Uniform decision problems for automatic semigrouos

Kambites, Mark and Otto, Friedrich

2006

MIMS EPrint: **2007.29**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

Available online at www.sciencedirect.com
ScienceDirect
Journal of Algebra 303 (2006) 789–809
JOURNAL OF Algebra
www.elsevier.com/locate/jalgebra

Uniform decision problems for automatic semigroups ☆

Mark Kambites *, Friedrich Otto

Fachbereich Mathematik/Informatik, Universität Kassel, 34109 Kassel, Germany

Received 7 October 2005

Available online 20 December 2005

Communicated by Derek Holt

Abstract

We consider various decision problems for automatic semigroups, which involve the provision of an automatic structure as part of the problem instance. With mild restrictions on the automatic structure, which are necessary to make the problem well defined, the uniform word problem for semigroups described by automatic structures is decidable. Under the same conditions, we show that one can also decide whether the semigroup is completely simple or completely zero-simple; in the case that it is, one can compute a Rees matrix representation for the semigroup, in the form of a Rees matrix together with an automatic structure for its maximal subgroup. On the other hand, we show that it is undecidable in general whether a given element of a given automatic monoid has a right inverse.

1. Introduction

Over the past two decades, one of the most successful and productive areas of computational algebra has been the theory of *automatic groups*. Roughly speaking, the description

☆ This research was supported by a Marie Curie Intra-European Fellowship within the 6th European Community Framework Programme.

* Corresponding author.
E-mail addresses: kambites@theory.informatik.uni-kassel.de (M. Kambites), otto@theory.informatik.uni-kassel.de (F. Otto).

0021-8693/$ – see front matter
doi:10.1016/j.jalgebra.2005.11.028

of a group by an automatic structure allows one efficiently to perform various computations involving the group, which may be hard or impossible given only a presentation. Groups which admit automatic structures also share a number of interesting structural and geometric properties [9]. More recently, many authors have followed a suggestion of Hudson [16] by considering a natural generalisation to the broader class of monoids or, even more generally, of semigroups, and a coherent theory has begun to develop [4–8,11–14, 22–26].

A number of authors have considered decision problems in automatic semigroups; for example, it has been established that the word problem for an automatic semigroup is always decidable in quadratic time [5] and in certain cases is P-complete [19]. In general, this research has assumed a fixed semigroup with automatic structure, and asked what computations can be performed in the semigroup. Since an automatic structure is a finite description of a (typically infinite) semigroup, one is also able to consider decision problems in which a semigroup, defined by means of an automatic structure, forms part of the problem instance. Some such problems, such as the isomorphism problem and the uniform word problem, are also studied by group theorists. Others are more particular to semigroup theory; for example, one can ask if there are algorithms to decide, given an automatic structure, whether the semigroup described is a group, is completely simple, is completely zero-simple, has a left/right/two-sided zero or identity, is right/left/two-sided cancellative and so forth.

The aim of this paper is to make a start upon addressing these issues. We begin, in Section 2 by developing a suitable theoretical foundation for the study of uniform problems involving automatic semigroups. In Sections 3 and 4 we present a number of algorithms for basic problems involving automatic semigroups; these include, amongst others, the uniform word problem and deciding the properties of right cancellativity, the existence of an identity and the existence of a zero. In Section 5, by contrast, we show that the existence of right inverses for a given element of a given automatic semigroup is not, in general, decidable. Finally, in Section 6, we present algorithms to decide if a given semigroup is completely simple or completely zero-simple and, in the event that it is, obtain a Rees matrix decomposition of the semigroup.

While this paper is theoretical in nature, the research documented also has a practical aspect. We aim not only to improve our understanding of computational issues involving automatic semigroups, but also to develop practical tools which will be of use to those working in the field. All the algorithms documented in this paper, along with a number of others, have been implemented by the first author using the GAP computer algebra system [10]. They will shortly be made available in the form of a GAP package, as a resource for researchers in the area.

2. Foundations

In this section, we begin by recalling some basic definitions which we shall require in the sections that follow. We then proceed to develop a suitable formalism for the study of uniform problems involving automatic structures. We assume a basic familiarity with finite automata; the reader with no experience in this area is advised to consult a textbook

such as [18]. Throughout this paper, we write $\mathbb{N}_0$ and $\mathbb{N}_+$ to denote the sets of nonnegative integers and strictly positive integers, respectively, and ϵ to denote the empty word.

2.1. Synchronous automata recognising relations

Let A be a finite alphabet, let \$ be a new symbol not in A, and let $A^{\$}$ be the alphabet $A \cup \{\$\}$. We define a function $\delta : A^* \times A^* \to (A^{\$} \times A^{\$})^*$ as follows. For $m, n \in \mathbb{N}_0$ and $a_1, \ldots, a_m, b_1, \ldots, b_n \in A$, let

$$\delta(a_1 \ldots a_m, b_1 \ldots b_n) = \begin{cases} (a_1, b_1) \ldots (a_m, b_m)(\$, b_{m+1}) \ldots (\$, b_n) & \text{if } m < n, \\ (a_1, b_1) \ldots (a_m, b_m) & \text{if } m = n, \\ (a_1, b_1) \ldots (a_n, b_n)(a_{n+1}, \$) \ldots (a_m, \$) & \text{if } m > n. \end{cases}$$

Intuitively, δ rewrites *pairs of words* over A as *words of pairs* over $A^{\$}$.

A *synchronous automaton* over A is a finite automaton over $(A^{\$} \times A^{\$})^*$. We say that such an automaton *recognises* or *accepts* a pair $(u, v) \in A^* \times A^*$ if it recognises $\delta(u, v)$; it recognises a relation $R \subseteq A^* \times A^*$ if it recognises exactly the image $\delta(R)$. A relation recognised by a synchronous automaton is called *synchronously rational*. The following proposition summarises some basic properties of synchronously rational relations, which we shall use throughout this paper without further comment; all are proved in [9].

Proposition 2.1. *The class of synchronously rational relations is closed under intersection, union, complement, composition and inversion. Synchronously rational relations are rational transductions, and hence preserve rational languages*; *in particular the projection of a synchronously rational relation onto either coordinate is a rational language, and the image of an element under a synchronously rational relation is a rational language.*

Moreover, all of these operations are effectively computable.

We shall also make use of the following fact, again without further comment.

Proposition 2.2. *There is an algorithm which, given two synchronous automata, decides if they accept the same relation.*

Proof. Two synchronous automata accept the same relation exactly if, when considered as normal finite automata, they accept the same language over $(A^{\$} \times A^{\$})^*$. The latter property is well known to be decidable (for example, by computing minimal deterministic automata). □

2.2. Automatic structures and interpretations

A (*synchronous*) *pre-automatic structure* Γ consists of

(i) a finite set $A(\Gamma)$ of *generators*;
(ii) a finite automaton recognising a language $L(\Gamma) \subseteq A(\Gamma)^+$;

(iii) a synchronous automaton recognising a relation $L_=(\Gamma)$ on $A(\Gamma)^+$, which is contained in $L(\Gamma) \times L(\Gamma)$; and
(iv) for each $a \in A(\Gamma)$, a synchronous automaton recognising a relation $L_a(\Gamma)$ on $A(\Gamma)^+$ which is contained in $L(\Gamma) \times L(\Gamma)$.

Where only one pre-automatic structure is under discussion, we shall for brevity write simply A, L, $L_=$ and L_a in place of $A(\Gamma)$, $L(\Gamma)$, $L_=(\Gamma)$ and $L_a(\Gamma)$, respectively.

