Sequential Automatic Algebras

Michael Brough, Bakhadyr Khoussainov, and Peter Nelson

Department of Computer Science, University of Auckland

Abstract. A *sequential automatic algebra* is a structure of the type $(A; f_1, \ldots, f_n)$, where A is recognised by a finite automaton, and functions $f_1, \ldots, f_n$ are total operations on A that are computed by input-output automata. Our input-output automata are variations of Mealy automata. We study some of the fundamental properties of these algebras and provide many examples. We give classification results for certain classes of groups, Boolean algebras, and linear orders. We also introduce different classes of sequential automatic algebras and give separating examples. We investigate linear orders considered as sequential automatic algebras. Finally, we outline some of the basic properties of sequential automatic unary algebras.

Introduction

The main contribution of this paper is the introduction of the various notions of sequential automatic algebras as natural sub-classes of the class of automatic structures. As such these algebras enjoy all the decidability and model-theoretic properties possessed by automatic structures. The goal is twofold. One is to provide examples and investigate fundamental properties of sequential automatic algebras. The other is to show that sequential automatic algebras have certain algebraic and algorithmic advantages as opposed to general automatic structures.

Algebras are structures of the form $(A; f_1, \ldots, f_n)$ where each f_i is a total operation on A. Usually operations are replaced by their graphs (in finite model theory for example). This transforms the algebra into a purely relational structure. Automata can then be used to recognize the graphs of the operations; this loses the input-output behavior of the functions because an automaton recognising the graph of an operation does not necessarily compute the output in a sequential manner. Here we propose to use input-output automata to represent operations of algebras in order to capture the input-output behavior of operations. Our input-output automata will be variations of Mealy automata.

Let Σ be an alphabet, and let $\Sigma_\Box = \Sigma \cup \{\Box\}$, where $\Box \notin \Sigma$. An **n-variable sequential Mealy automaton** $\mathcal{M}$ is a tuple (Q, q_0, Δ, O), where Q is the finite set of states, $q_0 \in Q$ the initial state, $\Delta : Q \times \Sigma_\Box^n \to Q \times \Sigma_\Box$ is the transition function, and $O : Q \to \Sigma^\star$ the final output function. Note that the automaton is *deterministic*. Such an automaton processes inputs of the form $(w_1, \ldots, w_n)$, where each $w_i \in \Sigma^\star$, and outputs a string from $\Sigma^\star$ as follows. Think of the automaton as having n input tapes, with w_i on the ith tape, and one output tape on which it writes symbols from $\Sigma_\Box$. The automaton moves each of its n heads

A. Beckmann, C. Dimitracopoulos, and B. Löwe (Eds.): CiE 2008, LNCS 5028, pp. 84–93, 2008.

simultaneously from left to right, reading a symbol from each of the input tapes, changes its state and writes a symbol on the output tape according to Δ (starting at q_0). If w_i is shorter than w_j, then the automaton assumes the $\Box$ symbol after the end of w_i. Once all input symbols are $\Box$, the automaton concatenates the string $O(s)$, where s is the current state, to the end of the string written to the output tape and then halts. The resulting string in the output tape, up to the first position where $\Box$ is written, is the output of $\mathcal{M}$. We require that once $\Box$ has been written to the output, all subsequent symbols written are $\Box$ and the final output function is empty. Thus writing $\Box$ can be thought of as terminating early. Each sequential n-variable Mealy automaton uniquely determines a function $f_{\mathcal{M}} : (\Sigma^\star)^n \to \Sigma^\star$ called a **sequential automatic operation**.

Definition 1. *An algebra* $\mathcal{A}=(A; f_1, \ldots, f_n)$ *is* **sequential automatic** *if the domain* A *is a regular language and operations* f_i *on* A *are sequential automatic.*

Operations computed by Mealy automata have been studied for many years. Mealy automata form a subclass of transducers, an area of active research in automata theory [3], and have also been studied by group theorists (see [9] for instance), initiated by Aleshin who used permutations computed by Mealy automata to solve the Burnside problem [1].

