Abstract

We study definability and complexity issues for automatic and ω-automatic structures. These are, in general, infinite structures but they can be finitely presented by a collection of automata. Moreover, they admit effective (in fact automatic) evaluation of all first-order queries. Therefore, automatic structures provide an interesting framework for extending many algorithmic and logical methods from finite structures to infinite ones.

We explain the notion of (ω-)automatic structures, give examples, and discuss the relationship to automatic groups. We determine the complexity of model checking and query evaluation on automatic structures for fragments of first-order logic. Further, we study closure properties and definability issues on automatic structures and present a technique for proving that a structure is not automatic. We give model-theoretic characterisations for automatic structures via interpretations. Finally we discuss the composition theory of automatic structures and prove that they are closed under finitary Feferman-Vaught-like products.

1. Introduction

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. [6, 18]) 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.

From a more general theoretical point of view, one may ask what classes of infinite structures are suitable for such an extension. More specifically what conditions must be satisfied by a class $K$ 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 $\mathfrak{A} \in K$ should be representable in a finite way.

Effective semantics (for a relevant logic $L$, e.g., first-order logic). Given any formula $\psi(\bar{x})$ of $L$ and (a presentation of) a structure $\mathfrak{A} \in K$, one can effectively produce a presentation of the set $\{ \bar{a} : \mathfrak{A} \models \psi(\bar{a}) \}$.

Note that effective semantics means in particular that the $L$-theory of every $\mathfrak{A} \in K$ is decidable. A class of infinite structures that have been studied quite intensively in model theory are recursive structures. There have recently been some papers proposing the study of recursive structures (e.g., recursive databases) for the issues just mentioned [14, 15, 22]. However, the class of recursive structures is too large since, in general, only the quantifier-free formulae admit effective evaluation algorithms. Other classes of infinite structures where the relationship of definability and complexity has been studied include metafinite structures [12] and constraint databases [20].

In this paper we consider automatic structures. While automatic groups have been studied rather intensively in computational group theory (see [9, 10]) a general notion of automatic structure has only been defined and investigated in a paper by Khoussainov and Nerode [19], and the theory of these structures is not well-developed yet. Informally, a relational structure $\mathfrak{A} = (A, R_1, \ldots, R_m)$ is automatic if its universe and its relations can be presented by finite automata. This means that we can find a regular language $L_\delta \subseteq \Sigma^*$ (which provides names for the elements of $\mathfrak{A}$) and a function $\nu : L_\delta \rightarrow A$ mapping every word $w \in L_\delta$ to the element of $\mathfrak{A}$ that it represents. The function $\nu$ must be surjective (every element of $\mathfrak{A}$ must be named) but need not be injective (elements can have more than one name). In addi-
tion it must be recognisable by finite automata whether two 
words in $L_\delta$ name the same elements, and, for each relation 
$R_i$ of $\mathfrak{A}$, whether a tuple of words in $L_\delta$ names a tuple 
belonging to $R_i$. A more detailed definition will be given in 
the next section.

We believe that automatic structures are very promising 
for the approach sketched above. Not only do automatic 
structures admit finite presentations, there also are numer-
umerous interesting examples and a large body of methods 
that has been developed in five decades of automata theory. 
Further, contrary to the class of recursive structures, automatic 
structures admit effective (in fact, automatic) evaluation of 
all first-order queries and possess many other pleasant algo-

rithmic properties.

The notion of an automatic structure can be modified 
and generalised in many directions, for instance by using au-
tomata over infinite words, or over finite or infinite trees. In 
this paper we study automatic and $\omega$-automatic structures 
only. Many results can be extended to tree-automatic struc-
tures without much change (see [2]), but for lack of space, 
we do not mention them here.

Here is an outline of this paper. In Section 2 we de-

fine the notions of automatic and $\omega$-automatic structures 
and mention some examples. For the purposes of this pa-

per, our most important examples of automatic structures 
are the expansions $\mathfrak{N}_p = \langle \mathbb{N}, + \rangle_p$ of Presburger arithmetic 
by a restricted divisibility predicate and tree structures $\text{Tree}(p)$. Our fundamental examples of $\omega$-automatic structures are $\mathfrak{N}_p$, the expansion of the additive real group $\langle \mathbb{R}, + \rangle$ by order and restricted divisibility, and $\text{Tree}^\omega(p)$, a natural extension of $\text{Tree}(p)$. We will also explain in Sec-
tion 2 the notion of an automatic group.

In Section 3 we show that first-order logic (and in fact its 
extension by the quantifier “there exist infinitely many”) has 
effective semantics on ($\omega$-automatic structures. We also 
study complexity results for model-checking and query-
evaluation for first-order logic and for some of its frag-

ments.

In Section 4 we study definability properties of auto-
matic structures and present a technique for proving that 
a structure is not automatic. As an application we prove, 
for instance, that neither Skolem arithmetic $\langle \mathbb{N}, \cdot \rangle$ nor the 
divisibility poset $\langle \mathbb{N}, \mid \rangle$ are automatic.

In Section 5 we present model-theoretic characterisations 
of automatic and $\omega$-automatic structures. We prove that 
a structure is automatic if and only if it is interpretable in 
$\mathfrak{N}_p$, or, equivalently, in $\text{Tree}(p)$ for some (and hence all) 
p $\geq 2$. Similarly, a structure is $\omega$-automatic if and only if it 
is interpretable in $\mathfrak{N}_p$ or $\text{Tree}^\omega(p)$.

Finally in Section 6 this characterisation is used to study 
the composition theory of automatic structures. We prove that 
automatic structures are closed under finitary products, 
unions, and similar constructions.

The goal of this paper is not to make significant new con-

tributions to automata theory. The main technical contribu-
tions of this paper are (1) an algorithm to evaluate the 
quantifier “there exists infinitely many”, (2) the complexity 
results for low level fragments of first-order logic on auto-
matic structures, (3) the proofs that certain interesting struc-
tures are not automatic and in particular, (4) the composition 
theorem for automatic structures. But the main purpose of 
this paper is conceptual: we want to explore to what extent 
automatic structures are a suitable framework for extending 
the methods of finite model theory to infinite structures. We 
believe that the model-theoretic characterizations of auto-
matic and $\omega$-automatic structures in terms of interpretability 
are particularly useful for this and also suggest a very gen-

eral way for obtaining other interesting classes of infinite 
structures suitable for such an approach: Fix a structure $\mathfrak{A}$ 
with ‘nice’ (algorithmic and/or model-theoretic) properties, 
and consider the class of all structures that are (first-order) 
interpretable in $\mathfrak{A}$. Obviously each structure in this class 
is finitely presentable (by an interpretation). Further, since 
many ‘nice’ properties are preserved by first-order interpre-
tations, 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 $\mathfrak{A}$.