An *interpretation* of a pre-automatic structure with respect to a semigroup S is a morphism $\sigma : A^+ \to S$ such that

(i) $\sigma(L) = S$;
(ii) for $u, v \in L$ we have $(u, v) \in L_=$ if and only if $\sigma(u) = \sigma(v)$; and
(iii) for each $a \in A$ and $u, v \in L_a$ we have $(u, v) \in L_a$ if and only if $\sigma(ua) = \sigma(v)$.

If such an interpretation exists, we say that the pre-automatic structure is an *automatic structure for* S, or that S is *described* by the automatic structure. A semigroup S is called *automatic* if it is described by some automatic structure. If, in addition, the interpretation restricts to a bijection from L to S, we say that the automatic structure has *uniqueness*, or is an *automatic cross-section* for S.

Proposition 2.3. *Two semigroups described by the same automatic structure are necessarily isomorphic.*

Proof. Suppose $\sigma : A^+ \to S$ and $\tau : A^+ \to T$ are interpretations of the same automatic structure. Define a map $\rho : S \to T$ by setting $\rho(s)$ to be the unique $t \in T$ such that $t = \tau(w)$ for some $w \in L$ with $\sigma(w) = s$. It is straightforward to verify that this map is a well-defined isomorphism of the semigroups. □

Proposition 2.3 tells us that an automatic structure uniquely defines a semigroup S up to isomorphism, but it does *not* guarantee that it uniquely defines an interpretation, even up to isomorphism. In general, an automatic structure need not contain sufficient information to associate to each generator a unique member of the language of representatives which represents the same element in every interpretation. As an elementary example, we consider a pre-automatic structure Γ with $A(\Gamma) = \{a, b, c\}$, $L(\Gamma) = \{a, b\}$,

$$L_=(\Gamma) = \big\{(a, a), (b, b)\big\} \quad \text{and} \quad L_a(\Gamma) = L_b(\Gamma) = L_c(\Gamma) = \big\{(a, b), (b, b)\big\}.$$

Clearly, Γ is an automatic structure for the two-element semigroup S with presentation

$$\langle a, b \mid aa = ab = ba = bb = b \rangle.$$

Define morphisms $\rho, \rho' : A^+ \to S$ by $\rho(a) = \rho'(a) = a$, $\rho(b) = \rho'(b) = b$ and $\rho(c) = a$ but $\rho'(c) = b$. It is straightforward to verify that ρ and ρ' are distinct interpretations of Γ. Indeed, since S has no nontrivial isomorphisms, they are not even equivalent up to isomorphic permutation of S.

We will need to perform computations not only with automatic structures but also with interpretations; for this we require a finite way of encoding of an interpretation. Formally, we say that an *assignment of generators* for the automatic structure with respect to a semigroup S is a function $\iota : A \to L$ with the property that there exists an interpretation $\sigma : A^+ \to S$ such that $\sigma(a) = \sigma(\iota(a))$ for all $a \in A$; such an interpretation is said to be *consistent* with ι. Two assignments of generators ι and ι' are called *equivalent* if $(\iota(a), \iota'(a)) \in L_=$ for every generator a. An *interpreted automatic structure* for a semigroup S is a pair (Γ, ι) of an automatic structure together with an assignment of generators with respect to S. This terminology is justified by the following proposition which is straightforward to prove.

Proposition 2.4. *An assignment of generators is consistent with a unique interpretation* (*up to isomorphic permutation of the semigroup described*). *Moreover, equivalent assignments of generators are consistent with the same interpretation.*

Conversely, an interpretation for an automatic structure is consistent with an assignment of generators, which is unique up to equivalence.

Given an automatic structure and a word $w = a_1 a_2 \dots a_n \in A^+$ with each $a_i \in A$, we define

$$L_w = L_{a_1} \circ L_{a_2} \circ \cdots \circ L_{a_n},$$

where $\circ$ denotes composition of relations. We extend this definition to the whole of A^* by letting $L_\epsilon = L_=$. By our observations above, we can compute a synchronous automaton recognising the language L_w for any $w \in A^*$. It is readily verified [9] that for any interpretation $\sigma : A^+ \to S$ of the automatic structure, we have

$$L_w = \big\{(u, v) \in L \mid \sigma(uw) = \sigma(v)\big\}.$$

2.3. Interpreted vs uninterpreted automatic structures

The distinction between automatic structures with and without interpretation is important for two reasons. Firstly, certain problems, such as the uniform word problem, involve elements of the semigroup expressed as words in the generators as part of the problem instance; these problems are not necessarily well defined in the absence of an interpretation. Secondly, even invariant properties of the semigroup, such as whether it has an identity, can be more straightforward to test when an assignment of generators is provided. We presently lack an algorithm which, given an automatic structure, computes an assignment of generators; in general it is not clear whether such an algorithm exists.

Question 2.5. *Is there an algorithm which, given an automatic structure, finds an assignment of generators which is consistent with some interpretation*?

However, there is a very large class of semigroups for which automatic structures admit essentially unique interpretations, and in which such an algorithm does exist.

We say that two elements s and t of a semigroup S are *right translationally equivalent* if $xs = xt$ for every element $x \in S$. Recall that a semigroup is called *left reductive* if no two distinct elements are right translationally equivalent [27, p. 84]. The class of left reductive semigroups is very large, including all monoids, left cancellative semigroups, inverse semigroups and of course groups. An example of a semigroup which is **not** left reductive is a nontrivial semigroup all of whose elements are left zeros.

Proposition 2.6. *Given an automatic structure, right translational equivalence is independent of the choice of interpretation, that is, words u and v represent right translationally equivalent elements in all interpretations or in none.*

Moreover, there is an algorithm which, given as input an automatic structure and two words u and v in the generators, decides if u and v represent right translationally equivalent elements.

Proof. It is readily verified that u and v are right translationally equivalent in any interpretation if and only if the relations L_u and L_v are equal. These relations can be computed independent of the interpretation, by composition of relations of the form L_a for various $a \in A$. We can then solve the problem by testing them for equality. □

Corollary 2.7. *An automatic structure for a left reductive semigroup admits a unique interpretation (up to isomorphism of the semigroup described). Moreover, there is an algorithm which, given as input an automatic structure for a left reductive semigroup, computes an assignment of generators.*

Proof. Suppose ι and ι' are assignments of generators and $a \in A$. Then $\iota(a)$ is right translationally equivalent to a, and hence to $\iota'(a)$. Since the semigroup is left reductive, it follows that $\iota(a)$ and $\iota'(a)$ represent the same element, and so we must have $(\iota(a), \iota'(a)) \in L_=$. Thus, ι and ι' are equivalent, and so by Proposition 2.4, they are consistent with the same unique interpretation.

To compute the interpretation, for each generator a we enumerate all words in L until we find a word $w \in L$ which is right translationally equivalent to a, and set $\iota(a) = w$. □

2.4. Modifying automatic structures

We shall make use of the following propositions, which allow us, given an interpreted automatic structure, to obtain automatic structures with nicer properties, for the same semigroup. They are essentially algorithmic restatements of results from [5,9].

