Analyzing Complexity in Classes of Automatic Structures

Jiamou Liu\textsuperscript{1}, Mia Minnes\textsuperscript{2}

\textsuperscript{1} Department of Computer Science, University of Auckland, New Zealand
jliu036@aucklanduni.ac.nz

\textsuperscript{2} Department of Mathematics, MIT, USA
minnes@math.mit.edu

Abstract. This paper addresses the complexity of several families of queries (including membership and isomorphism) in classes of unary automatic structures, and the state complexity of describing these classes. We focus on automatic equivalence relations, linear orders, trees, and graphs with finite degree. A unary automatic structure is one that can be described by finite automata over the unary alphabet. In each setting, we either greatly improve on known algorithms (reducing highly exponential bounds to small polynomials) or answer open questions about the existence of decision procedures by explicitly giving algorithms.

1 Introduction

A (relational) structure is automatic if its elements can be coded in a way such that the domain and all the relations of the structure are recognized by finite automata (precise definitions are in Section 2). Automatic structures form a large class of infinite structures with finite representations and effective semantics. In particular, for any automatic structure $\mathcal{A}$ and first-order query $\varphi$, one can effectively construct an automaton that recognizes all elements of $\mathcal{A}$ that satisfy $\varphi$. Such useful algorithmic and model-theoretical properties of automatic structures have led to extensive work in the area in recent years.

The field of automatic structures can be viewed as an extension of finite model theory in which one studies the interaction between logical definability and computational complexity. As finite model theory has found applications in databases [14] automatic structures have been used in relational databases and computer-aided verifications [19, 18]. However, this approach has limitations. In particular, since the configuration space of a Turing machine can be coded by a finite automaton [12], reachability is undecidable for automatic structures in general. On the other hand, unary automatic structures, those recognized by automata over a unary alphabet, have decidable monadic second-order theories. The restriction to a unary alphabet is a natural special case of automatic structures because any automatic structure has an isomorphic copy over the binary alphabet [16]. Moreover, if we consider the intermediate class of structures whose domain elements are encoded as finite strings over $1^*2^*$ reachability is no longer decidable because the grid can be coded.
Much is known about the complexity of automatic structures. Algebraic characterizations have been given for automatic Boolean algebras [12] and finitely generated automatic groups [15]. Some of these results give information about computational content: the isomorphism problem of automatic Boolean algebras is decidable. However, the full class of automatic structures has significant underlying complexity. From a computability theoretic point of view, the isomorphism problem and embedding problem for automatic structures are both \( \Sigma^1_1 \)-complete [12, 20]. Model-theoretically, we also find richness: [9] shows that the Scott ranks of automatic structures can be as high as possible, up to the successor of the first non-computable ordinal. Finally, the resource-bounded complexity of automatic structures has been studied. For a fixed automaton, the complexity of model-checking for quantifier-free formulas is LOGSPACE-complete and for existential formulas it is NPTIME-complete [2]. On the other hand, there are automatic structures whose first-order theories are nonelementary [3].

In this paper we consider classes of unary automatic structures and show they are simple with respect to two notions of complexity. First, we study the time complexity of natural decision problems in a fixed class \( K \) of structures. The membership problem asks, given a unary automatic presentation of \( \mathcal{A} \), decide if \( \mathcal{A} \in K \). The isomorphism problem asks, given unary automatic presentations of \( \mathcal{A} \) and \( \mathcal{B} \) from \( K \), decide if \( \mathcal{A} \cong \mathcal{B} \). For each class of unary automatic structures we consider, we give low polynomial time solutions to its membership problem. For unary automatic equivalence relations, linear orders, and trees we solve the isomorphism problem in low polynomial time; the isomorphism problem for graphs with finite degree is decidable with elementary time bound.

State complexity measures the descriptive complexity of regular languages, context-free grammars, and other classes of languages with finite representations. The state complexity of a regular language \( L \) is defined as the size of the smallest automaton recognizing \( L \). It has been studied since the 1950s [4, 21, 22] and motivated, in part, by the relationship between the runtime of realtime computations running on automata and the size of their state space. We generalize state complexity to structures (rather than sets): the state complexity of an automatic structure \( \mathcal{A} \) is defined to be the number of states in an optimal automaton presentation of a structure isomorphic to \( \mathcal{A} \). For unary automatic structures, we require that the presentation witnessing the state complexity also be a unary automatic structure. We prove that the state complexity of unary automatic equivalence relations, linear orders, and trees are each polynomial with respect to a natural representation of the structures. (In each section, we explicit describe this representation.) The study of state complexity of automatic structures is a new area and our initial results hopefully indicate its fruitfulness.

Paper organization. Section 2 introduces the terminologies and notation used throughout the paper. In particular it recalls the definitions of automatic structures and unary automatic structures and makes precise the definition of state complexity for unary automatic structures. Sections 3, 4, 5 and 6 discuss linear orders, equivalence relations, trees and graphs of finite degree (respectively).
2 Preliminaries

We assume the basic terminology and notation from automata theory (see [7] for example). For a fixed alphabet \( \Sigma \), a finite automaton is a tuple \( A = (S, \Delta, \iota, F) \) where \( S, \Delta, \iota, F \) are (respectively) the state space, transition function, initial state, and accepting states. In particular, if \( A \) is a finite automaton over the unary alphabet \( \{1\} \) it is called a unary automaton. 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. Bits of the \( n \) input words are read in parallel until all input strings have been completely processed. Formally, let \( \Sigma_\circ = \Sigma \cup \{\circ\} \) where \( \circ \) is a symbol not in \( \Sigma \). Given an \( n \)-tuple of words \( w_1, w_2, \ldots, w_n \in \Sigma^* \), we convert them to a word over the alphabet \((\Sigma_\circ)^n\) with length \( \max\{|w_1|, \ldots, |w_n|\} \) whose \( k \)th symbol \((\sigma_1, \ldots, \sigma_n)\) where \( \sigma_i \) is the \( k \)th symbol of \( w_i \) if \( k \leq |w_i| \), and is \( \circ \) otherwise. An \( n \)-ary relation \( R \) is FA recognizable if the set of words obtained in this way is a regular subset of \((\Sigma_\circ)^n \).

A relational structure \( S \) consists of a countable domain \( D \) and atomic relations on \( D \). A structure is called automatic over \( \Sigma \) if it is isomorphic to a structure whose domain is a regular subset of \( \Sigma^* \) and each of whose atomic relations is FA recognizable. A structure is called unary automatic if it is automatic over the alphabet \( \{1\} \). The structures \((\mathbb{N}; S)\) and \((\mathbb{N}; \leq)\) are both automatic structures. On the other hand, \((\mathbb{Q}; \leq)\) and \((\mathbb{N}; +)\) are automatic over \( \{0, 1\} \) but are not unary automatic. The structure \((\mathbb{N}; \times)\) is not automatic over any finite alphabet. For proofs of these facts, see the survey papers [10, 17].

Consider \( FO + \exists^\infty + \exists^{n,m} \), the first-order logic extended by quantifiers for there exist infinitely many and there exist \( n \) many mod \( m \) \((n, m \in \mathbb{N})\). The following theorem from [2, 5, 11, 16] connects this extended logic with automata.

**Theorem 1.** For an automatic structure, \( A \), there is an algorithm that, given a formula \( \varphi(\bar{x}) \) in \( FO + \exists^\infty + \exists^{n,m} \), produces an automaton whose language is those tuples \( \bar{a} \) from \( A \) that make \( \varphi \) true.

We study automatic structures \((D; R)\) where \( R \) is a binary relation over \( D \). Theorem 1 can be used to give decision procedures for some properties of binary relations. Table 1 lists the complexity of the associated algorithms if \( A_D \) \((n \text{ states})\) and \( A_R \) \((n \text{ states})\) are deterministic FA recognizing \( D \) and \( R \), respectively. The proofs may be found in Appendix A. Note that if \((D; R)\) is automatic over \( \Sigma \) and \( D = \Sigma^* \), then \( m = 1 \).

We use \( x \) to denote the string \(1^x\) and \( \mathbb{N} \) for the set of all such strings \( \{1\}^* \). In [1], Blumensath shows that a structure is unary automatic if and only if it is first-order interpretable in \( \mathcal{U} = (\mathbb{N}; 0, <, s, \{\text{mod}_m\}_{m>1}) \), where \( s \) is the successor relation and \( \text{mod}_m(x, y) \) holds if and only if \( x \equiv y \mod m \), and therefore proves the following.

**Theorem 2.** The monadic second order (MSO) theory of a unary automatic structure is decidable.

It will be helpful to understand the structure of regular subsets and relations of \( \{1\}^* \). The proofs are outlined in Appendix A.
Definition 1. The state complexity of an (unary) automatic structure $\mathcal{A}$ is the size of the smallest (unary) automaton $\mathcal{M}$ (optimal automaton) such that $\mathcal{M}$ recognizes a structure $\mathcal{B} \cong \mathcal{A}$. If $\mathcal{K}$ is a class of automatic structures with associated finite isomorphism invariant $R_\mathcal{A}$, the state complexity of $\mathcal{K}$ is $f$ such that for $A \in \mathcal{K}$, if $|R_\mathcal{A}| \leq n$, the state complexity of $\mathcal{A}$ is less than $f(n)$.

We look at the state complexity of three classes of unary automatic structures: equivalence relations, linear orders, and trees. We define natural isomorphism invariants for each class and show that the state complexity for each class is polynomial with respect to these invariants.
In the sequel, we make the following assumptions. All structures are infinite with domain \(\mathbb{N}\) (see Lemma 6 for justification). All automata are deterministic. Algorithms on unary automatic structures \((\mathbb{N}; R)\) have as input a synchronous 2-tape automaton recognizing \(R\). The size of the input is the size of this input automaton. The sets of \((1,1)\)-, \((\diamond, 1)\)-, and \((1,\diamond)\)- states are pairwise disjoint.

3 Linear Orders

A linear order is \(L = (\mathbb{N}; \leq_L)\) where \(\leq_L\) is total, reflexive, antisymmetric, and transitive. Table 1 immediately gives that the membership problem for automatic linear orders is decidable in time \(O(n^3)\). Blumensath [1] and Khoussainov and Rubin [13] characterized unary automatic linear orders. We use \(\omega, \omega^*\), and \(n\) to denote the order types of the positive integers, the negative integers, the integers, and the finite linear order of length \(n\) (respectively).

**Theorem 4 ([1, 13]).** A linear order is unary automatic if and only if it is isomorphic to a finite sum of linear orders of type \(\omega, \omega^*\) or \(n\).

**Corollary 1.** The isomorphism problem for unary automatic linear orders is decidable.

The corollary is proved by defining, for each unary automatic linear order \(L\), a sentence \(\varphi_L\) such that for any \(L'\), \(L \equiv L'\) if and only if \(L' \models \varphi_L\) (see Appendix B). Doing so leads to a triply exponential decision procedure. We significantly improve this bound by providing a quadratic time algorithm for the isomorphism problem for unary automatic linear orders.

3.1 Efficient solution to the isomorphism problem

**Theorem 5.** The isomorphism problem for unary automatic linear orders is decidable in quadratic time in the sizes of the input automata.

We use the notation from Section 2: given a unary automaton \(A\) it has parameters \(t, \ell\) which are the lengths of its \((1, 1)\)-tail and -loop.

**Lemma 2.** Suppose \(A\) is a unary automaton that represents a linear order \(L = (\mathbb{N}; \leq_L)\). For any \(t \leq j < \ell\), the sequence \((j + i\ell)_{i \in \mathbb{N}}\) is either an increasing chain in a copy of \(\omega\) in \(L\) or a decreasing chain in a copy of \(\omega^*\) in \(L\).

**Proof (Proof of Theorem 5).** Suppose \(L = (\mathbb{N}; <_L)\) is a unary automatic linear order represented by a unary automaton \(A\) with parameters \(t, \ell\). We will extract its canonical word \(a_L \in \{\omega, \omega^*, n\}^*\) by determining the relative ordering of the at most \(\ell\) many copies of \(\omega\) and \(\omega^*\) given by Lemma 2 and the elements \(0, \ldots, t - 1\).