Some simple examples of sequential automatic algebras are finite algebras, the tree algebra $(\{0,1\}^\star; Left, Right)$ where $Left(x) = x0$ and $Right(x) = x1$, $(\omega; +)$, and $(\omega; S)$ where $S(n) + 1$ for $n \in \omega$.

If a sequential n-variable Mealy automaton $\mathcal{M}$ never writes the symbol $\Box$, we call $\mathcal{M}$ a **strictly sequential n-variable Mealy automaton** and define a **strictly sequential automatic operation** and a **strictly sequential automatic algebra** accordingly. Strictly sequential automatic algebras form a subclass of sequential automatic algebras.

For the next definition we briefly explain finite automaton recognisable relations. An automaton $\mathcal{M}$ recognising a relation R of arity n behaves exactly as an n-variable Mealy automaton but with no outputs; instead $\mathcal{M}$ has a set $Q_f \subset Q$ of accepting states. $\mathcal{M}$ processes a tuple $(w_1, \ldots, w_n)$ in the same way as Mealy automata do, and accepts the tuple iff after processing the tuple, it is in one of the accepting states. Now we define automatic structures. These have been studied in [4], [5], [10], [12], [17].

Definition 2. *A relational structure* $\mathcal{A}=(A; R_1, \ldots, R_m)$ *is* **automatic** *if* A, R_1, …, R_m *are all finite automaton recognisable.*

We also use sequential automatic functions as mappings between equivalence classes. Let f be a function computed by an n-variable sequential Mealy automaton. Let E be an equivalence relation on A. We say that $f_{\mathcal{M}}$ **respects** E if for all $(w_1, \ldots, w_n)$, $(w'_1, \ldots, w'_n)$ the condition (w_1, w'_1), …, $(w_n, w'_n) \in E$ implies that $(f_{\mathcal{M}}(w_1, \ldots, w_n),\ f_{\mathcal{M}}(w'_1, \ldots, w'_n)) \in E$. If every operation of an algebra respects E then E is called a **congruence** relation of the algebra.

Definition 3. *Let $\mathcal{A}=(A; f_1, \ldots, f_n)$ be a sequential automatic algebra. Let E be a finite automaton recognisable congruence relation of $\mathcal{A}$. The quotient algebra $\mathcal{A}/E$ is called a* **generalised sequential automatic** *algebra. If E additionally satisfies the property that for $x = x'\sigma^k$, where x' does not end in σ, the equivalence class of x is $\{x'\sigma^n | n \in \omega\} \cap A$, then we call the factor algebra $\mathcal{A}/E$* **continuous generalised sequential automatic**.

Let $\mathcal{SSA}$, $\mathcal{SA}$, $\mathcal{CGSA}$ and $\mathcal{GSA}$ denote respectively the classes of strictly sequential automatic, sequential automatic, continuous generalised sequential automatic, and generalised sequential automatic algebras (all closed under isomorphisms). We have: $\mathcal{SSA} \subseteq \mathcal{SA} \subseteq \mathcal{CGSA} \subseteq \mathcal{GSA}$. We will provide separating examples for these later. This is unlike the case of automatic structures, where quotients by automatic congruence relations give no new structures.

The group $(\mathbb{Z}; +)$ is an example of a continuous generalised sequential automatic algebra. Here we represent numbers in base -2, allowing us to add integers without knowing their signs beforehand.

If $\mathcal{A}$ is (strictly, continuous generalised, generalised) sequential automatic and $\mathcal{B}$ is isomorphic to $\mathcal{A}$ then we call $\mathcal{A}$ a **(strictly, continuous generalised, generalised) sequential automatic presentation of** $\mathcal{B}$. Automatic presentations are defined similarly. Often we abuse our definitions and refer to algebras that have sequential automatic presentations as sequential automatic algebras, or structures with automatic presentations as automatic structures. When we describe an algebra as being automatic, it is to be taken as implicit that we are considering it as a relational structure.

