Theoretical Computer Science 412 (2011) 1705–1717

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Deciding the isomorphism problem in classes of unary automatic structures

Jiamou Liu [a,*], Mia Minnes [b]

[a] Department of Computer Science, University of Auckland, Auckland, New Zealand

[b] Department of Mathematics, MIT, Cambridge, MA, USA

ARTICLE INFO

Keywords:
Unary automatic structures
The isomorphism problem
Graphs

ABSTRACT

We solve the isomorphism problem for certain classes of unary automatic structures: unary automatic equivalence relations, unary automatic linear orders, and unary automatic trees. That is, we provide algorithms which decide whether two given elements of these classes are isomorphic. In doing so, we define new finite representations for these structures which give normal forms.[1]

1. Introduction

The *isomorphism problem*, which asks for a decision procedure to decide whether two given members of a class of structures are isomorphic, is central in studying the effective content of mathematical objects. Over finite structures, the isomorphism problem has been one of the most challenging problems in complexity theory (it belongs to NP but is neither known to be in P nor known to be NP-complete) [1]. Over computable structures (those where the domain and atomic relations of the structure are computable), it is well-known that the isomorphism problem is undecidable; in fact, it is complete for Σ^1_1, the first level of the analytical hierarchy [2]. In [3], Khoussainov and Nerode initiate a systematic study of *automatic structures*, those where elements are encoded as strings over a finite alphabet and whose domain and atomic relations are represented by finite automata (precise definitions in Section 2). Automatic structures form an intermediate class of structures between the finite structures and the computable structures. This paper focuses on the isomorphism problem for *unary automatic structures*, the subclass of automatic structures (which still contains all finite structures) of structures whose domains are encoded as strings over a one letter alphabet.

Automatic structures have decidable first-order theories [3]. In general, their monadic second-order theories (where quantification over sets is permitted) are undecidable; see [4–6] for an overview of automatic structures. Since key applications for automatic structures include modeling databases [7] and verifying programs [8], applying the transitive closure operator is often desirable. However, this operator is expressible in monadic second-order logic but is not first-order definable: reachability is undecidable for automatic structures in general. On the other hand, unary automatic structures 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 [5]. Moreover, if we consider the intermediate class of structures whose domain elements are encoded as finite strings over 1^*2^*, insufficient decidability strength results: since the infinite grid can be coded automatically over 1^*2^* 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.

* Corresponding author.
E-mail addresses: jliu036@ec.auckland.ac.nz (J. Liu), minnes@math.mit.edu (M. Minnes).

[1] A related paper which focuses on time and space complexity for these unary automatic structures has been submitted for publication by the authors elsewhere.

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.12.045

The broad decidability of unary automatic structures can be exploited when they are used to model streaming databases. Databases with entries encoded as strings of 1s are well-suited to situations in which provisional results must be updated on the fly (using the tally representation) and computations are performed in real time.

In this paper, we study the isomorphism problem in classes of unary automatic structures. These structures include unary automatic equivalence relations, linear orders, and trees. We use (known and new) characterizations of members of these classes to get normal forms and polynomial-time (in these normal forms) algorithms for the isomorphism problem.

The isomorphism problem has been studied for other collections of graphs. For automatic graphs, it is Σ^1_1-complete [9]. On the other hand, the isomorphism problem is decidable for equational graphs [10]. Any monadic second-order expressible question is decidable in the class of unary automatic graphs. However, the isomorphism problem is not a priori expressible in this way and it is not known whether it is decidable. This paper works towards a solution of this question by looking at special subclasses of unary automatic graphs.

Many natural graph problems (such as graph connectivity and reachability) are expressible in monadic second-order logic and are hence decidable for unary automatic graphs. Deciding these questions by a translation of monadic second-order formulas yields very slow algorithms (non-elementary complexity). Khoussainov et al. [11] exploited structural properties of unary automatic graphs with finite degree to solve these questions in polynomial-time.

In general, understanding the structural properties of a class of unary automatic structures leads to more efficient algorithms. Khoussainov and Rubin [12] and Blumensath [4] characterized unary automatic graphs in terms of relations between two finite graphs. This led to characterizations of unary automatic linear orders and equivalence structures as well (see Theorems 4.2 and 5.2). We prove an analogous result for unary automatic trees (see Theorem 6.3). We use these structural characterizations to define concise finite representations of unary automatic structures. These representations lead to polynomial-time algorithms for the isomorphism problem.

Paper organization. Section 2 recalls the definitions of finite automata and automatic structures. Section 3 discusses the special case of unary automatic structures. Sections 4–6 discuss equivalence structures, linear orders, and trees (respectively). We give polynomial-time algorithms solving the isomorphism problem for the class of structures considered in each section. We conclude in Section 7 and mention open questions.

The authors would like to thank the anonymous referees for helpful comments and corrections.

2. Preliminaries

A finite automaton can be seen as a restricted Turing machine which has a fixed finite bound on its resources and is allowed only a single read pass over the input data. More formally, a *finite automaton* $\mathcal{A}$ over a finite alphabet Σ is a tuple (S, ι, Δ, F), where S is a finite set of *states*, $\iota \in S$ is the *initial state*, $\Delta : S \times \Sigma \to S$ is the *transition function*, and $F \subset S$ is the set of *final* or *accepting* states. In this paper we require Δ to be a well-defined total function and hence $\mathcal{A}$ is a *complete* and *deterministic* automaton. An *input* to $\mathcal{A}$ is a finite string in Σ^*; the empty string is denoted by λ. A *computation* of $\mathcal{A}$ on a word $\sigma_1\sigma_2 \ldots \sigma_n \in \Sigma^*$ is a sequence of states, say $q_0, q_1, \ldots, q_n$, such that $q_0 = \iota$ and $(q_i, \sigma_{i+1}, q_{i+1}) \in \Delta$ for all $i \in \{0, 1, \ldots, n-1\}$. If $q_n \in F$ then the computation is *successful* and the automaton $\mathcal{A}$ *accepts* the word. As such, if $q \in F$ we say that q is an *accepting* state. The *language* of $\mathcal{A}$, $L(\mathcal{A})$, is the set of all words accepted by $\mathcal{A}$. In general, $D \subset \Sigma^*$ is *FA recognizable*, or *regular*, if D is the language of some finite automaton. If $\mathcal{A}$ is a finite automaton over the unary alphabet $\{1\}$ it is called a *unary automaton* and its language is a *unary automatic* subset of $\{1\}^*$.

A (relational) *structure* $\mathcal{S} = (D; R)$. *Synchronous 2-tape automata* recognize binary relations. Such automata have two input tapes, each of which contains one of the input words. Bits of the two input words are read in parallel at the same rate until both input strings have been completely processed. Formally, let $\Sigma_\diamond = \Sigma \cup \{\diamond\}$ where $\diamond$ is a symbol not in Σ. Given a pair of words $w_1, w_2 \in \Sigma^*$, the *convolution* of (w_1, w_2) is a word $\otimes(w_1, w_2)$ over the alphabet $(\Sigma_\diamond)^2$ with length $\max(|w_1|, |w_2|)$. The kth symbol of $\otimes(w_1, w_2)$ is (σ_1, σ_2) where σ_i is the kth symbol of w_i if $k \leq |w_i|$, and is $\diamond$ otherwise. A binary relation R is *FA recognizable* if the set of convolutions of all pairs $(w_1, w_2) \in R$ is a regular subset of $(\Sigma^2_\diamond)^*$.

A structure is called *automatic* over Σ if its domain is a regular subset of Σ^* and each of its basic 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 isomorphic to unary 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 isomorphic copies. The structure $(\mathbb{N}; \times)$ has no automatic isomorphic copy. For proofs of these facts, see the survey papers [13,14].

The class of languages recognizable by finite automata is closed under the rational operations studied by Kleene. These operations parallel Boolean set operations and the first-order quantifiers. Consider the first-order logic extended by $\exists^\omega$ (there exist infinitely many) and $\exists^{n,m}$ (there exist n many mod m, where n and m are natural numbers) quantifiers. We denote this logic by $\mathsf{FO} + \exists^\omega + \exists^{n,m}$. The following theorem from [15,3,16,5] connects this extended logic with automata. The automata corresponding to formulas with n free variables are synchronous n-tape automata, a natural extension of the automata for binary relations above.

