Theory Comput. Systems OF1–OF34 (2004)
DOI: 10.1007/s00224-004-1133-y

Theory of Computing Systems

Finite Presentations of Infinite Structures: Automata and Interpretations

Achim Blumensath and Erich Grädel

Mathematical Foundations of Computer Science,
Aachen University of Technology,
D-52065 Aachen, Germany
{blume,graedel}@informatik.rwth-aachen.de
www-mgi.informatik.rwth-aachen.de

Abstract. We study definability problems and algorithmic issues for infinite structures that are finitely presented. After a brief overview over different classes of finitely presentable structures, we focus on structures presented by automata or by model-theoretic interpretations. These two ways of presenting a structure are related. Indeed, a structure is automatic if, and only if, it is first-order interpretable in an appropriate expansion of Presburger arithmetic or, equivalently, in the infinite binary tree with prefix order and equal length predicate. Similar results hold for ω-automatic structures and appropriate expansions of the real ordered group. We also discuss the relationship to automatic groups.

The model checking problem for FO($\exists^\omega$), first-order logic extended by the quantifier "there are infinitely many", is proved to be decidable for automatic and ω-automatic structures. Further, the complexity for various fragments of first-order logic is determined. On the other hand, several important properties not expressible in FO, such as isomorphism or connectedness, turn out to be undecidable for automatic structures.

Finally, we investigate methods for proving that a structure does not admit an automatic presentation, and we establish that the class of automatic structures is closed under Feferman–Vaught-like products.

1. Computational Model Theory

The relationship between logical definability and computational complexity is an important issue in a number of different fields including finite model theory, databases, knowledge representation, and computer-aided verification. So far most of the research has been devoted to finite structures where the relationship between definability and complexity is by now fairly well understood (see, e.g., [19] and [34]) and has many applications in particular to database theory [1]. However, in many cases the limitation to finite structures is too restrictive. Therefore, in most of the fields mentioned above, there have been considerable efforts to extend the methodology from finite structures to suitable classes of infinite ones. In particular, this is the case for databases and computer-aided verification where infinite structures (like constraint databases or systems with infinite state spaces) are of increasing importance.

Computational model theory extends the research programme, the general approach, and the methods of finite model theory to interesting domains of infinite structures. From a general theoretical point of view, one may ask what domains of infinite structures are suitable for such an extension. More specifically, what conditions must be satisfied by a domain $\mathcal{D}$ of not necessarily finite structures such that the approach and methods of finite model theory make sense. There are two obvious and fundamental conditions:

Finite representations. Every structure $\mathfrak{A} \in \mathcal{D}$ should be representable in a finite way (e.g. by a binary string, by an algorithm, by a collection of automata, by an axiomatisation in some logic, by an interpretation, ...).

Effective semantics. For the relevant logics to be considered (e.g., first-order logic), the model-checking problem on $\mathcal{D}$ should be decidable. That is, given a sentence $\psi \in L$ and a representation of a structure $\mathfrak{A} \in \mathcal{D}$, it should be decidable whether $\mathfrak{A} \models \psi$.

These are just minimal requirements that may need to be refined according to the context and the questions to be considered. We may for instance also require:

Closure. For every structure $\mathfrak{A} \in \mathcal{D}$ and every formula $\psi(\bar{x})$, also $(\mathfrak{A}, \psi^{\mathfrak{A}})$, the expansion of $\mathfrak{A}$ with the relation defined by ψ belongs to $\mathcal{D}$.

Effective query evaluation. Suppose that we have fixed a way of representing structures. Given a representation of $\mathfrak{A} \in \mathcal{D}$ and a formula $\psi(\bar{x})$ we should be able to compute a representation of $\psi^{\mathfrak{A}}$ (or of the expanded structure $(\mathfrak{A}, \psi^{\mathfrak{A}})$).

Note that contrary to the case of finite structures, query evaluation does not necessarily reduce to model checking. Further, instead of just effectiveness of these tasks, it may be required that they can be performed within some complexity bounds.

After a brief survey on different classes of finitely presented structures in the next section, we focus on domains where structures are presented by two closely related methods, namely by finite *automata* or by model-theoretic *interpretations*. While *automatic groups* have been studied rather intensively in computational group theory (see [22] and [24]) a general notion of *automatic structures* has only been defined in [36] and its theory has been developed in [6] and [8]. These structures are defined in Section 3. Informally, a relational structure $\mathfrak{A} = (A, R_1, \ldots, R_m)$ is automatic if it is universe and its relations can be recognised by finite automata reading their inputs synchronously. We believe that automatic structures are very promising for the approach of computational model theory. Not only do automatic structures admit finite presentations, but there are also numerous interesting examples and a large body of methods that have been developed in five decades of automata theory. Further, automatic structures admit effective evaluation of all first-order queries and possess many other pleasant algorithmic properties.

Automatic structures can also be defined via interpretations. As we show in Section 4 a structure is automatic if, and only if, it is first-order interpretable in an appropriate expansion of Presburger arithmetic or, equivalently, in the infinite binary tree with prefix order and equal length predicate. Similar results hold for ω-automatic structures and appropriate expansions of the real ordered group.

Such results suggest a very general way for obtaining other interesting classes of infinite structures suitable for the approach of computational model theory: Fix a structure $\mathfrak{A}$ (or a class of such structures) with "nice" algorithmic and/or model-theoretic properties, and consider the class of all structures that are interpretable in $\mathfrak{A}$, for instance, via first-order or monadic second-order logic. Obviously each structure in this class is finitely presentable (by an interpretation). Further, since many "nice" properties are preserved under interpretations, every structure in the class inherits them from $\mathfrak{A}$. In particular, every class of queries that is effective on $\mathfrak{A}$ and closed under first-order operations is effective on the interpretation-closure of $\mathfrak{A}$.

In Section 5 we turn to decidability and complexity issues. It is shown that the model-checking problem for FO($\exists^\omega$), first-order logic extended by the quantifier "there are infinitely many", is decidable for automatic and ω-automatic structures, and the complexity for various fragments of first-order logic is investigated. On the other hand, we prove that several properties not expressible in FO, such as isomorphism of automatic structures or connectivity of automatic graphs, are undecidable.

While it is usually rather easy to show that a structure is automatic (by constructing an automatic presentation), it is often difficult to prove a structure does not admit any automatic presentation. In Section 6 we present some methods to achieve this goal.

In the final section, Feferman–Vaught-like products are introduced, and it is shown that every domain of structures that can be characterised via interpretations of a certain kind is closed under such products. In particular, this applies to automatic and ω-automatic structures.

2. Finitely Presentable Structures

We briefly survey some domains of infinite but finitely presentable structures which may be relevant for computational model theory.

Recursive structures are countable structures whose functions and relations are computable and therefore finitely presentable. They have been studied quite intensively in model theory since the 1960s (see, e.g., [2] and [23]). Although recursive model theory is very different from finite model theory, there have been some papers studying classical issues of finite model theory on recursive structures and recursive databases [28], [31], [32], [46]. However, for most applications, the domain of recursive structures is far too large. In general, only quantifier-free formulae admit effective evaluation algorithms.

Constraint databases are a modern database model admitting infinite relations that are finitely presented by quantifier-free formulae (constraints) over some fixed background structure. For example, to store geometrical data, it is useful to have not just a finite set as the universe of the database, but to include all real numbers 'in the background'. Also the presence of interpreted functions, like addition and multiplication, is desirable. The constraint database framework introduced by Kanellakis et al. [35] meets both requirements. Formally, a constraint database consists of a *context structure* $\mathfrak{A}$, like $(\mathbb{R}, <, +, \cdot)$, and a set $\{\varphi_1, \dots, \varphi_m\}$ of quantifier-free formulae defining the database relations. Constraint databases are treated in detail in [38].

Metafinite structures are two-sorted structures consisting of a finite structure $\mathfrak{A}$, a background structure $\mathfrak{R}$ (which is usually infinite but fixed), and a class of weight functions from the finite part to the infinite one. Simple examples are finite graphs whose edges are weighted by real numbers. For any fixed infinite structure $\mathfrak{R}$, the metafinite structures with background $\mathfrak{R}$ are finitely presentable and admit effective evaluation of logics that make use of arithmetic operations on $\mathfrak{R}$, but do not admit full quantification over its elements. Metafinite model theory has been developed in [27] and has been put to use for studying issues in database theory, optimisation, and descriptive complexity. In particular, metafinite structures have provided the basis for logical characterisations of complexity classes over real numbers [29].

Automatic structures are structures whose functions and relations are represented by finite automata. Informally, a relational structure $\mathfrak{A} = (A, R_1, \dots, R_m)$ is automatic if we can find a regular language $L_\delta \subseteq \Sigma^*$ (which provides names for the elements of $\mathfrak{A}$) and a function $\nu : L_\delta \to A$ mapping every word $w \in L_\delta$ to the element of $\mathfrak{A}$ that it represents. The function ν must be surjective (every element of $\mathfrak{A}$ must be named) but need not be injective (elements can have more than one name). In addition it must be recognisable by finite automata (reading their input words synchronously) whether two words in L_δ name the same elements, and, for each relation R_i of $\mathfrak{A}$, whether a given tuple of words in L_δ names a tuple in R_i. Automatic structures provide many examples of high relevance for computer science. There are also interesting connections to computational group theory, where *automatic groups* have already been studied quite intensively [22], [24]. The general notion of structures presentable by automata has been proposed in [36] and their theory has been developed in [6] and [8]. Recently, results on automatic linear orders were obtained in [18] and [37]. The notion of an automatic structure can be modified and generalised in many directions. By using automata over infinite words, we obtain the notion of **ω-automatic structures** (which, contrary to automatic structures, may have uncountable cardinality).

One of the main reasons for the importance of automatic and ω-automatic structures is that they admit effective (in fact, automatic) evaluation of all first-order queries. This follows immediately from the closure properties of regular and ω-regular relations and from the decidability of emptiness problems of finite automata. Indeed, we establish a more general result.

Theorem 2.1. *The model-checking problem for* $\mathrm{FO}(\exists^\omega)$, *first-order logic extended by the quantifier "there are infinitely many", is decidable on the domain of ω-automatic structures.*

The proof is given in Section 5.

Tree-automatic structures, which are defined by automata on finite or infinite trees, are further natural generalisations of automatic structures. They also admit effective evaluation of first-order formulae. The theory of tree-automatic structures has been developed in [6]. On the other hand, first-order logic is *not* effective on another popular extension of automatic graphs, the so-called **rational graphs** [41], which are defined by *asynchronous* multihead automata.

Tree-interpretable structures are structures that are interpretable in the infinite binary tree $\mathcal{T}^2 = (\{0,1\}^*, \sigma_0, \sigma_1)$ via a one-dimensional monadic second-order (MSO) interpretation (see Section 4 for details on interpretations). By Rabin's theorem, (MSO) formulae can be effectively evaluated on $\mathcal{T}^2$, and since MSO is closed under one-dimensional interpretations, the same holds for all tree-interpretable structures. Tree-interpretable structures form a proper subclass of the automatic structures that generalises various notions of infinite graphs that have been studied in logic, automata theory, and verification. Examples are the **context-free graphs** [42], [43], which are the configuration graphs of pushdown automata; the **HR-equational and VR-equational graphs** [15], which are defined via graph grammars; and the **prefix-recognisable graphs** [13], which can for instance be defined as graphs of form $(V, (E_a)_{a\in A})$ where V is a regular language and each edge relation E_a is a finite union of sets $X(Y \times Z) := \{(xy, xz) | x \in X,\ y \in Y,\ z \in Z\}$, for regular languages X, Y, Z.

It has been established in a series of papers that some of these classes coincide with tree-interpretable graphs (see [3], [7], and [13]).

Theorem 2.2. *For any graph $G = (V, (E_a)_{a\in A})$ the following are equivalent*:

(i) *G is tree-interpretable.*
(ii) *G is VR-equational.*
(iii) *G is prefix-recognisable.*
(iv) *G is the restriction to a regular set of the configuration graph of a pushdown automaton with ε-transitions.*

On the other hand, the classes of context-free graphs and of HR-equational graphs are strictly contained in the class of tree-interpretable graphs.

Tree-constructible structures: the Caucal hierarchy. The question arises whether there are even more powerful domains than the tree-interpretable structures on which (MSO) logic is effective. An interesting way to obtain such domains are tree constructions that associate a kind of tree unravelling with any structure. A simple variant is the **unfolding** of a labelled graph G from a given node v to the tree $\mathcal{T}(G, v)$. Courcelle and Walukiewicz [16], [17] show that the MSO theory of $\mathcal{T}(G, v)$ can be effectively computed from the MSO theory of (G, v). A more general operation, applicable to relational structures of any kind, has been invented by Muchnik [44]. Given a relational structure $\mathfrak{A} = (A, R_1, \dots, R_m)$, let its **iteration** $\mathfrak{A}^* = (A^*, R_1^*, \dots, R_m^*, suc, clone)$ be the structure with universe A^*, relations $R_i^* = \{(wa_1, \dots, wa_r) | w \in A^*, (a_1, \dots, a_r) \in R_i\}$, the successor relation ***suc*** $= \{(w, wa) | w \in A^*, a \in A\}$, and the predicate ***clone*** consisting of all elements of form waa. It is not difficult to see that unfoldings of graphs are first-order interpretable in their iterations. Muchnik's theorem states that the monadic theory of $\mathfrak{A}^*$ is decidable if the monadic theory of $\mathfrak{A}$ is (for proofs, see [5] and [49]). Define the domain of **tree-constructible structures** to be the closure of the domain of finite structures under (one-dimensional) MSO interpretations and iterations. By Muchnik's theorem, and since effective MSO model checking is preserved under interpretations, tree-constructible structures are finitely presentable and admit effective evaluation of MSO formulae.