For \(t \leq j < t + \ell\), we decide in linear time whether the sequence \((j + i\ell)_{i \in \mathbb{N}}\) is in a copy of \(\omega\) or \(\omega^*\). Two sequences \((j + i\ell)_{i \in \mathbb{N}}\) and \((k + i\ell)_{i \in \mathbb{N}}\) \((j < k)\) interleave if they belong to the same copy of \(\omega\) or \(\omega^*\) in \(L\). We check this by reading the \((\diamond, 1)\)-loops at most \(\ell\) many times. These loops also specify the relative order
of \((j + il)_{i \in \mathbb{N}}\) and \((k + il)_{i \in \mathbb{N}}\) if they do not interleave. Thus, in \(O(n^2)\) time, we compute descriptions of the predecessors of each \((j + il)_{i \in \mathbb{N}}\) in \(\mathcal{L}\) and an equivalence relation on \(\{t, \ldots, t + \ell - 1\}\) based on interleaving. Similarly, we can compute the predecessors of each \(j, 0 \leq j < t\) in \(O(n)\) time. To extract \(\alpha_L\) from \(A\), it remains to iterate through states according to their \(\mathcal{L}\)-predecessors, and decided if they belong to a finite block, a copy of \(\omega\), or a copy of \(\omega^2\). This can be done in \(O(n^2)\) (see Appendix B for details).

\[\text{⊓ ⊔}\]

### 3.2 State complexity

The order type of a unary automatic linear order \(L = (\mathbb{N}; \leq_L)\) is specified by \(\alpha_L \in \{\omega, \omega^*, \{n\}_{n \in \mathbb{N}}\}^*\). Let \(m_L\) be the number of copies of \(\omega\) or \(\omega^*\) in \(\alpha_L\) and let \(k_L\) be the sum of all \(n\) such that \(n\) appears in \(w_L\). Define \(|\alpha_L| = \max\{m_L, k_L\}\).

**Theorem 6.** The state complexity of a unary automatic linear order \(L\) is less than \(2m_L^2 + k_L^2 + 2k_Lm_L + k_L\) and more than \(2m_L^2 - k_L^2 + k_L\).

**Proof.** We use Lemma 2 to specify the \((1, 1)\)-states of the optimal automaton. For details, see Appendix B. \(\text{⊓ ⊔}\)

**Corollary 2.** The (unary) state complexity for the class of unary automatic linear orders is quadratic in the size of the associated parameter.

### 4 Equivalence Relations

A structure \(E = (\mathbb{N}; E)\) is an equivalence structure if \(E\) is an equivalence relation (reflexive, symmetric, and transitive). Table 1 immediately gives that the membership problem for automatic equivalence structures is decidable in time \(O(n^3)\). Blumensath [1] and Khoussainov and Rubin [13] described the structure of unary automatic equivalence structures.

**Theorem 7 ([1, 13]).** An equivalence structure has a unary automatic presentation if and only if it has finitely many infinite equivalence classes and there is a finite bound on the sizes of the finite equivalence classes.

The height of an equivalence structure \(E\) is a function \(h_E : \mathbb{N} \cup \{\infty\} \rightarrow \mathbb{N} \cup \{\infty\}\) where \(h_E(x)\) is the number of \(E\)-equivalence classes of size \(x\). Two equivalence structures \(E_1\) and \(E_2\) are isomorphic if and only if \(h_{E_1} = h_{E_2}\). By Theorem 7, the function \(h_E\) for unary automatic equivalence structure \(E\) is finitely nonzero.

**Corollary 3.** The isomorphism problem for unary automatic equivalence structures is decidable.

This corollary may be proved by defining an extended first order sentence which describes the height function \(h_E\). This sentence contains two alternations of quantifiers and leads to doubly exponential runtime for deciding whether two equivalence structures are isomorphic (see Appendix C). We significantly improve this bound via a quadratic time algorithm for the isomorphism problem.
4.1 Efficient solution to the isomorphism problem

**Theorem 8.** The isomorphism problem for unary automatic equivalence structures is decidable in quadratic time in the sizes of the input automata.

**Proof (Proof of Theorem 8).** If $E$ is recognized by a unary automaton $A$ with $n$ states, we will extract $h_E$ in $O(n^2)$ time. Recall the definitions of $t$, $\ell$, and $q_j$ from Section 2. Observe that each $j < t + \ell$ belongs to an infinite equivalence class if and only if there is an accepting state on the $(\cdot, 1)$ loop from $q_j$. Let $t \leq j < t + \ell$. If $j$ belongs to an infinite equivalence class then the set $\{j + i\ell\}_{i \in \mathbb{N}}$ is partitioned into $c$ infinite equivalence classes for some $c > 0$. It takes $O(n^2)$ time to compute the total number of infinite equivalence classes.

Each $0 \leq j < t + \ell$ such that $q_j$ has no accepting state on its $(\cdot, 1)$-loop may be responsible for infinitely many finite equivalence classes of the same size and finitely many other equivalence classes. We can find all $k$ such that $h_E(k) = \infty$ in $O(n)$ time. It remains to compute the sizes of equivalence classes for elements represented on the $(1, 1)$-loop but such that $h_E(k) < \infty$. This can be done by reading through the $(\cdot, 1)$-tails off the $(1, 1)$-tail and has runtime $O(n)$. The full proof (using various auxiliary lemmas) appears in Appendix C. □

4.2 State complexity

The height function $h_E$ is a finite isomorphism invariant for unary automatic equivalence structures. We will express the state complexity in terms of the height function $h_E$; let $h_{inf} = h_E(\infty)$ and $n_{inf} = \sum_{n: h_E(n) = \infty} n$. Define

$$|h_E| = \sum_{n: h_E(n) < \infty} nh_E + n_{inf} + h_{inf}.$$ 

**Theorem 9.** The state complexity of any unary automatic equivalence structures $E = (\mathbb{N}; E)$ is at least

$$\sum_{n: h_E(n) < \infty} n^2 h_E(n) + 2h_{inf}(n_{inf} + 1) + n_{inf} + 1$$

and at most

$$\sum_{n: h_E(n) < \infty} n^2 h_E(n) + \sum_{n: h_E(n) = \infty} n^2 + 2h_{inf}(n_{inf} + 1) + 1.$$ 

**Proof.** The upper bound is obtained by analysing the relationship between infinite equivalence classes, finite equivalence classes that occur infinitely often, and the $(1, 1)$-loop of the optimal automaton. The lower bound is computed by overlapping $(1, 1)$-states so they correspond to multiple equivalence classes. See Appendix C for details.

**Corollary 4.** The (unary) state complexity for the class of unary automatic equivalence structure is quadratic in the height function. □
5 Trees

5.1 Characterizing unary automatic trees

A tree is $T = (\mathbb{N}; \preceq_T)$ where $\preceq_T$ is a partial order (reflexive, antisymmetric, and transitive) with a root (the least element) and the set of ancestors of any node $x$, $\{y : y \preceq_T x\}$, is a finite linear order. Two nodes $x, y$ are incomparable, $x \nmid_T y$, if $x \not\preceq_T y$ and $y \not\preceq_T x$; an anti-chain of $T$ is a set of nodes which are pairwise incomparable. Table 1 gives an $O(n^3)$ algorithm to check if a given automatic relation is a partial order. Checking if $\preceq_T$ is total on every set of predecessors takes time $O(n^4)$. In Appendix D, we give an $O(n)$ time algorithm checking if a unary automatic partial order $(\mathbb{N}; \preceq_T)$ has a least element. Thus, the membership problem for unary automatic trees is decidable in time $O(n^4)$.

As we saw in previous sections, a good characterization of a class of unary automatic structures may lead to a better understanding of its complexity bounds.

We present such a characterization of unary automatic trees. A parameter set $\Gamma$ is a tuple $(\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_m, \sigma, \Xi)$ where $\mathcal{T}_0, \ldots, \mathcal{T}_m$ are finite trees (with disjoint domains $\mathcal{T}_i$), $\sigma : \{1, \ldots, m\} \to \mathcal{T}_0$ and $\Xi : \{1, \ldots, m\} \to \emptyset \cup \bigcup_i \mathcal{T}_i$ such that $X(i) \in T_i \cup \{\emptyset\}$.

**Definition 2.** A tree-unfolding of a parameter set $\Gamma$ is the tree $UF(\Gamma)$ defined as follows:

- $UF(\Gamma)$ contains one copy of $\mathcal{T}_0$ and infinitely many copies of each $\mathcal{T}_i$ ($1 \leq i \leq m$), $(\mathcal{T}_i^j)_{j \in \omega}$. If $x \in \mathcal{T}_i$, its copy in $\mathcal{T}_i^j$ is denoted by $(x, j)$
- For $1 \leq i \leq m$, if $X(i) \neq \emptyset$, the root of $\mathcal{T}_i^0$ is a child (immediate descendent) of $\sigma(i)$, and the root of $\mathcal{T}_i^{j+1}$ is a child of $(X(i), j)$ for all $j$.
- For $1 \leq i \leq m$, if $X(i) = \emptyset$, the root of $\mathcal{T}_i^j$ is a child of $\sigma(i)$ for all $j$.

**Theorem 10.** A tree $T$ is unary automatic if and only if there is a parameter set $\Gamma = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_m, \sigma, \Xi)$ such that $T \cong UF(\Gamma)$.

Suppose $T = (\mathbb{N}; \preceq_T)$ is recognized by a unary automaton $A$ with $n$ states and parameters $t, \ell$. We say that two disjoint sets $X$ and $Y$ of nodes in $T$ are incomparable if $(\forall x \in X)(\forall y \in Y)(x \nmid_T y)$. We sketch the proof of Theorem 10; the full proof is in Appendix D.

**Lemma 3.** For $t \leq j < t + \ell$, the set $(j + it)_{i \in \mathbb{N}}$ forms either an anti-chain or finitely many pairwise incomparable infinite chains in $T$.

Let $A = \{j : (j + it)_{i \in \mathbb{N}}\}$ is an anti-chain, $t \leq j < t + \ell$ and $C = \{t, \ldots, t + \ell - 1\}$ $- A$. For each $j \in C$, let $n_j$ be the number of infinite chains in $(j + it)_{i \in \mathbb{N}}$. For $0 \leq m < n_j$, we denote the infinite chain formed by $(j + (m + i)n_j)\ell_{i \in \mathbb{N}}$ by $W_{j,m}$. $W_{j,m}$ and $W_{k,m'}$ may belong to the same infinite path in $T$ (**interleave** in the sense of Section 3); if they do, then $n_j = n_k$. Any infinite path through $T$ must be given by element(s) in $C$. Hence, $T$ contains only finitely many infinite paths. We define a component of $T$ to be a connected subgraph of $T$ which contains exactly one infinite path and such that all the elements in the subgraph
are greater than or equal to \( t \). For \( j \in C \) and \( k \in A \), we can decide if any element in \((k+i\ell)_{i\in\mathbb{N}}\) belongs to a component of \( T \) intersecting with \((j+i\ell)_{i\in\mathbb{N}}\); if some element does, then \((k+i\ell)_{i\in\mathbb{N}}\) belongs to the same components as the class \((j+i\ell)_{i\in\mathbb{N}}\). If \( j, k \in A \) and neither \((j+i\ell)_{i\in\mathbb{N}}\) nor \((k+i\ell)_{i\in\mathbb{N}}\) intersects with any component of \( T \), we check if the union \((j+i\ell)_{i\in\mathbb{N}}\cup(k+i\ell)_{i\in\mathbb{N}}\) is a subset of infinitely many disjoint finite subtrees in \( T \), each of which contains the nodes \( j+i\ell \) and \( k+(i+m)\ell \) for some \( i \). We call these disjoint finite trees independent.

The above argument facilitates the definition of an equivalence relation \( \sim \) on \( \{t, \ldots, t+s-1\} \) as \( j \sim k \) if and only if

1. \( j \in C \) (or \( k \in C \)) and \((j+i\ell)_{i\in\mathbb{N}}\) and \((k+i\ell)_{i\in\mathbb{N}}\) belong to the same \( n_j \) (or \( n_k \)) components in \( T \); or,
2. \( j, k \in A \) and there is \( h \in C \) such that \( j \sim h \) and \( k \sim h \);
3. \( j, k \in A \) and \((j+i\ell)_{i\in\mathbb{N}}\) and \((k+i\ell)_{i\in\mathbb{N}}\) belong to the same collection of independent trees in \( T \).

We use \([j]\) to denote the \( \sim \)-equivalence class of \( j \).

**Proof (Theorem 10).** We now show that any unary automatic tree is isomorphic to the tree-unfolding \( \text{UF}(\Gamma) \) of some parameter set \( \Gamma = (T_0, T_1, \ldots, T_m, \sigma, X) \). Each \( \sim \)-equivalence class \([j]\) either represents infinitely many independent trees or finitely many components of \( T \). Components of \( T \) represented by \([j]\) are pairwise isomorphic and can be described by “unfolding” a finite graph of size \([|j|]\). In either case, the set of ancestors of \([j]\) in \( T \) is finite. This description may be translate to a parameter set for \( \Gamma \) (see Appendix D).

