ \{0,1\}^*1\#^+\mathcal{F}'(\alpha^+\#).$$

Let $\leq'$ be the order relation of $\mathcal{F}'$. The order relation $\leq$ of $\mathcal{F}$ is the reflexive and transitive closure of the relation

$$\preceq_V \ \cup \{(u, v) \mid u, v \in \mathcal{F}'(\{0,1\}^*1\#^+a^+\#), \ u \leq' v\} \ \cup$$
$$\{(wu, wv) \mid w \in \{0,1\}^*1\#^+, \ u, v \in \mathcal{F}'(\alpha^+\#), \ u \leq' v\}.$$

Hence, for all $w \in \{0,1\}^*1$, we have

$$\mathcal{F}(w\#) \cong \left\langle \biguplus \{\mathcal{F}'(w\#^n a^y\#) \mid n, y \in \mathbb{N}_+\} \uplus \biguplus \{\aleph_0 \cdot U^2_m \mid m \in \mathbb{N}_+\} \right\rangle .$$

From the definition of U_0 and U_1 and the properties of the forest $\mathcal{F}'$ it follows that for every $w \in \{0,1\}^*1$, $\mathcal{F}(w\#) \cong U_1$ if $w \in A$, and $\mathcal{F}(w\#) \cong U_0$ if $w \notin A$. Moreover, the construction of $\mathcal{F}$ from $\mathcal{F}' = \mathsf{forest}(\mathcal{B}')$ implies that we can construct from $\mathcal{B}'$ a finite automaton $\mathcal{B}$ with $\mathcal{F} = \mathsf{forest}(\mathcal{B})$. □

Remark 42. It is possible to generalize the proof of Lemma 41 to arbitrary levels of the arithmetical hierarchy. Hence, for every $n \geq 2$, there exist trees U_0, U_1 of height $n+1$ such that from a given index of a Σ^0_n-set $A \subseteq \{0,1\}^*1$ one can compute a finite automaton $\mathcal{B}$ such that $\mathcal{F} = \mathsf{forest}(\mathcal{B})$ is a forest of height $n+1$ with the following properties:

- The set of roots of $\mathcal{F}$ is $\{0,1\}^*1\#$.
- For every $w \in \{0,1\}^*1\#$, $\mathcal{F}(w) \cong U_1$ if $w \in A$, and $\mathcal{F}(w) \cong U_0$ if $w \notin A$.

This yields an alternative proof for the lower bound in Theorem 36(c). But only the (in fact more complicated) construction from Sections 4.2.1 and 4.2.2 yields the exact lower bounds stated in Theorem 36(a).

Also notice that the forests constructed in Sections 4.2.1 are not of the form $\mathsf{forest}(\mathcal{A})$ for a finite automaton $\mathcal{A}$. This is due to the children $\alpha^{m-x}e^x\$^+\#$ for a root $\alpha^m\#$ in U^i_m, see Figure 8.

In [26], the authors prove Σ_1^1-completeness for the isomorphism problem for automatic graphs. Their proof, as pointed out in [38], can be easily adapted to show the same lower bound for (non-well-founded) automatic *successor trees*. So, it is important to note that the following theorem refers to *order trees*, i.e., trees viewed as partial orders.

Proposition 43. *There exists an order tree T_{Good} such that the set of all finite automata $\mathcal{A}$ with $\mathsf{tree}(\mathcal{A}) \cong T$ is Σ_1^1-complete.*

Proof. The proof combines the ideas from [26] and Lemma 41. In [26], the authors prove Σ_1^1-completeness for the isomorphism problem for automatic graphs by a reduction from the isomorphism problem for computable trees. For the latter problem, a *computable tree* is a prefix-closed and decidable subset $T \subseteq (2\mathbb{N})^*$ together with the prefix order $\preceq$. Such a tree is represented by an index for T. The reason for restricting to prefix-closed subsets of $(2\mathbb{N})^*$ is the following: If $T_1, T_2 \subseteq (2\mathbb{N})^*$ are prefix-closed, then the trees $(T_1; \preceq)$ and $(T_2; \preceq)$ are isomorphic if and only if the structures $(\mathbb{N}^*; \preceq, T_1)$ and $(\mathbb{N}^*; \preceq, T_2)$ (where T_1 and T_2 are unary predicates) are isomorphic. The important point is that every node in $(\mathbb{N}^*; \preceq, T_i)$ has infinitely many children that do not belong to T_i.

In the following, we construct for any computable tree $T \subseteq \mathbb{N}^*$ an automatic order tree $\mathsf{aut}(T)$ such that for two computable trees T_1, T_2, we have $(T_1; \preceq) \cong (T_2; \preceq)$ (where $\preceq$ is the prefix relation) if and only if $\mathsf{aut}(T_1) \cong \mathsf{aut}(T_2)$. We start with the automatic presentation $(\{1\} \cup 1\{0,1\}^*1; \preceq)$ of the tree $(\mathbb{N}^*; \preceq)$. An isomorphism between these two trees is given by the computable mapping f with $f(n_1 n_2 \cdots n_k) = 10^{n_1}10^{n_2}1 \cdots 0^{n_k}1$ ($k \geq 0$).

Let us fix a computable tree $T \subseteq (2\mathbb{N})^*$. Since T and f are computable, the image $A = f(T) \subseteq \{1\} \cup 1\{0,1\}^*1 \subseteq \{0,1\}^*1$ is computable and hence a Σ_2^0-set. An index for A can be computed from an index for T. Now we apply Lemma 41 to the set A. We obtain a finite automaton $\mathcal{A}'$ (which can be constructed from an index for A) such that $\mathcal{F}' = \mathsf{forest}(\mathcal{A}')$ is a forest of height 3 with the following properties:

- The set of roots of $\mathcal{F}'$ is $\{0,1\}^*1\#$.
- For every $w \in \{0,1\}^*1$, the subtree rooted in $w\#$ is isomorphic to U_1 (resp. U_0) if $w \in A$ (resp. $w \notin A$).

In particular, any subtree rooted at $w \notin (\{1\} \cup 1\{0,1\}^*1)\#$ is isomorphic to U_0. Let $\mathcal{F}''$ be the subforest consisting of all trees of $\mathcal{F}'$ rooted at some word from $(\{1\} \cup 1\{0,1\}^*1)\#$. Since this language is regular, we can easily construct a finite automaton $\mathcal{A}''$ such that $\mathcal{F}'' = \mathsf{forest}(\mathcal{A}'')$.

Our automatic tree $\mathsf{aut}(T)$ results from the automatic forest $\mathcal{F}''$ by adding to the domain all nodes from $\{1\} \cup 1\{0,1\}^*1$ together with the prefix relation. Intuitively, the tree $\mathsf{aut}(T)$ results from the tree $(\mathbb{N}^*; \preceq)$ by appending to each node $x \in \mathbb{N}^*$ a copy of the tree U_1 (resp. U_0) if $x \in T$ (resp. $x \notin T$). Since $U_0 \ncong U_1$, it follows that $(T_1; \preceq) \cong (T_2; \preceq) \Leftrightarrow (\mathbb{N}^*; \preceq, T_1) \cong (\mathbb{N}^*; \preceq, T_2) \Leftrightarrow \mathsf{aut}(T_1) \cong \mathsf{aut}(T_2)$ for all computable trees $T_1, T_2 \subseteq (2\mathbb{N})^*$. Moreover, $\mathsf{aut}(T)$ is of the form $\mathsf{tree}(\mathcal{A})$ for a finite automaton $\mathcal{A}$ and $\mathcal{A}$ can be constructed effectively from an index for the computable tree T.

The statement of the theorem follows from the fact that there exists a computable tree T whose computable copies form a Σ_1^1-complete set [16]. Our previous construction transforms this tree T into a fixed tree $T_{\mathsf{Good}} = \mathsf{aut}(T)$, whose set of automatic presentations is Σ_1^1-complete. $\square$

As an immediate consequence, we obtain

Theorem 44. *The isomorphism problem for automatic order trees is Σ_1^1-complete.*

Recall that the Π_1^0-hardness of the isomorphism problem of automatic equivalence structures could be shown within a single automatic equivalence structure; see Proposition 11. Here, we obtain a similar result: On the set $\{0,1\}^* \cup (\{0,1\} \times \{0,1\})^*$ we define a partial order $\sqsubseteq$ as the reflexive and transitive closure of the following relation, where $u, u', v' \in \{0,1\}^*$ with $|u'| = |v'|$:

- $u \sqsubseteq u'$ if and only if u is a prefix of u'
- $u \sqsubseteq u' \otimes v'$ if and only if u is a prefix of u'

Then $(\{0,1\}^* \cup (\{0,1\} \times \{0,1\})^*; \sqsubseteq)$ is a tree, intuitively, it is obtained from the complete binary tree $(\{0,1\}^*; \preceq)$ by attaching $2^{|u|}$ new leaves to every node $u \in \{0,1\}^*$. For $L \subseteq \{0,1\}^* \cup (\{0,1\} \times \{0,1\})^*$ let us write $(L; \sqsubseteq)$ for $(L; \sqsubseteq \cap L^2)$.

