THE EQUATIONAL THEORY OF ω-TERMS FOR FINITE $\mathcal{R}$-TRIVIAL SEMIGROUPS*

JORGE ALMEIDA†

Centro de Matemática da Universidade do Porto and
Departamento de Matemática Pura
Faculdade de Ciências da Universidade do Porto
Rua do Campo Alegre, 687
4169-007 Porto, Portugal
E-mail: jalmeida@fc.up.pt

MARC ZEITOUN

LIAFA
Université Paris 7 et CNRS, Case 7014
2, place Jussieu
75251 Paris Cedex 05, France
E-mail: mz@liafa.jussieu.fr

A new topological representation for free profinite $\mathcal{R}$-trivial semigroups in terms of spaces of vertex-labeled complete binary trees is obtained. Such a tree may be naturally folded into a finite automaton if and only if the element it represents is an ω-term. The variety of ω-semigroups generated by all finite $\mathcal{R}$-trivial semigroups, with the usual interpretation of the ω-power, is then studied. A simple infinite basis of identities is exhibited and a linear-time solution of the word problem for relatively free ω-semigroups is presented. This work is also compared with recent work of Bloom and Choffrut on transfinite words.

*This work was started during the visit of both authors to the *Centro Internacional de Matemática*, in Coimbra, Portugal. Financial support of *Fundação Calouste Gulbenkian* (FCG), *Fundação para a Ciência e a Tecnologia* (FCT), *Faculdade de Ciências da Universidade de Lisboa* (FCUL), and *Reitoria da Universidade do Porto* is gratefully acknowledged. The work was also supported by the INTAS grant #99-1224.
†This paper has been written during this author's visit to the LIAFA with the support of the *Université Paris 7*. Work also supported by FCT through the *Centro de Matemática da Universidade do Porto*, and by the project POCTI/32817/MAT/2000, which is partially funded by the European Community Fund FEDER.

1. Introduction

Finite $\mathcal{R}$-trivial semigroups form a pseudovariety R which appears naturally as it is generated by the following classes of transformations of a finite chain: full decreasing transformations [16] or partial decreasing and order-preserving transformations [12]. The corresponding variety of languages also appears naturally: it is the smallest variety of languages containing, over a finite alphabet A, the languages B^+ with $B \subseteq A$, the letters $a \in A$, and which is closed under disjoint union and deterministic product [18].

Each regular $\mathcal{D}$-class of a finite $\mathcal{R}$-trivial semigroup forms a (left-zero) band, which places $\mathcal{R}$-trivial semigroups close enough to the well-known variety of (left regular) bands $[xyx = xy,\ x^2 = x]$. Syntactical techniques that work for R can often be extended to the pseudovariety DA, of all finite semigroups whose regular $\mathcal{D}$-classes are rectangular bands, which plays an important role not only in finite semigroup theory but also in temporal logic [22, 21].

The underlying question which motivated the work summarized in this paper is whether R is "completely tame" for the *canonical* signature $\kappa = \{\_ \cdot \_,\ \_^\omega\}$. This signature is the most commonly used in finite semigroup theory. The notation comes from [3] with a minor change resulting from the fact that we are only interested here in aperiodic semigroups. For R, the question of complete tameness roughly means the following: whether every finite system of equations with clopen constraints which has a solution in the free profinite semigroup *modulo* R also admits such a solution using only terms of the signature κ. From a simple yet rather useful solution of the word problem for ω-terms over the related pseudovariety J, of all finite $\mathcal{J}$-trivial semigroups, that is the identity problem in the signature κ, it is not hard to show that J is completely tame for the canonical signature [1].

Here we consider the related question of solving the word problem for ω-terms over the pseudovariety R and more generally studying the equational theory of R in the signature κ. With the tameness question in mind, we aim at a good understanding of this equational theory as well as efficient algorithms that may be used to test examples and allow us to deepen our intuition.

One can raise a question of a much more general nature, thus abstracting the word problem to a general pseudovariety and a suitable signature: under what general conditions on a pseudovariety V and a signature σ can one guarantee that the identity problem for V in the signature σ is decidable? This is of course a bit vague so we propose a restricted form of this question.

See [1] for undefined terms.

Question. Is there any recursively enumerable pseudovariety V for which there exists a recursively enumerable signature σ (consisting of computable implicit operations) such that V^σ has an unsolvable identity problem?

The results presented here are the following. We start with a new representation of pseudowords over R, namely as certain binary trees. Such trees are regular, that is they may be folded into finite automata by the identification of isomorphic subtrees, if and only if the pseudowords they represent are ω-terms. The minimal such representation of an ω-term is constructible in $O(mn)$-time, where m is the number of letters and n is the length of the term. This gives rise to an algorithm for solving the word problem for ω-terms over R which works in $O(mn)$-time where m is as above and n is now the length of the longest of two ω-terms whose equality over R is to be tested.

We also describe a basis of identities for the variety of *ω-semigroups* generated by R, where ω-semigroups means semigroups with an extra unary ω-power operation. Since no finite basis can be extracted from our basis, this variety is not finitely based.

This paper is meant as an extended abstract of the forthcoming paper with full details. Rather than presenting all technicalities which are required for the detailed account of the results, we concentrate on making clear the underlying ideas at the expense of not being completely precise. In particular, all proofs will be omitted. The reader interested in more details is referred to the full paper [6].

2. R-trees and R-automata

Elements of free profinite semigroups will be generally called *pseudowords*. We may also speak of *pseudowords over* V for a pseudovariety V to indicate the natural projections of pseudowords in free pro-V semigroups. For an introduction to relatively free profinite semigroups, see [1]. Recall in particular that formal equalities between pseudowords are known as *pseudoidentities* and have full descriptive power for defining pseudovarieties of semigroups. We write $\mathcal{C} \models u = v$ to indicate that the class $\mathcal{C}$ of finite semigroups satisfies the pseudoidentity $u = v$.

The basic ingredient in all our results is the following observation taken from [2].