Proposition 2.8. *There is an algorithm to solve the following problem*:

Instance: *an interpreted automatic structure with language of representatives L admitting an interpretation $\sigma : A^+ \to S$ and a finite automaton recognising a regular language $K \subseteq A^+$ such $K \setminus L$ is finite and $\sigma(K) = S$*;

Problem: *compute an automatic structure admitting the same interpretation and with language of representatives K.*

Proof. By [5, Propositions 5.3 and 5.7] there exists such an automatic structure. Moreover, the proofs of those results give an effective method of construction. □

Proposition 2.9. *There is an algorithm which, given as input an interpreted automatic structure* (Γ, ι) *for a semigroup* S*, computes an intepreted automatic structure* (Ω, ι') *for* S *such that*

(i) $A(\Omega) \subseteq A(\Gamma)$;
(ii) ι' *is the restriction of* ι *to* $A(\Omega)$;
(iii) *the interpretation induced by* ι' *restricts to a bijection between* $L(\Omega)$ *and* S; *and*
(iv) $A(\Omega) \subseteq L(\Omega)$.

Proof. First, we deal with the case that multiple of the generators represent the same element of S. Let B be a maximal subset of $A(\Gamma)$ in which no two elements represent the same element of S, that is, such that no two distinct elements of B are mapped by ι to words which are equivalent under the relation $L_=(\Gamma)$. Let ι' be the restriction of ι to B. Let $\alpha : A(\Gamma)^+ \to B^+$ be the unique morphism which takes each $a \in A(\Gamma)$ to the unique $b \in B$ such that $(\iota(a), \iota(b)) \in L_=(\Gamma)$. It is readily verified that applying α to $L(\Gamma)$, $L_=(\Gamma)$ and each $L_b(\Gamma)$ where $b \in B$ is an effective procedure which yields a new automatic structure in which no two generators represent the same element of S.

Now by Proposition 2.8 we may obtain an automatic structure whose language of representatives L contains the generators. Using the argument from [9, Theorem 2.5.1], we can construct an automaton recognising the language L' of words in L which are minimal with respect to the shortlex order (see [9, Section 2.5]) amongst words in L representating the same element of the semigroup. Clearly, the generators will be contained in L'. Now by Proposition 2.8, we obtain a suitable automatic structure with language of representatives L'. □

The previous proposition combines with the next one, to give us a more concise way to encode an interpreted automatic structure.

Proposition 2.10. *An automatic structure with the generators in the language of representatives admits a unique interpretation up to equivalence, and there is an algorithm which, given such an automatic structure, computes an appropriate assignment of generators.*

Proof. The identity function on the set of generators serves as an assignment of generators. It is easy to see that any other assignment of generators must be equivalent to this one. □

In view of Propositions 2.9 and 2.10, we may describe an interpretation of an automatic structure by including generators in the language of representatives; this avoids the need for explicit reference to the assignment of generators.

3. Algorithms for uninterpreted automatic structures

In this section, we consider decision problems for which the instance is an automatic structure without any further information about the assignment of generators.

First, we recall from [9, Section 5.1] that one can decide, starting only from a collection of synchronous automata, whether they form an automatic structure for a group. In particular, one can verify whether a given semigroup automatic structure describes a group. It is interesting to note that the corresponding problem with input specified as a finite presentation is known to be undecidable [20]. One wonders whether this difference results from (i) the difference in method of presentation, or (ii) the smaller class of semigroups under consideration. In particular, one might ask the following question.

Question 3.1. *Is there an algorithm to solve the following problem*?
Instance: *a finite presentation for a semigroup which is known to be automatic*;
Problem: *decide if the semigroup is a group.*

It is not presently known whether there is an algorithm which, given a finite presentation for an automatic semigroup, computes an automatic structure for the semigroup. If, in fact, there is such an algorithm, then the answer to the previous question is necessarily positive.

Another property which is easily decided is that of right cancellability. The following proposition and its corollary are essentially due to Silva and Steinberg [29, Proposition 9.4]. For completeness, we include a proof.

Proposition 3.2. *Let Γ be an automatic cross-section, and $\sigma : A^+ \to S$ an interpretation. Let $w \in A^+$. Then $\sigma(w)$ is right cancellable in S if and only if $L_w \circ L_w^{-1}$ is the diagonal relation on L.*

Proof. Let $\sigma : A^+ \to S$ be an interpretation. If $L_w \circ L_w^{-1}$ is *not* diagonal then there exist distinct words $u, v \in L$ such that $(u, v) \in L_w \circ L_w^{-1}$, that is, such that there exists $x \in L$ with $(u, x), (v, x) \in L_w$. But now $\sigma(u)\sigma(w) = \sigma(x) = \sigma(v)\sigma(w)$. But since the automatic structure is a cross-section, we have that $\sigma(u) \neq \sigma(v)$, so that $\sigma(w)$ is not right cancellable.

Conversely, if $\sigma(w)$ is not right cancellable, then $a\sigma(w) = b\sigma(w)$ for some distinct elements $a, b \in S$. Let u, v and x be words in L representing a, b and $a\sigma(w)$, respectively. Then we have $(u, x), (v, x) \in L_w$, so that $(u, v) \in L_w \circ L_w^{-1}$. Clearly $u \neq v$, so $L_w \circ L_w^{-1}$ is not diagonal. □

Corollary 3.3. *There is an algorithm for the following problem*:
Instance: *an automatic structure for a semigroup*;
Problem: *decide whether the semigroup is right cancellative.*

Proof. It suffices to check that each generator $a \in A$ is right cancellable by computing $L_a \circ L_a^{-1}$ and comparing with the diagonal relation on L. □

In an earlier draft of this paper, we asked whether one can decide algorithmically whether a given automatic structure describes a cancellative, or a left cancellative semi-

group. Cain [3] has recently shown that both of these properties are in general undecidable, even when the input is specified as an interpreted automatic structure.

Proposition 3.4. *There is an algorithm for the following problem*:
Instance: *an uninterpreted automatic structure*;
Problem: *decide whether the semigroup described has a left zero, and if so find a finite automaton recognising the language of all words in L representing left zeros.*

Proof. For a word $z \in L$ we have that z is a left zero if and only if $za = z$ in the semigroup for all generators $a \in A$, that is, if $(z, z) \in L_a$ for all generators $a \in A$. Hence, computing the intersection of all the languages L_a with the diagonal relation, and taking the projection onto one coordinate, gives the language of all words in L representing left zeros. If this language is empty, output **NO**. If not, output **YES** and the automaton computed for the language. □

Proposition 3.5. *There is an algorithm for the following problem*:
Instance: *an uninterpreted automatic structure*;
Problem: *decide whether the semigroup described has a zero, and if so find the word in L representing it.*

Proof. First apply Proposition 3.4 to check whether the semigroup has a left zero; if not, output **NO**. Otherwise, let K be the language of words in L representing left zeros.

Choose any word $w \in K$. Since a zero in a semigroup must be the unique left zero and unique right zero, the semigroup has a zero if and only if w represents a right zero. We can test this by checking whether $uw = w$ in the semigroup for all words $u \in L$, that is, whether $L_w = L \times \{w\}$. If so, output **YES**, otherwise output **NO**. □

Question 3.6. *Is there an algorithm for the following problem*?
Instance: *an uninterpreted automatic cross-section (A, L)*;
Problem: *decide if the semigroup presented is a monoid.*

4. Basic algorithms for interpreted automatic structures

In this section we consider various basic algorithmic problems which start with an interpreted automatic structure. We begin with the following simple proposition, the method of which is essentially taken from [9].