Theorem 2.1 (*Blumensath, Grädel; 1999. Khoussainov, Rubin; 1999*)**.** *For an automatic structure $\mathcal{S}$ there is an algorithm that, given a formula $\varphi(\bar{x})$ in $\mathsf{FO} + \exists^\omega + \exists^{n,m}$, produces an automaton whose language is those tuples $\bar{a}$ from $\mathcal{S}$ that make φ true.*

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	$\forall x, y\,(R(x,y) \implies R(y,x))$	$O(n^2)$
Anti-symmetry	$\forall x, y\,(R(x,y) \wedge R(y,x) \implies x = y)$	$O(n^2)$
Totality	$\forall x, y\,(R(x,y) \vee R(y,x))$	$O(m^2n^2)$
Transitivity	$\forall x, y, z\,(R(x,y) \wedge R(y,z) \implies R(x,z))$	$O(n^3)$

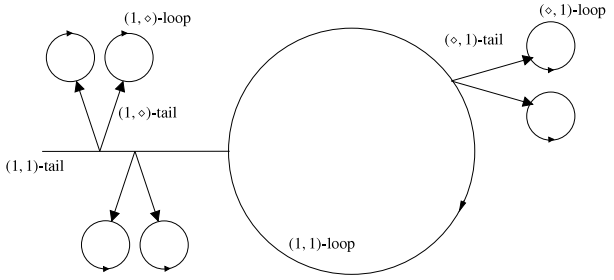

Fig. 1. General shape of a deterministic 2-tape unary automaton.

The proof of Theorem 2.1 along with the fact that there is a linear-time algorithm which tests whether the language of a given automaton is empty yield the following corollaries.

Corollary 2.2. *The* $(\mathsf{FO} + \exists^{\omega} + \exists^{n,m})$*-theory of an automatic structure* $\mathcal{S}$ *is decidable.*

Corollary 2.3. *Given deterministic automata* $\mathcal{A}_1$ *(m states) and* $\mathcal{A}_2$ *(n states), there is an* $O(mn)$*-time algorithm to build the deterministic union or intersection automaton (mn states) of* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *and an* $O(m)$*-time algorithm to build the deterministic complement automaton (m states) of* $\mathcal{A}_1$*.*

Let $(D; R)$ be an automatic structure (over Σ), with R a binary relation over D. 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. By Corollary 2.2, it is decidable whether $(D; R)$ satisfies these properties. In particular, to check if R is reflexive, we construct an automaton for $\{x \mid (x,x) \in R\}$ and check if $\{x \mid (x,x) \in R\} \cap D = D$. Similarly, to decide if R is symmetric, we construct an automaton $\mathcal{A}_1$ recognizing the relation $\{(y,x) \mid (x,y) \in R\}$ and check if $R = L(\mathcal{A}_1)$. For anti-symmetry, we construct an automaton for $S = \{(x,y) \mid x \neq y\}$ 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 $\{(x,y,z) \mid R(x,y) \wedge R(y,z) \wedge \neg R(x,z)\}$ and ask whether its language is empty. Note that if $D = \Sigma^*$ then $m = 1$.

3. Unary automatic structures

This section explores unary automatic structures and introduces terminology and notation used throughout the 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 $\{1\}^*$. The following lemma from [4] characterizes the regular subsets of $\{1\}^*$.

Lemma 3.1 (*Blumensath; 1999*)**.** *A set* $L \subseteq \mathbb{N}$ *is regular over the alphabet* $\{1\}^*$ *if and only if there are numbers* $t, \ell \in \mathbb{N}$ *such that* $L = L_1 \cup L_2$ *with* $L_1 \subseteq \{x : x < t\}$ *and* L_2 *the finite union* $\bigcup_{j=0,1,\ldots,r-1}\{t + i\ell + k_j : i \in \mathbb{N}\}$ *where* $k_j < \ell$ *for all* j*.*

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. □

Synchronous 2-tape unary automata recognize binary relations over $\mathbb{N}$. The general shape of these automata is given in Fig. 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 $q_0, \ldots, q_t, \ldots, q_m$ where $q_0, \ldots, q_{t-1}$ form the $(1, 1)$-tail and there is a transition from q_m 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.

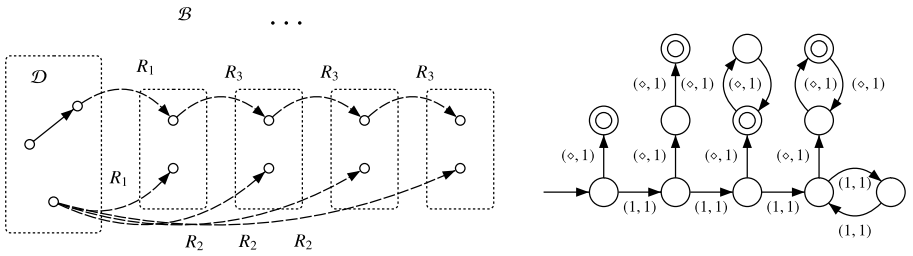

Fig. 2. An example of unwind$(\mathcal{B}, \mathcal{D}, \bar{R})$ and the synchronous 2-tape automaton for its edge relation. If we label $\mathcal{B} = \{a, b\}$ and $\mathcal{D} = \{0, 1, 2\}$ then $E_{\mathcal{D}} = \{(0, 1)\}, E_{\mathcal{B}} = \emptyset, R_1 = \{(1, a), (2, b)\}, R_2 = \{(2, b)\}, R_3 = \{(a, a)\}$, and $R_4 = \emptyset$.

Khoussainov and Rubin [12] and Blumensath [4] generalized Lemma 3.1 and gave a characterization of all binary relations on $\mathbb{N}$ which are recognized by some synchronous 2-tape automaton. In particular, if we view such a relation as the edge relation on the graph of nodes labelled by $\mathbb{N}$, the characterization relates all the unary automatic graphs to an *unwinding* or *ladder* of finite graphs. 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 $D \cup B^0 \cup B^1 \cup B^2 \cup \cdots$ and its edge set contains $E_D \cup E^0 \cup E^1 \cup \cdots$ 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 $(a, b) \in R_4$.

Lemma 3.2 (*Blumensath; 1999. Khoussainov, Rubin; 2001*)**.** *A graph is unary automatic if and only if it is isomorphic to unwind$(\mathcal{B}, \mathcal{D}, \bar{R})$ for some finite graphs $\mathcal{B}, \mathcal{D}$ and relations on these graphs given by $\bar{R}$.*

Proof. Suppose a graph is unary automatic and its edge relation is recognized by a synchronous 2-tape automaton $\mathcal{A}$. Using the terminology from above, we define the vertices of $\mathcal{D}$ to be the states on the $(1, 1)$-tail of $\mathcal{A}$. The edges of $\mathcal{D}$ are determined by (some of) the accepting states on the $(\diamond, 1)$- and $(1, \diamond)$-tails off the $(1, 1)$-tail. Similarly, the vertices of $\mathcal{B}$ are the states on the $(1, 1)$-loop of $\mathcal{A}$. The R_i relations are determined by the appropriate accepting states on the $(1, \diamond)$- and $(\diamond, 1)$-tails off the $(1, 1)$ states of $\mathcal{A}$. Reversing this construction gives a synchronous 2-tape automaton recognizing the edge relation of a graph isomorphic to a given unwinding. In Fig. 2, we provide an example of an automaton and unwinding pair to clarify the construction. □

In this paper we restrict our attention to (countably) infinite structures. The following lemma allows us to assume that the domain of each structure is $\mathbb{N}$ (rather than a regular subset of $\mathbb{N}$) without increasing the size of the associated unary automaton.

Lemma 3.3. *Let $(D; R)$ be a unary automatic structure with $D \subset \mathbb{N}$. Suppose this structure is 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 3.1. 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 3.1). 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 $\mathcal{A}_R$. □

Algorithms on unary automatic binary relations have as input a deterministic synchronous 2-tape unary automaton recognizing the relation. The *size* of the input is defined to be the number of states in this automaton. We say that a synchronous 2-tape automaton is *standard* if the lengths of all its loops and tails equal some number p, called the *loop constant*. If $\mathcal{A}$ is a standard automaton with n states and loop constant p, then $n = 8p^2$.

Lemma 3.4. *For each deterministic 2-tape unary automaton with n states there is an equivalent standard automaton with at most $8n^{2n}$ states.*

Proof. Let p be the least common multiple of the lengths of all loops and tails of $\mathcal{A}$. An easy estimate shows that p is no more than n^n. One can transform $\mathcal{A}$ into an equivalent standard automaton whose loop constant is p. Hence, there is a standard automaton equivalent to $\mathcal{A}$ whose size is bounded above by $8n^{2n}$. □

By Lemma 3.4, we assume all unary automatic structures are presented using standard automata. This assumption in general incurs a super-exponential cost in the state space. However, the standard automata provide natural normal forms for the structures and allow smoother algorithms.

We fix some notation for a standard automaton $\mathcal{A}$ with loop constant p. The $(1, 1)$-states are labelled $q_0, \ldots, q_{2p-1}$ as described above. For $0 \leq j < 2p$, let $W_j = \{x \in \mathbb{N} : \Delta(q_0, (1, 1)^x) = q_j\}$. Then $W_0, \ldots, W_{2p-1}$ partition $\mathbb{N}$ and we have $W_j = \{j\}$ for $0 \leq j < p$, $W_j = \{j + ip : i \in \mathbb{N}\}$ for $p \leq j < 2p$. We enumerate the elements of W_j as $v_i^j = j + ip$.