Lemma 2.1. *Let u, v be pseudowords and suppose $u = u_1 a u_2$, $v = v_1 b v_2$*

where a, b are letters such that $a \notin c(u_1)$, $b \notin c(v_1)$, and $c(u_1) = c(v_1)$. If $\mathsf{R} \models u = v$ then $a = b$ and $\mathsf{R} \models u_i = v_i$ $(i = 1, 2)$.

The same holds for the pseudovariety S of all finite semigroups using Proposition 3.5 of [4] combined with simple arguments of language theory. This leads to the *left basic factorization* of a pseudoword, which is well defined over both S and R:

$$u = u_1 a u_2 \text{ with } c(u) = c(u_1 a),\ a \in c(u) \setminus c(u_1). \tag{1}$$

For instance, the left basic factorization of $(ab^\omega a)^\omega$ is $a \cdot b \cdot b^{\omega-1} a (ab^\omega a)^{\omega-1}$. The idea explored in [5] is to iterate this factorization on both factors u_1 and u_2, keeping in mind the observation that every infinite product of pseudowords over R converges. In [5], this is carried out considering at the same level factorizations going rightwards, leading to (in general) infinite factorization trees of finite height.

For instance, for the ω-term $(ab^\omega a)^\omega$ we have the representation in Figure 1. An alternative representation in terms of labeled ordinals was also

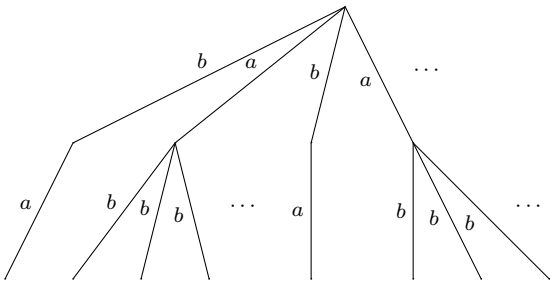


Figure 1. The Almeida-Weil tree representation of $(ab^\omega a)^\omega$

constructed in [5] which, for the same example, can be visualized as follows:

$$a \underbrace{bbbb\cdots}_{\omega} \underbrace{aa \underbrace{bbbb\cdots}_{\omega} aa \underbrace{bbbb\cdots}_{\omega} \cdots}_{\omega}$$

We present here yet another representation, which focuses on the binary flavor of the left basic factorization: each new factor u_i lies either to the left or to the right of the letter marker a which splits the previous factor u according to (1). For the same example, $(ab^\omega a)^\omega$, we obtain the infinite binary tree indicated in Figure 2, where ε stands for the empty word. In

view of the left basic factorization of $(ab^\omega a)^\omega = a \cdot b \cdot b^{\omega-1}a(ab^\omega a)^{\omega-1}$, the root is labeled by b, and the process is iterated on the left (with the pseudoword a) and on the right (with $b^{\omega-1}a(ab^\omega a)^{\omega-1}$). In this infinite tree

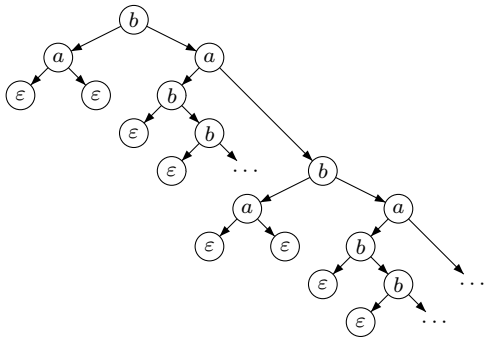


Figure 2. The R-tree of the pseudoword $(ab^\omega a)^\omega$

one recognizes immediately that certain subtrees are repeated in the sense that isomorphic copies are found several times, where isomorphism stands for topological isomorphism respecting labels. Rather than repeating a subtree T, we may as well point to a node r at which T already appeared with r as a root. This leads to a *folding* of the tree, in our example to the finite graph indicated in Figure 3 where we use the edge labels 0 for left

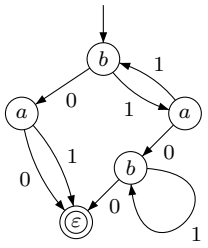


Figure 3. The minimal R-automaton of the pseudoword $(ab^\omega a)^\omega$

and 1 for right, the incoming arrow at the top represents the node coming from the root and the double circle represents the node coming from the leaves. Another example is given in Section 5.

Examining the procedure leading to the above representations of pseudowords over R, one obtains first a characterization of all vertex-labeled

complete binary trees which appear in this way, which we call R-*trees*. More generally, an R-*automaton* over an alphabet A is a deterministic $\{0,1\}$-automaton which is complete, except for the terminal states which have no arrows leaving from them, and which has the states labeled with elements from $A \cup \{\varepsilon\}$ so that the following properties are satisfied:

(1) a state is labeled ε if and only if it is terminal;
(2) for each state v, the labels of states which are accessible from v by a path starting with the edge labeled 0 miss exactly one letter from the labels of all the states accessible from v, namely the label of v.

In particular, R-trees are the acyclic R-automata. An *isomorphism of* R-*automata* is an isomorphism of the underlying automata, in the usual sense of automata theory [14], which respects the labeling of vertices. We will not distinguish between isomorphic R-automata.

The R-tree constructed by iterated left basic factorization from a pseudoword w is denoted $\mathcal{T}(w)$ and is called the R-*tree of* w. From Lemma 2.1 one may easily deduce the characterization of equality of pseudowords over R in terms of equality of their associated R-trees. Furthermore, there is a natural topology on R-trees which is relevant for our characterization: define the distance between two distinct R-trees to be 2^{-r} where r is the smallest integer such that, chopping off all nodes and edges at distance from the root greater than r, we obtain distinct vertex-labeled trees.

Theorem 2.2. *Over a finite alphabet, the correspondence associating to each pseudoword* w *its* R-*tree* $\mathcal{T}(w)$ *induces a homeomorphism between the free pro*-R *semigroup and the space of all* R-*trees.*

In particular, to every R-tree T corresponds at least one pseudoword w such that $\mathcal{T}(w) = T$. By Theorem 2.2, the value of such a pseudoword w over R depends only on T. We denote it by $w(T)$.