Proposition 4.1. *There is an algorithm to solve the following problem*:
Instance: *an interpreted automatic structure and a nonempty word u over the generators*;
Problem: *find a word in the language of representatives, which represents the same element as u.*

Proof. We use induction on the length of u. The base case is given by the assignment of generators. Now in general, write $u = va$ where a is a generator. By the inductive hypothesis, we can find a word w in the language of representatives representing the same element as v. Now a word x in the language of representatives represents u if and only if $(w, x) \in L_a$. Using standard operations on finite automata, it is straightforward to find such an x. □

Corollary 4.2 *(Uniform Word Problem). There is an algorithm to solve the following problem*:

Instance: *an interpreted automatic structure and two words u and v over the generators*;

Problem: *decide if u and v represent the same element.*

Proof. By Proposition 4.1, we can compute elements w_u and w_v in L representing u and v, respectively, and then check whether $(w_u, w_v) \in L_=$. □

With an interpretation available, it becomes an easy task to find the left identities (if any) of a given automatic semigroup, and hence also to decide if the semigroup is a monoid.

Proposition 4.3. *There is an algorithm for the following problem*:

Instance: *an interpreted automatic structure* (Γ, ι) *for a semigroup*;

Problem: *decide whether the semigroup described has a left identity, and if so find a finite automaton recognising the language of all words in L representing left identities.*

Proof. Note that for $w \in L$ and $a \in A$, we have $wa = a$ in the semigroup if and only if $(w, \iota(a)) \in L_a$. It follows that we can easily compute the set of words in L which stabilise the generators on the left, that is, which represent left identities in the monoid. If this language is empty, then output **NO**; otherwise, output **YES** and the automaton computed for the language. □

Proposition 4.4. *There is an algorithm for the following problem*:

Instance: *an interpreted automatic structure for a semigroup*;

Problem: *Decide whether the semigroup described is a monoid, and if so, find a word in L representing the identity.*

Proof. By Proposition 4.3, we can check that the semigroup has at least one left identity. If it does not, output **NO**; if it does, then let K be the (nonempty) language of words in L representing left identities.

Choose any word $w \in K$. Since the identity in a monoid is the unique left identity and the unique right identity, the semigroup is a monoid if and only if w represents a right identity. We can check if w represents a right identity by verifying that for each generator $a \in A$ we have $aw = a$ in the semigroup (or alternatively that $L_w = L_=$). If so, output **YES** and the word w; otherwise output **NO**. □

Recall that a *right inverse* [*left inverse*] of a monoid element s is a monoid element t such that $st = 1$ [respectively $ts = 1$]. An element with a left and a right inverse (which

necessary coincide) is called a *unit*; the set of all units in a monoid forms a subgroup of the monoid.

Proposition 4.5. *There is an algorithm to solve the following problem*:
Instance: *an interpreted automatic structure describing a monoid, and a nonempty word w in the generators*;
Problem: *decide whether w has a left inverse in the monoid, and obtain an automaton recognising the language of representatives in L of its left inverses.*

Proof. By Proposition 4.4, we can compute a word $e \in L$ representing the identity. Now a left inverse of w is an element represented by a word $w' \in L$ such that $(w', e) \in L_w$. It is straightforward to compute the language of such and check whether it is empty. □

In contrast, we shall see in Section 5 below that it is not in general possible to decide whether a word w represents an element with a right inverse.

Proposition 4.6. *There is an algorithm for the following problem*:
Instance: *an interpreted automatic structure describing a monoid and a nonempty word w in the generators*;
Problem: *decide whether w represents a unit in the monoid.*

Proof. By Proposition 2.9, we may assume that the automatic structure has uniqueness. First use the method above to check that w has a left inverse, and obtain the language of its left inverses. If w is to be a unit then it can have only one left inverse, so if this language has more than one element, output **NO**. Otherwise, let w' be the unique element of the language, and check whether $ww' = e$ in the semigroup. □

Let S be a semigroup, and 0 a new symbol not in S. We define a new semigroup S^0 with set of elements $S \cup \{0\}$, and multiplication given by

$$xy = \begin{cases} 0 & \text{if } x = 0 \text{ or } y = 0, \\ \text{the } S\text{-product } xy & \text{otherwise.} \end{cases}$$

The semigroup S^0 is called *S with an adjoined zero*. The following result is essentially an algorithmic restatement of [5, Proposition 3.13].

Proposition 4.7. *There is an algorithm for the following problem*:
Instance: *an interpreted automatic structure Γ describing a semigroup S*;
Problem: *compute an automatic structure for the semigroup S^0.*

5. Undecidability of the existence of a right inverse

The aim of this section is to demonstrate the existence of automatic monoids for which there is no algorithm to decide, given a word u over the generating set, whether u represents

an element with a right inverse in the monoid. Our proof is a variation of a well-known technique for encoding a Turing machine into a string-rewriting system (see, for example, [21]). Our result improves upon a recent result of Lohrey [19], who used a similar method to show that right-reachability for automatic monoids is, in general, undecidable. We use without further comment a number of standard results from the theory of string-rewriting; these can be found in [2].

Let $M = (Q, \Sigma, B, q_0, q_a, \delta)$ be a deterministic single-tape Turing machine, where Q is the finite set of states, Σ is the finite tape alphabet, $B \notin \Sigma$ is the blank symbol, $q_0 \in Q$ is the initial state, $q_a \in Q$ is the final (accepting) state and

$$\delta : \big((Q \setminus \{q_a\}) \times \Sigma\big) \to \big(Q \times \Sigma \times \{\lambda, \rho\}\big)$$

is the transition function. We assume that M halts if and when it enters the state q_a, and that the tape of M is unrestricted only to the right. Given a word $w \in \Sigma^*$ as input, the corresponding initial configuration of M is $q_0 w$, where we assume that w occupies the prefix of length $|w|$ of the tape. The word w is accepted by M if and only if the computation of M that starts from this initial configuration finally ends in the final state. As the tape is limited on the left, during a computation the head of M cannot ever move to the left of its start position. By $L(M)$ we denote the set of all words that are accepted by M.

From M we now construct a finite string-rewriting system R_M on the alphabet

$$\Gamma := Q \cup \Sigma \cup \bar{\Sigma} \cup \{d, h, \bar{h}\},$$

where $\bar{\Sigma} := \{\bar{s} \mid s \in \Sigma\}$ is a marked copy of Σ, and d, h, and $\bar{h}$ are three new symbols. The system R_M consists of the following rules:

$$\begin{array}{lllll}
(1) & qad & \to \bar{b}p & & \text{if } \delta(q,a) = (p,b,\rho),\\
(2) & qhd & \to \bar{b}ph & & \text{if } \delta(q,B) = (p,b,\rho),\\
(3) & \bar{c}qad & \to pcb & & \text{if } \delta(q,a) = (p,b,\lambda),\ c \in \Sigma,\\
(4) & \bar{c}qhd & \to pcbh & & \text{if } \delta(q,B) = (p,b,\lambda),\ c \in \Sigma,\\
(5) & q_a ad & \to q_a & & \text{for } a \in \Sigma,\\
(6) & \bar{a}q_a hd & \to q_a h & & \text{for } a \in \Sigma,\\
(7) & \bar{h}q_a hd & \to \varepsilon, & &\\
(8) & abd & \to adb & & \text{for } a, b \in \Sigma,\\
(9) & ahd & \to adh & & \text{for } a \in \Sigma.
\end{array}$$