4. Unary automatic 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). By Table 1, there is an $O(n^3)$ time algorithm that decides whether a given synchronous 2-tape unary automaton presents an equivalence relation. The main theorem of this section is the following.

Theorem 4.1. *The isomorphism problem for unary automatic equivalence structures is decidable in linear time in the sizes of the input standard automata.*

The *height*, $h^0_\mathcal{E}$, of an equivalence structure $\mathcal{E}$ is a function $\mathbb{N} \cup \{\infty\} \to \mathbb{N} \cup \{\infty\}$ such that $h^0_\mathcal{E}(x)$ 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 $h^0_{\mathcal{E}_1} = h^0_{\mathcal{E}_2}$. By the following characterization from [4,12], heights of unary automatic equivalence structures are finitely nonzero and $h^0_\mathcal{E}(\infty) \neq \infty$. If k is the size of the largest finite equivalence class of $\mathcal{E}$, $h^0_\mathcal{E}$ can be encoded by the finite function $h_\mathcal{E}$ with domain $k+2$ such that $h_\mathcal{E}(i) = h^0_\mathcal{E}(i)$ for $i \leq k$ and $h_\mathcal{E}(k+1) = h^0_\mathcal{E}(\infty)$.

Theorem 4.2 (*Blumensath; 1999. Khoussainov, Rubin; 2001*)**.** *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.*

Let $\mathcal{E}$ be recognized by a standard automaton $\mathcal{A} = (S, \iota, \Delta, F)$ with loop constant p (and hence $n = |S| = 8p^2$). Recall the definitions of q_j and W_j from Section 3. Observe that since equivalence relations are symmetric, for any $0 \leq j < 2p$, a state on the $(1, \diamond)$-tail or loop of q_j is accepting if and only if the corresponding state on the $(\diamond, 1)$-tail or loop is also accepting.

Lemma 4.3. *For $0 \leq j < 2p$, each element of W_j belongs to an infinite equivalence class if and only if the $(1, \diamond)$-loop from q_j has an accepting state. Moreover, in this case, W_j forms a subset of some equivalence class.*

Proof. If the $(1, \diamond)$-loop from q_j contains an accepting state, then for each $x \in W_j$, there exists infinitely many y with $(x, y) \in E$. Suppose further that $j \geq p$ (if $j < p$ then $W_j = \{j\}$ so we are done). Then, j is equivalent to $j + p(i+1) + D$ for some $0 \leq D < p$ and all $i > 0$. But, since q_j is on the $(1, 1)$-loop, the accepting state on the $(1, \diamond)$-loop of q_j also gives that $j + p$ is equivalent to $j + p(i+2) + D$ for the same D and all $i > 0$. Transitivity and symmetry then imply that $j, j + p$ are equivalent, hence all members of W_j are equivalent. On the other hand, suppose the $(1, \diamond)$-loop from q_j does not contain any accepting states. Then, for each i, the equivalence class of $j + pi$ must be a subset of $\{0, \ldots, j + p(i+1) - 1\}$, a finite set. □

Lemma 4.4. *For $0 \leq j < 2p$, if W_j does not belong to an infinite equivalence class then it is in an equivalence class of size less than p.*

Proof. Suppose q_j has no accepting state on its $(1, \diamond)$-loop. Define j_0 to be the least number in the (finite) equivalence class containing j. The size of the equivalence class of each $x \in W_j$ is the number of accepting states on the $(1, \diamond)$-tail from q_{j_0}. □

Lemma 4.5. *Given $\mathcal{A}$, Algorithm 1 computes the graph of $h_\mathcal{E}$ in time $O(n)$.*

Proof. By Lemma 4.3, the size of finite equivalence classes is bounded by the size of $(1, \diamond)$-tails, hence $h^0_\mathcal{E}(n) = 0$ for any $n > p$. By Lemma 4.4, for all x, if $h^0_\mathcal{E}(x)$ is finite then $h^0_\mathcal{E}(x) \leq p$. Algorithm 1 exploits the transitivity of the equivalence relation to reduce the number of states of the automaton which must be visited. Moreover, for each j that is considered, the algorithm must check whether at most $4p$ states are accepting. Thus, the runtime of Algorithm 1 is $O(p^2) = O(n)$. □

Algorithm 1 `Equivalence Relation Height`

```
Initialize array h[0 ... p + 1] to 0. Create list L = 0, ..., 2p − 1.
while L ≠ ∅ do
    Let j = least in L.
    if the (1, ◇)-loop from q_j contains no accepting states then
        if j < p then
            Add 1 to h(number of accepting states on (1, ◇)-tail from q_j).
        else
            Set h(number of accepting states on (1, ◇)-tail from q_j) to p + 1.
        end if
        Remove the indices of all these accepting states from L.
    else
        Increment h(p + 1) by 1.
        Remove the indices of all accepting states on (1, ◇)-tail and loop from L.
    end if
end while
```

Proof of Theorem 4.1. Let $\mathcal{A}_1$ (n states) and $\mathcal{A}_2$ (m states) be standard automata recognizing equivalence relations $E_1, E_2 \subseteq \mathbb{N}^2$. By Lemma 4.5, extracting $h_{\mathcal{E}_1}$ and $h_{\mathcal{E}_2}$ takes time $O(\max\{m, n\})$. By Lemma 4.3, $\mathrm{dom}(h_{\mathcal{E}_1}) \cup \mathrm{dom}(h_{\mathcal{E}_2}) \subseteq \max\{m+1, n+1\}$. Therefore, checking if $h_{\mathcal{E}_1} = h_{\mathcal{E}_2}$ takes $O(\max\{m, n\})$. □

5. Unary automatic linear orders

This section studies unary automatic linear orders. A *linear order* is total, reflexive, anti-symmetric, and transitive. By Table 1, checking if a binary relation recognized by an n-state unary automaton is a linear order takes $O(n^3)$. We prove the following theorem.

Theorem 5.1. *The isomorphism problem for unary automatic linear orders is decidable in linear time in the sizes of the input standard automata.*

The following theorem from [4,12] describes which linear orders have unary automatic presentations. We use ω to denote the order type of the natural numbers, $\omega^\star$ to denote the order type of the negative integers, and **1** to denote the singleton linear order.

Theorem 5.2 (*Blumensath; 1999. Khoussainov, Rubin; 2001*)**.** *A linear order $\mathcal{L} = (L; \leq_{\mathcal{L}})$ is unary automatic if and only if it is isomorphic to a finite sum of linear orders of type ω, $\omega^\star$ or* **1**.

By Theorem 5.2, each unary automatic linear order $\mathcal{L}$ can be written as a finite word $u_0 u_1 \dots u_k$ over the alphabet $\{1, \omega, \omega^\star\}$.[2] The *canonical word*, $w_{\mathcal{L}}$, of $\mathcal{L}$ is the minimal such word; 1ω and $\omega^\star 1$ do not appear as substrings of $w_{\mathcal{L}}$. Two unary automatic linear orders $\mathcal{L}_1$ and $\mathcal{L}_2$ are isomorphic if and only if $w_{\mathcal{L}_1} = w_{\mathcal{L}_2}$. Let $\mathcal{L} = (\mathbb{N}; \leq_{\mathcal{L}})$ be a linear order recognized by a standard unary automaton $\mathcal{A}$ with loop constant p. Recall the definitions of q_j and W_j from Section 3.

The following lemmas describe the possible relationships between W_j and W_k for $j < k$. It will be convenient to assign names to *all* states on the $(\diamond, 1)$-tails and loops; these names will be suggestive of the respective relationships. Note that since the linear order is total, whether states on the $(1, \diamond)$-tails and loops are accepting or rejecting is completely determined by the $(\diamond, 1)$-tails and loops.

For $0 \leq j < k < p$,	$q_{j<k} := \Delta(q_j, (\diamond, 1)^{k-j})$.
For $0 \leq j < p$ and $p \leq k < p + j$,	$q^t_{j<k} := \Delta(q_j, (\diamond, 1)^{k-j})$,
	$q^\ell_{j<k} := \Delta(q_j, (\diamond, 1)^{p+k-j})$.
For $0 \leq j < p$ and $p + j \leq k < 2p$,	$q^\ell_{j<k} := \Delta(q_j, (\diamond, 1)^{k-j})$.
For $p \leq j < k < 2p$,	$q^t_{j<k} := \Delta(q_j, (\diamond, 1)^{k-j})$,
	$q^\ell_{j<k} := \Delta(q_j, (\diamond, 1)^{p+k-j})$.
For $p \leq j < k < 2p$,	$q^t_{k<j} := \Delta(q_j, (\diamond, 1)^{p-k+j})$,
	$q^\ell_{k<j} := \Delta(q_j, (\diamond, 1)^{2p-k+j})$.
For $p \leq j < 2p$,	$q^\omega_j := \Delta(q_j, (\diamond, 1)^p)$.