We may translate an R-automaton $\mathcal{A}$ over an alphabet A into a usual automaton $\mathcal{A}'$ by taking $B = \{0,1\} \times A$ as the alphabet labeling the edges and by transferring the labels of vertices to the labels of edges as second components. This transformation is clearly reversible for B-automata which have properties that are easily identified, such as: no edges leave from terminal states, and all edges leaving from other vertices have labels with the same second component.

The folding procedure may be described as taking the quotient automaton under the congruence $\sim_w$, according to which two nodes are equivalent if the subtrees rooted at these nodes are identical. The resulting automaton

will be called the *minimal* R*-automaton* of the pseudoword and denoted $\mathcal{A}(w)$. The language over the alphabet B recognized by the automaton $\mathcal{A}(w)'$ is denoted $L(w)$, and it gives a trace history of how a specific empty factor is eventually obtained by iterated left basic factorization. Note that the R-tree $\mathcal{T}(w)$ may be reconstructed from the R-automaton $\mathcal{A}(w)$ by unfolding it.

Corollary 2.3. *The following conditions are equivalent for a pair u, v of pseudowords:*

(1) $\mathsf{R} \models u = v$*;*
(2) $\mathcal{T}(u) = \mathcal{T}(v)$*;*
(3) $\mathcal{A}(u) = \mathcal{A}(v)$*;*
(4) $L(u) = L(v)$*.*

In case the congruence $\sim_w$ has finite index, the folding procedure is nothing else than the minimization of the translated automaton followed by the reverse translation, and this is consistent with naming $\mathcal{A}(w)$ the minimal R-automaton of w. The natural question which this observation raises is under what conditions on a pseudoword its minimal R-automaton is finite. For a complete answer to this question, which will be given in the next section, we introduce some pseudowords associated with a given pseudoword.

The subtree T_p of all descendants of a node p of the R-tree $T = \mathcal{T}(w)$ of a pseudoword w is itself an R-tree and therefore it uniquely determines a pseudoword $v = w(T_p)$ over R which we call an R*-factor* of w and may also call the *value of the subtree* T_p and the *value of the node* p. In view of Theorem 2.2, we will usually identify the pseudoword v with the node p and thus use the same notation for pseudowords and nodes in R-trees. Taking into account how $\mathcal{T}(w)$ is constructed by iterated left basic factorization, in case v is a direct right descendant of another node, such pseudowords are called *relative tails* of w.

3. Periodicity and the word problem for ω-terms

By an ω*-term* we mean a well-formed expression in a set of letters using an associative operation of multiplication and a unary operation of ω-power. The following result is a sort of periodicity theorem which answers the question raised at the end of the previous section.

Theorem 3.1. *The following conditions are equivalent for a pseudoword w over* R*:*

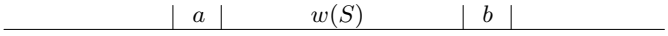
(1) w is represented by some ω-term;
(2) the set of R*-factors of w is finite;*
(3) the set of relative tails of w is finite;
(4) the folded R*-automaton of w is finite;*
(5) the language $L(w)$ is rational.

Thus, the word problem for ω-terms over R will be solved if we find a way to compute the folded R-automaton $\mathcal{A}(w)$ for each ω-term w since $\mathcal{A}(w)$ is finite and completely determines w. This may seem however a rather strange way of solving the word problem. A seemingly more natural approach would at first sight appear to be devising a finite confluent set of reduction rules which, applied repeatedly in any order until no further reduction is possible, leads to a canonical form of a given ω-term such that two ω-terms are equal over R if and only if they have the same canonical form. We have found no such system and we conjecture that there is none.

This motivates the construction of an algorithm that will efficiently produce an R-automaton recognizing the language $L(w)$ for an ω-term w, though not necessarily the minimal automaton $\mathcal{A}(w)$. Once this goal is achieved, the computation of $\mathcal{A}(w)$ can then be concluded by minimization of the R-automaton thus obtained. If the computed R-automaton is close to being minimal and one can further optimize the minimization procedure, then the computation of the minimal R-automaton may be close to being optimal. One can then invoke Corollary 2.3 to deduce an efficient solution of our word problem.

So, let us consider an ω-term w. Note that, unless w is a finite word, $\mathcal{T}(w)$ is an infinite tree and of course we do not want to use the word problem to fold it by comparing the pseudowords corresponding to its subtrees. The idea is to look closer at the syntactic structure of w and to relate the values $w(T_p)$ of subtrees of $T = \mathcal{T}(w)$ with certain terms that can be syntactically constructed from w.

The essential reason why the above idea works is best explained in a picture which represents one of a few typical cases, as depicted in Figure 4. The other cases are handled similarly. In it we consider a subtree S of $\mathcal{T}(w)$ and we assume that its root is a descendant of a node along its left line which in turn is the direct right descendant of some node. The value of $w(S)$ is then precisely the factor of w that can be found between the positions in w determined by a and b:



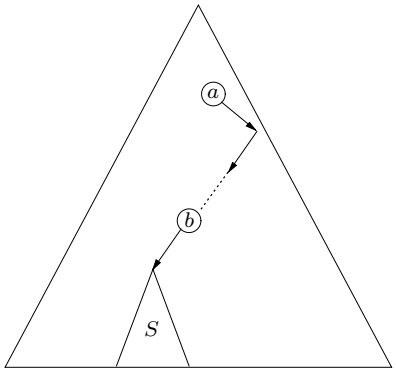


Figure 4. A subtree of $\mathcal{T}(w)$

The value of $w(S)$ may be computed syntactically from the given ω-term w since the indicated b is the first occurrence of this letter to the right of the indicated occurrence of a. Thus, by canonically splitting ω-powers so as to extract any required factors according to the formula

$$(uvw)^\omega = uv(wuv)^\omega,$$

which is valid in R, one can compute the successive values of subtrees until no new values are obtained. More precisely, one may compute, for each position i of a letter in w and each letter b which either occurs to the right of that position in w, or occurs within the same base of an ω-power as the position i, an ω-term $w(i,b)$ corresponding to the value of a subtree. Moreover every such value may be obtained in this way. It is clear that there are at most nm such ω-terms, where $n = |w|$ is the length of w disregarding ω-powers and $m = |c(w)|$ is the number of letters occurring in w.

