The Isomorphism Problem On Classes of Automatic Structures

Dietrich Kuske
LaBRI, CNRS, Université Bordeaux I
kuske@labri.fr

Jiamou Liu, Markus Lohrey∗
Universität Leipzig, Institut für Informatik
{jiamou,lohrey}@informatik.uni-leipzig.de

Abstract

Several new undecidability results on isomorphism problems for automatic structures are shown: (i) The isomorphism problem for automatic equivalence relations is \(\Pi_0^1\)-complete. (ii) The isomorphism problem for automatic trees of height \(n \geq 2\) is \(\Pi_{2n-3}^0\)-complete. (iii) The isomorphism problem for automatic linear orders is not arithmetical.

1 Introduction

The idea of an automatic structure goes back to Büchi and Elgot who used finite automata to decide, e.g., Presburger arithmetic [4]. Automaton decidable theories [7] and automatic groups [5] are similar concepts. A systematic study was initiated by Khoussainov and Nerode [10] who also coined the name “automatic structure”. In essence, a structure is automatic if the elements of the universe can be represented as 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 [1, 2, 12, 13, 14, 20]. 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 structures. A structure is computable, if its domain as well as all relations are recursive sets of finite words (or naturals). A well-studied problem for computable structures is the isomorphism problem, where it is asked 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 \(\Sigma_1^1\). In fact, \(\Sigma_1^1\)-completeness holds for many subclasses of computable structures, e.g.,

for linear orders, trees, undirected graphs, Boolean algebras, Abelian \(p\)-groups, see [3, 6]. \(\Sigma_1^1\)-completeness of the isomorphism problem for a class of computable structures implies non-existence of a good classification (in the sense of [3]) for that class.

In [12], it was shown that also for automatic structures the isomorphism problem is \(\Sigma_1^1\)-complete. By a direct interpretation, it follows that for the following classes the isomorphism problem is still \(\Sigma_1^1\)-complete [18]: automatic successor trees, automatic undirected graphs, automatic commutative monoids, automatic partial orders, automatic lattices of height 4, and automatic 1-ary functions. On the other hand, the isomorphism problem is decidable for automatic ordinals [13] and automatic Boolean algebras [12]. An intermediate class is the class of all locally-finite automatic graphs, for which the isomorphism problem is complete for \(\Pi_3^0\) (third level of the arithmetical hierarchy) [19].

For many interesting classes of automatic structures, the exact status of the isomorphism problem is open. In the recent papers [11, 20] it was asked for instance, whether the isomorphism problem is decidable for automatic equivalence relations and automatic linear orders. For the latter class, this question was already asked in [13]. In this paper, we answer these questions. Our main results are:

(i) The isomorphism problem for automatic equivalence relations is \(\Pi_0^1\)-complete.

(ii) The isomorphism problem for automatic (successor or order) trees of finite height \(n \geq 2\) (where the height of a tree is the maximal number of edges along a path from the root to a leaf) is \(\Pi_{2n-3}^0\)-complete.

(iii) The isomorphism problem for automatic linear orders is hard for every level of the arithmetical hierarchy.

Most hardness proofs for automatic structures, in particular the \(\Sigma_1^1\)-hardness proof for the isomorphism problem of automatic structures from [12], use transition graphs of Turing-machines (these graphs are easily seen to be automatic). This technique seems to fail for inherent reasons, when trying to prove our new results. The reason∗

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

†For background on the arithmetical hierarchy see, e.g., [21].
is most obvious for equivalence relations and linear orders. These structures are transitive but the transitive closure of
the transition graph of a Turing-machine cannot be automatic in general since its first-order theory can be undecidable.
Hence, we have to use a new strategy that is based on
Hilbert’s 10th problem. Recall that Matiyasevich proved
that every recursively enumerable set of natural numbers is
Diophantine [17]. This fact was used by Honkala to show
that the isomorphism problem of automatic trees of height
belongs to \( \Pi^0_3 \)-hard, and an inductive argument allows to similarly
construct trees \( U_n \) of height \( n \) whose sets of automatic
presentations are \( \Pi^0_{2n-3} \)-hard. Since we can also show that the
isomorphism problem of automatic trees of height \( n \)
\[ \{0,1\} \] is isomorphic to a structure whose set of automatic
presentations is \( \Pi^0_2 \)-hard. Together with containment in \( \Pi^0_i \) (as already observed in [20]), (ii) follows. Finally,
using a similar but technically more involved reduction, we
construct linear orders \( K_n \) whose set of automatic presentations
is hard for \( \Sigma^0_{i+1} \). Then (iii) follows. In fact, since
our proof is uniform on the levels in the arithmetical hierarchy,
the isomorphism problem for automatic linear orders is at least as hard as true arithmetic, i.e.,
the first-order theory of \((\mathbb{N};+,\times)\). At the moment it remains open whether the isomorphism problem for automatic linear orders is \( \Sigma^1_2 \)-complete.

A complete version of this extended abstract can be
found in [15].

2 Preliminaries

Let \( \mathbb{N}_+ = \mathbb{N} \setminus \{0\} \). Let \( p(x_1, \ldots, x_n) \in \mathbb{N}[x_1, \ldots, x_n] \) be a polynomial with coefficients in \( \mathbb{N} \). Define \( \text{Im}_{\mathbb{N}_+}(p) = \{ p(\bar{x}) \mid \bar{x} \in \mathbb{N}_+^n \} \). If \( p \neq 0 \), then \( \text{Im}_{\mathbb{N}_+}(p) \subseteq \mathbb{N}_+ \).

Details on the arithmetical hierarchy can be found for instance in [21]. With \( \Sigma^0_i \) we denote the \( i \)-th (existential) level of the arithmetical hierarchy; it is the class of all \( A \subseteq \mathbb{N} \) such that there exists a recursive predicate \( P \subseteq \mathbb{N}^{n+1} \) with \( A = \{ a \in \mathbb{N} \mid \exists x_1 \forall x_2 \ldots \forall x_n : (a, x_1, \ldots, x_n) \in P \} \),
where \( Q = \exists \) (\( Q = \forall \)) for odd (even). The set of complements of \( \Sigma^0_i \)-sets is denoted by \( \Pi^0_i \). By fixing some effective encoding of strings by natural numbers, we can talk about \( \Sigma^0_n \)-sets and \( \Pi^0_n \)-sets of strings over an arbitrary alphabet.
A typical example of a set, which does not belong to the arithmetical hierarchy is the first-order theory of \( (\mathbb{N};+,\times) \), which we denote by \( \text{FOTh}(\mathbb{N};+,\times) \).

We assume basic terminologies and notations from automata theory. For a fixed alphabet \( \Sigma \), a non-deterministic
finite automaton (NFA) is a tuple \( A = (S, \Delta, I, F) \) where \( S \) is the 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. A run of \( A \) on a word
is a sequence \( u = a_1a_2\cdots a_n (a_1, a_2, \ldots, a_n \in \Sigma) \) is a word over \( \Delta \) of the form \( r = (q_0, a_1, q_1)(q_1, a_2, q_2)\cdots(q_{n-1}, a_n, q_n) \), where \( q_0 \in I \). If moreover \( q_n \in F \), then \( r \) is an accepting run of \( A \) on \( u \). We will only apply these definitions in case \( n > 0 \), i.e., we will only speak of (accepting) runs on non-empty words.

