Word Structures and Their Automatic Presentations

Xiaoyang Gong
School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

Bakh Khoussainov
School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

Yuyang Zhuge
School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

Abstract

We study automatic presentations of the structures $(\mathbb{N}; S)$, $(\mathbb{N}; E_S)$, $(\mathbb{N}; \leq)$, and their expansions by a unary predicate U. Here S is the successor function, E_S is the undirected version of S, and $\leq$ is the natural order. We call these structures word structures. Our goal is three-fold. First, we study the isomorphism problem for automatic word structures by focusing on the following three problems. The first problem asks to design an algorithm that, given an automatic structure $\mathcal{A}$, decides if $\mathcal{A}$ is isomorphic to $(\mathbb{N}; S)$. The second asks to design an algorithm that, given two automatic presentations of $(\mathbb{N}; S, U_1)$ and $(\mathbb{N}; S, U_2)$, where U_1 and U_2 are unary predicates, decides if these structures are isomorphic. The third problem investigates if there is an algorithm that, given two automatic presentations of $(\mathbb{N}; \leq, U_1)$ and $(\mathbb{N}; \leq, U_2)$, decides whether $U_1 \cap U_2 \neq \emptyset$. We show that these problems are undecidable. Next, we study intrinsic regularity of the function S in the structure $Path_\omega = (\mathbb{N}; E_S)$. We build an automatic presentation of $Path_\omega$ in which S is not regular. This implies that S is not intrinsically regular in $Path_\omega$. For $U \subseteq \mathbb{N}$, let d_U be the function that computes the distances between the consecutive elements of U. We build automatic presentations of $(\mathbb{N}; \leq, U)$ where d_U can realise logarithmic, radical, intermediate, and exponential functions.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Logic

Keywords and phrases Automatic structures, the isomorphism problem, decidability, undecidability, regular relations

Digital Object Identifier 10.4230/LIPIcs.MFCS.2025.51

Funding The authors acknowledge the support of the NSFC grant 62172077, 62350710215 and Sichuan S&T Department grant 2025HJPJ0006.

1 Introduction

We investigate automatic presentations of the structures $(\mathbb{N}; \leq)$, $(\mathbb{N}; S)$, $(\mathbb{N}; E_S)$, $(\mathbb{N}; \leq, U)$, $(\mathbb{N}; S, U)$ and $(\mathbb{N}; E_S, U)$, where $\mathbb{N} = \{0, 1, 2, \ldots\}$, $\leq$ is the natural order, the function S is the successor function, E_S is the undirected version of S (that is $\{k, j\} \in E_S$ iff $k + 1 = j$), and U is a unary predicate on $\mathbb{N}$. The first structure is the ordinal ω, the second is the successor structure, and the third structure is the infinite path graph $Path_\omega$. For the last three structures, without loss of generality, we postulate that $0 \in U$ and U is infinite. Each of the last three structures with the predicate U determines the infinite binary word $\alpha_U = \alpha_U(0)\alpha_U(1)\ldots\alpha_U(i)\ldots$, where $\alpha_U(i) = 1$ if $i \in U$ and $\alpha_U(i) = 0$ otherwise. These binary words are full isomorphism invariants of $(\mathbb{N}; \leq, U)$, $(\mathbb{N}; S, U)$, and $(\mathbb{N}; E_S, U)$, i.e., $\alpha_{U_1} = \alpha_{U_2}$ if and only if $(\mathbb{N}; S, U_1) \cong (\mathbb{N}; S, U_2)$. Thus, here is a definition:

▶ **Definition 1.** *Call* $(\mathbb{N};\leq,U)$, $(\mathbb{N};S,U)$, $(\mathbb{N};E_S,U)$ ***ordered words***, ***successor words***, *and* ***graph words***, *respectively. We also call any of these structures* ***word structures***.

▶ **Definition 2.** *An* ***automatic presentation*** *of the ordered word structure* $(\mathbb{N};\leq,U)$ *is a triple* $(D;\leq_D,U_D)$, *where:*

- D *and* $U_D \subseteq D$ *are both FA recognizable languages,*
- $\leq_D$ *is a binary relation on* D *recognized by two heads synchronous finite automata, and*
- *The structure* $(D;\leq_D,U_D)$ *is isomorphic to* $(\mathbb{N};\leq,U)$.

Automatic presentations of $(\mathbb{N};S,U)$ and $(\mathbb{N};E_S,U)$ are defined similarly.

1.1 Automatic structures

A relational structure $(D;R_0^{n_1},\ldots,R_k^{n_k})$ is **automatic** if the domain D and the atomic relations $R_i^{n_i}$ (of arity n_i) are all finite automata recognizable. We assume the reader is familiar with the basics of finite automata, e.g., runs, acceptance, and recognizability[33]. For the reader, nevertheless, we explain finite automata recognizability of binary relations R on the set of strings Σ^* over the alphabet Σ. So, let $x = x_1 \ldots x_n$ and $y = y_1 \ldots y_m$ be two strings over Σ. Then we code the pair (x,y) as the string $z = x \oplus y$ so that $z = z_1 \ldots z_{\max\{m,n\}}$ is defined as follows: $z_i = (x_i, y_i)$ if $i \leq \min\{n,m\}$, $z_i = (\diamond, y_i)$ if $n < i \leq m$, and $z_i = (x_i, \diamond)$ if $m < i \leq n$, where $\diamond \notin \Sigma$ is a new symbol. The operation $\oplus$ is called the *convolution operation on strings*. The string $z = x \oplus y$ is a string over the finite alphabet $\Sigma' = \{(\sigma,\diamond) \mid \sigma \in \Sigma\} \cup \{(\diamond,\sigma) \mid \sigma \in \Sigma\} \cup \Sigma \times \Sigma$.

Let $\oplus(R)$ be the language over Σ' obtained by applying the convolution operation a to pair of strings in R. Call R **finite automata recognizable** if $\oplus(R)$ is recognized by a finite automaton. One can view finite automata over Σ' as two-head synchronous automata over Σ. The reader can consult standard papers on automatic structures for details and examples, e.g., [1, 28, 7, 32, 49, 31], although Definition 2 suffices for this paper. Finally, in this paper, we use the standard terminology and notations from the theory of finite automata.

The connection between automata, logic and structures traces back to the 1960s when Büchi proved the decidability of $S1S$ -the monadic second-order theory of $(\mathbb{N};S)$ -using automata [8]. M. Rabin extended this result, through the use of tree automata, by proving that the monadic second-order theory $S2S$ of the full infinite binary tree is decidable [47]. In computational group theory context, M. Thurston et al.[12] realized that geodesic words within Cannon's group of hyperbolic isometries are finite automata recognizable and defined the concept of automatic groups, which was extended to Cayley automatic groups in [27] while retaining its tractable essences. These and related papers constitute a foundation of modern research on automata, logic, their interactions, and applications. Also, automatic structures have attracted much attention in the last few decades, see surveys by Grädel [16], Bárány, Rubin, and Grädel [2], F. Stephan [52], and Rubin [50, 36]. Several key factors motivate this study. One is that automatic structures exhibit nice algorithmic and model-theoretic properties. The theory of every automatic structure is decidable. Automatic structures are closed under interpretations in first-order logic [7]. The other aspects involve connections to algebra (e.g., Gromov's theorem that finitely generated groups of polynomial growth are virtually nilpotent [17]) and additive combinatorics (e.g., Freiman's theorem that sets with small sums are contained in generalized arithmetic progressions [13]). These connections were revealed in the characterization theorem of finitely generated finite automata presentable groups [46] [45] and the proof that the group $(\mathbb{Q};+)$ has no automatic presentation [53].

The presentations use finite automata. There are also other presentations of structures: tree automatic presentations [22, 2, 21], computably enumerable presentations [29, 9, 15], co-computably enumerable presentations [39], semi-automatic presentations [24, 23], ω-automatic presentations [19, 26] and unary automatic presentations [30, 35], etc.

1.2 Background and contributions

We describe three, among many, research directions directly related to this work.

1. One direction focuses on first-order (FO) theories and interpretations of automatic structures in FO-logic and its extensions. For instance, Hodgson [20], Khoussainov and Nerode [28], Blumensath and Grädel [7] proved that the first-order theory of any automatic structure is decidable. This is known as the **decidability theorem** for automatic structures.

This result was extended to FO-logic with "there exists n many modulo m" quantifiers $\exists^{n,m}$ [38] and "there exist infinitely many" quantifier $\exists^{\omega}$ [7]. S. Rubin [50] studied extensions with Ramsey's quantifiers. Blumensath and Grädel characterized automatic structures in terms of interpretations [6]. They proved the **interpretability theorem** that a structure is automatic iff it has an interpretation in the infinite binary tree with two successor relations, the prefix order and the equal length predicates [6]. Finding extensions of the FO-logic that preserve the decidability and interpretability theorems is one of the key topics in the area.

2. The second direction focuses on the classical isomorphism problem. The problem asks to design an algorithm that, given two automatic structures, decides if the structures are isomorphic. The isomorphism problem pertains to describing the isomorphism invariants and extracting structure theorems. For instance, the Cantor-Bendixson ranks of automatic linear orders turned out to be a useful invariant. The study of Cantor-Bendixson ranks implied that an ordinal is automatic iff it is less than ω^{ω} [10]. Using this, Khoussainov, Stephan, and Rubin proved that the isomorphism problem for automatic ordinals is decidable. Thomas and Oliver fully described finitely generated automatic groups [46]. Khoussainov, Nies, Rubin and Stephan characterized and solved the isomorphism problem for automatic Boolean algebras [34]. Grädel and Blumensath were the first to prove that the isomorphism problem for automatic structures is undecidable [7]. Khoussainov, Nies, Rubin, and Stephan proved that the isomorphism problem for automatic structures is Σ^1_1-complete[34]. D. Kuske, M. Lohrey, and J. Liu proved that the isomorphism problems for automatic linearly ordered sets and automatic trees are undecidable, and Π^0_1-complete for automatic equivalence relations [40]. Moreover, B. Khoussainov and M. Ganardi show that the isomorphism problem for automatic equivalence relations on regular domains with polynomial growth is still undecidable[14].

