Analyzing Complexity in Classes of Automatic Structures

Jiamou Liu[1], Mia Minnes[2]

[1] Department of Computer Science, University of Auckland, New Zealand
jliu036@aucklanduni.ac.nz
[2] Department of Mathematics, MIT, USA
minnes@math.mit.edu

Abstract. This paper addresses the complexity of several families of queries in classes of unary automatic structures. The queries include membership and isomorphism. In addition, we study the state complexity of describing these classes. In particular, 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 φ, one can effectively construct an automaton that recognizes all elements of $\mathcal{A}$ that satisfy φ. 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. In a similar way to the use of finite model theory in reasoning about databases [**?**], automatic structures have been applied to areas where one is interested in the algorithmic properties of infinite structures such as databases and computer-aided verifications [**?**,16]. However, this approach has limitations. In particular, since the configuration space of a Turing machine can be coded by a finite automaton [11], reachability is undecidable for automatic structures in general. On the other hand, *unary automatic structures*, those recognized by automata over an 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 [14]. Moreover, if we consider the intermediate class of structures

whose domain elements are encoded as finite strings over $1^\star 2^\star$, insufficient decidability strength results: since the infinite grid can be coded automatically over $1^\star 2^\star$ and counter machines can be coded into the grid, reachability is not decidable in this class of structures. Thus, the class of unary automatic structures is a sensible context where reachability is decidable.

Much is known about the complexity of automatic structures. Various *algebraic characterizations* of special classes of automatic structures have demonstrated their corresponding simplicity. For example, explicit descriptions of automatic Boolean algebras [11] and finitely generated automatic groups [13] are known. Occasionally, these results have translated to information about the computational content of these classes: the isomorphism problem of automatic Boolean algebras was proved to be decidable. However, when we consider the class of automatic structures as a whole, we find significant underlying complexity. This complexity can be measured from a *computability theoretic* point of view: the isomorphism problem and embedding problem for automatic structures is Σ^1_1-complete [11, 17]. The *model-theoretic complexity* of automatic structures also gives evidence to their inherent richness: [8] shows that the Scott ranks of automatic structures can be as high as possible, fully covering the interval $[1, \omega_1^{CK} + 1]$ of ordinals (where ω_1^{CK} is the first non-computable ordinal). Finally, there is a body of work devoted to the *resource-bounded complexity* of automatic structures. A thorough study of the time and space complexity of model-checking and query-evaluation on automatic structures for fragments of first-order logic is given in [2]. In particular, when an automaton is fixed, the complexity of model-checking for quantifier-free formulae is LOGSPACE-complete and for existential formulae it is NPTIME-complete. On the other hand, there are automatic structures whose first-order theories are nonelementary [**?**].

In this paper we restrict our attention to classes of unary automatic structures. We prove that in various senses, the complexity of these classes is quite low. We consider two notions of complexity.

1. *Time complexity.* For a fixed class $\mathcal{K}$ of structures, we are interested in the time complexity of solving the following natural decision problems.

   Membership Problem. Given a unary automatic presentation of $\mathcal{A}$, decide if $\mathcal{A} \in \mathcal{K}$.

   Isomorphism Problem. Given unary automatic presentations of $\mathcal{A}$ and $\mathcal{B}$ from $\mathcal{K}$, decide if $\mathcal{A} \cong \mathcal{B}$.

   For all the classes of unary automatic structures we consider, we give low polynomial time solutions to the membership problem. For unary automatic equivalence relations, linear orders, and trees we provide algorithms that solve the isomorphism problem in low polynomial time. We show that the isomorphism problem for graphs with finite degree is decidable with elementary time bound.
2. *State complexity.* The notion of state complexity measures the descriptive complexity of regular languages, context-free grammars, and other classes of languages with finite representations. The state complexity (with respect to automata) of a regular language L is defined to be the size of the smallest

automaton with language L. Research into state complexity with respect to automata has been well-established since the 1950s [3, 18, 19]. A key motivation for it is in designing automata for real-time computation where the runtime of algorithms operating on the automata depends on the number of states.

In this paper, we generalize state complexity to structures (rather than sets). The *state complexity* of a automatic structure $(\mathbb{N}, R)$ is the number of states in an optimal automaton for R, a (unary) automaton recognizing R with the fewest possible states. 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, and hopefully fruitful, area. "

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 introduces the notion of unary state complexity. Sections 3, 4, 5 and 6 discuss linear orders, equivalence relations, trees and graphs of finite degree (respectively).

2 Preliminaries

We assume the basic terminologies and notations in automata theory and regular languages (see, for example, [6]). For a fixed alphabet Σ, a *finite automaton* is a tuple $\mathcal{A} = (S, \Delta, I, F)$ where S, Δ, I, F are respectively the state space, transition function, initial state and accepting states. In particular, if $\mathcal{A}$ 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 $\Sigma_\diamond = \Sigma \cup \{\diamond\}$ where $\diamond$ is a symbol not in Σ. Given an n-tuple of words $w_1, w_2, \ldots, w_n \in \Sigma^\star$, the *convolution* of $(w_1, \ldots, w_n)$ is a word $\otimes(w_1, \ldots, w_n)$ over the alphabet $(\Sigma_\diamond)^n$ with length $\max\{|w_1|, \ldots, |w_n|\}$. The k^{th} symbol of $\otimes(w_1, \ldots, w_n)$ is $(\sigma_1, \ldots, \sigma_n)$ where σ_i is the k^{th} symbol of w_i if $k \leq |w_i|$, and is $\diamond$ otherwise. An n-ary relation R is FA recognizable if the set of convolutions of all pairs $(w_1, \ldots, w_n) \in R$ is a regular subset of $(\Sigma_\diamond^n)^\star$.

A relational *structure* $\mathcal{S}$ consists of a countable domain D and atomic relations on D. A structure is called *automatic* over Σ if its domain is a regular subset of $\Sigma^\star$ and each of its atomic relations is FA recognizable. A structure is called *unary automatic* if it is automatic over the alphabet $\{1\}$. A (unary) automatic structure $\mathcal{A}$ isomorphic to a structure $\mathcal{B}$ is called a *(unary) automatic presentation* of $\mathcal{B}$ and $\mathcal{B}$ is *(unary) automatically presentable*. We sometimes abuse the terminology to refer to $\mathcal{B}$ as simply (unary) automatic. The structures $(\mathbb{N}; S)$ and $(\mathbb{N}; \leq)$ are both automatic structures. On the other hand, $(\mathbb{Q}; \leq)$ and $(\mathbb{N}; +)$ have isomorphic copies which are automatic over $\{0, 1\}$ but have no unary automatic

presentation. The structure $(\mathbb{N}; \times)$ has no automatic isomorphic presentation. For proofs of these facts, see the survey papers [9, 15].

Consider $FO + \exists^{\infty} + \exists^{n,m}$, the first-order logic extended by $\exists^{\infty}$ (there exist infinitely many) and $\exists^{n,m}$ (there exist n many mod m, where n and m are natural numbers) quantifiers. The following theorem from [2, 4, 10, 14] connects this extended logic with automata.

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

In this paper we study automatic structures in the form $(D; R)$ where R is a binary relation. Suppose $\mathcal{A}_D$ (m states) and $\mathcal{A}_R$ (n states) are deterministic finite automata recognizing D and R, respectively. Some first-order definable properties of binary relations are listed in Table 1. 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 $\mathcal{A}_1$ recognizing the relation $\{(y, x) : (x, y) \in R\}$ and check if $R = L(\mathcal{A}_1)$. 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 $R \cup L(\mathcal{A}_1) = D^2$. 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. Note that if $(D; R)$ is automatic over Σ and $D = \Sigma^{\star}$, then $m = 1$.

Table 1. Deciding properties of binary relations in automatic structures.

Property	First-order definition	Time complexity
Reflexivity	$\forall x\,(R(x, x))$	$O(mn)$
Symmetry	$\forall x, y\,(R(x, y) \implies R(y, x))$	$O(n^2)$
Antisymmetry	$\forall x, y\,(R(x, y) \wedge R(y, x) \implies x = y)$	$O(n^2)$
Totality	$\forall x, y\,(R(x, y) \vee R(y, x))$	$O(m^2 n^2)$
Transitivity	$\forall x, y, z\,(R(x, y) \wedge R(y, z) \implies R(x, z))$	$O(n^3)$

We now turn our attention to unary automatic structures which is the focus of this paper. Recall that a structure is *unary automatic* if it is automatic over the alphabet $\{1\}$. We use x to denote the string 1^x and $\mathbb{N}$ for the set of all such strings $\{1\}^*$. By [1], a structure is unary automatic if and only if it is first-order interpretable in the structure $\mathcal{U} = (\mathbb{N}; 0, <, s, \{ \mod_m\}_{m>1})$, where s is the successor relation and $x \mod_m y$ if and only if $x \equiv y \mod m$. Using a monadic second order interpretation of $\mathcal{U}$ in $(\mathbb{N}; s)$, one easily get the following result.