Tree-constructible graphs form the **Caucal hierarchy**, which was defined in [14] in a slighly different way. The definition easily extends to arbitrary structures: Let $\mathcal{C}_0$ be the class of finite structures, and let $\mathcal{C}_{n+1}$ be the class of structures that are interpretable in the iteration $\mathfrak{A}^*$ of a structure $\mathfrak{A} \in \mathcal{C}_n$. There are a number of different, but equivalent, ways to define the levels of the Caucal hierarchy. For instance, one can use the inverse rational mappings from [13] rather than monadic interpretations, and simple unfoldings rather than iterations without changing the hierarchy [12]. Equivalently, the hierarchy can be defined via higher-order pushdown automata. It is known that the Caucal hierarchy is strict, and that it does not exhaust the class of all structures with decidable MSO theory. We refer to [48] and [12] for details and further information.

Ground tree rewriting graphs are defined by tree rewriting [39]. Vertices are represented by finite trees and edges are generated by ground rewriting rules. In this way one can obtain graphs that are not tree-interpretable (for instance, the infinite two-dimensional grid), but for which, in addition to first-order theory, the reachability problem also remains decidable. While universal reachability and universal recurrence (and hence general MSO formulae) are undecidable on ground tree rewriting graphs, Löding [39] exhibits a fragment of CTL (permitting EF and EGF operations, but not EG, EFG, or until operations) that can be effectively evaluated on this class.

3. Automatic Structures and Automatic Groups

As usual in logic, we consider *structures* $\mathfrak{A} = (A, R_1, R_2, \dots, f_1, f_2, \dots)$ where A is a non-empty set, called the *universe* of $\mathfrak{A}$, where each $R_i \subseteq A^{r_i}$ is a relation on A, and every $f_j : A^{s_j} \to A$ is a function on A. The names of the relations and functions of $\mathfrak{A}$, together with their arities, form the *vocabulary* of $\mathfrak{A}$. We consider constants as functions of arity 0. A *relational* structure is a structure without functions. We can associate with every structure $\mathfrak{A}$ its *relational variant* which is obtained by replacing each function $f : A^s \to A$ by its graph $G_f := \{(\bar{a}, b) \in A^{s+1} | f(\bar{a}) = b\}$.

For a structure $\mathfrak{A}$ and a formula $\varphi(\bar{x})$, let $\varphi^{\mathfrak{A}} := \{\bar{a} | \mathfrak{A} \models \varphi(\bar{a})\}$ be the relation (or query) defined by φ on $\mathfrak{A}$.

We assume that the reader is familiar with the basic notions of automata theory and regular languages. One slightly non-standard aspect is that, in order to present a structure by a list of finite automata, we need a notion of regularity not just for languages $L \subseteq \Sigma^*$ but also k-ary relations of words, for $k > 1$. Instead of introducing synchronous multihead automata that take tuples $\bar{w} = (w_1, \dots, w_k)$ of words as inputs and work synchronously on all k components of $\bar{w}$, we reduce the case of higher arities to the unary one by encoding tuples $\bar{w} \in (\Sigma^*)^k$ by a single word $w_1 \otimes \cdots \otimes w_k$ over the alphabet

$(\Sigma \cup \{\Box\})^k$, called the *convolution* of $w_1, \dots, w_k$. Here $\Box$ is a padding symbol not belonging to Σ. It is appended to some of the words w_i to make sure that all components have the same length. More formally, for $w_1, \dots, w_k \in \Sigma^*$, with $w_i = w_{i1} \cdots w_{i\ell_i}$ and $\ell = \max\{|w_1|, \dots, |w_k|\}$,

$$w_1 \otimes \ldots \otimes w_k := \begin{bmatrix} w'_{11} \\ \vdots \\ w'_{k1} \end{bmatrix} \cdots \begin{bmatrix} w'_{1\ell} \\ \vdots \\ w'_{k\ell} \end{bmatrix} \in \big((\Sigma \cup \{\Box\})^k\big)^*,$$

where $w'_{ij} = w_{ij}$ for $j \leq |w_i|$ and $w'_{ij} = \Box$ otherwise. Now, a relation $R \subseteq (\Sigma^*)^k$ is called *regular* if $\{w_1 \otimes \cdots \otimes w_k | (w_1, \dots, w_k) \in R\}$ is a regular language. Below we do not distinguish between a relation on words and its encoding as a language.

Definition 3.1. A relational structure $\mathfrak{A}$ is *automatic* if there exist a regular language $L_\delta \subseteq \Sigma^*$ and a surjective function $\nu : L_\delta \to A$ such that the relation

$$L_\varepsilon := \{(w, w') \in L_\delta \times L_\delta | \nu w = \nu w'\} \subseteq \Sigma^* \times \Sigma^*$$

and, for all predicates $R \subseteq A^r$ of $\mathfrak{A}$, the relations

$$L_R := \{\bar{w} \in (L_\delta)^r | (\nu w_1, \dots, \nu w_r) \in R\} \subseteq (\Sigma^*)^r$$

are regular. An arbitrary (not necessarily relational) structure is automatic if and only if its relational variant is.

We write AutStr$[\tau]$ for the class of all automatic structures of vocabulary τ. Each structure $\mathfrak{A} \in$ AutStr$[\tau]$ can be represented, up to isomorphism, by a list $\mathfrak{d} = \langle M_\delta,\ M_\varepsilon,\ (M_R)_{R\in\tau}\rangle$ of finite automata that recognise L_δ, L_ε, and L_R for all relations R of $\mathfrak{A}$. When speaking of an *automatic presentation* of $\mathfrak{A}$ we either mean the function $\nu : L_\delta \to A$ or such a list $\mathfrak{d}$. An automatic presentation $\mathfrak{d}$ is called *deterministic* if all its automata are, and it is called *injective* if $L_\varepsilon = \{(u, u) | u \in L_\delta\}$ (which implies that $\nu : L_\delta \to A$ is injective).

Examples. (1) All finite structures are automatic.

(2) Important examples of automatic structures are Presburger arithmetic $(\mathbb{N}, +)$ and its expansion $\mathfrak{N}_p := (\mathbb{N}, +, |_p)$ by the relation

$$x|_p y \ :\text{iff } x \text{ is a power of } p \text{ dividing } y.$$

Using p-ary encodings (starting with the least significant digit) it is not difficult to construct automata recognising equality, addition, and $|_p$.

(3) Natural candidates for automatic structures are those consisting of words. (However, note that free monoids with at least two generators do *not* have automatic presentations.) Fix some alphabet Σ and consider the structure ***Tree***$(\Sigma) := (\Sigma^*, (\sigma_a)_{a\in\Sigma}, \preceq, \mathrm{el})$

where

$$\sigma_a(x) := xa, \qquad x \preceq y : \text{iff } \exists z(xz = y), \qquad \text{and} \qquad \mathrm{el}(x, y) : \text{iff } |x| = |y|.$$

Obviously, this structure is automatic as well.

The following two observations are simple, but useful:

(1) Every automatic structure admits an automatic presentation with alphabet $\{0, 1\}$ [6].
(2) Every automatic structure admits an injective automatic presentation [36].

Automatic Groups. The class of automatic structures that have been studied most intensively are automatic groups. Let $(G, \cdot)$ be a group and let $S = \{s_1, \dots, s_m\} \subseteq G$ be a set of semigroup generators of G. This means that each $g \in G$ can be written as a product $s_{i_1} \cdots s_{i_r}$ of elements of S and hence the canonical homomorphism $\nu : S^* \to G$ is surjective. The *Cayley graph* $\Gamma(G, S)$ of G with respect to S is the graph $(G, S_1, \dots, S_m)$ whose vertices are the group elements and where S_i is the set of pairs (g, h) such that $gs_i = h$. By definition $(G, \cdot)$ is *automatic* if there is a finite set S of semigroup generators and a regular language $L_\delta \subseteq S^*$ such that the restriction of ν to L_δ is surjective and provides an automatic presentation of $\Gamma(G, S)$. (In other words, the inverse image of equality,

$$L_\varepsilon = \{(w, w') \in L_\delta \times L_\delta | \nu w = \nu w'\},$$

and $\nu^{-1}(S_i)$, for $i = 1, \dots, m$, are regular.)

Note that it is not the group structure $(G, \cdot)$ itself that is automatic in the sense of Definition 3.1, but the Cayley graph. There are many natural examples of automatic groups (see [22] and [24]). The importance of this notion in computational group theory comes from the fact that an automatic presentation of a group yields (efficient) algorithmic solutions for computational problems that are undecidable in the general case.

ω-Automatic structures. The notion of an automatic structure can be modified and generalised in a number of different directions (see [6] and [36]). In particular, we obtain the interesting class ω-AutStr of ω-automatic structures. The definition is analogous to the one for automatic structures except that the elements of an ω-automatic structure are named by infinite words from some regular ω-language and the relations of the structure are recognisable by Büchi automata.

Examples. (1) All automatic structures are ω-automatic.

(2) The real numbers with addition, $(\mathbb{R}, +)$, and indeed the expanded structure $\mathfrak{R}_p := (\mathbb{R}, +, \leq, |_p, 1)$ are ω-automatic, where $p \geq 2$ is an integer and

$$x|_p y \ : \text{iff } \exists n, k \in \mathbb{Z}\colon x = p^n \text{ and } y = kx.$$

(3) The tree automatic structures $\boldsymbol{\mathit{Tree}}(\Sigma)$ extend in a natural way to the (uncountable) ω-automatic structures $\boldsymbol{\mathit{Tree}}^\omega(\Sigma) := (\Sigma^{\leq\omega}, (\sigma_a)_{a\in\sigma}, \preceq, \mathrm{el})$.

4. Characterising Automatic Structures via Interpretations

Interpretations constitute an important tool in mathematical logic. They are used to define a copy of a structure inside another one, and thus permit the transfer of definability, decidability, and complexity results among theories.

Definition 4.1. Let L be a logic, and let $\mathfrak{A} = (A, R_0, \dots, R_n)$ and $\mathfrak{B}$ be relational structures. A (k-dimensional) *L-interpretation* of $\mathfrak{A}$ in $\mathfrak{B}$ is a sequence

$$\mathcal{I} = \big\langle \delta(\bar{x}),\ \varepsilon(\bar{x}, \bar{y}),\ \varphi_{R_0}(\bar{x}_1, \dots, \bar{x}_r), \dots,\ \varphi_{R_n}(\bar{x}_1, \dots, \bar{x}_s) \big\rangle$$

of L-formulae of the vocabulary of $\mathfrak{B}$ (where each tuple $\bar{x}$, $\bar{y}$, $\bar{x}_i$ consists of k variables) such that

$$\mathfrak{A} \cong \mathcal{I}(\mathfrak{B}) := \big(\delta^{\mathfrak{B}},\ \varphi_{R_0}^{\mathfrak{B}}, \dots,\ \varphi_{R_n}^{\mathfrak{B}}\big)/\varepsilon^{\mathfrak{B}}.$$

To make this expression well-defined we require that $\varepsilon^{\mathfrak{B}}$ is a congruence relation on the structure $\big(\delta^{\mathfrak{B}},\ \varphi_{R_0}^{\mathfrak{B}}, \dots,\ \varphi_{R_n}^{\mathfrak{B}}\big)$. We denote the fact that $\mathcal{I}$ is an L-interpretation of $\mathfrak{A}$ in $\mathfrak{B}$ by $\mathcal{I} : \mathfrak{A} \leq_L \mathfrak{B}$. If $\mathfrak{A} \leq_L \mathfrak{B}$ and $\mathfrak{B} \leq_L \mathfrak{A}$ we say $\mathfrak{A}$ and $\mathfrak{B}$ are *mutually L-interpretable*.

The epimorphism $\big(\delta^{\mathfrak{B}},\ \varphi_{R_0}^{\mathfrak{B}}, \dots,\ \varphi_{R_n}^{\mathfrak{B}}\big) \to \mathfrak{A}$ is called a *coordinate map* and is also denoted by $\mathcal{I}$. If it is the identity function, i.e., $\mathfrak{A} = \mathcal{I}(\mathfrak{B})$, we say that $\mathfrak{A}$ is *L-definable* in $\mathfrak{B}$. An interpretation $\mathcal{I}$ is *injective* if its coordinate is injective, i.e., if $\varepsilon(\bar{x}, \bar{y}) \equiv \bar{x} = \bar{y}$.

Examples. (1) Recall that we write $a|_p b$ to denote that a is a power of p dividing b. Let $V_p : \mathbb{N} \to \mathbb{N}$ be the function that maps each number to the largest power of p dividing it. It is very easy to see that the structures $(\mathbb{N}, +, |_p)$ and $(\mathbb{N}, +, V_p)$ are mutually first-order interpretable. Indeed, we can define the statement $x = V_p(y)$ in $(\mathbb{N}, +, |_p)$ by the formula $x|_p y \wedge \forall z(z|_p y \to z|_p x)$. In the other direction, $V_p(x) = x \wedge \exists z(x + z = V_p(y))$ is a definition of $x|_p y$.

(2) For every $p \in \mathbb{N}$ we write $\boldsymbol{Tree}(p)$ for the tree structure $\boldsymbol{Tree}(\{0, \dots, p-1\})$. The structures $\mathfrak{N}_p$ and $\boldsymbol{Tree}(p)$ are mutually interpretable, for each $p \geq 2$ (see [6] and [26]).

If $\mathcal{I} : \mathfrak{A} \leq_{\mathrm{FO}} \mathfrak{B}$, then every first-order formula φ over the vocabulary of $\mathfrak{A}$ can be translated to a formula $\varphi^{\mathcal{I}}$ over the vocabulary of $\mathfrak{B}$ by replacing every relation symbol R by its definition φ_R, by relativising every quantifier to δ, and by replacing equalities by ε.

Lemma 4.2 (Interpretation Lemma). *If $\mathcal{I} : \mathfrak{A} \leq_{\mathrm{FO}} \mathfrak{B}$, then*

$$\mathfrak{A} \models \varphi(\mathcal{I}(\bar{b})) \quad \textit{iff} \quad \mathfrak{B} \models \varphi^{\mathcal{I}}(\bar{b}) \quad \textit{for all} \quad \varphi \in \mathrm{FO} \quad \textit{and} \quad \bar{b} \subseteq \delta^{\mathfrak{B}}.$$

This lemma states the most important property of interpretations. For any logic L, a notion of interpretation is considered suitable if a similar statement holds, and if the logic is closed under the operation $\varphi \mapsto \varphi^{\mathcal{I}}$. Note that in the case of MSO, arbitrary k-dimensional MSO interpretations are too strong since they translate sets to relations of arity k which takes us out of MSO. On the other hand, the Interpretation Lemma does hold for *one-dimensional* MSO *interpretations*.