We define a binary relation $>$ on Γ^* as follows:

$$\begin{aligned}
u > v \quad \Leftrightarrow \quad & |u|_d > |v|_d \quad \text{or} \quad |u|_d = n = |v|_d,\\
& u = u_0 d u_1 d \ldots u_{n-1} d u_n, \quad v = v_0 d v_1 d \ldots v_{n-1} d v_n, \quad \text{and}\\
& \exists j\colon\ |u_j| > |v_j| \text{ and } |u_i| = |v_i| \text{ for all } 0 \leqslant i < j.
\end{aligned}$$

It is easily seen that $>$ is the strict part of a partial ordering on Γ^* that is well founded. Further, whenever $u \to_{R_M} v$ holds, then $u > v$. Thus, R_M does not generate any infinite reduction sequences, that is, R_M is noetherian. As there are no nontrivial overlaps between the left-hand sides of the rules of R_M (recall that M is assumed to be a deterministic Turing machine), we see that R_M is also confluent. Thus, R_M is a finite convergent system, which implies that the set $\mathsf{IRR}(R_M)$ of irreducible words mod R_M is a regular set of normal forms for the monoid S_M that is given through the finite presentation $(\Gamma; R_M)$.

Claim 1. *The language* $\mathsf{IRR}(R_M)$ *forms the language of representatives for an automatic structure for* S_M.

Proof. It remains to show that, for each symbol $a \in \Gamma$, the right-multiplication relation L_a is synchronously regular. For each rule $(\ell \to r) \in R_M$, we see that ℓ ends with the symbol d. Thus, for each symbol $a \in \Gamma \setminus \{d\}$, if $u \in \mathsf{IRR}(R_M)$, then also $ua \in \mathsf{IRR}(R_M)$. Thus, for all these letters the corresponding language L_a is clearly synchronously regular.

It remains to consider the language L_d. Let $u \in \mathsf{IRR}(R_M)$. Then there are three mutually exclusive cases:

(1) ud is also irreducible modulo R_M;
(2) $u = xaz$, where $a \in \Sigma$, $z \in (\Sigma \cup \{h\})^+$, and $xad \in \mathsf{IRR}(R_M)$, which implies that $xadz$ is the irreducible descendant of $ud = xazd$ modulo R_M;
(3) $u = xyaz$ for some factors $x, y, z \in \mathsf{IRR}(R_M)$ and a letter $a \in \Sigma \cup \{h\}$ satisfying the following conditions:
   (a) yad is the left-hand side of one of the rules of type (1) to (7) of R_M,
   (b) if $a \in \Sigma$, then $z \in (\Sigma \cup \{h\})^*$, and if $a = h$, then $z = \varepsilon$.
   In this case the irreducible descendant of $ud = xyazd$ is the word xrz, where r is the right-hand side of the rule with left-hand side yad.

In the first case $(u, ud) \in L_d$, in the second case $(xaz, xadz) \in L_d$, while in the third case $(xyaz, xrz) \in L_d$. It is easily verified that these observations imply that the language $\delta(L_d)$ is regular so that L_d is synchronously regular as required. □

Thus, the monoid S_M is automatic.

Claim 2. *For each word* $w \in \Sigma^*$, $w \in L(M)$ *if and only if there exists an integer* $n > 0$ *such that*

$$\bar{h}q_0wh \cdot d^n \to_{R_M}^+ \varepsilon.$$

Proof. The reductions modulo R_M essentially just simulate the steps of the Turing machine M. Each step of the simulation digests one occurrence of the symbol d. Further occurrences of the symbol d are needed to reduce the encoding $\bar{h}\bar{x}q_ayh$ of the final configuration of M to the word $\bar{h}q_ah$, and another occurrence of the symbol d is needed for the final step reducing $\bar{h}q_ah$ to the empty word. □

Claim 3. *For each word $w \in \Sigma^*$, $\bar{h}q_0wh$ is right-invertible in S_M if and only if $w \in L(M)$.*

Proof. If $w \in L(M)$, then $\bar{h}q_0wh$ is right-invertible in S_M by Claim 2. Conversely, if $\bar{h}q_0wh$ is right-invertible in S_M, then there exists an irreducible word z such that $\bar{h}q_0wh \cdot z \to_{R_M}^{+} \varepsilon$ holds. As $\bar{h}q_0wh$ and z are both irreducible, rewrite steps can only be applied across the border of these two factors. It follows that $z = d^n$ for some positive integer n, which in turn implies by Claim 2 that $w \in L(M)$ holds. □

Thus, the halting problem for the Turing machine M reduces to the problem of finding a right inverse for an element in the automatic monoid S_M, giving the following result.

Theorem 5.1. *There exists a finitely presented automatic monoid S and an (interpreted) automatic structure for S, for which the following problem is in general undecidable*:

Instance: *a word w in the language of representatives*;

Problem: *decide if w represents an element with a right inverse in S.*

6. Completely simple and completely zero-simple semigroups

In this section, we present algorithms to decide if a given automatic structure represents a completely simple or completely zero-simple semigroup and, in the case that it does, to compute a Rees matrix decomposition for the semigroup. As well as being of interest in its own right, this shows that the use of automatic structures to describe semigroups facilitates the decision of quite complex structural properties.

Recall that a *primitive idempotent* in a semigroup S is an idempotent e with the property that for any nonzero idempotent f such that $ef = fe = f$, we have $e = f$. A semigroup is called *completely simple* if it has a primitive idempotent and no proper ideals. A semigroup with zero is called *completely zero-simple* if the zero is the only proper ideal, and it has no infinite descending chains of idempotents. For a detailed introduction to the theory of completely simple and completely zero-simple semigroups, including a number of equivalent definitions, see [15, Chapter 3].

The following construction, due to Rees [28], provides a way to describe completely zero-simple and completely simple semigroups. Let G be a group and I and Λ be sets. Let 0 be a symbol not in G, and let P be a $\Lambda \times I$ matrix with entries drawn from $G \cup \{0\}$. The *Rees matrix semigroup with zero* $M^0(G; I, \Lambda; P)$ is the semigroup with set of elements

$$(I \times G \times \Lambda) \cup \{0\}$$

and multiplication given by

$$(i, g, \lambda)(j, h, \mu) = \begin{cases} (i, gP_{\lambda j}h, \mu) & \text{if } P_{\lambda j} \in G; \\ 0 & \text{if } P_{\lambda j} = 0 \end{cases}$$

and $x0 = 0 = 0x$ for all elements x.

The matrix P is called *regular* if every row and every column contains a nonzero entry. If P contains no zero entries at all, then the set of nonzero elements of $M^0(G; I, \Lambda; P)$ forms a subsemigroup, called a *Rees matrix semigroup* (*without zero*) and denoted $M(G; I, \Lambda; P)$.

The following theorem is usually attributed to Rees, although it was essentially prefigured by Suschkewitz [30]. It provides the connection between completely simple semigroups and Rees matrix constructions.

Theorem 6.1. (Suschkewitz, 1928; Rees, 1940) *Let G be a group, I and Λ sets, and P a regular $\Lambda \times I$ matrix over $G \cup \{0\}$. Then $M^0(G; I, \Lambda; P)$ is a completely zero-simple semigroup. If P contains no zero entries, then $M(G; I, \Lambda; P)$ is a completely simple semigroup.*

Conversely, every completely simple or completely zero-simple semigroup is isomorphic to one of this form.