A brief outline of the paper is as follows. Section 1 describes basic properties of the four classes of sequential automatic algebras and provides examples to separate them. The section also provides a classification theorem for generalised sequential algebras in the cases of finitely generated groups, Boolean algebras, and ordinals. Section 2 proves that if linearly ordered sets are defined as sequential automatic algebras then the order must be obtained from the lexicographic order on strings. This implies that the monadic second order theory of each sequential automatic linear order algebra is decidable. This also implies, from the result of Kuske [13], there exists an automatic linear order not isomorphic to a sequential automatic linear order algebra. Section 3 studies sequential automatic unary algebras and proves that the reachability problem for such algebras is decidable. This contrasts with automatic unary algebras, where the reachability problem is undecidable [17], [4]. An example is given of a (unary) permutation algebra that is automatic as a relational structure, but has no presentation as a sequential automatic algebra.

Finally, we stress that the goal is to show that sequential automatic algebras are more tame structures than their automatic counterparts. The ultimate goal in the study of sequential automatic algebras is to investigate whether or not natural problems asked about sequential automatic algebras, e.g. the isomorphism and the elementary equivalence problems, are decidable. This is the first paper devoted to this study.

1 General Properties, Separating Examples and Classification Results

The first part of following proposition implies that all decidability properties enjoyed by the class of automatic structures are present for automatic sequential algebras (see [10]). The second part states that sequential automatic algebras are closed under finite Cartesian products; the proof is straightforward:

Proposition 1. (1) *Every generalised sequential automatic algebra is automatic. In particular, there is an algorithm which, given a generalised sequential automatic algebra $\mathcal{A}$, a first order formula $\phi(\bar{x})$, and a tuple $\bar{a} \in A$, decides if $\mathcal{A} \models \phi(\bar{a})$.* (2) *If $\mathcal{A}_1, \ldots, \mathcal{A}_k$ are (continuous generalised, generalised or strictly) sequential automatic then so is $\mathcal{A}_1 \times \ldots \times \mathcal{A}_k$.* □

Now we separate $\mathcal{SSA}$ from $\mathcal{SA}$, $\mathcal{SA}$ from $\mathcal{CGSA}$, and $\mathcal{CGSA}$ from $\mathcal{GSA}$. The proof of the following proposition is easy:

Proposition 2. *If $\mathcal{A} = (A; f_1, \ldots, f_n)$ is strictly sequential automatic then for each $a \in A$ and f_i the set $f^{-1}(a) = \{\bar{x} \mid f_i(\bar{x}) = a\}$ is finite. Hence, algebras from the following classes are strictly sequential automatic iff they are finite: groups, rings, Boolean algebras, lattices with complements, vector spaces.* □

Corollary 1. *$\mathcal{SSA}$ is a proper subset of $\mathcal{SA}$.*

Proof. The algebra $(\omega; f)$, where $f(x) = 0$ for all $x \in \omega$, is a separator. □

Proposition 3. *A group is sequential automatic iff it is finite. Hence $\mathcal{SA}$ is a proper subset of $\mathcal{CGSA}$.*

Proof. Let n be the length of the string for the unit element 1. When the automaton for multiplication reads a pair (x, x^{-1}) of length $> n$, it halts and outputs 1 based on prefixes of length n. If the group were infinite, there would be infinitely many such pairs and finitely many such prefixes, and so distinct (x, x^{-1}) and (y, y^{-1}) exist which share the same prefixes. Then (x, y^{-1}) therefore would output 1. A separating witness is the group $(\mathbb{Z}; +)$. □

Definition 4. *An algebra $\mathcal{A}$ is* **residually finite** *if for all distinct $x, y \in \mathcal{A}$ there is a homomorphism $\phi : \mathcal{A} \to F$ onto a finite algebra F such that $\phi(x) \neq \phi(y)$.*

Proposition 4. *All algebras in the class $\mathcal{CGSA}$ are residually finite.*