Theorem 45. *There exists an order tree T_{Good} such that the set of all finite automata $\mathcal{B}$ with $(L(\mathcal{B}); \sqsubseteq) \cong T_{\mathsf{Good}}$ is Σ_1^1-complete.*

Proof. Let $\mathcal{A}$ be a finite automaton over the alphabet $\Sigma = \{b_1, \ldots, b_m\}$ whose set of transitions equals $\Delta = \{t_1, \ldots, t_n\}$, where w.l.o.g. $m \leq n$. Define $f(b_i) = 0^i 1^{n-i}$ and $f(t_j) = 0^i 1^{n-i} \otimes 0^j 1^{n-j}$, where b_i is the letter of the transition t_j, and extend this function f to a monoid homomorphism from $(\Sigma \cup \Delta)^*$ to $(\{0,1\} \cup \{0,1\} \times \{0,1\})^*$. Then

$$\mathsf{tree}(\mathcal{A}) \cong (f(L(\mathcal{A}) \uplus \mathsf{Run}(\mathcal{A}, \mathsf{leaves}(\mathcal{A}))); \sqsubseteq)$$

and an automaton $\mathcal{B}$ accepting $f(L(\mathcal{A}) \uplus \mathsf{Run}(\mathcal{A}, \mathsf{leaves}(\mathcal{A})))$ can be constructed from $\mathcal{A}$. Hence the claim follows from Proposition 43. ◻

4.5 Context-free languages with the prefix order

Given two regular languages L_1 and L_2 such that $\varepsilon \in L_1 \cap L_2$, it is decidable whether the trees $(L_1; \preceq)$ and $(L_2; \preceq)$ are isomorphic.[8] Recently, this problem was shown to be $\mathsf{EXPTIME}$-complete (resp., P-complete) for the case that L_1 and L_2 are given by nondeterministic (resp., deterministic) finite automata [31]. Ésik showed that the same problem is undecidable for context-free languages and asks whether it becomes decidable for deterministic context-free languages. Based on the construction from the previous section, we can show that it is Σ_1^1-complete in this case. Details on deterministic context-free languages and deterministic pushdown automata can be found in [21]. We use $L(P)$ to denote the language recognized by a pushdown automaton P.

Theorem 46. *There is a tree T such that the set of deterministic pushdown automata P with $(L(P); \preceq) \cong T$ is Σ_1^1-complete.*

Proof. By Proposition 43 it suffices to show that for every tree of the form $\mathsf{tree}(\mathcal{A})$ (where $\mathcal{A}$ is a finite automaton) there exists (effectively) a deterministic pushdown automaton P such that $\mathsf{tree}(\mathcal{A}) \cong (L(P); \preceq)$.

So, let $\mathcal{A}$ be a finite automaton over the alphabet Σ. For a word $w = a_1 \cdots a_k$, let $\overleftarrow{w}$ be its *reversal*, i.e., $\overleftarrow{w} = a_k \cdots a_1$. Then consider the language

$$K = L(\mathcal{A}) \uplus \{w\overleftarrow{r} \mid w \in \mathsf{leaves}(\mathcal{A}), r \in \mathsf{Run}(\mathcal{A}, w)\} \subseteq \Sigma^+ \Delta^+,$$

where Δ is the set of transitions of $\mathcal{A}$. One can easily construct a deterministic pushdown automaton P with $L(P) = K$. Moreover, the mapping $f : K \to L(\mathcal{A}) \uplus \mathsf{Run}(\mathcal{A}, \mathsf{leaves}(\mathcal{A}))$ with $f(w) = w$ for all $w \in L(\mathcal{A})$ and $f(w\overleftarrow{r}) = r$ for all $w \in \mathsf{leaves}(\mathcal{A})$ and all $r \in \mathsf{Run}(\mathcal{A}, w)$ is an isomorphism between the trees $\mathsf{tree}(\mathcal{A})$ and $(L(P); \preceq)$. ◻

5 Automatic Linear Orders

More details on linear orders can be found in [36]. We use ω to denote the linear order (type of) $(\mathbb{N}; \leq)$ of the natural numbers, ω^* for $(\{-n \mid n \in \mathbb{N}\}; \leq)$, ζ for $(\mathbb{Z}; \leq)$, and $\mathbf{n}$ for the finite linear order (type) of size n. Let $I = (D_I; \leq_I)$ be a linear order and, for $i \in D_I$, let $L_i = (D_i; \leq_i)$ be a linear order. The *sum* $\sum_{i \in I} L_i$ is the linear order $(\{(x,i) \mid i \in D_I, x \in D_i\}; \leq)$ where for all $i, j \in D_I$, $x \in D_i$, and $y \in D_j$,

$$(x,i) \leq (y,j) \iff i <_I j \vee (i = j \wedge x \leq_i y)\,.$$

We use $L_1 + L_2$ to denote $\sum_{i \in \mathbf{2}} L_i$ and $L_1 \cdot L_2$ to denote the sum $\sum_{i \in L_2} L_1^i$ where $L_1^i = L_1$ for every $i \in L_2$. An *interval* or *convex subset* of a linear order $L = (D; \leq)$ is a subset $I \subseteq D$ such that $x, y \in I$ and $x < z < y$ imply $z \in I$. For $x, y \in D$ we write (x, y) for the interval $\{z \in D \mid x < z < y\}$.

[8] The condition $\varepsilon \in L_1 \cap L_2$ is just a convenient way to ensure that $(L_1; \preceq)$ and $(L_2; \preceq)$ are indeed trees.

Lemma 47. *If* $0 < 1$*, then* $(\{0,1\}^*1; \leq_{\mathsf{lex}}) \cong (\mathbb{Q}; \leq)$.

Proof. Let $u' \in \{0,1\}^*$ and $v \in \{0,1\}^*1$ such that $u = u'1 <_{\mathsf{lex}} v$. Then

$$u'01 <_{\mathsf{lex}} u <_{\mathsf{lex}} u0^{|v|}1 <_{\mathsf{lex}} v <_{\mathsf{lex}} v01\,.$$

Note that indeed $u0^{|v|}1 <_{\mathsf{lex}} v$: if u is a prefix of v, then $u0^{|v|}1$ and v differ (for the first time) at some position in the block $0^{|v|}$ where v carries 1. If u is no prefix of v, then $u0^{|v|}1$ and v differ (for the first time) at some position in u where u carries 0 and v carries 1.

Hence $(\{0,1\}^*1; \leq_{\mathsf{lex}})$ is countable, dense, and without endpoints. Thus, by Cantor's theorem (see [18]) it is isomorphic to $(\mathbb{Q}; \leq)$. $\square$

The goal of this section is to prove that the isomorphism problem for automatic linear orders is Σ^1_1-complete. The general strategy of the proof is similar to the proof of Proposition 43 for automatic order trees. We will reduce the (Σ^1_1-complete) isomorphism problem for computable linear orders to the isomorphism problem for automatic linear orders. For this, we will need the following lemma:

Proposition 48. *From an index* e *of a computable linear order* L*, one can compute an index of a computable set* $P(e) \subseteq \{0,1\}^*1$ *whose complement is dense in* $(\{0,1\}^*1; \leq_{\mathsf{lex}})$ *such that*

$$L \cong L' \iff (\{0,1\}^*1; \leq_{\mathsf{lex}}, P(e)) \cong (\{0,1\}^*1; \leq_{\mathsf{lex}}, P(e'))$$

for all indices e *and* e' *of computable linear orders* L *and* L'*, resp.*

Proof. Let e be an index of a computable linear order $L = (D; \leq)$. Then the product linear order $(\{0,1\}^*1; \leq_{\mathsf{lex}}) \cdot L$ is isomorphic to $(\{0,1\}^*1; \leq_{\mathsf{lex}}) \cong (\mathbb{Q}; \leq)$ (since it is countable, dense, and without endpoints) and an index for $(\{0,1\}^*1; \leq_{\mathsf{lex}}) \cdot L$ can be computed from e. Next, we use the well-known computable variant of Cantor's theorem: If L_1 and L_2 are computable copies of $(\mathbb{Q}; \leq)$, then there exists a computable isomorphism $f : L_1 \to L_2$ and an index for f can be computed from indices for L_1 and L_2 [9]. Applied to our situation, this means that from e we can compute an index for a computable isomorphism $f_e : (\{0,1\}^*1; \leq_{\mathsf{lex}}) \cdot L \to (\{0,1\}^*1; \leq_{\mathsf{lex}})$. Since the complement of $\{1\} \times D$ is dense in $(\{0,1\}^*1; \leq_{\mathsf{lex}}) \cdot L$, so is the complement of $P(e) = f_e(\{1\} \times D)$ in $(\{0,1\}^*1; \leq_{\mathsf{lex}})$. Since f_e is computable, the set $P(e)$ is computable too and an index for $P(e)$ can be computed from the index e.

The implication "$\Leftarrow$" follows since $(P(e); \leq_{\mathsf{lex}}) \cong L$ for any index e of the computable linear order L. Conversely, let $L = (D; \leq)$ and $L' = (D'; \leq')$ with index e and e', resp. Assume that $L \cong L'$. Then we have

$$\begin{aligned}
(\{0,1\}^*1; \leq_{\mathsf{lex}}, P(e)) &\cong (\{0,1\}^*1 \times D; \sqsubseteq, \{1\} \times D) \\
&\cong (\{0,1\}^*1 \times D'; \sqsubseteq', \{1\} \times D') \text{ since } L \cong L' \\
&\cong (\{0,1\}^*1; \leq_{\mathsf{lex}}, P(e))
\end{aligned}$$

where $\sqsubseteq$ is the order of $(\{0,1\}^*1; \leq_{\mathsf{lex}}) \cdot L$ and similarly for $\sqsubseteq'$. $\square$

The following section will, from $P \subseteq \{0,1\}^*1$, construct a linear order $\mathsf{aut}(P)$ and prove that $(\{0,1\}^*1; \leq_{\mathsf{lex}}, P) \cong (\{0,1\}^*1; \leq_{\mathsf{lex}}, P')$ if and only if $\mathsf{aut}(P) \cong \mathsf{aut}(P')$ (assuming the complements of P and P' are dense). The subsequent section will then prove that $\mathsf{aut}(P)$ is effectively automatic for P computable.