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

The following lemma from [1] characterizes the regular subsets of $\{1\}^{\star}$.

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

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 $\Delta(\iota, 1) = q_1$ and for all $1 \leq i < t$, $\Delta(q_i, 1) = q_{i+1}$. There is another sequence of states $S_2 = \{q_{t+1}, \ldots, q_{t+l}\}$ such that for all $t \leq j < l$, $\Delta(q_j, 1) = q_{j+1}$, and $\Delta(q_l, 1) = q_{t+1}$. 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 $t + il + k$, $i \in \omega$, 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, ℓ, we can define the corresponding unary automaton. □

The general shape of a 2-tape unary automata is given in Figure 1. We fix some terminology. States reachable from the initial state by reading inputs of type $(1, 1)$ are called $(1, 1)$-*states*. The set of $(1, 1)$-states is a disjoint union of a *tail* and a *loop*. We label the $(1, 1)$-states as $q_0, \ldots, q_t, \ldots, q_\ell$ where $q_0, \ldots, q_{t-1}$ form the $(1, 1)$-tail and there is a transition from q_ℓ to q_t to close the $(1, 1)$-loop. States reachable from a $(1, 1)$-state by reading inputs of type $(1, \diamond)$ are called $(1, \diamond)$-*states*. The set of $(1, \diamond)$-states reachable from any given q_i consist of a tail and a loop, called the $(1, \diamond)$-*tail* and *loop* from q_i, respectively. The $(\diamond, 1)$-*tails* and *loops* are defined similarly. The *tail length* of the automaton is t, the length of its $(1, 1)$-tail; the *loop length* is ℓ, the length of its $(1, 1)$-loop.

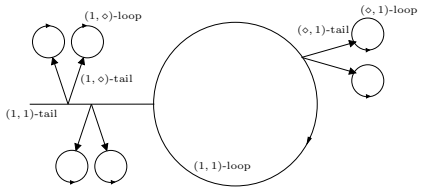

Fig. 1. General shape of a deterministic 2-tape unary automaton

Khoussainov and Rubin [12] and Blumensath [1] gave a characterization of all unary automatic binary relations on $\mathbb{N}$. Let $\mathcal{B} = (B, E_B)$ and $\mathcal{D} = (D, E_D)$ be finite graphs. Let R_1, R_2 be subsets of $D \times B$, and R_3, R_4 be subsets of $B \times B$. Consider the graph $\mathcal{D}$ followed by countably infinitely many copies of $\mathcal{B}$, ordered as $\mathcal{B}^0, \mathcal{B}^1, \mathcal{B}^2, \ldots$. We define the infinite graph $unwind(\mathcal{B}, \mathcal{D}, \bar{R})$ as follows. Its vertex set is $D \cup B^0 \cup B^1 \cup B^2 \cup \ldots$ and its edge set contains $E_D \cup E^0 \cup E^1 \cup \ldots$ as well as the following edges, for all $a, b \in B$, $d \in D$, and $i, j \in \omega$:

- (d, b^0) when $(d, b) \in R_1$, and (d, b^{i+1}) when $(d, b) \in R_2$,
- (a^i, b^{i+1}) when $(a, b) \in R_3$, and (a^i, b^{i+2+j}) when $(a, b) \in R_4$.

Theorem 3 ([1, 12]). *A graph is unary automatic if and only if it is isomorphic to unwind$(\mathcal{B}, \mathcal{D}, \bar{R})$ for some parameters $\mathcal{B}, \mathcal{D}, \bar{R}$.*

The *state complexity* of a regular language L is the number of states of the minimal deterministic finite automaton that recognize L [18]. We extend this definition to the state complexity of automatic structures.

Definition 1. *The* state complexity *of an (unary) automatic structure $\mathcal{A}$ is the size of the smallest (unary) automaton $\mathcal{M}$ such that $\mathcal{M}$ recognizes a structure $\mathcal{B} \cong \mathcal{A}$. We call $\mathcal{M}$ the* optimal automaton *for $\mathcal{A}$.*

Let $\mathcal{K}$ be a class of automatic structures such that each member $\mathcal{A}$ of $\mathcal{K}$ have a finite representation $R_{\mathcal{A}}$ which is *independent* on the automatic presentations, i.e., for $\mathcal{B} \in \mathcal{K}$, $\mathcal{A} \cong \mathcal{B}$ if and only if $R_{\mathcal{A}} = R_{\mathcal{B}}$. Let $|R_{\mathcal{A}}|$ denote the size of $R_{\mathcal{A}}$. The *state complexity* of the class $\mathcal{K}$ is a function f such that $f(n)$ is the largest state complexity of all $\mathcal{A} \in \mathcal{K}$ with $|R_{\mathcal{A}}| \leq n$.

In this paper we look at the state complexity of three classes of unary automatic structures: unary automatic equivalence relations, unary automatic linear orders and unary automatic trees. For members of each class, we define a finite representation that is independent on the automatic presentations, and we show that the state complexity for each class is polynomial with respect to the defined representations.

We make the following assumptions for the rest of the paper.

1. All structures are infinite with domain $\mathbb{N}$, the set of natural numbers. (See Lemma 2 for a justification of this assumption)
2. All automata are deterministic.
3. Algorithm on unary automatic structures $(\mathbb{N}; R)$ have as input a synchronous 2-tape automaton recognizing R. The size of the input is the number of states in the input automaton.
4. We assume the sets of (1,1)-, $(\diamond, 1)$-, and $(1, \diamond)$- states are pairwise disjoint. Therefore no (1,1)-state is also a $(\diamond, 1)$-state, etc.

Lemma 2. *Let $(D; R)$, $D \subset \mathbb{N}$, 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 ℓ 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, $\mathcal{A}_R$ 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 ℓ; 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 $\mathbb{N}$ will be given by $i\ell + k_j \mapsto ir + j$. Therefore, define $\mathcal{A}_{R'}$ to have a $(1, 1)$-tail of length $c'r$, a $(1, 1)$-loop of length cr, and copy the information from the state $i\ell + j$ in $\mathcal{A}_R$ to state $ir + j$ in $\mathcal{A}_{R'}$ (modifying the lengths of $(\diamond, 1)$- and $(1, \diamond)$-tails and loops appropriately). Then, $(\mathbb{N}; L(\mathcal{A}_{R'})) \cong (D; R)$ and since $r \leq \ell$, $\mathcal{A}_{R'}$ has no more states than $\mathcal{A}_R$. □

3 Linear Orders

This section studies unary automatic linear orders. A *linear order* is $\mathcal{L} = (\mathbb{N}; \leq_L)$ where $\leq_L$ is total, reflexive, anti-symmetric, and transitive. The following is immediate by Table 1.

Proposition 1. *The membership problem for automatic linear orders is decidable in time $O(n^3)$.*

The following theorem was proved by Blumensath [1] and Khoussainov and Rubin [12] and characterizes unary automatic linear orders. We use ω to denote the order type of the positive integers, ω^* to denote the order type of the negative integers, ζ to denote the order type of the integers ($\omega^\star$ followed by ω), and $\mathbf{n}$ to denote the finite linear order of length n.

Theorem 4 ([1, 12]). *A linear order is unary automatic if and only if it is isomorphic to a finite sum of linear orders of type ω, ω^* or $\mathbf{n}$.*

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

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

$$x <_L y \wedge \neg(\exists^\infty z)[x <_L z \wedge z <_L y].$$

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

$$(\exists^\infty y)[x <_L y \wedge \text{FinDis}(x, y)] \wedge (\forall z <_L x)[\neg\text{FinDis}(z, x)]$$

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

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

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

$$(\exists^\infty y)[x <_L y \wedge \text{FinDis}(x, y)] \wedge [(\exists^\infty z) z <_L x \wedge \text{FinDis}(z, x)]$$

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

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

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

$$(\exists x_0, \ldots, x_{k-1})[\bigwedge_{i=0}^{k-2}(x_i <_L x_{i+1}) \wedge \bigwedge_{i=0}^{k-1} \mathrm{In}^{\alpha_i}(x_i) \wedge (\forall y) \bigvee_{i=0}^{k-1} (\mathrm{FinDis}(x_i, y) \vee \mathrm{FinDis}(y, x_i))]$$

□

The sentence $\varphi_{\mathcal{L}}$ contains three alternations of quantifiers. To decide whether automatic linear orders $\mathcal{L}, \mathcal{L}'$ are isomorphic, we check if $\mathcal{L}' \vDash \varphi_{\mathcal{L}}$ (by Theorem 1. An exponential runtime blow-up occurs for each alternation of quantifiers in $\varphi_{\mathcal{L}}$ [10]. We now significantly improve this bound by providing a quadratic time algorithm for the isomorphism problem 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 $\mathcal{A}$ it has parameters t, ℓ which are the lengths of its $(1,1)$-tail and -loop. From now on, we will decompose the unary automatic linear orders as a sequence of copies of $\omega, \omega^*, \mathbf{n}$ (with no ζ).