2. Automatic structures and automatic groups

We assume that the reader is familiar with the basic 
notions of automata theory and regular languages. One 
slightly nonstandard aspect is that we need a notion of reg-

ularity not just for languages $L \subseteq \Sigma^*$ but also $k$-ary rela-
tions of words, for $k > 1$. The idea is that regular relations 
are defined by automata that take tuples $\bar{w} = (w_1, \ldots, w_k)$ 
of words as inputs and work synchronously on all $k$ com-
ponents of $\bar{w}$. To make this precise, we represent a tuple 
$\bar{w} \in (\Sigma^*)^k$ by a word $w_1 \otimes \cdots \otimes w_k$ over the alphabet 
$(\Sigma \cup \{\square\})^k$, called the convolution of $w_1, \ldots, w_k$. Here $\square$ 
is a padding symbol not belonging to $\Sigma$, which is appended 
to some of the words $w_i$ to make sure that all components 
have the same length. More formally, for $w_1, \ldots, w_k \in \Sigma^*$, 
with $w_i = w_{i1} \cdots w_{i\ell_i}$ and $\ell = \max\{|w_1|, \ldots, |w_k|\}$,

$$w_1 \otimes \cdots \otimes w_k := \begin{bmatrix} w_{i1} \\
\vdots \\\n w_{i\ell_i} \end{bmatrix} \ldots \begin{bmatrix} w_{j1} \\
\vdots \\\n w_{j\ell_j} \end{bmatrix} \in ((\Sigma \cup \{\square\})^k)^*$$

where $w_{ij} = w_j$ for $j \leq |w_i|$ and $w_{i\ell_i} = \square$ otherwise.

Now, a relation $R \subseteq (\Sigma^*)^k$ is called automatic or reg-

ular, if $\{w_1 \otimes \cdots \otimes w_k : (w_1, \ldots, w_k) \in R\}$ is a regular 
language. In the sequel we do not distinguish between a 
relation on words and its encoding as a language.

As usual in mathematical logic, we consider structures 
$\mathfrak{A} = (A, R_1, R_2, \ldots, f_1, f_2, \ldots)$ 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 each $f_j : A^{r_j} \rightarrow A$ 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 $f : A^r \rightarrow A$ by its graph $G_f := \{ (a, b) \in A^{r+1} : f(a) = b \}$.

Definition 2.1. A relational structure $\mathfrak{A}$ is automatic if there exist a regular language $L_\mathfrak{A} \subseteq \Sigma^*$ and a surjective function $\nu : L_\mathfrak{A} \rightarrow A$ such that the relation

$$L_\mathfrak{A} := \{ (w, w') \in L_\mathfrak{A} \times L_\mathfrak{A} : \nu w = \nu w' \} \subseteq \Sigma^* \times \Sigma^*$$

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

$$L_R := \{ \bar{w} \in (L_\mathfrak{A})^r : (\nu w_1, \ldots, \nu w_r) \in R \} \subseteq (\Sigma^*)^r$$

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

By an automatic presentation of a $\tau$-structure $\mathfrak{A}$ we either mean a pair $(\nu, \mathfrak{d})$ consisting of the function $\nu : L_\mathfrak{A} \rightarrow A$ and a collection $\mathfrak{d} = (M_\mathfrak{A}, M_\mathfrak{L}, (M_R)_{R \in \mathfrak{R}})$ of finite automata that recognise $L_\mathfrak{A}$, $L_\mathfrak{L}$, and $L_R$ for all relations $R$ of $\mathfrak{A}$, or we mean just the collection $\mathfrak{d}$ alone (Note that $\mathfrak{d}$ determines the structure that it presents up to isomorphism.) An automatic presentation is called deterministic if all its automata are, and it is called injective if $L_\mathfrak{A} = \{ (u, u) : u \in L_\mathfrak{A} \}$ (which implies that $\nu : L_\mathfrak{A} \rightarrow A$ is injective). We write $\text{AutStr}[\tau]$ for the class of all automatic structures of vocabulary $\tau$.

Examples. (1) All finite structures are automatic.

(2) Important examples of automatic structures are Presburger arithmetic $\langle \mathbb{N}, + \rangle$ and its expansions $\mathfrak{N}_p := \langle \mathbb{N}, +, \lceil p \rceil \rangle$ by the relation

$$x \mid p \iff x \text{ is a power of } p \text{ dividing } y.$$ Using $p$-ary encodings (starting with the least significant digit) it is not difficult to construct automata recognizing equality, addition and $\lceil p \rceil$.

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

$$\sigma_a(x) := xa,$$

$$x \leq y : \iff \exists z(xz = y),$$

$$\text{el}(x, y) : \iff |x| = |y|.$$ 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 $\{0, 1\}$ [2].

(2) Every automatic structure admits an injective automatic presentation [19].

Automatic Groups. The class of automatic structures that have been studied most intensively are automatic groups. Let $(G, \cdot)$ be a group and $S = \{ s_1, \ldots, s_m \} \subseteq G$ a set of semigroup generators of $G$. This means that each $g \in G$ can be written as a product $s_1 \cdots s_r$ of elements of $S$ and hence the canonical homomorphism $\nu : S^* \rightarrow G$ is surjective. The Cayley graph $\Gamma(G, S)$ of $G$ with respect to $S$ is the graph $(G, S_1, \ldots, S_m)$ 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 $L_\mathfrak{A} \subseteq S^*$ such that the restriction of $\nu$ to $L_\mathfrak{A}$ is surjective and provides an automatic presentation of $\Gamma(G, S)$.

(That is, the inverse image of equality,

$$L_\mathfrak{A} = \{ (w, w') \in L_\mathfrak{A} \times L_\mathfrak{A} : \nu w = \nu w' \},$$

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

Note that it is not the group structure $(G, \cdot)$ itself that is automatic in the sense of Definition 2.1, but the Cayley graph. There are many natural examples of automatic groups (see [9, 10]). 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.

Remark. 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 [19] it is not clear whether the converse holds. Indeed the definition of an automatic group requires that the function $\nu : L_\mathfrak{A} \rightarrow G$ is the restriction of the canonical homomorphism from $S^*$ to $G$. The mere condition that $\Gamma(G, S)$ is an automatic structure does not seem to imply this.

$\omega$-automatic structures. The notion of an automatic structure can be modified and generalised in a number of different directions (see [2, 19]). In particular, we obtain the interesting class $\omega$-$\text{AutStr}$ of $\omega$-automatic structures.

The definition is analogous to the one for automatic structures except that the elements of an $\omega$-automatic structure are named by infinite words from some regular $\omega$-language and the relations of the structure are recognisable by Büchi automata.