5.1 Construction of linear orders

A key technique used in the construction is the shuffle sum of a class of linear orders. Let I be a countable set. A *dense* I*-coloring* of a linear order $L = (D; \leq)$ is a mapping $c : D \to I$ such that for all $x, z \in D$ with $x < z$ and all $i \in I$ there exists $y \in (x, z)$ with $c(y) = i$. Equivalently, $c^{-1}(i)$ is dense in $(D; \leq)$ for all $i \in I$.

Definition 49. *Let $\mathcal{L}$ be a countable set of linear orders and let $c : \mathbb{Q} \to \mathcal{L}$ be a dense $\mathcal{L}$-coloring of $\mathbb{Q}$. The shuffle sum of $\mathcal{L}$, denoted $\mathsf{Shuf}(\mathcal{L})$, is the linear order $\sum_{x\in(\mathbb{Q};\leq)} c(x)$.*

Extending the classical back-and-forth construction of isomorphisms, one obtains that $(\mathbb{Q}; \leq, c_1) \cong (\mathbb{Q}; \leq, c_2)$ for any two dense I-colorings c_1 and c_2 of $\mathbb{Q}$. Hence, in the above definition, the isomorphism type of $\sum_{x\in(\mathbb{Q};\leq)} c(x)$ does not depend on the choice of the dense $\mathcal{L}$-coloring c, see e.g. [36]. This implies that $\mathsf{Shuf}(\mathcal{L})$ is indeed uniquely defined.

For $m, n \in \mathbb{N}_+$, let $L[m, n]$ be the finite linear order with $C(m, n)$ elements (recall from (2) on page 9 that the polynomial function $C(x, y) = (x + y)^2 + 3x + y$ is injective). For $\kappa \in \mathbb{N}_+ \cup \{\omega\}$, we define the class of linear orders $\mathcal{L}'_\kappa$ and the linear order L'_κ as follows:

$$\mathcal{L}'_\kappa = \{L[m, n] \mid m, n \in \mathbb{N}_+, m \neq n \text{ or } m = n \geq \kappa\} \text{ and} \tag{10}$$
$$L'_\kappa = \mathsf{Shuf}(\mathcal{L}'_\kappa)\,. \tag{11}$$

Next, two linear orders M_0 and M_1 are given by

$$M_0 = \mathsf{Shuf}\{L'_\kappa \mid 2 \leq \kappa < \omega\} \text{ and}$$
$$M_1 = \mathsf{Shuf}\{L'_\kappa \mid 2 \leq \kappa \leq \omega\}\,.$$

Finally, let $P \subseteq \{0, 1\}^*1$. Then the linear order $\mathsf{aut}(P)$ is obtained from $(\{0, 1\}^*1; \leq_{\mathsf{lex}}, P)$ by replacing every element of P by M_1 and every element of $\{0, 1\}^*1 \setminus P$ by M_0:

$$\mathsf{aut}(P) = \sum_{x\in(\{0,1\}^*1;\leq_{\mathsf{lex}})} L^P_x \quad \text{with} \quad L^P_x = \begin{cases} M_1 & \text{if } x \in P \\ M_0 & \text{if } x \notin P \end{cases}$$

By the very definition, $(\{0, 1\}^*1; \leq_{\mathsf{lex}}, P) \cong (\{0, 1\}^*1; \leq_{\mathsf{lex}}, R)$ implies $\mathsf{aut}(P) \cong \mathsf{aut}(R)$. The following example shows that the converse is false in general.

Example 50. Let $P = \{1\}$ and let R be the open interval $(1, 11)$. Thus, $(\{0, 1\}^*1; \leq_{\mathsf{lex}}, P) \not\cong (\{0, 1\}^*1; \leq_{\mathsf{lex}}, R)$. Since $\sum_{x\in((1,11);\leq_{\mathsf{lex}})} L_x$ with $L_x = M_1$ for all $x \in (1, 11)$ is isomorphic to M_1, we get $\mathsf{aut}(P) \cong \mathsf{aut}(R)$.

Note that above the complement of P is dense, but the complement of R is not. The following lemmas prepare the proof that density of the complement is the only obstacle, see Proposition 55.

Lemma 51. *Let $\mathsf{Shuf}(\mathcal{L}) = \sum_{p\in(\mathbb{Q};\leq)} c(p)$ be a shuffle sum. For all $x \in \mathsf{Shuf}(\mathcal{L})$ there exists $y > x$ such that the interval (x, y) contains an interval isomorphic to $\mathsf{Shuf}(\mathcal{L})$.*

Proof. Let $x = (q, a)$ with $q \in \mathbb{Q}$ and $a \in c(q)$. Choose an arbitrary $r \in \mathbb{Q}$ with $r > q$. A direct back-and-forth argument shows $((q, r); \leq, c) \cong (\mathbb{Q}; \leq, c)$ and therefore $\sum_{p\in((q,r);\leq)} c(p) \cong \sum_{p\in(\mathbb{Q};\leq)} c(p) = \mathsf{Shuf}(\mathcal{L})$, which implies the lemma. □

Lemma 52. *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be countable sets of linear orders. Let $L = (D; \leq)$ be a linear order and I_1 and I_2 intervals of L with $I_1 \cong \mathsf{Shuf}(\mathcal{L}_1)$, $I_2 \cong \mathsf{Shuf}(\mathcal{L}_2)$, and $I_1 \cap I_2 \neq \emptyset$. Then $\mathsf{Shuf}(\mathcal{L}_1)$ is isomorphic to an interval of $\mathsf{Shuf}(\mathcal{L}_2)$ or vice versa.*

Proof. Let $x \in I_1 \cap I_2$. By Lemma 51 there exist $y_1 \in I_1$ and $y_2 \in I_2$ such that (x, y_i) contains an interval isomorphic to $\mathsf{Shuf}(\mathcal{L}_i)$ for $i \in \{1, 2\}$. W.l.o.g. assume that $y_1 \leq y_2$ in L. Then (x, y_1) is contained in $I_2 \cong \mathsf{Shuf}(\mathcal{L}_2)$, which proves the lemma. □

Lemma 53. *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two countable sets of finite linear orders.*

(1) Any infinite interval of $\mathsf{Shuf}(\mathcal{L}_1)$ contains an interval isomorphic to $\mathsf{Shuf}(\mathcal{L}_1)$.
(2) If $\mathsf{Shuf}(\mathcal{L}_1)$ is isomorphic to some interval of $\mathsf{Shuf}(\mathcal{L}_2)$, then $\mathcal{L}_1 = \mathcal{L}_2$ (up to isomorphism).
(3) Let $L = (D; \leq)$ be a linear order and I_1 and I_2 intervals of L with $I_1 \cong \mathsf{Shuf}(\mathcal{L}_1)$, $I_2 \cong \mathsf{Shuf}(\mathcal{L}_2)$, and $I_1 \cap I_2 \neq \emptyset$. Then $\mathcal{L}_1 = \mathcal{L}_2$ (up to isomorphism).

Proof. For $i \in \{1,2\}$, let $c_i : \mathbb{Q} \to \mathcal{L}_i$ be a dense $\mathcal{L}_i$-coloring. For claim (1), let I be some infinite interval in $\sum_{x\in(\mathbb{Q};\leq)} c_1(x)$. Since $c_1(x)$ is finite for all $x \in \mathbb{Q}$, there are $x, y \in \mathbb{Q}$ such that $x < y$ and both $c_1(x)$ and $c_1(y)$ intersect I. Hence $\sum_{z\in((x,y);\leq)} c_1(z)$ is an interval in I. A direct back-and-forth argument shows $((x,y);\leq,c_1) \cong (\mathbb{Q};\leq,c_1)$ and therefore $\sum_{z\in((x,y);\leq)} c_1(z) \cong \sum_{x\in(\mathbb{Q};\leq)} c_1(x) \cong \mathsf{Shuf}(\mathcal{L}_1)$.

For claim (2), suppose $\sum_{x\in(\mathbb{Q};\leq)} c_1(x)$ is isomorphic to some interval I of $\sum_{x\in(\mathbb{Q};\leq)} c_2(x)$. Then there is an embedding $f : \sum_{x\in(\mathbb{Q};\leq)} c_1(x) \to \sum_{x\in(\mathbb{Q};\leq)} c_2(x)$ whose range I is convex.

First let $L \in \mathcal{L}_1$. Since c_1 is surjective, there exists $x \in \mathbb{Q}$ with $L = c_1(x)$. Let $a \in L$ and $(y,b) = f(x,a)$ with $y \in \mathbb{Q}$. The maximal finite interval of $\sum_{x\in(\mathbb{Q};\leq)} c_1(x)$ containing (x,a) is isomorphic to $c_1(x) = L$. Since f is an embedding with convex range, the maximal finite interval of $\sum_{x\in(\mathbb{Q};\leq)} c_2(x)$ containing $f(x,a)$ is isomorphic to L as well. Since it contains (y,b), it is at the same time isomorphic to $c_2(y)$. Hence, indeed, there exists $L' \in \mathcal{L}_2$ with $L \cong L'$.

For the converse implication, note that by (1), $I \cong \mathsf{Shuf}(\mathcal{L}_1)$ contains an interval isomorphic to $\mathsf{Shuf}(\mathcal{L}_2)$. By symmetry, we therefore obtain from the previous paragraph that, for any $L' \in \mathcal{L}_2$, there exists $L \in \mathcal{L}_1$ with $L \cong L'$. This proves claim (2).