Interpretations provide a general and powerful method to obtain classes of finitely presented structures with a set of desired properties. One fixes some structure $\mathfrak{B}$ having these properties and chooses a kind of interpretation that preserves them. Then one considers the class of all structures which can be interpreted in $\mathfrak{B}$. Each structure $\mathfrak{A}$ of this class can be represented by an interpretation $\mathcal{I} : \mathfrak{A} \leq_{\mathrm{FO}} \mathfrak{B}$ which is a finite object, and model checking and query evaluation for such structures can be reduced to the corresponding problem for $\mathfrak{B}$. If $\mathcal{I} : \mathfrak{A} \leq_{\mathrm{FO}} \mathfrak{B}$, then Lemma 4.2 implies that

$$\varphi^{\mathfrak{A}} = \{\bar{a} \mid \mathfrak{A} \models \varphi(\bar{a})\} = \{\mathcal{I}(\bar{b}) \mid \mathfrak{B} \models \varphi^{\mathcal{I}}(\bar{b})\}.$$

Hence, the desired representation of $\varphi^{\mathfrak{A}}$ can be constructed by extending the interpretation $\mathcal{I}$ to $\langle \mathcal{I}, \varphi^{\mathcal{I}} \rangle : (\mathfrak{A}, \varphi^{\mathfrak{A}}) \leq_{\mathrm{FO}} \mathfrak{B}$.

Automatic structures are closed under first-order interpretations.

Proposition 4.3. *If* $\mathfrak{A} \leq_{\mathrm{FO}} \mathfrak{B}$ *and* $\mathfrak{B}$ *is* (ω-) *automatic*, *then so is* $\mathfrak{A}$.

Proof. Since $\mathfrak{B}$ is automatic there are regular languages L_δ for the universe, L_ε for equality, and L_R for each relation R of $\mathfrak{B}$. By the closure of regular languages under boolean operations and projections it follows that, for each first-order formula φ, the language encoding the relation $\varphi^{\mathfrak{B}}$ is also regular. □

Corollary 4.4. *The classes of automatic*, *resp.* ω*-automatic*, *structures are closed under* (i) *extensions by definable relations*, (ii) *factorisations by definable congruences*, (iii) *substructures with definable universe*, *and* (iv) *finite powers*.

As stated above the class of automatic structures can be characterised via first-order interpretations.

Theorem 4.5. *For every structure* $\mathfrak{A}$, *the following are equivalent*:

(i) $\mathfrak{A}$ *is automatic*.
(ii) $\mathfrak{A} \leq_{\mathrm{FO}} \mathfrak{N}_p$ *for some* (*and hence all*) $p \geq 2$.
(iii) $\mathfrak{A} \leq_{\mathrm{FO}}$ ***Tree***(p) *for some* (*and hence all*) $p \geq 2$.

Proof. The facts that (ii) and (iii) are equivalent and that they imply (i) follow immediately from the mutual interpretability of $\mathfrak{N}_p$ and ***Tree***(p), from the fact that these structures are automatic, and from the closure of automatic structures under interpretation.

It remains to show that every automatic structure is interpretable in $\mathfrak{N}_p$ (or ***Tree***(p)). Suppose that $\mathfrak{d}$ is an automatic presentation of $\mathfrak{A}$ with alphabet $[p] := \{0, \ldots, p-1\}$ for some $p \geq 2$ (without loss of generality, we could take $p = 2$). For every word $w \in [p]^*$, let $\mathrm{val}(w)$ be the natural number whose p-ary encoding is w, i.e., $\mathrm{val}(w) := \sum_{i<|w|} w_i p^i$. By a classical result, sometimes called the Büchi–Bruyère Theorem, a relation $R \subseteq \mathbb{N}^k$

is first-order definable in $(\mathbb{N}, +, V_p)$ if and only if

$$\{(\mathrm{val}^{-1}(x_1), \ldots, \mathrm{val}^{-1}(x_k)) | (x_1, \ldots, x_k) \in R\}$$

is regular. (See [10] for a proof of this fact and for more information on the relationship between automata and definability in expansions of Presburger arithmetic.) The formulae that define in this sense the regular language and the regular relations in an automatic presentation of $\mathfrak{A}$ provide an interpretation of $\mathfrak{A}$ in $(\mathbb{N}, +, V_p)$. Hence also $\mathfrak{A} \leq_{\mathrm{FO}} \mathfrak{N}_p$. $\square$

For automatic groups we are not free to change the coordinate map. Indeed, the definition of an automatic group requires that the function $\nu : L_\delta \to G$ be the restriction of the canonical homomorphism from S^* to G. Hence the arguments used above give us a characterisation of automatic groups in terms of definability rather than interpretability.

Theorem 4.6. $(G, \cdot)$ *is an automatic group if and only if there exists a finite set* $S \subseteq G$ *of semigroup generators such that* $\Gamma(G, S)$ *is first-order definable in* $\textbf{\textit{Tree}}(S)$.

By definition, if G is an automatic group, then for some set S of semigroup generators, the Cayley graph $\Gamma(G, S)$ is an automatic structure. Contrary to a claim in [36] the converse does not hold. A counterexample, which has been pointed out by Senizergues, is the *Heisenberg group* $\mathfrak{H}$ which is the group of affine transformations of $\mathbb{Z}^3$ generated by the maps

$$\begin{aligned}
\alpha &: (x, y, z) \mapsto (x + 1, y, z + y),\\
\beta &: (x, y, z) \mapsto (x, y + 1, z),\\
\gamma &: (x, y, z) \mapsto (x, y, z + 1).
\end{aligned}$$

Using this matrix representation of $\mathfrak{H}$, it is not difficult to construct a (three-dimensional) interpretation of $\Gamma(\mathfrak{H}, S)$ in $(\mathbb{N}, +)$, which implies that $\Gamma(\mathfrak{H}, S) \in \mathrm{AutStr}$. However, in [22] it is shown that $\mathfrak{H}$ is not automatic.

Proposition 4.7. *There exist groups G with a set of semigroup generators S such that the Cayley graph* $\Gamma(G, S)$ *is an automatic structure without G being an automatic group.*

We now turn to ω-automatic structures. To provide a similar characterisation we can use an equivalent of the Büchi–Bruyère Theorem for encodings of ω-regular relations. One such result has been obtained by Boigelot et al. [9]. Using natural translations between ω-words over $[p]$ and real numbers, they prove that a relation over $[p]^\omega$ can be recognised by a Büchi automaton if and only if its translation is first-order definable in the structure $(\mathbb{R}, +, <, \mathbb{Z}, X_p)$ where $X_p \subseteq \mathbb{R}^3$ is a relation that explicitly represents the translation between $[p]^\omega$ and $\mathbb{R}$. $X_p(x, y, z)$ holds iff there exists a representation of x by a word in $[p]^\omega$ such that the digit at the position specified by y is z. A somewhat unsatisfactory aspect of this result is the assumption that the encoding relation X_p must be given as a basic relation of the structure. It would be preferable if more natural expansions of the additive real group $(\mathbb{R}, +)$ could be used instead.

We show here that this is indeed possible if, as in the case of $\mathfrak{N}_p$, we use a restricted variant of the divisibility relation. Recall that the structures $\mathfrak{R}_p$ and $\boldsymbol{Tree}^{\omega}(p)$ (introduced at the end of Section 3) are ω-automatic. As a first step we show that the behaviour of Büchi automata recognising regular relations over $[p]^{\omega}$ can be simulated by first-order formulae in $\boldsymbol{Tree}^{\omega}(p)$. Secondly we show that $\boldsymbol{Tree}^{\omega}(p)$ and $\mathfrak{R}_p$ are mutually interpretable. As a result we obtain the following model-theoretic characterisation of ω-automatic structures.

Theorem 4.8. *For every structure* $\mathfrak{A}$, *the following are equivalent*:

(i) $\mathfrak{A}$ *is* ω*-automatic.*
(ii) $\mathfrak{A} \leq_{\mathrm{FO}} \mathfrak{R}_p$ *for some* (*and hence all*) $p \geq 2$.
(iii) $\mathfrak{A} \leq_{\mathrm{FO}} \boldsymbol{Tree}^{\omega}(p)$ *for some* (*and hence all*) $p \geq 2$.

Proof. In order to construct interpretations of $\boldsymbol{Tree}^{\omega}(p)$ in $\mathfrak{R}_p$ and vice versa we define formulae which allow us to access the digits of, respectively, some number in $\mathfrak{R}_p$ and some word in $\boldsymbol{Tree}^{\omega}(p)$. In the latter case we set

$$\mathrm{dig}_k(x, y) := \exists z(\mathrm{el}(z, y) \wedge \sigma_k z \preceq x)$$

which states that the digit of x at position $|y|$ is k. For $\mathfrak{R}_p$ the situation is more complicated as some real numbers admit two encodings. The following formula describes that there is one encoding of x such that the digit at position y is k (this corresponds to the predicate X of [9]):

$$\mathrm{dig}_k(x, y) := \exists s \exists t(|x| = s + k \cdot y + t \;\wedge\; p \cdot y|_p s \;\wedge\; 0 \leq s \;\wedge\; 0 \leq t < y).$$

For $\mathfrak{R}_p \leq_{\mathrm{FO}} \boldsymbol{Tree}^{\omega}(p)$ we represent each number as a pair of words. The first one is finite and encodes the integer part, the other one is infinite and contains the fractional part. In the other direction we map finite words $a_1 \cdots a_r \in [p]^*$ to the interval $[2, 3]$ via

$$p^{-r+1} + \sum_{i=1}^{r} a_i p^{-i} + 2 \in [2, 3].$$

Infinite words $a_1 a_2 \cdots \in [p]^{\omega}$ are mapped to two intervals $[-1, 0]$ and $[0, 1]$ via

$$\pm \sum_i a_i p^{-i} \in [-1, 1].$$

This is necessary because some words, e.g., $0(p-1)^{\omega}$ and 10^{ω}, would be mapped to the same number otherwise. Now the desired interpretations can be constructed easily using the formulae dig_k defined above.

It remains to prove that if $R \subseteq ([p]^{\omega})^n$ is ω-regular, then it is definable in $\boldsymbol{Tree}^{\omega}(p)$. Let $M = (Q, [p]^n, \Delta, q_0, F)$ be a Büchi–automaton for R. Without loss of generality assume $Q = [p]^m$ for some m and $q_0 = (0, \ldots, 0)$. We prove the claim by constructing a formula $\psi_M(\bar{x}) \in \mathrm{FO}$ stating that there is a successful run of M on $x_1 \otimes \ldots \otimes x_n$. The run is encoded by a tuple $(q_1, \dots, q_m) \in ([p]^\omega)^m$ of ω-words such that the symbols of $q_1, \dots, q_m$ at some position equal $k_1, \dots, k_m$ iff the automaton is in state $(k_1, \dots, k_m)$ when scanning the input symbol at that position. $\psi_M(\bar{x})$ has the form

$$\exists q_1 \cdots \exists q_m [\mathrm{ADM}(\bar{q}, \bar{x}) \wedge \mathrm{START}(\bar{q}, \bar{x}) \wedge \mathrm{RUN}(\bar{q}, \bar{x}) \wedge \mathrm{ACC}(\bar{q}, \bar{x})],$$

where the admissibility condition $\mathrm{ADM}(\bar{x}, \bar{q})$ states that all components of $\bar{x}$ and $\bar{q}$ are infinite, $\mathrm{START}(\bar{x}, \bar{q})$ says that the first state is $\bar{0}$, $\mathrm{ACC}(\bar{x}, \bar{q})$ says that some final state appears infinitely often, and $\mathrm{RUN}(\bar{x}, \bar{q})$ ensures that all transitions are correct.

Define the following auxiliary formulae. To access the digits of a tuple of words at some position we define $\mathrm{Sym}_{\bar{a}}(\bar{x}, z) := \bigwedge_i \mathrm{dig}_{a_i}(x_i, z)$, and to characterise the ω-words of $[p]^{\leq\omega}$ we set

$$\mathrm{Inf}(x) := \forall y (x \preceq y \rightarrow x = y).$$

ADM and START are defined as

$$\mathrm{ADM}(\bar{q}, \bar{x}) := \bigwedge_{i=1}^{m} \mathrm{Inf}(q_i) \wedge \bigwedge_{i=1}^{n} \mathrm{Inf}(x_i),$$
$$\mathrm{START}(\bar{q}, \bar{x}) := \mathrm{Sym}_{\bar{0}}(\bar{q}, \varepsilon),$$

RUN states that at every position a valid transition is used,

$$\mathrm{RUN}(\bar{q}, \bar{x}) := \forall z \bigvee_{(\bar{k}, \bar{a}, \bar{k}') \in \Delta} \big(\mathrm{Sym}_{\bar{k}}(\bar{q}, z) \wedge \mathrm{Sym}_{\bar{a}}(\bar{x}, z) \wedge \mathrm{Sym}_{\bar{k}'}(\bar{q}, \sigma_0 z)\big),$$

and ACC says that there is one final state which appears infinitely often in $\bar{q}$:

$$\mathrm{ACC}(\bar{q}, \bar{x}) := \bigvee_{\bar{k} \in F} \forall z \exists z' \big(|z'| > |z| \wedge \mathrm{Sym}_{\bar{k}}(\bar{q}, z')\big). \qquad \square$$

5. Model Checking and Query Evaluation

In this section we study decidability and complexity issues for automatic structures. Two fundamental algorithmic problems are:

Model checking. Given a (presentation of a) structure $\mathfrak{A}$, a formula $\varphi(\bar{x})$, and a tuple of parameters $\bar{a}$ in $\mathfrak{A}$, decide whether $\mathfrak{A} \models \varphi(\bar{a})$.