Lemma 3. *Suppose $\mathcal{A}$ 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 ω in $\mathcal{L}$ or a decreasing chain in a copy of ω^* in $\mathcal{L}$.*

Proof. If $\Delta(q_j, (\diamond, 1)^\ell) \in F$ 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 $\Delta(q_j, (1, \diamond)^\ell) \in F$ 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 $(\diamond, 1)$- and $(1, \diamond)$- tails off q_j; let ℓ_1, ℓ_2 be the lengths of the $(\diamond, 1)$- and $(1, \diamond)$- 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, $\Delta(q_j, (\diamond, 1)^{c\ell}) \in F$ for all c. Hence, $\Delta(q_j, (\diamond, 1)^{t_1}) \in F$. Similarly, $\Delta(q_j, (\diamond, 1)^{t_2}) \notin F$. Therefore, each $x > j + (i+1)\ell + \max\{t_1, t_2\}$ 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 $k = \max\{t_1, t_2\} + \ell$. 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 $\Delta(q_j, (\diamond, 1)^{k+s}) \in F$ and hence for any c, $j + i\ell + c\ell <_L r + c\ell$. The second inequality implies that $\Delta(q_j, (1, \diamond)^{k+s-\ell}) \in F$ and is in the $(1, \diamond)$-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 $<_{\mathcal{L}}$-between $j + i\ell$ and $j + (i+1)\ell$ must be smaller than r; there are only finitely many such elements. □

Proof (Proof of Theorem 5). Suppose $\mathcal{L} = (\mathbb{N}; <_L)$ is a unary automatic linear order represented by a unary automaton $\mathcal{A}$ with parameters t, ℓ. We will extract its canonical word $\alpha_{\mathcal{L}} \in \{\omega, \omega^*, \mathbf{n}\}^\star$. To do so, we need only determine the relative ordering of the at most ℓ many copies of ω and ω^* given by Lemma 3 and the elements $0, \ldots, t-1$.

For $t \leq j < t + \ell$, we can use Lemma 3 to decide in linear time whether the sequence $(j + i\ell)_{i\in\mathbb{N}}$ is in a copy of ω or ω^*. 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 ω or ω^* 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 $(\diamond, 1)$-tail off q_j and c_2 larger than the length of the $(\diamond, 1)$-tail off q_k satisfying $\Delta(q_j, (\diamond, 1)^{c_1})$ and $\Delta(q_k, (\diamond, 1)^{c_2})$ 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 $(1, \diamond)$ states for the $(\diamond, 1)$ ones.

To check if the above conditions hold, we can read the states $\Delta(q_j, (\diamond, 1)^{k-j+d_1\ell})$, $\Delta(q_k, (\diamond, 1)^{j-k+d_2\ell})$ 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 $(\diamond, 1)$ loops (thus, fewer than n) and deciding interleaving takes $O(n^2)$ time.

Notice that while considering the cases above, we also determine 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 $(\diamond, 1)$-tail off q_j such that $\Delta(q_j, (\diamond, 1)^c) \in F$ 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 $j, k \in V_\ell$ $(0 \leq \ell \leq s)$ then $(j+i\ell)_{i\in\mathbb{N}}$ and $(k+i\ell)_{i\in\mathbb{N}}$ interleave. We simultaneously define, for $t \leq j < t + \ell$, $Left(j) = \{k : (k + i\ell)_{i\in\mathbb{N}} <_L (j + i\ell)_{i\in\mathbb{N}}\}$; since we have to consider all j, k jointly, this takes $O(n^2)$ time.

It remains to define $Left(j)$ for $0 \leq j < t$. If there is $t \leq k < t + \ell$ and d_1, d_2 such that $\Delta(q_j, (\diamond, 1)^{d_1}) \in F$ and $\Delta(q_j, (1, \diamond)^{d_2}) \in F$ and $d_1 \equiv d_2 \equiv (k - j) \mod \ell$ then j is $<_L$-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 $Left(j)$ undefined. Otherwise, the elements $<_L$-below j are obtained by examining the $(1, \diamond)$-tail and -loop of q_j and the $(\diamond, 1)$-tails and loops of q'_j for $j' < j$ and putting k in $Left(j)$ if the state corresponding to it is accepting. We can define all $Left(j)$ for $0 \leq j < t$ simultaneously in $O(n)$ time.

We are now ready to give an algorithm for extracting $\alpha_{\mathcal{L}}$ from $\mathcal{A}$. We iterate through processing each $(1,1)$ state which has a defined $Left$ 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 $Left(j) = B$. If $j < t$ add j to the set B and update $\alpha = \alpha\mathbf{1}$. 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 $\alpha = \alpha\omega$. Otherwise, $(j+i\ell)_{n\in\mathbb{N}}$ is decreasing and we update $B = B \cup V_r$ and $\alpha = \alpha\omega^*$. 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 α: set α_L to be the result of replacing all 1ω in α to ω, all ω^*1 in α to ω^*, and all sequences of $\mathbf{1}$s of length n to $\mathbf{n}$. This algorithm has runtime quadratic in the size of $\mathcal{A}$. 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. □

3.2 State complexity

Let $\mathcal{L} = (\mathbb{N}; \leq_L)$ be a unary automatic linear order. By Theorem 4, the order type of $\mathcal{L}$ is specified by $\alpha_{\mathcal{L}} \in \{\omega, \omega^*, \{\mathbf{n}\}_{n\in\mathbb{N}}\}^\star$. Let $m_{\mathcal{L}}$ be the number of instances of ω or ω^* in $\alpha_{\mathcal{L}}$ and let $k_{\mathcal{L}}$ be the sum of all n such that $\mathbf{n}$ appears in $w_{\mathcal{L}}$. We will express the state complexity of $\mathcal{L}$ in terms of the pair $(m_{\mathcal{L}}, k_{\mathcal{L}})$, whose size is defined to be $\max\{m_{\mathcal{L}}, k_{\mathcal{L}}\}$.

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

Proof. By Lemma 3, the optimal automaton $\mathcal{A}$ for $\mathcal{L}$ has $m_{\mathcal{L}} + k_{\mathcal{L}}$ $(1,1)$-states: $k_{\mathcal{L}}$ many states on the $(1,1)$-tail and $m_{\mathcal{L}}$ many states on the $(1,1)$-loop. Each state on the $(1,1)$-loop represents a copy of ω or ω^* in $\mathcal{L}$ and since this is the minimal automaton there is no interleaving. To specify whether each copy of ω or ω^* is increasing or decreasing and the relative ordering of copies of ω and $\omega^\star$, we need $2\ell = 2m_{\mathcal{L}}$ $(1,\diamond)$ or $(\diamond,1)$ 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 ω, ω^*, we need up to $2(k_{\mathcal{L}} - j + m_{\mathcal{L}})$ states off q_j. Therefore, an upper bound to the number of states in an optimal unary automaton is

$$m_{\mathcal{L}}(2m_{\mathcal{L}}) + \sum_{j=0}^{k_{\mathcal{L}}-1} 2\left(k_{\mathcal{L}} - j + m_{\mathcal{L}}\right) = 2m_{\mathcal{L}}^2 + k_{\mathcal{L}}^2 + 2k_{\mathcal{L}}m_{\mathcal{L}} + k_{\mathcal{L}}.$$

We can improve this bound by realizing that the states on the $(1,1)$-loop corresponding to the rightmost and leftmost symbols in $\alpha_{\mathcal{L}}$ require fewer $(\diamond,1)$ and $(1,\diamond)$ states. The greatest saving of states occurs if the rightmost and leftmost elements of α_L are finite, say $\mathbf{a}$ and $\mathbf{b}$ and the corresponding elements of $\mathcal{L}$ 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 $(1, \diamond)$ or $(\diamond, 1)$ state. We therefore save

$$\sum_{i=0}^{a+b-1} 2(k_{\mathcal{L}} - i + m_{\mathcal{L}}) - 1 \leq 2k_{\mathcal{L}}^2 + 2k_{\mathcal{L}}m_{\mathcal{L}}$$

states. Thus, the minimal automaton must have at least

$$2m_{\mathcal{L}}^2 + k_{\mathcal{L}}^2 + 2k_{\mathcal{L}}m_{\mathcal{L}} + k_{\mathcal{L}} - 2k_{\mathcal{L}}^2 - 2k_{\mathcal{L}}m_{\mathcal{L}} = 2m_{\mathcal{L}}^2 - k_{\mathcal{L}}^2 + k_{\mathcal{L}}$$

states. □

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

This section explores unary automatic equivalence relations. A structure $\mathcal{E} = (\mathbb{N}; E)$ is an *equivalence structure* if E is an equivalence relation (reflexive, symmetric, and transitive). We use Table 1 for the following.