Finally, claim (3) follows from claim (2) and Lemma 52. ◻

Lemma 54. *The linear order M_1 is not isomorphic to any interval of the linear order M_0 and vice versa.*

Proof. Let $M_i = \sum_{x\in(\mathbb{Q};\leq)} c_i(x)$ for $i \in \{1,2\}$, where $c_0 : \mathbb{Q} \to \{L'_\kappa \mid 2 \leq \kappa < \omega\}$ and $c_1 : \mathbb{Q} \to \{L'_\kappa \mid 2 \leq \kappa \leq \omega\}$ are dense colorings. We start with the following claim:

Claim: Let $i \in \{0,1\}$, f be an embedding of M_i into M_{1-i}, and $(x,a) \in M_i$, $(y,b) = f(x,a)$ (with $x, y \in \mathbb{Q}$, $a \in c_i(x)$, $b \in c_{1-i}(y)$). Then $c_i(x) \cong c_{1-i}(y)$.

To see this, note that (y,b) belongs to an interval isomorphic to $c_{1-i}(y)$ by definition. Moreover, since f is an embedding with convex range, it also belongs to an interval isomorphic to $c_i(x)$. The claim follows from Lemma 53(3).

Let us now prove the lemma and assume first, towards a contradiction, that f is an embedding of M_1 into M_0 with convex range. Let $x \in \mathbb{Q}$ such that $c_1(x) = L'_\omega$. Chose an element $a \in L'_\omega$ and consider $(x,a) \in M_0$. Let $(x',a') = f(x,a) \in M_0$ with $x' \in \mathbb{Q}$. By the above claim, we get $c_0(x') \cong c_1(x) = L'_\omega$, which contradicts the fact that no order isomorphic to L'_ω belongs to the range of c_0.

For the other case, suppose that f is an embedding of M_0 into M_1 with convex range I. Choose $x, y \in \mathbb{Q}$ with $x < y$ and $c_0(x) \not\cong c_0(y)$ and let $a \in c_0(x)$, $b \in c_0(y)$. Let $(x',a') = f(x,a) \in I$ and $(y',b') = f(y,b) \in I$. Using our claim, we get $c_1(x') \cong c_0(x) \not\cong c_0(y) \cong c_1(y')$. Since $(x,a) < (y,b)$ in M_0, we must have $(x',a') < (y',b')$ in M_1. Because of $c_1(x') \not\cong c_1(y')$, we have $x' < y'$ in $\mathbb{Q}$. Take $z' \in \mathbb{Q}$ with $x' < z' < y'$ and $c' \in c_1(z') = L'_\omega$. Then $(z',c') \in I$, hence $(z,c) = f^{-1}(z',c')$ is defined. Our claim yields $c_0(z) \cong c_1(z') = L'_\omega$, which is a contradiction. ◻

Proposition 55. *Let the complements of $P, R \subseteq \{0,1\}^*1$ be dense in $(\{0,1\}^*1; \leq_{\mathsf{lex}})$. Then $\mathsf{aut}(P) \cong \mathsf{aut}(R)$ if and only if $(\{0,1\}^*1; \leq_{\mathsf{lex}}, P) \cong (\{0,1\}^*1; \leq_{\mathsf{lex}}, R)$.*

Proof. The if-direction is trivial since any isomorphism from $(\{0,1\}^*1; \leq_{\mathsf{lex}}, P)$ to $(\{0,1\}^*1; \leq_{\mathsf{lex}}, R)$ induces an isomorphism from $(P; \leq_{\mathsf{lex}})$ to $(R; \leq_{\mathsf{lex}})$. For the other implication, we show that the elements of P are in one-to-one correspondence with the maximal intervals in $\mathsf{aut}(P)$ of type M_1.

Let I be an interval of $\mathsf{aut}(P)$ with $I \cong M_1$ and suppose there is $x \in \{0,1\}^*1 \setminus P$ with $(\{x\} \times M_0) \cap I \neq \emptyset$. Hence an interval of type M_1 intersects an interval of type M_0. Lemma 52 implies that M_0 is isomorphic to an interval of type M_1 or vice versa, which leads to a contradiction by Lemma 54. Thus, $I \subseteq P \times M_1$.

Let $p \in P$. Then $\{p\} \times M_1$ is an interval in $\mathsf{aut}(P)$ of type M_1. Let $I \supsetneq \{p\} \times M_1$ be an interval of type M_1 properly larger than $\{p\} \times M_1$. Then there is $x \in \{0,1\}^*1$ with $p \neq x$ such that I contains an element of the form (x,u). Since the complement of P is dense in $(\{0,1\}^*1; \leq_{\mathsf{lex}})$, there is $y \in \{0,1\}^*1 \setminus P$ such that $p <_{\mathsf{lex}} y <_{\mathsf{lex}} x$ or $x <_{\mathsf{lex}} y <_{\mathsf{lex}} p$. Since I is an interval, we have

$(\{y\} \times M_0) \cap I \neq \emptyset$, contradicting the above observation $I \subseteq P \times M_1$. Hence, the maximal intervals in $\mathsf{aut}(P)$ of type M_1 are precisely the intervals of the form $\{p\} \times M_1$. Since the corresponding statement holds for the maximal intervals in $\mathsf{aut}(R)$ of type M_1, any isomorphism from $\mathsf{aut}(P)$ to $\mathsf{aut}(R)$ induces an isomorphism from $(P; \leq_{\mathsf{lex}})$ to $(R; \leq_{\mathsf{lex}})$. $\square$

Let e and e' be indices of computable linear orders L and L' and let $P(e)$ and $P(e')$ be the computable sets computed in Proposition 48. Since the complements of these sets are dense, we have $L \cong L'$ if and only if $\mathsf{aut}(P(e)) \cong \mathsf{aut}(P(e'))$. In the following section, we will prove that an automatic presentation of the linear order $\mathsf{aut}(P(e))$ can be computed from e.

5.2 Automaticity

In this section, let $P \subseteq \{0,1\}^*1$ be a Π^0_1-set with dense complement. We will effectively construct an automatic presentation of $\mathsf{aut}(P)$. The following definition will be useful: Let $\mathcal{A}$ be a finite automaton over the linearly ordered alphabet Σ (hence, a lexicographic order is available on Σ^*). Fix an arbitrary linear order on the set of transition tuples of $\mathcal{A}$, so that a lexicographic order is available on $\mathsf{Run}(\mathcal{A})$. Then we define the linear order

$$\mathsf{lin}(\mathcal{A}) = (\mathsf{Run}(\mathcal{A}); \leq),$$

where $r_1 \leq r_2$ for $r_1, r_2 \in \mathsf{Run}(\mathcal{A})$ if and only if

$$\mathsf{lab}(r_1) <_{\mathsf{lex}} \mathsf{lab}(r_2) \text{ or } (\mathsf{lab}(r_1) = \mathsf{lab}(r_2) \text{ and } r_1 \leq_{\mathsf{lex}} r_2).$$

Intuitively, we obtain $\mathsf{lin}(\mathcal{A})$ from $(L(\mathcal{A}); \leq_{\mathsf{lex}})$ by replacing every word $w \in L(\mathcal{A})$ by the finite linear order with $|\mathsf{Run}(\mathcal{A}, w)|$ many elements. Note that the isomorphism type of $\mathsf{lin}(\mathcal{A})$ does not depend on the chosen linear order on the set of transition tuples of $\mathcal{A}$. Clearly, $\mathsf{lin}(\mathcal{A})$ is automatic. In this section, we will realize $\mathsf{aut}(P)$ as $\mathsf{lin}(\mathcal{A})$ for a finite automaton $\mathcal{A}$.

Recall that for $u \in \{0,1\}^*1$, $\mathsf{num}(u) \in \mathbb{N}_+$ is the unique natural number such that u is the binary expansion of $\mathsf{num}(u)$ (least significant bit first). Since $P \in \Pi^0_1$, the set $\{\mathsf{num}(u) \mid u \in P\}$ is the complement of a recursively enumerable set. Hence (from an index of P), one can compute $\ell \in \mathbb{N}_+$ and two polynomials $p_1, p_2 \in \mathbb{N}[x, y_1, \ldots, y_\ell]$ such that for all $u \in \{0,1\}^*1$:

$$u \in P \iff \forall \overline{n} \in \mathbb{N}_+^\ell : p_1(\mathsf{num}(u), \overline{n}) \neq p_2(\mathsf{num}(u), \overline{n}) .$$

In the rest of this section, we fix the number ℓ and the polynomials p_1 and p_2. We define the two languages

$$D = (0^*1)^+ \text{ and } E = ((0^+1)^{\ell+1}\{a,b,c\})^+ .$$

On the alphabet $\{0, 1, a, b, c\}$ let us fix the order

$$0 < 1 < a < b < c,$$

which gives us a lexicographic order $\leq_{\mathsf{lex}}$ on $\{0,1,a,b,c\}^*$. For $u \in D$ let $\mathrm{col}_D(u) \in \mathbb{N}$ be the length of the last 0-block, i.e., $\mathrm{col}_D(u0^n1) = n$ for $u \in (0^*1)^*$. For $u \in ((0^+1)^{\ell+1}\{a,b,c\})^*$, $n_1, \ldots, n_{\ell+1} \in \mathbb{N}_+$, and $d \in \{a,b,c\}$ let $\mathrm{col}_E(u0^{n_1}10^{n_2}1 \cdots 0^{n_{\ell+1}}1d) = (n_1, \ldots, n_{\ell+1}, d)$.

Lemma 56. *The following holds:*

- $(D; \leq_{\mathsf{lex}}) \cong (\mathbb{Q}; \leq)$ *and* col_D *is a dense* $\mathbb{N}$*-coloring of* $(D; \leq_{\mathsf{lex}})$.
- $(E; \leq_{\mathsf{lex}}) \cong (\mathbb{Q}; \leq)$ *and* col_E *is a dense* $(\mathbb{N}_+^{\ell+1} \times \{a,b,c\})$*-coloring of* $(E; \leq_{\mathsf{lex}})$.