Examples. (1) All automatic structures are $\omega$-automatic.
(2) The real numbers with addition, \((\mathbb{R}, +)\), and indeed the expanded structure \(\mathfrak{R}_p := (\mathbb{R}, +, \leq, |, 1)\) are \(\omega\)-automatic, where 
\[
x \mid y : if \exists n, k \in \mathbb{Z} : x = p^n \text{ and } y = kx.
\]
(3) The tree automatic structures \(\text{Tree}(\Sigma)\) extend in a natural way to the (uncountable) \(\omega\)-automatic structures \(\text{Tree}^* = (\Sigma^{\leq \omega}, (\sigma_n)_{n \in \mathbb{N}, \leq, t})\).

3. Model-checking and query-evaluation

In this section we study decidability and complexity issues for automatic structures. For a structure \(\mathfrak{A}\) and a formula \(\varphi(x)\), let \(\varphi_{\mathfrak{A}} := \{ \bar{a} : \mathfrak{A} \models \varphi(\bar{a}) \}\) be the relation (or query) defined by \(\varphi\) on \(\mathfrak{A}\). 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 a pair \((\nu, \delta)\) representing \(\mathfrak{A}\), construct an automaton that recognises \(\nu^{-1}(\varphi_{\mathfrak{A}})\).

We first observe that all first-order queries on automatic structures are effectively computable. 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”.

**Proposition 3.1.** Given an injective presentation \((\nu, \delta)\) of an automatic or \(\omega\)-automatic structure \(\mathfrak{A}\) and a formula \(\varphi(\bar{x}) \in \text{FO}(\exists^\omega)\) one can effectively construct an automaton recognising \(\nu^{-1}(\varphi_{\mathfrak{A}})\).

**Proof.** For FO-formulae this follows readily from classical results on the closure properties of regular (\(\omega\))-languages.

In case of automatic structures the quantifier \(\exists^\omega\) can be handled using a pumping argument. Consider for simplicity the formula \(\exists^\omega x \psi(x, y)\). There are infinitely many \(x\) satisfying \(\psi\) iff for any \(m\) there are infinitely many \(x\) 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 \(\psi\) 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. Note that the injectivity of \((\nu, \delta)\) ensures that each of the infinitely many words encodes a different element of \(\mathfrak{A}\).

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

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

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

Assume there are words \(v_0, \ldots, v_s \in V(\nu)\) belonging to different finite \(\approx^*\)-classes. Denote the run (sequence of states) of \(M\) on \(w \otimes v_1\) by \(s_1\). Define \(I_{ij} := \{ k < \omega : s_k = \nu_i[k]\}\). 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., \(\bigcup I_{ij} = \omega\). Thus, at least one \(I_{ij}\) is infinite. For each \([v_i]_*\) there is a position \(n_i\) such that \(v_i[n_i, \omega) = v_i'[n_i, \omega)\) for all \(v, v' \in [v_i]_*\). Let \(m\) be the maximum of \(n_0, \ldots, n_s\). Fix \(i, j\) such that \(I_{ij}\) is infinite. Since \(v_i \neq v_j\) 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 \(u := v_i[n_0, \ldots, n_s]\). Then \(w \otimes u \in L(M)\) iff \(w \otimes u \in L(M)\) implies that \(u \in [v_i]_*\). But \(u[n, \omega) \neq v_i[m, \omega)\) in contradiction to the choice of \(m\).

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

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

Clearly, \(\approx^*\) and \(\text{equal}\) can be recognised by \(\omega\)-automata. By the claim above,

\[
\varphi(\bar{x}) = \exists v (\psi(\bar{x}, v) \wedge \neg \text{finite}(\bar{x}, v)).
\]

Hence, we can construct an automaton recognising \(\varphi_{\mathfrak{A}}\).

**Corollary 3.2.** The \(\text{FO}(\exists^\omega)\)-theory of any automatic structure and of any \(\omega\)-automatic with injective presentation is decidable.

As an immediate consequence we conclude that full arithmetic \((\mathbb{N}, +, \cdot)\) is neither automatic, nor \(\omega\)-automatic. For most of the common extensions of first-order logic used in finite model theory, such as transitive closure logics, fixed point logics, monadic second-order 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 vary much depend on how automatic structures are presented. We focus here on presentations by deterministic automata because these admit boolean operations to be performed in polynomial time, whereas for nondeterministic 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, we denote by $|\mathfrak{d}|$ the maximal size of the automata in $\mathfrak{d}$, and for an automatic presentation $(\nu, \mathfrak{d})$ of the structure $\mathfrak{A}$, we define $\lambda^\mathfrak{A} : \mathcal{N} \to \mathcal{N}$ to be the function

$$\lambda^\mathfrak{A}(a) := \min \{|x| : \nu(x) = a\}$$

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

While we have seen above that query-evaluation and model-checking for first-order formulae are effective on $\text{AutStr}$, 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 the the complexity of $\text{Th}(\mathfrak{N}_\nu)$ is non-elementary (see [11]).

**Proposition 3.3.** 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 $\Sigma_0$ and $\Sigma_1$ denote, respectively the class of quantifier-free and the class of existential first-order formulae.

**Theorem 3.4.** (i) Given a presentation $\mathfrak{d}$ of a relational structure $\mathfrak{A} \in \text{AutStr}$, a tuple $\bar{a}$ in $\mathfrak{A}$, and a quantifier-free formula $\varphi(\bar{x}) \in \text{FO}$, the model-checking problem for $(\mathfrak{A}, \bar{a}, \varphi)$ is in

- $\text{DTIME}[O(|\varphi| \lambda^\mathfrak{A}(\bar{a}) |\mathfrak{d}| \log |\mathfrak{d}|)]$ and
- $\text{DSPACE}[O(|\varphi| + \log |\mathfrak{d}| + \log \lambda^\mathfrak{A}(\bar{a}))]$.

(ii) The structure complexity of model-checking for quantifier-free formulae is $\text{LOGSPACE}$-complete with respect to $\text{FO}$-reductions.

(iii) The expression complexity is $\text{ALOGTIME}$-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 $\varphi$. Then, all what remains is to evaluate a boolean formula which can be done in $\text{DTIME}[O(|\varphi|)]$ and $\text{ATIME}[O(\log |\varphi|)] \subseteq \text{DSPACE}[O(\log |\varphi|)]$ (see [5]). 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 $O(\lambda^\mathfrak{A}(\bar{a}) |\mathfrak{d}| \log |\mathfrak{d}|)$ and space $O(\log |\mathfrak{d}| + \log \lambda^\mathfrak{A}(\bar{a}))$.

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 $\text{LOGSPACE}$-algorithm to evaluate $\varphi$ 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 $\text{LOGSPACE}$-complete problem $\text{DETREACH}$, reachability by deterministic paths, (see e.g. [18]) 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 $0^v$ iff $0^{|V|} \in L(M)$. Thus,

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

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 $\text{ALOGTIME}$-complete (see [5]).

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 do to this and hence need to consider the case of quantifier-free formulae with function symbols separately. This class is denoted $\Sigma_0 + \text{fun}$. The following lemma is essentially due to Epstein et al. [9].