Completely simple and completely zero-simple semigroups and Rees matrix constructions are of fundamental importance in the theory of semigroups. Various authors have considered the relationship between Rees matrix constructions and automaticity properties. In [6], it is shown that a finitely generated completely simple semigroup is automatic if and only if its maximal subgroups are automatic; a consequence of results in [7] is that the same applies in the completely zero-simple case. However, the methods used in these papers are essentially nonconstructive, in that they presuppose knowledge of the Rees matrix representation for the semigroup.

In this section, we present two algorithms relating automatic structures to completely simple and completely zero-simple semigroups; the first takes as input an interpreted automatic structure, and decides whether the semigroup represented is completely zero-simple. Our second algorithm takes as input an automatic structure presupposed to represent a completely zero-simple semigroup, and computes a Rees matrix representation for the semigroup; the latter takes the form of an automatic structure for the (necessarily unique up to isomorphism) maximal subgroup, and a sandwich matrix of words from the language of representatives. We also show how to apply these results to completely simple semigroups without zero.

6.1. Deciding complete simplicity and complete zero-simplicity

In this section, we show that there is an algorithm which, given an automatic structure, decides whether the semigroup described is completely simple or completely zero-simple. We shall require the following lemma.

Lemma 6.2. *There is an algorithm for the following problem*:

Instance: *an interpreted automatic structure representing a semigroup S, and two nonempty words w and e in the generators such that e represents an idempotent in S*;

Problem: *decide which of the following comprehensive and mutually exclusive conditions applies*:

(A) *w has an infinite number of left inverses with respect to e*;
(B) *w has a finite number of left inverses with respect to e, one of which is also a right inverse*;
(C) *w has a finite number of left inverses with respect to e, none of which is a right inverse.*

Proof. By Proposition 2.9, we may assume that the automatic structure has uniqueness. By Proposition 4.1, we may assume that w and e lie in the language of representatives.

We begin by computing the language K of left inverses for w with respect to e. This is the set of all words w' such that $(w', e) \in L_w$. If K is infinite, output (A). Otherwise, check each $w' \in K$ in turn to see if $ww' = e$ in the semigroup. If one does, output (B); otherwise, output (C). □

Theorem 6.3. *There is an algorithm for the following problem*:
Instance: *an interpreted automatic structure for a semigroup S*;
Problem: *decide if S is completely zero-simple.*

Proof. By Proposition 2.9, we may assume that we are given an interpreted automatic structure with uniqueness, and that the language of representatives contains the generators. By Proposition 3.5, we can check that the semigroup has a zero. If not, output **NO**. If so, let z be the (unique) word in L representing the zero. Check which, if any, of the generators represent zero.

Now for every generator a which does not represent zero, calculate the set of words in L representing elements which stabilise a on the left (that is, the projection onto the first coordinate of $L_a \cap (A^+ \times \{a\})$). Call it SL_a. In a completely zero-simple semigroup, the elements which stabilise a nonzero element s on the left are exactly the idempotents in the $\mathcal{R}$-class of s; it follows that (i) there must be one such, (ii) there can only be finitely many such (by the Main Theorem of [1]) and (iii) they are all idempotents. Check these three conditions, and if any fail, output **NO**.

Let E be the union of the SL_a's; thus, E is a (finite) set of nonzero idempotents. (If S is indeed completely zero-simple then it follows easily from Theorem 6.1 that there must be a generator in every $\mathcal{R}$-class, and hence that E contains all the nonzero idempotents of S; however, we cannot directly verify this.)

Since E is finite, we can check which words in E stabilise each generator $a \in A$ on the right; for each $a \in A$, let SR_a be the set of such. Check that every element of E lies in SR_b for some generator b. For every pair of nonzero generators a and b, check:

(i) that SL_a intersects SR_b in at most one element, and in *exactly* one element if and only if ba represents a nonzero element in the semigroup;
(ii) that SL_a and SL_b are either equal or disjoint; and
(iii) that SR_a and SR_b are either equal or disjoint.

It follows from the Rees theorem that all of these conditions must hold in a completely zero-simple semigroup, so if any fails, output **NO**.

Our next objective is to verify that idempotents in the same SL-class [SR-class] are $\mathcal{R}$-related [$\mathcal{L}$-related] in the semigroup. For this, it will suffice to verify that for every

$a \in A$ and every pair of elements $e, f \in SL_a$ [$e, f \in SR_a$] we have $ef \mathcal{H} f$ [$ef \mathcal{H} e$] in the semigroup. To verify this, we invoke Lemma 6.2. If the semigroup is completely simple then it is easily verified that ef must have finitely many left inverses with respect to f [respectively e], one of which is also right inverse. Hence, we can use Lemma 6.2 to check that $ef \mathcal{H} f$ [respectively $ef \mathcal{H} e$], outputting **NO** if it transpires that ef has no left inverses, infinitely many left inverses or finitely many left inverses none of which is a right inverse, with respect to f [respectively e].

To proceed further, we employ the following lemma.

Lemma 6.4. *There is an algorithm to perform the following task*:

Instance: *a word $w \in L$ which does not represent zero*;

Problem: either *find an idempotent in E which is $\mathcal{L}$-related to w and an idempotent in E which is $\mathcal{R}$-related to w,* or *discover that the semigroup is not completely zero-simple.*

Proof. First, check which idempotents in E stabilise w on the right. For each such idempotent e, check whether there exists $q \in L$ such that $qw = e$ in the semigroup, that is, such that $(q, e) \in L_w$. Since a completely zero-simple semigroup has an idempotent in every $\mathcal{L}$-class, if there are none such, output that the semigroup is not completely zero-simple. Otherwise, assume we have found a nonzero idempotent e which is $\mathcal{L}$-related to w, and such that $qw = e$.

Now if the semigroup is completely zero-simple then, by a standard result, wq is also a nonzero idempotent, and, moreover, this idempotent must have a representative in E. We can locate this representative by solving the uniform word problem; call it f. Now if the semigroup is completely zero-simple then $wq = f$ is $\mathcal{R}$-related to w, so that $fw = w$. Check this condition; if it fails, output **NO**. Otherwise we have $fw = w$ and $wq = f$ so that $w \mathcal{R} f$, as required. □

Using Lemma 6.4, we check that every generator a is $\mathcal{R}$-related to an idempotent (and hence to every idempotent) in SL_a and $\mathcal{L}$-related to an idempotent in SR_b. In a completely zero-simple semigroup this must be the case, so if not, output **NO**.

Now, we check that for every pair of generators b and a with nonzero product, the product ba is $\mathcal{R}$-related to an idempotent in SL_b and $\mathcal{L}$-related to an idempotent in SR_a. Again, we can check this by Lemma 6.4, and it must be satisfied in a completely zero-simple semigroup, so if it fails, output **NO**.

Finally, we check that for every pair of generators a, b, there exists a word $c_1 \dots c_n \in L$ which does not represent zero, and which has the property that $SL_{c_1} = SL_a$ and $SR_{c_n} = SR_b$. Once again, if this fails, output **NO**.

We claim, at this point, that the semigroup is completely zero-simple.

First, we claim that for every word $a_1 \dots a_n \in A^+$, the element represented is $\mathcal{R}$-related to a_1 and $\mathcal{L}$-related to a_n in the semigroup. If $n = 1$ there is nothing to prove. If $n = 2$ then we have checked that a_1a_2 is $\mathcal{L}$-related to an idempotent in SR_{a_2}, which in turn is $\mathcal{L}$-related to a_2 itself. Similarly, a_1a_2 is $\mathcal{R}$-related to an idempotent in SL_{a_1} which in turn is $\mathcal{R}$-related to a_1 itself.