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 \( \Sigma_o = \Sigma \cup \{\emptyset\} \) where \( \emptyset \notin \Sigma \). For words \( w_1, w_2, \ldots, w_n \in \Sigma^* \), their convolution is a word \( w_1 \otimes \cdots \otimes w_n \in (\Sigma_0^*)^n \) with 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 \( \sigma_i \) is the \( k \)-th symbol of \( w_i \) if \( k \leq |w_i| \), and \( \sigma_i = \emptyset \) otherwise. An \( n \)-ary relation \( R \) is \textit{FA recognizable} if the set of all convolutions of tuples \( (w_1, \ldots, w_n) \in R \) is a regular language.

A relational structure \( S \) consists of a domain \( D \) and
atomic relations on the set \( D \). We will only consider structures with countable domain. For a set \( \{S_i \mid i \in I\} \) of
relational structures over the same signature, we denote with \( \bigcup \{S_i \mid i \in I\} \) the disjoint union of these structures. With \( S_1 \uplus S_2 \) we denote the disjoint union of two structures \( S_1, S_2 \). A structure \( S \) is called \textit{automatic} over \( \Sigma \) if its domain is a regular subset of \( \Sigma^* \) and each of its atomic relations is \textit{FA recognizable}; any tuple \( P \) of automata that accept the
domain and the relations of \( S \) is called an \textit{automatic presentation of \( S \)}; in this case, we write \( S(P) \) for \( S \). If an
automatic structure \( S \) is isomorphic to a structure \( S' \), then \( S \) is called an \textit{automatic copy of \( S' \) and \( S' \) is automatically presentable}. In this paper we sometimes abuse the terminology referring to \( S' \) as simply automatic and calling an automatic presentation of \( S \) also automatic presentation of \( S' \). We also simplify our statements by saying “given/compute an automatic structure \( S' \)” for “given/compute an automatic presentation \( P \) of a structure \( S(P) \)”. The structures \( (\mathbb{N};\leq) \) and \( (\mathbb{N};\leq) \) are both automatic. On the other hand, \( (\mathbb{N};\times) \) and \( (\mathbb{N};\geq) \) have no automatic copies (see [9, 20] and [22]).

Let \( \text{FO} + \exists \forall \) be first-order logic extended by the quantifier \( \exists \forall \) (there exist infinitely many). The following theorem (see [20] for references and generalizations) lays out the main motivation for investigating automatic structures.

**Theorem 2.1** From an automatic presentation \( P \) and a formula \( \varphi(x) \in \text{FO} + \exists \forall \) in the signature of \( S(P) \), one can compute an NFA whose language consists of those tuples \( \bar{a} \) from \( S(P) \) that make \( \varphi \) true. In particular, the \( \text{FO} + \exists \forall \) theory of any automatic structure \( S \) is (uniformly) decidable.

Let \( K \) be a class of automatic structures closed under isomorphism. The isomorphism problem for \( K \) is the set of pairs \( (P_1, P_2) \) of automatic presentations with \( S(P_1) \cong S(P_2) \in K \). The isomorphism problem for the class of
all automatic structures is complete for $\Sigma^1_1$ — the first level of the analytical hierarchy [12] (this holds already for automatic successor trees). However, if one restricts to special subclasses of automatic structures, this complexity bound can be reduced. For example, for the class of automatic ordinals and also the class of automatic Boolean algebras, the isomorphism problem is decidable. Another interesting result is that the isomorphism problem for locally finite automatic graphs is $\Pi^0_1$-complete [19]. All these classes of automatic structures have the nice property that one can decide whether a given automatic presentation describes a structure from this class. Thm. 2.1 implies that this property also holds for the classes of equivalence relations, trees of height at most $n$, and linear orders, i.e., the classes considered in this paper.

3 Automatic Trees

A tree is a structure $T = (V; \leq)$, where $\leq$ is a partial order with a least element, called the root, and 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. The level of a node $x \in V$ is $\{|y \mid y < x\}$, where the height of $T$ is the supremum of the levels of all nodes in $V$; it may be infinite, but this paper deals with trees of finite height only. One may also view a tree 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 $\{x \mid x < v\}$. The edge relation $E$ is FO-definable in $(V; \leq)$. In this paper, we assume the partial order definition for trees, but will quite often refer to them as graphs for convenience. We use $T_n$ to denote the class of automatic trees with height at most $n$. Let $n$ be fixed. Then the tree order $\leq$ is uniformly FO-definable from the edge relation on the class of all trees of height at most $n$. Moreover, it is decidable whether a given automatic graph belongs to $T_n$, since the class of trees of height $n$ can be axiomatized in first-order logic.

In this section, we prove that the isomorphism problem for $T_n$ is $\Pi^0_{2n-3}$-complete. We start with the upper bound:

**Proposition 3.1** The isomorphism problem for the class $T_n$ of automatic trees of height at most $n$ is (i) decidable for $n = 1$ and (ii) in $\Pi^0_{2n-3}$ for all $n \geq 2$.

**Proof.** We first show that $T_1 \cong T_2$ is decidable for automatic trees $T_1, T_2 \in T_n$ at height at most $1$: It suffices to compute the cardinality of $T_i$ ($i \in \{1, 2\}$) which is possible since the universes of $T_1$ and $T_2$ are regular languages.

Now let $n \geq 2$ and consider $T_1, T_2 \in T_n$. Let $T_i = (V_i, E_i)$, w.l.o.g. $V_1 \cap V_2 = \emptyset$, and $V = V_1 \cup V_2$, $E = E_1 \cup E_2$. For any node $u \in V$, let $T(u)$ denote the subtree (of either $T_1$ or $T_2$) rooted at $u$ and let $E(u)$ be the set of children of $u$. For $k = n - 2, n - 3, \ldots, 0$, we will define inductively a $\Pi^0_{2n-2k-3}$-predicate $\text{iso}_k(u_1, u_2)$ for $u_1, u_2 \in V$. This predicate expresses that $T(u_1) \cong T(u_2)$ provided $u_1$ and $u_2$ belong to level at least $k$. The result will follow since $T_1 \cong T_2$ if and only if $\text{iso}_0(T_1, T_2)$ holds, where $r_x$ is the root of $T_x$.

For $k = n - 2$, the trees $T(u_1)$ and $T(u_2)$ have height at most 2. The statement $\text{iso}_{n-2}(u_1, u_2)$ can be defined as follows: For all $\kappa \in \mathbb{N} \cup \{\emptyset\}$ and all $\ell \geq 1$ we have

$$\exists x_1, \ldots, x_\ell \in E(u_1) : \bigwedge_{1 \leq i < j \leq \ell} x_i \neq x_j \land \bigwedge_{i=1}^\ell |E(x_i)| = \kappa$$

if and only if

$$\exists y_1, \ldots, y_\ell \in E(u_2) : \bigwedge_{1 \leq i < j \leq \ell} y_i \neq y_j \land \bigwedge_{i=1}^\ell |E(y_i)| = \kappa.$$ 