**Lemma 3.5.** Given a tuple $\bar{w}$ of words over $\Sigma$, and an automaton $\mathfrak{A} = (Q, \Sigma, \delta, q_0, F)$ recognising the graph of a
which can clearly be computed in logarithmic space.

\[ w \]

where \( \| \)

regard to \( \varphi \) values of all functions appearing in

(i) Our algorithm proceeds in two steps. First the

mains which can be evaluated as above. We need to evaluate

with the innermost one. Then all functions can be replaced

functions are guessed nondeterministically. Each simula-

input. Components of the input corresponding to values of

automata of the relation and of all functions on the given

functions. The algorithm simultaneously simulates the

machine for some polynomial \( p \).

components of the input correspond to values of

functions are guessed nondeterministically. We assume that

functions. If they are nested the result can be of

length \( |\varphi| |b| + \lambda^3(a) \). This yields the bounds given above.

It is sufficient to present a nondeterministic log-

space algorithm for evaluating a single fixed atom contain-

parameter this yields not a many-to-one but a truth-table re-

(iii) The expression complexity is PTIME-complete with

regard to \( \leq_{m}^{\log} \)-reductions.

Proof. (i) Our algorithm proceeds in two steps. First the

values of all functions appearing in \( \varphi \) are calculated starting

with the innermost one. Then all functions can be replaced

by their values and a formula containing only relations re-

mains which can be evaluated as above. We need to evaluate

at most \( |\varphi| \) functions. If they are nested the result can be of

length \( |\varphi| |b| + \lambda^3(a) \). This yields the bounds given above.

(ii) It is sufficient to present a nondeterministic log-

space algorithm for evaluating a single fixed atom contain-

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 nondeterministically. Each simul-

ation 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 \( w_{0} q_{w} w_{1} \) with \( w = w_{0} q_{w} w_{1} \) 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 recog-

nised by automata. We assume that \( f(a) = c \) 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 \( \mathfrak{A} = (A, P, f) \)

where \( \mathfrak{A} = (A, P, f) \) is automatic. Hence, the mapping tak-

ing \( w \) to the pair \( q_{0} w \) and \( P f^{p(|w|)} x \) is the desired reduction

which can clearly be computed in logarithmic space.

Remark. Theorem 3.6 says that, on any fixed automatic

structure, quantifier-free formulae can be evaluated in

quadratic time. This extends the result of [9] that the

word problem for every automatic group is solvable in

quadratic time. Indeed, for every automatic group \( G \) generated by

\( s_{1}, \ldots, s_{m} \), the structure \( (G, e, g \mapsto g s_{1}, \ldots, g \mapsto g s_{m}) \)

is just a functional way of presenting the Cayley graph and

therefore automatic. Each instance of the word problem is

described by a quantifier-free sentence (a term equation) on

this structure.

Theorem 3.7. (i) Given a presentation \( \mathfrak{A} \) of a structure \( \mathfrak{A} \)

in \( \text{AutStr}^{\tau} \), a tuple \( a \) in \( \mathfrak{A} \), and a formula \( \varphi(x) \in \Sigma_{1} \),

the model-checking problem for \( \mathfrak{A}, a, \varphi \) is in

\[ \text{DTIME} \left[ O(|\varphi| |b| \log |b| + \lambda^3(a)) \right] \]

and

\[ \text{DSpace} \left[ O(|\varphi| |b| + \lambda^3(a)) \right] \].

(ii) The structure complexity of the model-checking problem

for quantifier-free formulae with functions is in

\( \text{NLOGSPACE} \).

(iii) The expression complexity is \( \text{PTIME} \)-complete with

regard to \( \leq_{m}^{\log} \)-reductions.

Proof. (i) As above we can run the corresponding automa-

ton for every atom appearing in \( \varphi \) on the encoding of \( a \). But

now there are some elements of the input missing which we

to have to guess. Since we have to ensure that the guessed

inputs are the same for all automata, the simulation is per-

formed simultaneously.

The algorithm determines which atoms appear in \( \varphi \) and

simulates the product automaton constructed from the automa-

ta for those relations. At each step the symbol for the

quantified variables is guessed nondeterministically. 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 space-state number of

steps. Since this cardinality is \( O(|b|^{|w|}) \) a closer inspection

of the algorithm yields the given bounds.

(ii) We reduce the \( \text{NPTIME} \)-complete non-universality

problem for nondeterministic automata over a unary alphabet

(see [21, 17]), 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 deter-

ministic 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

the model-checking has to be performed for more than one pa-

rameter this yields not a many-to-one but a truth-table re-

duction.

Let \( \mathcal{M} = (Q, \{0\}, \Delta, q_{0}, F) \) be a nondeterministic fi-

nite automaton over the alphabet \( \{0\} \). We construct an

automaton \( \mathcal{M}^* \) such that there are at most two transitions

outgoing at every state. This is done by replacing all transi-

tions form some given state by a binary tree of transi-

tions with new states as internal nodes. Of course, this

changes the language of the automaton. Since in \( \mathcal{M} \) 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 \( M \).

\( M' \) is still non-deterministic. To make it deterministic we add a second component to the labels of each transitions which is either 0 or 1. This yields an automaton \( M'' \) such that \( M \) accepts the word \( 0^k \) 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

\[ \mathfrak{B} \models \exists y R0^{kn}y \quad \text{iff} \quad 0^{kn} \otimes y \in L(M'') \]

\[ \quad \quad \quad \quad \quad \quad \quad \text{iff} \quad 0^n \in L(M). \]

It follows that

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

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

(iii) is shown by coding computations of Turing machines. The proof can be found in [2].

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

**Theorem 3.8.** Given a presentation \( \mathfrak{d} \) of a structure \( \mathfrak{A} \) in \( \text{AutStr} \) and a formula \( \varphi(x) \), an automaton representing \( \varphi^{\mathfrak{A}} \) can be computed

(i) in time \( O(|\mathfrak{d}| \cdot O(|\varphi|)) \) and space \( O(|\varphi| \cdot \log |\mathfrak{d}|) \) in the case of quantifier-free \( \varphi(x) \), and

(ii) in time \( O(2^{\log |\varphi|} \cdot |\mathfrak{d}|) \) and space \( O(\log |\varphi| \cdot |\mathfrak{d}|) \) in the case of existential formulae \( \varphi(x) \).

In particular, the structure complexity of query-evaluation is in \( \text{LOGSPACE} \) for quantifier-free formulae and in \( \text{PSPACE} \) for existential formulae. The expression complexity is in \( \text{PSPACE} \) for quantifier-free formulae and in \( \text{EXPSPACE} \) for existential formulae.