Conversely, suppose that \( \Gamma = (T_0, T_1, \ldots, T_m, \sigma, X) \) is a parameter set. Let \( t = |T_0|, \ell = \sum_{r=1}^{m} |T_r| \) and \( \alpha_r = \sum_{i=1}^{\ell-1} |T_i| \) for \( r = 1, \ldots, m \). We consider the isomorphic copy \( (\mathbb{N}; \leq_T) \cong \text{UF}(\Gamma) \) where \( T_0 \mapsto \{0, \ldots, |T_0|\} \) and the \( j \)th copy of \( T_r \) maps to \( \{t+(j-1)\ell+\alpha_r, \ldots, t+(j-1)\ell+\alpha_{r+1}-1\} \). The appropriate unary automaton recognizing \( \text{UF}(\Gamma) \) will have parameters \( t, \ell \); the \((\emptyset, 1)\) states, \((1, \emptyset)\) states, and \( F \) can be deduced from \( \sigma \) and \( X \).

\[ \square \]

### 5.2 Efficient solution to the isomorphism problem

Two tree-unfoldings may be isomorphic even if the associated parameter sets are not isomorphic term-by-term. We define a restrict parameter set which will, in fact, be a canonical isomorphic invariant for unary automatic trees. Fix a computable linear order \( \preceq \) on the set of finite trees.

**Definition 3.** The canonical parameter set of a unary automatic tree \( T = (\mathbb{N}; \leq_T) \) is the parameter set \( \Gamma = (T_0, T_1, \ldots, T_m, \sigma, X) \) such that \( \text{UF}(\Gamma) \cong T \) and which is minimal in the following sense:

1. As finite trees, \( T_1 \preceq \ldots \preceq T_m \).
2. If \( T_i \cong T_j \), \( \sigma(i) = \sigma(j) \), and \( X(i) = X(j) = \emptyset \) then \( i = j \).
3. Each \( T_i (1 \leq i \leq m) \) is minimal: If \( X(i) \neq \emptyset \) then if \( y_1 \preceq_T y_2 \leq_T X(i) \) the subtree with domain \( \{z : y_1 \preceq_T z \land y_2 \not\preceq_T z\} \) is not isomorphic to the subtree with domain \( \{z : y_2 \preceq_T z \land X(i) \not\preceq_T z\} \).

9
4. \( T_0 \) is minimal: \( T_0 \) has the fewest possible nodes and for all \( 1 \leq i \leq m \) where \( \text{X}(i) \neq \emptyset \), there is no \( y \in T_0 \) such that \( y \leq_T \sigma(i) \) and the subtree with domain \( \{ z : y \leq_T z \wedge \sigma(i) \not< T z \} \) is isomorphic to \( T_i \).

**Lemma 4.** Suppose \( T, T' \) are unary automatic trees with canonical parameter sets \( \Gamma, \Gamma' \). Then, \( T \cong T' \) if and only if \( \Gamma, \Gamma' \) have the same number \( (m) \) of finite trees, \( (T_0, \sigma) \cong (T'_0, \sigma') \), and for \( 1 \leq i \leq m \), \( (T_i, \text{X}(i)) \cong (T'_i, \text{X}'(i)) \). \( \square \)

The canonical parameter set can now be used to define an extended first-order formula \( \varphi_T \) which specifies the isomorphism type of \( T \) (as in Corollaries 1 and 3). This is sufficient to prove that the isomorphism problem for unary automatic trees is decidable. However, the following results will significantly improve the time complexity of the associated decision procedure.

**Theorem 11.** The isomorphism problem for unary automatic trees is decidable in time \( O(n^4) \) in the sizes of the input automata.

Suppose we can compute the canonical parameter set of a tree from a unary automaton. Given two unary automatic trees, we could use Lemma 4 and a decision procedure for isomorphism on finite trees to solve the isomorphism problem on unary automatic trees. The proof of the following lemma is quite long and may be found in Appendix D.

**Lemma 5.** If \( \leq_T \) is recognized by unary automaton with \( n \) states, there is an \( O(n^4) \) time algorithm that computes the canonical parameter set of \( T \).

**Proof (Theorem 11).** Suppose \( T_1, T_2 \) are presented by unary automata \( A_1, A_2 \) with \( n_1, n_2 \) states (respectively). Let \( n = \max\{n_1, n_2\} \) By Theorem 10 and Lemma 5, deciding if \( T_1 \cong T_2 \) reduces to checking finitely many isomorphisms of finite trees. The appropriate canonical parameter sets are built in \( O(n^4) \) time and each have \( O(n^2) \) finite trees, each of size \( O(n) \). Hence, this isomorphism algorithm runs in \( O(n^4) \) time. \( \square \)

### 5.3 State complexity

Suppose \( T = \text{UF}(\Gamma) \) and \( \Gamma = (T_0, T_1, \ldots, T_m, \sigma, \text{X}) \) is the canonical parameter set of \( T \). Let \( t = |T_0| \) and \( \ell = \sum_{i=1}^m |T_i| \).

**Theorem 12.** The state complexity of unary automatic tree \( T \) is less than \( (t + \ell)^2 - t\ell + t + \ell \) and greater than \( \ell^2 \).

**Proof.** We use the automaton built in the proof of Theorem 10 for the upper bound. For the lower bound, note that the least number of states is obtained when \( T_1, \ldots, T_\ell \) are pairwise nonisomorphic. \( \square \)

**Corollary 5.** The (unary) state complexity of a unary automatic tree \( T \) is quadratic in the parameters \( t, \ell \) of its canonical parameter set.

10
6 Graphs of Finite Degree

A graph \( G = (\mathbb{N}; R) \) is of finite degree if \( (\forall x)(\neg\exists^\infty y)(R(x, y) \lor R(y, x)) \). If \( R \) is recognized by a unary automaton, \( G \) is of finite degree if and only if there is no accepting state on any \((\infty, 1)\)- or \((1, \infty)\)-loops. Therefore, the membership problem is decidable in linear time. In [8], Khoussainov, Liu, and Minnes investigated a range of algorithmic properties of unary automatic graphs of finite degree. For example, they showed that the reachability problem for unary automatic graphs of finite degree can be decided in polynomial time in the sizes of the input vertices and the automaton. We will make use of the following.

**Theorem 13 ([8]).** For a unary automatic graph \( G \) of finite degree, we can construct a unary automaton that recognizes the reachability relation on \( G \) in polynomial time; connectedness of \( G \) is decidable in \( O(n^3) \) time.

However, [8] left open the decidability of the isomorphism problem. We now settle this question and provide an algorithm deciding the isomorphism problem for unary automatic graphs of finite degree.

**Theorem 14.** The isomorphism problem for unary automatic graphs of finite degree is decidable in elementary time.

For brevity, we assume \( G \) is undirected and sketch the proof (the details are in Appendix E). We use the following characterization from [8]. Given a finite graph \( F = (V_F; E_F) \) and a map \( \sigma : V_F \to P(V_F) \), \( F_\sigma \) to be the disjoint union of infinitely many copies of \( F \) with added edges: there is an edge between \( x \in F_i \) and \( y \in F_{i+1} \) if and only if \( y \in \sigma(x) \).

**Theorem 15 ([8]).** A graph of finite degree \( G = (\mathbb{N}; R) \) is unary automatic if and only if there are finite graphs \( D, F \) and a map \( \sigma : V_F \to P(V_F) \) such that \( G \cong G' \) and \( G' \) is a disjoint union of \( D \) and \( F_\sigma \) with possible additional edges between \( D \) and \( F_0 \). Furthermore, the parameters \( D, F, \sigma \) can be extracted in time \( O(n^3) \) from a unary automaton recognizing \( R \).

A *component* of \( G \) is the transitive closure of a vertex under the edge relation. By Theorem 15, any infinite component in \( G \) has nonempty intersection with almost all \( F_i \). Therefore, \( G \) has at most \( |V_F| \) many infinite components. Similarly, any finite component of \( G \) has size at most \( |V_D + V_F| \). Let \( G_{\text{Fin}} \) be the subset of \( G \) containing only its finite components. By Theorem 15, if \( C \) is any finite component of \( G \) then either \( C \cap F_j \neq \emptyset \) for some \( j < \ell \) or \( C \) has infinitely many isomorphic copies in \( G \). Moreover, there are only finitely many isomorphism classes of finite components of \( G \), and we can decide which of these classes correspond to infinitely many components in \( G \). Using the decidability of finite graph isomorphism, given two graphs \( G, G' \) we can decide if \( G_{\text{Fin}} \cong G'_{\text{Fin}} \).

Since \( G \) contains only finitely many infinite components, it remains to prove that, given two infinite components of unary automatic graphs, we can check if they are isomorphic. Note that each infinite component of \( G \) is recognizable by a unary automaton using operations on the automaton from Theorem 13.
Therefore, it suffices to prove that we can decide whether two infinite connected unary automatic graphs are isomorphic.

We will give sufficient conditions (expressible in MSO) which guarantee that an infinite connected graph $H$ will be isomorphic to $G$. Take a subgraph $M$ of $3k$ vertices in $H$. We say that $M$ is a $F_{x3}$-type if $M$ intersects with each $P_i$ at exactly one vertex, and if we let $v_i = M \cap P_i$, then the three sets of vertices $\{v_1, \ldots, v_k\}$, $\{v_{k+1}, \ldots, v_{2k}\}$, $\{v_{2k+1}, \ldots, v_{3k}\}$ respectively form three copies of $F$, with $v_i, v_{k+i}, v_{2k+i}$ corresponding to the same vertex in $F$. Also, the edge relation between these three copies of $F$ must respect the mapping $\sigma$. Suppose each vertex $v$ in $H$ belongs a unique subgraph that is a $F_{x3}$-type; and, for each $F_{x3}$-type $M$, there is a unique $F_{x3}$-type $N$ that is a successor of $M$, i.e., all edges between $M$ and $N$ are from the last copy of $F$ in $M$ to the first copy of $F$ in $N$ such that they respect the mapping $\sigma$. Suppose also that there exists a unique $F_{x3}$-type $M_0$ which is not the successor of any other $F_{x3}$-types and any other $F_{x3}$-type is the successor of a unique $F_{x3}$-type. Note that the successor relation between the $F_{x3}$-types resembles the unfolding operation on finite graphs.

These conditions on $H$ are expressible in MSO and lead to an isomorphism: map $M_0$ isomorphically to the first 3 copies of $F$ in $G$, and then map the other vertices according to the successor relation and $\sigma$. The MSO formula is written out in Appendix E. By Theorem 2, satisfiability of an MSO sentence is decidable for unary automatic graphs. Therefore the isomorphism problem for unary automatic graphs of finite degree is decidable. Note that formalizing the conditions on $H$ requires only finitely many alternations of quantifiers (regardless of the size of the automaton presenting it) and so the decision procedure is elementary in terms of the size of the input automaton.
References

A Proofs for Section 2: Preliminaries

A.1 Deciding properties of automatic binary relations

To check if \( R \) is reflexive, we construct an automaton for \( \{ x : (x,x) \in R \} \) and check if \( \{ x : (x,x) \in R \} \cap D = D \). Similarly, to decide if \( R \) is symmetric, we construct an automaton \( A_1 \) recognizing the relation \( \{ (y,x) : (x,y) \in R \} \) and check if \( R = L(A_1) \). For antisymmetry, we construct an automaton for \( S = \{ (x,y) : x \neq y \} \) and determine whether \( R \cap R_1 \cap S = \emptyset \). To decide if \( R \) is total, it suffices to check whether \( R \cup L(A_1) = D \). Finally, to settle whether \( R \) is transitive, we construct the automaton \( \{ (x,y,z) : R(x,y) \& R(y,z) \& \neg R(x,z) \} \) and ask whether its language is empty.

A.2 The shape of unary automata

![General shape of a deterministic 2-tape unary automaton](image)

**Fig. 1.** General shape of a deterministic 2-tape unary automaton

**Lemma 1.** \([1]\) A set \( L \subseteq \mathbb{N} \) is unary automatic if and only if there are \( t_L, \ell_L \in \mathbb{N} \) such that \( L = L_1 \cup L_2 \) with \( L_1 \subseteq \{ 0, 1, \ldots, t_L - 1 \} \) and \( L_2 \) is a finite union of sets in the form \( \{ j + i \ell_L \}_{i \in \omega} \) where \( t_L \leq j < t_L + \ell_L \).