Query evaluation. Given a presentation of a structure $\mathfrak{A}$ and some formula $\varphi(\bar{x})$, compute a presentation of $(\mathfrak{A}, \varphi^{\mathfrak{A}})$. That is, given automata for the relations of $\mathfrak{A}$, construct an automaton that recognises $\varphi^{\mathfrak{A}}$.

Decidability. We first observe that all first-order queries on (ω-)automatic structures are effectively computable since the construction in Proposition 4.3 is effective. In fact, this is the case not only for first-order logic but also for formulae containing the quantifier $\exists^\omega$ meaning "there are infinitely many". To prove this result for the case of ω-automatic structures we require some preparations.

Lemma 5.1. *Let $R \subseteq \Sigma^\omega \otimes \Gamma^\omega$ be regular. There is a regular relation $R' \subseteq R$ such that*

(i) *if $(x, y) \in R$, then there is some y' such that $(x, y') \in R'$, and*
(ii) *for all x there is at most one y with $(x, y) \in R'$.*

Proof. Without loss of generality, assume that $R \neq \emptyset$. First we introduce some notation. By $v[i, k)$ we denote the factor $v_i \cdots v_{k-1}$ of $v = v_0 v_1 \cdots \in \Sigma^\omega$. Similarly, $v[i, \omega)$ is equal to $v_i v_{i+1} \cdots$, and $v[i] := v[i, i+1)$.

Let $\mathfrak{A} = (Q, \Sigma \times \Gamma, \Delta, q_0, F)$ be a Büchi automaton recognising R. Fix an ordering $\sqsubseteq$ of Q such that all final states are less than non-final ones. For a run (sequence of states) $\varrho \in Q^\omega$ define

$$\begin{aligned}
&\operatorname{Inf}(\varrho) := \{q \in Q \mid \text{there are infinitely many } i \text{ such that } \varrho[i] = q\},\\
&\mu(\varrho) := \min \operatorname{Inf}(\varrho).
\end{aligned}$$

Let $f_i(\varrho)$ be the greatest number such that $\varrho[k, f_i(\varrho))$ contains exactly i times the state $\mu(\varrho)$ where k is the least position such that all states appearing only finitely often are contained in $\varrho[0, k)$.

Denote the lexicographically least run of $\mathfrak{A}$ on $x \otimes y$ by $\varrho(x, y)$. Fix $x \in \Sigma^\omega$ and set $\mu(y) := \mu(\varrho(x, y))$, $f_i(y) := f_i(\varrho(x, y))$. We define an order on Γ^ω by

$$\begin{aligned}
y \leq y' \quad \text{iff} \quad & \mu(y) < \mu(y'),\\
\text{or} \quad & \mu(y) = \mu(y') \quad \text{and there is some } n \text{ such that}\\
& f_n(y) < f_n(y') \quad \text{and} \quad f_i(y) = f_i(y') \quad \text{for all} \quad i < n,\\
\text{or} \quad & \mu(y) = \mu(y'), \quad f_i(y) = f_i(y') \quad \text{for all } i, \text{ and}\\
& y \text{ is lexicographically less than or equal to } y'.
\end{aligned}$$

It should be obvious that the relation $\leq$ is regular. Finally, R' is defined by

$$R' := \{(x, y) \in R \mid \text{there is no } y' < y \text{ such that } (x, y') \in R\}.$$

Clearly, R' is regular, is contained in R, and satisfies (ii). Hence, it remains to prove (i). We directly construct the minimal element of $V := \{y \mid (x, y) \in R\}$ as follows. Let $Y_{-1} \subseteq V$ be the subset of those y with minimal $\mu(y)$. We define a sequence of sets $Y_{-1} \supseteq Y_0 \supseteq Y_1 \supseteq \cdots$ by

$$Y_i := \{\, y \in Y_{i-1} \mid f_i(y) \leq f_i(y') \text{ for all } y' \in Y_{i-1} \,\}.$$

For $i \geq 0$, fix some element $y_i \in Y_i$ such that $y_i[0, f_i)$ is lexicographically minimal. Hence, $y_0 \geq y_1 \geq \cdots$. Define $\hat{y}$ by

$$\hat{y}[n] := \lim_k y_k[n].$$

We claim that $\hat{y}$ is the minimal element of V.

(a) $\hat{y}$ exists. Set $f_n := \lim_k f_n(y_k) = f_n(y_n)$. The pointwise limit of y_k exists since $y_{n+1}[0, f_n) = y_n[0, f_n)$ for all n. For, otherwise, there is some $k < f_n$ such that

$$y_{n+1}[0, k) = y_n[0, k) \quad \text{and} \quad y_{n+1}[k] \neq y_n[k].$$

Since $y_n, y_{n+1} \in Y_n$ it follows that $y_n[k] < y_{n+1}[k]$. On the other hand, $y' := y_n[0, f_n)y_{n+1}[f_n, \omega)$ is in V and thus in Y_{n+1}, as $\varrho(x, y_n)[f_n] = \varrho(x, y_{n+1})[f_n]$. Thus, $y_{n+1}[k] \leq y'[k] = y_n[k]$ by choice of y_{n+1}. Contradiction.

(b) $\hat{y} \in V$. An accepting run of $\mathfrak{A}$ on $x \otimes y$ is given by $\hat{\varrho}$ where

$$\hat{\varrho}[f_{n-1}, f_n) = \varrho(x, y_n)[f_{n-1}, f_n)$$

since $\hat{\varrho}[f_n] = \mu(y_0) \in F$ for all n.

(c) $\hat{y}$ is minimal. Suppose $y' \leq y$ for some $y' \in V$. Then $\mu(y') \leq \mu(\hat{y})$ and, by induction on n, one can show that $y' \in Y_n$ since $f_n(y') \leq f_n(\hat{y})$. Thus by construction $\mu(y') = \mu(\hat{y})$ and $f_n(y') = f_n(\hat{y})$. Suppose $y' < y$. Then y' must be lexicographically less than $\hat{y}$ and there exists some k such that $y'[0, k) = \hat{y}[0, k)$ and $y'[k] < \hat{y}[k]$. Choose n such that $f_{n-1} \leq k < f_n$. Then $y_n \leq y'$ by construction. However, $y_n[0, f_n) = \hat{y}[0, f_n)$ and hence $y_n[0, k) = y'[0, k)$ which implies that $y'[k] \geq y_n[k] = \hat{y}[k]$. Contradiction. □

Proposition 5.2. *Every* (ω-)*automatic structure has an injective presentation.*

Proof. For automatic structures this result is due to Khoussainov and Nerode [36].

Let $\nu : D \to A$ be a presentation of an ω-automatic structure $\mathfrak{A}$. By the preceding lemma applied to the relation $\{(x, y) | \nu x = \nu y\}$ there is a function e such that

(i) $\nu x = \nu e x$ for all $x \in \Sigma^\omega$, and
(ii) $\nu x = \nu y$ implies $ex = ey$.

Thus we obtain a regular subset $D' \subseteq D$ containing exactly one representation for each element of $\mathfrak{A}$ by defining $D' := \{x \in D | ex = x\}$. □

We say that a logic L *effectively collapses* to $L_0 \subseteq L$ on a structure $\mathfrak{A}$ if, given a formula $\varphi(\bar{x}) \in L$, one can compute a formula $\varphi_0(\bar{x}) \in L_0$ such that $\varphi_0^{\mathfrak{A}} = \varphi^{\mathfrak{A}}$.

Proposition 5.3.

(1) FO($\exists^\omega$) *effectively collapses to* FO *on* $\boldsymbol{Tree}(p)$.
(2) FO($\exists^\omega$) *effectively collapses to* FO *on* $\boldsymbol{Tree}^\omega(p)$.

Proof. (1) In the case of automatic structures the quantifier $\exists^\omega$ can be handled using a pumping argument. Consider for simplicity the formula $\exists^\omega x \psi(x, y)$. By induction the formula ψ is equivalent to some first-order formula and, hence, there is some automaton recognising the relation defined by ψ. There are infinitely many x satisfying ψ iff for any m there are infinitely many such elements whose encoding is at least m symbols longer than that of y. If we take m to be the number of states of the automaton for ψ

then, by the Pumping Lemma, the last condition is equivalent to the existence of at least one such x. Thus $\exists^\omega x\psi(x, y) \equiv \exists x(\psi(x, y) \wedge$ "x is long enough") for which we can obviously construct an automaton.

(2) For ω-automatic structures the proof is more involved.

Let M be a deterministic Muller automaton with s states recognising the language $L(M) \subseteq \Gamma^\omega \otimes \Sigma^\omega$. For $w \in \Gamma^\omega$ let $V(w) := \{v \in \Sigma^\omega | w \otimes v \in L(M)\}$.

Let $v, w \in \Sigma^\omega$ and define $v \approx^* w$ iff $v[n, \omega) = w[n, \omega)$ for some n. Let $[v]_* := \{v' \in V(w) | v' \approx^* v\}$ be the $\approx^*$-class of v in $V(w)$.

Claim. *$V(w)$ is infinite if and only if there is some $v \in \Sigma^\omega$ such that $[v]_* \in V(w)/{\approx^*}$ is infinite.*

Proof. ($\Leftarrow$) is trivial and ($\Rightarrow$) is proved by showing that $V/{\approx^*}$ contains at most s finite $\approx^*$-classes.

Assume there are words $v_0, \dots, v_s \in V(w)$ belonging to different finite $\approx^*$-classes. Denote the run of M on $w \otimes v_i$ by ϱ_i. Define $I_{ij} := \{k < \omega | \varrho_i[k] = \varrho_j[k]\}$. Since there are only s states, for each $k < \omega$ there have to be indices i, j such that $k \in I_{ij}$, i.e., $\bigcup_{i,j} I_{ij} = \omega$. Thus, at least one I_{ij} is infinite. For each $[v_i]_*$ there is a position n_i such that $v[n_i, \omega) = v'[n_i, \omega)$ for all $v, v' \in [v_i]_*$. Let m be the maximum of $n_0, \dots, n_s$. Fix i, j such that I_{ij} is infinite. Since $v_i \not\approx^* v_j$ there is a position $m' > m$ such that $v_i[m, m') \neq v_j[m, m')$. Choose some $m'' \in I_{ij}$ with $m'' \geq m'$. Let $u := v_i[0, m)v_j[m, m'')v_i[m'', \omega)$. Then $w \otimes v_i \in L(M)$ iff $w \otimes u \in L(M)$ which implies that $u \in [v_i]_*$. However, $u[m, \omega) \neq v_i[m, \omega)$ in contradiction to the choice of m. □

To finish the proof let $\varphi(\bar{x}) := \exists^\omega y\psi(\bar{x}, y)$ and $\mathfrak{A}$ be ω-automatic. One can express that $[v]_*$ is finite by

$$\text{finite}(\bar{x}, v) := \exists n \forall v'[\psi(\bar{x}, v') \wedge v \approx^* v' \rightarrow \text{equal}(v, v', n)],$$

where

$$\text{equal}(v, v', n) := n = 1^i 0^\omega \ \wedge\ v[i, \omega) = v'[i, \omega).$$

Clearly, $\approx^*$ and equal can be recognised by ω-automata. By the claim above,

$$\varphi(\bar{x}) \equiv \exists v(\psi(\bar{x}, v) \wedge \neg\text{finite}(\bar{x}, v)).$$

Hence, we can construct an automaton recognising $\varphi^{\mathfrak{A}}$. □

We are now ready to prove Theorem 2.1, saying that the model checking problem for FO($\exists^\omega$) is decidable on ω-automatic structures.

Proof of Theorem 2.1. Given a presentation of an ω-automatic structure $\mathfrak{A}$, and a formula $\varphi \in$ FO($\exists^\omega$) we compute an injective interpretation $\mathcal{I} : \mathfrak{A} \leq_{\text{FO}} \textbf{\textit{Tree}}^\omega(p)$. Since FO($\exists^\omega$) is closed under *injective* interpretations, we have that $\varphi^{\mathcal{I}} \in$ FO($\exists^\omega$) and $\mathfrak{A} \models \varphi$

iff $\boldsymbol{Tree}^{\omega}(p) \models \varphi^{\mathcal{I}}$. By Proposition 5.3 we can effectively compute a formula $\psi \in \mathrm{FO}$ that is equivalent to $\varphi^{\mathcal{I}}$ on $\boldsymbol{Tree}^{\omega}(p)$. Since the first-order theory of any ω-automatic structure is decidable, the result follows. $\square$

Corollary 5.4. *The* FO($\exists^{\omega}$) *theory of any* (ω)*-automatic structure is decidable.*

As an immediate consequence we conclude that full arithmetic $(\mathbb{N}, +, \cdot)$ is neither automatic nor ω-automatic. For most of the common extensions of first-order logic used in finite model theory, such as transitive closure logics, fixed point logics, MSO logic, or first-order logic with counting, the model-checking problem on automatic structures becomes undecidable.

Complexity. The complexity of model checking can be measured in three different ways. First, one can fix the formula and ask how the complexity depends on the input structure. This measure is called *structure complexity*. The *expression complexity* on the other hand is defined relative to a fixed structure in terms of the formula. Finally, one can look at the *combined complexity* where both parts may vary.

Of course, the complexity of these problems may very much depend on how automatic structures are presented. Since the decision methods for $\mathfrak{N}_p$ and $\boldsymbol{Tree}(p)$ are automaton based, a presentation $\mathfrak{d}$ consisting of a list of automata, is more suitable for practical purposes than using an interpretation. Here, we focus on presentations by *deterministic* automata because these admit boolean operations to be performed in polynomial time, whereas for non-deterministic automata, complementation may cause an exponential blow-up.