Lemma 5.3. *For $p \leq j < 2p$, W_j either forms an infinite increasing chain or an infinite decreasing chain.*

Proof. If $q^\omega_j \in F$ then for all $i \in \mathbb{N}$, $v^j_i <_{\mathcal{L}} v^j_{i+1}$. Thus, the sequence $v^j_0, v^j_1, v^j_2, \dots$ is an infinite increasing chain. If not, then $q^\omega_j \notin F$ implies that $\Delta(q_j, (1, \diamond)^p) \in F$. Thus, $v^j_0, v^j_1, v^j_2, \dots$ is an infinite decreasing chain. □

Hence, there is an $O(n)$ test checking if a given W_j is an increasing chain or a decreasing chain.

Lemma 5.4. *For $p \leq j < 2p$, W_j is a subset of one copy of ω or one copy of $\omega^\star$ in $\mathcal{L}$.*

Proof. By Lemma 5.3, it is sufficient to prove that any two elements in W_j are separated by at most finitely many elements of $\mathcal{L}$. Consider $v^j_i, v^j_{i'}$ with $i < i'$. Suppose

$$v^j_i \leq_{\mathcal{L}} 2j + s + p(i + i' + r) \leq_{\mathcal{L}} v^j_{i'}$$

for some $r \geq 1$ and $s \geq 0$. By the first inequality, $\Delta(q_j, (\diamond, 1)^{(r+i')p+s+j}) \in F$. By the second inequality, $\Delta(q_j, (1, \diamond)^{(r+i)p+s+j}) \in F$. This contradicts the anti-symmetry of $\mathcal{L}$. Therefore, any z such that $v^j_i \leq_{\mathcal{L}} z \leq_{\mathcal{L}} v^j_{i'}$ must satisfy $z < 2j + (i + i' + 1)p$; there are only finitely many such z. □

[2] This word denotes the linear order $u_0 + u_1 + \cdots + u_k$.

Lemma 5.5. *Let $p \leq j < k < 2p$. If $q^{\ell}_{j<k} \in F \ \wedge \ q^{\ell}_{k<j} \notin F$ then W_j* precedes *W_k:*

$$\forall x \in W_j \, \forall y \in W_k \, (x <_{\mathcal{L}} y) \, .$$

Similarly, if $q^{\ell}_{j<k} \notin F \ \wedge \ q^{\ell}_{k<j} \in F$ then W_k precedes *W_j:*

$$\forall x \in W_j \, \forall y \in W_k \, (y <_{\mathcal{L}} x) \, .$$

Proof. Suppose $q^{\ell}_{j<k} \in F$ and $q^{\ell}_{k<j} \notin F$. We first show it must be the case that $q^{t}_{j<k} \in F$ and $q^{t}_{k<j} \notin F$. Assume for a contradiction that $q^{t}_{j<k} \notin F$. Then (for any i)

$$v^j_{i+2} <_{\mathcal{L}} v^k_i <_{\mathcal{L}} v^j_i <_{\mathcal{L}} v^k_{i+2}$$

but also, $v^k_{i+2} <_{\mathcal{L}} v^j_{i+2}$, contradicting anti-symmetry. Similarly, assume for a contradiction that $q^{t}_{k<j} \in F$. Then (for any i)

$$v^j_{i+3} <_{\mathcal{L}} v^k_i <_{\mathcal{L}} v^j_{i+1} <_{\mathcal{L}} v^k_{i+2}$$

and $v^k_{i+2} <_{\mathcal{L}} v^j_{i+3}$, a contradiction. Thus $v^j_i <_{\mathcal{L}} v^k_{i+r}$ and $v^j_{i+r} <_{\mathcal{L}} v^k_i$ for any i, r. Hence, W_j precedes W_k.

An analogous argument shows that if we assume that $q^{\ell}_{j<k} \notin F \ \wedge \ q^{\ell}_{k<j} \in F$ then $q^{t}_{j>k} \in F$ and $q^{t}_{k<j} \in F$. Thus, in this case, W_k precedes W_j. □

Lemma 5.6. *Let $p \leq j < k < 2p$. If $q^{\ell}_{j<k} \in F \ \wedge \ q^{\ell}_{k<j} \in F$ then W_j and W_k* interleave *within the same copy of ω in $\mathcal{L}$. If $q^{\ell}_{j<k} \notin F \ \wedge \ q^{\ell}_{k<j} \notin F$ then W_j and W_k* interleave *within the same copy of $\omega^\star$ in $\mathcal{L}$.*

Proof. Suppose $q^{\ell}_{j<k} \in F$ and $q^{\ell}_{k<j} \in F$. Then for all i, $v^k_i <_{\mathcal{L}} v^j_{i+2} <_{\mathcal{L}} v^k_{i+3} <_{\mathcal{L}} v^j_{i+5}$. In particular, this implies W_j and W_k are both increasing. Moreover, there are constants $C, d \in \mathbb{Z}$ (depending on which of $q^{t}_{j<k}$ and $q^{t}_{k<j}$ are final) such that $v^j_i <_{\mathcal{L}} v^k_{i+d} <_{\mathcal{L}} v^j_{i+1}$ for all $i \geq C$. Using Lemma 5.4, we conclude that W_j and W_k are in the same copy of ω in $\mathcal{L}$. Symmetrically, if $q^{\ell}_{j>k} \in F$ and $q^{\ell}_{k>j} \in F$ then W_j and W_k are both decreasing and they are in the same copy of $\omega^\star$ in $\mathcal{L}$. □

The proof in Lemma 5.6 can be slightly generalized to see that for $p \leq h < j < k < 2p$, if W_h, W_j interleave and W_j, W_k interleave then W_h, W_k interleave.

Lemma 5.7. *For $0 \leq j < p$ and $p \leq k < 2p$, $\{j\}$ interleaves with W_k if and only if $p \leq k < p + j$ and*

$$q^{t}_{j<k} \in F \iff q^{\ell}_{j<k} \notin F.$$

Proof. It is only possible for $\{j\}$ to interleave with W_k if it is ordered in a different way with respect to v^k_0 than with respect to v^k_i for $i > 0$. If $p + j \leq k < 2p$ then all elements of W_k are represented on the $(\diamond, 1)$-loop off q_j and so the ordering of j with respect to *all* of them is determined by $q^{\ell}_{j<k}$. So, we suppose $p \leq k < p + j$. If $(q^{t}_{j<k} \in F) \iff (q^{\ell}_{j<k} \in F)$, then either $j <_{\mathcal{L}} v^k_i$ for all i or $v^k_i <_{\mathcal{L}} j$ for all i. Thus, in this case, there is no interleave. Finally, consider the case where $q^{t}_{j<k} \in F$ but $q^{\ell}_{j<k} \notin F$. Then, for all $i > 0$,

$$v^k_i <_{\mathcal{L}} j <_{\mathcal{L}} v^k_0.$$

By Lemma 5.3, this implies that W_k is part of an $\omega^\star$ chain, and that j is interleaved in this chain. The symmetric case ($q^{t}_{j<k} \notin F$ but $q^{\ell}_{j<k} \in F$) is analogous. □

Lemma 5.8. *Algorithm 2 extracts $w_{\mathcal{L}}$ from $\mathcal{A}$ in time $O(n)$.*

Proof. Informally, the algorithm works from least to greatest elements in $\mathcal{L}$, checking whether relevant states in $\mathcal{A}$ are accepting or rejecting to build up $w_{\mathcal{L}}$. More precisely, we define and use sets Left(i) to indicate which W_j preceed W_i. We notice that Lemma 5.6 allows us to partition $\{p, \ldots, 2p-1\}$ into sets each of which correspond to a single copy of ω or $\omega^\star$ in $w_{\mathcal{L}}$. Using Lemma 5.7, we add to these sets some elements in $\{0, \ldots, p-1\}$ which fall inside these chains. In Algorithm 2, the resulting sets are labelled V_ℓ. The computation of the sets Left(i), V_ℓ requires visiting each $(\diamond, 1)$-state at most once. Once these sets have been computed the algorithm must check whether at most p many states are in F. Thus, the algorithm runs in $O(n + p^2) = O(n)$. □

Proof of Theorem 5.1. Given two standard automata $\mathcal{A}_1$ (with n states) and $\mathcal{A}_2$ (with m states) recognizing linear orders $\leq_{\mathcal{L}_1}, \leq_{\mathcal{L}_2} \subseteq \mathbb{N}^2$, Lemma 5.8 gives $w_{\mathcal{L}_1}$ and $w_{\mathcal{L}_2}$ in time $O(\max\{m, n\})$. □