By arranging appropriately the order in which the calculation of these ω-terms derived from w is performed, one may thus achieve the computation of all values of subtrees of $\mathcal{T}(w)$ in $O(mn)$-time. Moreover, from the computation it is clear that the value of the left descendant of any node v in $\mathcal{T}(w)$ with value $w(i,b)$ is $w(i,c)$, where c is the only letter of $w(i,b)$ such that every other letter in $w(i,b)$ also occurs in $w(i,c)$; if j is the position in w corresponding to the first occurrence of c in $w(i,b)$, the value of the right descendant of v is $w(j,b)$; finally the label of v is c. Thus basically, for each new encountered pair (i,a), one computes the list of letters occurring in $w(i,a)$ and the reverse order in which they are found in terms of

the left basic factorization iterated on the left factors. For convenience of calculation, one adds a position 0 corresponding to the beginning of w and a letter $\#$ to mark the end of w.

Theorem 3.2. *There is an algorithm which computes in time* $O(|w|\cdot|c(w)|)$ *an* R*-automaton recognizing* $L(w)$ *for any given* ω*-term* w*.*

In our earlier example, $w = (ab^\omega a)^\omega$, one starts with the initial state $w(0,\#)$ and looks for the last letter to occur for the first time, namely b. Its position is 2, so the state $w(0,\#)$ is labeled b and it has an edge labeled 0 leading to $w(0,b)$ and an edge labeled 1 leading to $w(2,\#)$. More systematically, one computes successively each row in the next table, where the last three columns are computed immediately from the previous two and are only indicated to facilitate understanding the procedure. In the second and third columns of a row (i,a) we indicate, for each letter, the first position in w "after" position i and before the first occurrence of a to the right of i, where the letter is found, if such a position exists. The term "after" here has to be understood in the sense that if the position i is found within the base of an ω-power, then we are allowed to read again the whole base before leaving the ω-power. Identifying all states labeled ε,

(position, letter)	a	b	label	left	right
$(0,\#)$	1	2	b	$(0,b)$	$(2,\#)$
$(0,b)$	1	–	a	$(0,a)$	$(1,b)$
$(2,\#)$	3	2	a	$(2,a)$	$(3,\#)$
$(0,a)$	–	–	ε	–	–
$(1,b)$	–	–	ε	–	–
$(2,a)$	–	2	b	$(2,b)$	$(2,a)$
$(3,\#)$	1	2	b	$(3,b)$	$(2,\#)$
$(2,b)$	–	–	ε	–	–
$(3,b)$	1	–	a	$(3,a)$	$(1,b)$
$(3,a)$	–	–	ε	–	–

the resulting R-automaton is given in Figure 5 where we add as an index to the state-label the pair (i,a) which determines the state. The minimization of this automaton identifies two pairs of states according to the equalities $w(0,b) = w(3,b)$, $w(0,\#) = w(3,\#)$ and produces exactly the R-automaton of Figure 3. We stress that the fact that we do get these equalities of ω-

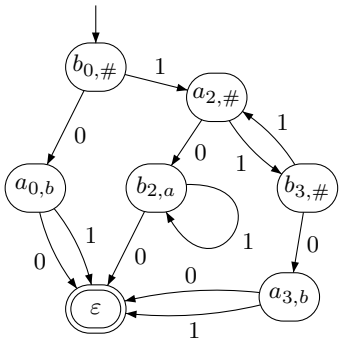


Figure 5. The computed R-automaton of the pseudoword $(ab^\omega a)^\omega$

terms over R comes from automata manipulations rather than by invoking again the word problem.

For the minimization procedure for finite deterministic automata, there is an algorithm due to Hopcroft [13] which works in $O(\ell s \log s)$-time (see [15] for a recent complete complexity analysis), where ℓ is the number of letters and s stands for the number of states. Combining with Theorem 3.2, we obtain the following result.

Theorem 3.3. *There is an algorithm which computes in time* $O(m^2 n \log(mn))$ *the minimal* R*-automaton recognizing* $L(w)$ *for any given* ω*-term* w*, where* $m = |c(w)|$ *and* $n = |w|$*.*

Since R-automata are so special one should be able to do better. The R-automaton of a word is acyclic, and in this case, we already have a linear algorithm due to Revuz [17]. If there are cycles (that is, if we started from a term involving at least one ω-power) then we will adapt this algorithm to completely fold a finite R-automaton in time $O(mn)$.

The additional difficulty for minimizing R-automata comes from the cycles: in the acyclic version, a height function measuring the longest path starting from each state is computed at the beginning of the algorithm. The situation is then simple, in that the minimization can only identify states having the same height.

If we do have cycles, such paths can be infinite. But, due to the specific stucture of R-automata, all cycles are disjoint and 1-labeled. For that reason we can, after a preprocessing phase, treat separately the states belonging to cycles and the other states.

The analog of Revuz's height function is the level, obtained by letting edges in cycles have weight zero. Proceeding then bottom-up, the prepro-

cessing stage consists in rolling paths coming to a cycle, if this does not change the language. Consider for example a usual automaton with a single initial state q_0, one simple path from q_0 to q_1 labeled v and one cycle around q_1 labeled u, as pictured in Figure 6. If $v = u'u^r$ with $r \geq 0$ and u' a suffix

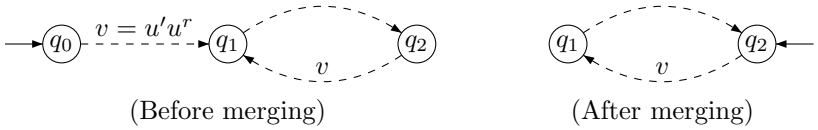


Figure 6. Merging a path ending in a cycle

of u, then we do not change the language by rolling the simple path around the cycle, that is, by only retaining the cycle and choosing as the new initial state the unique state q_2 of the cycle such that $q_2 \cdot v = q_1$. The next step for cycles is to minimize them one by one, *i.e.*, to find the primitive roots of their labels. We then identify, at one level, all equal cycles and all states which do not belong to a cycle. This is essentially Revuz's algorithm.