**Proof.** Enumerate the state space of the product automaton and output the transition function. \( \square \)

### 4. Structures that are not automatic

To prove that a structure is automatic, we just have to find a suitable presentation. But 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.\(^1\)

\[ \begin{array}{|c|c|c|}
\hline
\text{Model-Checking} & \text{Structure-Complexity} & \text{Expression-Complexity} \\
\hline
\Sigma_0 & \text{LOGSPACE-complete} & \text{ALOGTIME-complete} \\
\Sigma_0 + \text{fun} & \text{NLOGSPACE} & \text{PTIME-complete} \\
\Sigma_1 & \text{NPTIME-complete} & \text{PSPACE-complete} \\
\hline
\end{array} \]

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 use growth rates for the length of the encodings of elements of definable sets.

**Proposition 4.1.**\(^2\) Let \( \mathfrak{A} \) be an automatic structure with injective presentation \( (\nu, \lambda) \), and let \( f : A^n \to A \) be a function of \( \mathfrak{A} \). Then there is a constant \( m \) such that \( \lambda^f(\bar{a}) \leq \lambda^0(\bar{a}) + m \) 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 \( Rb \) holds.

This result deals with a single application of a function or relation. In the remaining part of this section we will study the effect of applying functions iteratively, i.e., we will 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., [7, Proposition V.1.11]).

**Lemma 4.2.** Let \( \mathfrak{A} \) be a structure in \( \text{AutStr} \) with presentation \( \mathfrak{d} \), and let \( B \) be an FO(\( \exists^* \))-definable subset of \( A \). Then \( \lambda^B(\bar{a}) \) 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 4.3.** Let \( \mathfrak{A} \in \text{AutStr} \) with presentation \( \mathfrak{d} \), let \( f_1, \ldots, f_r \) be finitely many operations of \( \mathfrak{A} \) with arities \( r_1, \ldots, r_r \), respectively, and let \( E = \{ e_1, e_2, \ldots \} \) be some subset of \( A \) with \( \lambda^0(e_1) \leq \lambda^0(e_2) \leq \cdots \). Then \( G_n(E) \), the \( n \)th generation of \( E \), is defined inductively by

\[ G_1(E) := \{ e_1 \}, \]

\[ G_n(E) := \{ e_n \} \cup G_{n-1}(E) \]

\[ \cup \{ f_i(\bar{a}) : \bar{a} \in G_{n-1}(E), 1 \leq i \leq r \}. \]

Putting everything together we obtain the following result. The case of finitely generated substructures already appeared in [19].
Proposition 4.4. Let $\mathfrak{A}$ an injective presentation of an automatic structure $\mathfrak{A}$, let $f_1, \ldots, f_r$ be finitely many definable operations on $\mathfrak{A}$ and let $E$ be a definable subset of $\mathfrak{A}$. Then there is a constant $m$ such that $|\lambda^1(\alpha)| \leq mm$ for all $\alpha \in G_n(E)$. In particular, $|G_n(E)| \leq |\Sigma|^{m+1}$ where $\Sigma$ is the alphabet of $\mathfrak{A}$.

The proof consists of a simple induction on $n$.

Theorem 4.5. 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 $([N, \mid])$.

(iv) Skolem arithmetic $([\mathbb{N}, -1])$.

Proof. (i) We show that $\{a, b\}^2 \subseteq G_{n+1}(a, b)$ by induction on $n$. We have $\{a, b\} \subseteq \{a, aa, b\} = G_2(a, b)$ for $n = 1$, and for $n > 1$
\[
G_{n+1}(a, b) = \{uv : u, v \in G_n(a, b)\} \\
\supseteq \{uv : u, v \in \{a, b\}^2 \} \\
= \{a, b\}^{2^n}.
\]

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 $([\mathbb{N}, -1])$.

(iii) Suppose $([\mathbb{N}, \mid]) \in \text{AutStr}$. We define the set of primes
\[
P_x : \text{iff } x \neq 1 \land \forall y(y \mid x \rightarrow y = 1 \lor y = x),
\]
the set of powers of some prime
\[
Q_x : \text{iff } \exists y(Py \land \forall z(z \mid x \land z \neq 1 \rightarrow y \mid z)),
\]
and a relation containing all pairs $(n, pn)$ where $p$ is a prime divisor of $n$
\[
Sxy : \text{iff } x \mid y \land \exists z(z \mid Qy \land \neg Pz \land z \mid y \land \neg z \mid x).
\]
The least common multiple of two numbers is
\[
\text{lcm}(x, y) = z : \text{iff } x \mid z \land y \mid z \\
\land \neg \exists u(u \neq z \land x \mid u \land y \mid u \land u \mid z).
\]

For every $n \in \mathbb{N}$ there are only finitely many $m$ with $\text{Sum}_m$. Therefore $S$ satisfies the conditions of Proposition 4.1. Consider the set generated by $P$ via $S$ and $\text{lcm}$, and let $\gamma(n) := |G_n(P)|$ be the cardinality of $G_n(P)$. If $([\mathbb{N}, \mid])$ is in AutStr then $([\mathbb{N}, \mid], P, Q, S) \in \text{AutStr}$ and $\gamma(n) \in 2^{2^n}$ by Proposition 4.4. 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) $p_1, \ldots, p_n^a$ (via $S$),
(2) $\gamma(n - 1)$ numbers with $k_1 = 0$, and
(3) for every $0 < k_1 < n$, $\gamma(n - 2)$ numbers of a mixed form (via $\text{lcm}$).

All in all we obtain
\[
\gamma(n) \geq n + \gamma(n - 1) + (n - 1)\gamma(n - 2) + 1 \\
\geq n\gamma(n - 2) + (n - 1)\gamma(n - 2) + 1 \\
\geq n\gamma(n - 2) \cdot 3\gamma(1) \\
\geq (n + 1)! \\
\in 2^{\Omega(n \log n)}. \\
\]

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 [2]).

(2) The structure $([\mathbb{N}, \bot])$ where $\bot$ stands for having no common divisor is automatic.

5. Characterising automatic structures via interpretations

Interpretations are important in mathematical logic, for model-theory in particular. They are used to define a copy of a structure inside another one, and thus permit to transfer definability, decidability, and complexity results among theories.