Algorithm 2 `Linear order canonical word`

Initialize $B = \emptyset$, $w = \lambda$. Create list $L = 0, \ldots, 2p-1$.
Initialize each Left$(i) = \emptyset$. Initialize an array of sets V_ℓ all to be empty.
for each $(\diamond, 1)$-state given by pair $j < k$ **do**
  if $0 \le j < k < p$ **then**
    If $q_{j<k} \in F$, put $j \in$ Left(k);
    if $q_{j<k} \notin F$, put $k \in$ Left(j).
  else if $0 \le j < p, p \le k < p + j$ **then**
    If $q^t_{j<k} \in F, q^\ell_{j<k} \in F$, put $j \in$ Left(k);
    if $q^t_{j<k} \notin F, q^\ell_{j<k} \notin F$, put $k \in$ Left(j);
    if $q^t_{j<k} \in F, q^\ell_{j<k} \notin F$, remove j from L and ensure $\{j, k\} \subset V_\ell$ for some ℓ;
    if $q^t_{j<k} \notin F, q^\ell_{j<k} \in F$, remove j from L and ensure $\{j, k\} \subset V_\ell$ for some ℓ.
  else if $0 \le j < p, p + j \le k < 2p$ **then**
    If $q^\ell_{j<k} \in F$, put $j \in$ Left(k);
    if $q^\ell_{j<k} \notin F$, put $k \in$ Left(j).
  else if $p \le j < k < 2p$ **then**
    If $q^\ell_{j<k} \in F, q^\ell_{k<j} \notin F$ put $j \in$ Left(k);
    if $q^\ell_{j<k} \notin F, q^\ell_{k<j} \in F$ put $k \in$ Left(j);
    if $q^\ell_{j<k} \in F, q^\ell_{k<j} \in F$ remove k from L and ensure $\{j, k\} \subset V_\ell$ for some ℓ;
    if $q^\ell_{j<k} \notin F, q^\ell_{k<j} \notin F$ remove k from L and ensure $\{j, k\} \subset V_\ell$ for some ℓ.
  end if
end for
while $L \neq \emptyset$ **do**
  Let i be least in L such that Left$(i) = B$
  if $i < p$ **then**
    Put $B = B \cup \{i\}$ and $w = w \cdot 1$. Remove i from L.
  else
    Let j be least such that $i, j \in V_\ell$ for some ℓ.
    if W_j forms an increasing chain **then**
      Put $B = B \cup V_\ell$ and $w = w \cdot \omega$. For $k \in V_\ell$, remove k from L.
    else
      Put $B = B \cup V_\ell$ and $w = w \cdot \omega^\star$. For $k \in V_\ell$, remove k from L.
    end if
  end if
end while
In w, combine any $1 \cdot \omega$ to ω and any $\omega^\star \cdot 1$ to $\omega^\star$.

6. Unary automatic trees

We now turn to unary automatic trees. A structure $\mathcal{T} = (T; \le_\mathcal{T})$ is a *tree* if $\le_\mathcal{T}$ is a partial order on T (reflexive, anti-symmetric, and transitive) with a root (least element) and such that for all nodes $x \in T$, the set $\{y : y \le_\mathcal{T} x\}$ is a finite linear order. Table 1 lists efficient tests for most of the requirements for being a tree. However, checking if $\le_\mathcal{T}$ has a root requires verifying the first-order sentence $\exists x \forall y\, (x \le_\mathcal{T} y)$. The alternation of quantifiers implies an exponential-time decision procedure for general automatic binary relations [17]. This can be improved for unary automatic binary relations.

Lemma 6.1. *If $(\mathbb{N}; R)$ is a partial order where R is recognized by a unary automaton (not necessarily standard) with n states, there is an $O(n)$ time algorithm which checks for an R-least element.*

Proof. Let m be the number of $(1, 1)$-states in $\mathcal{A}$. If there is an R-least element x, then $x < m$. Indeed, if $x \ge m$, there is $y < m$ such that $\Delta(\iota, (1, 1)^x) = \Delta(\iota, (1, 1)^y)$. Let $q = \Delta(\iota, (1, 1)^x)$. Because x is an R-least element we have that $R(x, y)$ and so $\Delta(q, (1, \diamond)^{x-y}) \in F$; similarly, $R(x, 2x - y)$ implies that $\Delta(q, (\diamond, 1)^{x-y}) \in F$. However, this means that $R(y, x)$, a contradiction with anti-symmetry of R.

The R-least element x (if it exists), must satisfy that for all $z < x < y$

$$\Delta(q_x, (\diamond, 1)^{y-x}) \in F \quad \text{and} \quad \Delta(q_z, (1, \diamond)^{x-z}) \in F.$$

Reading each $(\diamond, 1)$ and $(1, \diamond)$ state at most once is sufficient to find such an R-least element or decide that one does not exist. (Note that we are using our assumption from Section 1 that the given unary automaton is complete.) In particular, Algorithm 3 does this and runs in $O(n)$. □

Combining Lemma 6.1 with Table 1 gives the following theorem.

Algorithm 3 *R*-least element

Initialize the list $L = 0, \ldots, m-1$.
while $L \neq \emptyset$ **do**
  Let j be the first element in L.
  if all $(\diamond, 1)$-states out of q_j are accepting **then**
    j is the R-least element; return **true**.
  else
    delete j from L.
    for $k \in L$ **do**
      if $\Delta(q_j, (\diamond, 1)^{k-j}) \in F$ **then** delete k from L.
      if $\Delta(q_j, (1, \diamond)^{k-j}) \notin F$ **then** delete k from L.
    end for
  end if
end while
return **false**

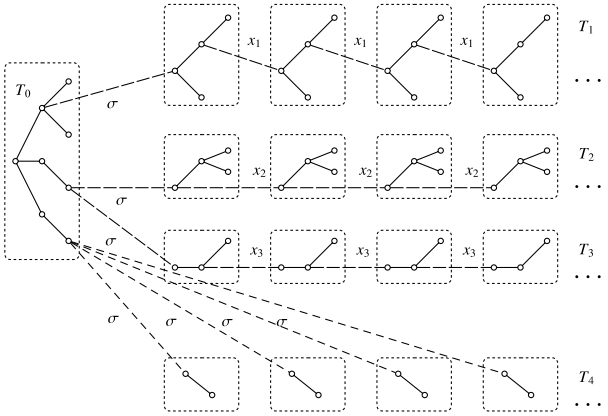

Fig. 3. An example of a tree-unfolding.

Theorem 6.2. *There exists an $O(n^4)$ time algorithm that decides if a unary automatic binary structure is a tree.*

Theorem 6.3 is a characterization of unary automatic trees which will lead to an efficient algorithm for the isomorphism problem. This theorem is similar in spirit to the unwinding description of unary automatic graphs in [12,4] that was discussed as Lemma 3.2. A *parameter set* Γ is a tuple $(T_0, T_1, \ldots, T_m, \sigma, X)$ where $T_0, T_1, \ldots, T_m$ are finite trees, $\sigma : \{1, \ldots, m\} \to T_0$, and $X : \{1, \ldots, m\} \to \{\emptyset\} \cup \bigcup_i T_i$ such that $X(i) \in T_i \cup \{\emptyset\}$. A *tree-unfolding* of a parameter set Γ is a tree $\mathsf{UF}(\Gamma)$ that contains one copy of T_0 and infinitely many copies of T_i for each $i \in \{1, \ldots, m\}$ connected as follows. The root of $\mathsf{UF}(\Gamma)$ is the root of T_0. For $i \in \{1, \ldots, m\}$, if $X(i) \neq \emptyset$, the root of the first copy of T_i is an immediate descendent of $\sigma(i)$ and the root of each subsequent copy of T_i is an immediate descendent of the copy of $X(i)$ in the previous copy of T_i. Otherwise, if $X(i) = \emptyset$, the root of each copy of T_i is an immediate descendent of $\sigma(i)$ (see Fig. 3).

Theorem 6.3. *A tree $\mathcal{T} = (\mathbb{N}, \leq_T)$ is unary automatic if and only if there is a parameter set $\Gamma = (T_0, T_1, \ldots, T_m, \sigma, X)$ such that $\mathcal{T} \cong \mathsf{UF}(\Gamma)$.*

We will need a few definitions and lemmas to prove this theorem. Suppose $\leq_{\mathcal{T}}$ is recognized by a standard automaton $\mathcal{A}$ with loop constant p. Recall from Section 5 the definition of W_j and the labels of states of $\mathcal{A}$. In particular, we will use the notations $q^t_{j<k}$ and $q^\ell_{j<k}$. However, since $\leq_{\mathcal{T}}$ is a partial (rather than linear) order, the $(1, \diamond)$ states are not determined by their $(\diamond, 1)$ counterparts. Hence, we use $q^t_{j>k}$ and $q^\ell_{j>k}$ to denote the appropriate $(1, \diamond)$ states. Two nodes $x, y \in \mathcal{T}$ are *incomparable*, $x|_{\mathcal{T}} y$, if $x \not\leq_{\mathcal{T}} y$ and $y \not\leq_{\mathcal{T}} x$. For $p \leq j < 2p$, W_j is a *chain* if $v^j_0 <_{\mathcal{T}} v^j_1 <_{\mathcal{T}} \ldots$; W_j is an *antichain* if $v^j_i |_{\mathcal{T}} v^j_k$ for all $i \neq k$.