3. The third direction concerns the intrinsic regularity of relations. A relation R in a structure $\mathcal{A}$ is **intrinsically regular** if from any automatic presentation $\mathcal{B}$ of $\mathcal{A}$ we can extract a finite automaton that recognizes the relation R in $\mathcal{B}$ [38]. For instance, all first-order definable relations (with and without parameters) are intrinsically regular. For any first-order logic formula $\phi(x, \bar{y})$, automatic structure $\mathcal{A}$, the set of all $\bar{p} \in A$ such that $|\{a \mid \mathcal{A} \models \phi(a, \bar{p})\}| = k \mod n$ is intrinsically regular [38]. In an automatic partially ordered set of bounded width, the set of all x that belong to an infinite path is also intrinsically regular [37].

Let R be an intrinsically regular relation in $\mathcal{A}$. Given a formula $\phi(\bar{x}, \bar{y})$ we can consider the expression $\mathcal{R}\bar{x}\phi(\bar{x}, \bar{y})$ (binding $\bar{x}$) with the following semantics: $\{\bar{b} \mid \{\bar{a} \mid \mathcal{A} \models \phi(\bar{a}, \bar{b})\} = R\}$. So, intrinsically regular relations R allow us to define $FO + \mathcal{R}$ logic that extends the first order logic with the generalised quantifier $\mathcal{R}$. Such extensions preserve the decidability and interpretability theorems for automatic structures. Thus, intrinsic regularity is intimately connected with the questions of extending the decidability and interpretability theorems.

To refute that R is intrinsically regular in $\mathcal{A}$, we need to build an automatic presentation $\mathcal{B}$ of $\mathcal{A}$ in which R is not regular. For instance, consider the k-ary representation of integers, $k > 1$. These presentations make the structure $(\mathbb{Z}; S)$ automatic. In these presentations addition $+$ relation, the relations $\leq$ and $mod\ (n)$ are regular. Clearly, the relation $+$ is not intrinsically regular in $(\mathbb{Z}; S)$ because in all unary automatic presentations of $(\mathbb{Z}; S)$

the addition is not regular. However, proving that $\leq$ and *mod* (n) are not intrinsically regular requires non-trivial arguments. Indeed, there are automatic presentations of $(\mathbb{Z}; S)$ in which the *mod* (n) relations are not regular [38]. Given that the *mod* (n) relations are intrinsically regular in $(\mathbb{Z}; \leq)$ [38], we conclude that the relation $\leq$ in $(\mathbb{Z}; S)$ is also not intrinsically regular. Moreover, there are presentations of $(\mathbb{Z}; S)$ in which $\leq$ is not regular but all *mod* (n) are regular [38]. These presentations can be argued to be "exotic" as they show that regularity of the successor S implies neither regularity of $\leq$ nor regularity of *mod* (n); they also show that regularity of S and *mod* (n) does not imply regularity of $\leq$. More examples of non-intrinsically regular relations are in [38] [43] [49]. These examples bring insights into the understanding of regular relations, their interactions within automatic presentations, and the exploration of extensions to decidability and interpretability theorems.

There are certainly other exciting directions of research in automatic structures. We mention the complexity-theoretic issues. For instance, it is important to find exact complexity-theoretic bounds on the first order theories of automatic structures. D. Kuske and M. Lohrey proved that the first order theory of automatic graphs of bounded degree is triple exponential [41]. Blumensath and Gradel proved that there are automatic structures whose first order theories are non-elementary [6]. Also, D. Berdinsky has investigated the existence of automatic presentations for various types of vertex transitive graphs [11] [4] .

Our contributions for this work are listed below.

1. The isomorphism problem for $(\mathbb{N}; S)$. Our first isomorphism problem asks if we can decide that given an automatic structure $\mathcal{A}$ is isomorphic to $(\mathbb{N}; S)$. Although not explicitly stated in the literature, the question is natural and has been known since 1996 [44]. Note that there is an algorithm that, given an automatic structure $(D; f)$ decides if f satisfies the following two axioms: (a) f is injective, and (b) there is exactly one element $y \in D$ without f-pre-image. the structure $(\mathbb{N}; S)$ satisfies these two axioms. However, even if $(D; f)$ satisfies (a) and (b), this does not imply that $(D; f)$ is isomorphic to $(\mathbb{N}; S)$. We prove that the problem of deciding if an automatic structure is isomorphic to $(\mathbb{N}; S)$ is undecidable. Our proof shows that this isomorphism problem is Π^0_2-complete. Our solution is based on coding the configuration graphs of total Turing machines (TMs) into automatic presentations of $(\mathbb{N}; S)$. Recall that a TM, whose inputs are strings over a given alphabet Σ, is called **total** if for all $u \in \Sigma^\star$ the machine M halts on u by either accepting or rejecting u. The set of all total Turing machines is a Π^0_2-complete set [25]. Our solution is based on a careful analysis and coding of the configuration graphs of reversible TMs. Lemma 6 provides the coding. Theorem 7 transforms the configuration graphs of total TMs into the structure $(\mathbb{N}; S)$. Both, the lemma and the theorem, involve non-trivial reasoning.

Now we explain why the isomorphism problem for $(\mathbb{N}; S)$ is of particular importance. Let us compare the structure $(\mathbb{N}; S)$ with automatic ordinal ω which is similar to $(\mathbb{N}; S)$. First, it is decidable if a given automatic structure $\mathcal{A}$ is isomorphic to ω. Hence, the isomorphism problem for ω is decidable. Note that every automatic presentation of ω produces an automatic presentation of $(\mathbb{N}; S)$ while the opposite is not true. Hence, it is natural to ask if one can extend the decidability result for ω to $(\mathbb{N}; S)$. There is model-theoretic and computability theoretic evidence towards a positive answer. Indeed, in the $L_{\omega_1,\omega}$ language, the structure $(\mathbb{N}; S)$ is easier to describe than ω. Formally, the Scott rank of $(\mathbb{N}; S)$ is smaller than the Scott rank of ω [1]. Third, the index set of ω is Π^0_3-complete[2] [48]. In contrast, the index set of $(\mathbb{N}; S)$ is a Π^0_2-complete set. In summary, from a descriptional complexity

[1] D. Scott describes every countable structure $\mathcal{A}$ by an $L_{\omega_1,\omega}$-sentence up to isomorphism [51]. The Scott rank measures the ordinal complexity of such sentences [18]. The higher the Scott rank of the structure $\mathcal{A}$, the harder it is to describe it in $L_{\omega_1,\omega}$-language.

[2] The index set of structure $\mathcal{A}$ is the class of all Turing machines computing the atomic diagram of $\mathcal{A}$.

viewpoint, ω is harder to describe than $(\mathbb{N}; S)$. Likewise, from the viewpoint of computability, the index set of ω is more complex than the one for $(\mathbb{N}; S)$. Yet, the isomorphism problem for automatic ω is decidable whereas the isomorphism problem for $(\mathbb{N}; S)$ is not.

2. The isomorphism problem for automatic successor word structures. Our second isomorphism problem asks if there exists an algorithm that, given two automatic successor word structures $(D_1; S_1, U_1)$ and $(D_2; S_2, U_2)$, decides if $(D_1; S_1, U_1)$ and $(D_2; S_2, U_2)$ are isomorphic. F. Stephan poses the question in [52]. Just as our first question, this problem has remained unresolved since the start of the theory of automatic structures in 1996 .In Theorem 11, we prove that no algorithm exists that decides this problem.

Here are a few important comments about our solution. First, due to the previous result (that the isomorphism problem for $(\mathbb{N}; S)$ is a Π^0_2-complete set), it is already undecidable if a given automatic structure is a successor word structure. Therefore, we want to know if there is an algorithm that, under the promise that given automatic structures $\mathcal{A}$ and $\mathcal{B}$ are successor words, decides if these structures are isomorphic. In case, when $\mathcal{A}$ or $\mathcal{B}$ is not a successor word structure, the algorithm can give any answer (not even terminate). In this sense, solving the problem is more complex than the one for $(\mathbb{N}; S)$ because we need to consider all algorithms that (1) might not even halt on some inputs, but (2) halt on all inputs from the Π^0_2-complete set of automatic successor word structures. Second, we stress that the solution to the first problem does not, in any direct way, imply a solution to the second problem. We use two new techniques. The first technique, through Lemma 13, reduces the equality problem for TMs to the equality problem of the running times of TMs. The second technique, via Lemma 14, encodes the running times of pairs of TMs (M_1, M_2) into automatic successor word structures $\mathcal{A}(M_1, M_2)$. This transformation is a key to the proof. The transformation is not necessarily commutative, that is, the structures $\mathcal{A}(M_1, M_2)$ and $\mathcal{A}(M_2, M_1)$ could be non-isomorphic. These two techniques are new and non-trivial.