Proposition 2. *The membership problem for automatic equivalence structures is decidable in time $O(n^3)$.*

Blumensath [1] and Khoussainov and Rubin [12] described the structure of unary automatic equivalence structures.

Theorem 7 ([1, 12]). *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 $\mathcal{E}$ is a function $h_{\mathcal{E}} : \mathbb{N} \cup \{\infty\} \to \mathbb{N} \cup \{\infty\}$ such that $h_{\mathcal{E}}(x)$ is the number of E-equivalence classes of size x. Two equivalence structures $\mathcal{E}_1$ and $\mathcal{E}_2$ are isomorphic if and only if $h_{\mathcal{E}_1} = h_{\mathcal{E}_2}$. By Theorem 7, the function $h_{\mathcal{E}}$ for unary automatic equivalence structure $\mathcal{E}$ is finitely nonzero.

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

$$(\exists y_1 \cdots \exists y_{n-1})[\bigwedge_{i=1}^{n-1} (y_i \neq x \wedge E(y_i, x)) \wedge (\forall z)[(\bigwedge_{i=1}^{n-1} (z \neq y_i) \wedge z \neq x) \to \neg E(x, z)].$$

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 $H_n = h_{\mathcal{E}}(n)$.

$$(\exists x_1 \cdots \exists x_H) \left[\bigwedge_{i=1}^{H} (\exists^\infty y) E(x_i, y) \ \wedge \bigwedge_{i,j=1; i \neq j}^{H} \neg E(x_i, x_j) \right.$$
$$\left. \wedge \ \forall x ((\exists^\infty y) E(x, y) \rightarrow \bigvee_{i=1}^{H} E(x, x_i)) \right] \ \wedge \bigwedge_{n: H_n = \infty} \left[(\exists^\infty x) \mathrm{size}_n(x) \right]$$

$$\bigwedge_{m,n: H_n = m} (\exists y_1 \cdots \exists y_m) \left[\bigwedge_{i=1}^{m} \mathrm{size}_n(y_i) \wedge \bigwedge_{i,j=1; i \neq j}^{m} \neg E(y_i, y_j) \right]$$

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. □

The sentence $\varphi_{\mathcal{E}}$ contains two alternations of quantifiers, each causes an exponential blow-up in the size of the automaton corresponding to $\varphi_{\mathcal{E}}$ [10]. This implies that the decision procedure given by Corollary 3 has doubly exponential runtime in the sizes of the input automata. We now significantly improve this bound by providing 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.*

Let $\mathcal{E}$ be recognized by a unary automaton $\mathcal{A}$ with n states. Recall the definitions of t, ℓ, 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 $(\diamond, 1)$ 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 $\Delta(q_j, (\diamond, 1)^{(k-i)\ell})$ is accepting. Let $c > 0$ be the least number such that $\Delta(q_j, (\diamond, 1)^{c\ell}) \in F$. To compute c, we examine $\Delta(q_j, (\diamond, 1)^{d\ell})$ 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 $(\diamond, 1)$-loop off q_j.

Lemma 4. *The set $\{j + i\ell\}_{i \in \mathbb{N}}$ is partitioned into c infinite equivalence classes.*

Proof. Since c is the least such that $\Delta(q_j, (\diamond, 1)^{c\ell})$ is accepting, elements in $\{j, j + \ell, \ldots, j + (c-1)\ell\}$ are pairwise non$E$-equivalent. Moreover for each i, $E(j + i\ell, j + (i + c)\ell)$. □

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 $(\diamond, 1)$-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^i_{s+i} such that the state $r^i_s = \Delta(q_{j_i}, (\diamond, 1)^{(j_{s+i}-j_i)+m^i_{s+i}\ell}) \in F$; moreover, these are the only accepting states on the $(\diamond, 1)$ tail of q_{j_i}. A corresponding set is *maximal* if it is not a subset of a larger corresponding set.

Lemma 5. *For any k, $h_{\mathcal{E}}(k) = \infty$ 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 $c \geq 0$, $\{j_1 + c\ell, j_2 + (c + m^1_2)\ell, \ldots, j_k + (c + m^1_k)\ell\}$ is an $\mathcal{E}$-equivalence class of size k. On the other hand, suppose there are infinitely many $\mathcal{E}$-equivalence classes of size k. For $t \leq j < t + \ell$, let ℓ_j and t_j be the lengths of the $(\diamond, 1)$-loop and -tail off q_j, 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. □

Proof (Proof of Theorem 8). To decide whether two unary automatic equivalence structures $\mathcal{E}_1, \mathcal{E}_2$ are isomorphic we first use the unary automata recognizing E_1 and E_2 to compute their height functions and then check if $h_{\mathcal{E}_1} = h_{\mathcal{E}_2}$. Hence, we begin by giving an algorithm for extracting the height function of a unary automatic equivalence structure $\mathcal{E}$ from a unary automaton $\mathcal{A}$.

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 $\{j' + i\ell\}_{i \in \mathbb{N}}$ accounts for exactly c infinite equivalence classes in $\mathcal{E}$, as described in Lemma 4. At the same time, we can compute all $0 < k < t + \ell$ such that $\Delta(q_{j'}, (\diamond, 1)^{k - j' + m\ell}) \in F$ (where $m < n$). For any such k, $\{k + i\ell\}_{i \in \mathbb{N}}$ is covered by the c infinite equivalence classes from $\{j' + i\ell\}_{i \in \mathbb{N}}$. Using the complexity analysis before Lemma 4, we see that the runtime of computing the total number of infinite equivalence classes is

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

It remains to consider $0 \leq j < t + \ell$ such that q_j has no accepting state on its $(\diamond, 1)$-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 5, we can effectively find all k such that $h_{\mathcal{E}}(k) = \infty$ by searching for the appropriate corresponding set. This can be done by reading the $(\diamond, 1)$-tails off the $(1, 1)$-loop and thus takes times $O(n)$.

By the proof of Lemma 5, for any finite equivalence class K, if $x \geq p$ for all $x \in K$ and $|K| = k$, then $h_{\mathcal{E}}(k) = \infty$. Hence, it only remains to compute the sizes of equivalence classes for elements in $\{t, \ldots, p\}$, which requires reading through the $(\diamond, 1)$-tails off the $(1, 1)$-tail. Again this step has runtime $O(n)$.

In summary, the algorithm that computes $h_{\mathcal{E}}$ from $\mathcal{A}$ has runtime $O(n^2)$. Note that the domain of $h_{\mathcal{E}}$ is a subset of $\{1, \ldots, n, \infty\}$ so comparing it with $h_{\mathcal{E}'}$ 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. □

4.2 State complexity

Given a unary automatic equivalence structure $\mathcal{E} = (\mathbb{N}; E)$, we want to define the optimal unary automaton for $\mathcal{E}$. We will express the state complexity in terms of the height function $h_{\mathcal{E}}$; define the size of $h_{\mathcal{E}}$ to be

$$|h_{\mathcal{E}}| = \sum_{n:h_{\mathcal{E}}(n)<\infty} nh_{\mathcal{E}} + n_{inf} + h_{inf}.$$

Let $h_{\text{inf}} = h_{\mathcal{E}}(\infty)$ and $n_{\text{inf}} = \sum_{n:h_{\mathcal{E}}(n)=\infty} n$.

Lemma 6. *Let $\mathcal{A}$ be a unary automaton recognizing $\mathcal{E}$, then $n_{inf} \leq \ell$.*

Proof. For any n, $h_{\mathcal{E}}(n) = \infty$ if and only if there are $t \leq j_1 < j_2 < \cdots < j_n < t+\ell$ such that $\Delta(q_{j_1}, (\diamond, 1)^{j_i - j_1}) \in F$ for all $i = 1, \ldots, n$ and no other $(\diamond, 1)$ 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 $\mathcal{E} = (\mathbb{N}; E)$ is at least*

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

and at most

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

Proof. We say a collection of $(1,1)$-states $\{r_1, \ldots, r_m\}$ in $\mathcal{A}$ *represents* an E-equivalence class K if for each $x \in K$, $\Delta(\iota, (1,1)^x) = r_i$ 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 $(\diamond, 1)$-tail off r_i. In an optimal unary automaton recognizing $\mathcal{E}$, the length of the $(\diamond, 1)$-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 $(\diamond, 1)$ and $(1, \diamond)$ states associated to the class K is $2\sum_{i=1}^{|K|}(|K| - i) = |K|^2 - |K|$. 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 $\mathcal{E}$ is represented by a single state on the $(1,1)$-loop of an automaton $\mathcal{A}$, then $\ell > h_{\text{inf}}$. Moreover, the $(\diamond, 1)$-loop out of with each such state must have size at least ℓ. 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 ℓ and the $(\diamond, 1)$-loop out of each r_i has length ℓ/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 $h_{\inf}\ell$ and a single accepting state, $\Delta(r,(\diamond,1)^{h_{\inf}\ell})$. With this representation, $1+2h_{\inf}\ell$ states are used for all the infinite equivalence classes (as opposed to more than $h_{\inf}^2+h_{\inf}$). By Lemma 6 and the above discussion, the smallest possible length for the $(1,1)$-loop is $(n_{\inf}+1)$. For each n such that $h_{\mathcal{E}}(n)=\infty$, there are n^2-n $(1,\diamond)$- and $(\diamond,1)$-states off the $(1,1)$-loop. Thus, there must be at least

$$1+2h_{\inf}\ell+\sum_{n:h_{\mathcal{E}}(n)=\infty} n^2$$

states on the $(1,1)$-loop and its peripheries.