Lemma 6.4. *For $p \leq j < 2p$, W_j is a chain or an antichain. Also, W_j is a chain if and only if for each $x \in W_j$, the set $\{y : x <_{\mathcal{T}} y\}$ is infinite.*

Proof. Let $p \leq j < 2p$. Suppose $q^\omega_j \in F$. Then $v^j_i <_{\mathcal{T}} v^j_{i+1} <_{\mathcal{T}} v^j_{i+2}$ for all i. Hence, W_j is a chain and for any $x \in W_j$, the set $\{y : x <_{\mathcal{T}} y\}$ is infinite.

On the other hand, suppose $q^\omega_j \notin F$. Since $\mathcal{T}$ is a tree, there are no infinite $<_{\mathcal{T}}$-descending chains. Hence, $\Delta(q_j, (1, \diamond)^p) \notin F$. Therefore, for any r, $v^j_i \not<_{\mathcal{T}} v^j_{i+r}$ and $v^j_{i+r} \not<_{\mathcal{T}} v^j_i$ and W_j is an antichain. Assume for a contradiction that there is some i

such that $\{y : v_i^j <_{\mathcal{T}} y\}$ is infinite. In particular, there is some k such that $\{v_s^k : v_i^j <_{\mathcal{T}} v_s^k\}$ is infinite and so $q_{j<k}^{\ell} \in F$. Hence, $v_i^j, v_{i+1}^j <_{\mathcal{T}} v_{i+2}^k$. Since the set of $<_{\mathcal{T}}$ predecessors of v_{i+2}^k is linearly ordered, $v_i^j <_{\mathcal{T}} v_{i+1}^j$ or $v_{i+1}^j <_{\mathcal{T}} v_i^j$, a contradiction. □

Prima facie, there are 2^{10} many possibilities for the interactions between W_j and W_k in the tree order since each interaction is determined by whether each of the following states is accepting or not:

$$q_j^{\omega}, q_k^{\omega}, q_{j<k}^t, q_{j<k}^{\ell}, q_{j>k}^t, q_{j>k}^{\ell}, q_{k<j}^t, q_{k<j}^{\ell}, q_{k>j}^t, q_{k>j}^{\ell}.$$

However, we can use the fact that $\mathcal{A}$ recognizes a tree partial order to eliminate the possibilities dramatically. The following lemma collects the requisite observations; it is proved using properties of trees, such as that the set of predecessors of any tree element is finite and linearly ordered.

Lemma 6.5. *Let $p \le j < k < 2p$.*

1. $q_{j>k}^{\ell} \notin F \ \wedge\ q_{k>j}^{\ell} \notin F$.
2. $\neg\big(q_{j<k}^t \in F \ \wedge\ q_{j>k}^t \in F\big)$. $\quad \neg\big(q_{k<j}^t \in F \ \wedge\ q_{k>j}^t \in F\big)$.
3. $\neg\big(q_{j>k}^t \in F \ \wedge\ q_{k>j}^t \in F\big)$. $\quad \neg\big(q_{j<k}^t \in F \ \wedge\ q_{k<j}^t \in F\big)$.
4. *If $q_j^{\omega} \notin F$, then $q_{j<k}^{\ell} \notin F$.* $\quad$ *If $q_k^{\omega} \notin F$, then $q_{k<j}^{\ell} \notin F$.*
5. *If $q_j^{\omega} \in F$, then* $\neg\big(q_{j<k}^t \notin F \ \wedge\ q_{k>j}^t \in F\big)$. *If $q_k^{\omega} \in F$, then* $\neg\big(q_{j>k}^t \in F \ \wedge\ q_{k<j}^t \notin F\big)$.
6. *If $q_j^{\omega} \in F$, then* $\neg\big(q_{j<k}^t \in F \ \wedge\ q_{j<k}^{\ell} \notin F\big)$. *If $q_k^{\omega} \in F$, then* $\neg\big(q_{k<j}^t \in F \ \wedge\ q_{k<j}^{\ell} \notin F\big)$.
7. *If $q_j^{\omega} \in F$ and $q_k^{\omega} \notin F$, then $q_{k<j}^t \notin F \ \wedge\ q_{j>k}^t \notin F$.*
   If $q_j^{\omega} \notin F$ and $q_k^{\omega} \in F$, then $q_{j<k}^t \notin F \ \wedge\ q_{k>j}^t \notin F$.
8. *If $q_j^{\omega} \in F$ and $q_k^{\omega} \in F$, then* $\neg\big(q_{j<k}^{\ell} \in F \wedge q_{k<j}^{\ell} \notin F\big)$.
9. *If $q_j^{\omega} \in F$ and $q_k^{\omega} \in F$, then*

$$q_{j<k}^t \notin F \wedge q_{j>k}^t \notin F \implies \left[q_{j<k}^{\ell} \notin F \ \wedge\ q_{k<j}^t \notin F \ \wedge\ q_{k<j}^{\ell} \notin F\right].$$

Lemma 6.5 allows us to conclude that if both W_j and W_k are antichains then $q_{j<k}^{\ell} \notin F$ and $q_{k<j}^{\ell} \notin F$ and at most one of $q_{j<k}^t, q_{j>k}^t, q_{k<j}^t, q_{k>j}^t$ is in F. If both W_j and W_k are increasing chains then Lemma 6.5 shows that whether $q_{j<k}^t$, $q_{j>k}^t$ and $q_{k>j}^t$ are accepting or rejecting completely determines the values of the other variables. In the case where W_j is an increasing chain but W_k is an antichain, we see that $q_{j>k}^t$, $q_{k<j}^t$, and $q_{k<j}^{\ell}$ cannot be in F. Moreover, the value of $q_{j<k}^t$ determines either $q_{j<k}^{\ell}$ or $q_{k>j}^t$. The situation in the case where W_j is an antichain and W_k is increasing is similar. Table 2 summarizes the interactions between W_j and W_k in $\mathcal{T}$ based on this information. The first eight columns denote whether key states are accepting (the value 1 denotes membership in F; 0 denotes nonmembership in F). The next column gives a representative diagram of the $<_{\mathcal{T}}$ order of typical elements in W_j and W_k.

Lemma 6.6. *Any unary automatic tree is isomorphic to the tree-unfolding $\mathsf{UF}(\Gamma)$ of some parameter set $\Gamma = (T_0, T_1, \ldots, T_m, \sigma, X)$.*

Proof. Let $\mathcal{T} = (\mathbb{N}; \le_{\mathcal{T}})$ be a tree recognized by a standard unary automaton $\mathcal{A} = (S, I, \Delta, F)$ with loop constant p. The set $\{y : y < p\}$ is a forest under $\le_{\mathcal{T}}$. We define an equivalence relation $\sim$ on $\{y : y \ge p\}$ by $x \sim y$ if and only if there are j, k such that $x \in W_j$, $y \in W_k$ and W_j, W_k are not incomparable (see Table 2). There are finitely many $\sim$-equivalence classes $M_1, \ldots, M_s$. Each M_i is a forest under $\le_{\mathcal{T}}$. If $i \ne i'$ and $x \in M_i$, $y \in M_{i'}$, then $x|_{\mathcal{T}} y$.

The parameter set for $\mathcal{T}$ has finite trees $T_0, T_1, \ldots, T_s$. For $i > 0$, T_i is a subtree of M_i and a distinguished node x_i connects one copy of T_i to the root of the next copy. We extract the pairs (T_i, x_i) from $\mathcal{A}$ as follows. For each $1 \le i \le s$, let $C_i = \{j : W_j \subseteq M_i \ \wedge\ W_j \text{ is a chain}\}$ and $D_i = \{j : W_j \subseteq M_i \ \wedge\ W_j \text{ is an antichain}\}$. The finite tree T_i has $|C_i| + |D_i|$ many nodes, each representing a unique W_j. The union of all nodes in the representative ordering of (W_j, W_k) for $j, k \in C_i$ (from Table 2) forms a linear order under $<_{\mathcal{T}}$. Let $c_1^i <_{\mathcal{T}} \ldots <_{\mathcal{T}} c_{|C_i|}^i$ be the $|C_i|$-many $<_{\mathcal{T}}$-greatest nodes in this finite linear order, and set $x_i = c_{|C_i|}^i$. Note that each c_j^i belongs to a different W_j. For $1 \le j \le |D_i|$, let d_j be the $<_{\mathcal{T}}$-least node in W_j satisfying $c_1^i <_{\mathcal{T}} d_j$. Define T_i to be the finite tree under $<_{\mathcal{T}}$ with domain $\{c_1^i, \ldots, c_{|C_i|}^i\} \cup \{d_1^i, \ldots, d_{|D_i|}^i\}$. Then c_1^i is the root of T_i. Let T_0 be the finite tree formed by nodes in $\{y : y < p\} \cup \bigcup_{1 \le i \le s}\{x \in M_i : x <_{\mathcal{T}} c_1^i \ \vee\ x|_{\mathcal{T}} c_1^i\}$. Note that we must include the possibility that $x|_{\mathcal{T}} c_1^i$: for example, in the seventh line of Table 2, v_0^k will be incomparable to the root of c_i^1 (where $W_k \subseteq M_i$). T_0 may be computed by examining whether $(\diamond, 1)$- and $(1, \diamond)$-states are accepting and by using the case analysis in Table 2. To conclude the definition of Γ, for $1 \le i < s$, set $\sigma(i) = x$ such that $x \in T_0$ and $x <_{\mathcal{T}} c_1^r$ and $\forall y \in T_0\ (y <_{\mathcal{T}} c_1^i \to y <_{\mathcal{T}} x)$. □