3. The agreement problem. Let $\mathcal{A} = (D; \leq, U)$ be an automatic ordered word structure. Let $\tilde{U}$ be the image of U under the isomorphism from $(D; \leq)$ onto $(\mathbb{N}; \leq)$. Clearly, automatic ordered word structures $(D_1; \leq, U_1)$ and $(D_2; \leq, U_2)$ are isomorphic iff $\tilde{U}_1 = \tilde{U}_2$. It is unknown if the isomorphism problem for automatic ordered word structures is decidable. Here we attack the **agreement problem**, an approximation to the isomorphism problem. The automatic structures $(D_1; \leq, U_1)$ and $(D_2; \leq, U_2)$ **agree** on $n \in \mathbb{N}$ if $n \in \tilde{U}_1$ and $n \in \tilde{U}_2$. If no $n \in \mathbb{N}$ exists on which the structures agree then we say that these structures **disagree**.

The agreement problem asks to design an algorithm that, given two automatic ordered word structures $(D_1; \leq, U_1)$ and $(D_2; \leq, U_2)$, decides if these two structures agree on some n. The agreement problem is equivalent to $\tilde{U}_1 \cap \tilde{U}_2 \neq \emptyset$. We prove that the agreement problem is Σ^0_1-complete. The proof utilises two techniques. The first technique non-trivially implements the transformation idea from Lemma 14 used in Theorem 11. Our second technique employs the methods from D. Kuske, J. Liu, and M. Lohrey who showed the undecidability of the isomorphism problem for automatic equivalence relations [40]. Note that automatic ordered word structures $(D_1; \leq, U_1)$ and $(D_2; \leq, U_2)$ are isomorphic if and only if (a) the structures $(D_1; \leq, U_1)$ and $(D_2; \leq, D_2 \setminus U_2)$ disagree and (b) the structures $(D_1; \leq, D_1 \setminus U_1)$ and $(D_2; \leq, U_2)$ disagree. Thus, the agreement problem can viewed as an approximation to the isomorphism problem for automatic ordered word structures.

4. Intrinsic regularity in $Path_\omega = (\mathbb{N}; E_S)$. One can prove that a unary relation U in $Path_\omega$ is intrinsically regular if and only if U is finite or co-finite. The case for binary relations turns out to be more complex. The difficult case, for instance, is the successor relation S. The difficulty of whether or not S is intrinsically regular is attested by the following two facts. First, any automatic presentation of $(\mathbb{N}, S)$ is an automatic presentation of $Path_\omega$.

Second, all known automatic presentations of the structure $Path_\omega$ preserve the regularity of S. It is therefore natural to conjecture that the relation S is an intrinsically regular relation in $Path_\omega$. Counter-intuitively, in Theorem 24 we build an automatic presentation of $Path_\omega$ in which the successor relation S is not regular. Our proof employs techniques from [38] and non-trivially adjusts them for building such an automatic presentation of $Path_\omega$. As discussed in the introduction, this result implies that we can not extend the decidability theorem for the $Path_\omega$ using the generalised quantifier that corresponds to the relation S.

5. Exotic automatic word structures. Let $\mathcal{A}$ be an automatic word structure with predicate U. Let $u_0, u_1, u_2, \ldots$ be the listing of all natural numbers in U in increasing order, where $u_0 = 0$. Define $d_U(n) = u_{n+1} - u_n$, where $n \in \mathbb{N}$, the **distance function realised by $\mathcal{A}$**. The distance functions are full isomorphism invariants of word structures. Thus, the study of automatic word structures is equivalent to the study of the distance functions d_U. From Theorem 7, one shows that for any total TM M there is an automatic successor word structure $\mathcal{A}$ whose distance function bounds from above the running time of M. In contrast, the distance functions in automatic ordered words are bounded by exponential functions. See Corollaries 9 and 10. The interesting case, then, is the study of distance functions in automatic ordered word structures. In Section 6, we show that exponential, intermediate, polynomial, radical, and logarithmic functions can be realized as distance functions.

2 The isomorphism problem for $(\mathbb{N}; S)$

We prove that no algorithm exists that, given an automatic structure $\mathcal{A}$, decides if $\mathcal{A}$ is isomorphic to $(\mathbb{N}; S)$. For the proof we recall standard terminology about Turing machines.

A Turing Machine (TM) M is a tuple $(Q, \Sigma, \delta, q_0, F, R)$, where Q is the set of states, Σ is the alphabet, δ is a set of transitions of the form $(q, b) \to (q', c, \epsilon)$, $q_0 \in Q$ is the initial state and $F, R \subset Q$ are the sets of accepting and rejecting states, respectively. Transitions will also be called *instructions* of M. The transition $(q, b) \to (q', c, \epsilon)$ is interpreted as when M reads b in state q, moves to state q', writes c in the head position, and moves in the direction of $\epsilon \in \{l, r\}$ in the tape. We assume that all our TMs are deterministic.

A **configuration** C of M is a string uqv, where $u, v \in \Sigma^*$ and $q \in Q$. This indicates that the tape content is uv, M is at state q, and the head of M is at the first symbol of the string v. An **initial configuration** is $q_0 v$, where $v \in \Sigma^\star$. A configuration uqv is **halting** if $q \in F \cup R$. For configurations C and C', we write $C \to C'$ if there is an instantaneous move of M from C to C' through one of the instructions. If $C \to C'$ occurs via instruction i, we also denote this as $i(C) = C'$. There are no transitions from halting configurations.

A **run** of the machine M is a sequence of configurations C_1, C_2, ..., C_i, ... such that for all j we have $C_j \to C_{j+1}$. We say that M is **total** if its run starting at any initial configuration halts (in a halting configuration). For a technical reason, we will assume that our TMs M never **stall**, that is, for any non-halting configuration C there is a C' such that $C \to C'$. We will transform the TMs M into equivalent (reversible) Turing machines M'. The Turing machines M', as opposed to M, will be allowed to stall.

▶ **Definition 3.** *A configuration C is **legal** if it belongs to a run that starts at an initial configuration. Else, C is **illegal**. A maximal run of a TM M is called a **chain**.*

Let $(Conf(M), \to)$ be the directed graph consisting of all configurations of M and the edge relation $\to$. This graph is automatic. We call this graph the **configuration graph** of M. Thus, we can use graph-theoretic notations for configuration graphs. For every configuration $C \in Conf(M)$, we have *out-deg*(C) the cardinality of all outgoing edges from C. Similarly, *in-deg*(C) is the cardinality of all in-going edges into C.

The non-stalling condition (mentioned above) guarantees that for all configurations C, C is a halting configuration if and only if $\textit{out-deg}(C) = 0$. Because our Turing machines M are deterministic, for all configurations C of M we have $\textit{out-deg}(C) \leq 1$.

▶ **Definition 4.** *A Turing Machine M is called* ***reversible*** *if for all configurations $C \in Conf(M)$ we have $\textit{in-deg}(C) \leq 1$.*

For technical reasons, we can extend the concepts above to multi-tape Turing machines M. For instance, this can be done by coding the contents of the tapes of M into strings, e.g., through the convolution operation (see Section 1.1). To explain this let us assume that M is a two-tape TM. Let the pair $(abbqba, baqaa)$ represent the contents of the first and the second tapes, respectively, where q is a state of M. Note that both tapes have the same state of M. We code this two-tape configuration as the string

$$\begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix} \begin{bmatrix} b \\ q \end{bmatrix} \begin{bmatrix} q \\ a \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix} \begin{bmatrix} a \\ \diamond \end{bmatrix}.$$

We use this type of coding, based on the convolution operation, to represent each configuration in the configuration graphs of multi-tape Turing machines.

Turing Machines T_1 and T_2 are **equivalent** if for all $w \in \Sigma^\star$ we have T_1 accepts w if and only if T_2 accepts w, and T_1 rejects w if and only if T_2 rejects w. We briefly present the proof of the following known result [3] for the reader's intuition, as it will be useful later.

▶ **Lemma 5.** *There exists a transformation r that, given a TM M, outputs a reversible TM M_r equivalent to M.*

Proof. We outline the construction. Let M be a Turing machine. We build the TM M_r as follows. The configurations of M_r are 2-tuples (C, h) where C is a configuration of M and $h \in \delta^*$ is a sequence of instructions in δ. The transitions of M_r are defined as $(C, h) \to (C', h\ i)$ if and only if $i(C) = C'$. Thus, M_r simulates M with instruction i and records the instruction i on the second tape. ◀

The configuration graph $(Conf(M_r), \to)$ satisfies the following important property:

- Every chain is either finite or is isomorphic to $(\mathbb{N}; S)$. The graph $(Conf(M_r), \to)$ contains no finite cycles.

Note that finite chains have the **start** and the **last** configurations. A configuration C is the start (or the bottom) configuration in the chain if $\textit{in-deg}(C) = 0$. A configuration C is the last (or the top) configuration if $\textit{out-deg}(C) = 0$.

Note that even if M is total, the configuration graph of M_r might contain infinite chains. We would like to strengthen the lemma above by modifying M_r into a reversible equivalent TM M_τ such that totality of M implies that every chain in M_τ is finite.

▶ **Lemma 6.** *There is a transformation τ which, given a TM M, outputs M_τ such that:*

1. *M_τ is reversible.*
2. *M_τ is equivalent to M.*
3. *Configuration graph of M_τ consists of finite chains if and only if M is total.*

Proof. The configurations of TM M_τ are (C, h_1, h_2, D, Cnt), where $C \in Conf(M)$, h_1, $h_2 \in \delta^\star$, $D \in \{\uparrow, \downarrow\}$, and $Cnt \in \{0, 1\}$. We explain the roles of h_1, h_2, D, and Cnt:

1. The string h_1 is the instruction sequence executed so far (as in M_r) but the head of h_1 can be at any position.
2. The string h_2 checks if h_1 is a consistent sequence of instructions. The head for h_2 is at the right end of h_2.