**Proof.** We describe the shape of an arbitrary deterministic 1-tape unary automaton \( \mathcal{A} = (S, \iota, \Delta, F) \). If \( n = |S| \) there are \( t, \ell \leq n \) so that the following holds. There is a sequence of states \( S_1 = \{ q_1, q_2, \ldots, q_t \} \) such that \( \Delta(t, 1) = q_1 \) and for all \( 1 \leq i < t, \Delta(q_i, 1) = q_{i+1} \). There is another sequence of states \( S_2 = \{ q_{t+1}, \ldots, q_{t+\ell} \} \) such that for all \( t \leq j < l, \Delta(q_j, 1) = q_{j+1} \), and \( \Delta(q_l, 1) = q_{l+1} \). Every final state in \( S_1 \) recognizes exactly one word less than \( t \), and every final state in \( S_2 \) recognizes the set of all words \( t + i + k, i \in \omega \), for some fixed \( k < l \). The language of such an automaton has the form described in the statement of the lemma; given an \( L \) from the statement of the lemma and its parameters \( t, \ell \), we can define the corresponding unary automaton. \( \square \)

**Lemma 6.** Let \((D; R), D \subseteq \mathbb{N} \) be a unary automatic binary relation presented by \( \mathcal{A}_D \) and \( \mathcal{A}_R \). There is a deterministic 2-tape unary automaton \( \mathcal{A}_R' \), \( |\mathcal{A}_R'| \leq |\mathcal{A}_R| \), such that \((\mathbb{N}; L(\mathcal{A}_R')) \cong (D; R)\).
Proof. Let \( t \) and \( \ell \) be as described in Lemma 1. We outline the proof in the case when the parameter \( t \) associated with \( D \) is 0. Since \( R \) is a binary relation over the domain \( D \), \( A_R \) must satisfy the following requirements: the \((1,1)\)-tail has length \( c' \ell \) for some constant \( c' \); the \((1,1)\)-loop has length \( c \ell \) for some constant \( c \); the lengths of all loops and tails containing accepting states are multiples of \( \ell \); and, there are no accepting states on any tail or loops off any \((1,1)\)-states of the form \( q_i \ell + h \) where \( h \neq k_j \) (where \( k_j \) is as defined in Lemma 1). The isomorphism between \( D \) and \( N \) will be given by \( i \ell + k_j \mapsto i \ell + j \). Therefore, define \( A_R' \) to have a \((1,1)\)-tail of length \( c' r \), a \((1,1)\)-loop of length \( c r \), and copy the information from the state \( i \ell + j \) in \( A_R \) to state \( i \ell + j \) in \( A_R' \) (modifying the lengths of \((\cdot, \cdot)\)-tails and loops appropriately). Then, \((N; L(A_R')) \cong (D; R)\) and since \( r \leq \ell \), \( A_R' \) has no more states than \( A_R \). \( \square \)

B Proofs for Section 3: Linear Orders

B.1 Defining \( \alpha_L \) in extended first order logic

Corollary 1 The isomorphism problem for unary automatic linear orders is decidable.

Proof. Let \( \mathcal{L} = (\mathbb{N}; \leq) \) be a unary automatic linear order. We will define \( \varphi_L \) such that a linear order \( \mathcal{L}_1 \) is isomorphic to \( \mathcal{L} \) if and only if \( \mathcal{L}_1 \models \varphi_L \). To do so, we define the following auxiliary formulas. For \( x, y \in \mathbb{N} \), let \( \text{FinDis}(x, y) \) be

\[ x < L y \land \neg(\exists^\infty z)[x < L z \land z < L y] \]

For \( x \in \omega \), let \( \text{In}^\omega(x) \) be the formula

\[ (\exists^\infty y)[x < L y \land \text{FinDis}(x, y)] \land (\forall z < L x)[\neg \text{FinDis}(z, x)] \]

Let \( \text{In}^\omega^*(x) \) be the formula

\[ [(\exists^\infty y)y < L x \land \text{FinDis}(y, x)] \land (\forall z > L x)[\neg \text{FinDis}(x, z)] \]

Let \( \text{In}^Z(x) \) be the formula

\[ (\exists^\infty y)[x < L y \land \text{FinDis}(x, y)] \land [(\exists^\infty z)z < L x \land \text{FinDis}(z, x)] \]

For any \( n \in \omega \), let \( \text{In}^n(x) \) be the formula

\[
(\exists y_1, \ldots, y_{n-1})[x < L y_1 \land \bigwedge_{i=1}^{n-2}(y_i < L y_{i+1}) \land (\forall z)[\neg \text{FinDis}(z, x)] \land (\forall z)[\text{FinDis}(x, z) \rightarrow z = x \lor \bigvee_{i=1}^{n-1}(z = y_i)]
\]

15
By Theorem 4, \( \mathcal{L} \) can be uniquely described up to isomorphism by its canonical word \( \alpha_{\mathcal{L}} = \alpha_0 \cdots \alpha_{2^n-1} \in \{ \omega, \omega^*, \zeta, (n)_{n \in \omega} \}^* \), where \( \alpha_{\mathcal{L}} \) has no substring of the form \( \omega^* \omega \), \( n \omega \) or \( \omega^* n \). Hence, we define \( \varphi_\mathcal{L} \) as follows

\[
(\exists x_0, \ldots, x_{k-1}) \left[ \bigwedge_{i=0}^{k-2} (x_i <_L x_{i+1}) \land \bigwedge_{i=0}^{k-1} \text{In}^{\alpha_i}(x_i) \land (\forall y) \bigvee_{i=0}^{k-1} (\text{FinDis}(x_i, y) \lor \text{FinDis}(y, x_i)) \right]
\]

\[\square\]

B.2 Efficient algorithms for the isomorphism problem

**Lemma 2.** Suppose \( A \) is a unary automaton that represents a linear order \( \mathcal{L} = (\mathbb{N}; \leq_L) \). For any \( t \leq j < \ell \), the sequence \( (j+i\ell)_{i \in \mathbb{N}} \) is either an increasing chain in a copy of \( \omega \) in \( \mathcal{L} \) or a decreasing chain in a copy of \( \omega^* \) in \( \mathcal{L} \).

**Proof.** If \( \Delta(q_j, (1,0)^\ell) \in F \) then \( j + i\ell <_L j + (i + 1)\ell \) for all \( i \) so \( (j + i\ell)_{i \in \mathbb{N}} \) is an increasing chain in \( \mathcal{L} \). Otherwise, totality of \( \mathcal{L} \) implies that \( \Delta(q_j, (1,0)^\ell) \in F \) hence \( (j + i\ell)_{i \in \mathbb{N}} \) is a decreasing chain in \( \mathcal{L} \).

Suppose \( (j + i\ell)_{i \in \mathbb{N}} \) forms an increasing chain. Let \( t_1, t_2 \) be the lengths of the \( (1,1-) \) and \( (1,0-) \) tails off \( q_j \); let \( \ell_1, \ell_2 \) be the lengths of the \( (0,1-) \) and \( (1,0-) \) loops off \( q_j \). We consider two cases. First, suppose \( \ell_1 = \ell_2 = 1 \). Since \( (j + i\ell)_{i \in \mathbb{N}} \) is increasing, \( \Delta(q_j, (1,0)^\ell) \in F \) for all \( c \). Hence, \( \Delta(q_j, (1,0)^{\ell_1}) \in F \).

Similarly, \( \Delta(q_j, (1,0)^{\ell_2}) \notin F \). Therefore, each \( x > j + (i + 1)\ell \) satisfies \( x >_L j + (i + 1)\ell \) and there are only finitely many elements \( <_L \)-below \( j + (i + 1)\ell \). This leaves only finitely many possible elements \( <_L \)-between \( j + i\ell \) and \( j + (i + 1)\ell \).

On the other hand, suppose \( \ell_1 \ell_2 > 1 \). Let \( k = \max\{t_1, t_2\} + 1 \). Suppose there is \( i \geq 0 \) and \( r = j + i\ell + k + s \), \( s \geq 0 \) such that

\[
j + i\ell <_L r <_L j + (i + 1)\ell.
\]

The first inequality is equivalent to \( \Delta(q_j, (1,0)^{k+s}) \in F \) and hence for any \( c \), \( j + i\ell + c\ell <_L r + c\ell \). The second inequality implies that \( \Delta(q_j, (1,0)^{k+s-1}) \in F \) and is in the \( (1,0) \)-loop off \( q_j \). So, for any \( c' \geq 0 \), \( r + c'\ell_2 <_L j + (i + 1)\ell \).

Therefore,

\[
j + i\ell + (\ell_1\ell_2)\ell <_L r + (\ell_1\ell_2)\ell = r + (\ell_1\ell)\ell_2 <_L j + (i + 1)\ell.
\]

This is a contradiction because \( (j + i\ell)_{i \in \mathbb{N}} \) is increasing whereas \( \ell_1 \ell_2 > 1 \). Thus, any element \( <_L \)-between \( j + i\ell \) and \( j + (i + 1)\ell \) must be smaller than \( r \); there are only finitely many such elements. \( \square \)

**Theorem 5.** The isomorphism problem for unary automatic linear orders is decidable in quadratic time in the sizes of the input automata.
Proof. Suppose \( \mathcal{L} = (\mathbb{N}; <_L) \) is a unary automatic linear order represented by a unary automaton \( A \) with parameters \( t, \ell \). We will extract its canonical word \( \alpha_{\mathcal{L}} \in \{\omega, \omega^*, \mathbf{n}\}^* \) by determining the relative ordering of the at most \( \ell \) many copies of \( \omega \) and \( \omega^* \) given by Lemma 2 and the elements 0, \ldots, \( t - 1 \).

For \( t \leq j < t + \ell \), we can use Lemma 2 to decide in linear time whether the sequence \((j + i\ell)_{i \in \mathbb{N}}\) is in a copy of \( \omega \) or \( \omega^* \). We say two sequences \((j + i\ell)_{i \in \mathbb{N}}\) and \((k + i\ell)_{i \in \mathbb{N}}\) \((j < k)\) interleave if they belong to the same copy of \( \omega \) or \( \omega^* \) in \( \mathcal{L} \). Given \( j < k \), the corresponding sequences interleave if and only if

- both are increasing, there is \( c_1 \) larger than the length of the \((\cdot, 1)\)-tail off \( q_j \)
- and \( c_2 \) larger than the length of the \((\cdot, 1)\)-tail off \( q_k \) satisfying \( \Delta(q_j, (\cdot, 1)^{(c_1)}) \)
- and \( \Delta(q_k, (\cdot, 1)^{(c_2)}) \) are both in \( F \), \( c_1 \equiv (k - j) \mod \ell \), \( c_2 \equiv (j - k) \mod \ell \);
- or, both sequences are decreasing chains and the above conditions hold when we substitute the \((1, \cdot)\) states for the \((\cdot, 1)\) ones.

To check if the above conditions hold, we can read the states \( \Delta(q_j, (\cdot, 1)^{k-j+d_1\ell}) \), \( \Delta(q_k, (\cdot, 1)^{j-k+d_2\ell}) \) for increasing values of \( d_1, d_2 \) until we either find an appropriate state or loop back to a state we already visited. Thus, we read at most as many states as there are in the \((\cdot, 1)\) loops (thus, fewer than \( n \)) as many as \( \ell \) times each. Hence, deciding interleaving takes \( O(n^2) \) time.

Notice that while considering the cases above, we also consider the relative order of \((j + i\ell)_{i \in \mathbb{N}}\) and \((k + i\ell)_{i \in \mathbb{N}}\) if they do not interleave. That is, \((j + i\ell)_{i \in \mathbb{N}} <_L (k + i\ell)_{i \in \mathbb{N}}\) if and only if there is \( c \) greater than the length of the \((\cdot, 1)\)-tail off \( q_j \) such that \( \Delta(q_j, (\cdot, 1)^{(c)}) \in F \) and \( c \equiv (k - j) \mod \ell \). Thus, in \( O(n^2) \) time, we can partition \( \{t, \ldots, t + \ell - 1\} \) into \( V_0, \ldots, V_s \) such that if \( j < k \) and \( j, k \in V_l \) \((0 \leq \ell \leq s)\) then \((j + i\ell)_{i \in \mathbb{N}}\) and \((k + i\ell)_{i \in \mathbb{N}}\) interleave. We simultaneously define, for \( t \leq j < t + \ell \), \( \text{Left}(j) = \{j : (k + i\ell)_{i \in \mathbb{N}} <_L (j + i\ell)_{i \in \mathbb{N}}\}\).