Finally, notice that rolling paths around cycles may change the level of states that lie above them. We therefore have to recompute this level. The recomputation is local, we just update correctly levels of states we are about to treat.

Here is a more detailed, but informal, commented sketch of the algorithm (see [7] which will be devoted to a detailed presentation and a correctness proof):

(1) Given a finite R-automaton $\mathcal{A}$, compute its cycle structure. For this one can use for instance Tarjan's algorithm [20] which computes the strongly connected components of a graph.
(2) Compute an initial *level* function that measures, for each state, the maximum weight of a path to the terminal state, assigning weight 0 to edges in cycles and weight 1 to all other edges. This can be done efficiently by a simple traversal of the graph that is further used to assign a level value to each edge that is not in a cycle, a value which is initialized to the level of the end state plus 1. Both these level functions will be updated in the main loop of the algorithm as a result of rolling paths with all edges labeled 1 around cycles to which they lead. The level of edges serves as a mechanism to propagate to higher levels changes coming from identifications done at lower levels.

(3) From this point on, we construct successive equivalence relations on sets of states which are approximations to the congruence on $\mathcal{A}$ whose quotient determines the minimized R-automaton. We do so level by level, at each stage suitably joining elements into equivalence classes. The first step consists in putting the final state into its own class.

(4) This is the main cycle in the algorithm. Proceed by increasing level $n \geq 1$, as in the following loop. At the end of level n, all vertices processed in it will have level-value n and they will all be assigned to an equivalence class, which remains unaltered at higher levels.

For each non-terminal state v, denote by $0_v, 1_v$ the edges starting from v labeled $0, 1$, respectively. If $\mathsf{level}(0_v) \leq n$, then let $\zeta(v)$ denote the pair consisting of label $\lambda(v)$ of the state v and the class $[v0]$ containing the state at the end of the edge 0_v.

(a) Call subroutine $\mathsf{Level}(n)$ which returns the list S of states whose current level-value is n.

(b) For each vertex in S which lies in a cycle, put it in its own singleton class.

(c) Roll 1-labeled paths leading to cycles in S around the corresponding cycles by testing for each successive vertex v away from the cycle whether $\zeta(v)$ is defined and whether it coincides with $\zeta(w)$, where w is the unique state in the cycle such that for all sufficiently large k, $v1^k = w1^k$. In the negative case, do not proceed with the test for states u such that $u1 = v$. In the affirmative case, add v to the class of w, as a result of which the edge 1_v becomes a cycle-edge and thus no longer contributes to the level function; this leads us to reduce $\mathsf{level}(v)$ to n and $\mathsf{level}(e)$ to $n+1$ for every edge e which ends at state v.

(d) Since the previous step may change the level functions, lowering to level n states that were previously considered at higher levels, we call subroutine $\mathsf{Level}(n)$ again. This will return an updated value for S which contains the previous value since the previous step only affects the level-values of states at higher levels.

(e) For each cycle C in S, do the following steps which suitably merges all equivalence classes of states in the cycle according to their identification in the minimized R-automaton:

i. compute the (circular) word W_C whose letters are the successive $\zeta(w)$ with w in C;
ii. compute the primitive W'_C root of W_C; this can be done by computing the shortest border[a] u of W_C such $u^{-1}W_C$ is also a border; that this computation can be performed in linear time in terms of the length of W_C follows from the fact that the list of all borders can be computed within this time-complexity [11];
iii. compute the minimal conjugate V_C of W'_C; this can be done in linear time in terms of the length W'_C [9, 19];
iv. merge classes of states in C according to the periodic repetition of V_C in W_C.

(f) To merge classes of states in different cycles C of S start by lexicographically sorting the words V_C using bucket sort [10]. This determines in particular which cycles have the same words $V = V_C$ and their classes associated with corresponding positions in V are merged.
(g) To merge the remaining states v in S into classes start by lexicographically sorting (by a bucket sort) the associated triples $(\lambda(v), [v_1], [v_2])$, where v_1, v_2 denote the ends of the edges $0_v, 1_v$, respectively. As in the previous step, this determines in particular which states have the same associated triples, and those that do are merged into the same class.
(h) Increment n by 1 and proceed until a subroutine call returns the empty list.

To complete the description of the algorithm, it remains to indicate what the subroutine $\mathsf{Level}(n)$ does. It starts by updating the level-value of the beginning state v of each edge e such that $\mathsf{level}(e) = n$ according to the formula

$$\mathsf{level}(v) = \begin{cases} \max\{\mathsf{level}(e), \mathsf{level}(f)\} & \text{if } e \text{ is not in a cycle} \\ & \text{and } \{e, f\} = \{0_v, 1_v\} \\ \max_x \mathsf{level}(x) & \text{otherwise} \end{cases}$$

where the second maximum runs over all edges x with label 0 which start in the cycle that contains v. Then return all states for which the new

[a] A border of a word w is a word which is both a prefix and a suffix of w.

level-value is n.

Theorem 3.4. *The above algorithm minimizes a given* R*-automaton with s states in time $O(s)$.*

Taking into account Corollary 2.3, this gives our solution of the word problem:

Theorem 3.5. *The word problem for ω-terms over* R *can be solved in time $O(mn)$, where m is the number of letters involved and n is the maximum of the lengths of the ω-terms to be tested.*

4. Equations for ω-terms

Consider the set Σ consisting of the following identities:

$$(xy)^\omega x^\omega = (xy)^\omega x = x(yx)^\omega = (xy)^\omega$$
$$(x^\omega)^\omega = x^\omega$$
$$(x^r)^\omega = x^\omega \quad (r \geq 2)$$

It is immediately checked that R satisfies these identities (viewed as pseudoidentities).

Theorem 4.1. *The set Σ is a basis for the variety R^κ of ω-semigroups generated by* R*.*

The proof of this result depends on the construction of canonical forms presented in the next section. There we will show that the canonical form $\mathsf{cf}(w)$ of an ω-term w uniquely determines the value of w over R and that the equality $w = \mathsf{cf}(w)$ can be formally deduced from Σ. Hence, if two ω-terms v and w have the same value over R, then they have the same canonical form and therefore the identity $v = w$ may be formally deduced from Σ. This will show that indeed Σ is a basis of identities for R^κ.