In other words: for every $\kappa \in \mathbb{N} \cup \{\emptyset\}$, $u_1$ and $u_2$ have the same number of children with exactly $\kappa$ children. Since $\text{FO} + \exists \forall \exists$ is uniformly decidable for automatic structures, this is indeed a $\Pi^0_1$-sentence (note that $2n - 2k - 3 = 1$ for $k = n - 2$). For $0 \leq k < n - 2$, we define $\text{iso}_k(u_1, u_2)$ inductively as follows: For all $v \in E(u_1) \cup E(u_2)$ and all $\ell \geq 1$ we have

$$\exists x_1, \ldots, x_\ell \in E(u_1) : \bigwedge_{1 \leq i < j \leq \ell} x_i \neq x_j \land \bigwedge_{i=1}^\ell \text{iso}_{k+1}(v, x_i)$$

if and only if

$$\exists y_1, \ldots, y_\ell \in E(u_2) : \bigwedge_{1 \leq i < j \leq \ell} y_i \neq y_j \land \bigwedge_{i=1}^\ell \text{iso}_{k+1}(v, y_i).$$

By quantifying over all $v \in E(u_1) \cup E(u_2)$, we quantify over all isomorphism types of trees that occur as a subtree rooted at a child of $u_1$ or $u_2$. For each of these isomorphism types $\tau$, we express that $u_1$ and $u_2$ have the same number of children $x$ with $T(x)$ of type $\tau$. Since by induction, $\text{iso}_{k+1}(v, x_i)$ and $\text{iso}_{k+1}(v, y_i)$ are $\Pi^0_{2n-2k-5}$-statements, $\text{iso}_k(u_1, u_2)$ is a $\Pi^0_{2n-2k-3}$-statement.

The rest of this section is devoted to proving that the isomorphism problem for the class $T_n$ of automatic trees of height at most $n \geq 2$ is also $\Pi^0_{2n-3}$-hard (and therefore complete). For this, we provide a generic reduction from an arbitrary $\Pi^0_{2n-3}$-predicate $P_n(x_0)$ to the isomorphism problem for $T_n$. In the following lemma and its proof, all quantifiers with unspecified range run over $\mathbb{N}$. For any $P_n(x_0)$, there exist $\Pi^0_{2n-3}$-predicates $P_i(x_0, x_1, y_1, x_2, y_2, \ldots, x_{n-i}, y_{n-i})$ for $2 \leq i < n$ such that

$$\text{iso}_k(u_1, u_2)$$
(a) for all \(2 \leq i \leq n - 1\), \(P_{i+1}(\tau)\) is logically equivalent to \(\forall x_{n-i} \exists y_{n-i} : P_i(\tau, x_{n-i}, y_{n-i})\), and

(b) if \(\forall y_{n-i} : \neg P_i(\tau, x_{n-i}, y_{n-i})\) holds, then also \(\forall x'_{n-i} \geq x_{n-i} \forall y_{n-i} : \neg P_i(\tau, x'_{n-i}, y_{n-i})\).

where \(\tau = (x_0, x_1, \ldots, x_{n-1}, y_{n-1})\).

**Proof.** The predicates \(P_i\) are constructed by induction, starting with \(i = n - 1\) down to \(i = 2\) where the construction of \(P_2\) does not assume that (a) or (b) hold true for \(P_{i+1}\). So let \(2 \leq i < n\) such that \(P_{i+1}(\tau)\) is a \(\Pi^0_{(i+1)-3}\)-predicate. Then there exists a \(\Pi^0_{(i+2)-3}\)-predicate \(P(\tau, x_{n-i}, y_{n-i})\) such that \(P_{i+1}(\tau)\) is logically equivalent to \(\forall x_{n-i} \exists y_{n-i} : P(\tau, x_{n-i}, y_{n-i})\). But this is logically equivalent to

\[
\forall x_{n-i} \forall x'_{n-i} \leq x_{n-i} \exists y_{n-i} : P(\tau, x'_{n-i}, y_{n-i}) .
\]

(1)

Let \(\varphi(\tau, x_{n-i})\) be the formula \(\forall x'_{n-i} \leq x_{n-i} \exists y_{n-i} : P(\tau, x'_{n-i}, y_{n-i})\). Then for any \(x_{n-i} \in \mathbb{N}\),

\[
\neg \varphi(\tau, x_{n-i}) \implies \forall x \geq x_{n-i} : \neg \varphi(\tau, x).
\]

(2)

Since \(\forall x'_{n-i} \leq x_{n-i}\) is a bounded quantifier, the formula \(\varphi(\tau, x_{n-i})\) belongs to \(\Sigma^0_{(i+2)-2}\) (see for example [21, p. 61]). Thus there is a \(\Pi^0_{(i+2)-3}\)-predicate \(P(\tau, x_{n-i}, y_{n-i})\) such that

\[
\varphi(\tau, x_{n-i}) \iff \exists y_{n-i} : P(\tau, x_{n-i}, y_{n-i}) .
\]

(3)

Therefore (1) (and therefore \(P_{i+1}(\tau)\)) is logically equivalent to \(\forall x_{n-i} \exists y_{n-i} : P(\tau, x_{n-i}, y_{n-i})\), which shows statement (a). For (b) note that \(\forall y_{n-i} : \neg P_i(\tau, x_{n-i}, y_{n-i})\) if and only if (by (3)) \(\neg \varphi(\tau, x_{n-i})\), which by (2) implies \(\forall x \geq x_{n-i} : \neg \varphi(\tau, x)\). By (3) again, this is equivalent to \(\forall x \geq x_{n-i} \forall y_{n-i} : \neg P_i(\tau, x, y_{n-i})\).

Let us fix the predicates \(P_i\) for the rest of Sec. 3. By induction on \(2 \leq i \leq n\), we construct the following trees:

- test trees \(T^2_k \in T_i\) for \(\tau \in \mathbb{N}^{i+2(n-i)}\) (which depend on \(P_i\)) and

- trees \(U^i_k \in T_i\) for \(k \in \mathbb{N}_{+} \cup \{\omega\}\).

The crucial properties of these trees are the following, where \(\tau \in \mathbb{N}^{i+2(n-i)}\):

**(P1)** \(P_i(\tau)\) if and only if \(T^2_k \cong U^i_k\).

**(P2)** \(\neg P_i(\tau)\) if and only if \(T^2_k \cong U^i_m\) for some \(m \in \mathbb{N}_{+}\).

For \(3 \leq i \leq n\), the idea is that \(T^2_k \cong U^i_m\) if and only if \(k = \inf\{\omega \cup \{x_{n-i+1} \mid \forall y_{n-i+1} \in \mathbb{N}_{+} : \neg P_{i+1}(\tau, x_{n-i+1}, y_{n-i+1})\}\}\). Property (P1) is certainly sufficient for proving \(\Pi^0_{(i+2)-3}\)-hardness of the isomorphism problem of automatic trees of height \(n\), property (P2) and therefore the trees \(U^i_m\) for \(m < \omega\) are used in the inductive step. We also need the following property for the construction:

**(P3)** No leaf of any of the trees \(T^2_k\) or \(U^i_k\) is a child of the root.

In Section 3.1, we will describe the trees \(T^2_k\) and \(U^i_k\) of height at most \(i\) and prove (P1) and (P2). Condition (P3) will be obvious from the construction. Section 3.2 is then devoted to proving the effective automaticity of these trees.

### 3.1 Construction of trees

We start with a few definitions: A forest is a disjoint union of trees. Let \(H\) and \(J\) be two forests. The forest \(H^\omega\) is the disjoint union of countably many copies of \(H\). Formally, if \(H = (V, E)\), then \(H^\omega = (V \times \mathbb{N}, E')\) with \(((v, i), (w, j)) \in E'\) if and only if \((v, w) \in E\) and \(i = j\). We have \(H \sim J\) for \(H^\omega \cong J^\omega\). Then \(H \sim J\) if they are formed, up to isomorphism, by the same set of trees (i.e., any tree is isomorphic to some connected component of \(H\) if and only if it is isomorphic to some connected component of \(J\)). If \(r\) does not belong to the domain of \(H\), then we denote with \(r \circ H\) the tree that results from adding \(r\) to \(H\) as the least element.