It remains to define \( \text{Left}(j) \) for \( 0 \leq j < t \). If there is \( t \leq k < t + \ell \) and \( d_1, d_2 \) such that \( \Delta(q_j, (\cdot, 1)^{(d_1)}) \in F \) and \( \Delta(q_k, (\cdot, 1)^{(d_2)}) \in F \) and \( d_1 \equiv d_2 \equiv (k - j) \mod \ell \) then \( j < L \)-between elements of \((k + i\ell)_{i \in \mathbb{N}}\) and we will determine its representative’s position in \( \alpha_{\mathcal{L}} \) when we consider \( q_k \). Therefore, in this case, leave \( \text{Left}(j) \) undefined. Otherwise, the elements \( <_L \)-below \( j \) are obtained by examining the \((1, \cdot)\)-tail and -loop of \( q_j \) and the \((\cdot, 1)\)-tails and loops of \( q_j' \) for \( j' < j \) and putting \( k \) in \( \text{Left}(j) \) if the state corresponding to it is accepting. We can define all \( \text{Left}(j) \) for \( 0 \leq j < t \) simultaneously in \( O(n) \) time.

We are now ready to give an algorithm for extracting \( \alpha_{\mathcal{L}} \) from \( A \). We iterate through processing each \((1, 1)\) state which has a defined \( \text{Left} \) set. Initialize the set \( B \) to empty and the word \( \alpha = \lambda \). Let \( q_j \) be least state that remains to be processed such that \( \text{Left}(j) = B \). If \( j < t \) add \( j \) to the set \( B \) and update \( \alpha = \alpha 1 \).

However, if \( j \geq t \) then let \( i \) be least such that \( i, j \in V_r \) for some \( r \). If \( (j + i\ell)_{n \in \mathbb{N}} \) is increasing, update \( B = B \cup V_r \) and \( \alpha = \omega \). Otherwise, \((j + i\ell)_{n \in \mathbb{N}} \) is decreasing and we update \( B = B \cup V_r \) and \( \alpha = \omega^* \). Remove \( q_j \) and all states in \( V_r \) from the list of states which remain to be processed. Once we have processed all states, we smooth \( \alpha \): set \( \alpha_L \) to be the result of replacing all \( 1\omega \) in \( \alpha \) to \( \omega \), all \( \omega^*1 \) in \( \alpha \) to \( \omega^* \), and all sequences of \( 1\)s of length \( n \) to \( \mathbf{n} \). This algorithm has runtime quadratic in the size of \( A \). Since any two unary automatic linear orders \( \mathcal{L}_1, \mathcal{L}_2 \) are isomorphic if and only if \( \alpha_{\mathcal{L}_1} = \alpha_{\mathcal{L}_2} \), running this algorithm on each input
automaton and then comparing the results gives a quadratic time solution to the isomorphism problem.

\[ \square \]

B.3 State complexity

**Theorem 6.** The state complexity of a unary automatic linear order \( L = (\mathbb{N}; \leq_L) \) is less than \( 2m_L^2 + k_L^2 + 2k_Lm_L + k_L \) and more than \( 2m_L^2 - k_L^2 + k_L \).

**Proof.** By Lemma 2, the optimal automaton \( A \) for \( L \) has \( m_L + k_L \) states: \( k_L \) many states on the (1,1)-tail and \( m_L \) many states on the (1,1)-loop. Each state on the (1,1)-loop represents a copy of \( \omega \) or \( \omega^* \) in \( L \) and since this is the minimal automaton there is no interleaving. To specify whether each copy of \( \omega \) or \( \omega^* \) is increasing or decreasing and the relative ordering of copies of \( \omega \) and \( \omega^* \), we need \( 2\ell = 2m_L \) (1,0) or (0,1) states off each (1,1)-loop state. To specify the ordering of the singleton elements represented by the states on the (1,1)-tail with respect to each other and to copies of \( \omega, \omega^* \), we need up to \( 2(k_L - j + m_L) \) states off \( q_j \). Therefore, an upper bound to the number of states in an optimal unary automaton is

\[
m_L(2m_L) + \sum_{j=0}^{k_L-1} 2(k_L - j + m_L) = 2m_L^2 + k_L^2 + 2k_Lm_L + k_L.
\]

We can improve this bound by realizing that the states on the (1,1)-loop corresponding to the rightmost and leftmost symbols in \( \alpha_L \) require fewer (0,1) and (1,0) states. The greatest saving of states occurs if the rightmost and leftmost elements of \( \alpha_L \) are finite, say \( a \) and \( b \) and the corresponding elements of \( L \) are represented by the first \( a + b \) (1,1)-tail states. In this case, each of these (1,1)-tail states has only one associated (1,0) or (0,1) state. We therefore save

\[
\sum_{i=0}^{a+b-1} 2(k_L - i + m_L) - 1 \leq 2k_L^2 + 2k_Lm_L
\]

states. Thus, the minimal automaton must have at least

\[
2m_L^2 + k_L^2 + 2k_Lm_L + k_L - 2k_L^2 - 2k_Lm_L = 2m_L^2 - k_L^2 + k_L
\]

states. \[ \square \]

C Proofs for Section 4: Equivalence Structures

C.1 Defining \( h_E \) in extended first order logic

**Corollary 3** The isomorphism problem for unary automatic equivalence structures is decidable.
Proof. For each \( n \in \omega \), define the formula \( \text{size}_n(x) \) as
\[
(\exists y_1 \cdots \exists y_{n-1})(\bigwedge_{i=1}^{n-1} (y_i \neq x \land E(y_i, x)) \land (\forall z)(\bigwedge_{i=1}^{n-1} (z \neq y_i) \land z \neq x) \rightarrow \neg E(x, z)).
\]
By Theorem 7, any unary automatic equivalence structure \( \mathcal{E} \) with height \( h_\mathcal{E} \) can be defined by the sentence \( \varphi_\mathcal{E} \) that is the conjunction of the following, where we let \( H = h_\mathcal{E}(\infty) \) and \( H_n = h_\mathcal{E}(n) \).
\[
(\exists x_1 \cdots \exists x_H) \left[ \bigwedge_{i=1}^H (\exists^\infty y) E(x_i, y) \land \bigwedge_{i,j=1; i \neq j} \neg E(x_i, x_j) \right] \\
\land \forall x((\exists^\infty y)E(x, y) \rightarrow \bigvee_{i=1}^H E(x, x_i)) \land \bigwedge_{n: H_n = \infty} [(\exists^\infty x)\text{size}_n(x)]
\]
\[
\bigwedge_{m,n: H_n = m} (\exists y_1 \cdots \exists y_m) \left[ \bigwedge_{i=1}^m \text{size}_n(y_i) \land \bigwedge_{i,j=1; i \neq j} \neg E(y_i, y_j) \right]
\]
For any equivalence structure \( \mathcal{E}_1, \mathcal{E}_1 \cong \mathcal{E} \) if and only if \( \mathcal{E}_1 \models \varphi_\mathcal{E} \). By Theorem 1, the isomorphism problem is decidable. \( \square \)

C.2 Efficient solution to the isomorphism problem

**Theorem 8** The isomorphism problem for unary automatic equivalence structures is decidable in quadratic time in the sizes of the input automata.

Let \( \mathcal{E} \) be recognized by a unary automaton \( A \) with \( n \) states. Recall the definitions of \( t, \ell, \) and \( q_j \) from Section 2. Observe that each \( j < t + \ell \) belongs to an infinite equivalence class if and only if there is an accepting state on the \((0,1)\) loop from \( q_j \). Let \( t \leq j < t + \ell \). If \( j \) belongs to an infinite equivalence class then for all \( i \in \mathbb{N}, j + i\ell \) is in an infinite equivalence class. By Theorem 7, there are only finitely many infinite equivalence classes in \( \mathcal{E} \). Hence, for some \( i \) and \( k, i \neq k, E(j + i\ell, j + k\ell) \). This means \( \Delta(q_j, (0, 1)^{i+k}) \) is accepting. Let \( c > 0 \) be the least number such that \( \Delta(q_j, (0, 1)^c) \in F \). To compute \( c \), we examine \( \Delta(q_j, (0, 1)^d) \) for increasing values of \( d \) until we find an accepting state or repeat a state. Thus, we need to examine at most as many states as the length of the \((0,1)\)-loop off \( q_j \).

**Lemma 7.** The set \( \{j + i\ell\}_{i \in \mathbb{N}} \) is partitioned into \( c \) infinite equivalence classes.

**Proof.** Since \( c \) is the least such that \( \Delta(q_j, (0, 1)^c) \) is accepting, elements in \( \{j, j + \ell, \ldots, j + (c-1)\ell\} \) are pairwise non-\( E \)-equivalent. Moreover for each \( i, E(j + i\ell, j + (i + c)\ell) \). \( \square \)
We now consider the finite equivalence classes. Given $k$ $(1, 1)$-loop states $q_{j_1}, \ldots, q_{j_k}$ each of which has no accepting state on its $(\varnothing, 1)$-loop, we say that 
\{$q_{j_1}, \ldots, q_{j_k}$\} is a corresponding set if for each $q_{j_i}$ and $s = 1, \ldots, k - i$ there is 
$m_{s+i}^j$ such that the state $r_{i}^j = \Delta(q_{j_i}, (\varnothing, 1)^{(j_{i+1} - j_i) + m_{s+i}^j}) \in F$; moreover, these 
are the only accepting states on the $(\varnothing, 1)$ tail of $q_{j_i}$. A corresponding set is maximal if it is not a subset of a larger corresponding set.

**Lemma 8.** For any $k$, $h_E(k) = \infty$ if and only if there is a maximal corresponding set of size $k$.

**Proof.** If $q_{j_1}, \ldots, q_{j_k}$ form a maximal corresponding set then for each $c \geq 0$, 
\{$j_1 + cf, j_2 + (c + m_1^j)f, \ldots, j_k + (c + m_1^k)f$\} is an $E$-equivalence class of size $k$. On 
the other hand, suppose there are infinitely many $E$-equivalence classes of size $k$. For $t \leq j < t + \ell$, let $\ell_j$ and $t_j$ be the lengths of the $(\varnothing, 1)$-loop and -tail off $q_{j_i}$, respectively. Let $p = \max\{t_j + \ell_j : t \leq j < t + \ell\}$. Consider an equivalence 
class \{$x_1, \ldots, x_k$\} where $p \leq x_i < x_{i+1}$ (for all $1 \leq i < k$). For $1 \leq i < k$ define 
$j_i$ such that $x_i = j_i + m\ell$ for some $m$ and $t \leq j_i < t + \ell$. Then \{$q_{j_1}, \ldots, q_{j_k}$\} is a 
maximal corresponding set. \(\square\)

**Proof (Proof of Theorem 8).** To decide whether two unary automatic equivalence structures $E_1, E_2$ are isomorphic we first use the unary automata recognizing $E_1$ and $E_2$ to compute their height functions and then check if $h_{E_1} = h_{E_2}$. Hence, we 
begin by giving an algorithm for extracting the height function of a unary automatic equivalence structure $E$ from a unary automaton $A$.

For each $0 \leq j < t + \ell$, if $j$ is in an infinite equivalence class then there is 
$t \leq j' < t + \ell$ in the same class and we can find $j'$ in linear time. For $j'$ we compute $c$ such that \{$j' + if$\}_{i \in \mathbb{N}} accounts for exactly $c$ infinite equivalence classes in $E$, as described in Lemma 7. At the same time, we can compute all 
$0 < k < t + \ell$ such that $\Delta(q_{j'}, (\varnothing, 1)^{k-j'} + m\ell) \in F$ (where $m < n$). For any such $k$, 
\{$k+if$\}_{i \in \mathbb{N}} is covered by the $c$ infinite equivalence classes from \{$j' + if$\}_{i \in \mathbb{N}}. Using 
the complexity analysis before Lemma 7, we see that the runtime of computing the total number of infinite equivalence classes is 

\[
O\left( \sum_{j=t}^{t+\ell-1} (\ell_j + t_j) + (t + \ell)(t_j + t_j) \right) = O(n^2).
\]

It remains to consider $0 \leq j < t + \ell$ such that $q_j$ has no accepting state on 
its $(\varnothing, 1)$-loop. Note that each $q_j$ may be responsible for infinitely many finite 
equivalence classes of the same size and finitely many other equivalence classes. 
By Lemma 8, we can effectively find all $k$ such that $h_E(k) = \infty$ by searching for the 
appropriate corresponding set. This can be done by reading the $(\varnothing, 1)$-tails 
of the $(1, 1)$-loop and thus takes times $O(n)$.

By the proof of Lemma 8, for any finite equivalence class $K$, if $x \geq p$ for 
all $x \in K$ and $|K| = k$, then $h_E(k) = \infty$. Hence, it only remains to compute 
the sizes of equivalence classes for elements in \{$t, \ldots, p$\}, which requires reading 
through the $(\varnothing, 1)$-tails off the $(1, 1)$-tail. Again this step has runtime $O(n)$.
In summary, the algorithm that computes $h_E$ from $A$ has runtime $O(n^2)$. Note that the domain of $h_E$ is a subset of $\{1, \ldots, n, \infty\}$ so comparing it with $h_{E'}$ takes linear time. Therefore, the isomorphism problem for unary automatic equivalence relations is solved in quadratic time in the maximum of the sizes of the input automata.