We can define an automaton $\mathcal{A}$ which recognizes $\mathcal{E}$ using a $(1,1)$-tail of length $\sum_{n:h_{\mathcal{E}}(n)<\infty} nh_{\mathcal{E}}(n)$ and $(1,1)$-loop of length $n_{\inf}+1$. The total size of $\mathcal{A}$ is

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

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 $h_{\mathcal{E}}(|K|)=\infty$, $h_{\mathcal{E}}(|K'|)<\infty$. With optimal overlapping, the minimum number of states in a unary automaton recognizing $\mathcal{E}$ is

$$\sum_{n:h_{\mathcal{E}}(n)<\infty} n^2h_{\mathcal{E}}(n)+\sum_{n:h_{\mathcal{E}}(n)=\infty} n^2+2h_{\inf}(n_{\inf}+1)+1-c$$

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 $c<\sum_{n:h_{\mathcal{E}}(n)=\infty}(n^2-n)$, as required. □

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

5 Trees

5.1 Characterizing unary automatic trees

This section investigates unary automatic trees. A *tree* is $\mathcal{T}=(\mathbb{N};\leq_T)$ where $\leq_T$ is a partial order t(reflexive, antisymmetric, and transitive) with a root (the least element) and such that the set of *ancestors* of any node x, $\{y:y\leq_T x\}$, is a finite linear order. Two nodes x,y are *incomparable*, $x|_T y$, if $x\not\leq_T y$ and $y\not\leq_T x$; an *anti-chain* of $\mathcal{T}$ is a set of nodes which are pairwise incomparable. Table 1 gives an $O(n^3)$ algorithm for checking whether a given automaton

recognizes a partial order. Checking if $\leq_T$ is total on every set of predecessors $((\forall x, y, z)[y, z \leq_T x \rightarrow y \leq_T z \vee z \leq_T y])$ takes time $O(n^4)$. Checking the existence of a root (least element) 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.

Lemma 7. *There is an $O(n)$ time algorithm that checks if a unary automatic partial order $(\mathbb{N}; \leq_T)$ has a least element.*

Proof. Suppose $\leq_T$ is recognized by unary automaton $\mathcal{A}$ with parameters t, ℓ (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 $q = \Delta(\iota, (1,1)^x) = \Delta(\iota, (1,1)^y)$. By definition of x, $x <_T y$ and $x <_T 2x-y$. Therefore, $\Delta(q, (\diamond, 1)^{y-x}) \in F$ and $\Delta(q, (\diamond, 1)^{x-y}) \in F$. But, this implies that $y <_T x$, a contradiction. 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 $\mathcal{A}$ at most once. □

Proposition 3. *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 complexity bounds. We present such a characterization of unary automatic trees A *parameter set Γ* is a tuple $(\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_m, \sigma, X)$ where $\mathcal{T}_0, \ldots, \mathcal{T}_m$ are finite trees (with disjoint domains T_i), $\sigma : \{1, \ldots, m\} \rightarrow T_0$ and $X : \{1, \ldots, m\} \rightarrow \{\emptyset\} \cup \bigcup_i T_i$ such that $X(i) \in T_i \cup \{\emptyset\}$.

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

- *UF(Γ) 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 $x \in T_i$, 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 $\mathcal{T}$ is unary automatic if and only if there is a parameter set $\Gamma = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_m, \sigma, X)$ such that $\mathcal{T} \cong$ UF(Γ).*

Suppose $\mathcal{T} = (\mathbb{N}; \leq_T)$ is recognized by a unary automaton $\mathcal{A}$ with n states and parameters t, ℓ. We say that two disjoint sets X and Y of nodes in $\mathcal{T}$ are *incomparable* if $(\forall x \in X)(\forall y \in Y)(x|_T y)$.

Lemma 8. *For $t \leq j < t + l$, the set $(j + i\ell)_{i \in \mathbb{N}}$ forms either an anti-chain or finitely many pairwise incomparable infinite chains in T.*

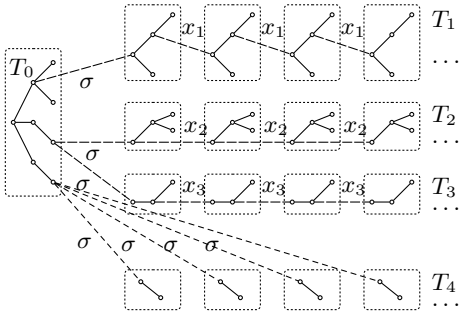

Fig. 2. An example of a tree-unfolding.

Proof. If there is no c such that $\Delta(q_j, (\diamond, 1)^{c\ell})$ 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, $j+m\ell <_T j + (m + in_j)\ell$ for all i and for $0 \leq m < n_j$, thus making $(j + (m + in_j)\ell)_{i\in\mathbb{N}}$ an infinite chain; furthermore elements in $\{j, j + \ell, \ldots, j + (n_j - 1)\ell\}$ are pairwise incomparable. □

By Lemma 8, 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 $\mathcal{T}$ (*interleave* in the sense of Section 3). Fix $j, k \in C$. If there is no m such that $\Delta(q_j, (\diamond, 1)^{k-j+m\ell})$ 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 $\Delta(q_j, (\diamond, 1)^{k-j+m\ell}) \in F$.

Lemma 9. *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, $\Delta(q_j, (\diamond, 1)^{k-j+m\ell})$ is accepting. Hence, $j <_T k + m\ell$ and $k + m\ell \in W_{k,m_1}$ for some $0 \leq m_1 < n_k$. Therefore, $m_1 \equiv m \mod n_k$ and $W_{j,0}, W_{k,m_1}$ interleave in $\mathcal{T}$. Similarly, since $j + n_j\ell <_T k + n_j\ell + m\ell$, there is $0 \leq m_2 < n_k$ such that $m_2 \equiv n_j + m \mod n_k$ and $W_{j,0}, W_{k,m_2}$ interleave in $\mathcal{T}$. 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 $\Delta(q_k, (\diamond, 1)^{j-k+r\ell}) \in F$. 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 $\mathcal{T}$ must be given by element(s) in C. Therefore, Lemma 9 implies that $\mathcal{T}$ contains only finitely many infinite paths We define a *component* of $\mathcal{T}$ to be a connected subgraph of $\mathcal{T}$ 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 9, $(j + i\ell)_{i\in\mathbb{N}}$ belongs to exactly n_j components, $B_0, \ldots, B_{n_j-1}$. If there is no m such that $\Delta(q_j, (\diamond, 1)^{k-j+m\ell})$ 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 $\Delta(q_j, (\diamond, 1)^{k-j+m\ell}) \in F$. 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 $\mathcal{T}$. If there is m such that $\Delta(q_j, (\diamond, 1)^{k-j+m\ell}) \in F$, 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 in $\mathcal{T}$, 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 $\mathcal{T}$; 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 $\mathcal{T}$.

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 $\mathrm{UF}(\Gamma)$ of some parameter set $\Gamma = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_m, \sigma, X)$. For each $\sim$-equivalence class $[j]$, either $[j]$ represents infinitely many independent trees or $[j]$ represents finitely many components of $\mathcal{T}$.

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

Conversely, we will show that if $\Gamma = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_m, \sigma, X)$ is a parameter set, $\mathrm{UF}(\Gamma)$ is a unary automatic tree $\mathcal{T}$. Let $t = |T_0|$, $\ell = \Sigma_{r=1}^{m} |T_r|$ and $\alpha_r = \Sigma_{i=1}^{r-1} |T_i|$ for $r = 1, \ldots, m$. We consider the isomorphic copy $(\mathbb{N}; \leq_T) \cong \mathrm{UF}(\Gamma)$ where $T_0 \mapsto \{0, \ldots, |T_0|\}$ and the j^{th} copy of $\mathcal{T}_r$ 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, ℓ. 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 ℓ. Each q_j on the $(1, 1)$-loop has a $(\diamond, 1)$-tail and $(\diamond, 1)$-loop, each of length ℓ. All $(1, 1)$-states are in F. Let $\varphi_0 : T_0 \to \{0, \ldots, t - 1\}$ and

$\varphi_r : T_r \to \{t + \alpha_r, \ldots, t + \alpha_{r+1} - 1\}$ be isomorphisms that preserve the tree order. We use φ_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 σ, X from the parameter set to specify those state 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 $(\mathbb{N}; L(\mathcal{A})) \cong \mathrm{UF}(\Gamma)$. □

5.2 Efficient solution to the isomorphism problem

We wish to use the characterization of unary automatic trees to solve the isomorphism problem. However, two tree-unfoldings may be isomorphic even if the associated parameter sets are not isomorphic term-by-term. For example, if $\Gamma = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_m, \sigma, X)$ is any parameter set and $\Gamma' = (\mathcal{T}_0', \mathcal{T}_1, \ldots, \mathcal{T}_m, \sigma', X)$ where $\mathcal{T}_0'$ is the subtree of $\mathrm{UF}(\Gamma)$ containing one copy of each $\mathcal{T}_0, \ldots, \mathcal{T}_m$ and σ' is obtained from σ by setting $\sigma(i) = X(i)$ if $X(i) \neq \emptyset$, then $\mathrm{UF}(\Gamma) \cong \mathrm{UF}(\Gamma')$. In previous section, we obtained canonical isomorphism invariants: $\alpha_{\mathcal{L}}$ for linear orders and $h_{\mathcal{E}}$ for equivalence structures. We now define an analogue for trees. Fix a computable linear order $\preceq$ on the set of finite trees.