3. The symbol D represents the direction of the head for h_1. If $D = \uparrow$, then the head moves to the right along h_1, and to the left otherwise.
4. If $Cnt = 1$, then M_τ simulates M for one step.

We represent configurations as $(C, h'\underline{j}h'', h_2, D, Cnt)$ where the underlying symbol indicates the head positions. Initial configurations of M_τ are of the form $(q_0 w, \lambda, \lambda, \uparrow, 0)$ for some $w \in \Sigma^*$. Thus, $h'\underline{j}h''$ says that the head position for the tape h_1 is at symbol j. There is no need to specify the position of the head for h_2 because it is always at the right end of h_2. Since the values of D and Cnt are finite, these can be counted as states of M_τ. Therefore, M_τ is a 3-tape Turing Machine. Now we describe transitions of M_τ:

R1: $(C,\ h_1\underline{\diamond},\ \lambda,\ \uparrow,\ 0) \to (C',\ h_1\underline{j},\ \lambda,\ \uparrow,\ 1)$, $C' = j(C)$.
Intuition: This mimics one computation step of M.

R2: $(C, h_1\underline{j},\ \lambda,\ \uparrow, 1) \to (C, h_1\underline{j},\ \lambda,\ \downarrow, 0)$.
Intuition: Preparation to verify if h_1 is a valid sequence of instructions.

R3: $(C, h'j_1\underline{j_2}h'', h_2, \downarrow, 0) \to (C', h'\underline{j_1}j_2h'', h_2j_2, \downarrow, 0)$, where we have $j_2(C') = C$.
Intuition: This rule attempts to validate the instruction sequence h_1 and records the validation history on tape h_2. Note that h_2 copies a suffix of h_1 in reverse.

R4: $(C, \underline{j_2}h'', h_2, \downarrow, 0) \to (C', \underline{j_2}h'', h_2j_2, \uparrow, 0)$, where $j_2(C') = C$ and C' is initial configuration.
Intuition: Just like Rule 2, this is a preparation step to return the head for h_1 to the rightmost position.

R5: $(C, h'\underline{j_1}j_2h'', h_2j_3, \uparrow, 0) \to (C', h'j_1\underline{j_2}h'', h_2, \uparrow, 0)$, where we have $j_3 = j_1$ and $C' = j_1(C)$
Intuition: This rule tries to reduce the size of h_2 and at the same time validates consistency between h_1 and h_2.

Thus, R1 is the simulation rule, R2 and R4 are preparation rules, and R3 and R5 are validation rules. Furthermore, if M on input w halts in t steps, then M_τ on input w runs in time proportional to t^2. One checks that the construction is correct. ◀

▶ **Theorem 7.** *The set of all automatic structures (D, E) isomorphic to $(\mathbb{N}; S)$ is Π^0_2-complete. Hence, the problem if an automatic structure is isomorphic to $(\mathbb{N}; S)$ is undecidable.*

Proof. Let $Tot = \{M \mid \text{Turing machine } M \text{ is total}\}$. This is a standard Π^0_2-complete set [25]. We reduce Tot to the set $Is(S) = \{(D; E) \mid (D; E) \text{ is automatic and } (D; E) \cong (\mathbb{N}; S)\}$. Note that $Is(S)$ is a Π^0_2-set. Indeed $(D; E) \in Is(S)$ if and only if E is an injective function (this can be computably verified) and there exists a $0_D \in D$ such that 0_D has no pre-image with respect to E and for every $x \in D$ there is an n such that $E^n(0_D) = x$. This is a Π^0_2 definition of $Is(S)$.

Let M be a Turing machine. Consider the equivalent Turing machine M_τ built in Lemma 6. Consider the configuration graph $(Conf(M_\tau);\ \to)$ of all configurations of M_τ.

Let L be chain in $(Conf(M_\tau); \to)$.

This chain has the first element, $first(L)$, such that every other configuration in L is reachable from $first(L)$ through a finite number of transitions $\to$. If L is finite then it has its top element as well. The top element of L is its final element (with no successor), where $L = C_1 \to C_2 \to \ldots \to C_k \to \ldots$ and $C_1 = first(L)$. We introduce $C_1^r \leftarrow^r C_2^r \leftarrow^r \ldots \leftarrow^r C_k^r \leftarrow^r \ldots$ the "reverse" copy of the chain, where C^r is obtained from C by renaming each character c in C by the new character c^r. The following lemma utilizes two copies of $(Conf(M_\tau), \to)$ to build an automatic structure (D_M, E_M) which captures totality of any TM M.

▶ **Lemma 8.** *For any TM M, there exists an automatic structure (D_M, E_M) such that the structure (D_M, E_M) is isomorphic to $(\mathbb{N}; S)$ if and only if the configuration graph $(Conf(M_\tau), \to)$ consists of only finite chains.*

Let M be a Turing machine. We construct the Turing machine M_τ as in Lemma 6. Based on M_τ we build the structure $(D_M; E_M)$ as follows:

1. The domain D_M is the set $\{C \mid C \in Conf(M_\tau)\} \cup \{C^r \mid C \in Conf(M_\tau)\}$.
2. The set E_M is the union of the transitions $\to$ and $\leftarrow^r$ together with the following two sets of transitions:
   a. Transitions (C, C^r) such that C is the top element of a chain L in $Conf(M_\tau)$.
   b. Transitions (C^r, C'), where $C = first(L)$ for some chain L and C' is the length-lexicographic successor of C among the set of all first elements of chains in $Conf(M_\tau)$.

One can depict the structure $(D_M; E_M)$ as follows. Let $L_1, L_2, \ldots, L_n, \ldots$ be the list of all chains of the graph $(Conf(M_\tau), \to)$ ordered by the length-lexicographical order of their first elements . Amend this list as follows: $L_1, L_1^r, L_2, L_2^r, \ldots, L_n, L_n^r, \ldots$. Starting from the first element of L_1, The relation E_M successively applies $\to$ transition until the top element of L_1 is reached. Once the top element C of L_1 is reached, the relation E_M makes a transition to C^r, the reverse of C, and then reverses $\to$ transitions by applying $\leftarrow^r$ until the reverse of the first element of L_1 is reached. Then E_M transits to the first element of L_2, and repeats the process[3].

It is clear that $(D_M; E_M)$ is an automatic graph. In this graph the in-degree of each element is at most 1. Also, the out-degree of each element is exactly 1. If the Turing machine M is total, then every chain in the space $(Conf(M_\tau); \to)$ is finite. Moreover, the configuration graph excludes cycles. Therefore, E_M starting from the first element in L_1 reaches all elements in D_M. If M is not total, then there must exist an infinite chain L in the directed graph $(Conf(M_\tau); \to)$. Let L_i be the first such infinite chain. Hence, once E_M is applied to the first element of L_i, all the transitions in L_i stay inside L_i. Hence, no element in L_{i+1} is reachable from the first element of L_1. Therefore $(D_M; E_M)$ is isomorphic to $(\mathbb{N}; S)$ if and only if M is total. We proved the lemma, and hence the theorem. ◀

2.1 Distance functions on automatic successor word structures

We apply the results of the previous section to build various types of distance functions for automatic successor word structures. Given $\mathcal{A}$ an automatic word structure with predicate U, let $u_0, u_1, u_2, \ldots$ be the listing of all natural numbers in U in increasing natural order, where $u_0 = 0$. Recall that the function $d_U(n) = u_{n+1} - u_n$, where $n \in \mathbb{N}$, is called **the distance function realised by** $\mathcal{A}$. Lemma 6 and Theorem 7 give us tools that code various distance functions in automatic successor word structures.

Let M be a Turing machine with running time T_M. Construct the Turing machine M_τ as in Lemma 6. Then for all $x \in \{0,1\}^\star$ we have $T_M(x) \leq T_{M_\tau(x)} \leq CT_M(x)^3$, where $C \geq 1$ is a fixed constant. Now consider the automatic successor word structure $(D_M; E_M, U)$, where $(D_M; E_M)$ is constructed in Theorem 7, and $U = \{0,1\}^\star \subseteq D_M$. From the construction of $(D_M; E_M)$ and the unary predicate U, it is not hard to see that $2CT_M(u_n)^3 \leq d_U(n)$, where u_n is the length-lexicographic order of $\{0,1\}^\star$.

[3] One might suggest to simplify this description and propose that E_M moves from the top of L_i to the bottom of L_{i+1} directly. This, indeed, removes the need for reverse configurations C^r and the transitions $\leftarrow^r$. However, this simplification does not guarantee that the relation built in this way is regular.

▶ **Corollary 9.** *For any total Turing machine M there exists an automatic successor word structure $\mathcal{A}$ such that the distance function d_U of $\mathcal{A}$ bounds the running time T_M.* ◀

▶ **Corollary 10.** *There exist automatic successor word structures whose distance functions are non-elementary. (This corollary strengthens and generalizes the results in [5])* ◀

3 Automatic successor word structures

Our interest is in the set $Is(SW)$ of all pairs $(\mathcal{A}, \mathcal{B})$ such that both $\mathcal{A}$ and $\mathcal{B}$ are isomorphic automatic successor word structures. The question is this: Is there an algorithm that, given two automatic successor word structures, decides if these structures are isomorphic? This is a promise problem in the sense that we care about those input that are word automatic successor structures. If any of the inputs is not a word automatic structure, then we do not care about the output of the algorithm. See **our contribution** in the Introduction Section.

Now we prove that this isomorphism problem for automatic successor word structures is undecidable. Our proof uses the construction of the structures $(D_M; E_M)$ from Theorem 7.