Proof. For the first statement, let $u, v \in D$ with $u'1 = u <_{\mathsf{lex}} v$ and let $n \in \mathbb{N}$. Then

$$u'01 <_{\mathsf{lex}} u <_{\mathsf{lex}} u0^{|v|}10^n1 <_{\mathsf{lex}} v <_{\mathsf{lex}} v01 .$$

Note that indeed $u0^{|v|}10^n1 <_{\mathsf{lex}} v$: if u is a prefix of v, then $u0^{|v|}10^n1$ and v differ (for the first time) at some position in the block $0^{|v|}$ where v carries 1. If u is no prefix of v, then $u0^{|v|}10^n1$ and v

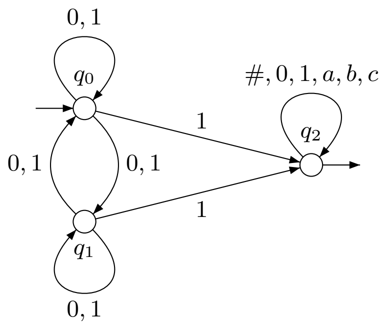


Fig. 11. The automaton $\mathcal{A}_x$

differ (for the first time) at some position in u where u carries 0 and v carries 1. Hence $(D; \leq_{\mathsf{lex}})$ is countable, dense and without endpoints and therefore isomorphic to $(\mathbb{Q}; \leq)$. Furthermore, $\mathrm{col}_D^{-1}(n)$ is dense for all $n \in \mathbb{N}$, i.e., col_D is indeed a dense $\mathbb{N}$-coloring.

For the second statement, let $u, v \in E$ with $u'1x_u = u <_{\mathsf{lex}} v = v'1x_v$ (where $x_u, x_v \in \{a, b, c\}$), $n_1, \ldots, n_{\ell+1} \in \mathbb{N}_+$, and $d \in \{a, b, c\}$. Then

$$u'01a <_{\mathsf{lex}} u <_{\mathsf{lex}} u(0^{|v|}1)^{\ell+1}a0^{n_1}10^{n_2}1\cdots 0^{n_{\ell+1}}1d <_{\mathsf{lex}} v <_{\mathsf{lex}} v(01)^{\ell+1}a\,.$$

Arguments analogous to those above show the claims regarding E and col_E. □

Lemma 57. *From a polynomial $p \in \mathbb{N}[x, y, z_1, \ldots, z_{\ell+1}]$, one can compute a nondeterministic finite automaton $\mathcal{A}_p$ with $L(\mathcal{A}_p) = \{0,1\}^*1\#D\#E$ that has precisely $p(\mathsf{num}(u), \mathrm{col}_D(v), \overline{n})$ accepting runs on $u\#v\#w \in \{0,1\}^*1\#D\#E$ with $\mathrm{col}_E(w) = (\overline{n}, d)$ for any $d \in \{a, b, c\}$.*

Proof. We proceed by induction on the construction of the polynomial p. First, consider the polynomial $p = x$ (for which the argument is very similar to the proof of Lemma 39). Let $\mathcal{A}_x$ be the automaton from Figure 11. Note that for each 1 preceeding any $\#, a, b, c$, the automaton can move from q_0 and q_1 resp. to q_2, then the rest of the input is processed deterministically. If $\mathcal{A}_x$ moves at input position k to the state q_2, then there are 2^{k-1} possible runs until reaching input position k. Hence, if $p_1 < p_2 < \cdots < p_n$ are the 1-positions in u ($p_1 \geq 1$, $p_n = |u|$) then $\mathcal{A}_x$ has $\sum_{i=1}^{n} 2^{p_i - 1} = \mathsf{num}(u)$ accepting runs on $u\#v\#w$.

Next consider the polynomial $p = y$ and let $\mathcal{A}_y$ be the automaton from Figure 12. Let $v \in D$. Since $0^{\mathrm{col}_D(v)}1$ is the longest suffix of v from 0^*1, there are precisely $\mathrm{col}_D(v)$ runs labeled v from q_1 to q_3. Since the rest of the automaton is deterministic, there are precisely $\mathrm{col}_D(v)$ many accepting runs on $u\#v\#w \in \{0,1\}^*1\#D\#E$.

Now consider the polynomial $p = z_i$ for some $1 \leq i \leq \ell + 1$. For $i = \ell$, the automaton $\mathcal{A}_{z_i}$ is depicted in Figure 13. Let $w \in E$ with $\mathrm{col}_E(w) = (n_1, \ldots, n_\ell, n_{\ell+1}, d)$. Then $0^{n_\ell}10^{n_{\ell+1}}1d$ is the longest suffix of w from $0^*10^*1\{a, b, c\}$. Hence there are precisely n_ℓ runs labeled w from q_0 to q_4. Since the rest of the word $u\#v\#w$ is processed deterministically, there are precisely n_ℓ many accepting runs on $u\#v\#w \in \{0,1\}^*1\#D\#E$.

The rest of the proof follows that of Lemma 9. □

Using the above lemma, one can compute a nondeterministic finite automaton $\mathcal{A}$ with

$$L(\mathcal{A}) = \{0,1\}^*1\#D\#E$$

such that for all $u \in \{0,1\}^*1$, $v \in D$, and $w \in E$ with $\mathrm{col}_E(w) = (\overline{n}, n_{\ell+1}, d)$ and $\overline{n} = (n_1, \ldots, n_\ell)$:

$$|\mathsf{Run}(\mathcal{A}, u\#v\#w)| = \begin{cases} C(p_1(\mathsf{num}(u), \overline{n}) + n_{\ell+1}, p_2(\mathsf{num}(u), \overline{n}) + n_{\ell+1}) & \text{if } d = a \text{ and } \mathrm{col}_D(v) = 0 \\ C(\mathrm{col}_D(v) + n_1, \mathrm{col}_D(v) + n_1) & \text{if } d = a \text{ and } \mathrm{col}_D(v) > 0 \\ C(n_1, n_1 + n_2) & \text{if } d = b \\ C(n_1 + n_2, n_1) & \text{if } d = c\,. \end{cases} \tag{12}$$

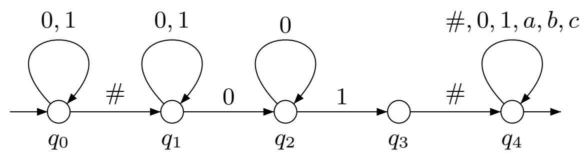


Fig. 12. The automaton $\mathcal{A}_y$

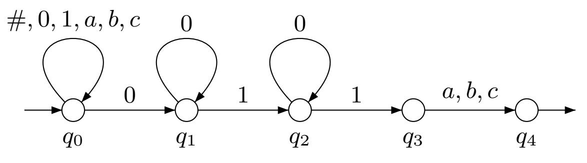


Fig. 13. The automaton $\mathcal{A}_{z_\ell}$

Recall the notations $\mathsf{Run}(\mathcal{A}, u)$, $\mathsf{Run}(\mathcal{A}, K)$, $\mathsf{Run}(\mathcal{A})$ and $\mathsf{lab}(r)$ from Section 2.3. Also recall the definition of the linear order $\mathsf{lin}(\mathcal{A})$ from the beginning of this section. Let $\mathsf{lin}(\mathcal{A}) = (\mathsf{Run}(\mathcal{A}); \sqsubseteq)$. We show in four steps that $\mathsf{lin}(\mathcal{A}) \cong \mathsf{aut}(P)$:

Step 1: Let $u\#v\#w \in \{0,1\}^*1\#D\#E$. Then the set $\mathsf{Run}(\mathcal{A}, u\#v\#w)$ of runs $r \in \mathsf{Run}(\mathcal{A})$ accepting $u\#v\#w$ is a finite interval in $\mathsf{lin}(\mathcal{A})$ whose size is given by (12).

Step 2: Let $u\#v \in \{0,1\}^*1\#D$. We analyze the restriction of the linear order $\mathsf{lin}(\mathcal{A})$ to the set $\mathsf{Run}(\mathcal{A}, u\#v\#E)$. Since $u\#v\#E$ is an interval in $(\{0,1\}^*1\#D\#E; \leq_{\mathsf{lex}})$, so is $\mathsf{Run}(\mathcal{A}, u\#v\#E)$ in $\mathsf{lin}(\mathcal{A})$. It is obtained from $(u\#v\#E; \leq_{\mathsf{lex}})$ by replacing $u\#v\#w$ by a linear order of size $|\mathsf{Run}(\mathcal{A}, u\#v\#w)|$ (which is given by (12)). The size of this linear order depends on $\mathrm{col}_E(w)$ and $\mathrm{col}_D(v)$ (but $\mathrm{col}_D(v) \in \mathbb{N}$ is constant since we fixed v). Hence, by Lemma 56, any size that appears at all appears densely, i.e., $(\mathsf{Run}(\mathcal{A}, u\#v\#E); \sqsubseteq)$ is the shuffle sum of the class of finite linear orders $\mathcal{L}_{u\#v} := \{\mathbf{n} \mid \exists w \in E : n = |\mathsf{Run}(\mathcal{A}, u\#v\#w)\}$. To determine this class precisely, we distinguish the cases $\mathrm{col}_D(v) = 0$ and $\mathrm{col}_D(v) > 0$.