In the following we always assume that the vocabulary of the given automatic structures and the alphabet of the automata we deal with are fixed. Furthermore, the vocabulary is assumed to be relational when not stated otherwise. For a (deterministic) presentation $\mathfrak{d}$ of an automatic structure $\mathfrak{A}$, we denote by $|\mathfrak{d}|$ the maximal size of the automata in $\mathfrak{d}$, and, fixing some surjective function $\nu : L(M_\delta) \to A$, we define $\lambda : A \to \mathbb{N}$ to be the function

$$\lambda(a) := \min\{|x| \mid \nu(x) = a\}$$

mapping each element of $\mathfrak{A}$ to the length of its shortest encoding. Finally, let $\lambda(a_1, \ldots, a_r)$ be an abbreviation for $\max\{\lambda(a_i) \mid i = 1, \ldots, r\}$.

While we have seen above that query evaluation and model checking for first-order formulae are effective on AutStr, the complexity of these problems is non-elementary, i.e., it exceeds any fixed number of iterations of the exponential function. This follows immediately from the fact that the complexity of $\mathrm{Th}(\mathfrak{N}_p)$ is non-elementary (see [26]).

Proposition 5.5. *There exist automatic structures such that the expression complexity of the model-checking problem is non-elementary.*

It turns out that model checking and query evaluation for quantifier-free and existential formulae are still—to some extent—tractable. As usual, let Σ_0 and Σ_1 denote, respectively, the class of quantifier-free and the class of existential first-order formulae.

	Structure complexity	Expression complexity
Model checking		
Σ_0	LOGSPACE-complete	ALOGTIME-complete
$\Sigma_0 +$ fun	NLOGSPACE	PTIME-complete
Σ_1	NPTIME-complete	PSPACE-complete
Query Evaluation		
Σ_0	LOGSPACE	PSPACE
Σ_1	PSPACE	EXPSPACE

Theorem 5.6.

(i) *Given a presentation* $\mathfrak{d}$ *of a relational structure* $\mathfrak{A} \in$ AutStr, *a tuple* $\bar{a}$ *in* $\mathfrak{A}$, *and a quantifier-free formula* $\varphi(\bar{x}) \in$ FO, *the model-checking problem for* $(\mathfrak{A}, \bar{a}, \varphi)$ *is in*

$$\mathrm{DTIME}\big[\mathcal{O}\big(|\varphi|\lambda(\bar{a})|d|\log|d|\big)\big] \quad \textit{and}$$
$$\mathrm{DSPACE}\big[\mathcal{O}\big(\log|\varphi| + \log|d| + \log\lambda(\bar{a})\big)\big].$$

(ii) *The structure complexity of model checking for quantifier-free formulae is* LOGSPACE-*complete with respect to first-order reductions.*

(iii) *The expression complexity is* ALOGTIME-*complete with regard to deterministic log-time reductions.*

Proof. (i) To decide whether $\mathfrak{A} \models \varphi(\bar{a})$ holds, we need to know the truth value of each atom appearing in φ. Then all what remains is to evaluate a boolean formula which can be done in $\mathrm{DTIME}\big[\mathcal{O}\big(|\varphi|\big)\big]$ and $\mathrm{ATIME}\big[\mathcal{O}\big(\log|\varphi|\big)\big] \subseteq \mathrm{DSPACE}\big[\mathcal{O}\big(\log|\varphi|\big)\big]$ (see [11]). The value of an atom $R\bar{x}$ can be calculated by simulating the corresponding automaton on those components of $\bar{a}$ which belong to the variables appearing in $\bar{x}$. The naïve algorithm to do so uses time $\mathcal{O}\big(\lambda(\bar{a})|\mathfrak{d}|\log|\mathfrak{d}|\big)$ and space $\mathcal{O}\big(\log|\mathfrak{d}| + \log\lambda(\bar{a})\big)$.

For the time-complexity bound we perform this simulation for every atom, store the outcome, and evaluate the formula. Since there are at most $|\varphi|$ atoms the claim follows.

To obtain the space bound we cannot store the value of each atom. Therefore we use the LOGSPACE-algorithm to evaluate φ and, every time the value of an atom is needed, we simulate the run of the corresponding automaton on a separate set of tapes.

(ii) We present a reduction of the LOGSPACE-complete problem DETREACH, reachability by deterministic paths (see, e.g., [34]), to the model-checking problem. Given a graph $\mathfrak{G} = (V, E, s, t)$ we construct the automaton $M = (V, \{0\}, \Delta, s, \{t\})$ with

$$\Delta := \{(u, 0, v) | u \neq t,\ (u, v) \in E \text{ and there is no } v' \neq v \text{ with } (u, v') \in E\}$$
$$\cup \{(t, 0, t)\}.$$

That is, we remove all edges originating at vertices with out-degree greater than 1 and add a loop at t. Then there is a deterministic path from s to t in $\mathfrak{G}$ iff M accepts some word 0^n iff $0^{|V|} \in L(M)$. Thus,

$$(V, E, s, t) \in \mathrm{DETREACH} \qquad \text{iff} \quad \mathfrak{A} \models P0^{|V|},$$

where $\mathfrak{A} = (B, P)$ is the structure with the presentation $(\{0\}^*, L(M))$. A closer inspection reveals that the above transformation can be defined in first-order logic.

(iii) Evaluation of boolean formulae is ALOGTIME-complete (see [11]). □

For most questions we can restrict attention to relational vocabularies and replace functions by their graphs at the expense of introducing additional quantifiers. When studying quantifier-free formulae we will not want to do this and hence need to consider the case of quantifier-free formulae with function symbols separately. This class is denoted $\Sigma_0 + \text{fun}$.

Lemma 5.7. *Given a tuple $\overline{w}$ of words over Σ, and an automaton $\mathfrak{A} = (Q, \Sigma, \delta, q_0, F)$ recognising the graph of a function f, the calculation of $f(\overline{w})$ is in*

$$\text{DTIME}\left[\mathcal{O}(|Q^2|\log|Q|(|Q| + |\overline{w}|))\right] \quad \textit{and} \quad \text{DSPACE}\left[\mathcal{O}(|Q|\log|Q| + \log|\overline{w}|)\right].$$

Proof. The following algorithm simulates $\mathfrak{A}$ on input $w_0 \otimes \ldots \otimes w_{n-1} \otimes x$ where x is the result that we want to calculate. For every position i of the input, the set Q_i of states which can be reached for various values of x is determined. At the same time the sets Q_i and Q_{i+1} are connected by edges E_i labelled by the symbol of x by which the second state could be reached. When a final state is found, x can be read off the graph.

We use the function Step$(Q, \overline{a})$ depicted in Figure 1 to compute Q_{i+1} and E_i from Q_i and the input symbol $\overline{a}$. If E is realised as an array containing, for every $q \in Q$, the values q' and c such that $(q', c, q) \in E$, then this function needs space $\mathcal{O}(|Q|\log|Q|)$ and time

$$\mathcal{O}\big(|Q|(|Q|\log|Q| + |Q|\log|Q|)\big) = \mathcal{O}\big(|Q^2|\log|Q|\big).$$

Two slightly different algorithms are used to obtain the time- and space-complexity bounds (see Figure 2). The first one simply computes all sets Q_i and E_i and determines x. The second one reuses space and keeps only one set Q_i and E_i in memory. Therefore

```
Step($Q, \overline{a}$)
  $Q' := \emptyset$
  $E := \emptyset$
  **forall** $q \in Q$
    **forall** $c \in \Sigma$
      $q' := \delta(q, \overline{a}c)$
      **if** $q' \notin Q'$ **then**
        $E := E \cup \{(q, c, q')\}$
        $Q' := Q' \cup \{q'\}$
      **end**
    **end**
  **return** $(Q', E)$
```

Fig. 1. Computing Q_{i+1} and E_i from Q_i.

```
Input: $\mathfrak{A} = (Q, \Sigma, \delta, q_0, F), \overline{w}$
$Q_0 := \{q_0\}$
$i := 0$
**while** $Q_i \cap F = \emptyset$
  **if** $i < |\overline{w}|$ **then**
    $\overline{a} := \overline{w}[i]$
  **else**
    $\overline{a} := \overline{\Box}$
  $(Q_{i+1}, E_i) := \text{Step}(Q_i, \overline{a})$
  $i := i + 1$
**end**
let $q \in Q_i \cap F$
**while** $i > 0$
  $i := i - 1$
  let $(q', c, q) \in E_i$
  $x[i] := c$
  $q := q'$
**end**
**return** $x$
```

```
Input: $\mathfrak{A} = (Q, \Sigma, \delta, q_0, F), \overline{w}$
$Q := \{q_0\}$
$i := 0$
**while** $Q \cap F = \emptyset$
  **if** $i < |\overline{w}|$ **then**
    $\overline{a} := \overline{w}[i]$
  **else**
    $\overline{a} := \overline{\Box}$
  $(Q, E) := \text{Step}(Q, \overline{a})$
  $i := i + 1$
**end**
let $q \in Q \cap F$
**while** $i > 0$
  $i := i - 1$
  $Q := \{q_0\}$
  **for** $k = 0, \ldots, i - 1$
  **if** $k < |\overline{w}|$ **then**
    $\overline{a} := \overline{w}[k]$
  **else**
    $\overline{a} := \overline{\Box}$
  $(Q, E) := \text{Step}(Q, \overline{a})$
  **end**
  let $(q', c, q) \in E$
  $x[i] := c$
  $q := q'$
**end**
**return** $x$
```

Fig. 2. Two algorithms to compute $f(\overline{w})$.

it has to start the computation from the beginning in order to access old values of E_i in the second part.

In the first version the function Step is invoked $|x|$ times, and the second part is executed in time $\mathcal{O}\big(|x||Q| \log|Q|\big)$.

The space needed by the second version consists of storage for Q, E, and the counters i and k. Hence, $\mathcal{O}(|Q| + |Q| \log|Q| + \log|x|)$ bits are used.

Since $\mathfrak{A}$ recognises a function, the length of x can be at most $|Q| + |\overline{w}|$ (see Proposition 6.1 for a detailed proof). This yields the given bounds. $\Box$

Theorem 5.8.

(i) *Let τ be a vocabulary which may contain functions. Given the presentation $\mathfrak{d}$ of a structure $\mathfrak{A}$ in* AutStr$[\tau]$, *a tuple $\bar{a}$ in $\mathfrak{A}$, and a quantifier-free formula $\varphi(\bar{x}) \in$* FO$[\tau]$, *the model-checking problem for $(\mathfrak{A}, \bar{a}, \varphi)$ is in*

$$\text{DTIME}\big[\mathcal{O}\big(|\varphi||\mathfrak{d}^2| \log|\mathfrak{d}|\big(|\varphi||\mathfrak{d}| + \lambda(\bar{a})\big)\big)\big] \quad \textit{and}$$
$$\text{DSPACE}\big[\mathcal{O}\big(|\varphi|\big(|\varphi||\mathfrak{d}| + \lambda(\bar{a})\big) + |\mathfrak{d}| \log|\mathfrak{d}|\big)\big].$$

(ii) *The structure complexity of the model-checking problem for quantifier-free formulae with functions is in* NLOGSPACE.

(iii) *The expression complexity is* PTIME*-complete with regard to* $\leq_{\mathrm{m}}^{\log}$*-reductions.*

Proof. (i) Our algorithm proceeds in two steps. First the values of all functions appearing in φ are calculated starting with the innermost one. Then all functions can be replaced by their values and a formula containing only relations remains which can be evaluated as above. We need to evaluate at most $|\varphi|$ functions. If they are nested the result can be of length $|\varphi||\mathfrak{d}| + \lambda(\bar{a})$. This yields the bounds given above.

(ii) It is sufficient to present a non-deterministic log-space algorithm for evaluating a single fixed atom containing functions. The algorithm simultaneously simulates the automata of the relation and of all functions on the given input. Components of the input corresponding to values of functions are guessed non-deterministically. Each simulation only needs counters for the current state and the input position which both use logarithmic space.

(iii) Let M be a $p(n)$ time-bounded deterministic Turing machine for some polynomial p. A configuration (q, w, p) of M can be coded as word $w_0 q w_1$ with $w = w_0 w_1$ and $|w_0| = p$. Using this encoding both the function f mapping one configuration to its successor and the predicate P for configurations containing accepting states can be recognised by automata. We assume that $f(c) = c$ for accepting configurations c. Let q_0 be the starting state of M. Then M accepts some word w if and only if the configuration $f^{p(|w|)}(q_0 w)$ is accepting if and only if $\mathfrak{A} \models P f^{p(|w|)}(q_0 w)$ where $\mathfrak{A} = (A, P, f)$ is automatic. Hence, the mapping taking w to the pair $q_0 w$ and $P f^{p(|w|)} x$ is the desired reduction which can clearly be computed in logarithmic space. $\square$

Theorem 5.8 says that, on any fixed automatic structure, quantifier-free formulae can be evaluated in quadratic time. This extends a well-known result on automatic groups [22].

Corollary 5.9. *The word problem for every automatic group is solvable in quadratic time.*

Proof. Let G be an automatic group with semigroup generators $s_1, \ldots, s_m$. We present the Cayley graph of G in a functional way, by the structure $(G, e, g \mapsto gs_1, \ldots, g \mapsto gs_m)$ which is therefore automatic. Each instance of the word problem is described by a quantifier-free sentence (a term equation) on this structure, which can be checked in quadratic time by Theorem 5.8. $\square$

Theorem 5.10.

(i) *Given a presentation* $\mathfrak{d}$ *of a structure* $\mathfrak{A}$ *in* AutStr, *a tuple* $\bar{a}$ *in* $\mathfrak{A}$, *and a formula* $\varphi(\bar{x}) \in \Sigma_1$, *the model-checking problem for* $(\mathfrak{A}, \bar{a}, \varphi)$ *is in*

$$\mathrm{NTIME}\big[\mathcal{O}\big(|\varphi||\mathfrak{d}|\lambda(\bar{a}) + |\mathfrak{d}^{\mathcal{O}(|\varphi|)}\big)\big] \quad \textit{and}$$
$$\mathrm{NSPACE}\big[\mathcal{O}\big(|\varphi|(|\mathfrak{d}| + \log|\varphi|) + \log\lambda(\bar{a})\big)\big].$$

(ii) *The structure complexity of model checking for Σ_1-formulae is* NPTIME-*complete with respect to $\leq_{\mathrm{tt}}^{\mathrm{p}}$-reductions.*

(iii) *The expression complexity is* PSPACE-*complete with regard to $\leq_{\mathrm{m}}^{\log}$-reductions.*

Proof. (i) As above we can run the corresponding automaton for every atom appearing in φ on the encoding of $\bar{a}$. However, now there are some elements of the input missing which we have to guess. Since we have to ensure that the guessed inputs are the same for all automata, the simulation is performed simultaneously.