▶ **Theorem 11.** *The isomorphism problem for automatic successor word structures is undecidable.*

Proof. Our goal is to show that no algorithm exists that, given two automatic presentations of successor word structures, decides if these structures are isomorphic. In our proof, we use the operator τ defined in Lemma 6. By this lemma, we always assume that our Turing machines are reversible. Moreover, a Turing machine M is total if and only if, in the configuration graph $(Conf(M_\tau); \to)$ of M, all chains are finite. Our proof also uses the following folklore fact from computability theory [25]:

▶ **Lemma 12.** *No algorithm exists that given two total Turing machines M_1 and M_2 decides if M_1 is equivalent to M_2.*

We assume that the outputs of Turing machines are either *accept* or *reject*. Let M be a total Turing machine (over an alphabet Σ). For each $w \in \Sigma^\star$, let $T_M(w)$ be the number of steps taken for M to halt its computation on w. Then the following lemma can be established upon Lemma 12:

▶ **Lemma 13.** *No algorithm exists that given two total TMs M_1 and M_2 decides if $T_{M_1} = T_{M_2}$.*

Let M and M' be Turing machines. For M and M', we build the structure $\mathcal{A}(M, M')$ using $(D_M; E_M)$ and $(D_{M'}; E_{M'})$ as constructed in the proof of Theorem 7.

We assume that the domains D_M and $D_{M'}$ are disjoint. We build $\mathcal{A}(M, M')$ as follows.

1. The domain $D(M, M')$ is $D_M \cup D_{M'}$.
2. The unary predicate $U(M, M')$ is the set of all initial configurations of M and M'.
3. To build the binary relation $S_{(M,M')}$ we need a few notations. In the structures $(D_M; E_M)$ and $(D_{M'}; E_{M'})$, respectively, let us list all the chains ordered according to the length-lexicographic order on the first elements:
   $$L_{1,M},\ L^r_{1,M},\ \ldots,\ L_{i,M},\ L^r_{i,M},\ \ldots \text{ and } L_{1,M'},\ L^r_{1,M'},\ \ldots,\ L_{i,M'},\ L^r_{i,M'},\ \ldots.$$
   In each of these lists consider the sub-sequences of chains whose first elements are legal configurations: $L_{i_1,M},\ \ldots,\ L_{i_k,M},\ \ldots$ and $L_{j_1,M'},\ \ldots,\ L_{j_k,M'},\ \ldots$. For these sub-sequences, the set of first elements of the chains are regular. Denote these first elements as: $\ell_{i_1,M},\ \ldots,\ \ell_{i_k,M}, \ldots$ and $\ell_{j_1,M'},\ \ldots,\ \ell_{j_k,M'}, \ldots$. Define the relation $S_{(M,M')}$ by:
   - $S_{(M,M')}(\ell^r_{i_k,M}) = \ell_{j_k,M'}$ [Simulate M' on the same input as M]
   - $S_{(M,M')}(\ell^r_{j_k,M'}) = E_M(\ell^r_{i_k,M})$ [Both M' and M are simulated on same input.]

- $S_{(M,M')}(E_M^{-1}(\ell_{i_{k+1},M})) = E_{M'}(\ell^r_{j_k,M'})$
- $S_{(M,M')}(E_{M'}^{-1}(\ell_{j_{k+1},M'})) = \ell_{i_{k+1},M}$
- In other cases where no rule can be matched , the successor function will just follow $E_M, E_{M'}$, i.e.

$$S_{(M,M')}(w) = E_M(w), w \in D_M \quad S_{(M,M')}(w) = E_{M'}(w), w \in D_{M'}$$

▶ **Lemma 14.** *$\mathcal{A}(M, M')$ is automatic. If M and M' are total, then $\mathcal{A}(M, M')$ is a successor word structure.*

▶ **Lemma 15.** *Let M_1 and M_2 be total. Then $T_{M_1} = T_{M_2}$ iff $\mathcal{A}(M_1, M_2) \cong \mathcal{A}(M_2, M_1)$.*

If $T_{M_1} = T_{M_2}$ then $\mathcal{A}(M_1, M_2)$ and $\mathcal{A}(M_2, M_1)$ are isomorphic. Assume that $T_{M_1} \neq T_{M_2}$. Let w be the first element, in the length-lexicographic order, such that $T_{M_1}(w) \neq T_{M_2}(w)$. Assume that ℓ_{i_k,M_1} is the initial configuration that corresponds to w. Then in the structure $\mathcal{A}(M_1, M_2)$, the elements ℓ_{i_k,M_1} and ℓ_{j_k,M_2} belong to the unary predicate. The distance between these two elements ℓ_{i_k,M_1} and ℓ_{j_k,M_2} equals $2T_{M_1}(w) + 1$. Similarly, the elements ℓ_{j_k,M_2} and ℓ_{i_k,M_1} belong to the unary predicate in $\mathcal{A}(M_2, M_1)$. The distance between ℓ_{j_k,M_2} and ℓ_{i_k,M_1} in $\mathcal{A}(M_2, M_1)$ equals $2T_{M_2}(w) + 1$. Obviously, these two distances are not equal as $T_{M_1}(w) \neq T_{M_2}(w)$. If there is an isomorphism f from $\mathcal{A}(M_1, M_2)$ onto $\mathcal{A}(M_2, M_1)$, then for the isomorphism f we must have $f(\ell_{i_k,M_1}) = \ell_{j_k,M_2}$ and $f(\ell_{j_k,M_2}) = \ell_{i_k,M_1}$. This is impossible because the distances between these pairs of elements are not equal. ◀

4 The agreement problem

Recall that the ordered word structures are structures isomorphic to $(\mathbb{N}; \leq, U)$, where $\leq$ is the natural order and U is a unary predicate. The isomorphism problem for automatic ordered word structures is a natural follow-up to the isomorphism problem for automatic successor word structures. V. Bárány was the first who asked and investigated the isomorphism problem for automatic ordered word structure [1][Question 5.9.3]. The techniques used in addressing the isomorphism problem for automatic successor word structures can not be applied in this case. One reason is that the distance functions d_U in automatic ordered word structures are bounded by exponential functions while the function d_U for automatic successor word structures can bound from above any given computable function (see Corollary 9). Another reason is that the set of all automatic ordered word structures is decidable as opposed to Π^0_2-completeness of the set of all automatic presentations of successor word structures.

Let $\mathcal{A} = (D; \leq, U)$ be an automatic ordered word structure. Let $\tilde{U}$ be the image of U in $\mathbb{N}$ under the isomorphism from $(D; \leq)$ onto $(\mathbb{N}; \leq)$.

▶ **Proposition 16.** *Two word structures $\mathcal{A}_1 = (D_1; \leq, U_1)$ and $\mathcal{A}_2 = (D_2; \leq, U_2)$ are isomorphic if and only if $\tilde{U}_1 = \tilde{U}_2$.* ◀

In this section we attack the following **agreement problem**.

▶ **Definition 17.** *Automatic word structures $\mathcal{A}_1 = (D_1; \leq, U_1)$ and $\mathcal{A}_2 = (D_2; \leq, U_2)$ **agree** if the set $\tilde{U}_1 \cap \tilde{U}_2$ is a non-empty set. Otherwise, we say that these structures **disagree**.*

The agreement problem asks to design an algorithm that given automatic ordered word structures $\mathcal{A}_1 = (D_1; \leq, U_1)$ and $\mathcal{A}_2 = (D_2; \leq, U_2)$ decides if these two structures agree.

▶ **Theorem 18.** *The agreement problem for automatic ordered word structures is Σ^0_1-complete.*

Proof. Let $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. We set $\mathbb{N}[\bar{x}]$ to be the set of all polynomials over $\mathbb{N}$ in $\bar{x}$ variables. We use the following theorem from [42] that solves Hilbert's 10th problem.

▶ **Theorem 19.** *For any Σ_1^0-complete $X \subseteq \mathbb{N}_+$, there exist two polynomials $p_1(x, \bar{x})$ and $p_2(x, \bar{x})$ from $\mathbb{N}[x, \bar{x}]$ such that $n \in X$ if and only if for some $\bar{x}$ we have $p_1(n, \bar{x}) = p_2(n, \bar{x})$.*

From now on we fix $k = |\bar{x}|$ which witnesses this theorem. This theorem implies Σ_1^0-completeness of the following set: $Agr = \{(p_1, p_2) \mid p_1, p_2 \in \mathbb{N}[\bar{x}] \ \& \ \exists \bar{x}(p_1(\bar{x}) = p_2(\bar{x}))\}$.

We define a computable bijection $\sigma = cnt \circ \pi : \mathbb{N}_+^k \to \mathbb{N}$, where :

- The function $\pi : (x_1, \ldots, x_k) \mapsto a_1^{x_1} \ldots a_k^{x_k}$ encodes tuples as words in $a_1^+ \ldots a_k^+$;
- The function cnt maps each word $a_1^{x_1} \ldots a_k^{x_k}$ to its position in the length-lexicographic ($\leq_{\ell\ell}$) ordering of $a_1^+ \ldots a_k^+$.

Using σ, given two polynomials $p_1,\ p_2 \in \mathbb{N}[\bar{x}]$, we define the following function:

$$\psi_i(\bar{x}) = p_i(\bar{x}) + \sum_{\sigma(\bar{z}) < \sigma(\bar{x})} p_1(\bar{z}) + p_2(\bar{z}).$$

Let U_1 and U_2 be the ranges of ψ_1 and ψ_2, respectively. As the sets U_1 and U_2 depend on p_1 and p_2 which we write as $U_1(p_1, p_2)$ and $U_2(p_1, p_2)$, respectively. Here is a lemma that connects the set Agr with the sets U_1 and U_2.