C.3 State complexity

**Lemma 9.** Let $A$ be a unary automaton recognizing $E$, then $n_{\text{inf}} \leq \ell$.

**Proof.** For any $n$, $h_E(n) = \infty$ if and only if there are $t \leq j_1 < j_2 < \cdots < j_n < t + \ell$ such that $\Delta(q_{j_i}, (\omega, 1)^{j_i-j_1}) \in F$ for all $i = 1, \ldots, n$ and no other $(\omega, 1)$ states off $q_i$ are accepting. These $q_{j_i}$ may not be shared among disjoint equivalence classes, hence $n_{\text{inf}} \leq \ell$.

**Theorem 9** The state complexity of any unary automatic equivalence structures $E = (\mathbb{N}; E)$ is at least

$$\sum_{n: h_E(n) < \infty} n^2 h_E(n) + 2h_{\text{inf}}(n_{\text{inf}} + 1) + n_{\text{inf}} + 1$$

and at most

$$\sum_{n: h_E(n) < \infty} n^2 h_E(n) + \sum_{n: h_E(n) = \infty} n^2 + 2h_{\text{inf}}(n_{\text{inf}} + 1) + 1.$$ 

**Proof.** We say a collection of $(1, 1)$-states $\{r_1, \ldots, r_m\}$ in $A$ represents an $E$-equivalence class $K$ if for each $x \in K$, $\Delta(x, (1, 1)^x) = r_i$ for some $1 \leq i \leq m$. Let $K$ be a finite equivalence class. It must be represented by some $\{r_1, \ldots, r_m\}$ where there are $m - i$ accepting states on the $(\omega, 1)$-tail off $r_i$. In an optimal unary automaton recognizing $E$, the length of the $(\omega, 1)$-tails off $r_i$ states is minimized by arranging the $r_1, \ldots, r_j$ consecutively. In this case, the tail off $r_i$ contains $m - i$ states; by symmetry, the number of $(\omega, 1)$ and $(1, \omega)$ states associated to the class $K$ is $2 \sum_{i=1}^{\lfloor \frac{|K|}{2} \rfloor} (|K| - i) = |K|^2 - |K|$. Counting the $(1, 1)$ states representing $K$, there are $|K|^2$ states associated to $K$. Note that in the optimal automaton, if there are infinitely many equivalence classes of the same size, they are all represented by the same $(1, 1)$ states.

If each infinite equivalence class of $E$ is represented by a single state on the $(1, 1)$-loop of an automaton $A$, then $\ell > h_{\text{inf}}$. Moreover, the $(\omega, 1)$-loop out of with each such state must have size at least $\ell$. In this case, $\ell + 1$ states are associated with each infinite equivalence class. One may hope to reduce the number of states by using multiple states $r_1, \ldots, r_k$ to represent an infinite equivalence class $K$. In this case, $k$ must be a divisor of $\ell$ and the $(\omega, 1)$-loop out of each $r_i$ has length $\ell/k$. Thus, at least $k + k(\ell/k) = k + \ell$ states are associated with $K$, which is no improvement. However, we can reduce the number of states by using a single $(1, 1)$-loop state $r$ to represent all the (finitely many) infinite components. To
do so, define a \((\diamond, 1)\)-loop (respectively, \((1, \diamond)\)-loop) out of \(r\) with length \(h_{\text{inf}}\ell\) and a single accepting state, \(\Delta(r, (\diamond, 1)^{h_{\text{inf}}\ell})\). With this representation, \(1 + 2h_{\text{inf}}\ell\) states are used for all the infinite equivalence classes (as opposed to more than \(h_{\text{inf}}^2 + h_{\text{inf}}\)). By Lemma 9 and the above discussion, the smallest possible length for the \((1,1)\)-loop is \((n_{\text{inf}}+1)\). For each \(n\) such that \(h_E(n) = \infty\), there are \(n^2 - n\) \((1, \diamond)\)- and \((\diamond, 1)\)-states off the \((1,1)\)-loop. Thus, there must be at least
\[
1 + 2h_{\text{inf}}\ell + \sum_{n : h_E(n) = \infty} n^2
\]
states on the \((1,1)\)-loop and its peripheries.

We can define an automaton \(A\) which recognizes \(E\) using a \((1,1)\)-tail of length \(\sum_{n : h_E(n) < \infty} nh_E(n)\) and \((1,1)\)-loop of length \(n_{\text{inf}} + 1\). The total size of \(A\) is
\[
\sum_{n : h_E(n) < \infty} n^2 h_E(n) + 1 + 2h_{\text{inf}}(n_{\text{inf}} + 1) + \sum_{n : h_E(n) = \infty} n^2.
\]
An optimal automaton must have at most this many states.

To obtain a lower bound, we note that there may be overlap between states in the \((1,1)\)-loop representing equivalence classes of different sizes, some of which occur infinitely often and others which do not. In particular, this can only occur for \(E\)-equivalence classes \(K, K'\) where \(|K| > |K'|\) and \(h_E(|K|) = \infty\), \(h_E(|K'|) < \infty\). With optimal overlapping, the minimum number of states in a unary automaton recognizing \(E\) is
\[
\sum_{n : h_E(n) < \infty} n^2 h_E(n) + \sum_{n : h_E(n) = \infty} n^2 + 2h_{\text{inf}}(n_{\text{inf}} + 1) + 1 - c
\]
for some \(c\). Moreover, \(c\) is no more than all \((\diamond, 1)\)- and \((1, \diamond)\)-states associated with finite equivalence classes occurring infinitely often, so \(c < \sum_{n : h_E(n) = \infty}(n^2 - n)\), as required.

\section{Proofs for Section 5: Trees}

\subsection{Checking if a unary automatic relation has a least element}

Checking the existence of a root (least element) of an automatic binary relation may take exponential time because of the impact of alternation of quantifiers on the size of the automaton for the query. We improve this exponential bound when \(\leq_T\) is recognized by a unary (rather than arbitrary) automaton.

\begin{lemma}
There is an \(O(n)\) time algorithm that checks if a unary automatic partial order \((\mathbb{N}; \leq_T)\) has a least element.
\end{lemma}

\begin{proof}
Suppose \(\leq_T\) is recognized by unary automaton \(A\) with parameters \(t, \ell\) (as in Section 2). If there is a least element \(x\) then \(x < t + \ell\). Indeed, if \(x \geq t + \ell\), there is \(t \leq y < t + \ell\) such that \(q = \Delta(t, (1,1)^x) = \Delta(t, (1,1)^y)\). By definition of \(x\),
where $x <_T y$ and $x <_T 2x - y$. Therefore, $\Delta(q, (1, 0)^x - y) \in F$ and $\Delta(q, (0, 1)^x - y) \in F$. But, this contradict antisymmetry of $<_T$. Thus, to check for a root it is sufficient to check if each of $\{0, \ldots, t + \ell - 1\}$ is the root and this procedure examines each state of $A$ at most once. \hfill \Box

### D.2 Sample tree unfolding

Below is an example of a tree unfolding. The parameter set is $(T_0, T_1, T_2, T_3, T_4, \sigma, X)$.

![Tree Unfolding Diagram](image)

**Fig. 2.** An example of a tree-unfolding.

### D.3 Characterization of unary automatic trees

**Theorem 10.** A tree $T$ is unary automatic if and only if there is a parameter set $\Gamma = (T_0, T_1, \ldots, T_m, \sigma, X)$ such that $T \cong UF(\Gamma)$.

**Lemma 3.** For $t \leq j < t + \ell$, the set $(j + i\ell)_{i \in \mathbb{N}}$ forms either an anti-chain or finitely many pairwise incomparable infinite chains in $T$.

**Proof.** If there is no $c$ such that $\Delta(q_j, (\cdot, 1)^{c\ell})$ is accepting, then $(j + i\ell)_{i \in \mathbb{N}}$ is an anti-chain. Otherwise, let $n_j$ be the least such $c$. In this case, $(j + i\ell)_{i \in \mathbb{N}}$ is partitioned into exactly $n_j$ pairwise incomparable chains in $T$. Indeed, $j + m\ell <_T j + (m + n_j)\ell$ for all $i$ and for $0 \leq m < n_j$, thus making $(j + (m + n_j)\ell)_{i \in \mathbb{N}}$ an infinite chain; furthermore elements in $\{j, j + \ell, \ldots, j + (n_j - 1)\ell\}$ are pairwise incomparable. \hfill \Box

By Lemma 3, let $A = \{j : (j + i\ell)_{i \in \mathbb{N}}$ is an anti-chain, $t \leq j < t + \ell\}$ and $C = \{t, \ldots, t + \ell - 1\} - A$. For each $j \in C$, let $n_j$ be the number of infinite chains in $(j + i\ell)_{i \in \mathbb{N}}$. For $0 \leq m < n_j$, we denote the infinite chain formed by $(j + (m + in_j)\ell)_{i \in \mathbb{N}}$ by $W_{j,m}$.

We consider the circumstances in which $W_{j,m}$ and $W_{k,m'}$ belong to the same infinite path in $T$ (interleave in the sense of Section 3). Fix $j, k \in C$. If there is no $m$ such that $\Delta(q_j, (\cdot, 1)^{k-j+m\ell})$ is accepting, then no $W_{j,s}$ and $W_{k,s'}$ belong to the same infinite path in $T$. Otherwise, let $m$ be the least number such that $\Delta(q_j, (\cdot, 1)^{k-j+m\ell}) \in F$.  

23
**Lemma 11.** With $m$ as above, then $n_j = n_k$ and $(j + i\ell)_{i \in \mathbb{N}}$ and $(k + i\ell)_{i \in \mathbb{N}}$ form exactly $n_j$ pairwise incomparable infinite chains.

*Proof.* By assumption, $\Delta(q_j,(\varnothing,1)^{k-j+m\ell})$ is accepting. Hence, $j_0 < T_0$ and $k + m\ell \in W_{k,m}$, for some $0 \leq m_1 < n_k$. Therefore, $m_1 \equiv m \mod n_k$, and $W_{j_0}$, $W_{k,m}$ interleave in $T$. Similarly, since $j + n_2 < T_0 < T_k$ and $k + n_2 + m\ell$, there is $0 \leq m_2 < n_k$ such that $m_2 = n_j + m \mod n_k$ and $W_{j_0}$, $W_{k,m}$ interleave in $T$. Therefore, $W_{k,m_1}$, $W_{k,m_2}$ interleave and by definition this implies $m_1 = m_2$. Hence, $n_j = cn_k$ for some $c > 0$. Since there is some interleaving between $j$ and $k$ sets, there is $r$ such that $\Delta(q_k,(\varnothing,1)^{j-k+r\ell}) \in F$. Repeating the above argument with the roles of $j$ and $k$ reversed, we see that $n_k = c'n_j$ for some $c' > 0$. Thus, $n_j = n_k$ and the union of $(j + i\ell)_{i \in \mathbb{N}}$ and $(k + i\ell)_{i \in \mathbb{N}}$ contains exactly $n_j$ pairwise disjoint infinite chains: for all $0 \leq i < n_j$, $W_{j,i}$ and $W_{k,m}$ interleave if and only if $m' = m + i \mod n_j$. □

Any infinite path through $T$ must be given by element(s) in $C$. Therefore, Lemma 11 implies that $T$ contains only finitely many infinite paths. We define a component of $T$ to be a connected subgraph of $T$ which contains exactly one infinite path and such that all the elements in the subgraph are greater than or equal to $t$. Fix $j \in C$ and $k \in A$. By Lemma 11, $(j + i\ell)_{i \in \mathbb{N}}$ belongs to exactly $n_j$ components, $B_0, \ldots, B_{n_j-1}$. If there is no $m$ such that $\Delta(q_j,(\varnothing,1)^{k-j+m\ell})$ is accepting, then no element in the anti-chain $(k + i\ell)_{i \in \mathbb{N}}$ belongs to any $B_r$. Otherwise, let $m$ be the least such that $\Delta(q_j,(\varnothing,1)^{k-j+m\ell}) \in F$. Each $j + i\ell$ has $k + (i + m)\ell$ as a descendent. Therefore $(k + i\ell)_{i \in \mathbb{N}}$ is partitioned into a finite set $\{k + i\ell : 0 < i < m\}$ and exactly $n_j$ infinite classes $\{(k + (m + s + in_j)\ell)_{i \in \mathbb{N}} : s = 0, \ldots, n_j-1\}$, each belonging to a unique $B_r$.