Table 2
Relationship between W_j and W_k in tree $\mathcal{T}$, based on F in $\mathcal{A}$.

q_j^ω	q_k^ω	$q_{j<k}^t$	$q_{j<k}^\ell$	$q_{j>k}^t$	$q_{k<j}^t$	$q_{k<j}^\ell$	$q_{k>j}^t$	Ordering
1	1	1	1	0	0	1	1	$v_0^j \to v_1^j \to v_0^k \to v_2^j \cdots v_i^k \to v_{i+2}^j \to v_{i+1}^k$
1	1	1	1	0	1	1	0	$v_0^j \to v_0^k \to v_1^j \to v_1^k \cdots v_i^j \to v_i^k \to v_{i+1}^j$
1	1	0	1	1	1	1	0	$v_0^k \to v_0^j \to v_1^k \to v_1^j \cdots v_i^k \to v_i^j \to v_{i+1}^k$
1	1	0	0	0	0	0	0	Incomparable
1	0	1	1	0	0	0	1	$v_0^j \to v_1^j \to v_2^j \cdots v_i^j \to v_{i+1}^j$, $v_0^j \to v_1^j \to v_2^j \cdots v_i^j \to v_{i+1}^j \nearrow v_i^k$, $v_0^j \to v_1^j \to v_2^j \nearrow v_1^k$, $v_0^j \to v_1^j \nearrow v_0^k$
1	0	1	1	0	0	0	0	$v_0^j \to v_1^j \to v_2^j \cdots v_i^j \to v_{i+1}^j$, $v_0^j \to v_1^j \to v_2^j \cdots v_i^j \nearrow v_i^k$, $v_0^j \to v_1^j \to v_2^j \nearrow v_1^k$, $v_0^j \to v_1^j \nearrow v_0^k$
1	0	0	1	0	0	0	0	$v_0^j \to v_1^j \to v_2^j \cdots v_i^j \to v_{i+1}^j$, $v_0^j \to v_1^j \to v_2^j \cdots v_i^j \nearrow v_{i+1}^k$, $v_0^j \to v_1^j \to v_2^j \nearrow v_2^k$, $v_0^j \to v_1^j \nearrow v_1^k$, $v_0^k \to v_1^k \nearrow v_0^j$
1	0	0	0	0	0	0	0	Incomparable
0	1	0	0	0	1	1	0	$v_0^k \to v_1^k \to v_2^k \cdots v_i^k \to v_{i+1}^k$, $v_0^k \to v_1^k \to v_2^k \cdots v_i^k \nearrow v_{i+1}^j$, $v_0^k \to v_1^k \to v_2^k \nearrow v_2^j$, $v_0^k \to v_1^k \nearrow v_1^j$, $v_0^j \to v_1^j \nearrow v_0^k$
0	1	0	0	0	0	1	0	$v_0^k \to v_1^k \to v_2^k \cdots v_i^k \to v_{i+1}^k$, $v_0^k \to v_1^k \to v_2^k \cdots v_i^k \nearrow v_{i+2}^j$, $v_0^k \to v_1^k \to v_2^k \nearrow v_3^j$, $v_0^k \to v_1^k \nearrow v_2^j$, $v_0^j \to v_1^j \nearrow v_0^k$, v_1^j
0	1	0	0	0	0	0	0	Incomparable
0	0	1	0	0	0	0	0	$v_0^j \to v_0^k$, $v_i^j \to v_i^k$
0	0	0	0	1	0	0	0	$v_0^k \to v_0^j$, $v_i^k \to v_i^j$
0	0	0	0	0	1	0	0	v_0^j, $v_0^k \to v_1^j$, $v_i^k \to v_{i+1}^j$
0	0	0	0	0	0	0	1	v_0^j, $v_1^j \to v_0^k$, $v_{i+1}^j \to v_i^k$
0	0	0	0	0	0	0	0	Incomparable

Lemma 6.7. *If $\Gamma = (T_0, T_1, \ldots, T_m, \sigma, X)$ is a parameter set, $\mathsf{UF}(\Gamma)$ is a unary automatic tree $\mathcal{T}$.*

Proof. Let $t = |T_0|$, $\ell = \sum_{r=1}^{m} |T_r|$, and $\alpha_r = \sum_{i=1}^{r-1} |T_i|$ for $r = 1, \ldots, m$. Given Γ, we consider the isomorphic copy $(\mathbb{N}; \leq_{\mathcal{T}}) \cong \mathsf{UF}(\Gamma)$ where $T_0 \mapsto \{0, \ldots, t-1\}$, and the jth 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 for $\leq_{\mathcal{T}}$ will have a $(1, 1)$-tail of length t and a $(1, 1)$-loop of length ℓ. The states on the $(1, 1)$-tail are $\{q_0, \ldots, q_{t-1}\}$ and the states on the $(1, 1)$-loop are $\{q_t, \ldots, q_{t+\ell-1}\}$; $\iota = q_0$. Each q_j on the $(1, 1)$-tail (so $0 \leq j < t$) 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 (so $t \leq j < t + \ell$) has a $(\diamond, 1)$-tail and $(\diamond, 1)$-loop, each of length ℓ. All $(1, 1)$-states are accepting. Let the bijection $\varphi_0 : T_0 \to \{0, \ldots, t-1\}$ satisfy $\varphi_0(x) < \varphi_0(y)$ whenever $x <_{T_0} y$. For each $j < k < t$, we make

- $\Delta(q_j, (\diamond, 1)^{k-j})$ accepting if $\varphi_0^{-1}(j) <_{T_0} \varphi_0^{-1}(k)$, and
- $\Delta(q_j, (1, \diamond)^{k-j})$ accepting if $\varphi_0^{-1}(k) <_{T_0} \varphi_0^{-1}(j)$.

Let the bijection $\varphi_r : T_r \to \{t + \alpha_r, \ldots, t + \alpha_{r+1} - 1\}$ satisfy $\varphi_r(x) < \varphi_r(y)$ whenever $x <_{T_r} y$. An analogous (but slightly more complicated) construction uses $\varphi_1, \ldots, \varphi_m$ 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 $(\mathbb{N}; L(\mathcal{A})) \cong \mathsf{UF}(\Gamma)$. □

Proof of Theorem 6.3. Lemmas 6.6 and 6.7 give the characterization. □

Corollary 6.8. *If $\mathcal{T}$ is recognized by a standard automaton with n states, an $O(n)$ algorithm gives a parameter set Γ where $\mathcal{T} = \mathsf{UF}(\Gamma)$.*

Proof. The construction outlined in the proof of Lemma 6.6 uses finitely many table lookups in Table 2 and a single traversal of the transition diagram of the automaton recognizing $\mathcal{T}$. □

Corollary 6.9. *If Γ is a parameter set with $t = |T_0|$ and $\ell = \sum_{i=1}^{m} |T_i|$ then there is a standard unary automaton $\mathcal{A}$ with $O(t^2\ell^2)$ states such that $\mathsf{UF}(\Gamma) \cong (\mathbb{N}; L(\mathcal{A}))$.*

We now show that the isomorphism problem for unary automatic trees is decidable. Observe that two tree-unfoldings may be isomorphic even if the associated parameter sets are not isomorphic term-by-term. Ideally, we are looking for an isomorphism invariant which does not have this flaw. To obtain one, we begin by fixing a computable linear order $\prec$ on the set of finite trees. We assume that the finite trees can be efficiently encoded as natural numbers such that asking if one is $\prec$-below another takes constant time. We define the *canonical representation* of a unary automatic tree $\mathcal{T} = (\mathbb{N}; \leq_{\mathcal{T}})$ to be the minimal parameter set $\Gamma = (T_0, T_1, \ldots, T_m, \sigma, X)$ with $\mathsf{UF}(\Gamma) \cong \mathcal{T}$, where minimality is defined as follows.