Proof. We first show this for sequential automatic algebras $\mathcal{A} = (A; f_1, \ldots, f_n)$. For $n \in \omega$, define ϕ_n to be the function such that for a string $w = \sigma_0 \ldots \sigma_k$, if $k \leq n$ then $\phi_n(w) = w$, otherwise $\phi_n(w) = \sigma_0 \ldots \sigma_n$. Define $\mathcal{F}_n = (F; f_1', \ldots, f_n')$, where F is the image of $\mathcal{A}$ under ϕ_n and $f_i' = \phi_n \circ f_i \circ \phi_n^{-1}$; ϕ_n is a homomorphism from $\mathcal{A}$ to $\mathcal{F}_n$. Given two elements x, y of $\mathcal{A}$, let $n = \max(|x|, |y|)$ be the length of the longest string. Then $\phi_n : \mathcal{A} \to \mathcal{F}_n$ is such that $\phi_n(x) \neq \phi_n(y)$. For a continuous generalised sequential algebra $\mathcal{A}/E$, we construct $\mathcal{F}$ from $\mathcal{A}$ as above, and then take $\mathcal{F}/E'$ where E' is such that $x \cong_{E'} x'$ iff there exist $y, y' \in \mathcal{A}$ such that $y \cong_E y'$ and $\phi(y) = x$, $\phi(y') = x'$. □

Corollary 2. *$\mathcal{CGSA}$ is a proper subclass of $\mathcal{GSA}$.*

Proof. The algebra $\mathcal{A} = (\omega; f)$, with $f(0) = 0$ and $f(n) = n-1$ is generalised sequential automatic but not residually finite and hence not continuous generalised sequential automatic.

This additive group of rational numbers is not residually finite. Hence:

Corollary 3. *The group $(\mathbb{Q}; +)$ does not belong to $\mathcal{CGSA}$.* □

Now we give a classification result for the classes of finitely generated groups, Boolean algebras, and ordinals as continuous generalised sequential automatic algebras. These have been classified as automatic structures (see [12], [8]). We recall some definitions. For groups, a finitely generated group is **virtually abelian** if it contains an abelian subgroup of finite index. A finitely generated group is an automatic structure iff it is virtually abelian [14]. For Boolean algebras, B_ω denotes the Boolean algebra of finite and co-finite subsets of ω. An infinite Boolean algebra is an automatic structure iff it is isomorphic to a finite Cartesian product of B_ω [12]. For ordinals, an ordinal is automatic iff it is strictly less than ω^ω [17]. In order to treat ordinals as algebras we introduce the concept of a linear order algebra. Let $\mathcal{L} = (L; \leq)$ be a linearly ordered (lo) set. Define the function $f : L^2 \to L$ as $f(x, y) = \min(x, y)$. This function has the following properties for all $x, y, z \in L$: **1)** $f(x, y) = f(y, x)$; **2)** $f(x, y) = x$ or $f(x, y) = y$; **3)** if $f(x, y) = x$ and $f(y, z) = y$ then $f(x, z) = x$. Given any function $f : A \to A$ satisfying these conditions, we define a lo set by $x \leq_f y$ iff $f(x, y) = x$. We call algebras $(A; f)$, where f satisfies the above conditions, **linear order (lo) algebras**. These transformations between lo sets and lo algebras preserve the isomorphism type, and a lo set is automatic iff the corresponding lo algebra is automatic (as relational structures).

Definition 5. *Let $\mathcal{L}_1, \mathcal{L}_2$ be two linear order algebras. The algebra $\mathcal{L}_1 +_< \mathcal{L}_2$ is over $L_1 \cup L_2$ such that if $x \in L_1$ and $y \in L_2$ then $\min(x, y) = \min(y, x) = x$, and otherwise $\min(x, y)$ follows from L_1 or L_2. The algebra $\mathcal{L}_1 \times_< \mathcal{L}_2$ is over the Cartesian product of L_1 and L_2 such that $\min((x_1, y_1), (x_2, y_2))$ is (x_1, y_1) iff $\min(x_1, x_2) = x_1$, or $x_1 = x_2$ and $\min(y_1, y_2) = y_1$.*

Proposition 5. *Given two sequential automatic linear order algebras $\mathcal{L}_1, \mathcal{L}_2$, the algebras $\mathcal{L}_1 +_< \mathcal{L}_2$ and $\mathcal{L}_1 \times_< \mathcal{L}_2$ are sequential automatic. In particular, all ordinals less than ω^ω are sequential automatic linear order algebras.* □

Theorem 1. *1. A finitely generated group is continuous generalised sequential automatic iff it is virtually abelian.*
2. An infinite Boolean algebra is continuous generalised sequential automatic iff it is isomorphic to a finite Cartesian product of B_ω to itself.
3. An ordinal (as an lo algebra) is sequential automatic iff it is strictly below ω^ω.