The algorithm determines which atoms appear in φ and simulates the product automaton constructed from the automata for those relations. At each step the symbol for the quantified variables is guessed non-deterministically. Note that the values of those variables may be longer than the input so we have to continue the simulation after reaching its end for at most the cardinality of the state-space number of steps. Since this cardinality is $\mathcal{O}\big(|\mathfrak{d}|^{|\varphi|}\big)$ a closer inspection of the algorithm yields the given bounds.

(ii) We reduce the NPTIME-complete non-universality problem for non-deterministic automata over a unary alphabet (see [40] and [33]), given such an automaton, check whether it does not recognise the language 0^*, to the given problem. This reduction is performed in two steps. First the automaton must be simplified and transformed into a deterministic one, then we construct an automatic structure and a formula $\varphi(x)$ such that $\varphi(a)$ holds for several values of a if and only if the original automaton recognises 0^*. As model checking has to be performed for more than one parameter this yields not a many-to-one but a truth-table reduction.

Let $M = (Q, \{0\}, \Delta, q_0, F)$ be a non-deterministic finite automaton over the alphabet $\{0\}$. We construct an automaton M' such that there are at most two transitions outgoing at every state. This is done be replacing all transitions from some given state by a binary tree of transitions with new states as internal nodes. Of course, this changes the language of the automaton. Since in M every state has at most $|Q|$ successors, we can take trees of fixed height $k := \lceil \log|Q| \rceil$. Thus, $L(M') = h(L(M))$ where h is the homomorphism taking 0 to 0^k. Note that the size of M' is polynomial in that of M.

M' still is non-deterministic. To make it deterministic we add a second component to the labels of each transition which is either 0 or 1. This yields an automaton M'' such that M accepts the word 0^n iff there is some $y \in \{0, 1\}^{kn}$ such that M'' accepts $0^{kn} \otimes y$.

M'' can be used in a presentation $\mathfrak{d} := (\{0, 1\}^*, L(M''))$ of some $\{R\}$-structure $\mathfrak{B}$. Then

$$\mathfrak{B} \models \exists y\, R0^{kn} y \qquad \text{iff} \quad 0^{kn} \otimes y \in L(M'') \quad \text{iff} \quad 0^n \in L(M).$$

It follows that

$$L(M) = 0^* \qquad \text{iff} \quad \mathfrak{B} \models \exists y\, R0^{kn} y \quad \text{for all} \quad n < 2|Q|.$$

The part ($\Rightarrow$) is trivial. To show ($\Leftarrow$) let n be the least number such that $0^n \notin L(M)$. By assumption $n \geq 2|Q|$. However, then we can apply the Pumping Lemma and find some number $n' < n$ with $0^{n'} \notin L(M)$. Contradiction.

(iii) Let M be a $p(n)$ space-bounded Turing machine for some polynomial p. We encode configurations as words appending enough spaces to increase their length to $p(n)+1$. Let $L_\vdash := \{ c_0 \otimes c_1 | c_0 \vdash c_1 \}$ be the transition relation of M. The run of M on input w is encoded as sequence of configurations separated by some marker #. $L_\vdash$ can be used to check whether some word x represents a run of M. Let y be the suffix of x obtained by removing the first configuration. The word $x \otimes y$ has the form

$$\begin{matrix} c_0 & \# & c_1 & \# & \multirow{2}{*}{\cdots} & \# & c_{s-1} & \# & c_s \\ c_1 & \# & c_2 & \# & & \# & c_s & \# & \end{matrix} .$$

Thus x encodes a valid run iff $x \otimes y \in L_T$ where

$$L_T := \left(L_\vdash \begin{matrix} \# \\ \# \end{matrix} \right)^* (\Sigma^* \otimes \varepsilon).$$

Clearly, the language L_F of all runs whose last configuration is accepting is regular. Finally, we need two additional relations. Both the prefix relation $\preceq$ and the shift s are regular where $s(ax) := x$ for $a \in \Sigma$ and $x \in \Sigma^*$. Therefore, the structure $\mathfrak{A} := (A, T, F, s, \preceq)$ is automatic, and it should be clear that

$$w \in L(M) \qquad \text{iff} \quad \mathfrak{A} \models \varphi_w\big(q_0 w \Box^{k-|w|}\#\big),$$

where $k := p(|w|)$ and

$$\varphi_w(x) := \exists y_0 \cdots \exists y_{k+1} \left(\bigwedge_{i \le k} s y_i = y_{i+1} \wedge x \preceq y_0 \wedge T y_0 y_{k+1} \wedge F y_0 \right).$$

$\varphi_w(x)$ states that there is an accepting run y_0 of M starting with configuration x. $y_1, \dots, y_{k+1}$ are used to remove the first configuration from y_0, so we can use T to check whether y_0 is valid.

Clearly, the mapping of w to φ_w and $q_0 w \Box^{k-|w|}\#$ can be computed in logarithmic space. □

We now turn to the query-evaluation problem for these formula classes.

Theorem 5.11. *Given a presentation $\mathfrak{d}$ of a structure $\mathfrak{A}$ in* AutStr *and a formula $\varphi(\bar{x})$, an automaton representing $\varphi^{\mathfrak{A}}$ can be computed*

(i) *in time $\mathcal{O}\big(|\mathfrak{d}^{\mathcal{O}(|\varphi|)}\big)$ and space $\mathcal{O}\big(|\varphi| \log|\mathfrak{d}|\big)$ in the case of quantifier-free $\varphi(\bar{x})$, and*
(ii) *in time $\mathcal{O}\big(2^{|\mathfrak{d}^{\mathcal{O}(|\varphi|)}}\big)$ and space $\mathcal{O}\big(|\mathfrak{d}^{\mathcal{O}(|\varphi|)}\big)$ in the case of existential formulae $\varphi(\bar{x})$.*

In particular, the structure complexity of query evaluation is in LOGSPACE *for quantifier-free formulae and in* PSPACE *for existential formulae. The expression complexity is in* PSPACE *for quantifier-free formulae and in* EXPSPACE *for existential formulae.*

Proof. Enumerate the state-space of the product automaton and output the transition function. □

Undecidability. In the remainder of this section we present some undecidability results for automatic structures. When we say that a property P of automatic structures is undecidable, we mean that there is no algorithm that, given an automatic presentation of a structure, decides whether the structure has property P.

Lemma 5.12. *The configuration graph of any Turing machine is automatic.*

Proof. We encode a configuration of a Turing machine M with state q, tape contents w, and head position p by the word w_0qw_1 where $w = w_0w_1$ and $|w_0| = p$. At every transition $w_0qw_1 \vdash_M w_0'q'w_1'$ only the symbols around the state are changed. This can be checked by an automaton. □

Corollary 5.13. REACHABILITY *is undecidable for automatic structures.*

This is an immediate consequence of Lemma 5.12 and the undecidability of the halting problem. For further results, we make use of a normal form for Turing machines.

Lemma 5.14. *For any deterministic* 1*-tape Turing machine* M, *one can effectively construct a deterministic* 2*-tape Turing machine* M' *such that the configuration graph of* M' *consists of a disjoint union of*

(a) *a countably infinite number of infinite acyclic paths with a first but without a last element*, *and*
(b) *for each word* $x \in L(M)$, *one path starting at an initial configuration and ending in a loop.*

Proof. It is well known that any Turing machine M can be translated into an equivalent reversible Turing machine M' (see [4]). We slightly modify this construction. While simulating M on its first tape the machine M' appends to the second tape the transitions performed at each step. Hence, the storage content of M' at each step is a pair (y, z) where y describes a configuration of M, and z is a sequence of transitions of M. Further, we define M' such that if M terminates without accepting, then M' diverges, i.e., its computation is infinite and not periodic (for instance, it moves the head to the right at every step). Thus, if $x \notin L(M)$, then the run of M' on input x consists of an infinite path of type (a).

The construction as presented so far does not suffice since there may exist configurations that cannot be reached from any initial configuration. We have to ensure that every path containing such a configuration is of type (a). Clearly, every such path must have a first element since z never decreases. We modify M' such that whenever M reaches an accepting configuration, M' reverses its computation, using the transitions stored on the second tape, but without changing the content of the second tape. That is, from configurations with storage content (y, z) where y is accepting for M, M' will either reach a configuration with storage content (y_0, z) where y_0 is an initial configuration of M and z represents the computation of M from y_0 to z, or M' will detect that no such y_0 exists.

In the second case, M' diverges as above. In the first case, M' enters a cycle as follows: it restarts the simulation of M from y_0 (leaving z unchanged and checking at every step that the sequence of transitions on the second tape is correct). When the accepting configuration (y, z) is reached again, the computation is again reversed completing the cycle.

Note that with this modification the run of M' on inputs x with $x \in L(M)$ is of type (b). Further, the nodes of the configuration graphs with two incoming edges (the starting points of cycles) are precisely the accepting configurations that are reachable from an initial configuration. □

Theorem 5.15. *It is undecidable whether two automatic structures are isomorphic.*

Proof. Let M be a deterministic 1-tape Turing machine, and let M' be the associated Turing machine as described in the preceding lemma. Given M, we can effectively construct an automatic presentation of the configuration graph G of M'. Further, we construct an automatic presentation of the graph H consisting of $\aleph_0$ copies of $(\omega, \ suc)$. Then $G \cong H$ iff $L(M) = \emptyset$. Since the emptiness problem for Turing machines is undecidable, so is the isomorphism problem for automatic structures. □

We recall that a directed graph is called connected if its underlying undirected graph is. Further, it is called strongly connected if every node is reachable from every other node via a directed path.

Theorem 5.16. *It is undecidable whether an automatic graph is* (i) *connected or* (ii) *strongly connected.*

Proof. Given a Turing machine M, we again construct an automatic presentation of the configuration graph G of the associated machine M' from Lemma 5.14. Let I be the set of initial configurations of M'; obviously, I is automatic. Further, let X_2 be the set of nodes of G with two predecessors and let X_0 be the set of nodes without predecessors. Finally, let G' be the graph obtained from G by connecting all nodes from $X_2 \cup (X_0 - I)$ with each other. Since X_2 and X_0 are first-order definable it follows that the resulting graph G' is still automatic. Further, G' is connected if all components of G correspond to accepting computations of M or to computations from unreachable configurations. Hence, G' is connected iff M accepts all inputs, which is undecidable.

Finally, note that the inverse of the graph G'' obtained from G' by adding to each edge is also first-order definable from G' and is strongly connected if and only if G' is connected. Hence, strong connectedness of automatic graphs is also undecidable. □

6. Structures that Are Not Automatic

To prove that a structure is automatic, we just have to find a suitable presentation. However, how can we prove that a structure is not automatic? The main difficulty is that a priori nothing is known about how elements of an automatic structure are named by words of the regular language.[1]

Besides the two obvious criteria, namely, that automatic structures are countable and that their first-order theory is decidable, not much is known. The only non-trivial criterion that is available at present for general structures uses growth rates for the length of the encodings of elements of definable sets. For recent progress on the classification of automatic linear orders see [18] and [37].

Proposition 6.1 [21]. *Let $\mathfrak{A}$ be an automatic structure with injective presentation $(\nu, \mathfrak{d})$, and let $f : A^n \to A$ be a function of $\mathfrak{A}$. Then there is a constant m such that $\lambda(f(\bar{a})) \leq \lambda(\bar{a}) + m$ for all $\bar{a} \in A^n$.*

The same is true if we replace f by a relation R where for all $\bar{a}$ there are only finitely many values b such that $R\bar{a}b$ holds.

This result deals with a single application of a function or relation. In the remaining part of this section we study the effect of applying functions iteratively, i.e., we consider some definable subset of the universe and calculate upper bounds on the length of the encodings of elements in the substructure generated by it. First we need bounds for the (encodings of) elements of some definable subsets. The following lemma follows easily from classical results in automata theory (see, e.g., Proposition V.1.1 of [20]).

Lemma 6.2. *Let $\mathfrak{A}$ be a structure in* AutStr *with presentation $\mathfrak{d}$, and let B be an* FO($\exists^\omega$)*-definable subset of A. Then $\lambda(B)$ is a finite union of arithmetical progressions.*

In the process of generating a substructure we have to count the number of applications of functions.

Definition 6.3. Let $\mathfrak{A} \in$ AutStr with presentation $\mathfrak{d}$, let $R_1, \dots, R_s$ be finitely many relations of $\mathfrak{A}$ with arities $r_1 + 1, \dots, r_s + 1$, respectively, and let $E = \{e_1, e_2, \dots\}$ be some subset of A with $\lambda(e_1) \leq \lambda(e_2) \leq \cdots$. Then $G_n(E)$, the nth *generation* of E, is defined inductively by

$$G_1(E) := \{e_1\},$$
$$G_n(E) := \{e_n\} \cup G_{n-1}(E) \cup \{b \mid (\bar{a}, b) \in R_i,\ \bar{a} \in G_{n-1}^{r_i}(E),\ 1 \leq i \leq s\}.$$

Putting everything together we obtain the following result. The case of finitely generated substructures already appeared in [36].

Proposition 6.4. *Let $\mathfrak{d}$ an injective presentation of an automatic structure $\mathfrak{A}$, let $f_1, \dots, f_r$ be finitely many definable operations on $\mathfrak{A}$, and let E be a definable subset of A. Then there is a constant m such that $\lambda(a) \leq mn$ for all $a \in G_n(E)$. In particular, $|G_n(E)| \leq |\Sigma^{mn+1}|$ where Σ is the alphabet of $\mathfrak{d}$.*

[1] In the case of automatic groups, where the naming function is fixed, more techniques are available such as the k-fellow traveller property, see [22].