#### 3.1.1 Induction base: construction of \(T^2_0, T^2_1\) and \(U^2_0, U^1_0\)

For notational simplicity, we write \(k\) for \(1 + 2(n-2)\). Hence, \(P_2\) is a \(k\)-ary predicate. By Matiyasevich’s theorem, we find two non-zero polynomials \(p_1(x_1, \ldots, x_\ell)\), \(p_2(x_1, \ldots, x_\ell) \in \mathbb{N}[\pi], \ell > k\), such that for any \(\tau \in \mathbb{N}^k\):

\[

P_2(\tau) \iff \forall \pi \in \mathbb{N}^{k-\ell} : p_1(\pi, \tau) \neq p_2(\tau, \pi).

\]

It is well known that the function \(C(\tau) : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}\) with

\[

C(x, y) = (x + y)^2 + 3x + y

\]

is injective \((C(x, y))/2\) defines a pairing function, see e.g. [8]). For two numbers \(m, n \in \mathbb{N}_{+}\), let \(T[m, n]\) denote the tree of height 1 with exactly \(C(m, n)\) leaves. Then define the following forests, where \(k \in \mathbb{N}_{+} \cup \{\omega\}\):

\[

H^2 = \bigcup\{T[m, n] \mid m, n \in \mathbb{N}_{+}, m \neq n\}

\]

\[

H^\omega_2 = \bigcup\{T[p_1(\tau, x) + x_{\ell+1}, p_2(\tau, x) + x_{\ell+1}] \mid \pi \in \mathbb{N}^{k-\ell}, x_{\ell+1} \in \mathbb{N}_{+}\}

\]

\[

J^\omega_2 = \bigcup\{T[x, x] \mid x \in \mathbb{N}_{+}, x > \kappa\}

\]

Note that \(J^\omega_2 = H^2\). Moreover, the forests \(J^\omega_2(\kappa \in \mathbb{N}_{+} \cup \{\omega\})\) are pairwise non-isomorphic, since \(C\) is injective. The tree \(T^2_2\) (resp. \(U^2_2\)) is obtained from \(H^\omega_2\) (resp. \(J^\omega_2\)) by taking countably many copies and adding a root:

\[

T^2_2 = r \circ (H^\omega_2)^\omega

\]

and \(U^2_2 = r \circ (J^\omega_2)^\omega\),

see Fig. 1 and 2. The following lemma states (P1) for the \(\Pi^0_{k}\)-predicate \(P_2\), i.e., for \(i = 2\).

**Lemma 3.3** For all \(\tau \in \mathbb{N}^k\), \(P_2(\tau) \iff T^2_2 \cong U^2_2\).
Proof. By (5), it suffices to show that \( P_2(\tau) \) holds if and only if \( H^2_2 \sim J^2_2 \). So first assume \( P_2(\tau) \) holds. We have to prove that the forests \( H^2_2 \) and \( J^2_2 = H^2 \) contain the same trees (up to isomorphism). Clearly, every tree from \( H^2_2 \) is contained in \( H^2 \). For the other direction, let \( \tau \in N^{\ell-k}_+ \) and \( x_{\ell+1} \in N_+ \). Then the tree \( T[p_1(\tau, x), x_{\ell+1}, p_2(\tau, x) + x_{\ell+1}] \) occurs in \( H^2_2 \). Since \( P_2(\tau) \) holds, we have \( p_1(\tau, x) \neq p_2(\tau, x) \) and therefore \( p_1(\tau, x) + x_{\ell+1} \neq p_2(\tau, x) + x_{\ell+1} \). Hence this tree also occurs in \( H^2 \). Conversely suppose \( H^2_2 \sim H^2 \) and let \( \tau \in N^{\ell-k}_+ \). Then the tree \( T[p_1(\tau, x) + x_{\ell+1}, p_2(\tau, x) + x_{\ell+1}] \) occurs in \( H^2_2 \) and therefore in \( H^2 \). Hence \( p_1(\tau, x) \neq p_2(\tau, x) \). Since \( \tau \) was chosen arbitrarily, this implies \( P_2(\tau) \).

Now consider the forest \( H^2_2 \) once more. If it contains a tree of the form \( T[m, m] \) for some \( m \) (necessarily \( m \geq 2 \)), then it contains all trees \( T[x, x] \) for \( x \geq m \). Hence, \( H^2_2 \sim J^2_2 \) for some \( \kappa \in N_+ \cup \{\omega\} \), which implies \( T^2_\kappa \cong U^2_\kappa \) for some \( \kappa \in N_+ \cup \{\omega\} \). Thus, with Lemma 3.3 we get:

\[
\neg P_2(\tau) \iff T^2_\kappa \not\cong U^2_\kappa \\
\iff \exists m \in N_+ : T^2_\kappa \not\cong U^2_m
\]

Thus, we proved (P2) for the \( \Pi^0_1 \) predicate \( P_2 \). This finishes the construction of the trees \( T^2_\kappa \) and \( U^2_\kappa \) for \( \kappa \in N_+ \cup \{\omega\} \), and the verification of properties (P1) and (P2). Clearly, also (P3) holds for \( T^2_\kappa \) and \( U^2_\kappa \) (all maximal paths have length 2).

### 3.1.2 Induction step: construction of \( T^i+1_\tau \) and \( U^i+1_\kappa \)

Again, we write \( k \) for \( 1 + 2(n - i - 1) \). Thus, \( P_{i+1} \) is a \( k \)-ary predicate and \( P_i \) a \( (k + 2) \)-ary one. We now apply the induction hypothesis. For any \( \tau \in N^k_+ \), \( x, y \in N_+ \), \( \kappa \in N_+ \cup \{\omega\} \) let \( T^i_{\kappa xy} \) and \( U^i_{\kappa} \) be trees of height at most \( i \) such that:

\[
P_i(\tau, x, y) \iff T^i_{\kappa xy} \cong U^i_{\kappa} \\
\neg P_i(\tau, x, y) \iff \exists m \in N_+ : T^i_{\kappa xy} \not\cong U^i_m.
\]

In a first step, we build trees \( T^i_{\kappa xy} \) and \( U^i_{\kappa, x} \) (\( x \in N_+ \)) from \( T^i_{\kappa xy} \) and \( U^i_\kappa \), resp., by adding \( x \) leaves as children of the root. This ensures:

\[
T^i_{\kappa xy} \cong U^i_{\kappa, x'} \iff x = x' \land T^i_{\kappa xy} \cong U^i_{\kappa}, \quad (6)
\]

since, by property (P3), no leaf of any of the trees \( T^i_{\kappa xy} \) or \( U^i_{\kappa} \) is a child of the root. Next, we collect these trees into forests as follows:

\[
H^{i+1} = \bigcup \{U'_{m, x} \mid x, m \in N_+ \},
\]

\[
H^i_{\kappa} = H^{i+1} \cup \bigcup \{T^i_{\kappa xy} \mid x, y \in N_+ \},
\]

\[
J^i_{\kappa} = H^{i+1} \cup \bigcup \{U^i_{\kappa, x} \mid 1 \leq x < \kappa \} \quad \text{for} \quad \kappa \in N_+ \cup \{\omega\}.
\]