Using the completeness theorem for equational logic, it is now easy to deduce that

Corollary 4.2. *The variety R^κ is not finitely based.*

The proof of Corollary 4.2 consists in verifying that the semigroup

$$S_p = \langle a, e, f : a^p = 1,\ ea = ef = e^2 = e,\ fa = fe = f^2 = f,$$
$$ae = e,\ af = f\rangle,$$

where p is a positive integer, with the unary operation σ defined by taking

$$\sigma(e) = e,\ \sigma(f) = f,\ \sigma(1) = e,\ \sigma(a^k) = f\ (k \in \mathbb{Z} \setminus p\mathbb{Z})$$

as the ω-power, satisfies the identities in the basis Σ for r relatively prime with p, but fails $(x^p)^\omega = x^\omega$.

5. Canonical forms

From a finite R-automaton one can read off an ω-term in a canonical way. The definition is recursive in the number of states:

- for the trivial R-automaton, in which there is only one state, take the empty pseudoword 1;
- for an R-automaton in which the initial state r, labeled a, is not the end of an edge labeled 1, take uav where u and v are respectively the ω-terms canonically associated with the R-subautomata obtained by changing the initial state respectively to $r0$ and $r1$;
- for an R-automaton in which the initial state r, labeled a, is the end of an edge labeled 1, take $(uav)^\omega$ where u and v are respectively the ω-terms canonically associated with the R-subautomata obtained by changing the initial state respectively to $r0$ and $r1$ and, in the second case, changing the end of the edge labeled 1 pointing to r to a terminal state.

We call the resulting ω-term for $\mathcal{A}(w)$ the *canonical form* of a given ω-term w and denote it $\mathsf{cf}(w)$. For example, since the minimal R-automaton of the ω-term $w = (ab^\omega a)^\omega$ is that given by Figure 3, the canonical form is $\mathsf{cf}(w) = (abb^\omega a)^\omega$.

A factor u of an ω-term w is *fringy* if there is a factorization $u = va$ with $c(u) = c(w)$ and a a letter which is not in $c(v)$. A Σ-*fringy decomposition* of an ω-term w is a finite sequence $w_1, \ldots, w_n$ of ω-terms such that each w_i is a fringy factor of $w_1 \cdots w_n$ and the identity $w = w_1 \cdots w_n$ is a consequence of Σ. The following technical result is used to compute the canonical form.

Proposition 5.1. *Let w be an ω-term.*

(1) If $w = uav$, a is a letter, and the factor ua is fringy, then it is possible to deduce the identity $\mathsf{cf}(w) = \mathsf{cf}(u)\, a\, \mathsf{cf}(v)$ from Σ.
(2) If w admits a Σ-fringy decomposition, then the identity $\mathsf{cf}(w^\omega) = (\mathsf{cf}(w))^\omega$ is a consequence of Σ.
(3) Let w_0 and w be ω-terms such that

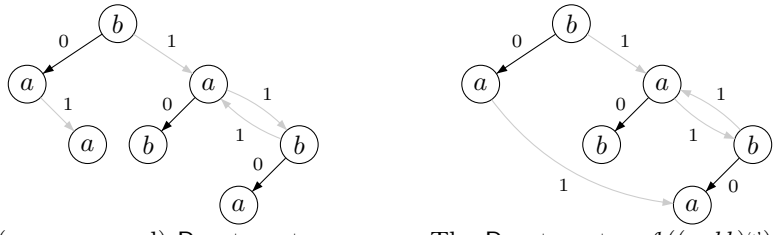
- $\mathsf{R} \not\models w = w^2$,
- $c(w_0) \subseteq c(w)$,
- *either* $c(w_0) \subsetneq c(w)$, *or* $\mathsf{R} \not\models w_0 = w_0^2$.

Then, there are ω*-terms* u, v *such that* $uv = w$ *is a consequence of* Σ *and there are positive integers* r, s *such that* $w_0 w^r u$ *and* $(vu)^s$ *admit* Σ*-fringy decompositions.*

Rules (1) and (2) allow to "push" cf operators downwards in the term tree of an ω-term, while preserving equality modulo Σ. However, they only can be used starting from particular forms, namely where fringy factors occur. To make fringy factors appear, we use item (3). For instance, we may deduce from Σ the following identities:

$$\begin{aligned}
\mathsf{cf}((ab^\omega a)^\omega) &= \mathsf{cf}\big((ab \cdot b^\omega a)^\omega\big) && \text{(note that } ab \text{ and } b^\omega a \text{ are fringy)}\\
&= \big(\mathsf{cf}(ab \cdot b^\omega a)\big)^\omega && \text{by (2)}\\
&= \big(\mathsf{cf}(a) \cdot b \cdot \mathsf{cf}(b^\omega a)\big)^\omega && \text{by (1)}\\
&= (ab\,\mathsf{cf}(b^\omega)a)^\omega && \text{by (1)}\\
&= (ab\,\mathsf{cf}(b)^\omega a)^\omega && \text{by (2)}\\
&= (abb^\omega a)^\omega = (ab^\omega a)^\omega
\end{aligned}$$

In this example, fringy factors needed to apply Proposition 5.1 appear with almost no extra work. As another example, consider $w = (aabb)^\omega$, whose automaton and wrapped automaton computed by our algorithms are shown in Figure 7. Using $\mathcal{A}((aabb)^\omega)$ and the definition of the canonical form, we



A (non wrapped) R-automaton The R-automaton $\mathcal{A}((aabb)^\omega)$

Figure 7. R-automata associated to $(aabb)^\omega$

get $\mathsf{cf}((aabb)^\omega) = aab(baab)^\omega$. On the other hand, one deduces from Σ,

using Proposition 5.1:

$$\begin{aligned}
\mathsf{cf}((aabb)^\omega) &= \mathsf{cf}\big(aab(ba\cdot ab)^\omega\big) && \text{using } x(yx)^\omega = (xy)^\omega\\
&= aab\cdot \mathsf{cf}\big((ba\cdot ab)^\omega\big) && \text{by Prop. 5.1 (1)}\\
&= aab\cdot \big(\mathsf{cf}(ba\cdot ab)\big)^\omega && \text{by Prop. 5.1 (1)}\\
&= aab(baab)^\omega
\end{aligned}$$

The first equality $(aabb)^\omega = aab(ba\cdot ab)^\omega$ deduced by Σ makes fringy factors appear. Proposition 5.1 (3) essentially states that one can always produce these fringy factors when they are needed.