If $\mathrm{col}_D(v) = 0$, then $\mathcal{L}_{u\#v}$ consists of the linear orders $L[p_1(\mathsf{num}(u), \overline{n}) + n_{\ell+1}, p_2(\mathsf{num}(u), \overline{n}) + n_{\ell+1}]$ (recall that $L[x, y]$ denotes the finite linear order with $C(x, y)$ many elements) for $\overline{n} \in \mathbb{N}_+^\ell$ and $n_{\ell+1} \in \mathbb{N}_+$ and the linear orders $L[m, n]$ for $m \neq n$. If $u \in P$, then $p_1(\mathsf{num}(u), \overline{n}) + n_{\ell+1} \neq p_2(\mathsf{num}(u), \overline{n}) + n_{\ell+1}$ for all values of $\overline{n}$ and $n_{\ell+1}$ and therefore $\mathcal{L}_{u\#v} = \mathcal{L}'_\omega$, where the set $\mathcal{L}'_\omega$ was defined in (10). If $u \notin P$, let $\kappa \in \mathbb{N}_+$ be minimal with $p_1(\mathsf{num}(u), \overline{n}) + n_{\ell+1} = \kappa = p_2(\mathsf{num}(u), \overline{n}) + n_{\ell+1})$ for some values of $\overline{n}$ and $n_{\ell+1}$. Then $\mathcal{L}_{u\#v} = \mathcal{L}'_\kappa$ and $\kappa \geq 2$ since it is the sum of two positive integers $p_1(\mathsf{num}(u), \overline{n})$ and $n_{\ell+1}$.

On the other hand, if $\mathrm{col}_D(v) > 0$, then the set $\mathcal{L}_{u\#v}$ consists of the finite linear orders $L[\mathrm{col}_D(v) + n, \mathrm{col}_D(v) + n]$ for $n \in \mathbb{N}_+$ and $L[m, n]$ for $m \neq n$. Hence $\mathcal{L}_{u\#v} = \mathcal{L}'_{\mathrm{col}_D(v)+1}$.

In summary,

$$(\mathsf{Run}(\mathcal{A}, u\#v\#E); \sqsubseteq) \cong \begin{cases} L'_\omega & \text{if } \mathrm{col}_D(v) = 0 \text{ and } u \in P \\ L'_\kappa & \text{for some } 2 \leq \kappa < \omega \text{ if } \mathrm{col}_D(v) = 0 \text{ and } u \notin P \\ L'_{c(v)+1} & \text{if } \mathrm{col}_D(v) > 0 \end{cases} \tag{13}$$

Step 3: Next, let $u \in \{0,1\}^*1$. We analyze the restriction of $\mathsf{lin}(\mathcal{A})$ to the set $\mathsf{Run}(\mathcal{A}, u\#D\#E)$. Since $u\#D\#E$ is an interval in $(\{0,1\}^*1\#D\#E; \leq_{\mathsf{lex}})$, so is $\mathsf{Run}(\mathcal{A}, u\#D\#E)$ in $\mathsf{lin}(\mathcal{A})$. It is obtained from $(u\#D; \leq_{\mathsf{lex}})$ by replacing $u\#v$ by the linear order $(\mathsf{Run}(\mathcal{A}, u\#v\#E); \sqsubseteq)$ whose type is given by (13). Since u is fixed, this type depends on $\mathrm{col}_D(v)$ only such that, by Lemma 56, any type that appears at all appears densely, i.e., $(\mathsf{Run}(\mathcal{A}, u\#D\#E); \sqsubseteq)$ is the shuffle sum of the class $\{\mathcal{L}'_\kappa \mid 2 \leq \kappa < \omega\}$ if $u \notin P$ and $\{\mathcal{L}'_\kappa \mid 2 \leq \kappa \leq \omega\}$ otherwise. Hence,

$$(\mathsf{Run}(\mathcal{A}, u\#D\#E); \sqsubseteq) \cong \begin{cases} M_1 & \text{if } u \in P \\ M_0 & \text{otherwise.} \end{cases} \tag{14}$$

Step 4: Finally, $\mathsf{lin}(\mathcal{A})$ is obtained from $(\{0,1\}^*1; \leq_{\mathsf{lex}})$ by replacing u by the finite linear order $(\mathsf{Run}(\mathcal{A}, u\#D\#E); \sqsubseteq)$ whose type is given by (14). Hence, indeed $\mathsf{lin}(\mathcal{A}) \cong \mathsf{aut}(P)$.

Thus, we proved the following statement:

Proposition 58. *From an index of a Π^0_1-set $P \subseteq \{0,1\}^*1$, one can compute a finite automaton $\mathcal{A}$ with $\mathsf{lin}(\mathcal{A}) \cong \mathsf{aut}(P)$.*

From Propositions 48, 55, 58, and the fact that the isomorphism problem for computable linear orders is Σ^1_1-complete, it follows that the isomorphism problem for automatic linear orders is Σ^1_1-complete as well. In fact, we can sharpen this result even to automatic linear orders of Hausdorff rank 1, which we define next.

For a linear order $L = (A; \leq)$ we define the equivalence relation $\equiv_L$ on A by: $a \equiv_L b$ if and only if the interval (a, b) is finite. Every equivalence class of $\equiv_L$ is an interval of L of order type $\mathbf{n}$ ($n \in \mathbb{N}$), ω, ω^*, or ζ. We define the *finite condensation* of L as the linear order $C(L)$, whose domain is the set of equivalence classes of $\equiv_L$ and $[a] \leq [b]$ if and only if $a \leq b$. In other words, $C(L)$ results from L by identifying two elements a, b of L if the interval (a, b) is finite. In [27] it was shown that for every automatic linear order L there exists a natural number n such that $C^n(L) = C^{n+1}(L)$; the least such n is called the *Hausdorff rank* of L. A linear order L has Hausdorff rank 1, if after identifying all $a, b \in L$ such that the interval (a, b) is finite, one obtains a dense order or the singleton linear order. The result of [27] mentioned above suggests that the isomorphism problem might be simpler for linear orders of low Hausdorff rank. But this is not the case:

Proposition 59. *There is a linear order L_{Good} of Hausdorff rank 1 such that the set of all finite automata $\mathcal{A}$ with $\mathsf{lin}(\mathcal{A}) \cong L_{\mathsf{Good}}$ is Σ^1_1-complete.*

Proof. There is a computable linear order, for which the set of all computable copies is Σ^1_1-complete. A concrete example is the Harrison order $\omega_1^{\mathrm{CK}}(1+\eta)$ (here ω_1^{CK} is the Church-Kleene ordinal and η is the order type of the rationals). Let h be an index of $\omega_1^{\mathrm{CK}}(1+\eta)$ and set $L = \mathsf{aut}(P(h))$. Note that the linear orders L'_κ from (11) have Hausdorff rank 1. Hence M_0 and M_1 are dense sums of linear orders of Hausdorff rank 1, i.e., their Hausdorff rank is 1 as well. Hence, L is a dense sum of orders of Hausdorff rank 1 as well, implying that the Hausdorff rank of L is 1 too.

We reduce the Σ^1_1-complete set of indices of $\omega_1^{\mathrm{CK}}(1+\eta)$ [16] to the set of all finite automata $\mathcal{A}$ with $\mathsf{lin}(\mathcal{A}) \cong L$: Let e be an index of some computable linear order K. By Proposition 48, we can compute an index for the set $P(e)$. From this index, we can compute a finite automaton $\mathcal{A}$ with $\mathsf{lin}(\mathcal{A}) \cong \mathsf{aut}(P(e))$ by Proposition 58. Then $K \cong \omega_1^{\mathrm{CK}}(1+\eta)$ if and only if $(\{0,1\}^*1; \leq, P(e)) \cong (\{0,1\}^*1; \leq, P(h))$ by Proposition 48 if and only if $\mathsf{aut}(P(e)) \cong \mathsf{aut}(P(h)) = L$ by Proposition 55. $\square$

As an immediate consequence, we obtain

Theorem 60. *The isomorphism problem of automatic linear orders is Σ^1_1-complete.*

As in the case of equivalence structures and of order trees, we can construct a single automatic linear order such that the whole complexity of the isomorphism problem can be found in this linear order. To this aim, let $\Sigma_2 = \{0,1\} \times \{0,1\}$ and set $u \otimes v \leq^2_{\mathsf{lex}} u' \otimes v'$ if

$$u <_{\mathsf{lex}} u' \text{ or } u = u' \text{ and } v \leq_{\mathsf{lex}} v' ,$$

where $|u| = |v|$ and $|u'| = |v'|$.[9] Now the following result can be shown as Theorem 45:

Theorem 61. *There exists a linear order L_{Good} such that the set of finite automata $\mathcal{B}$ with $(L(\mathcal{B}); \leq^2_{\mathsf{lex}}) \cong L_{\mathsf{Good}}$ is Σ^1_1-complete.*

Remark 62. The definition of $\leq^2_{\mathsf{lex}}$ can easily be extended to $\leq^n_{\mathsf{lex}}$ for $n \geq 3$, cf. [2]. We wonder whether a linear order L is automatic if and only if there exists $n \in \mathbb{N}$ and a regular language $K \subseteq (\{0,1\}^n)^*$ such that $L \cong (K; \leq^n_{\mathsf{lex}})$.