Definition 3. *The* canonical parameter set *of a unary automatic tree* $\mathcal{T} = (\mathbb{N}; \leq_T)$ *is the parameter set* $\Gamma = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_m, \sigma, X)$ *such that* $UF(\Gamma) \cong \mathcal{T}$ *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* $y_1 \leq_T y_2 \leq_T X(i)$ *the subtree with domain* $\{z : y_1 \leq_T z \wedge y_2 \not\leq_T z\}$ *is not isomorphic to the subtree with domain* $\{z : y_2 \leq_T z \wedge X(i) \not\leq_T z\}$.
4. T_0 *is minimal:* T_0 *has the fewest possible nodes and for all* $1 \leq i \leq m$ *where* $X(i) \neq \emptyset$, *there is no* $y \in T_0$ *such that* $y \leq_T \sigma(i)$ *and the subtree with domain* $\{z : y \leq_T z \wedge \sigma(i) \not\leq_T z\}$ *is isomorphic to* T_i.

Lemma 10. *Suppose* $\mathcal{T}, \mathcal{T}'$ *are unary automatic trees with canonical parameter sets* Γ, Γ'. *Then,* $\mathcal{T} \cong \mathcal{T}'$ *if and only if* Γ, Γ' *have the same number* (m) *of finite trees,* $(\mathcal{T}_0, \sigma) \cong (\mathcal{T}_0', \sigma')$, *and for* $1 \leq i \leq m$, $(\mathcal{T}_i, X(i)) \cong (\mathcal{T}_i', X'(i))$.

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

$$\begin{aligned}(\mathcal{T}_0, \sigma) &\cong (\{y : y \in T_0 \wedge (\forall 1 \leq i \leq m) \neg t_i \leq_T y\}; \leq_T) \\ &\cong (\{y : y \in T_0' \wedge (\forall 1 \leq i \leq m) \neg t_i' \leq_{T'} y\}; \leq_{T'}) \cong (\mathcal{T}_0', \sigma')\end{aligned}$$

□

The canonical parameter set can now be used to define an extended first-order formula $\varphi_{\mathcal{T}}$ which specifies the isomorphism type of $\mathcal{T}$ (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 $O(n^3)$ 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 10 and a decision procedure for isomorphism on finite trees to solve the isomorphism problem on unary automatic trees.

Lemma 11. *If $\leq_T$ is recognized by unary automaton with n states, there is an $O(n^3)$ time algorithm that computes the canonical parameter set of $\mathcal{T}$*

Proof. We divide the construction into two pieces: first compute some parameter set Γ for $\mathcal{T}$, then compute the canonical parameter set from Γ. 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 $(\diamond, 1)$-tail and loop out of each state on the $(1, 1)$-loop. For each $t \leq j < t + \ell$, let ℓ_j be the length of $(\diamond, 1)$-loop out of q_j, and $\tilde{t}_j$ be sum of the lengths of $(\diamond, 1)$-tail and $(1, \diamond)$-tail out of q_j. Checking (as many as) ℓ_j many states on the $(\diamond, 1)$-loop and $\tilde{t_j}$ other states allows us to determine both n_j and the class $[j]$. In all, this takes time $O\left(\sum_{j=t}^{t+\ell-1}(\ell_j + \tilde{t}_j)\right)$

Suppose $[j]$ represents finitely many components in $\mathcal{T}$. Each component is obtained by unfolding a finite tree $\mathcal{T}'$ of size $|[j]|$ on some $x \in T'$. The tree order $\leq_{T'}$ can be computed by reading all the $(\diamond, 1)$- and $(1, \diamond)$-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 ℓ_j and computing all $\mathcal{T}'$ takes time $O(\sum_{j=t}^{t+\ell-1} \ell_j + \tilde{t}_j)$. We need n_j isomorphic copies of $\mathcal{T}'$ in Γ, a total of $O(n_j|[j]|)$ nodes. Thus, to define all $\mathcal{T}'$ 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 $\mathcal{T}'$ isomorphic to these independent trees, we read the $(\diamond, 1)$- and $(1, \diamond)$-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 $\mathcal{T}'$ 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 $\mathcal{T}$.

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 $(\diamond, 1)$-tail out of q_k and ℓ_k be the length of the $(\diamond, 1)$-loop out of q_k. We describe an

algorithm that compute the parents of $[j]$. The algorithm processes the subtree of $\mathcal{T}$ 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 $\Delta(q_k, (\diamond,1)^{i\ell+j-k}) \in F$. We can find such an i if and only if there are infinitely many independent trees associated to $[j]$ descending from k in $\mathcal{T}$.
- *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 $\{t_{k_i},\ldots,t_{k_i}+\ell_{k_i}\}$ defined as

  $$U_i = \{x : \Delta(q_k,(\diamond,1)^{x\ell+j-k}) \in F\}, \qquad V_i = \{x : \Delta(q_{k_i},(\diamond,1)^{x\ell+j-k_i}) \in F\},$$

  and $D_i = U_i - V_i$. Let $\ell' = \max\{\ell_{k_1},\ldots,\ell_{k_r}\}$ and let $D'_i = \{x + i\ell_k : x \in D_i \wedge x + i\ell_k < \ell'\ell_k\}$. 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 $\Delta(q_k,(\diamond,1)^{i'\ell+j-k}) \in F$, there are only finitely many independent trees represented by $[j]$ descending from k. Moreover, if such an i' exists, it must be on the $(\diamond,1)$-loop off q_k and so we can stop looking for it after we have checked all ℓ_k states. For Case 2, note that $x \in D'_1 \cap \cdots \cap D'_r$ if and only if k is the immediate ancestor for all nodes in $\{j + (x + i\ell'\ell_k)\ell\}_{i\in\mathbb{N}}$, if and only if k is the immediate ancestor for some node in $\{j + (x + i\ell'\ell_k)\ell\}_{i\in\mathbb{N}}$. Therefore if $D'_1 \cap \cdots \cap D'_r \neq \emptyset$, then k is a parent of $[j]$. Suppose $D'_1 \cap \cdots \cap D'_r = \emptyset$ and k is a parent of $[j]$. Then k is the immediate ancestor for $\{j + (s + im)\ell\}_{i\in\mathbb{N}}$ for some $m > \ell'\ell_k$ and $s < m$. If $s < \ell'\ell_k$, then $s \in D'_1 \cap \cdots \cap D'_r$. Therefore $s \geq \ell'\ell_k$. Say $s = s' + i\ell'\ell_k$ where $s' < \ell'\ell_k$. Then k is the immediate ancestor of $j + (s + \ell'\ell_k m)\ell = j + (s' + (m+i)\ell\ell_k)\ell$. Therefore $s' \in D'_1 \cap \cdots \cap D'_r$. Contradiction. Hence the algorithm is correct.

Complexity. Checking if a leaf node k is a parent for $[j]$ takes time $O(\ell_k)$. When k is an internal node, computing U_i and V_i takes time $O(\ell_k\ell_{k_i})$. The size of each U_i and V_i is bounded by ℓ_{k_i}, therefore computing D_i takes $O(\ell_{k_i})$. Computing each D'_i takes time $O(\ell'\ell_k)$. 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 $O(t\hat{\ell}^2)$, where $\hat{\ell}$ is the maximal $(\diamond,1)$-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 intersection operations, each taking time $O(\hat{\ell}^2)$. Since $\hat{\ell} < n$, 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. □

With the above claim in hand, we can resume our construction of the parameter set Γ. The finite tree $\mathcal{T}_0$ in Γ contains all nodes in $\{0,\ldots,t-1\}$ and finitely many independent trees. Deciding which independent trees to put into $\mathcal{T}_0$ uses the claim and therefore takes $O(n^3)$. Computing the tree order $\leq_{T_0}$ on

$\{0, \ldots, t-1\}$ requires reading the $(\diamond, 1)$- and $(1, \diamond)$-tail out of each q_k $(0 \leq k < t)$ at most once. This steps again takes time $O(n)$. Thus, in time $O(n^3)$, we have computed $\Gamma = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_m, \sigma, X)$ such that $\mathcal{T} \cong \mathrm{UF}(\Gamma)$. Since nodes in $\mathcal{T}_0$ can be parents to more than one anti-chain, $m \leq t\ell + \sum_{j \in C} n_j \leq t\ell + n$.