We have considered $j, k$ if either $j$ or $k$ (or both) are in $C$. Now, suppose $j, k \in A$ and neither $(j + i\ell)_{i \in \mathbb{N}}$ nor $(k + i\ell)_{i \in \mathbb{N}}$ intersects with any component of $T$. If there is $m$ such that $\Delta(q_j,(\varnothing,1)^{k-j+m\ell}) \in F$, then the union $(j + i\ell)_{i \in \mathbb{N}} \cup (k + i\ell)_{i \in \mathbb{N}}$ is a subset of infinitely many disjoint finite subtrees of $T$, each of which contains (at least) the nodes $j + i\ell$ and $k + (i + m)\ell$ for some $i$. We call these disjoint finite trees independent.

The above argument facilitates the definition of an equivalence relation $\sim$ on $\{t, \ldots, t + \ell - 1\}$ as $j \sim k$ if and only if

1. $j \in C$ (or $k \in C$) and $(j + i\ell)_{i \in \mathbb{N}}$ and $(k + i\ell)_{i \in \mathbb{N}}$ belong to the same $n_j$ (or $n_k$) components in $T$; or,
2. $j, k \in A$ and there is $h \in C$ such that $j \sim h$ and $k \sim h$;
3. $j, k \in A$ and $(j + i\ell)_{i \in \mathbb{N}}$ and $(k + i\ell)_{i \in \mathbb{N}}$ belong to the same collection of independent trees in $T$.

We use $[j]$ to denote the $\sim$-equivalence class of $j$.

*Proof (Theorem 10).* We now show that any unary automatic tree is isomorphic to the tree-unfolding $FU(T)$ of some parameter set $\Gamma = (T_0, T_1, \ldots, T_m, \sigma, X)$. For each $\sim$-equivalence class $[j]$, either $[j]$ represents infinitely many independent trees or $[j]$ represents finitely many components of $T$.  

24
In the first case, the independent trees represented by \([j]\) are pairwise isomorphic. Moreover, the set of ancestors of these independent trees in \(T\) is finite because they are not in a component of \(T\). In the second case, the components of \(T\) represented by \([j]\) are pairwise isomorphic. Each of these components can be described by “unfolding” a finite graph, \(T_e\), of size \(|[j]|\): each \(k \sim j\) contributes one vertex to \(T_e\) and the edges are specified by the relations between \((j + i\ell)\in\mathbb{N}\), \((k + i\ell)\in\mathbb{N}\) discussed above; the root of a later copy of \(T_e\) is a child of a fixed node in the immediately preceding copy. Observe that, as in the first case, the set of ancestors in \(T\) of the component is finite. It is immediate to translate this description to an appropriate parameter set for \(\Gamma\).

Conversely, we will show that if \(\Gamma = (T_0, T_1, \ldots, T_m, \sigma, X)\) is a parameter set, \(UF(\Gamma)\) is a unary automatic tree \(T\). Let \(l = |T_0|, \ell = \sum_{r=1}^m |T_r|\) and \(\alpha_r = \sum_{i=1}^r |T_i|\) for \(r = 1, \ldots, m\). We consider the isomorphic copy \((N_i \leq r) \cong UF(\Gamma)\) where \(T_0 \rightarrow \{0, \ldots, |T_0|\}\) and the \(j\)th copy of \(T_e\) maps to \(\{t + (j - 1)\ell + \alpha_r, \ldots, t + (j - 1)\ell + \alpha_r + 1 - 1\}\). The appropriate unary automaton will have parameters \(t, \ell\). Each \(q_j\) on the \((1, 1)\)-tail has \((\diamond, 1)\)- and \((1, \diamond)\)-tails of length \(t\), and a \((\diamond, 1)\)-loop of length \(\ell\). Each \(q_j\) on the \((1, 1)\)-loop has a \((\diamond, 1)\)-tail and \((\diamond, 1)\)-loop, each of length \(\ell\). All \((1, 1)\)-states are in \(F\). Let \(\varphi_0 : T_0 \rightarrow \{0, \ldots, t - 1\}\) and \(\varphi_r : T_r \rightarrow \{t + \alpha_r, \ldots, t + \alpha_r + 1 - 1\}\) be isomorphisms that preserve the tree order. We use \(\varphi_0\) to specify which \((1, \diamond)\)- and \((\diamond, 1)\)-tail states from the \((1, 1)\)-tail are accepting. Similarly, we use \(\varphi_1, \ldots, \varphi_m\) and \(\sigma, \gamma\) from the parameter set to specify those states in \((\diamond, 1)\)-loops off the \((1, 1)\)-tail and in \((\diamond, 1)\)-tails and loops off the \((1, 1)\)-loop that are accepting. Then \((N_i \leq r) \cong UF(\Gamma)\).

D.4 Canonical parameter sets for trees

**Lemma 4** Suppose \(T, T'\) are unary automatic trees with canonical parameter sets \(\Gamma, \Gamma'\). Then, \(T \cong T'\) if and only if \(\Gamma, \Gamma'\) have the same number \((m)\) of finite trees, \((T_0, \sigma) \cong (T'_0, \sigma')\), and for \(1 \leq i < m\), \((T_i, X(i)) \cong (T'_i, X'(i))\).

**Proof.** It is easy to see that if \(T\) and \(T'\) have term-by-term isomorphic canonical parameter sets they are isomorphic. Conversely, suppose \(T \cong T'\) and their canonical parameter sets are \((T_0, \ldots, T_m, \sigma, X)\) and \((T'_0, \ldots, T'_m, \sigma', X')\), respectively. Each infinite subtree of the form \(\{(y : \sigma(i) \leq y) : \leq r\}, 1 \leq i \leq m\), which contains infinitely many copies of \(T_0\), embeds into a subtree of \(T'\). By (2) in Definition 3, \(m_1 = m_2\). By the minimality condition on \(T_i, T'_i\) and by the ordering of the finite trees in each parameter set, the subtree of \(T\) containing infinitely many copies of \(T_i\) can embed into the subtree of \(T'\) containing infinitely many copies of \(T'_i\) for all \(1 \leq i \leq m_1\) and vice versa. Similarly for \(T_i, T'_i\) such that \(X(i) = X'(i') = 0\). By minimality of \(T_0, T'_0\), \(\forall 1 \leq i \leq m_1 (T_i, X(i)) \cong (T'_i, X'(i))\). Let \(t_1\) be the root of the first copy of \(T_1\) in \(T\) and \(t'_1\) be the root of the first copy of \(T'_1\) in \(T'\).

\[
(T_0, \sigma) \cong \{(y : y \in T_0 \land (\forall 1 \leq i \leq m) - t_i \leq T y) : \leq T\} \\
\cong \{(y : y \in T'_0 \land (\forall 1 \leq i \leq m) - t'_i \leq T' y) : \leq T'\} \cong (T'_0, \sigma') \quad \Box
\]

**Lemma 5** If \(\leq r\) is recognized by unary automaton with \(n\) states, there is an \(O(n^4)\) time algorithm that computes the canonical parameter set of \(T\).
Proof. We divide the construction into two pieces: first compute some parameter set \( \Gamma \) for \( T \), then compute the canonical parameter set from \( \Gamma \). Recall the proof that any unary automaton has an associated parameter set (from the proof of Theorem 10). Computing the sets \( A \) and \( C \) requires searching for the appropriate accepting states on the \((\circ,1)\)-tail and loop out of each state on the \((1,1)\)-loop. For each \( t \leq j < t + \ell \), let \( \ell_j \) be the length of \((\circ,1)\)-loop out of \( q_j \), and \( \hat{\ell}_j \) be sum of the lengths of \((\circ,1)\)-tail and \((1,\circ)\)-tail out of \( q_j \). Checking (as many as) \( \ell_j \) many states on the \((\circ,1)\)-loop and \( \hat{\ell}_j \) other states allows us to determine both \( n_j \) and the class \([j]\). In all, this takes time \( O \left( \sum_{j=t}^{t+\ell-1} (\ell_j + \hat{\ell}_j) \right) \).

Suppose \([j]\) represents finitely many components in \( T \). Each component is obtained by unfolding a finite tree \( T' \) of size \(|[j]|\) on some \( x \in T' \). The tree order \( \leq_{T'} \) can be computed by reading all the \((\circ,1)\)- and \((1,\circ)\)-states out of each \( q_k \) where \( k \sim j \). The node \( x \in T' \) is the \( \leq_{T'} \)-maximal node that is in some \((k+i\ell)_{i \in \mathbb{N}} \) with \( k \in C \). Again, the number of states out of \( q_j \) that need to be read is \( \ell_j \) and computing all \( T' \) takes time \( O(\sum_{j=t}^{t+\ell-1} \ell_j + \hat{\ell}_j) \). We need \( n_j \) isomorphic copies of \( T' \) in \( \Gamma \), a total of \( O(n_j|[j]|) \) nodes. Thus, to define all \( T' \) in the parameter set corresponding to these \( \sim \)-equivalence classes takes \( O(n^2) \).

On the other hand, \([j]\) might represent infinitely many pairwise isomorphic independent trees, each of which contains \(|[j]|\) nodes. To compute \( T' \) isomorphic to these independent trees, we read the \((\circ,1)\)- and \((1,\circ)\)-tails out of each \( q_k \) with \( k \sim j \). This takes time \( O(n) \). We call a node \( x \in \{0, \ldots, t-1\} \) a parent of \([j]\) if it is the immediate ancestor of infinitely many trees represented by \([j]\). If \([j]\) has \( c \) parents then there will be \( c \) copies of \( T' \) in the parameter set we are building, each of which has \( X(i) = \emptyset \) and with different values of \( \sigma(i) \).

Claim. There is an \( O(n^3) \) algorithm computing all parents of \( \sim \)-equivalence classes representing independent trees in \( T \).

Proof (of claim). Suppose \([j]\) represents infinitely many independent trees whose roots are from \((j+i\ell)_{i \in \mathbb{N}} \). For each \( 0 \leq k < t \), let \( t_k \) be the length of the \((\circ,1)\)-tail out of \( q_k \) and \( \ell_k \) be the length of the \((\circ,1)\)-loop out of \( q_k \). We describe an algorithm that compute the parents of \([j]\). The algorithm processes the subtree of \( T \) restricted to \{0, \ldots, t-1\}, beginning at the leaves and moving downwards (we process a node only once all of its children have been processed). For each node \( k \) we determine whether it is a parent of \([j]\).

- Case 1. If \( k \) is a leaf node, we search for \( t_k \leq i < t_k + \ell_k \) such that \( \Delta(q_k,(\circ,1)^{i+t+j-k}) \in F \). We can find such an \( i \) if and only if there are infinitely many independent trees associated to \([j]\) descending from \( k \) in \( T \).
- Case 2. If \( k \) is an internal node but has no children which are parents of \([j]\), process it as though it were a leaf node. Otherwise, let \( k_1, \ldots, k_r \) be children of \( k \) which are parents of \([j]\). Let \( U_i, V_i, D_i \) be subsets of \( \{t_{k_1}, \ldots, t_{k_r} + \ell_{k_r}\} \) defined as

\[
U_i = \{ x : \Delta(q_k,(\circ,1)^{x+t+j-k}) \in F \}, \quad V_i = \{ x : \Delta(q_k,(\circ,1)^{x+t+j-k_i}) \in F \},
\]

and \( D_i = U_i - V_i \). Let \( \ell' = \max \{ \ell_{k_1}, \ldots, \ell_{k_r} \} \) and let \( D'_i = \{ x + i\ell_k : x \in D_i \land x + i\ell_k < \ell' \ell_k \} \). Then \( k \) is a parent of \([j]\) if and only if \( D'_1 \cap \cdots \cap D'_r \neq \emptyset \).
Correctness. In Case 1, if there is no \(i' \geq t_k\) such that \(\Delta(q_k, (\circ, 1)^{\ell \ell_j - k}) \in F\), there are only finitely many independent trees represented by \([j]\) descending from \(k\). Moreover, if such an \(i'\) exists, it must be on the \((\circ, 1)\)-loop off \(q_k\) and so we can stop looking for it after we have checked all \(t_k\) states. For Case 2, note that \(x \in D'_1 \cap \cdots \cap D'_i\) if and only if \(k\) is the immediate ancestor for all nodes in \([j + (x + i\ell t_k)\ell]\) \(\ell \in \mathbb{N}\), if and only if \(k\) is the immediate ancestor for some node in \([j + (s + im)\ell]\) \(\ell \in \mathbb{N}\). Therefore if \(D'_1 \cap \cdots \cap D'_i \neq \emptyset\), then \(k\) is a parent of \([j]\). Suppose \(D'_1 \cap \cdots \cap D'_i = \emptyset\) and \(k\) is a parent of \([j]\). Then \(k\) is the immediate ancestor for \([j + (s + im)\ell]\) \(\ell \in \mathbb{N}\) for some \(m > \ell t_k\) and \(s < m\). If \(s < \ell t_k\), then \(s \in D'_1 \cap \cdots \cap D'_i\). Therefore \(s' \in D'_1 \cap \cdots \cap D'_i\). Contradiction. Hence the algorithm is correct.