These examples illustrate how Proposition 5.1 may be used to prove the following theorem already announced in Section 4.

To prove that Σ implies $\mathsf{cf}(w) = w$, we start with $\mathsf{cf}(w)$. Then, using Σ, one may split ω-powers by bringing out of them fringy factors and push inside the canonical form "calculation" at the cost of taking a power of a conjugate of the original base of the ω-power, and thus reduce the number of nested ω-powers. For fringy factors, we also use Proposition 5.1 to bring out the last letter and thus reduce the content. In either type of step, a non-negative integer parameter is reduced and therefore this process terminates in a finite number of steps. Once no more canonical forms remain to be "computed", all transformations can be traced back without the cf operator using Σ to recover the ω-term w.

Theorem 5.2. *For every ω-term w, the identity $\mathsf{cf}(w) = w$ is a consequence of Σ.*

Note that we do not claim that the above procedure serves to compute the canonical form of a given ω-term. But, even if it would serve that purpose, the algorithm to solve the word problem obtained by such a computation followed by a test of graphical equality of the canonical forms would not be efficient. This is shown by the following example in which the canonical form computation has to take too long to perform.

Indeed the canonical form of an ω-term may be exponentially longer than the given ω-term. This is so for the sequence of ω-terms

$$w_k = (\cdots((a_1a_1b_1b_1)^\omega a_2a_2b_2b_2)^\omega\cdots a_ka_kb_kb_k)^\omega.$$

For instance, $k = 1$ corresponds to the ω-term $(aabb)^\omega$, whose canonical form was previously computed using $\mathcal{A}((aabb)^\omega)$. For $k = 2$, the automaton

is more involved. By the same method, one can compute its canonical form:

$$\underbrace{aab(baab)^\omega ccd}_{u_0}\cdot\Big(\underbrace{daab(baab)^\omega c}_{u_1}\cdot\underbrace{cddaab}_{u_2}\cdot\underbrace{(baab)^\omega ccd}_{u_3}\Big)^\omega$$

where $a = a_1$, $b = b_1$, $c = a_2$, $d = b_2$. Here, u_0 is a fringy factor of this factorization, and u_1, u_2, u_3 are fringy factors of the external ω-power.

An ω-term w is *reduced* if it has no subterm of the form $(x^r)^\omega$ with $r > 1$, no subterm of the form $y^\omega z$, with $c(z) \subseteq c(y)$, and no subterm of the form $(xy^\omega z)^\omega$, where x, z may be empty and $c(xz) \subseteq c(y)$. Note that Σ may be used to deduce the equality of any ω-term with a shorter one in reduced form. Surprisingly, a canonical factorization is not necessarily reduced. For instance, the ω-term inside the ω-power of $\mathsf{cf}(w_2)$ is a square.

Coming back to the calculation of $\mathsf{cf}(w_k)$, one can show that, in general:

$$|w_k| = 4k,\ |c(w_k)| = 2k,\ |\mathsf{cf}(w_k)| = (25\cdot 3^{k-1} - 11)/2.$$

Another example is given by the sequence v_k defined by

$$v_k = (a_k(\cdots(a_2(a_1)^\omega)^\omega\cdots)^\omega.$$

Here we can compute $|v_k|$ to be k, and the number of states of $\mathcal{A}(v_k)$ to be $(k+2)(k+3)/2 - 5$, while the number of states of the R-automaton of v_k computed by our algorithm is $(k+2)(k+3)/2 - 3$. This shows that when we allow m and n to grow at equal rates, both the algorithm of Section 3 to compute a finite R-automaton for an ω-term and the minimization algorithm have optimal complexity rate.

It is also possible to have an exponential decrease in length in going from an ω-term to its canonical form, even for ω-terms for which there are no obvious reductions, as shown by Proposition 5.3 below.

Let us examine next why the first strategy that comes to mind to reduce ω-terms does not work. It would consist in taking rules that result from our proposed basis of identities for the variety of ω-semigroups generated by R by orienting the identities in the length-reducing direction:

r1. $(xy)^\omega x \to (xy)^\omega$
r2. $(xy)^\omega x^\omega \to (xy)^\omega$
r3. $(x^\omega)^\omega \to x^\omega$
r4. $(x^r)^\omega \to x^\omega$ $(r \geq 2)$
r5. $x(yx)^\omega \to (xy)^\omega$

In general, for any set of reduction rules, its use is the following: whenever in a term w we find a subterm u which matches the pattern of the left

side of a rule $\ell \to r$, then we may replace in w an occurrence of u by the resulting value of applying the rule to u to obtain a new ω-term w'; we then write $w \Rightarrow w'$. If none of the above reduction rules may be applied to an ω-term w then we say that w is *irreducible*.

Proposition 5.3. *Consider the sequence z_n, defined recursively by $z_0 = 1$, $z_{n+1} = (z_n a_n z_n)^\omega$, where the a_n are distinct letters. Then the z_n are irreducible ω-terms such that $|z_n| = 2^n - 1$ and $|\mathsf{cf}(z_n)| = n$.*

In particular, it follows from Proposition 5.3 that the system of reduction rules (r1)–(r5) cannot be confluent since it is length-reducing. One can try to apply the *Knuth-Bendix algorithm* by adding rules to obtain confluence without losing the Noetherian property. But, since our system is essentially infinite (not only because of the parameter $r \geq 2$, which does not bother much here but because x, y stand for arbitrary terms, that is they serve as a means of giving patterns for terms), this quickly becomes an unmanageable task.