We now use Γ to obtain a canonical parameter set for $\mathcal{T}$. For each $1 \leq i \leq m$ 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 $\{z : y_1 \leq_T z \wedge y_2 \not\leq_T z\}$ is isomorphic to the subtree with domain $\{z : y_2 \leq_T z \wedge X(i) \not\leq_T z\}$. If such y_1, y_2 exist, remove all $z \geq_{T_i} y_1$ 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 [5], 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 $x \leq_T \sigma(i)$, and the subtree of T_0 with domain $\{y : x \leq_T y \wedge t_i \not\leq_T y\}$ 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 Γ to a canonical parameter set takes $O(n^3)$ and thus we have built such a canonical parameter set in $O(n^3)$ time. □

Proof (Theorem 11). Suppose $\mathcal{T}_1, \mathcal{T}_2$ are presented by unary automata $\mathcal{A}_1, \mathcal{A}_2$ with n_1, n_2 states (respectively). Let $n = \max\{n_1, n_2\}$ By Theorem 10 and Lemma 11, deciding if $\mathcal{T}_1 \cong \mathcal{T}_2$ reduces to checking finitely many isomorphisms of finite trees. The appropriate canonical parameter sets are built in $O(n^3)$ time and each have $O(n^2)$ finite trees, each of size $O(n)$. Hence, this isomorphism algorithm runs in $O(n^3)$ time. □

5.3 State complexity

Suppose $\mathcal{T} = \mathrm{UF}(\Gamma)$ and $\Gamma = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_m, \sigma, X)$ is the canonical parameter set of T. Let $t = |T_0|$ and $\ell = \sum_{i=1}^{m} |T_i|$. The proof of Theorem 10 gives an upper bound on the state complexity of unary automatic trees in terms of t and ℓ.

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

Proof. The automaton $\mathcal{A}$ built in the proof of Theorem 10 has size $t(t + \ell) + 2\ell^2 + t + \ell$. To further reduce the number of states, we can permute the domain

of the tree so that if $j \in A$ then q_j has a $(\diamond, 1)$-tail of length ℓ and a $(\diamond, 1)$-loop of length 1, and if $j \in C$ then q_j has a $(\diamond, 1)$-loop of length ℓ and no $(\diamond, 1)$-tail. Therefore the size of $\mathcal{A}$ is $t(t+\ell) + \ell^2 + t + \ell = (t+\ell)^2 - t\ell + t + \ell$.

When $\mathcal{T}_1, \ldots, \mathcal{T}_m$ are pairwise non-isomorphic, the loop length of $\mathcal{A}$ is at least ℓ and there are at least ℓ $(\diamond, 1)$-states out of each q_j on the $(1, 1)$-loop. Therefore the state complexity is bounded below by ℓ^2. □

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

6 Graphs of Finite Degree

This section studies unary automatic graphs of finite degree. A graph $\mathcal{G} = (\mathbb{N}; R)$ is of *finite degree* if $(\forall x)(\neg\exists^\infty y)(R(x,y) \vee R(y,x))$. If R is recognized by a unary automaton, $\mathcal{G}$ is of finite degree if and only if there is no accepting state on any $(\diamond, 1)$- or $(1, \diamond)$-loops. Therefore, the membership problem is decidable in linear time. In [7], 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. In particular, when the automaton is fixed, deciding if there is a path from vertex x to vertex y takes linear time in x and y. Furthermore, [7] showed that connectedness of unary automatic graphs of finite degrees is decidable in $O(n^3)$ time and the following result which we will use later.

Theorem 13 ([7]). *Given a unary automatic graph $\mathcal{G}$ of finite degree, we can effectively construct a unary automaton that recognizes the reachability relation on $\mathcal{G}$ in polynomial time.*

However, [7] 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 simplicity, we assume $\mathcal{G}$ is undirected. The case of directed graphs can be treated in a similar manner. We use the following characterization from [7]. Given a finite graph $\mathcal{F} = (V_\mathcal{F}; E_F)$ and a map $\sigma : V_\mathcal{F} \to \mathcal{P}(V_\mathcal{F})$, we define $\mathcal{F}_{\sigma^\omega}$ to be the disjoint union of infinitely many copies of $\mathcal{F}$ with added edges: there is an edge between $x \in \mathcal{F}_i$ and $y \in \mathcal{F}_{i+1}$ if and only if $y \in \sigma(x)$.

Theorem 15 ([7]). *A graph of finite degree $\mathcal{G} = (\mathbb{N}; R)$ is unary automatic if and only if there are finite graphs $\mathcal{D}, \mathcal{F}$ and a map $\sigma : V_\mathcal{F} \to P(V_\mathcal{F})$ such that $\mathcal{G} \cong \mathcal{G}'$ and $\mathcal{G}'$ is a disjoint union of $\mathcal{D}$ and $\mathcal{F}_{\sigma^\omega}$ with possible additional edges between $\mathcal{D}$ and $\mathcal{F}_0$. Furthermore, the parameters $\mathcal{D}, \mathcal{F}, \sigma$ can be extracted in time $O(n^2)$ from a unary automaton recognizing R.*

A *component of* $\mathcal{G}$ is the transitive closure of a vertex under the edge relation, R. By Theorem 15, any infinite component in $\mathcal{G}$ has nonempty intersection with almost all $\mathcal{F}_i$. Therefore, $\mathcal{G}$ has at most $|V_{\mathcal{F}}|$ many infinite components. Similarly, any finite component of $\mathcal{G}$ has size at most $|V_{\mathcal{D}} + V_{\mathcal{F}}|$. We say a component C *starts in* $\mathcal{F}_i$ if $C \cap \mathcal{F}_i \neq \emptyset$ and for $j < i$, $C \cap \mathcal{F}_j = \emptyset$. Let $\ell = |V_{\mathcal{F}}|$.

Lemma 12. *For any finite graph $\mathcal{H}$, there are infinitely many components in $\mathcal{G}$ isomorphic to $\mathcal{H}$ if and only if $\mathcal{H} \cong C$ for some component C starting in $\mathcal{F}_\ell$.*

Proof. Suppose $C \cong \mathcal{H}$ and C starts in $\mathcal{F}_\ell$. For each j, $\{(v, i+j) : (v,i) \in C\}$ is a finite component isomorphic to $\mathcal{H}$. On the other hand, if $\mathcal{H}$ is isomorphic to infinitely many components in $\mathcal{G}$, it is isomorphic to some C' that starts in $\mathcal{F}_k$ for $k \geq \ell$. Then$\{(v, i+\ell-k) : (v,i) \in C'\}$ is the desired component. □

Let $\mathcal{G}_{\text{Fin}}$ be the subset of $\mathcal{G}$ containing only its finite components. By Theorem 15 and Lemma 12, if C is any finite component of $\mathcal{G}$ then either $C \cap \mathcal{F}_j \neq \emptyset$ for some $j < \ell$ or C has infinitely many isomorphic copies in $\mathcal{G}$. By Lemma 12, $\mathcal{G}_{\text{Fin}}$ there are only finitely many isomorphism classes of finite components of $\mathcal{G}$, and we can decide which of these classes correspond to infinitely many components in $\mathcal{G}$. Since finite graph isomorphism is decidable, given two graphs $\mathcal{G}, \mathcal{G}'$ we can decide if $\mathcal{G}_{\text{Fin}} \cong \mathcal{G}'_{\text{Fin}}$.