- As finite trees, $T_1 \preceq \cdots \preceq T_m$.
- Each T_i $(1 \leq i \leq m)$ is minimal in that, for all y_1, y_2, if $y_1 <_{\mathcal{T}} y_2 <_{\mathcal{T}} x_i$ then the subtree with domain $\{z : y_1 \leq_{\mathcal{T}} z, y_2 \not\leq_{\mathcal{T}} z\}$ is not isomorphic to the subtree with domain $\{z : y_2 \leq_{\mathcal{T}} z, x_i \not\leq_{\mathcal{T}} z\}$. Also, if t_i is the root of the first copy of T_i $(1 \leq i \leq m)$ then there is no $y \in T_0$ such that $y <_{\mathcal{T}} \sigma(i)$ and the subtree with domain $\{z : y \leq_{\mathcal{T}} z, t_i \not\leq_{\mathcal{T}} z\}$ is isomorphic to T_i.
- The canonical representation is then the parameter set which satisfies the above conditions and in which T_0 has the fewest nodes.

Lemma 6.10. *Suppose $\mathcal{T}, \mathcal{T}'$ are unary automatic trees with canonical representations Γ, Γ'. Then, $\mathcal{T} \cong \mathcal{T}'$ if and only if Γ, Γ' have the same number (m) of finite trees, $(T_0, \sigma) \cong (T_0', \sigma')$, and for $1 \leq i \leq m$, $(T_i, x_i) \cong (T_i', x_i')$.*

Proof. It is easy to see that if $\mathcal{T}$ and $\mathcal{T}'$ have term-by-term isomorphic canonical representation they are isomorphic. Conversely, suppose $\mathcal{T} \cong \mathcal{T}'$ and have canonical representations $(T_0, \ldots, T_m, \sigma, X)$ and $(T_0', \ldots, T_m', \sigma', X')$, respectively. Each infinite subtree of the form $(\{y : \sigma(i) \leq y\}; \leq_{\mathcal{T}})$, $1 \leq i \leq m$, which contains infinitely many copies of T_i, embeds into a subtree of $\mathcal{T}'$. By the minimality condition on T_i, T_i' and by the ordering of the finite trees in each parameter set, the subtree of $\mathcal{T}$ containing infinitely many copies of T_i can embed into the subtree of $\mathcal{T}'$ containing infinitely many copies of T_i' for all $1 \leq i \leq m$ and vice versa. By minimality of $T_0, T_0', \forall 1 \leq i \leq m$ $(T_i, x_i) \cong (T_i', x_i')$. Let t_i be the root of the first copy of T_i in $\mathcal{T}$, let t_i' be the root of the first copy of T_i' in $\mathcal{T}'$.

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

Suppose we can compute the canonical representation of a tree from a unary automaton. Given two unary automatic trees, we could use Lemma 6.10 and a decision procedure for isomorphism of finite trees to solve the isomorphism problem on unary automatic trees.

Lemma 6.11. *Given a tree-unfolding $\mathsf{UF}(\Gamma)$ with n the sum of the sizes of all finite trees in Γ, there is an $O(n^2)$ algorithm that computes the canonical representation of $\mathsf{UF}(\Gamma)$.*

Proof. Suppose $\Gamma = (T_0, T_1, \ldots, T_m, \sigma, X)$. For each $1 \leq i \leq m$, look for $y_1, y_2 \in T_i$ such that $y_1 <_{\mathcal{T}} y_2 <_{\mathcal{T}} x_i$, and the subtree of T_i with domain $\{z : y_1 \leq_{\mathcal{T}} z, y_2 \not\leq_{\mathcal{T}} z\}$ is isomorphic to the subtree with domain $\{z : y_2 \leq_{\mathcal{T}} z, x_i \not\leq_{\mathcal{T}} z\}$. If such y_1, y_2 exist, remove all $z \geq_{T_i} y_1$ from T_i. Thus, each T_i satisfies the minimality condition for the canonical representation. Since the isomorphism problem for finite trees is decidable in linear time [18], this step can be done in time $O(m|T_i|^2)$.

For each $1 \leq i \leq m$, let t_i be the root of the first copy of T_i. Look for $x \in T_0$ such that $x <_{\mathcal{T}} \sigma(i)$, and the subtree of T_0 with domain $\{y : x \leq_{\mathcal{T}} y, t_i \not\leq_{\mathcal{T}} y\}$ 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. This step can be done in time $O(m|T_i|^2)$.

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 $O(m|T_i|^2)$.

The last step in transforming our parameter set to the canonical presentation is to order the finite trees in increasing $\prec$-order. By assumption on the complexity of $\prec$, applying a sorting algorithm on m finite trees takes $O(m \log m)$. Since $n = |T_1| + \cdots + |T_m|$, the algorithm takes time $O(n^2)$ (Table 3). □

Table 3
Summary of results on unary automatic structures.

Problems	Equivalence structures	Linear orders	Trees
Membership problem	$O(n^3)$	$O(n^3)$	$O(n^4)$
Isomorphism problem	$O(\max\{n_1, n_2\})$	$O(\max\{n_1, n_2\})$	$O(\max\{n_1^2, n_2^2\})$

Theorem 6.12. *If $\mathcal{T}_1, \mathcal{T}_2$ are unary automatic trees presented by standard automata $\mathcal{A}_1$ (n_1 states) and $\mathcal{A}_2$ (n_2 states), an $O(\max\{n_1^2, n_2^2\})$-time algorithm decides if $\mathcal{T}_1 \cong \mathcal{T}_2$.*

Proof. By Corollary 6.8 and Lemma 6.11, we can convert the standard automata presenting $\mathcal{T}_1$ and $\mathcal{T}_2$ to canonical representations of the trees. Then, the isomorphism problem reduces to checking finitely many isomorphisms of finite trees. The sum of sizes of finite trees in each parameter set is bounded by n_i. Hence, it takes $O(n_i^2)$ to compute each canonical representation and then check if they are equal. □

7. Conclusion and future work

We described algorithms deciding the isomorphism problems for unary automatic equivalence structures, linear orders, and trees. This settled the question of whether such algorithms existed. Moreover, we considered a normal form for the automata involved, with respect to which the time complexity of the algorithms was polynomial. The membership problem for each of these classes was also shown to take polynomial-time with respect to any input unary automaton. It is still open whether the isomorphism problem for unary automatic graphs is decidable, and if so, what complexity class it lies in.

References

[1] A. Aho, J. Hopcroft, D. Jeffrey, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.
[2] H. Rogers Jr., Theory of Recursive Functions and Effective Computability, McGraw-Hill Book Company, 1967.
[3] B. Khoussainov, A. Nerode, Automatic presentations of structures, in: D. Leivant (Ed.), International Workshop on Logic and Computational Complexity, in: LNCS, vol. 960, Springer-Verlag, 1995, pp. 367–392.
[4] A. Blumensath, Automatic structures, Diploma Thesis, RWTH Aachen, October 1999.
[5] S. Rubin, Automatic structures, Ph.D. Thesis, University of Auckland, 2004.
[6] M. Minnes, Computability and complexity properties of automatic structures and their applications, Ph.D. Thesis, Cornell University, 2008.
[7] M. Vardi, Model checking for database theoreticians, in: Proc. 10th International Conference on Database Theory, 2005.
[8] B. Khoussainov, M. Minnes, Model theoretic complexity of automatic structures (extended abstract), in: M. Agrawal, et al. (Eds.), Proc. 5th TAMC, in: LNCS, vol. 4978, Springer-Verlag, 2008, pp. 520–531.
[9] B. Khoussainov, A. Nies, S. Rubin, F. Stephan, Automatic structures: richness and limitations, in: Proc. 19th LICS, IEEE Computer Society, 2004, pp. 44–53.
[10] B. Courcelle, The monadic second-order logic of graphs IV: definability properties of equational graphs, Annals of Pure and Applied Logic 49 (1990) 193–255.
[11] B. Khoussainov, J. Liu, M. Minnes, Unary automatic graphs: an algorithmic perspective, in: M. Agrawal, et al. (Eds.), Proc. 5th TAMC, in: LNCS, vol. 4978, Springer-Verlag, 2008, pp. 548–559.
[12] B. Khoussainov, S. Rubin, Graphs with automatic presentations over a unary alphabet, Journal of Automata, Languages and Combinatorics 6 (4) (2001) 467–480.
[13] S. Rubin, Automata presenting structures: a survey of the finite string case, Bulletin of Symbolic Logic 14 (2) (2008) 169–209.
[14] B. Khoussainov, M. Minnes, Three lectures on automatic structures, in: Proceedings of Logic Colloquium 2007, Cambridge University Press, 2008.
[15] B. Hodgson, On direct products of automaton decidable theories, Theoretical Computer Science 19 (1982) 331–335.
[16] A. Blumensath, E. Grädel, Automatic structures, in: Proc. 15th LICS, IEEE Computer Society, 2000, pp. 51–62.
[17] B. Khoussainov, A. Nerode, Automata Theory and its Applications, Birkhauser, Boston, Massachusetts, 2001.
[18] J. Hopcroft, J. Wong, Linear time algorithm form isomorphism of planar graphs (preliminary report), in: Proc. 6th STOC, 1974, pp. 172–184.