Proof. For part 1), the proof essentially follows [8]. Let G be a finitely generated virtually abelian group. We can assume that that our group G has a finitely generated free abelian normal subgroup N of finite index. Then any element $g \in G$ can be given in the form $g = fn$ where $f \in F = G/N, n \in N$. Given two elements $g_1, g_2 \in G$, then $g_1 g_2 = f_1 n_1 f_2 n_2 = f_1 f_2 f_2^{-1} n_1 f_2 n_2$. Since N is normal, $\phi_f(n) = f^{-1} n f$ gives an automorphism of N for any f. Therefore when we multiply $f_1 n_1$ by $f_2 n_2$ we get $(f_1 f_2)n$ where $n = \phi_{f_2}(n_1) n_2 \in N$.

We describe in general terms the sequential automatic encoding of G. We represent group elements as a pair (f, n) where $f \in F$ and $n \in N$. Because F is finite and the F part of the product depends only on the F part of the two inputs, we can encode f in the first digit. The rest of the string encodes n. When computing the product $(f_1, n_1)(f_2, n_2)$, the first state outputs $f_1 f_2$ and then branches to one of finitely many subautomata depending on f_2.

We need to show that a sequential automaton can compute $n_1 \phi_{f_2}(n_2)$. The subgroup N is, by assumption, isomorphic to $\mathbb{Z}^r$ for some $r \in \omega$. Using the encoding of $\mathbb{Z}$ and Proposition 1, we can encode elements of N and compute their sums. All automorphisms of $N \cong \mathbb{Z}^r$ can be determined from linear extension of their action on a minimal generating set, and therefore correspond to matrix multiplications. That means we can associate with each $f \in F$ an integer matrix M_f which computes the automorphism ϕ_f on the vector representation of N. Multiplication by M_f can be computed by a sequential automaton.

To compute $n_1 \phi_f(n_2)$, we combine the addition automaton for N and the automaton for M_f, so that the first input string is given straight to the N automaton, and the second input string is processed by the M_f automaton, the output of which is given as the second input of the N automaton. Now one uses the construction described to give a sequential presentation of G.

For part 2), by Proposition 1 we need to consider B_ω. We encode subsets of ω in binary. The final character of a binary word is interpreted as being recurring (so when $\square$ is read following a 0(1) it is treated as a 0(1)). Our language can code any finite or co-finite set. Given this encoding, the algebra B_ω is a continuous generalised sequential algebra.

The last part of the theorem follows from Proposition 5. $\square$

2 Linear Order Algebras

Here we investigate sequential automatic linear order (lo) algebras. The lexicographical order will play an essential role in describing these algebras.

Example 1. *Let L be a regular language. Consider $\preceq_L$, the lexicographic order restricted to L. The algebra $(L; \min_{\preceq})$ is sequential automatic.* $\square$

As a corollary, one obtains that the dense order (the order of rational numbers) is a sequential automatic lo algebra; this is isomorphic to the $\preceq$ order on the set $\{w101 \in \{0,1\}^\star \mid w \text{ does not have 101 as a substring}\}$.

Example 2. *An infinite sequential automatic lo algebra over a* unary alphabet *is isomorphic to $\omega + n$, $n + \omega^\star$, or $\omega + \omega^\star$, where $\omega^\star$ is ω reversed.* $\square$

Theorem 2. *A linear order algebra is sequential automatic iff it is isomorphic to the lexicographic order on a regular language.*

Proof. One direction is explained in Example 1. The idea for the other direction is that we have to decide which string is the minimum at the first position where they differ, because then we have to output the next digit from either one or the other. This is like the lexicographic order. Given a sequential automatic lo algebra $\mathcal{L} = (L; f)$ we explicitly construct a regular language and an isomorphism. Let $\mathcal{M}$ denote the automaton for recognising L. For the isomorphism, we use a slightly generalised form of sequential automata: instead of computing a function from Σ^* to itself, we have two alphabets Σ and Σ' and compute a function from Σ^* to $(\Sigma')^*$. Let Σ be the alphabet for $\mathcal{L}$, and $\Sigma' = \{0, \ldots, |\Sigma|\}$.