▶ **Lemma 20.** *For $p_1, p_2 \in \mathbb{N}[\bar{x}]$, we have the following. The set $U_1 \cap U_2$ is non-empty if and only if the polynomials p_1 and p_2 agree on some $\bar{x}$, that is, $(p_1, p_2) \in Agr$.*

Consider the following structures: $(\mathbb{N}; \leq, U_1(p_1, p_2))$ and $(\mathbb{N}; \leq, U_2(p_1, p_2))$. These can be viewed as ordered counterparts of $\mathcal{A}(\mathcal{M}_1, \mathcal{M}_2)$ and $\mathcal{A}(\mathcal{M}_2, \mathcal{M}_1)$ from Theorem 11.

▶ **Lemma 21.** *For any given polynomials $p_1, p_2 \in \mathbb{N}[\bar{x}]$, the two structures $(\mathbb{N}; \leq, U_1(p_1, p_2))$ and $(\mathbb{N}; \leq, U_2(p_1, p_2))$ have automatic presentations.*

Proof. We use the ideas from D. Kuske et al. [40]. Let $\mathcal{M} = (S, \Delta, I, F)$ be an NFA over alphabet Σ. For $w \in \Sigma^\star$, let $Run(\mathcal{M}, w)$ be the set of all accepting runs of $\mathcal{M}$ on w. Let $Run(\mathcal{M})$ be all accepting runs of $\mathcal{M}$. The following is proved in [40].

▶ **Lemma 22.** *For any polynomial $p(\bar{x}) \in \mathbb{N}[\bar{x}]$, we can build a non-deterministic finite automaton $\mathcal{M}_p$ over the alphabet Σ_k such that $L(\mathcal{M}_p) = a_1^+ \ldots a_k^+$ and for all $(x_1, \ldots, x_k) \in \mathbb{N}_+^k$, we have $p(\bar{x}) = |Run(\mathcal{M}_p, a_1^{x_1} \ldots a_k^{x_k})|$.*

Let $p_1, p_2 \in \mathbb{N}[\bar{x}]$. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be the NFA constructed according to the lemma above. So, $\mathcal{M}_1 = \mathcal{M}_{p_1}$ and $\mathcal{M}_2 = \mathcal{M}_{p_2}$. We can assume that the sets of states of these two automata are disjoint. Hence, $Run(\mathcal{M}_1) \cap Run(\mathcal{M}_2) = \emptyset$.

Now we build an automatic presentation $\mathcal{A}_1 = (D_1; \leq_1, S_1)$ of $(\mathbb{N}; \leq, U_1(p_1, p_2))$. An automatic presentation $\mathcal{A}_2$ of the structure $(\mathbb{N}; \leq, U_2(p_1, p_2))$ is built similarly.

1. The domain D_1 is the set $Run(\mathcal{M}_1) \cup Run(\mathcal{M}_2)$. The domain D_1 is a regular set.
2. To define the linear order $\leq_1$ on D_1 we use the relation $R = \{(r, w) \mid r \in Run(\mathcal{M}, w)\}$. Clearly, the relation R is regular. If $(r, w) \in R$, then we write $label(r) = w$. Given two elements (accepting runs) r_1 and r_2 from D_1, we define the order $\leq_1$ as follows:
   a. When $label(r_1) = label(r_2)$:
      - If $r_1 \in \mathcal{M}_1$ and $r_2 \in \mathcal{M}_2$, then $r_1 <_1 r_2$.
      - For runs r_1 and r_2 of the same automaton, $r_1 \leq_1 r_2$ whenever $r_1 \leq_{\ell\ell} r_2$.
   b. If $label(r_1) <_{\ell\ell} label(r_2)$, the ordering is defined as $r_1 <_1 r_2$.

It is not hard to see that the relation $\leq_1$ is regular.

Thus, we have built an automatic structure $(D_1; \leq_1)$. This structure is isomorphic to the linearly ordered set $(\mathbb{N}; \leq)$ via the bijective function $\xi[p_1, p_2] : \mathbb{N} \to D_1$ defined by

$$\xi[p_1, p_2](n) = r \text{ if } |\{r' \mid r' \leq_1 r\}| = n.$$

Let S_1 be the image of the unary relation $U_1(p_1, p_2)$ in $\mathbb{N}$ under the isomorphism $\xi[p_1, p_2]$. Our goal is to show that S_1 is a regular set. If so, then we will have built the automatic presentation $(D_1; \leq_1, S_1)$ of the structure $(\mathbb{N}; \leq, U_1(p_1, p_2))$.

The remaining part is to prove the regularity of S_1. We claim that S_1 is identical to the following set $\{r \in \mathcal{M}_1 \mid \forall t \in \mathcal{M}_1(label(t) = label(r) \to (t \leq_1 r)\}$.

Indeed, the set S_1 collects every r_s which are greater than any other run $r_w \in \mathcal{M}_1$ with the same label $label(r_w) = label(r_s) = w$. Moreover, no two distinct runs in S_1 can share the same label w. Thus, it's sufficient to prove that $r_s = \xi[p_1, p_2](\psi_1(\pi^{-1}(w))$ for any input $w = label(r_s)$.

Note that $\{r \mid r \leq_1 r_s\}$ can be partitioned into two part:

- $\{r \mid label(r) <_{\ell\ell} r_s\}$, whose cardinality is the sum $\sum_{w' <_{\ell\ell} w} |\{r \in D_1 | label(r) = w'\}|$.
- $\{r \in \mathcal{M}_1 \mid label(r) = w, r \leq_1 r_s\}$ which equals to $\{r \in \mathcal{M}_1 \mid label(r) = w\}$ due to maximality of r_s.

The following facts hold for any input w:

1. $|\{r \in \mathcal{M}_1 \mid label(r) = w\}| = |Run(\mathcal{M}_1, w)| = p_1(\pi^{-1}(w))$.
2. $|\{r \in \mathcal{M}_2 \mid label(r) = w\}| = |Run(\mathcal{M}_2, w)| = p_2(\pi^{-1}(w))$.
3. $|\{r \in D_1 \mid label(r) = w\}| = p_1(\pi^{-1}(w)) + p_2(\pi^{-1}(w))$

We finally conclude that $|\{r | r \leq_1 r_s\}| = \sum_{\sigma(w') < \sigma(w)} p_1(\pi^{-1}(w)) + p_2(\pi^{-1}(w)) + p_1(\pi^{-1}(w))$. Therefore, $r_s = \xi[p_1, p_2](\psi_1(\pi^{-1}(w)))$ holds for all w. ◀

Thus, with Lemma 20 and Lemma 21 in our hand, we have proved the theorem. ◀

▶ **Corollary 23.** *Given two automatic ordered word structures $\mathcal{A}_1$ and $\mathcal{A}_2$, it's Π^0_1-complete to decide if $\tilde{U}_1 \subseteq \tilde{U}_2$.* ◀

5 Is S intrinsically regular in $(\mathbb{N}; E_S)$?

Any automatic presentation of $(\mathbb{N}; S)$ is an automatic presentation of $(\mathbb{N}; E_S)$. We prove that the reverse does not hold. This is counter-intuitive. Indeed, in a computability-theoretic setting, the computability of E_S implies the computability of S. Contrasting this, our result shows that it is not possible to build finite automata for S from finite automata recognizing the relation E_S. Our proof non-trivially adapts ideas from [38, Theorem 4.1, page 447].

▶ **Theorem 24.** *There is an automatic presentation of structure $(\mathbb{N}; E_S)$ in which the successor relation S is not regular.*

Proof. The domain D of the desired automatic structure is Σ^* with $\Sigma = \{0, 1\}$. In our construction, we will use the following three auxiliary functions from [38]:

1. Every binary string x is uniquely decomposed into:
   - $ep(x) : \Sigma^* \to \Sigma^*$: bits of x at even positions (indexing starts at 0).
   - $op(x) : \Sigma^* \to \Sigma^*$: bits of x at odd positions

   Thus, x and its pair $\langle ep(x), op(x)\rangle$ are equivalent representations.

2. Let $n_x = |ep(x)|$ and $m_x = |op(x)|$. Thus, $n_x = m_x$ or $n_x = m_x + 1$ and $|x| = n_x + m_x$. Both $ep(x)$ and $op(x)$ are interpreted as natural numbers written in the least-significant-bit-first binary form.
3. Define $next : \Sigma^* \to \Sigma^*$ as: $next(x) = (ep(x) + 2op(x) + 1) \mod 2^{n_x}$.

Let z be a binary string. Call z the **start-point** if $ep(z) = 0^*$. Our fourth auxiliary function $f : \Sigma^\star \to \Sigma^\star$ is the following:

(a) If x is such that $n_x \leq 2$ then $f(x)$ is the length-lexicographic successor of x.
(b) If $\langle next(x), op(x)\rangle$ is not a start-point, then $f(x) = \langle next(x), op(x)\rangle$.
(c) If $\langle next(x), op(x)\rangle$ is a start-point and $op(x) < 2^{m_x} - 1$ then $f(x) = y$, where $|y| = |x|, ep(y) = 0^{n_x}$ and $op(y) = op(x) + 1$
(d) If $\langle next(x), op(x)\rangle$ is a startpoint and $op(x) = 2^{m_x} - 1$ then we have $f(x) = 0^{n_x+m_x+1}$.

We informally explain f. Think of $op(x)$ as a parameter of input x viewed as $\langle ep(x), op(x)\rangle$. Starting at $ep(x) = 0^{n_x}$, f consecutively adds $2op(x) + 1$ to $ep(x)$. Once, $ep(x)$ runs through all elements in $\{0, \ldots, 2^{n_x} - 1\}$ and comes back to 0^{n_x}, the value $op(x)$ is incremented by 1.