Now assume that $a_1 \dots a_n$ is a counterexample of minimal length (necessarily 3 or more), say not $\mathcal{L}$-related to a_n. Now certainly we have $a_1a_2 \mathcal{L} a_2$. Now since $\mathcal{L}$ is right

compatible, we see that $a_1 \dots a_n \mathcal{L}\, a_2 \dots a_n$. But by the minimality assumption, $a_2 \dots a_n$ is $\mathcal{L}$-related to a_n. Since $\mathcal{L}$ is transitive, this gives the required contradiction. A symmetric argument applies in the case that $a_1 \dots a_n$ is not $\mathcal{R}$-related to a_1.

It follows now that the semigroup has a single nonzero $\mathcal{D}$-class. Indeed, suppose $a_1 \dots a_m, b_1 \dots b_n \in A^+$ represent nonzero elements. Then $a_1 \dots a_m \mathcal{R}\, a_1$ and $b_1 \dots b_n \mathcal{L}\, b_n$. Now there is a word $c_1 \dots c_q \in L$ which does not represent zero such that $SL_{c_1} = SL_{a_1}$ and $SR_{c_q} = SR_{b_n}$. It follows that

$$a_1 \dots a_m \mathcal{R}\, a_1 \mathcal{R}\, c_1 \mathcal{R}\, c_1 \dots c_q \mathcal{L}\, c_q \mathcal{L}\, b_n \mathcal{L}\, b_1 \dots b_n,$$

so that $a_1 \dots a_m \mathcal{D} b_1 \dots b_n$ as required. Thus, the semigroup has a single nonzero $\mathcal{D}$-class.

Moreover, we have seen that every element lies in the $\mathcal{R}$-class [$\mathcal{L}$-class] of one of the finitely many generators. So the semigroup has only finitely many $\mathcal{R}$-classes and $\mathcal{L}$-classes, and hence also only finitely many $\mathcal{H}$-classes. Since an $\mathcal{H}$-class can contain at most one idempotent, we conclude that the semigroup has only finitely many idempotents, and that the semigroup is completely zero-simple as required. □

A corollary is a corresponding result for completely simple semigroups.

Corollary 6.5. *There is an algorithm for the following problem*:
Instance: *an interpreted automatic structure describing a semigroup S*;
Problem: *decide whether S is completely simple.*

Proof. Clearly, S is completely simple if and only if S^0 is completely zero-simple. Use Proposition 4.7 to obtain an automatic structure for S^0, and then apply Theorem 6.3. □

6.2. Computing the Rees matrix structure

In this section, we present an algorithm which, given an automatic structure for a completely zero-simple semigroup, computes its Rees matrix structure.

Theorem 6.6. *There is an algorithm for the following problem*:
Instance: *an interpreted automatic structure Γ which describes a completely zero-simple semigroup S*;
Problem: *construct* (i) *an interpreted automatic structure Δ for the maximal subgroup G of S and* (ii) *a finite matrix Q with entries drawn from $L(\Delta) \cup \{0\}$, such that S is isomorphic to the Rees matrix semigroup $M^0(G; I, \Lambda; Q)$ (where Q is interpreted as a matrix over $G \cup \{0\}$ in the obvious way).*

Proof. By Proposition 2.9, we may assume that the automatic structure Γ has uniqueness. We begin by using the procedure from the proof of Theorem 6.3 to find the (necessarily finite) set E of representatives in $L(\Gamma)$ of idempotents. Again, we sort these into $\mathcal{L}$-classes and $\mathcal{R}$-classes according to how they stabilise each other. Let I and Λ be the sets of nonzero $\mathcal{R}$- and $\mathcal{L}$-classes, respectively. For $i \in I$ and $\lambda \in \Lambda$ let $H_{i\lambda}$ denote the $\mathcal{H}$-class

$i \cap \lambda$ and let $e_{i\lambda}$ be the word representing the (necessarily unique) idempotent in $H_{i\lambda}$ if it exists.

Choose distinguished elements $i_0 \in I$ and $\lambda_0 \in \Lambda$ such that $H_{i_0\lambda_0}$ contains an idempotent, that is, such that $e_{i_0\lambda_0}$ exists.

Next, we wish to choose for each $i \in I$ a word $r_i \in L(\Gamma)$ representing an element of $H_{i\lambda_0}$. If $H_{i\lambda_0}$ contains an idempotent then we simply set $r_i = e_{i\lambda_0}$. Otherwise, we are involved in slightly more work. The words we seek are exactly those of the form $awb \in L(\Gamma)$ where a and b are generators in the $\mathcal{R}$-class i and the $\mathcal{L}$-class λ_0, respectively, and awb does not represent zero. Since the $\mathcal{H}$-class cannot be empty, there must be such a word; hence, we can find one by enumerating the set of such words until we find one which does not represent zero. Similarly, we choose for each $\lambda \in \Lambda$ a word $q_\lambda \in L(\Gamma)$ representing an element of $H_{i_0\lambda}$. Notice that by construction we have

$$r_{i_0} = q_{\lambda_0} = q_{\lambda_0} r_{i_0} = e_{\lambda_0 i_0}.$$

We now construct an $\Lambda \times I$ matrix P with entries drawn from $L(\Gamma) \cup \{0\}$, where 0 is a new symbol. For each $i \in I$ and $\lambda \in \Lambda$, let $P_{\lambda i}$ be the word in L representing the same element as the product $q_\lambda r_i$. Notice that this element lies in $H_{i_0\lambda_0}$. It follows from the proof of [15, Theorem 3.2.3] that the semigroup S is isomorphic to the Rees matrix semigroup $M(H_{i_0\lambda_0}; I, \Lambda; P)$, where P is intepreted as a matrix over $H_{i_0\lambda_0} \cup \{0\}$ in the obvious way.

It remains to construct an automatic structure for the maximal subgroup $H_{i_0\lambda_0}$. It follows from standard facts about completely zero-simple semigroups that for each generator $a \in A(\Gamma)$, there exists a unique $i_a \in I$, a unique $\lambda_a \in \Lambda$ and a unique $b_a \in H_{\lambda_0 i_0}$ such that $a = r_{i_a} b_a s_{\lambda_a}$. For each $a \in A(\Gamma)$, we compute (for example, by enumeration and testing, although more efficient means are available) a representative $w_a \in L(\Gamma)$ for b_a.

Define a new alphabet

$$A(\Delta) = \{c_a \mid a \in A\} \cup \{d_{\lambda i} \mid \lambda \in \Lambda,\ i \in I,\ P_{\lambda i} \neq 0\}.$$

We view $A(\Delta)$ as an alphabet of generators for the maximal subgroup $H_{i_0\lambda_0}$, where each c_a represents the element $b_a \in H_{i_0\lambda_0}$, and each $d_{\lambda i}$ represents the element represented by the sandwich matrix entry $P_{\lambda i}$.