For distinct strings w_1, w_2, let i be the first position where they differ. If one is a prefix of the other, then this position is immediately after the end of this string, when a blank symbol $\square$ is first read. Let $u = f(w_1, w_2)$ be the output string. For $j < i$, $w_1(j) = w_2(j) = u(j)$. At position i there is a choice: $u(i)$ must equal either $w_1(i)$ or $w_2(i)$, determining the rest of the string. The ordering of w_1 and w_2 gives an ordering on the pair of symbols $\{w_1(i), w_2(i)\}$; any pair of inputs that reach the current state of the automaton and read these symbols next must obey this ordering.

If we look at all pairs of symbols that can be read at this state, we get an ordering on each pair. We would like to combine these orderings to give an ordering on all of these symbols, and extend that to give an ordering on $\Sigma \cup \{\square\}$. However, it is possible to get orderings on pairs which are not consistent (that is, they cannot be simultaneously true in a linear ordering, for example $a < b$, $b < c$, $c < a$). We deal with this by keeping track of the state of $\mathcal{M}$. Given a state q of $\mathcal{M}$, let Σ_q be the set of all symbols (including $\square$) which can be read from q as part of a path to a final state of $\mathcal{M}$. If we have a reachable state (q, r) in the product of $\mathcal{M}$ and the automaton for f, the orderings given by r for pairs in Σ_q must be consistent and so we can combine them to give an ordering on Σ_q and then extend this to an ordering on $\Sigma \cup \{\square\}$.

We construct a sequential automaton $\mathcal{A}$ mapping $L \subset \Sigma^*$ into $(\Sigma')^*$. Let the states be the product of $\mathcal{M}$ and the automaton for f. When a symbol σ is read, we treat the f part as being given the input (σ, σ) and the $\mathcal{M}$ part as being given σ, and output the element of $\{0, \ldots, n\}$ corresponding to the position of σ in the ordering on $\Sigma \cup \{\square\}$ associated with the pair of states. The final output function for each state outputs the element of $\{0, \ldots, n\}$ corresponding to the position of $\square$ in this ordering.

From the construction of $\mathcal{A}$, it preserves the order of $\mathcal{L}$ when we take the lexicographic ordering on its image, and the function computed by $\mathcal{A}$ is injective. It remains to show that the image of L under this function is a regular language over Σ'. Construct another automaton by taking $\mathcal{A}$ and swapping the inputs and outputs on the transitions. We add a single final state q_0, and from each state $p = (q, r)$, where q is a final state in $\mathcal{M}$, we add a transition to q_0 when $O(p)$ (the final output function in $\mathcal{A}$) is read. This gives an automaton which

recognises exactly $\mathcal{A}(L)$. Therefore $\mathcal{A}$ is an isomorphism from L to a regular language with the lexicographic order. □

Corollary 4. *The monadic second order theory of any sequential linear order algebra is decidable.*

Proof. From the theorem above, the result follows from the fact that the lexicographical order is MSO definable in the binary tree [16]. □

Corollary 5. *There exists an automatic linear order* $(L; \leq)$ *which is not isomorphic to any sequential automatic linear order algebra.*

Proof. In [13] Kuske constructed an automatic linear order that is not isomorphic to the lexicographical order on any regular set. □

The third application is this. A *word* is $(\omega; \leq, P)$, where P is a unary relation. A word is automatic if it is isomorphic to an automatic structure. The word $(\omega; \leq, P)$ can be transformed into the following algebra $(\omega; \min_{\leq}, f_P)$, where $f_P(x)$ is the maximum y such that $y \leq x$ and $P(y)$. Call this a word algebra. Two words are isomorphic iff their corresponding word algebras are isomorphic.