Definition 5.1. A $(k$-dimensional) interpretation of a relational $\sigma$-structure $\mathfrak{A} = (A, R_1, \ldots, R_m)$ in a $\tau$-structure $\mathfrak{B}$ is given by a sequence
\[
I = (\delta(x), e(x, y), \varphi_{R_1}(x_1, \ldots, x_{r_1}), \ldots)
\]
of first-order formulae of vocabulary $\tau$ (where each tuple $x, y, x_i$ consists of $k$ variables), provided that there exists a surjective map $h : \delta^\mathfrak{B} \rightarrow A$, called the coordinate map of the interpretation such that the following hold:

(i) For all $\bar{b}, \bar{c} \in \delta^\mathfrak{B}$
\[
\mathfrak{B} \models e(\bar{b}, \bar{c}) \text{ iff } h(\bar{b}) = h(\bar{c}),
\]

(ii) For all $\bar{b} \in \delta^\mathfrak{B}$
\[
\mathfrak{B} \models \delta(x) \text{ iff } x = \bar{b},
\]
(ii) for every relation $R_j$ of $\mathfrak{A}$ and all $\bar{b}_1, \ldots, \bar{b}_{r_j} \in \delta^{2b}$

$$\mathfrak{B} \models \varphi_{R_j}(\bar{b}_1, \ldots, \bar{b}_{r_j}) \text{ iff } (h(\bar{b}_1), \ldots, h(\bar{b}_{r_j})) \in R.$$ 

That is, the formula $\varepsilon(\bar{x}, \bar{y})$ defines a congruence on the structure $(\delta^{2b}, \varphi^{R_1}_{R_1}, \ldots, \varphi^{R_m}_{R_m})$ such that $h$ is an isomorphism from the quotient structure $(\delta^{2b}/\varepsilon, \varphi^{R_1}_{R_1}/\varepsilon, \ldots, \varphi^{R_m}_{R_m}/\varepsilon)$ to $\mathfrak{A}$. In the case that $\mathfrak{A}$ is this quotient structure itself (rather than just being isomorphic to it) we say that $\mathfrak{A}$ is definable in $\mathfrak{B}$. Obviously, $\mathfrak{A}$ is definable in $\mathfrak{B}$ if and only if there is an interpretation of $\mathfrak{A}$ in $\mathfrak{B}$ whose coordinate map is the canonical projection, mapping every tuple $\bar{b} \in \delta^{2b}$ to its equivalence class $\bar{b}/\varepsilon$.

If $\mathfrak{A}$ is a structure including not only relations but also functions then, by definition, an interpretation of $\mathfrak{A}$ in $\mathfrak{B}$ is an interpretation of the relational variant of $\mathfrak{A}$ (where functions are replaced by their graphs) in $\mathfrak{B}$.

We write $\mathfrak{A} \leq_{\text{FO}} \mathfrak{B}$ to denote that there exists an interpretation of $\mathfrak{A}$ in $\mathfrak{B}$. If both $\mathfrak{A} \leq_{\text{FO}} \mathfrak{B}$ and $\mathfrak{B} \leq_{\text{FO}} \mathfrak{A}$ we say $\mathfrak{A}$ and $\mathfrak{B}$ are mutually interpretable.

Examples. (1) Recall that we write $a \mid b$ to denote that $a$ is a power of $b$ 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 $(\mathbb{N}, +, \mid_p)$ and $(\mathbb{N}, +, V_p)$ are mutually interpretable. Indeed we can define the statement $x = V_p(y)$ in $(\mathbb{N}, +, \mid_p)$ by the formula $x \mid_p y \land \forall z(z \mid_p y \rightarrow z \mid_p x)$. In the other direction, $V_p(x) = x \land \exists z(x + z = V_p(y))$ is a definition of $x \mid_p y$.

(2) For every $p \in \mathbb{N}$ we write $\text{Tree}(p)$ for the tree structure $\text{Tree}(\{0, \ldots, p-1\})$. The structures $\mathfrak{N}_p$ and $\text{Tree}(p)$ are mutually interpretable, for each $p \geq 2$ (see [2, 11]).

Observe that Proposition 3.1 implies an interesting closure property for $\text{AutStr}$ and $\omega\text{-AutStr}$.

Proposition 5.2. The classes of automatic and $\omega$-automatic structures are closed under interpretations, i.e., if $\mathfrak{B}$ is a $\omega$-automatic and $\mathfrak{A} \leq_{\text{FO}} \mathfrak{B}$, then so is $\mathfrak{A}$.

Corollary 5.3. The classes of automatic, resp. $\omega$-automatic, structures are closed under (i) extensions by definable relations, (ii) factorisations by definable congruences, (iii) substructures with definable universe, and (iv) finite powers.

The model-theoretic characterisation of automatic structures is given in the following theorem. It states that the structure $\mathfrak{N}_p$ (and $\text{Tree}(p)$) is complete for $\text{AutStr}$, i.e., a structure $\mathfrak{A}$ belongs to $\text{AutStr}$ if and only if $\mathfrak{A} \leq_{\text{FO}} \mathfrak{N}_p$.

Theorem 5.4. For every structure $\mathfrak{A}$, the following are equivalent:

(i) $\mathfrak{A}$ is automatic.

(ii) $\mathfrak{A} \leq_{\text{FO}} \mathfrak{N}_p$ for some (and hence all) $p \geq 2$.

(iii) $\mathfrak{A} \leq_{\text{FO}} \text{Tree}(p)$ 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 $\text{Tree}(p)$, from the fact that these structures are automatic, and from the closure of automatic structures under interpretations.

It remains to show that every automatic structure is interpretable in $\mathfrak{N}_p$ (or $\text{Tree}(p)$). Suppose that $\mathfrak{A}$ 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 $\text{val}(w)$ be the natural number whose $p$-ary encoding is $w$, i.e., $\text{val}(w) := \sum_{i < |w|} w_ip^i$. 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

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

is regular. (See [4] 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 $\mathfrak{A} \leq_{\text{FO}} \mathfrak{N}_p$.

For automatic groups we are not free to change the coordinate map, therefore the arguments used above give a characterisation in terms of definability rather than interpretability.

Theorem 5.5. $(G, \cdot)$ is an automatic group if and only if there exists a finite set $S \subseteq G$ of semigroup generators such that $I(G, S)$ is $\text{FO}$-definable in $\text{Tree}(S)$.

We now turn to $\omega$-automatic structures. To provide a similar characterisation we can use an equivalent of the Büchi-Bruyère Theorem for encodings of $\omega$-regular relations. One such result has been obtained recently by Boigelot, Rassart and Wolper [3]. Using natural translations between $\omega$-words over $[p]$ and real numbers, they prove that a relation over $[p]^\omega$ can be recognised by a Büchi automaton if an 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 $\text{Tree}^\omega(p)$ (introduced at the end of Section 2) are $\omega$-automatic. As a first
step we show that the behavior of Büchi automata recognizing regular relations over $[p]^w$ can be simulated by first-order formulae in $\text{Tree}^e(p)$. Second we show that $\text{Tree}^e(p)$ and $\mathfrak A_p$ are mutually interpretable. As a result we obtain the following model-theoretic characterisation of $\omega$-automatic structures.

**Theorem 5.6.** For every structure $\mathfrak A$, the following are equivalent

(i) $\mathfrak A$ is $\omega$-automatic.

(ii) $\mathfrak A \leq_{\text{FO}} \mathfrak A_p$ for some (and hence all) $p \geq 2$.

(iii) $\mathfrak A \leq_{\text{FO}} \text{Tree}^e(p)$ for some (and hence all) $p \geq 2$.