Let R be the set of words in L which represent elements of $H_{i_0\lambda_0}$. Define a function $\phi : R \to B^+$ by

$$\phi(a_1 a_2 \ldots a_n) = c_{a_1} d_{\lambda_{a_1} i_{a_2}} c_{a_2} \ldots d_{\lambda_{a_{n-1}} i_{a_n}} c_{a_n}$$

where each i_k and λ_k are such that a_k represents an element of $H_{\lambda_k i_k}$. Let $L(\Delta) = \phi(L \cap R)$. It follows from [17, Proof of Theorem 4.6] that $L(\Delta)$ is a regular language and is straightforward to compute. Now recalling that $P_{\lambda_0 i_0} = e_{\lambda_0 i_0}$, one can show that

$$L_=(\Delta) = \big\{(u\phi, v\phi) \mid (u, v) \in L_= \cap (R \times R)\big\}$$

while for each $c_a \in B$ we have

$$L_{c_a}(\Delta) = \big\{(u\phi, v\phi) \mid (u, v) \in L_{w_a} \cap (R \times R)\big\}$$

and for each $d_{\lambda i} \in B$

$$L_{d_{\lambda i}}(\Delta) = \big\{(u\phi, v\phi) \mid (u, v) \in L_{P_{\lambda i}} \cap (R \times R)\big\}.$$

It can now be shown (for example, using arguments similar to those in [17, Proof of Theorem 4.6]) that each of these relations is synchronously rational and straightforward to compute. This completes the construction of an automatic structure for the maximal subgroup $H_{i_0\lambda_0}$.

Finally, we apply the function ϕ to each nonzero entry in the sandwich matrix P, to obtain a new $\Lambda \times I$ matrix with entries drawn from K. □

As an immediate corollary, we obtain a corresponding result for the case of completely simple semigroups.

Corollary 6.7. *There is an algorithm for the following problem*:

Instance: *an interpreted automatic structure Γ which represents a completely simple semigroup S*;

Problem: *construct* (i) *an automatic structure Δ for the maximal subgroup G of S and* (ii) *a finite matrix Q with entries drawn from $L(\Delta)$, such that S is isomorphic to the Rees matrix semigroup $M(G; I, \Lambda; Q)$* (*where Q is interpreted as a matrix over G in the obvious way*).

Proof. By Proposition 4.7, we can adjoin a zero to S to obtain a completely zero-simple semigroup S^0. By Theorem 6.6 we can construct a Rees matrix representation $M^0(G; I, \Lambda; P)$ for S^0 together with an automatic structure for G. It is easily verified that P has no zero entries and $M(G; I, \Lambda; P)$ is a Rees matrix representation for S. □

Acknowledgment

The first author thanks Kirsty for all her support and encouragement.

References

[1] H. Ayik, N. Ruškuc, Generators and relations of Rees matrix semigroups, Proc. Edinburgh Math. Soc. 42 (1999) 481–495.
[2] R.V. Book, F. Otto, String-Rewriting Systems, Springer-Verlag, New York, 1993.
[3] A.J. Cain, Cancellativity is undecidable for automatic semigroups, Q. J. Math., in press.
[4] C.M. Campbell, E.F. Robertson, N. Ruškuc, R.M. Thomas, Direct products of automatic semigroups, J. Aust. Math. Soc. 69 (2000) 19–24.
[5] C.M. Campbell, E.F. Robertson, N. Ruškuc, R.M. Thomas, Automatic semigroups, Theoret. Comput. Sci. 250 (2001) 365–391.
[6] C.M. Campbell, E.F. Robertson, N. Ruškuc, R.M. Thomas, Automatic completely-simple semigroups, Acta Math. Hungar. 95 (2002) 201–215.
[7] L. Descalço, N. Ruškuc, On automatic Rees matrix semigroups, Comm. Algebra 30 (2002) 1207–1226.

[8] A.J. Duncan, E.F. Robertson, N. Ruškuc, Automatic monoids and change of generators, Math. Proc. Cambridge Philos. Soc. 127 (1999) 403–409.
[9] D.B.A. Epstein, et al., Word Processing in Groups, Jones and Bartlett, Boston, 1992.
[10] The GAP-Group, GAP—Groups, Algorithms, and Programming, Version 4.4, 2005, http://www.gap-system.org.
[11] M. Hoffmann, D. Kuske, F. Otto, R.M. Thomas, Some relatives of automatic and hyperbolic groups, in: G.M.S. Gomes, J.E. Pin, P.V. Silva (Eds.), Semigroups, Algorithms, Automata and Languages, World Scientific, Singapore, 2003, pp. 379–406.
[12] M. Hoffmann, N. Ruškuc, R.M. Thomas, Automatic semigroups with subsemigroups of finite Rees index, Internat. J. Algebra Comput. 12 (2002) 463–476.
[13] M. Hoffmann, R.M. Thomas, Automaticity and commutative semigroups, Glasg. Math. J. 44 (2002) 167–176.
[14] M. Hoffmann, R.M. Thomas, Notions of automaticity in semigroups, Semigroup Forum 66 (2003) 337–361.
[15] J.M. Howie, Fundamentals of Semigroup Theory, Clarendon Press, New York, 1995.
[16] J.F.P. Hudson, Regular rewrite systems and automatic structures, in: J. Almeida, G.M.S. Gomes, P.V. Silva (Eds.), Semigroups, Automata and Languages, World Scientific, Singapore, 1996, pp. 145–152.
[17] M.E. Kambites, Automatic Rees matrix semigroups over categories, arXiv: math.RA/0509313, 2005.
[18] M.V. Lawson, Finite Automata, Chapman and Hall, Boca Raton, FL, 2003.
[19] M. Lohrey, Decidability and complexity in automatic monoids, Internat. J. Found. Comput. Sci. 16 (4) (2005) 707–722.
[20] A. Markov, The impossibility of certain algorithms in the theory of associative systems, Dokl. Akad. Nauk SSSR (N.S.) 77 (1951) 19–20.
[21] F. Otto, When is an extension of a specification consistent? Decidable and undecidable cases, J. Symbolic Comput. 12 (3) (1991) 255–273.
[22] F. Otto, On s-regular prefix-rewriting systems and automatic structures, in: T. Asano, H. Imai, D.T. Lee, S. Nakano, T. Tokuyama (Eds.), COCOON '99, Proc., in: Lecture Notes in Comput. Sci., vol. 1627, Springer-Verlag, Berlin, 1999, pp. 422–431.
[23] F. Otto, On Dehn functions of finitely presented bi-automatic monoids, J. Autom. Lang. Comb. 5 (4) (2000) 405–419.
[24] F. Otto, On the Dehn functions of automatic monoids, in: T. Imaoka (Ed.), Third Symposium on Algebra, Languages and Computation, Proc., Osaka, 1999, Shimane Univ., Matsue, 2000, pp. 55–63.
[25] F. Otto, N. Ruškuc, Confluent monadic string-rewriting systems and automatic structures, J. Autom. Lang. Comb. 6 (3) (2001) 375–388.
[26] F. Otto, A. Sattler-Klein, K. Madlener, Automatic monoids versus monoids with finite convergent presentations, in: T. Nipkow (Ed.), RTA '98, Proc., in: Lecture Notes in Comput. Sci., vol. 1379, Springer-Verlag, Berlin, 1998, pp. 32–46.
[27] M. Petrich, Introduction to Semigroups, Merrill Research and Lecture Series, Merrill, Columbus, OH, 1973.
[28] D. Rees, On semi-groups, Proc. Cambridge Philos. Soc. 36 (1940) 387–400.
[29] P.V. Silva, B. Steinberg, A geometric characterization of automatic monoids, Q. J. Math. 55 (3) (2004) 333–356.
[30] A.K. Suschkewitz, Über die endlichen Gruppen ohne das Gesetz der eindeutigen Umkehrbarkeit, Math. Ann. 99 (1928) 30–50.