The proof consists of a simple induction on n.

Theorem 6.5. *None of the following structures has an automatic presentation.*

(i) *Any trace monoid* $\mathfrak{M} = (M, \cdot)$ *with at least two non-commuting generators* a *and* b.
(ii) *Any structure* $\mathfrak{A}$ *in which a pairing function* f *can be defined.*
(iii) *The divisibility poset* $(\mathbb{N}, |)$.
(iv) *Skolem arithmetic* $(\mathbb{N}, \cdot)$.

Proof. (i) We show that $\{a, b\}^{\leq 2^n} \subseteq G_{n+1}(a, b)$ by induction on n. We have $\{a, b\} \subseteq \{a, aa, b\} = G_2(a, b)$ for $n = 1$, and, for $n > 1$,

$$\begin{aligned} G_{n+1}(a, b) &= \{uv \mid u, v \in G_n(a, b)\} \\ &\supseteq \{uv \mid u, v \in \{a, b\}^{\leq 2^{n-1}}\} \\ &= \{a, b\}^{\leq 2^n}. \end{aligned}$$

Therefore, $|G_n(a, b)| \geq 2^{2^n}$ and the claim follows.

(ii) is analogous to (i), and (iv) immediately follows from (iii) as the divisibility relation is definable in $(\mathbb{N}, \cdot)$.

(iii) Suppose $(\mathbb{N}, |) \in$ AutStr. We define the set of primes

$$Px \qquad \text{iff} \quad x \neq 1 \wedge \forall y(y|x \rightarrow y = 1 \vee y = x),$$

the set of powers of some prime

$$Qx \qquad \text{iff} \quad \exists y(Py \wedge \forall z(z|x \wedge z \neq 1 \rightarrow y|z)),$$

and a relation containing all pairs (n, pn), where p is a prime divisor of n,

$$Sxy \qquad \text{iff} \quad x|y \wedge \exists^{=1} z(Qz \wedge \neg Pz \wedge z|y \wedge \neg z|x).$$

The least common multiple of two numbers is

$$\operatorname{lcm}(x, y) = z \qquad \text{iff} \quad x|z \wedge y|z \wedge \neg\exists u(u \neq z \wedge x|u \wedge y|u \wedge u|z).$$

For every $n \in \mathbb{N}$ there are only finitely many m with Snm. Therefore S satisfies the conditions of Proposition 6.1. Consider the set generated by P via S and lcm, and let $\gamma(n) := |G_n(P)|$ be the cardinality of $G_n(P)$. If $(\mathbb{N}, |)$ is in AutStr then $(\mathbb{N}, |, P, Q, S) \in$ AutStr, and $\gamma(n) \in 2^{\mathcal{O}(n)}$ by Proposition 6.4. Let $P = \{p_1, p_2, \ldots\}$. For $n = 1$ we have $G_1(P) = \{p_1\}$. Generally, $G_n(P)$ consists of

(1) numbers of the form $p_1^{k_1}$,
(2) numbers of the form $p_2^{k_2} \cdots p_n^{k_n}$, and
(3) numbers of a mixed form.

In n steps we can create

(1) $p_1, \dots, p_1^n$ (via S),
(2) $\gamma(n-1)$ numbers with $k_1 = 0$, and
(3) for every $0 < k_1 < n$, $\gamma(n-2) - 1$ numbers of a mixed form (via lcm).

All in all we obtain

$$\begin{aligned}
\gamma(n) &\geq n + \gamma(n-1) + (n-1)(\gamma(n-2) - 1) \\
&= \gamma(n-1) + (n-1)\gamma(n-2) + 1 \\
&\geq n\gamma(n-2) \qquad (\text{as } \gamma(n-1) > \gamma(n-2)) \\
&\geq n(n-2)\cdots 3\gamma(1) \qquad (\text{without loss of generality assume that } n \text{ is odd}) \\
&= n(n-2)\cdots 3 \\
&\geq ((n+1)/2)! \\
&\in 2^{\Omega(n \log n)}.
\end{aligned}$$

Contradiction. □

Remark. (1) Since it is easy to construct a *tree-automatic* presentation of Skolem arithmetic this result implies that the class of structures with tree-automatic presentation strictly includes the class of automatic structures (see [6]).

(2) The structure $(\mathbb{N}, \perp)$, where $\perp$ stands for having no common divisor, is automatic. To see this, we represent each number $n \in \mathbb{N}$ by a pair (w, k) where $w = w_0 w_1 \cdots \in \{0,1\}^*$ such that $w_i = 1$ iff the ith prime divides n, and k is the number of elements $m < n$ with the same set of prime divisors as n. Then $(w, k) \perp (w', k')$ iff w and w' represent disjoint sets which can obviously be checked by an automaton.

7. Composition of Structures

The composition method developed by Feferman and Vaught [25] and by Shelah [45] (see also [30] and [47]) considers compositions (products and sums) of structures according to some index structure and allows one to compute—depending on the type of composition—the first-order or MSO theory of the whole structure from the respective theories of its components and the monadic theory of the index structure.

The characterisation given in Section 4 can be used to prove closure of automatic structures under such compositions of finitely many structures. A generalised product—as it is defined below—is a generalisation of a direct product, a disjoint union, and an ordered sum. We will prove that given a finite sequence $(\mathfrak{A}_i)_i$ of structures first-order interpretable in some structure $\mathfrak{C}$, all their generalised products are also first-order interpretable in $\mathfrak{C}$.

The definition of such a product is a bit technical. Its relations are defined in terms of the types of the components of its elements. The *atomic n-type* $\operatorname{atp}_{\mathfrak{A}}(\bar{a})$ of a tuple $(a_0, \dots, a_{n-1})$ in a structure $\mathfrak{A}$ is the conjunction of all atomic and negated atomic formulae $\varphi(\bar{x})$ such that $\varphi(\bar{a})$ holds in $\mathfrak{A}$.

We first look at how a direct product and an ordered sum can be defined using types.

Example. (1) Let $\mathfrak{A} := \mathfrak{A}_0 \times \mathfrak{A}_1$ where $\mathfrak{A}_i = (A_i, R_i)$, for $i \in \{0, 1\}$, and R is a binary relation. The universe of $\mathfrak{A}$ is $A_0 \times A_1$. Some pair $(\bar{a}, \bar{b})$ belongs to R iff $(a_0, b_0) \in R_0$ and $(a_1, b_1) \in R_1$. This is equivalent to the condition that the atomic types of $a_0 b_0$ and of $a_1 b_1$ both include the formula Rx_0x_1.

(2) Let $\mathfrak{A} := \mathfrak{A}_0 + \mathfrak{A}_1$ where $\mathfrak{A}_i = (A_i, <_i)$, for $i \in \{0, 1\}$, and $<_0$, $<_1$ are partial orders. The universe of $\mathfrak{A}$ is $A_0 \dot{\cup} A_1 \cong A_0 \times \{\diamond\} \cup \{\diamond\} \times A_1$, and we have

$$\begin{aligned}
\bar{a} < \bar{b} \quad &\text{iff} \quad \bar{a} = (a_0, \diamond), \quad \bar{b} = (b_0, \diamond), \quad \text{and} \quad a_0 <_0 b_0,\\
&\text{or} \quad \bar{a} = (\diamond, a_1), \quad \bar{b} = (\diamond, b_1), \quad \text{and} \quad a_1 <_1 b_1,\\
&\text{or} \quad \bar{a} = (a_0, \diamond), \quad \bar{b} = (\diamond, b_1).
\end{aligned}$$

Again, the condition $a_i <_i b_i$ can be expressed using types.

Definition 7.1. Let $\tau = \{R_0, \dots, R_s\}$ be a finite relational vocabulary, let r_j be the arity of R_j, and let $\hat{r} := \max\{r_0, \dots, r_s\}$. Let $(\mathfrak{A}_i)_{i \in I}$ be a sequence of τ-structures, and let $\mathfrak{I}$ be an arbitrary relational σ-structure with universe I.

Fix for each $k \le \hat{r}$ an enumeration $\{t_0^k, \dots, t_{n(k)}^k\}$ of the atomic k-types and set

$$\sigma_k := \sigma \mathbin{\dot{\cup}} \{D_0, \dots, D_{k-1}\} \mathbin{\dot{\cup}} \{T_l^m \mid m \le k, l \le n(m)\}.$$

The σ_k-expansion $\mathfrak{I}(\bar{b})$ of $\mathfrak{I}$ belonging to a sequence $\bar{b} \in \left(\prod_{i \in I}(A_i \dot{\cup} \{\diamond\})\right)^k$ is given by

$$\begin{aligned}
D_l^{\mathfrak{I}}(\bar{b}) &:= \{i \in I \mid (b_l)_i \neq \diamond\},\\
(T_l^m)^{\mathfrak{I}(\bar{b})} &:= \{i \in I \mid \operatorname{atp}_{\mathfrak{A}}((b_{j_0})_i \cdots (b_{j_{m-1}})_i) = t_l^m \text{ and}\\
&\qquad \{j \mid (b_j)_i \neq \diamond\} = \{j_0, \dots, j_{m-1}\}\}.
\end{aligned}$$

For $D \subseteq \mathbb{B}^I$ and $\beta_j \in \mathrm{FO}[\sigma_{r_j}]$, $\mathcal{C} := (\mathfrak{I}, D, \beta_0, \dots, \beta_s)$ defines the *generalised product* $\mathcal{C}(\mathfrak{A}_i)_{i \in I} := (A, R_0, \dots, R_s)$ of $(\mathfrak{A}_i)_{i \in I}$ where

$$\begin{aligned}
A &:= \bigcup_{\bar{d} \in D} \prod_{i \in I} \chi_{d_i}\left(\{\diamond\}, A_i\right),\\
R_i &:= \{\bar{b} \in A^{r_i} \mid \mathfrak{I}(\bar{b}) \models \beta_i\},
\end{aligned}$$

and $\chi_b(a_0, a_1) := a_b$.

Example (*continued*).

(1) For the direct product of $\mathfrak{A}_0 \times \mathfrak{A}_1$ we would set $\mathfrak{I} := (I)$ with $I = \{0, 1\}$, $D := \{(1, 1)\}$, and

$$\beta := \bigvee_{l \in L} T_l^2 0 \wedge \bigvee_{l \in L} T_l^2 1,$$

where L is the set of atomic types containing the formula Rx_0x_1.

(2) In this case we would set $\mathfrak{I} := (I)$ with $I = \{0, 1\}$, $D := \{(1, 0), (0, 1)\}$, and

$$\beta := \left(D_0 0 \wedge D_1 0 \wedge \bigvee_{l \in L} T_l^2 0 \right) \vee \left(D_0 1 \wedge D_1 1 \wedge \bigvee_{l \in L} T_l^2 1 \right) \vee (D_0 0 \wedge D_1 1),$$

where L is the set of atomic types containing the formula $x_0 < x_1$.

Theorem 7.2. *Let τ be a finite relational vocabulary, let $\mathcal{K}$ be a class of τ-structures containing all finite τ-structures, and let there be a structure $\mathfrak{C}$ such that $\mathcal{K} \subseteq \{\mathfrak{A} | \mathfrak{A} \leq_{\mathrm{FO}} \mathfrak{C}\}$. Let $\mathfrak{I}$ be a finite relational σ-structure, let $(\mathfrak{A}_i)_{i \in I}$ be a sequence of structures in $\mathcal{K}$, and let $\mathcal{C} = (\mathfrak{I}, D, \bar{\beta})$ be a generalised product. Then $\mathcal{C}(\mathfrak{A}_i)_{i \in I} \in \mathcal{K}$, and an interpretation $\mathcal{C}(\mathfrak{A}_i)_{i \in I} \leq_{\mathrm{FO}} \mathfrak{C}$ can be constructed effectively from the interpretations $\mathfrak{A}_i \leq_{\mathrm{FO}} \mathfrak{C}$ and $\mathfrak{I} \leq_{\mathrm{FO}} \mathfrak{C}$.*

Proof. Let $\tau = \{R_0, \dots, R_s\}$. Without loss of generality assume that $I = \{0, \dots, |I| - 1\}$ and that $\mathfrak{C}$ contains constants 0 and 1. We have to construct an interpretation of $\mathfrak{A} := \mathcal{C}(\mathfrak{A}_i)_{i \in I}$ in $\mathfrak{C}$. Let r_j be the arity of R_j. Consider n_i-dimensional interpretations

$$\mathcal{I}^i := \left\langle h^i, \delta^i(\bar{x}^i), \varepsilon^i(\bar{x}^i, \bar{y}^i), \varphi_0^i(\bar{x}_0^i, \dots, \bar{x}_{r_0-1}^i), \dots, \varphi_s^i(\bar{x}_0^i, \dots, \bar{x}_{r_s-1}^i) \right\rangle$$

of $\mathfrak{A}_i$ in $\mathfrak{C}$. We represent an element a of $\mathfrak{A}$ by a tuple of $(|I| + n_0 + \dots + n_{|I|-1})$ elements

$$\bar{x} := \left(\bar{d}, \bar{x}^0, \dots, \bar{x}^{|I|-1} \right),$$

where $\bar{d} \in D$ determines which components are empty and $\bar{x}^i$ encodes the ith component of a. The desired interpretation is constructed as follows:

$$\mathcal{I} := \left\langle h, \delta(\bar{x}), \varepsilon(\bar{x}, \bar{y}), \varphi_0(\bar{x}_0, \dots, \bar{x}_{r_0-1}), \dots, \varphi_s(\bar{x}_0, \dots, \bar{x}_{r_s-1}) \right\rangle,$$

where

$$h(\bar{d}, \bar{x}^0, \dots, \bar{x}^{|I|-1}) := \left(\chi_{d_0}\left(\diamond, h^0(\bar{x}^0) \right), \dots, \chi_{d_{|I|-1}}\left(\diamond, h^{|I|-1}(\bar{x}^{|I|-1}) \right) \right),$$
$$\delta(\bar{d}, \bar{x}^0, \dots, \bar{x}^{|I|-1}) := \bigvee_{\bar{c} \in D} \left(\bar{d} = \bar{c} \wedge \bigwedge_{i : c_i = 1} \delta^i(\bar{x}^i) \right),$$

and

$$\varepsilon(\bar{d}, \bar{x}^0, \dots, \bar{x}^{|I|-1}, \bar{e}, \bar{y}^0, \dots, \bar{y}^{|I|-1}) := \bar{d} = \bar{e} \wedge \bigwedge_{i < |I|} \left(d_i = 1 \rightarrow \varepsilon^i(\bar{x}^i, \bar{y}^i) \right).$$

In order to define φ_j we consider an interpretation $\mathcal{I}^I$ of $\mathfrak{I}$ in $\mathfrak{C}$. Since $\mathfrak{I}$ is finite such an interpretation exists. Let $\alpha_j := \beta_j^{\mathcal{I}^I}$ be the formula defining R_j. Note that β_j contains additional relations D_l and T_l^m which are not in σ. Thus α_j is a sentence over the vocabulary τ extended by the symbols D_l and T_l^m for appropriate l and m. We have to replace them in order to obtain a definition of φ_j. Let $\bar{x}_0, \dots, \bar{x}_{r_j-1}$ be the parameters of φ_j where

$$\bar{x}_k = (\bar{d}_k, \bar{x}_k^0, \dots, \bar{x}_k^{|I|-1})$$

for $k < r_j$. D_l and T_l^m can be defined by

$$D_l i := (d_l)_i = 1 \quad \text{and} \quad T_l^m i := (t_l^m)^{\mathcal{I}^i}(\bar{x}_0^i, \dots, \bar{x}_{r_j-1}^i).$$

Note that those definitions are only valid because i ranges over a finite set. φ_j can now be defined as α_j with D_l and T_l^m replaced by the above definitions.