**Proof.** In order to construct interpretations of $\text{Tree}^e(p)$ in $\mathfrak A_p$ and vice versa, we define formulae which allow to access the digits of, respectively, some number in $\mathfrak A_p$ and some word in $\text{Tree}^e(p)$. In the later case we set

$$\text{dig}_k(x, y) := \exists z (\text{el}(x, y) \land \sigma_k z \leq x)$$

which states that the digit of $x$ at position $|y|$ is $k$. For $\mathfrak A_p$ 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 [3].)

$$\text{dig}_k(x, y) := \exists s \exists t (|x| = s + k \cdot y + t \land p \cdot y |_p s \land 0 \leq s \land 0 \leq t \leq y)$$

For $\mathfrak A_p \leq_{\text{FO}} \text{Tree}^e(p)$ 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

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

Infinite words $a_1a_2\cdots \in [p]^w$ are mapped to two intervals $[-1, 0]$ and $[0, 1]$ via

$$\pm \sum_{i} a_ip^{-i} \in [-1, 1].$$

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

It remains to prove that if $R \subseteq ([p]^*)^p$ is $\omega$-regular then it is definable in $\text{Tree}^e(p)$. Let $M = (Q, [p]^n, \Delta, q_0, F)$ be a Büchi-automaton for $R$. W.l.o.g. assume $Q = [p]^m$ for some $m$ and $q_0 = (0, \ldots, 0)$. We prove the claim by constructing a formula $\psi_M(x) \in \text{FO}$ stating that there is a successful run of $M$ on $x_1 \otimes \cdots \otimes x_n$. The run is encoded by a tuple $(q_1, \ldots, q_m) \in ([p]^w)^m$ of $\omega$-words such that the symbols of $q_1, \ldots, q_m$ at some position equal $k_1, \ldots, k_m$ iff the automaton is in state $(k_1, \ldots, k_m)$ when scanning the input symbol at that position. $\psi_M(x)$ has the form

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

where the admissibility condition $\text{ADM}(\bar x, \bar q)$ states that all components of $\bar x$ and $\bar q$ are finite, $\text{START}(\bar x, \bar q)$ says that the first state is 0, $\text{ACC}(\bar x, \bar q)$ that some final state appears infinitely often, and $\text{RUN}(\bar q, \bar x)$ 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 $\text{Sym}_a(\bar x, z) := \bigwedge_i \text{dig}_a(x_i, z)$, and to characterise the $\omega$-words of $[p]^w$ we set

$$\text{Inf}(x) := \forall y (x \leq y \rightarrow x = y).$$

$\text{ADM}$ and $\text{START}$ are defined as

$$\text{ADM}(\bar q, \bar x) := \bigwedge_i \text{Inf}(q_i) \land \bigwedge_i \text{Inf}(x_i),$$

$$\text{START}(\bar q, \bar x) := \text{Sym}_q(\bar q, \epsilon),$$

$\text{RUN}$ states that at every position a valid transition is used

$$\text{RUN}(\bar q, \bar x) := \forall z \bigvee_{(k, a, k') \in \Delta} (\text{Sym}_k(\bar q, z) \land \text{Sym}_a(\bar x, z) \land \text{Sym}_{k'}(\bar q, \sigma_0 z)), $$

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

$$\text{ACC}(\bar q, \bar x) := \bigvee_{k \in F} \forall z \exists z' (|z'| > |z| \land \text{Sym}_k(\bar q, z')), \square$$

6. **Composition of structures**

The composition method developed by Feferman and Vaught, and by Shelah 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 monadic second-order theory of the whole structure from the respective theories of its components and the monadic theory of the index structure. 

The characterisation given in the previous section can be used to prove closure of automatic structures under such compositions of finitely many structures (see [23, 13, 16]). 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 which belong to some class $K$ containing a complete structure, all their generalised products are members of $K$ as well.

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

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

Example. (1) Let $A := A_0 \times A_1$ where $A_i = (A_i, R_i)$, for $i \in \{0, 1\}$, and $R$ is a binary relation. The universe of $A$ is $A_0 \times A_1$. Some pair $(a, b)$ 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 $a_0b_0$ and of $a_1b_1$ both include the formula $R_{x_0x_1}$.

(2) Let $A := A_0 + A_1$ where $A_i = (A_i, <_i)$, for $i \in \{0, 1\}$, and $<_0, <_1$ are partial orders. The universe of $A$ is $A_0 \cup A_1 \cong A_0 \times \{0\} \cup \{1\} \times A_1$, and we have
\[
\hat{a} < \hat{b} \text{ iff } \hat{a} = (a_0, \emptyset), \hat{b} = (b_0, \emptyset) \text{ and } a_0 <_0 b_0, \\
or \hat{a} = (\emptyset, a_1), \hat{b} = (\emptyset, b_1) \text{ and } a_1 <_1 b_1, \\
or \hat{a} = (a_0, \emptyset), \hat{b} = (\emptyset, b_1) \text{ and } a_0 <_1 b_1.
\]
Again, the condition $a_i <_i b_i$ can be expressed using types.

Definition 6.1. Let $\tau = \{R_0, \ldots, R_s\}$ be a finite relational vocabulary, $r_j$ the arity of $R_j$, and $\hat{r} := \max\{r_0, \ldots, r_s\}$. Let $(\mathfrak{A}_i)_{i \in I}$ be a sequence of $\tau$-structures, and $\mathcal{I}$ be an arbitrary relational $\sigma$-structure with universe $I$.

Fix for each $k \leq \hat{r}$ an enumeration $\{t^k_i \mid i \in I\}$ of the atomic $k$-types and set
\[
\sigma_k := \sigma \cup \{D_{00}, \ldots, D_{k-1}\} \\
\cup \{T^m_{i} : m \leq k, l \leq n(m)\}.
\]

The $\sigma_k$-expansion $\mathfrak{J}(\hat{b})$ of $\mathcal{I}$ belonging to a sequence $\hat{b} \in (I_0 \cup \{\emptyset\})^k$ is given by
\[
D_{i}(\mathfrak{J}(\hat{b})) = \{i \in I : (b)_i \neq \emptyset\}, \\
(T^m_{i})(\mathfrak{J}(\hat{b})) = \{i \in I : atP_{t^m_{i}}((b)_1, \ldots, (b)_{m-1}, (b)) = t^m_i \\
\text{ and } \{j : (b)_j = \emptyset\} = \{0, \ldots, j_{m-1}\}\).
\]

For $D \subseteq I^\tau$ and $\beta_1 \in FO(\sigma_{\hat{r}}), C := (\mathfrak{I}, D, \beta_0, \ldots, \beta_s)$ defines the generalised product $C(\mathfrak{A}_i)_{i \in I}$ of $(\mathfrak{A}_i)_{i \in I}$ where
\[
A := \bigcup_{d \in D} \prod_{i \in I} \chi_d, \{\emptyset\}, A_i, \\
R_i := \{\hat{b} \in A^\tau : \mathfrak{J}(\hat{b}) = \beta_i\}, \\
\text{ and } \chi_d(a_0, a_1) := a_b.
\]

Example. (continued)
(1) For the direct product of $A_0 \times A_1$, we would set $\mathcal{I} := (I)$ with $I = \{0, 1\}$, $D := \{(1, 1)\}$, and
\[
\beta := \bigvee_{i \in L} T^0_i \land \bigvee_{i \in L} T^1_i,
\]
where $L$ is the set of atomic types containing the formula $R_{x_0x_1}$.