For string $x = \langle ep(x), op(x)\rangle$, one can inductively check the fact that there is a unique $0 \leq u \leq 2^{n_x} - 1$ such that $ep(x) = u \cdot (2op(x)+1) \mod 2^{n_x}$. Based on this, we define function $U(x) = u$ which, given the string x, outputs the solution to the equation $ep(x) = u \cdot (2op(x)+1)$ modulo 2^{n_x}. So, $U(x)$ is the distance between x and the start-point $\langle 0^{n_x}, op(x)\rangle$.

Set $U_> = \{x \mid U(x) > 2^{n_x-1}\}$, which has the following property in our construction:

▶ **Lemma 25** ([38]). *The unary predicate $U_>$ is not regular.* ◀

Using Lemma 25, we now define the desired function g:

1. If $n_x \leq 2$, then $g(x)$ is the successor of x with respect to length-lexicographic ordering.
2. If $0 < U(x) < 2^{n_x-1}$, then $g(x) = f^{-1}(x)$.
3. If $U(x) = 0$, then $g(x) = \langle 10^{n_x} - 1, op(x)\rangle$.
4. If $2^{n_x-1} \leq U(x) < 2^{n_x} - 1$, then $g(x) = f(x)$.
5. If $(U(x) = 2^{n_x} - 1) \wedge (op(x) < 2^{m_x} - 1)$, then $g(x) = f^{-1}(\langle 10^{n_x-1}, op(x) + 1\rangle)$.
6. If $(U(x) = 2^{n_x} - 1) \wedge (op(x) = 2^{m_x} - 1)$, then $g(x) = f^{-1}(10^{n_x+m_x})$.

The function g is a successor on the domain $\{0, 1\}^\star$. Call the rules (2) and (4) above **non-regular rules**. Consider now the graph of g: $Graph_g(x, y) = \{(x, y) \mid y = g(x)\}$. The non-regular rules, by Lemma 25, guarantee that the successor function g is not a regular relation. Thus, we have a non-automatic presentation $(\{0, 1\}^\star, g)$ of the successor structure.

Consider the edge relation E_g defined by g. We show that E_g is regular. For this, consider the rules (3), (5), and (6). Call them **regular rules** as they are finite automata recognizable. The edge relation E_g can be defined as follows. Consider a pair $\{x, y\}$.

1. If $U(x), U(y) \notin \{0, 2^{n_x} - 1, 2^{n_x-1}\}$, then $E(x, y)$ holds when $f(x) = y \vee f(y) = x$.
2. If $U(x) = 0$, then $E(x, y)$ holds when $(U(y) = 1 \vee U(y) = 2^{n_x-1}) \wedge op(x) = op(y)$.
3. If $U(x) = 2^{n_x-1}$, then $E(x, y)$ holds when $y = f(x) \vee (U(y) = 0 \wedge op(x) = op(y))$.
4. If $U(x) = 2^{n_x} - 1 \wedge op(x) < 2^{m_x} - 1$, then $E(x, y)$ holds when $x = f(y) \vee (U(y) = 2^{n_x-1} - 1 \wedge op(y) = op(x) + 1)$.
5. If $U(x) = 2^{n_x} - 1 \wedge op(x) = 2^{m_x} - 1$, then $E(x, y)$ holds when $x = f(y) \vee (U(y) = 2^{n_x-1} - 1 \wedge op(y) = 0 \wedge |y| = |x| + 1)$.

The regularity of these rules depends on the regularity of the unary relation $U(x) \in \{0, 2^{n_x-1}, 2^{n_x} - 1, 2^{n_x-1} - 1\}$ given by the regular rules (3), (5) and (6) in the definition of g. More formally, this unary relation can be recognised as follows:

1. $U(x) = 0$ if $ep(x) = 0$.
2. $U(x) = 2^{n_x-1}$ if $ep(x) \in 10^*$.

3. $U(x) = 2^{n_x} - 1$ if $\langle next(x), op(x)\rangle$ is a startpoint.
4. $U(x) = 2^{n_x - 1} - 1$ if $\exists y(U(y) = 2^{n_x - 1} \wedge f(x) = y)$.

So, the relation E_g is regular. The theorem is proved. ◀

6 Distance functions on ordered words

The distance functions on automatic successor word structures can bound, by Corollary 9, any computable function. In contrast, the distance functions on automatic ordered word structures are bounded by an exponential function. Hence, one would like to study distance functions realised by automatic ordered word structures.

A *radical function* is $r(n) = \lfloor \sqrt[a]{n} \rfloor$ where $a > 1$. A function $f : \mathbb{N} \to \mathbb{N}$ is *intermediate* if asymptotically f is strictly between 2^n and any polynomial with non-negative coefficients. An example is $b^{\lceil \sqrt[a]{n} \rceil}$, where $a, b \in \mathbb{N}$, $a, b \geq 2$.

▶ **Theorem 26.** *Polynomials, exponents, radical, logarithmic, and intermediate functions can all be realised on automatic word ordered structures as distance functions.* ◀

7 $(\mathbb{N}; S)$ versus $(\mathbb{N}; \leq)$

Let $M_a = \{n \mid n \text{ is a multiple of } a\}$, where $n \geq 1$. In known automatic presentations of $(\mathbb{N}; S)$, the relation $\leq$ is regular. The unary predicate M_a is definable in the extension of the FO-logic based on $\leq$ and with "there exists n many mod m" quantifier $\exists^{n,m}$. Hence, if $\leq$ is regular, then so is the set M_a [38]. Known automatic presentations $(D; S_D)$ of $(\mathbb{N}; S)$ (over alphabet Σ) in which $\leq$ is not regular[38] have the following two properties:

- **Sparseness Property**: The function $n \mapsto |D \cap \Sigma^n|$ is polynomial.
- **Unboundedness Property**: The length difference between the pairs (a, b) with $a \leq b$ and $|b| < |a|$ is unbounded.

The Sparseness Property is used to show that M_a, $a \geq 1$, are all regular. The Unboundedness property is the key to show that $\leq$ is not regular. Two questions arise. The first question asks if "sparseness" is necessary to show that the sets M_a are regular. The second asks if the unboundedness property is necessary to show that $\leq$ is not regular in automatic presentations of the successor structure. The following theorem answers the first question.

▶ **Theorem 27.** *There is an automatic presentation $(\{0,1\}^*; S)$ of $(\mathbb{N}; S)$ such that*
1. *The transitive closure $\leq_S$ of S is not regular.*
2. *For all integers $a \geq 1$ the sets M_a are regular.* ◀

To answer the second question, we have the next definition:

▶ **Definition 28.** *A pair (a, b) in an automatic presentation $(D; S_D)$ of $(\mathbb{N}; S)$ is **non-standard** if $a \leq_{S_D} b$ and $|a| > |b|$, where $\leq_{S_D}$ is the transitive closure of S_D.*

The following proposition is easy to prove.

▶ **Proposition 29.** *For any presentation $(D; S_D)$ of $(\mathbb{N}; S)$, if the length difference between the components of non-standard pairs is unbounded, then $(D; S_D, \leq_{S_D})$ is not automatic.* ◀

Now we answer the second question.

▶ **Theorem 30.** *There is an automatic presentation of $(D; S_D)$ of $(\mathbb{N}; S)$ with no non-standard pairs such that $(D; \leq_{S_D})$ is not automatic but $(D; S_D, M_2, M_3, M_4, \ldots)$ is automatic.* ◀

References

1 Vince Bárány. *Automatic presentations of infinite structures*. Phd thesis, RWTH Aachen University, Germany, 2007. URL: http://darwin.bth.rwth-aachen.de/opus3/volltexte/2007/2019/.

2 Vince Bárány, Erich Grädel, and Sasha Rubin. Automata-based presentations of infinite structures. In Javier Esparza, Christian Michaux, and Charles Steinhorn, editors, *Finite and Algorithmic Model Theory*, volume 379 of *London Mathematical Society Lecture Note Series*, pages 1–76. Cambridge University Press, 2011.

3 C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973. doi:10.1147/rd.176.0525.

4 Dmitry Berdinsky and Khoussainov Bakh. Cayley automatic representations of wreath products. *International Journal of Foundations of Computer Science*, 27(02):147–159, 2016. doi:10.1142/S0129054116400049.

5 Dmitry Berdinsky, Sanjay Jain, Bakh Khoussainov, and Frank Stephan. String compression in fa-presentable structures. *Theoretical Computer Science*, 947:113705, 2023. doi:10.1016/j.tcs.2023.113705.

6 Achim Blumensath and Erich Grädel. Automatic structures. In *15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, California, USA, June 26-29, 2000*, pages 51–62. IEEE Computer Society, 2000. doi:10.1109/LICS.2000.855755.

7 Achim Blumensath and Erich Grädel. Finite presentations of infinite structures: Automata and interpretations. *Theory Comput. Syst.*, 37(6):641–674, 2004. doi:10.1007/s00224-004-1133-y.

8 J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960. doi:10.1002/malq.19600060105.

9 Cristian Calude, Peter Hertling, Bakh Khoussainov, and Yongge Wang. Recursively enumerable reals and chaitin omega numbers. *Theor. Comput. Sci.*, 255(1-2):125–149, 2001. doi:10.1016/S0304-3975(99)00159-0.

10 Christian Delhommé. Automaticité des ordinaux et des graphes homogènes. *Comptes Rendus Mathematique*, 339(1):5–10, 2004. doi:10.1016/j.crma.2004.03.035.

11 Berdinsky Dmitry and Bakh Khoussainov. On automatic transitive graphs. In Arseny M. Shur and Mikhail V. Volkov, editors, *Developments in Language Theory*, volume 8633 of *Lecture Notes in Computer Science*, pages 1–12, Cham, 2014. Springer International Publishing. doi:10.1007/978-3-319-09698-8_1.