Corollary 6. *If* $(\omega; \min_{\leq}, f_P)$ *is a sequential automatic word algebra then* $(\omega; \leq, P)$ *is a morphic word. Hence its monadic second order theory is decidable.*

Proof. The order can be replaced with the length-lexicographic order in which P is still regular. From results in [2], it follows that P is a characteristic of a morphic word. The rest follows from Thomas and Carton in [8]. □

3 Sequential Automatic Unary Algebras

Here we consider algebras of the form $\mathcal{A} = (A; f_1, \ldots, f_n)$, where each f_i is a unary function on A. These algebras are called **unary algebras**. A unary algebra can be transformed into a graph (A, E) where $E(x, y)$ iff $f_1(x) = y \vee \ldots \vee f_n(x) = y$. The **reachability problem** for the unary algebra $\mathcal{A}$ is the set $\{(x, y) \mid$ in the graph (A, E) there is a path from x to $y\}$. For unary automatic algebras even with one unary operation the reachability problem is Σ_1^0-complete [17] [4]. In contrast, we have:

Theorem 3. *If* $\mathcal{A} = (A; f_1, \ldots, f_n)$ *is a sequential automatic unary algebra then the reachability problem for* $\mathcal{A}$ *is decidable in PSPACE.*

Proof. Suppose we want to check if (x, y) is in the reachability relation, and that y has length n. If $f(z)$ has length $|f(z)| \leq n$, then either $|z| \leq n$ or f terminates early, after at most n digits are read. In the second case, any string with the same first n digits as z would give the same output. Since all our functions f_i are sequential, we need only consider the first $n + 1$ digits when determining if y is reachable. We need one extra digit to make sure we don't mistake a string of which y is a prefix for y itself. The algorithm is a straightforward search: begin

at x, for each f_i branch to search on the first $n+1$ digits of $f_i(x)$, terminate a branch when a string is reached that has already been visited, terminate the algorithm either positively when y is reached, or negatively when all branches terminate. Since we only consider the first $n+1$ digits, the search space is finite, so the algorithm will always terminate. □

Proposition 6. *There exists a sequential automatic unary algebra $\mathcal{A}$ in which the reachability relation is not recognised by pushdown automata.*

Proof. Consider the unary algebra $(A; f)$, where $A = 0^*1^*2^*$ and $f(0^i1^j2^k) = 0^{i+1}1^{j+1}2^{k+1}$. Let x be the string 012. The set of elements reachable from 012 is $\{0^n1^n2^n \mid n \in \omega\}$, which is not recognised by pushdown automata. □

An **algebra of permutations** is a unary algebra $(A; f_1, \ldots, f_n)$ where each $f_i : A \to A$ is a bijection.

Proposition 7. *There is a PSPACE algorithm that, given a strictly sequential automatic unary algebra $\mathcal{A} = (A; f_1, \ldots, f_n)$, where $A = \Sigma^\star$, decides if $\mathcal{A}$ is an algebra of permutations.*

Proof. Take a Mealy automaton $\mathcal{M}$ computing one of the operations of the algebra. Assume that all states are reachable. A strictly sequential Mealy automaton gives a permutation on $\Sigma^\star$ iff each state gives a permutation of its input and all of the final output strings are empty.

For each state $s \in \mathcal{M}$, we check whether the final output string $O(s)$ is empty, and whether for each pair of symbols $(\sigma_1, \sigma_2) \in \Sigma^2$, $\sigma_1 \neq \sigma_2$ the output when σ_1 is read in state s is distinct from when σ_2 is read. The space required is a counter to run through the states which is proportional to $\log(|\mathcal{M}|)$. □

Theorem 4. *(see also [9]): The length of a cycle on strings of length n of a strictly sequential automatic permutation over Σ is of the form $\prod_{i=1}^{n} a_i$ where $1 \leq a_i \leq |\Sigma|$.*

Proof. There are at most $|\Sigma|$ strings of length 1, and since these are permuted amongst themselves, cycles on strings of length 1 have lengths $1 \leq L \leq |\Sigma|$.