Finally, we can also show that the elementary equivalence problem for automatic linear orders is undecidable:

Theorem 63. *The set of automatic presentations of linear orders that are elementary equivalent to L'_ω from (11) is Π^0_1-complete. In particular, the elementary equivalence problem for automatic linear orders is Π^0_1-complete.*

Proof. Let $P \subseteq \{0,1\}^*1$ be Π^0_1-complete. For $u \in \{0,1\}^*1$, write $\mathsf{aut}(u)$ for the linear order $(\mathsf{Run}(\mathcal{A}, u\#1\#E); \sqsubseteq)$. By (13), this linear order is isomorphic to L'_ω if $u \in P$ and isomorphic to some L'_m for $m < \omega$ if $u \notin P$. Note that L'_m contains some maximal finite interval of size $C(m,m)$ and L'_ω does not. Hence, for every $m < \omega$ there is a first-order sentence that distinguishes L'_m and L'_ω. Thus, $\mathsf{aut}(u) \equiv L'_\omega$ if and only if $u \in P$. $\square$

5.3 Scattered linear orders

Recall that a linear order L is *scattered* if $(\mathbb{Q}; \leq)$ cannot be embedded into L. Typical examples of scattered linear orders are finite linear orders, ω, ω^*, and ζ. In [27] it was shown to be decidable, whether a given automatic linear order is scattered. In this section, we show that the isomorphism problem for scattered automatic linear orders is simpler than for general linear orders. An important tool in this proof are ordered trees (not to be confused with order trees). An *ordered tree* is a relational structure $T = (V; \leq, R)$, where $(V; \leq)$ is an order tree in the sense of Section 4 and the binary relation R is a disjoint union $\biguplus_{v \in V} \leq_v$, where $\leq_v$ is a linear order on the set of children of v. For a node v, let $E(v)$ be the set of children and write $R(v)$ for the linear order $(E(v); \leq_v)$. An ordered tree $T = (V; \leq, R)$ is *discrete* if, for every $v \in V$, the linear order $R(v)$ is finite or isomorphic to ω, to ω^*, or to ζ.

For $\ell \in \mathbb{N}$, let $\mathcal{O}_\ell$ denote the set of automatic presentations of discrete ordered trees of height at most ℓ and let $\mathcal{O} = \bigcup_{\ell \in \mathbb{N}} \mathcal{O}_\ell$. Note that one can axiomatize in the logic FSO the order type of ω (there are infinitely many elements but for every x the set of all y with $y < x$ is finite). Hence, one can axiomatize the order types ω^* and $\zeta = \omega^* + \omega$ as well. Theorem 5 implies that the class $\mathcal{O}_\ell$ is decidable for every $\ell \in \mathbb{N}$. By Theorem 13, even $\mathcal{O}$ is decidable.

Lemma 64. *The isomorphism problem for the class $\mathcal{O}$ of discrete automatic ordered trees of finite height can be reduced to $\mathsf{FOTh}(\mathbb{N}; +, \times)$.*

Proof. From $\mathcal{P}_1, \mathcal{P}_2 \in \mathcal{O}$, we can compute $\ell \in \mathbb{N}$ with $\mathcal{P}_1, \mathcal{P}_2 \in \mathcal{O}_\ell$. If $\ell = 0$, then $(\mathcal{P}_1, \mathcal{P}_2) \in \mathsf{Iso}(\mathcal{O})$. Now suppose $\ell > 0$ and let $T_i = (V_i; \leq_i, R_i)$ be the discrete ordered tree represented by $\mathcal{P}_i$. Let r_i be the root of T_i. For $v \in V_i$, let $\mathcal{P}_i(v)$ be an automatic presentation for the ordered subtree of T_i rooted at v (it can be computed from $\mathcal{P}_i$ and v). Furthermore, for $v \in V_i \setminus \{r_i\}$ let $v+1$ be the right sibling (i.e., the next element in the linear order $R_i(w)$ where w is the parent node of v) and $v-1$ the left sibling. Inductively, we define $v + (n+1) = (v+n)+1$ and $v - (n+1) = (v-n)-1$ for $n \in \mathbb{N}_+$. Note that $v+n$ and $v-n$ need not be defined, and that we can decide whether $v+n$ is defined since the tree T_i is automatic. To simplify notation in the formula below, let $\mathcal{P}_i(v+n)$ be an automatic presentation for the empty tree in case $v+n$ is not defined. Then we have $(\mathcal{P}_1, \mathcal{P}_2) \in \mathsf{Iso}(\mathcal{O})$ if and only if

$$(\mathcal{P}_1, \mathcal{P}_2) \in \mathsf{Iso}(\mathcal{O}_{\ell-1}) \ \vee\ \exists v_1 \in E(r_1)\ \exists v_2 \in E(r_2)\ \forall n \in \mathbb{Z} : (\mathcal{P}_1(v_1+n), \mathcal{P}_2(v_2+n)) \in \mathsf{Iso}(\mathcal{O}_{\ell-1})$$

By induction, this is a formula from $\Sigma^0_{2\ell}$. $\square$

[9] Bárány [2] uses a similar construction, but bases it on the length-lexicographic order instead of the lexicographic order.

Theorem 65. *The isomorphism problem for scattered automatic linear orders can be reduced to* $\mathsf{FOTh}(\mathbb{N}; +, \times)$.

Proof. We reduce this isomorphism problem to the isomorphism of automatic discrete ordered trees of finite height. Let $L = (A; \leq)$ be a scattered automatic linear order. Let n be the Hausdorff rank of L; it is finite by [27]. Since L is scattered, we have $C^n(L) \cong \mathbf{1}$. By induction on n, we define a discrete ordered tree T_L as follows: The leaves of T_L are the elements of L. If $n = 0$, then L is a linear order with a single element and T_L is a single node tree. Now assume that $n > 0$ and that the discrete ordered tree $T_{C(L)}$ is already defined. Then, the leaves of $T_{C(L)}$ are the equivalence classes w.r.t. $\equiv_L$. We obtain T_L be attaching to each equivalence class B the elements of B as new children and order them by $\leq$. Recall that the order type of L restricted to an equivalence classes w.r.t. $\equiv_L$ is indeed either finite, ω, ω^*, or ζ. Hence, T_L is a discrete ordered tree. Moreover, since L is automatic, the ordered tree T_L is effectively automatic since the equivalence relation $\equiv_L$ is definable in FSO. Finally, two scattered automatic linear orders L_1 and L_2 are isomorphic if and only if $T_{L_1} \cong T_{L_2}$. □

While the above theorem shows that the isomorphism problem for scattered linear orders is substantially simpler than for arbitrary linear orders, we still do not have any lower bound. We do not even know whether this problem is decidable or not.

5.4 Context-free languages with lexicographic orders

Given two regular languages L_1 and L_2 and a linear order on their alphabet, it is decidable whether $(L_1; \leq_{\mathsf{lex}}) \cong (L_2; \leq_{\mathsf{lex}})$ [41]. For context-free languages, the same problem is undecidable [13] and Ésik asks whether the problem becomes decidable for deterministic context-free languages. In the light of the current paper, it is natural to ask for the exact recursion-theoretic level of undecidability. In this section, we show that it is Σ^1_1-complete for deterministic context-free languages.

The class of linear orders that can be realized as the lexicographic order on a deterministic context-free language coincides with the class of *algebraic linear orders* [4]. An algebraic linear order is a component of the initial solution of a first-order recursion scheme over the continuous categorical algebra of countable linear orders equipped with the sum operation and the one-element linear order as a constant. Hence, every algebraic linear order L can be represented by a deterministic pushdown automaton P recognizing L.

Theorem 66. *There is a linear order L of Hausdorff rank 1 such that the set of deterministic pushdown automata P with $(L(P); \leq_{\mathsf{lex}}) \cong L$ is Σ^1_1-complete. In particular, the isomorphism problem for the class of algebraic linear orders is Σ^1_1-complete.*

Proof. By Theorem 60, it suffices to construct from a given finite automaton $\mathcal{A}$ a deterministic pushdown automaton P such that $\mathsf{lin}(\mathcal{A}) \cong (L(P); \leq_{\mathsf{lex}})$. The construction is very similar to those in the proof of Theorem 46.

Let $\mathcal{A}$ be a finite automaton over the alphabet Σ. Recall from the proof of Theorem 46 that $\overleftarrow{w}$ denotes the reversal of the word w. Then consider the language

$$K = \{w\overleftarrow{r} \mid w \in L(\mathcal{A}), r \in \mathsf{Run}(\mathcal{A}, w)\} \subseteq \Sigma^+\Delta^+$$

where Δ is the set of transitions of $\mathcal{A}$. One can easily construct a deterministic pushdown automaton P with $L(P) = K$. On $\Sigma \cup \Delta$ we fix an order with $\delta < a$ for all $\delta \in \Delta$ and $a \in \Sigma$. For any $w \in L(\mathcal{A})$, we have

$$(\mathsf{Run}(\mathcal{A}, w); \sqsubseteq) \cong \mathbf{n} \cong (\{\overleftarrow{r} \mid r \in \mathsf{Run}(\mathcal{A}, w)\}; \leq_{\mathsf{lex}})$$

where n is the number of accepting runs of $\mathcal{A}$ on w. Moreover, since every symbol from Δ is smaller than every symbol from Σ, we have the following for all $w_1, w_2 \in L(\mathcal{A})$ and $r_1 \in \mathsf{Run}(\mathcal{A}, w_1)$, $r_2 \in \mathsf{Run}(\mathcal{A}, w_2)$ with $w_1 \neq w_2$: $w_1\overleftarrow{r_1} \leq_{\mathsf{lex}} w_2\overleftarrow{r_2}$ if and only if $w_1 \leq_{\mathsf{lex}} w_2$. Hence, $(L(P); \leq_{\mathsf{lex}})$ can be obtained from $(L(\mathcal{A}); \leq_{\mathsf{lex}})$ by replacing every word w by the finite linear order with $|\mathsf{Run}(\mathcal{A}, w)|$ many elements. Thus, we have $(K; \leq_{\mathsf{lex}}) \cong \mathsf{lin}(\mathcal{A})$. □

6 Complexity of isomorphisms between automatic structures

We conclude this paper with an application of Theorem 44 and 60. The following corollary shows that although automatic structures look simple (especially for automatic trees), there may be no "simple" isomorphism between two automatic copies of the same structure. Recall that the class of hyperarithmetical sets is $\Sigma^1_1 \cap \Pi^1_1$, where Π^1_1 is the set of complements of Σ^1_1-sets. This class is stratified into levels Δ^0_α for each computable ordinal α (the levels $\Delta^0_n = \Sigma^0_n \cap \Pi^0_n$ for $n \in \mathbb{N}$ constitute an initial part of this hierarchy), see [1,35] for more details. These levels can be defined by computable infinitary formulas over $(\mathbb{N}; +, \times)$ of certain ranks [1]. An isomorphism f between two automatic structures with domains L_1 and L_2, respectively, is a Δ^0_α-isomorphism (resp. hyperarithmetical isomorphism), if the set $\{(x, f(x)) \mid x \in L_1\}$ belongs to Δ^0_α (resp., is hyperarithmetical).