Since $\mathcal{G}$ 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 $\mathcal{G}$ 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. To prove Theorem 14 we will give a MSO definition (in the language of graphs) of the isomorphism type of a connected graph $\mathcal{G} = (\mathbb{N}; R)$ and then use the decidability of the MSO theory of unary automatic structures. We first define auxiliary MSO-formulae. For a fixed set S and $k \in \mathbb{N}$, let $\text{Partition}_k^S(P_1, \ldots, P_k)$ be the formula

$$\left(\bigwedge_{i=1}^{k} \exists^\infty x (x \in P_i)\right) \wedge \left(\bigwedge_{1 \leq i \neq j \leq k} P_i \cap P_j = \emptyset\right) \wedge \left(S = \bigcup_{i=1}^{k} P_i\right)$$

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

$$\bigwedge_{i=1}^{k} (\exists^{=1} x_i)(x_i \in X \cap Y_i) \wedge \bigwedge_{i,j=1}^{k} \left(x_i \in X \cap Y_i, x_j \in X \cap Y_j \rightarrow E_{\mathcal{F}}(x_i, y_j) \leftrightarrow E_{\mathcal{F}}(v_i, v_j)\right)$$

For a finite graph $\mathcal{F}$ of size k and a mapping $\sigma : V_{\mathcal{F}} \rightarrow P(V_{\mathcal{F}})$, let $\mathcal{F}_{\times 3}$ be the finite subgraph $\mathcal{F}_{\sigma^\omega} \restriction \mathcal{F}_0 \cup \mathcal{F}_1 \cup \mathcal{F}_2$. Label $V_{\mathcal{F}_{\times 3}}$ by $v_1, \ldots, v_{3k}$ where $\{v_{ik+1}, \ldots, v_{(i+1)k}\}$ 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

$\mathrm{Succ}^{\mathcal{F}}_{\sigma}(X, Y, Z_1, \ldots, Z_{3k})$ to be

$$\begin{aligned}
&\mathrm{Type}^{\mathcal{F}}(X, Z_1, \ldots, Z_{3k}) \wedge \mathrm{Type}^{\mathcal{F}}(Y, Z_1, \ldots, Z_{3k}) \wedge (X \cap Y = \emptyset) \\
&\wedge \bigwedge_{(i,j):v_j \in \sigma(v_i)} (\forall x \in X \cap Z_{2k+i})(\forall y \in Z_j)(E_{\mathcal{F}}(x,y)) \\
&\wedge \bigwedge_{(i,j):v_j \notin \sigma(v_i)} (\forall x \in X \cap Z_{2k+i})(\forall y \in Z_j)(\neg E_{\mathcal{F}}(x,y)) \\
&\wedge \bigwedge_{(i,j) \notin \{2k+1,\ldots,3k\} \times \{1,\ldots,k\}} (\forall x \in X \cap Z_i)(\forall y \in Z_j)(\neg E_{\mathcal{F}}(x,y)).
\end{aligned}$$

We are now ready to prove the theorem.

Proof (Theorem 14). Suppose $\mathcal{G}$ is a connected unary automatic graph of finite degree. By Theorem 15, we can find $\mathcal{D}, \mathcal{F}, \sigma$ such that $\mathcal{G}$ is isomorphic to $\mathcal{D}$ followed by $\mathcal{F}_{\sigma^\omega}$. Since $\mathcal{D}$ and the edge relation between $\mathcal{D}$ and $\mathcal{F}_0$ are finite, they can be used as parameters in a MSO sentence. Hence, for simplicity we assume that $\mathcal{D}$ is empty. Let $V_{\mathcal{F}} = \{v_0, \ldots, v_k\}$ and recall the definition of $\mathcal{F}_{\times 3}$ above. We define $\varphi_{\mathcal{G}}$ as $(\exists P_1 \cdots \exists P_{3k})(\psi_{\mathcal{G}}(P_1, \ldots, P_{3k})$, where $\psi_{\mathcal{G}}(\overline{Z})$ is the conjunction of the following formulas:

1. $\mathrm{Partition}^{\mathbb{N}}_{3k}(\overline{Z})$
2. $(\forall x \exists X)[x \in X \wedge \mathrm{Type}^{\mathcal{F}_{\times 3}}(X, \overline{Z})]$
3. $(\forall X)[\mathrm{Type}^{\mathcal{F}_{\times 3}}(X, \overline{Z}) \rightarrow (\exists^{=1} Y)\mathrm{Succ}^{\mathcal{F}}_{\sigma}(X, Y, \overline{Z})]$
4. $(\exists X)[\mathrm{Type}^{\mathcal{F}_{\times 3}}(X, \overline{Z}) \wedge (\forall Y)[(\mathrm{Type}^{\mathcal{F}_{\times 3}}(Y, \overline{Z}) \wedge X \cap Y = \emptyset)$
   $\rightarrow [\neg \mathrm{Succ}^{\mathcal{F}}_{\sigma}(X, Y, \overline{Z}) \wedge (\exists^{=1} W)\mathrm{Succ}^{\mathcal{F}}_{\sigma}(W, Y, \overline{Z})]]]$

Claim. If $\mathcal{H}$ is an infinite connected graph, $\mathcal{H} \models \varphi_{\mathcal{G}}$ if and only if $\mathcal{H} \cong \mathcal{G}$.

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

Since $\mathcal{H} \models \varphi_{\mathcal{G}}$, each vertex v in $\mathcal{H}$ belongs a unique subgraph that is a $\mathcal{F}_{\times 3}$-type; and, for each $\mathcal{F}_{\times 3}$-type $\mathcal{M}$, there is a unique $\mathcal{F}_{\times 3}$-type $\mathcal{N}$ that is a *successor* of $\mathcal{M}$, i.e., all edges between $\mathcal{M}$ and $\mathcal{N}$ are from the last copy of $\mathcal{F}$ in $\mathcal{M}$ to the first copy of $\mathcal{F}$ in $\mathcal{N}$ such that they respect the mapping σ. Lastly there exists a unique $\mathcal{F}_{\times 3}$-type $\mathcal{M}_0$ which is not the successor of any other $\mathcal{F}_{\times 3}$-types and any other $\mathcal{F}_{\times 3}$-type is the successor of a unique $\mathcal{F}_{\times 3}$-type. Note that the successor relation between the $\mathcal{F}_{\times 3}$-types resembles the unfolding operation on finite graphs.

Therefore to set up an isomorphism from $\mathcal{H}$ to $\mathcal{G}$, we only need to map $\mathcal{M}_0$ isomorphically to the first 3 copies of $\mathcal{F}$ in $\mathcal{G}$, and then map the other vertices according to the successor relation and mapping σ. □

By Theorem 2, satisfiability 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 $\varphi_{\mathcal{G}}$ 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. □

References

1. A. Blumensath. *Automatic Structures*. Diploma thesis, RWTH Aachen, October 1999.
2. A. Blumensath and E. Grädel. Automatic structures. In *Proc. 15th LICS*, pages 51–62. IEEE Computer Society, 2000.
3. K. Salomaa C. Campeanu, K. Culik II and S. Yu. State complexity of basic operations on finite languages, automata implementation. In O. Boldt et al., editor, *Proc. of Fourth International Workshop on Implementing Automata, WIA'99*, volume 2214 of *LNCS*, pages 60–70, 2001.
4. B.R. Hodgson. On direct products of automaton decidable theories. *Theoretical Computer Science*, 19:331–335, 1982.
5. J.E. Hopcroft and J.K. Wong. Linear time algorithm form isomorphism of planar graphs (preliminary report). In *Proc. 6th STOC*, pages 172–184, 1974.
6. J. Ullman J.E. Hopcroft, R. Motwani. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2001.
7. B. Khoussainov, J. Liu, and M. Minnes. Unary automatic graphs: An algorithmic perspective. In M. Agrawal et al., editor, *Proc. 5th TAMC*, volume 4978 of *LNCS*, pages 548–559. Springer-Verlag, 2008.
8. B. Khoussainov and M. Minnes. Model theoretic complexity of automatic structures (extended abstract). In M. Agrawal et al., editor, *Proc. 5th TAMC*, volume 4978 of *LNCS*, pages 520–531. Springer-Verlag, 2008.
9. B. Khoussainov and M. Minnes. Three lectures on automatic structures. In *Proceedings of Logic Colloquium 2007*. Cambridge University Press, 2008.
10. B. Khoussainov and A. Nerode. Automatic presentations of structures. In D. Leivant, editor, *International Workshop on Logic and Computational Complexity*, volume 960 of *LNCS*, pages 367–392. Springer-Verlag, 1995.
11. B. Khoussainov, A. Nies, S. Rubin, and F. Stephan. Automatic structures: Richness and limitations. In *Proc. 19th LICS*, pages 44–53. IEEE Computer Society, July 2004.
12. B. Khoussainov and S. Rubin. Graphs with automatic presentations over a unary alphabet. *Journal of Automata, Languages and Combinatorics*, 6(4):467–480, 2001.
13. G. Olver and R. Thomas. Automatic presentation for finitely generated groups. In *Proc. of the 5th International Conference on Development in Language Theory*, LNCS, pages 130–144. Springer, 2002.
14. S. Rubin. *Automatic Structures*. PhD thesis, University of Auckland, 2004.
15. S. Rubin. Automata presenting structures: A survey of the finite string case. *Bulletin of Symbolic Logic*, 14(2):169–209, June 2008.
16. M.Y. Vardi. Model checking for database theoreticians. In *Proc. 10th International Conference on Database Theory*, 2005.
17. N. Vinokurov. Complexity of some natural problems in automatic structures. *Siberian Mathematical Journal*, 46:56–61, 2005.
18. S. Yu. Chapter 2: Regular languages. *Handbook of Formal Languages*, 1997.
19. S. Yu. State complexity: recent results and open problems. *Fundamenta Informaticae*, 64(1-4):471–480, 2005.