We proceed by induction. Given a cycle of length L_n on strings of length n, we consider these strings truncated to the first $n-1$ digits. These strings, by the induction hypothesis, are in a cycle of length $L_{n-1} = \prod_{i=1}^{n-1} a_i$ where $1 \leq a_i \leq |\Sigma|$. Truncating to the first $n-1$ digits gives a homomorphism, so L_{n-1} must divide L_n. Since L_n/L_{n-1} cannot exceed $|\Sigma|$, the result follows. □

Corollary 7. *There are algebras of permutations which are automatic but not strictly sequential automatic.*

Proof. Let $p : \omega \to \omega$ be any nonzero polynomial with positive coefficients. Let L be a regular language over Σ such that $growth_L(n) = |L \cap \Sigma^n| = p(n)$ for all n. Such languages exist (see [15] or [11]). Define the function $f : L \to L$ as follows: If $x \in L$ is not the largest length-lexicographically among all $y \in L$ with

$|x| = |y|$, then $f(x)$ is the length-lexicographically least element $z \in L$ greater than x such that $|x| = |z|$; if $x \in L$ is the length-lexicographically greatest among all $y \in L$ with $|x| = |y|$, then $f(x)$ is the length-lexicographically least element $z \in L$ such that $|z| = |x|$. The structure (L, f) is an automatic relational structure. The range of p coincides with the set of all lengths of cycles of the permutation f. By the theorem above, (L, f) cannot be isomorphic to a strictly sequential automatic permutation. □

It is an open question if there is an algorithm that, given two sequential automatic permutations, decides whether the permutations are isomorphic.

References

1. Aleshin, M.: Finite automata and Burnside's problem for periodic groups. Mat. Notes 11, 199–203 (1972)
2. Bárány, V.: A Hierarchy of Automatic Words having a Decidable MSO Theory. In: Caucal, D. (ed.) Online Proceedings of the 11th Journées Montoises, Rennes (2006)
3. Béal, M., Carton, O., Prieur, C., Sakarovitch, J.: Squaring transducers: an efficient procedure for deciding functionality and sequentiality. Theor. Comput. Sci. 292(1), 45–63 (2003)
4. Blumensath, A.: Automatic Structures. Diploma Thesis, RWTH Aachen (1999)
5. Blumensath, A., Grädel, E.: Automatic Structures. In: 15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, pp. 51–62 (2000)
6. Blumensath, A., Grädel, E.: Finite presentations of infinite structures: automata and interpretations. Theory of Computing Systems 37(6), 641–674 (2004)
7. Cannon, J., Epstein, D., Holt, D., Levy, S., Paterson, M., Thurston, W.: Word processing in groups. Jones and Bartlett (1992)
8. Carton, O., Thomas, W.: The monadic theory of morphic infinite words and generalizations. Information and Computation 176, 51–76 (2002)
9. Grigorchuk, R., Nekrashevich, V., Sushanski, V.: Automata, Dynamical systems, and groups. Tr. Mat. Inst. Steklova 231(Din. Sist. Avtom i Beskon. Gruppy), 134–214 (2000)
10. Khoussainov, B., Nerode, A.: Automatic Presentations of Structures. LNCS, vol. 960, pp. 367–392. Springer, Heidelberg (1995)
11. Khoussainov, B., Rubin, S.: Automatic Structures: Overview and Future Directions. Journal of Automata, Languages and Combinatorics 8, 287–301 (2003)
12. Khoussainov, B., Nies, A., Rubin, S., Stephan, F.: Automatic Structures: Richness and Limitations. In: LICS, pp. 44–53. IEEE Computer Society, Los Alamitos (2004)
13. Kuske, D.: Is Cantor's theorem automatic?. LNCS, vol. 2850, pp. 332–343. Springer, Heidelberg (2003)
14. Oliver, G., Thomas, R.: Automatic Presentations for Finitely Generated Groups. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 693–704. Springer, Heidelberg (2005)
15. Saloma, A., Soittola, M.: Automata-theoretic Aspects of Formal Power Series. Springer, Heidelberg (1978)
16. Rabin, M.: Decidability of second order theories and automata on infinite trees. Trans. AMS 141, 1–35 (1969)
17. Rubin, S.: Automatic Structures. PhD Thesis, University of Auckland (2004)