12 David B. A. Epstein, M. S. Paterson, J. W. Cannon, D. F. Holt, S. V. Levy, and W. P. Thurston. *Word Processing in Groups*. A. K. Peters, Ltd., USA, 1992.

13 G. A. Freĭman. Addition of finite sets. *Sov. Math., Dokl.*, 5:1366–1370, 1964.

14 Moses Ganardi and Bakh Khoussainov. Automatic equivalence structures of polynomial growth. In Maribel Fernández and Anca Muscholl, editors, *28th EACSL Annual Conference on Computer Science Logic, CSL 2020, January 13-16, 2020, Barcelona, Spain*, volume 152 of *LIPIcs*, pages 21:1–21:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CSL.2020.21.

15 Alex Gavryushkin, Bakh Khoussainov, and Frank Stephan. Reducibilities among equivalence relations induced by recursively enumerable structures. *Theor. Comput. Sci.*, 612:137–152, 2016. doi:10.1016/j.tcs.2015.11.042.

16 Erich Grädel. Automatic structures: Twenty years later. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 21–34. ACM, 2020. doi:10.1145/3373718.3394734.

17 Michael Gromov. Groups of polynomial growth and expanding maps (with an appendix by Jacques Tits). *Publications Mathématiques de l'IHÉS*, 53:53–78, 1981. doi:10.1007/BF02698687.

18 Matthew Harrison-Trainor. An introduction to the scott complexity of countable structures and a survey of recent results. *Bull. Symb. Log.*, 28(1):71–103, November 2022. doi:10.1017/bsl.2021.62.

19 Greg Hjorth, Bakh Khoussainov, Antonio Montalbán, and André Nies. From automatic structures to borel structures. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 431–441. IEEE, IEEE Computer Society, 2008. doi:10.1109/LICS.2008.28.

20 Bernard R. Hodgson. *Théories décidables par automate fini*. PhD thesis, Université de Montréal, 1976.

21 Sanjay Jain, Bakh Khoussainov, Philipp Schlicht, and Frank Stephan. Tree-automatic scattered linear orders. *Theoretical Computer Science*, 626:83–96, 2016. doi:10.1016/j.tcs.2016.02.008.

22 Sanjay Jain, Bakh Khoussainov, Philipp Schlicht, and Frank Stephan. The isomorphism problem for tree-automatic ordinals with addition. *Information Processing Letters*, 149:19–24, 2019. doi:10.1016/j.ipl.2019.05.004.

23 Sanjay Jain, Bakh Khoussainov, and Frank Stephan. Finitely generated semiautomatic groups. *Comput.*, 7(2-3):273–287, 2018. doi:10.3233/COM-180089.

24 Sanjay Jain, Bakh Khoussainov, Frank Stephan, Dan Teng, and Siyuan Zou. Semiautomatic structures. *Theory Comput. Syst.*, 61(4):1254–1287, 2017. doi:10.1007/s00224-017-9792-7.

25 Hartley Rogers Jr. *Theory of recursive functions and effective computability (Reprint from 1967)*. MIT Press, Cambridge, MA, USA, 1987. URL: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=3182.

26 Lukasz Kaiser, Sasha Rubin, and Vince Bárány. Cardinality and counting quantifiers on omega-automatic structures. In Susanne Albers and Pascal Weil, editors, *STACS 2008, 25th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 21-23, 2008, Proceedings*, volume 1 of *LIPIcs*, pages 385–396. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany, 2008. doi:10.4230/LIPIcs.STACS.2008.1360.

27 Olga Kharlampovich, Bakh Khoussainov, and Alexei Miasnikov. From automatic structures to automatic groups. *Groups, Geometry, and Dynamics*, 8(1):157–198, 2014. doi:10.4171/GGD/221.

28 Bakh Khoussainov and Nerode Anil. Automatic presentations of structures. In Daniel Leivant, editor, *Logic and Computational Complexity*, pages 367–392, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. doi:10.1007/3-540-60178-3_93.

29 Bakh Khoussainov, Steffen Lempp, and Theodore A. Slaman. Computably enumerable algebras, their expansions, and isomorphisms. *Int. J. Algebra Comput.*, 15(3):437–454, 2005. doi:10.1142/S0218196705002281.

30 Bakh Khoussainov, Jiamou Liu, and Mia Minnes. Unary automatic graphs: An algorithmic perspective. In Manindra Agrawal, Ding-Zhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation, 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings*, volume 4978 of *Lecture Notes in Computer Science*, pages 542–553, Berlin, Heidelberg, 2008. Springer. doi:10.1007/978-3-540-79228-4_47.

31 Bakh Khoussainov and Mia Minnes. Three lectures on automatic structures. In *Proceedings of Logic Colloquium*, 2007.

32 Bakh Khoussainov and Mia Minnes. Model-theoretic complexity of automatic structures. *Annals of Pure and Applied Logic*, 161(3):416–426, 2009. Papers presented at the Symposium on Logical Foundations of Computer Science 2007. doi:10.1016/j.apal.2009.07.012.

33 Bakh Khoussainov and Anil Nerode. *Automata Theory and its Applications*, volume 21. Springer Science & Business Media, 2001. doi:10.1007/978-1-4612-0171-7.

34 Bakh Khoussainov, André Nies, Sasha Rubin, and Frank Stephan. Automatic structures: Richness and limitations. *Log. Methods Comput. Sci.*, 3(2), 2007. doi:10.2168/LMCS-3(2:2)2007.

35 Bakh Khoussainov and Sasha Rubin. Graphs with automatic presentations over a unary alphabet. *J. Autom. Lang. Comb.*, 6(4):467–480, April 2001. doi:10.25596/jalc-2001-467.

36 Bakh Khoussainov and Sasha Rubin. Automatic structures: overview and future directions. *J. Autom. Lang. Comb.*, 8(2):287–301, April 2003. doi:10.25596/jalc-2003-287.

37 Bakh Khoussainov, Sasha Rubin, and Frank Stephan. On automatic partial orders. In *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 168–177. IEEE Computer Society Press, June 2003. doi:10.1109/LICS.2003.1210056.

38 Bakh Khoussainov, Sasha Rubin, and Frank Stephan. Definability and regularity in automatic structures. In Volker Diekert and Michel Habib, editors, *STACS 2004, 21st Annual Symposium on Theoretical Aspects of Computer Science, Montpellier, France, March 25-27, 2004, Proceedings*, volume 2996 of *Lecture Notes in Computer Science*, pages 440–451. Springer, 2004. doi:10.1007/978-3-540-24749-4_39.

39 Bakh Khoussainov, Theodore A. Slaman, and Pavel Semukhin. P^0_1-presentations of algebras. *Arch. Math. Log.*, 45(6):769–781, 2006. doi:10.1007/s00153-006-0013-3.

40 Dietrich Kuske, Jiamou Liu, and Markus Lohrey. The isomorphism problem on classes of automatic structures with transitive relations. *Transactions of the American Mathematical Society*, 365(10):5103–5151, October 2013. doi:10.1090/S0002-9947-2013-05766-2.

41 Dietrich Kuske and Markus Lohrey. First-order and counting theories of *omega*-automatic structures. In Luca Aceto and Anna Ingólfsdóttir, editors, *Foundations of Software Science and Computation Structures, 9th International Conference, FOSSACS 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25-31, 2006, Proceedings*, volume 3921 of *Lecture Notes in Computer Science*, pages 322–336, Berlin, Heidelberg, 2006. Springer. doi:10.1007/11690634_22.

42 Yuri Vladimirovich Matiyasevich. *Hilbert's tenth problem*. MIT press, 1993.

43 André Nies and Pavel Semukhin. Finite automata presentable abelian groups. In Sergei N. Artemov and Anil Nerode, editors, *Logical Foundations of Computer Science*, pages 422–436, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. doi:10.1007/978-3-540-72734-7_29.

44 André Nies, Frank Stephan, Markus Lohrey, Pavel Semukhin, Dimitry Berdinsky, and et al. Personal communications (over many years).

45 André Nies and Richard M. Thomas. Fa-presentable groups and rings. *Journal of Algebra*, 320(2):569–585, 2008. Computational Algebra. doi:10.1016/j.jalgebra.2007.04.015.

46 Graham P. Oliver and Richard M. Thomas. Automatic presentations for finitely generated groups. In Volker Diekert and Bruno Durand, editors, *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, volume 3404 of *Lecture Notes in Computer Science*, pages 693–704. Springer, 2005. doi:10.1007/978-3-540-31856-9_57.

47 Michael O Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the american Mathematical Society*, 141:1–35, 1969.

48 Downey Rodney and Melnikov Alexander. *Computable Structure Theory: A Unified Approach*. Springer, 2024.

49 Sasha Rubin. *Automatic structures*. PhD thesis, University of Auckland New Zealand, 2004.

50 Sasha Rubin. Automata presenting structures: A survey of the finite string case. *The Bulletin of Symbolic Logic*, 14(2):169–209, 2008. doi:10.2178/bsl/1208442827.

51 Dana Scott. Logic with denumerably long formulas and finite strings of quantifiers. *Journal of Symbolic Logic*, 36(1):1104–329, 1965. doi:10.2307/2271545.

52 Frank Stephan. Automatic structures — recent results and open questions. *Journal of Physics: Conference Series*, 622(1):012013, May 2015. doi:10.1088/1742-6596/622/1/012013.

53 Todor Tsankov. The additive group of the rationals does not have an automatic presentation. *Journal of Symbolic Logic*, 76(4):1341–1351, 2011. doi:10.2178/jsl/1318338853.