The tree \( T^i_{\kappa xy} \) (resp. \( U^i_{\kappa, x} \)) is obtained from the forest \( H^i_{\kappa} \) (resp. \( J^i_{\kappa} \)) by taking countably many copies and adding a root:

\[
T^i_{\kappa xy} = r \circ (H^{i+1}_\omega)^\omega \quad \text{and} \quad U^i_{\kappa, x} = r \circ (J^{i+1}_{\kappa})^\omega, \quad (7)
\]

see Fig. 3 and 4. Note that the height of any of these trees is one more than the height of the forests defining them and therefore at most \( i + 1 \). Since none of the connected components of the forests \( H^i_{\kappa} \) and \( J^i_{\kappa} \) is a singleton, none of the trees in (7) has a leaf that is a child of the root and therefore (P3) holds. The next lemma states (P1) for \( i + 1 \):

**Lemma 3.4** For all \( \tau \in N^k_+ \), \( P_{i+1}(\tau) \iff T^i_{\kappa xy} \cong U^i_{\kappa, x} \).
Proposition 3.6 For the $T_{i+1}$ from Section 3.1, it is actually easier to work with dags $H$ if for all $x \geq 1$ the trees from $U_{i,x}$ for any $m, x, x' \in \mathbb{N}_+$, this implies the existence of $x', y' \geq 1$ with $T_{x,y'} \equiv U_{i,x}$ By (6), this is equivalent to $x = x'$ and $T_{x,y'} \equiv U_{i,x}$. Now the induction hypothesis implies that $P_i(\tau, x, y)$ holds. Since $x \geq 1$ was chosen arbitrarily, we get $P_{i+1}(\tau)$.

Conversely suppose $P_{i+1}(\tau)$. Let $T$ belong to $H$. By the induction hypothesis, it is one of the trees $U_{i,x}$ for some $x \in \mathbb{N}_+, \kappa \in \mathbb{N}_+ \cup \{\omega\}$. In any case, it also belongs to $H_{i+1}$. Hence it remains to show that any tree of the form $U_{i,x}$ belongs to $H_{i+1}$. So let $x \in \mathbb{N}_+$. Then, by $P_{i+1}(\tau)$, there exists $y \in \mathbb{N}_+$ with $P_i(\tau, x, y)$. By the induction hypothesis, we have $T_{x,y} \equiv U_{i,x}$ and therefore $T_{x,y} \equiv U_{i,x}$ (which belongs to $H_{i+1}$ by the very definition). □

Lemma 3.5 For all $\tau \in \mathbb{N}_+^{\omega}$, there exists $\kappa \in \mathbb{N}_+ \cup \{\omega\}$ such that $T_{\tau}^{i+1} \equiv U_{\kappa}^{i+1}$.

Proof. It suffices to prove that $H^{i+1} \sim J_{\kappa}^{i+1}$ for some $\kappa \in \mathbb{N}_+ \cup \{\omega\}$. Let $\kappa$ be the smallest value in $\mathbb{N}_+ \cup \{\omega\}$ with $\forall x \geq \kappa \forall y : \sim P_i(\tau, x, y)$. By property (b) from Lemma 3.2 for $P_i$, we get $\forall x \leq \kappa \exists y : P_i(\tau, x, y)$. By the induction hypothesis, we get $\forall x \geq \kappa \forall y : T_{x,y} \neq U_{i,x}$ and $\forall x < \kappa \exists y : T_{x,y} \equiv U_{i,x}$. Thus, $H_{i+1}$ contains, apart from the trees in $H_{\kappa+1} = \bigcup\{U_{i,x} \mid x, m \in \mathbb{N}_+\}$, exactly the trees from $\{U_{i,x} \mid 1 \leq x < \kappa\}$, i.e., $H_{i+1} \sim J_{\kappa}^{i+1}$. □

Lemma 3.4 and 3.5 immediately imply also (P2) for $i+1$. Finally, (P1) for $i = n$ gives:

Proposition 3.6 For the $\Pi^0_3$-predicate $P_n(x)$ we have for all $c \in \mathbb{N}_+$: $P_n(c)$ if and only if $T^n_\sigma \equiv U^n_\sigma$.

It remains to show that the trees $T^n_\sigma$ and $U^n_\sigma$ are effectively automatic – this is the topic of the next section.

3.2 Automaticity

For constructing automatic presentations for the trees from Section 3.1, it is actually easier to work with dags (directed acyclic graphs). The height of a dag $D$ is the length (number of edges) of a longest directed path in $D$. We only consider dags of finite height. A root of a dag is a node without incoming edges. A dag $D = (V, E)$ can be unfolded into a forest $\text{unfold}(D)$ in the usual way: Nodes of $\text{unfold}(D)$ are directed paths in $D$ that cannot be extended to the left (i.e., the initial node of the path is a root) and there is an edge between two paths $p, p'$ if and only if $p'$ extends $p$ by one more edge. For a node $v \in V$ of $D$, we define the tree $\text{unfold}(D, v)$ as follows: First we restrict $D$ to those nodes that are reachable from $v$ and then we unfold the resulting dag. We need the following lemma.

Lemma 3.7 From given $k \in \mathbb{N}$ and an automatic dag $D = (V, E)$ of height at most $k$, one can construct effectively an automatic presentation $P$ with $S(P) \cong \text{unfold}(D)$.

Proof. The universe for our automatic copy of $\text{unfold}(D)$ is the set $P$ of all convolutions $v_1 \otimes v_2 \otimes \cdots \otimes v_m$, where $v_1$ is a root and $(v_i, v_{i+1}) \in E$ for all $1 \leq i < m$. Since $D$ has height at most $k$, we have $m \leq k+1$. Since the edge relation of $D$ is automatic and since the set of all roots in $D$ is first-order definable and hence regular, $P$ is indeed a regular set. Moreover, the edge relation of $\text{unfold}(D)$ becomes clearly FA recognizable on $P$.

For $2 \leq i \leq n$, let $F^i$ be the forest $\bigcup\{T^n_\sigma \mid \sigma \in \mathbb{N}_+^{i+2(n-i)}\} \cup \{U^n_\kappa \mid \kappa \in \mathbb{N}_+ \cup \{\omega\}\}$.

By induction over $i$, we will prove:

Proposition 3.8 There is an an automatic copy $F^i$ of $F^i$ and an isomorphism $f^i : F^i \rightarrow F^i$ that maps (i) the root of the tree $T^n_\sigma$ to $\alpha^\sigma$ (for all $\sigma \in \mathbb{N}_+^{i+2(n-i)}$), (ii) the root of the tree $U^n_\kappa$ to $\epsilon$, and (iii) the root of the tree $U^n_m$ to $\emptyset^m$ (for all $m \in \mathbb{N}_+$).

This will give the desired result since $T^n_\sigma$ is then isomorphic to the connected component of $\mathbb{F}^\sigma$ that contains the word $\alpha^\sigma$ (and similarly for $U^n_\kappa$). Note that this connected component is again (effectively) automatic by Thm. 2.1, since the forest $F^i$ has bounded height.

By Lemma 3.7, it suffices to construct an automatic dag $D^i$ such that there is an isomorphism $h : \text{unfold}(D^i) \rightarrow F^i$ that is the identity on the set of roots of $D^i$.