Corollary 67. *There exist two isomorphic automatic order trees (and two isomorphic automatic linear orders) without a hyperarithmetical isomorphism.*

Proof. We only prove the corollary for automatic order trees, the same proof works for automatic linear orders (using Theorem 60). By Proposition 43, there exists an automatic order tree T such that the set of all automatic copies of T is Σ^1_1-complete and hence not hyperarithmetical. It follows that for each level Δ^0_α of the hyperarithmetical hierarchy there exist two copies of T without a Δ^0_α-isomorphism: To see this, assume that for any two automatic copies T_1 and T_2 of T there exists a Δ^0_α-isomorphism. Hence, for an automatic presentation $\mathcal{P}$ of an order tree we have $T \cong \mathcal{S}(\mathcal{P})$ if and only if there exists a Δ^0_α-isomorphism between T and $\mathcal{S}(\mathcal{P})$. But this is a hyperarithmetical statement (one has to state that there exists an index for an infinitary computable formula, which defines a Δ^0_α-isomorphism between T and $\mathcal{S}(\mathcal{P})$). Hence, the set of all automatic copies of T is hyperarithmetical, which contradicts the fact that this set is Σ^1_1-complete.

Now, we can transfer the proof of [1, Theorem 8.20] from computable structures to automatic structures.[10] The proof is exactly the same. One only has to change the computable infinitary sentence ϕ in the proof of [1, Theorem 8.18] so that the models of ϕ are exactly all automatic linear orders (instead of all computable structures) with elements named by constants from a fixed set B. □

7 Conclusion

This paper looks at the isomorphism problem of some typical classes of automatic structures. Such classes include equivalence structures, order trees, and linear orders. We have shown that (i) the isomorphism problem for automatic equivalence structures is Π^0_1-complete and (ii) the isomorphism problem for automatic order trees and linear orders is Σ^1_1-complete. For order trees, we proved better complexity bounds under certain restrictions. For instance, we have shown that the isomorphism problem for automatic well-founded order trees and automatic trees of bounded height is recursively equivalent to first-order arithmetic. For automatic trees of height $n \geq 2$, the isomorphism problem turned out to be Π^0_{2n-3}-complete. We also showed that the isomorphism problem for scattered linear orders can be reduced to true arithmetic, but any lower bound for this problem is missing.

Acknowledgments. We would like to thank Alexander Kartzow for valuable comments.

References

1. C. J. 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.

[10] [1, Theorem 8.20] states that if $\mathcal{S}$ is a computable structure such that for every level Δ^0_α of the hyperarithmetical hierarchy there exist two copies of $\mathcal{S}$ without a Δ^0_α-isomorphism, then there exist two copies of $\mathcal{S}$ without a hyperarithmetical isomorphism.

2. V. Bárány. A hierarchy of automatic omegawords having a decidable mso theory. *RAIRO Theor. Inform. Appl.*, 42(3):417–450, 2008.
3. V. Bárány, L. Kaiser, and S. Rubin. Cardinality and counting quantifiers on omega-automatic structures. In *Proceedings of STACS 2008*, pages 385–396. IFIB Schloss Dagstuhl, 2008.
4. S. L. Bloom and Z. Ésik. Algebraic linear orderings. *Internat. J. Found. Comput. Sci.*, 22(2):491–515, 2011.
5. A. Blumensath and E. Grädel. Automatic structures. In *Proceedings of LICS 2000*, pages 51–62. IEEE Computer Society Press, 2000.
6. A. Blumensath and E. Grädel. Finite presentations of infinite structures: Automata and interpretations. *Theory Comput. Syst.*, 37(6):641–674, 2004.
7. W. Calvert, D. Cenzer, V. Harizanov, and A. Morozov. Effective categoricity of equivalence structures. *Ann. Pure Appl. Logic*, 141:61-78, 2006.
8. W. Calvert and J. F. Knight. Classification from a computable viewpoint. *Bull. Symbolic Logic*, 12(2):191–218, 2006.
9. D. Cenzer and J. B. Remmel. Complexity Theoretic Model Theory and Algebra. In Y. L. Ershov, S. S. Goncharov, V. Marek, A. Nerode and J. Remmel, editors, *Handbook of Recursive Mathematics, Vol 1*, pages 381–513. Elsevier, 1998.
10. Ch. Delhommé. Non-automaticity of ω^ω. 2001. Manuscript.
11. Ch. Delhommé. Automaticité des ordinaux et des graphes homogènes. *C. R. Acad. Sci. Paris, Ser. I*, 339:5–10, 2004.
12. C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Am. Math. Soc.*, 98:21–51, 1961.
13. Z. Ésik. An undecidable property of context-free linear orders. *Inform. Process. Lett.*, 111(3):107–109, 2011.
14. D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, and W. P. Thurston. *Word processing in groups.* Jones and Bartlett, Boston, 1992.
15. Y. L. Ershov, S. S. Goncharov, V. W. Marek, A. Nerode and J. Remmel. *Handbook of Recursive Mathematics: Volume 1,2.* Elsevier, 1998.
16. S. S. Goncharov and J. F. Knight. Computable structure and antistructure theorems. *Algebra i Logika*, 41(6):639–681, 2002.
17. D. R. Hirschfeldt and W. M. White. Realizing levels of the hyperarithmetic hierarchy as degree spectra of relations on computable structures. *Notre Dame J. Form. Log.*, 43(1):51–64 (2003), 2002.
18. W. Hodges. *Model Theory.* Cambridge University Press, 1993.
19. B. R. Hodgson. On direct products of automaton decidable theories. *Theoret. Comput. Sci.*, 19:331–335, 1982.
20. J. Honkala. On the problem whether the image of an N-rational series equals N. *Fund. Inform.*, 73(1-2):127–132, 2006.
21. J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages and computation.* Addison–Wesley, Reading, MA, 1979.
22. H. Ishihara, B. Khoussainov, and S. Rubin. Some results on automatic structures. In *Proceedings of LICS 2002*, pages 235–244. IEEE Computer Society Press, 2002.
23. B. Khoussainov and M. Minnes. Model theoretic complexity of automatic structures. In *Proceedings of TAMC 2008*, number 4978 in Lecture Notes in Computer Science, pages 514–525. Springer, 2008.
24. B. Khoussainov and A. Nerode. Automatic presentations of structures. In *LCC: International Workshop on Logic and Computational Complexity*, number 960 in Lecture Notes in Computer Science, pages 367–392, 1995.
25. B. Khoussainov and A. Nerode. Open questions in the theory of automatic structures. *Bulletin of the EATCS*, 94, pages 181–204, 2008.
26. B. Khoussainov, A. Nies, S. Rubin, and F. Stephan. Automatic structures: richness and limitations. *Log. Methods Comput. Sci.*, 3(2):2:2, 18 pp. (electronic), 2007.
27. B. Khoussainov, S. Rubin, and F. Stephan. Automatic linear orders and trees. *ACM Trans. Comput. Log.*, 6(4):675–700, 2005.
28. D. Kuske. Where automatic structures benefit from weighted automata. In *Bozapalidis Festschrift*, number 7020 in Lecture Notes in Computer Science, pages 257–271. Springer, 2011.
29. D. Kuske, J. Liu, and M. Lohrey. The isomorphism problem for ω-automatic trees. In *Proceedings of CSL 2009*, number 6247 in Lecture Notes in Computer Science, pages 396–410. Springer, 2010.
30. D. Kuske and M. Lohrey. Some natural decision problems in automatic graphs. *J. Symbolic Logic*, 75(2):678–710, 2010.

31. M. Lohrey and C. Mathissen. Isomorphism of regular trees and words. In *Proceedings of ICALP 2011*, number 6756 in Lecture Notes in Computer Science, pages 210–221. Springer, 2011.
32. J. Liu and M. Minnes. Analysing Complexity in Classes of Unary Automatic Structures. In *Proceedings of LATA 2009*, number 5457 in Lecture Notes in Computer Science, pages 514–525. Springer, 2009.
33. Y. V. Matiyasevich. *Hilbert's Tenth Problem.* MIT Press, Cambridge, Massachusetts, 1993.
34. A. Nies. Describing groups. *Bull. Symbolic Logic*, 13(3):305–339, 2007.
35. H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1968.
36. J. Rosenstein. *Linear Ordering.* Academic Press, 1982.
37. S. Rubin. *Automatic Structures.* PhD thesis, University of Auckland, 2004.
38. S. Rubin. Automata presenting structures: A survey of the finite string case. *Bull. Symbolic Logic*, 14:169–209, 2008.
39. A. Salomaa and M. Soittola. *Automata-theoretic aspects of formal power series.* Springer, 1978.
40. R. I. Soare. *Recursively enumerable sets and degrees.* Perspectives in Mathematical Logic. Springer, 1987.
41. W. Thomas. On frontiers of regular trees. *RAIRO Theor. Inform. Appl.*, 20:371–381, 1986.
42. T. Tsankov. The additive group of the rationals does not have an automatic presentation. `http://arxiv.org/abs/0905.1505`.
43. A. Weber. On the valuedness of finite transducers. *Acta Informat.*, 27:749–780, 1990.