(2) In this case we would set $\mathcal{I} := (I)$ with $I = \{0, 1\}$, $D := \{(0, 1), (0, 1)\}$, and
\[
\beta := \left(\bigvee_{i \in L} T^0_i \land \bigvee_{i \in L} T^1_i\right) \lor \left(\bigvee_{i \in L} T^0_i \land \bigvee_{i \in L} T^1_i\right),
\]
where $L$ is the set of atomic types containing the formula $x_0 < x_1$.

Theorem 6.2. Let $\tau$ be a finite relational vocabulary, and $K$ a class of $\tau$-structures containing all finite $\tau$-structures and a structure $C$ which is complete for $K$ with regard to many-dimensional FO-interpretations.

Let $\mathcal{I}$ be a finite relational $\sigma$-structure, let $(\mathfrak{A}_i)_{i \in I}$ be a sequence of structures in $K$, and $C = (\mathfrak{I}, D, \beta)$ a generalised product. Then $C(\mathfrak{A}_i)_{i \in I} \in K$, and an interpretation $C(\mathfrak{A}_i)_{i \in I} \models FO C$ can be constructed effectively from the interpretations $\mathfrak{A}_i \models FO C$ and $\mathcal{I} \models FO C$.

Proof. Let $\tau = \{R_0, \ldots, R_s\}$. W.l.o.g. assume that $I = \{0, \ldots, |I| - 1\}$ and that $C$ contains constants $0$ and $1$. We have to construct an interpretation of $\mathfrak{A} := C(\mathfrak{A}_i)_{i \in I} \in C$.

Let $r_j$ be the arity of $R_j$. Consider $n_i$-dimensional interpretations
\[
\mathcal{T}^i := \left\langle h^i, \delta^i(x^i), \psi^i(x^i, y^i), \varphi_{00}^i(x_0, \ldots, x_{n_i-1}), \ldots, \varphi_{10}^i(x_0, \ldots, x_{n_i-1}) \right\rangle
\]
of $\mathfrak{A}_i \in C$, where $\varphi_{kl}(x_0, \ldots, x_{n_i-1})$ contains constants $0$ and $1$.

We represent an element $a$ of $\mathfrak{A}$ by a tuple of $(|I| + n_0 + \cdots + n_{|I| - 1})$ elements
\[
\hat{x} := (\hat{d}, \hat{x}^0, \ldots, \hat{x}^{|I| - 1})
\]
where $\hat{d} \in D$ determines which components are empty and $\hat{x}^i$ encodes the $i^{th}$ component of $a$. The desired interpretation is constructed as follows.

\[
\mathfrak{I} := \left\langle h, \delta(\hat{x}), \varphi_{00}(\hat{x}_0, \ldots, \hat{x}_{n_i-1}), \ldots, \varphi_{10}(\hat{x}_0, \ldots, \hat{x}_{n_i-1}) \right\rangle
\]

where
\[
h(\hat{d}, \hat{x}^0, \ldots, \hat{x}^{|I| - 1}) := \bigvee_{i \in D} \left(\chi_d, (\hat{h}^i(\hat{x}^i, \hat{x}^{|I| - 1})) \right) \cup \bigvee_{i \in D} \left(\chi_{d_i}(\hat{h}^i(\hat{x}^i, \hat{x}^{|I| - 1})) \right) \cup \bigvee_{i \in D} \left(\chi_{d_i}(\hat{h}^i(\hat{x}^i, \hat{x}^{|I| - 1})) \right),
\]
\[
\delta(\hat{d}, \hat{x}^0, \ldots, \hat{x}^{|I| - 1}) := \bigvee_{i \in D} \left(\delta(\hat{x}^i, \hat{x}^{|I| - 1}) \right).
\]
and
\[ \varepsilon(d, \bar{x}, \ldots, \bar{x}^{I-1}, \bar{\varepsilon}, \bar{y}, \ldots, \bar{y}^{I-1}) := d = \bar{\varepsilon} \land \bigwedge_{i < |I|} (d_i = 1 \rightarrow \varepsilon^i(\bar{x}_i, \bar{y}_i)). \]

In order to define \( \varphi_j \) we consider an interpretation \( \mathcal{I} \) of \( \mathcal{J} \) in \( \mathcal{E} \). Since \( \mathcal{J} \) is finite such an interpretation exists. Let \( \alpha_j := \beta_j \mathcal{I} \) be the formula defining \( R_j \). Note that \( \beta_j \) contains additional relations \( D_l \) and \( T_l^m \) which are not in \( \sigma \). Thus \( \alpha_j \) is a sentence over the signature \( \tau \) 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 \( \varphi_j \). Let \( \bar{x}_0, \ldots, \bar{x}_{r_j-1} \) be the parameters of \( \varphi_j \) where
\[ \bar{x}_k = (\bar{d}_k, \bar{x}_k^0, \ldots, \bar{x}_k^{I-1}) \]
for \( k < r_j \). \( D_l \) and \( T_l^m \) can be defined by
\[ D_l i := (d_l)_i = 1 \quad \text{and} \quad T_l^m i := (t_l^m)^i (\bar{x}_0^i, \ldots, \bar{x}_{r_j-1}^i). \]

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

Obviously, all steps in the construction above are effective.

\[ \square \]

**Corollary 6.3.** Both, AutStr and \( \omega \)-AutStr are effectively closed under finitary generalised products.

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

**Corollary 6.4.** Let \( \mathcal{A}_0, \ldots, \mathcal{A}_{n-1} \in \text{AutStr} \). Then there exists automatic presentations of

(i) the direct product \( \prod_{i < n} \mathcal{A}_i; \)

(ii) the disjoint union \( \bigcup_{i < n} \mathcal{A}_i; \)

(iii) the \( \omega \)-fold disjoint union \( \omega \cdot \mathcal{A}_0 \) of \( \mathcal{A}_0 \).

**Corollary 6.5.** Let \( \mathcal{A}_0, \ldots, \mathcal{A}_{n-1} \in \text{AutStr} \) be ordered structures. There exists automatic presentations of

(i) the ordered sum \( \sum_{i < n} \mathcal{A}_i \) and

(ii) the \( \omega \)-fold ordered sum \( \sum_{\omega} \mathcal{A}_0 \) of \( \mathcal{A}_0 \).

**References**