Obviously, all steps in the construction above are effective. □

Corollary 7.3. *Both* AutStr *and* ω-AutStr *are effectively closed under finitary generalised products.*

As promised we immediately obtain closure under several types of compositions.

Corollary 7.4. *Let* $\mathfrak{A}_0, \dots, \mathfrak{A}_{n-1} \in$ AutStr. *Then there exists automatic presentations of*

(i) *the direct product* $\prod_{i<n} \mathfrak{A}_i$,
(ii) *the disjoint union* $\dot{\bigcup}_{i<n} \mathfrak{A}_i$, *and*
(iii) *the* ω*-fold disjoint union* $\omega \cdot \mathfrak{A}_0$ *of* $\mathfrak{A}_0$.

Corollary 7.5. *Let* $\mathfrak{A}_0, \dots, \mathfrak{A}_{n-1} \in$ AutStr *be ordered structures. There exist automatic presentations of*

(i) *the ordered sum* $\sum_{i<n} \mathfrak{A}_i$ *and*
(ii) *the* ω*-fold ordered sum* $\sum_{i<\omega} \mathfrak{A}_0$ *of* $\mathfrak{A}_0$.

8. Conclusion

We have seen different methods of how to represent infinite structures in a finite way effectively. Among the most interesting classes of finitely presentable structures are automatic and ω-automatic structures. Besides the original presentations based on finite automata these classes also admit characterisations in terms of first-order interpretations into expansions of Presburger arithmetic or the additive group of reals. We have shown that these representations allow for effective model checking and algorithms for operations like unions, products, and the evaluation of queries. We have also given complexity bounds for some of these problems. Finally, we have started to investigate algebraic properties of automatic structures. In particular, we have developed methods to prove that certain structures are not automatic.

Most of the techniques that we have used can be generalised to other classes of infinite structures. For instance, we can modify the automaton model to obtain tree-automatic or rational structures. The approach based on interpretations is even more general. Fixing any structure $\mathfrak{A}$ with good algorithmic or model-theoretic properties, we can consider the class of all structures interpretable in $\mathfrak{A}$ (for any suitable notion of interpretation). Since many relevant properties of structures are preserved by interpretations, every structure in such a class inherits them from $\mathfrak{A}$. This is a very broad and versatile approach that encompasses many of the classes of finitely presentable structures appearing in the literature.

So far we have mostly studied rather powerful classes that contain as many interesting structures as possible, while preserving decidability of first-order or MSO logic. An obvious drawback of this generality is complexity. For instance, even rather simple formulae are quite difficult to evaluate on automatic structures or tree-interpretable structures. For future studies it might be interesting to search for classes for which the algorithmic problems are somewhat more manageable. For classes defined by interpretations one might choose structures with relatively modest complexity and notions of interpretations that do not blow up the complexity too much. We believe that this is a promising line of future research that may also be relevant for practical applications.

References

[1] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*, Addison-Wesley, Reading, MA, 1995.

[2] C. Ash and J. Knight, *Computable Structures and the Hyperarithmetical Hierarchy*, Elsevier, Amsterdam, 2000.

[3] K. Barthelmann, On equational simple graphs, Tech. Rep. 9, Institut für Informatik, Universität Mainz, 1997.

[4] C. Bennett, Logical reversibility of computation, *IBM Journal of Research and Development*, **17** (1973), 525–532.

[5] D. Berwanger and A. Blumensath, The monadic theory of tree-like structures, in *Automata, Logic, and Infinite Games* (E. Grädel, W. Thomas, and T. Wilke, eds.), Springer-Verlag, Berlin, 2002.

[6] A. Blumensath, Automatic structures, Diplomarbeit, RWTH Aachen, 1999.

[7] A. Blumensath, Prefix-recognisable graphs and monadic second-order logic, Tech. Rep. AIB-06-2001, RWTH Aachen, 2001.

[8] A. Blumensath and E. Grädel, Automatic structures, in *Proceedings of the* 15*th IEEE Symposium on Logic in Computer Science*, 2000, pp. 51–62.

[9] B. Boigelot, S. Rassart, and P. Wolper, On the expressiveness of real and integer arithmetic automata, in *Proceedings of the* 25*th International Colloquium on Automata, Languages and Programming, ICALP* 98, Lecture Notes in Computer Science, vol. 1443, 1998, Springer-Verlag, Berlin, pp. 152–163.

[10] V. Bruyère, G. Hansel, C. Michaux, and R. Villemaire, Logic and p-recognizable sets of integers, *Bulletin of the Belgion Mathematical Society*, **1** (1994), 191–238.

[11] S. Buss, The boolean formula value problem is in ALOGTIME, in *Proceedings* 19*th ACM Symposium on the Theory of Computing*, 1987, pp. 123–131.

[12] A. Carayol and S. Wöhrle, The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata, *Proceedings of FSTTCS*, Lecture Notes in Computer Science, vol. 2914, Springer-Verlag, Berlin, 2003, pp. 112–123.

[13] D. Caucal, On infinite transition graphs having a decidable monadic theory, in *Automata, Languages and Programming,* 23*rd International Colloquium, ICALP* 96, Lecture Notes in Computer Science, vol. 1099, Springer-Verlag, Berlin, 1996, pp. 194–205.

[14] D. Caucal, On infinite terms having a decidable monadic theory, in *Proceedings of the* 27*th International Symposium on Mathematical Foundations of Computer Science, MFCS* 02, Lecture Notes in Computer Science, vol. 2420, Springer-Verlag, Berlin, 2002, pp. 165–176.

[15] B. Courcelle, The monadic second-order logic of graphs, II: Infinite graphs of bounded width, *Mathematical System Theory*, **21** (1989), 187–221.

[16] B. Courcelle, The monadic second-order logic of graphs, IX: Machines and their behaviours, *Theoretical Computer Science*, **151** (1995), 125–162.

[17] B. Courcelle and I. Walukiewicz, Monadic second-order logic, graph coverings and unfoldings of transition systems, *Annals of Pure and Applied Logic*, **92** (1998), 35–62.

[18] C. Delhomme, V. Goranko, and T. Knapik, Automatic linear orderings, Submitted.

[19] H.-D. Ebbinghaus and J. Flum, *Finite Model Theory*, Springer-Verlag, Berlin, 1995.

[20] S. Eilenberg, *Automata*, *Languages*, *and Machines*, vol. A, Academic Press, New York, 1974.

[21] C. C. Elgot and J. E. Mezei, On relations defined by generalized finite automata, *IBM Journal of Research Development*, **9** (1965), 47–68.

[22] D. Epstein, J. Cannon, D. Holt, S. Levy, M. Paterson, and W. Thurston, *Word Processing in Groups*, Jones and Bartlett, Boston, MA, 1992.

[23] Y. L. Ershov, S. S. Goncharov, A. Nerode, and J. B. Remmel, *Handbook of Recursive Mathematics*, North-Holland, Amsterdam, 1998.

[24] B. Farb, Automatic groups: a guided tour, *L' Enseignement Mathématique*, **38** (1992), 291–313.

[25] S. Feferman and R. L. Vaught, The first order properties of products of algebraic systems, *Fundamenta Mathematicae*, **XLVII** (1959), 57–103.

[26] E. Grädel, Simple interpretations among complicated theories, *Information Processing Letters*, **35** (1990), 235–238.

[27] E. Grädel and Y. Gurevich, Metafinite model theory, *Information and Computation*, **140** (1998), 26–81.

[28] E. Grädel and A. Malmström, 0–1 laws for recursive structures, *Archive of Mathematical Logic*, **38** (1999), 205–215.

[29] E. Grädel and K. Meer, Descriptive complexity theory over the real numbers, in *Mathematics of Numerical Analysis*: *Real Number Algorithms* (J. Renegar, M. Shub, and S. Smale, eds.), Lectures in Applied Mathematics, vol. 32, AMS, Providence, RI, 1996, pp. 381–403.

[30] Y. Gurevich, Monadic second-order theories, in *Model-Theoretic Logics* (J. Barwise and S. Feferman, eds.), Springer-Verlag, Berlin, 1985, pp. 479–506.

[31] D. Harel, Towards a theory of recursive structures, in *Proceedings of the* 23*rd International Symposium on Mathematical Foundations of Computer Science*, *MFCS* 98, of Lecture Notes in Computer Science, vol. 1450, Springer-Verlag, Berlin, 1998, pp. 36–53.

[32] T. Hirst and D. Harel, More about recursive structures: descriptive complexity and zero–one laws, in *Proceedings of the* 11*th IEEE Symposium on Logic in Computer Science*, 1996, pp. 334–348.

[33] H. Hunt, III, D. Rosenkrantz, and T. Szymanski, On the equivalence, containment, and covering problems for the regular and context-free languages, *Journal of Computer and System Sciences*, **12** (1976), 222–268.

[34] N. Immerman, *Descriptive Complexity*, Graduate Texts in Computer Science, Springer-Verlag, New York, 1998.

[35] P. Kanellakis, G. Kuper, and P. Revesz, Constraint query languages, *Journal of Computer and Systems Sciences*, **51** (1995), 26–52. (An extended abstract appeared in the *Proceedings of PODS* '90).

[36] B. Khoussainov and A. Nerode, Automatic presentations of structures, in *Logic and Computational Complexity*, Lecture Notes in Computer Science vol. 960, Springer-Verlag, Berlin, 1995, pp. 367–392.

[37] B. Khoussainov, S. Rubin, and F. Stephan, On automatic partial orders, in *Proceedings of the* 18*th Annual IEEE Symposium on Logic in Computer Science*, *LICS*, 2003, pp. 168–177.

[38] G. Kuper, L. Libkin, and J. Paredaens, eds., *Constraint Databases*, Springer-Verlag, Berlin, 2000.

[39] C. Löding, Model-checking infinite systems generated by ground tree rewriting, in *Proceedings of Foundations of Software Science and Computation Structures*, *FoSSaCS* 2002, Lecture Notes in Computer Science, vol. 2303, Springer-Verlag, Berlin, 2002, pp. 280–294.

[40] A. R. Meyer and L. J. Stockmeyer, Word problems requiring exponential time, in *Proceedings of the* 5*th ACM Symposium on the Theory of Computing*, 1973, pp. 1–9.

[41] C. Morvan, On rational graphs, in *Proceedings of FOSSACS* 2000, Lecture Notes in Computer Science, vol. 1784, Springer-Verlag, Berlin, 2000, pp. 252–266.

[42] D. Muller and P. Schupp, Groups, the theory of ends, and context-free languages, *Journal of Computer and System Sciences*, **26** (1983), 295–310.

[43] D. Muller and P. Schupp, The theory of ends, pushdown automata, and second-order logic, *Theoretical Computer Science*, **37** (1985), 51–75.

[44] A. L. Semenov, *Decidability of Monadic Theories*, Lecture Notes in Computer Science vol. 176, Springer-Verlag, Berlin, 1984, pp. 162–175.

[45] S. Shelah, The monadic theory of order, *Annals of Mathematics*, **102** (1975), 379–419.

[46] A. Stolboushkin, Towards recursive model theory, in *Logic Colloquium* 95 (J. Makowsky and E. Ravve, eds.), Lecture Notes in Logic, vol. 11, Springer-Verlag, Berlin, 1998, pp. 325–338.

[47] W. Thomas, Ehrenfeucht games, the composition method, and the monadic theory of ordinal words, in *Structures in Logic and Computer Science. A Selection of Essays in Honor of Andrzej Ehrenfeucht* (G. Rozenberg, A. Salomaa, and J. Mycielski, eds.), Lecture Notes in Computer Science, vol.1261, Springer-Verlag, Berlin, 1997, pp. 118–143.

[48] W. Thomas, Constructing infinite graphs with a decidable MSO-theory, in *Proceedings of the* 28*th International Symposium on Mathematical Foundations of Computer Science*, *MFCS* 03, Lecture Notes in Computer Science, vol. 2747, Springer-Verlag, Berlin, 2003, pp. 113–124.

[49] I. Walukiewicz, Monadic second-order logic on tree-like structures, *Theoretical Computer Science*, **275** (2001), 311–346.

Received October 2, 2002. *Online publication September* 13, 2004.