Complexity. Checking if a leaf node \(k\) is a parent for \([j]\) takes time \(O(\ell t_k)\). When \(k\) is an internal node, computing \(U_i\) and \(V_i\) takes time \(O(\ell t_k)\). The size of each \(U_i\) and \(V_i\) is bounded by \(\ell t_k\), therefore computing \(D_i\) takes \(O(\ell t_k)\). Computing each \(D_i'\) takes time \(O(\ell t_k)\). We need to do the above operations at most \(t\) times (at most once for each node in \([0, \ldots, t-1]\)). Therefore, the algorithm takes time \(O(t \ell^2)\), where \(\ell\) is the maximal \((\circ, 1)\)-loop length out of all \(q_k\), \(k \in [0, \ldots, t-1]\). We iterate the intersection operation \(r\) times to compute the intersection of all the \(D_i'\)'s; therefore, we perform a total of at most \(t r\) intersection operations, each taking time \(O(\ell^2)\). Since \(\ell < n\), the algorithm takes time \(O(n^3)\). We can run the above algorithm simultaneously for all equivalence classes \([j]\) representing independent trees without increasing the time complexity. \(\square\)

With the above claim in hand, we can resume our construction of the parameter set \(\Gamma\). The finite tree \(T_0\) in \(\Gamma\) contains all nodes in \([0, \ldots, t-1]\) and finitely many independent trees. Deciding which independent trees to put into \(T_0\) uses the claim and therefore takes \(O(n^3)\). Computing the tree order \(\preceq_{\Gamma_0}\) on \([0, \ldots, t-1]\) requires reading the \((\circ, 1)\)- and \((1, \circ)\)-tail out of each \(q_k\), \(0 \leq k < t\) at most once. This steps again takes time \(O(n)\). Thus, in time \(O(n^4)\), we have computed \(\Gamma = (T_0, T_1, \ldots, T_m, \sigma, X)\) such that \(T \cong \text{UF}(\Gamma)\). Since nodes in \(T_0\) can be parents to more than one anti-chain, \(m \leq \ell t + \sum_{j \not\in \circ X} n_j \leq \ell t + n\).

We now use \(\Gamma\) to obtain a canonical parameter set for \(T\). For each \(1 \leq i < m\) with \(X(i) \neq \emptyset\), look for \(y_1, y_2 \in T_i\) such that \(y_1 < T y_2 < T X(i)\), and the subtree of \(T_i\) with domain \(\{ z : y_1 \leq T z \land y_2 \not\in T z \}\) is isomorphic to the subree with domain \(\{ z : y_2 \leq T z \land X(i) \not\in T z \}\). If such \(y_1, y_2\) exist, remove all \(z \geq T y_1, y_2\) from \(T_i\). For each \(1 \leq i < j \leq m\) such that \(X(i) = X(j) = \emptyset\), if \(T_i \cong T_j\) and \(\sigma(i) = \sigma(j)\) then remove \(T_j\). Thus, each \(T_i\) satisfies the minimality condition for the canonical parameter set. Since the isomorphism problem for finite trees is decidable in linear time \([6]\), this step can be done in time \(O(\sum_{i=1}^m |T_i|^2)\).

For each \(1 \leq i \leq m\) with \(X(i) \neq \emptyset\), let \(t_i\) be the root of \(T_i \times \{0\}\). Look for \(x \in T_0\) such that \(x = T \sigma(i)\), and the subtree of \(T_0\) with domain \(\{ y : x \leq T y \land t_i \not\in T y \}\) is isomorphic to \(T_i\). If such an \(x\) exists, remove all \(y \geq T_0, x\) from \(T_0\). Now \(T_0\) satisfies the minimality condition. Again this step can be done in time \(O(\sum_{i=1}^m |T_i|^2)\).
For each \(1 \leq i \leq m\), search for the \(<_{T_0}\)-least \(x\) such that the subtree of \(T_0\) with domain \(\{z \in T_0 : x \leq_{T_0} z\}\) is isomorphic to a subtree of \(T_i\) with domain \(\{z \in T_i : y \leq_{T_i} z\}\) for some \(y <_{T_i} X(i)\). If such an \(x\) exists, remove all \(y \geq_{T_0} x\) from \(T_0\). This step ensures that \(T_0\) has the fewest possible nodes with respect to \(T_i\); it can be done in time \(O(\sum_{i=1}^{m} |T_i|^2)\).

Finally, we permute \(T_1, \ldots, T_m\) so that \(T_1 \preceq \ldots \preceq T_m\). We assume that finite trees can be efficiently encoded as natural numbers and hence applying a sorting algorithm on \(m\) of them takes \(O(m \log m)\). Whenever we find \(T_i \cong T_j(i \neq j)\) with \(\sigma(i) = \sigma(j)\) and \(X(i) = X(j) = \emptyset\), keep \(T_i\) and delete \(T_j\). Converting \(\Gamma\) to a canonical parameter set takes \(O(n^3)\) and thus we have built such a canonical parameter set in \(O(n^3)\) time. □

### E Proofs for Section 6: Graphs of finite degree

**Theorem 14.** The isomorphism problem for unary automatic graphs of finite degree is decidable in elementary time.

We prove Theorem 14 for connected infinite graphs by giving a MSO definition (in the language of graphs) of the isomorphism type of a connected graph \(G = (\mathbb{N}; R)\) and then using the decidability of the MSO theory of unary automatic structures. We first define auxiliary MSO-formulas. For a fixed set \(S\) and \(k \in \mathbb{N}\), let \(\text{Partition}^S_k(P_1, \ldots, P_k)\) be the formula

\[
\left( \bigwedge_{i=1}^{k} \exists^\omega x(x \in P_i) \right) \land \left( \bigwedge_{1 \leq i \neq j \leq k} P_i \cap P_j = \emptyset \right) \land \left( S = \bigcup_{i=1}^{k} P_i \right)
\]

For a finite graph \(\mathcal{F} = (\{v_1, \ldots, v_k\}; E_\mathcal{F})\), \(\text{Type}^\mathcal{F}(X, Y_1, \ldots, Y_k)\) is

\[
\bigwedge_{i=1}^{k} (\exists^\omega x_i(x_i \in X \cap Y_i) \land \bigwedge_{i,j=1}^{k} (x_i \in X \cap Y_i, x_j \in X \cap Y_j \rightarrow E_\mathcal{F}(x_i, y_j) \leftrightarrow E_\mathcal{F}(v_i, v_j))
\]

For a finite graph \(\mathcal{F}\) of size \(k\) and a mapping \(\sigma : V_\mathcal{F} \rightarrow P(V_\mathcal{F})\), let \(\mathcal{F}_\times^k\) be the finite subgraph \(\mathcal{F}_\sigma \upharpoonright F_0 \cup F_1 \cup F_2\). Label \(V_{\mathcal{F}_\times^k}\) by \(v_1, \ldots, v_{3k}\) where \(\{v_{i+1}, \ldots, v_{i+1+k}\}\) belong to the \(i\)-th copy of \(\mathcal{F}\) and for each \(1 \leq i \leq k\), the vertices \(v_i, v_{k+i}, v_{2k+i}\) all correspond to the same vertex in \(\mathcal{F}\). Define the formula \(\text{Succ}^\mathcal{F}_\sigma(X, Y_1, \ldots, Z_{3k})\) to be

\[
\text{Type}^\mathcal{F}(X, Z_1, \ldots, Z_{3k}) \land \text{Type}^\mathcal{F}(Y, Z_1, \ldots, Z_{3k}) \land (X \cap Y = \emptyset) \\
\land \bigwedge_{(i,j) : x_j \in \sigma(v_i)} (\forall x \in X \cap Z_{2k+i})(\forall y \in Z_j)(E_\mathcal{F}(x, y)) \\
\land \bigwedge_{(i,j) : x_j \notin \sigma(v_i)} (\forall x \in X \cap Z_{2k+i})(\forall y \in Z_j)(\neg E_\mathcal{F}(x, y)) \\
\land \bigwedge_{(i,j) \notin \{2k+1, \ldots, 3k\} \times \{1, \ldots, k\}} (\forall x \in X \cap Z_i)(\forall y \in Z_j)(\neg E_\mathcal{F}(x, y)).
\]
We are now ready to prove the theorem.

Proof (Theorem 14). Suppose \( G \) is a connected unary automatic graph of finite degree. By Theorem 15, we can find \( D, F, \sigma \) such that \( G \) is isomorphic to \( D \) followed by \( F_{\sigma^k} \). Since \( D \) and the edge relation between \( D \) and \( F_0 \) are finite, they can be used as parameters in a MSO sentence. Hence, for simplicity we assume that \( D \) is empty. Let \( V_F = \{v_0, \ldots, v_k\} \) and recall the definition of \( F_{\times 3} \) above. We define \( \varphi_G \) as \((\exists P_1 \cdots \exists P_{3k})(\psi_G(P_1, \ldots, P_{3k}))\), where \( \psi_G(Z) \) is the conjunction of the following formulas:

1.  Partition\(_{3k}(Z)\)
2.  \((\forall x)(\exists X)[x \in X \land \text{Type}_{\times 3}(X, Z)]\)
3.  \((\forall X)[\text{Type}_{\times 3}(X, Z) \rightarrow (\exists^\exists Y)\text{Succ}_F(X, Y, Z)]\)
4.  \((\exists X)[\text{Type}_{\times 3}(X, Z) \land (\forall Y)[(\text{Type}_{\times 3}(Y, Z) \land X \cap Y = \emptyset) \rightarrow \neg \text{Succ}_F(X, Y, Z) \land (\exists^\exists W)\text{Succ}_F(W, Y, Z)]]\)

Claim. If \( H \) is an infinite connected graph, \( H \models \varphi_G \) if and only if \( H \cong G \).

Proof (of claim). If \( H \cong G \) then clearly \( H \models \varphi_G \). On the other hand, suppose \( H \models \varphi_G \). Then \( H \) can be partitioned into \( 3k \) sets \( P_1, \ldots, P_{3k} \). Take a subgraph \( M \) of \( 3k \) vertices in \( H \). We say that \( M \) is a \( F_{\times 3} \)-type if \( M \) intersects each \( P_i \) at exactly one vertex, and if we let \( v_i = M \cap P_i \), then the three sets of vertices \( \{v_1, \ldots, v_k\}, \{v_{k+1}, \ldots, v_{2k}\}, \{v_{2k+1}, \ldots, v_{3k}\} \) respectively form three copies of \( F \), with \( v_i, v_{k+i}, v_{2k+i} \) corresponding to the same vertex in \( F \). Also, the edge relation between these three copies of \( F \) respects the mapping \( \sigma \).

Since \( H \models \varphi_G \), each vertex \( v \) in \( H \) belongs to a unique subgraph that is a \( F_{\times 3} \)-type; and, for each \( F_{\times 3} \)-type \( M \), there is a unique \( F_{\times 3} \)-type \( N \) that is a successor of \( M \), i.e., all edges between \( M \) and \( N \) are from the last copy of \( F \) in \( M \) to the first copy of \( F \) in \( N \) such that they respect the mapping \( \sigma \). Lastly there exists a unique \( F_{\times 3} \)-type \( M_0 \) which is not the successor of any other \( F_{\times 3} \)-types and any other \( F_{\times 3} \)-type is the successor of a unique \( F_{\times 3} \)-type. Note that the successor relation between the \( F_{\times 3} \)-types resembles the unfolding operation on finite graphs.

Therefore to set up an isomorphism from \( H \) to \( G \), we only need to map \( M_0 \) isomorphically to the first 3 copies of \( F \) in \( G \), and then map the other vertices according to the successor relation and mapping \( \sigma \).

By Theorem 2, satisifiability of any MSO sentence is decidable for unary automatic graphs. Therefore the isomorphism problem for unary automatic graphs of finite degree is decidable. Note that the definition of \( \varphi_G \) contains only finitely many alternations of quantifiers (regardless of the size of the automaton presenting it), therefore the decision procedure is elementary in terms of the size of the input automaton.