3.2.1 Induction base: the automatic dag $D^2$

Recall that, for $i = 2$, we used two polynomials $p_1$ and $p_2$ from Matiyasevich’s theorem and constructed the trees $T^n_\sigma$ and $U^n_\kappa$ that then formed the forest $F^2$. To show automaticity of this forest (more precisely: of a suitable dag $D^2$), we therefore have to represent polynomials by automata. The basis for this representation, that is inspired by Honkala’s work [8], is provided by the following construction.

For a symbol $a$, let $\Sigma^a_k$ denote the alphabet $\Sigma^a_k = \{(\alpha, \emptyset)^k \setminus \{(\emptyset, \ldots, \emptyset)\}$ and let $a_i$ denote the $i^{th}$ component of $\sigma \in \Sigma^a_k$. For $\pi = (e_1, \ldots, e_k) \in \mathbb{N}_+^k$, define $a^\pi = a^{e_1} \otimes a^{e_2} \otimes \cdots \otimes a^{e_k} \in (\Sigma^a_k)^\ast$.

For a language $L$, we write $\otimes (L)$ for the language $

\{u_1 \otimes u_2 \otimes \cdots \otimes u_k \mid u_1, \ldots, u_k \in L\}$. 

\{
Lemma 3.9 There exists an algorithm that, given a non-zero polynomial \( p(\tau) \in \mathbb{N}[\tau] \) in \( k \) variables, constructs an NFA \( A[p(\tau)] \) on the alphabet \( \Sigma_k^2 \) with \( L(A[p(\tau)]) = \otimes_k(a^*) \) such that for all \( \tau \in \mathbb{N}_k^2 \), \( A[p(\tau)] \) has exactly \( p(\tau) \) accepting runs on input \( a^* \).

Proof. The NFA \( A[p(\tau)] \) is build by induction on the construction of the polynomial \( p(\tau) \); the base case is provided by the polynomials 1 and \( x_i \).

Let \( A[1] \) be a deterministic automaton with \( L(A[1]) = \otimes_k(a^*) \). Next, suppose \( p(x_1, \ldots, x_k) = x_i \) for some \( 1 \leq i \leq k \). Let \( A = \{ q_1, q_2 \}, \{ q_1, \Delta, q_2 \} \) with \( \Delta = \{ (q_1, \sigma, q_j) \mid j \in \{1, 2\}, \sigma \in \Sigma_k^2, \sigma^* = a \} \cup \{ (q_2, \sigma, q_2) \mid \sigma \in \Sigma_k^2 \} \). When the NFA \( A \) runs on an input word \( a^* \), it has exactly \( c_i \) many times the chance to move from state \( q_1 \) to the final state \( q_2 \). Therefore there are exactly \( c_i = p(\tau) \) many accepting runs on \( a^* \). Then \( A[p(\tau)] \) is the direct product of \( A \) and a deterministic automaton accepting \( \otimes_k(a^*) \).

Let \( p(\tau) = p_1(\tau) + p_2(\tau) \) be polynomials in \( \mathbb{N}[\tau] \). Assume as inductive hypothesis that there are two NFA \( A[p_i(\tau)] = (S_i, \Delta_i, I_i, F_i) \) such that the number of accepting runs of \( A[p_i(\tau)] \) on \( a^* \) equals \( p_i(\tau) \) for \( i \in \{1, 2\} \).

For \( p(\tau) = p_1(\tau) + p_2(\tau) \), let \( A[p(\tau)] \) denote the disjoint union of \( A[p_1(\tau)] \) and \( A[p_2(\tau)] \). For any word \( a^* \), the number of accepting runs of \( A[p(\tau)] \) on \( a^* \) is equal to the sum of the numbers of accepting runs of \( A[p_i(\tau)] \) and \( A[p_2(\tau)] \) on \( a^* \), which is \( p(\tau) \).

Lemma 3.10 Let \( q_1, q_2 \in \mathbb{N}[x_1, \ldots, x_\ell] \) and let \( a \) be some symbol. There is an automatic forest of height 1 over an alphabet \( \Sigma^2_k \) such that: (i) the set of roots is \( \otimes_k(a^*) \), (ii) the leaves are words from \( \Gamma^a \), and (iii) the tree rooted at \( a^* \) is isomorphic to \( T[q_1(\tau), q_2(\tau)] \).

Proof. Set \( p(\tau) = C(q_1(\tau), q_2(\tau)) \) (\( C \) is defined in (4)) and let \( A[p(\tau)] = (S, I, \Delta, F) \) be the NFA over the alphabet \( \Sigma^2_k \) from Lemma 3.9. Define the NFA \( B[p(\tau)] = (S, I, \Delta', F) \) with alphabet \( \Delta \) and \( \Delta' = \{ (p, (p, \sigma, q_2), q_2) \mid (p, \sigma, q_2) \in \Delta \} \) it accepts the set of accepting runs of \( A[p(\tau)] \). Let \( \pi : \Delta^* \rightarrow \Sigma_k^2 \) be the projection morphism with \( \pi(p, \sigma, q) = \sigma \). Then, for all \( \tau \in \mathbb{N}_k^2 \), the size of \( \pi^{-1}(a^*) \cap L(B[p]) \) equals the number of accepting runs of \( A[p(\tau)] \) on \( a^* \), which is \( p(\tau) \).

Then \( L \) is regular and \( E \) is FA recognizable, i.e., \( (L; E) \) is an automatic graph. It is actually a forest of height 1, the words from \( \otimes_k(a^*) \) form the roots, and the tree rooted at \( a^* \) has precisely \( p(\tau) \) leaves, i.e., it is isomorphic to \( T[q_1(\tau), q_2(\tau)] \).

From now on, we use the notations from Sec. 3.1.1. By Lemma 3.10, we can compute automatic forests \( F_1 \) and \( F_2 \) over alphabets \( \Sigma_{e+1}^2 \uplus \Gamma_1 \) and \( \Sigma_2^2 \uplus \Gamma_2 \), resp., such that:

(a) the roots of \( F_1 \) (resp. \( F_2 \)) are the words from \( \otimes_{e+1}(a^*) \) (resp. \( \otimes_2(b^*) \)),
(b) the leaves of \( F_i \) are words from \( \Gamma_i^a \) (\( i \in \{1, 2\} \)),
(c) the tree rooted at \( a^{e+1} \) is isomorphic to the tree \( T[p_1(\tau) + e_{e+1}, p_2(\tau) + e_{e+1}] \) for \( \tau \in \mathbb{N}_{e+1}^k, e_{e+1} \in \mathbb{N}_e \),
(d) the tree rooted at \( b^{e_{e+2}} \) is isomorphic to \( T[e_1, e_2] \) for \( e_1, e_2 \in \mathbb{N}_e \).

We can assume that the alphabets \( \Gamma_1, \Gamma_2, \Sigma_{e+1}^2, \) and \( \Sigma_2^2 \) are mutually disjoint. Let \( F = (V_F, E_F) \) be the disjoint union of \( F_1 \) and \( F_2 \); it is effectively automatic. The universe of the automatic dag \( D^2 \) is the regular language

\[ \otimes_k(a^*) \cup b^* \cup (\$ \otimes \mathbb{N}) \]

where \$ is a new symbol. We have the following edges:

- For \( u, v \in V_F \), \( \$^m \cup u \) is connected to \( \$^n \cup v \) if and only if \( m = n \) and \( (u, v) \in E_F \). This produces \( \mathbb{N}_c \) many copies of \( F \).