6. Comparison with related work of Bloom and Choffrut

Bloom and Choffrut [8] have established results which bare some similarities with those reported in this paper. They consider posets labeled with elements of a finite nonempty alphabet under two operations:

- *series concatenation*: PQ extends the orders of each factor by declaring the elements in P to precede all elements in Q;
- *ω-power*: $P^\omega = \omega \times P$ under lexicographic order.

The finite or countable labeled posets over an alphabet A, considered up to isomorphism, constitute an algebra $P(A)$ which satisfies the following identities:

- $x(yz) = (xy)z$,
- $(xy)^\omega = x(yx)^\omega$,
- $(x^n)^\omega = x^\omega$ for $n \geq 1$.

The subalgebra generated by the singletons labeled with letters a from an alphabet A is denoted $W(A)$. Note that its elements are labeled ordinals of length less than ω^ω.

A *tail* of an ordinal α with labeling function f corresponding to an ordinal $\beta < \alpha$ is the unique ordinal γ such that $\beta + \gamma = \alpha$ with the labeling

function $g(\delta) = f(\beta+\delta)$. A labeled ordinal is *tail-finite* if it has only finitely many tails.

The main results in the paper [8] are the following:

(1) A labeled ordinal belongs to $W(A)$ if and only if it has length less than ω^ω and it is tail-finite, if and only if it is defined by an ω-term.
(2) The variety V generated by the $P(A)$ is equal to the variety generated by the $W(A)$ and it is defined by the above identities. Moreover, $W(A)$ is free over A in this variety.
(3) The variety V is not finitely based.
(4) The equality of two ω-terms u_1, u_2 in V can be decided (with the help of *Choueka automata*) in $O(m^2 n_1^2 n_2^2)$-time, where n_i denotes the length of u_i and m the maximum of their heights in terms of nested ω-powers.

Our results are similar, so the natural question is if they are connected by some causal relationship. Our basis of identities for R^κ is obtained from the above basis for V by adding the identities

$$(x^\omega)^\omega = x^\omega, \quad (xy)^\omega x^\omega = (xy)^\omega x = (xy)^\omega.$$

In particular, we have the inclusion $\mathsf{R}^\kappa \subseteq V$ which does not appear to be obvious taking into account the rather different types of generators considered for the two varieties. That the inclusion is proper is easy to show: for instance, the identity $x^\omega x = x^\omega$ fails in the algebra $P(A)$ since the ordinals $\omega + 1$ and ω labeled with only one letter are distinct.

Finally, since the two varieties are different, the solution of the word problem for the free algebra in one of the varieties is insufficient to solve the word problem for the other. A curious difference in the two algorithms is that the complexity of our algorithm depends on the size of the alphabet whereas that of Bloom and Choffrut does not depend directly on this parameter and instead depends on the height of nested ω-powers. Note however that for reduced ω-terms in the sense of Section 5, the indicated height is bounded by the number of letters which appear in the term.

Acknowledgment.

The authors wish to thank the anonymous referees for their careful reading of a preliminary version of the paper and for their comments.

References

1. J. Almeida. Finite semigroups: an introduction to a unified theory of pseudovarieties. In G. M. S. Gomes, J.-E. Pin, and P. V. Silva, editors, *Semigroups, Algorithms, Automata and Languages*, pages 3–64, Singapore, 2002. World Scientific.
2. J. Almeida and A. Azevedo. The join of the pseudovarieties of $\mathcal{R}$–trivial and $\mathcal{L}$–trivial monoids. *J. Pure Appl. Algebra*, 60:129–137, 1989.
3. J. Almeida and B. Steinberg. On the decidability of iterated semidirect products and applications to complexity. *Proc. London Math. Soc.*, 80:50–74, 2000.
4. J. Almeida and P. G. Trotter. Hyperdecidability of pseudovarieties of orthogroups. *Glasgow Math. J.*, 43:67–83, 2001.
5. J. Almeida and P. Weil. Free profinite -trivial monoids. *Int. J. Algebra Comput.*, 7:625–671, 1997.
6. J. Almeida and M. Zeitoun. An automata-theoretic approach of the word problem for ω-terms over R. In preparation.
7. J. Almeida and M. Zeitoun. A linear time minimization algorithm for disjoint loop automata. In preparation.
8. S.L. Bloom and Ch. Choffrut. Long words: The theory of concatenation and ω-power. *Theor. Comp. Sci.*, 259:533–548, 2001.
9. K. S. Booth. Lexicographically least circular substrings. *Inform. Process. Lett.*, 10:240–242, 1980.
10. Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.
11. M. Crochemore and W. Rytter. *Text Algorithms*. The Clarendon Press Oxford University Press, New York, 1994. With a preface by Zvi Galil.
12. P. M. Higgins. Divisors of semigroups of order-preserving mappings of a finite chain. *Int. J. Algebra Comput.*, 5:725–742, 1995.
13. J. E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Z. Kohavi, editor, *Theory of machines and computations (Proc. Internat. Sympos., Technion, Haifa, 1971)*, pages 189–196, New York, 1971. Academic Press.
14. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Mass., 1979.
15. T. Knuutila. Re-describing an algorithm by Hopcroft. *Theor. Comp. Sci.*, 250:333–363, 2001.
16. J.-E. Pin. *Varieties of Formal Languages*. Plenum, London, 1986. English translation.
17. D. Revuz. Minimisation of acyclic deterministic automata in linear time. *Theor. Comp. Sci.*, 92:181–189, 1992.
18. M. P. Schützenberger. Sur le produit de concaténation non ambigu. *Semigroup Forum*, 13:47–75, 1976.
19. Y. Shiloach. Fast canonization of circular strings. *J. Algorithms*, 2:107–121, 1981.
20. R. E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal*

on Computing, 1(2):146–160, June 1972.
21. P. Tesson and D. Thérien. Diamonds are forever: the variety DA. In G. M. S. Gomes, J.-E. Pin, and P. V. Silva, editors, *Semigroups, Algorithms, Automata and Languages*, pages 475–499, Singapore, 2002. World Scientific.
22. D. Thérien and Th. Wilke. Over words, two variables are as powerful as one quantifier alternation. In *STOC '98 (Dallas, TX)*, pages 234–240. ACM, New York, 1999.