- \( a^* \) is connected to all words from \( \$^* \otimes (\{ a^* | \pi \in \mathbb{N}_{e+1}^a \} \cup \{ b^{e_{e+2}} | e_1 \neq e_2 \}) \).

- By point (c) and (d) above, this means that the tree unfolding \( (D^2, a^*) \) has \( \mathbb{N}_c \) many subtrees isomorphic to \( T[p(\tau) + e_{e+1}, p_2(\tau) + e_{e+1}] \) for \( \tau \in \mathbb{N}_{e+1}^k, e_{e+1} \in \mathbb{N}_e \), and \( T[e_1, e_2] \) for \( e_1, e_2 \in \mathbb{N}_e, e_1 \neq e_2 \). Hence, unfold\( (D^2, a^*) \cong T^2 \).

- \( e \) is connected to all words from \( \$^* \otimes \{ b^{e_{e+2}} | e_1 \neq e_2 \} \).

By (d) above, this means that the tree unfolding \( (D^2, e) \) has \( \mathbb{N}_c \) many subtrees isomorphic to \( T[e_1, e_2] \) for \( e_1, e_2 \in \mathbb{N}_e, e_1 \neq e_2 \). Hence, unfold\( (D^2, e) \cong U^2 \).

- \( b^m (m \in \mathbb{N}_e) \) is connected to all words from \( \$^* \otimes \{ b^{e_{e+2}} | e_1 \neq e_2 \} \) or \( e_1 = e_2 > m \). By (d), this means that the tree unfolding \( (D^2, b^m) \) has \( \mathbb{N}_c \) many subtrees isomorphic to \( T[e_1, e_2] \) for all \( e_1, e_2 \in \mathbb{N}_e, e_1 \neq e_2 \) or \( e_1 = e_2 > m \). Thus, unfold\( (D^2, b^m) \cong U_m^m \).

Hence, unfold\( (D^2) \cong F^2 \) and the roots are as required in Prop. 3.8. Moreover, it is clear that \( D^2 \) is automatic.
3.2.2 Induction step: the automatic dag $D^{i+1}$

Suppose $D^i = (V, E)$ is such that $F^i = \text{unfold}(D^i)$ as is described in Prop. 3.8. We use the notations from Sec. 3.1.2. We first build another automatic dag $D'$, whose unfolding contains (copies of) all trees $U'_{m,x}$ ($k \in \mathbb{N}_+ \cup \{\omega\}$, $x \in \mathbb{N}_+$) and $T'_{xy}$ ($c \in \mathbb{N}_k$, $x, y \in \mathbb{N}_+$). Recall that the set of roots of $D'$ is $\otimes_{k+2}(a^{xy})$ and $b^* \subseteq V$. The universe of $D'$ consists of the following regular set, where $\sharp, \sharp_1$, and $\sharp_2$ are new symbols:

$$(V \setminus b^*) \cup (\sharp^+ \otimes b^*) \cup \sharp_1^+ \otimes \sharp_2.$$

We have the following edges in $D'$:

- All edges from $E$ except those with an initial node in $b^*$ are present in $D'$.
- $a^{xy} \in V$ is additionally connected to all words of the form $\sharp_1^{i+2}$ for $r \in \mathbb{N}_k$, $x, y \in \mathbb{N}_+$, and $1 \leq i \leq x$. This ensures that the subtree rooted at $a^{xy}$ gets $x$ new leaves, which are children of the root $a^{xy}$. Thus $\text{unfold}(D', a^{xy}) \cong T'_{xy}$.
- $\sharp^+ \otimes b^m$ for $x \in \mathbb{N}_+$ and $m \in \mathbb{N}$ is connected to (i) all nodes to which $b^m$ is connected in $D'$ and to (ii) all nodes from $\sharp_2^{i+2}$ for $1 \leq i \leq x$. Hence, $\text{unfold}(D', \sharp^+ \otimes b^m) \cong U'_{m,x}$ in case $m \in \mathbb{N}_+$ and $\text{unfold}(D', \sharp^+ \otimes \epsilon) \cong U'_{\omega,x}$.

In summary, $D'$ is a dag, whose unfolding consists of (copies of) $U'_{m,x}$ rooted at $\sharp^+ \otimes \epsilon$, $U'_{m,x}$ ($m \in \mathbb{N}_+$) rooted at $\sharp^+ \otimes b^m$, and $T'_{xy}$ rooted at $a^{xy}$.

From the automatic dag $D'$, we now build in a final step the automatic dag $D^{i+1}$. This is very similar to the constructions of $D^i$ and $D'$ above. Let $V'$ be the universe of $D'$. The universe of $D^{i+1}$ is the regular language

$$\otimes_k(a^+) \cup b^* \cup (\sharp^+ \otimes V').$$

The edges are as follows:

- For $u, v \in V'$, $S^u \otimes v$ is connected to $S^u \otimes v$ if and only if $m = n$ and $(u, v)$ is an edge of $D'$. This generates $N_0$ many copies of $D'$.
- $a^r$ is connected to every word from $S^* \otimes (\{a^{xy} \mid x, y \in \mathbb{N}_+\}) \cup (\sharp^+ \otimes b^+)$.
- $\epsilon$ is connected to all words from $S^* \otimes (\sharp^+ \otimes b^+)$.

This finishes the proof of Prop. 3.8.

**Theorem 3.11** For $n \geq 2$, the set of automatic presentations of $U^n_{0, m}$ is hard for $\Pi^0_{2n-3}$. Hence, the isomorphism problem for automatic trees of height at most $n$ is $\Pi^0_{2n-3}$-complete. The isomorphism problem for automatic trees of finite height is recursively equivalent to FOTh($\mathbb{N}; +, \times$).

**Proof.** For the first statement, let $P_n(x_0)$ be any $\Pi^0_{2n-3}$-predicative and let $c \in \mathbb{N}_+$. Above, we constructed the automatic forest $F^n$ of height $n$. The tree $T^n_0$ is isomorphic to the tree rooted at $a^c$ and therefore effectively automatic. By Prop. 3.6, it is isomorphic to $U^n_0$ if and only if $P_n(c)$ holds. It follows that $U^n_0$ is automatic, hence the isomorphism problem for automatic trees of height $\leq n$ is hard (and therefore complete by Prop. 3.1) for $\Pi^0_{2n-3}$.

We now come to the third statement. Since the proof of Prop. 3.1 is uniform in the level $n$, we can compute from two automatic trees $T_1, T_2$ of finite height an arithmetical formula, which is true if and only if $T_1 \cong T_2$. The other direction follows from the first statement because of the uniformity in constructing the trees $T^n_0$ and $U^n_0$. $\square$

From Thm. 3.11 we can easily deduce a corollary on automatic equivalence structures. An equivalence structure is of the form $E = (D; E)$ where $E$ is an equivalence relation on $D$.

**Corollary 3.12** There exists an equivalence relation whose set of automatic presentations is $\Pi^0_{1}$-hard. Hence, the isomorphism problem for automatic equivalence structures is $\Pi^0_{1}$-complete.

**Proof.** By Thm. 3.11 it suffices to show that the isomorphism problem for $T_2$ is recursively equivalent to the isomorphism problem for automatic equivalence structures. First, let $E = (V; \equiv)$ be an automatic equivalence structure and let $\leq_{\text{lex}}$ be the length-lexicographic order on $V$. Now build the tree $T(E)$ of height at most 2 as follows: Let $r$ be a new letter that serves as root. Its children are the $\leq_{\text{lex}}$-minimal elements $u$ of the equivalence classes of $\equiv$, and the children of $u$ are the remaining elements of the equivalence class $[u]$. It is clear that $T(E)$ is a tree of height at most 2 such that $E_1 \cong E_2$ if and only if $T(E_1) \cong T(E_2)$. Moreover, from an automatic presentation for $E$, one can compute an automatic presentation for $T(E)$.

For the reverse reduction, let $T$ be a tree of height 2. We construct an equivalence structure $E(T)$ as follows: W.l.o.g.
assume that $T$ is not a single node. Then we first add to each
cchild of the root of $T$ a further child such that every maximal
path in $T$ has length 2. Let $T'$ be the resulting tree. Then
the elements of $\mathcal{E}(T)$ are the leaves of $T'$ and two leaves $u$
and $v$ are equivalent if and only if they have the same parent
node. Again it is easy to see that (i) from an automatic presen-
tation for $T$, one can compute an automatic presentation
for $\mathcal{E}(T)$ and (ii) $T_1 \cong T_2$ if and only if $\mathcal{E}(T_1) \cong \mathcal{E}(T_2)$. □

Let us close this section, with a brief discussion on the
isomorphism problem for computable trees of finite height.

Theorem 3.13 For every $n \geq 1$, the isomorphism problem
for computable trees of height at most $n$ is $\Pi_2^0$-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 $\Pi_2^0$-statement. For the inductive step, we can reuse
the arguments from the proof of Prop. 3.1.

For the lower bound, we first deal with the case $n = 1$. It
is known that the problem whether a given recursively enu-
erable set is infinite is $\Pi_2^0$-complete [21]. For a given de-
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 $T_M$ and connect it to all leaves. If $L(M)$ is infinite,
then $T_M$ is isomorphic to the height-1 tree with $\aleph_0$ leaves.
If $L(M)$ is finite, then there exists $n \in \mathbb{N}$ such that $T_M$
is isomorphic to the height-1 tree with $n$ leaves. We can
use this construction as the base case for our construction in
Sec. 3.1.2. This yields the lower bound for all $n \geq 1$. □

4 Automatic Linear Orders

Our main result for automatic linear orders is:

Theorem 4.1 For any $n \in \mathbb{N}$, there exists a linear order $K_n$, whose set of automatic presentations is hard for $\Sigma_{n}^0$. The isomorphism problem for the class of automatic linear orders is at least as hard as $\text{FOTh}(\mathbb{N}; +, \times)$.

The proof of this result follows our arguments for trees of
finite height but is technically more involved. Looking back
to the proof of Thm. 3.11, we see that trees are used in order
to encode sets of sets of . . . sets of natural numbers. For
linear orders, we replace the basic tree operation of gluing
together a set of trees into a single tree by adding a new
root by the shuffle sum. The shuffle sum of a countable
set of linear order types $\mathcal{L}$ is constructed as follows: First,
we densely color $\mathbb{Q}$ with the order types in $\mathcal{L}$, i.e., for all
rationals $x < y$ and all $L \in \mathcal{L}$ there exists $x < z < y$ such
that $z$ is colored with the order type $L$. The shuffle sum of $\mathcal{L}$
is the linear order that results from $(\mathbb{Q}, <)$ by replacing each
$L$-colored rational $(\in \mathcal{L})$ with the order $L$. Assuming that
every order type in $\mathcal{L}$ starts with some ordinal $\omega \cdot i (i \in \mathbb{N})$
and does not contain $\omega \cdot i$ as an interval elsewhere, the shuffle
sum of $\mathcal{L}$ encodes the set $\mathcal{L}$ as a linear order. In our proof
of Thm. 4.1 we use iterated shuffle sums. In order to stay
within automatic linear orders, we have to realize shuffle
sums in an automatic way, details can be found in the full
version [15] of this paper.

In [13], it is shown that every automatic linear order has
finite FC-rank (for a definition, see e.g. [13]). A linear or-
der $L$ has FC-rank 1, if after identifying all $x, y \in L$ such
that the interval $[x, y]$ is finite, one obtains a dense ordering
or the singleton linear order. The result of [13] mentioned
above suggests that the isomorphism problem might be sim-
pler for linear orders of low FC-rank.

Corollary 4.2 The isomorphism problem for automatic
linear orders of FC-rank 1 is at least as hard as $\text{FOTh}(\mathbb{N}; +, \times)$.

Proof. We provide a reduction from the isomorphism
problem for automatic linear orders of arbitrary rank. If $L$
is an automatic linear order, then so is $\tilde{L} = ((-1, 0) + 1, 2) \cdot L$. This linear order is obtained from $L$ by replacing
each point with a copy of the rational numbers in $(-1, 0) \cup [1, 2]$. Then $\tilde{L}$ has FC-rank 1: Only the copies of 0 and 1 will be
identified, and the resulting order is isomorphic to $(\mathbb{Q}, \leq)$.
Moreover, $L$ is isomorphic to the set of all $x \in \tilde{L}$ satisfying
$\exists z > x \forall y : (x < y \leq z \rightarrow y = z)$. Hence $L_1 \cong L_2$ if and
only if $\tilde{L}_1 \cong \tilde{L}_2$, which completes the reduction. □

5 Arithmetical isomorphisms

We conclude this paper with an application of
Thms. 3.11 and 4.1. 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. An
isomorphism $f$ between two automatic structures with do-
 mains $L_1$ and $L_2$, resp., is a $\Sigma_k^0$-isomorphism, if the set
$\{ (x, f(x)) \mid x \in L_1 \}$ belongs to $\Sigma_k^0$.

Corollary 5.1 For any $k \in \mathbb{N}$, there exist two isomorphic
automatic trees of finite height (and two automatic linear
orders) without any $\Sigma_k^0$-isomorphism.

Proof. Assume that between any two isomorphic automatic
trees of finite height, there always exists a $\Sigma_k^0$-
isomorphism. Then the isomorphism problem for automatic
trees of finite height would belong to $\Sigma^0_{k+2}$ (which contradicts Thm. 3.11): two automatic trees $T_1 = (D_1, E_1)$ and $T_2 = (D_2, E_2)$ of finite height are isomorphic if there exists a $\Sigma^0_k$-predicate $P(x, y)$ such that for all $x_1, x_2 \in D_1$ there exist $y_1, y_2 \in D_2$ (and vice versa) such that: $P(x_1, y_1)$, $P(x_2, y_2)$, $x_1 \leftrightarrow y_1 = y_2$, and $((x_1, x_2) \in E_1 \leftrightarrow (y_1, y_2) \in E_2)$. Since $P$ is a $\Sigma^0_k$-predicate, this is a $\Sigma^0_{k+2}$-statement, which expresses the existence of a $\Sigma^0_{k+2}$ isomorphism from $T_1$ to $T_2$. For linear orders we can argue in the same way.

6 Open problems and outlook

The main open problem, which remains, is the precise complexity of the isomorphism problem for automatic linear orders as well as the isomorphism problem for automatic scattered linear orders where our technique seems not to work. To our knowledge, it is also open whether the isomorphism problem for automatic (in the sense of [10] or [5]) is decidable. In [16], we prove that the isomorphism problem for scattered linear orders as well as the isomorphism problem for automatic graphs of finite height is not analytical. This proof uses techniques similar to those in this paper.

References