THE WORD PROBLEM FOR ω-TERMS OVER DA

A. MOURA

Abstract. In this paper, we solve the word problem for ω-terms over DA. We extend to DA the ideas used by Almeida and Zeitoun to solve the analogous problem for the pseudovariety R applying also a representation by automata of implicit operations on DA, which was recently obtained by the author. Considering certain types of factors of an implicit operation on DA, we can prove that a pseudoword on DA is an ω-term if and only if the associated minimal DA-automaton is finite. Finally, we complete the result by effectively computing in polynomial time the minimal DA-automaton associated to an ω-term.

1. Introduction

The pseudovariety DA, the class of finite monoids whose regular $\mathcal{D}$-classes are aperiodic semigroups, has been the subject of recent studies due to its various applications. It is known that languages whose syntactic monoids lie in DA have important algebraic, combinatorial, automata-theoretical and logical characterizations that enable us to solve problems in computational and complexity theory (see Tesson and Thérien [13]).

On the other hand, word problems have long played an important role in various branches of Mathematics. In this paper, we solve the word problem for ω-terms over DA, which consists of deciding if two ω-terms are equal over all elements of this pseudovariety. Almeida and Zeitoun [5, 4] solved the analogous problem for the pseudovariety R. Based on this work, we characterize ω-terms over DA by the finiteness of certain types of sets of factors and by the finiteness of the associated minimal DA-automaton. We also construct in polynomial time this minimal DA-automaton.

In [11], we exhibited three representations of implicit operations over DA: by means of labeled trees of finite height, by means of *quasi-ternary* labeled trees, and by means of labeled linear orderings. The paper has also an improvement of the representation by *quasi-ternary* labeled trees, which may be infinite, consisting of *wrapping* the DA-tree of an implicit operation. We obtain a representation by means of DA-automata and we prove here that an ω-term has a finite representation by the minimal DA-automaton. Since this paper depends on several definitions and results from [11], the reader should refer to that paper as needed.

The paper is organized as follows. In Section 2, we introduce some notions and notation about implicit signatures, concluding the corresponding section from [11]. We also recall the notion of central basic factorization of an implicit operation on DA and the representation of implicit operations on DA by automata. Based on Almeida and Zeitoun [5], we construct in Section 3 certain types of factors of

2000 *Mathematics Subject Classification.* 20M05; 20M07; 20M35.

Key words and phrases. Finite monoid; pseudovariety; word problem; pseudoword; omega-term; aperiodic; regular D-class.

an implicit operation. This allows us to characterize an ω-term on DA by the finiteness of these sets of factors and by the finiteness of the associated minimal DA-automaton, which is done in Section 4. Finally, in Section 5 we exhibit an algorithm to compute a finite DA-automaton associated to an ω-term and we prove that the minimal DA-automaton associated to an ω-term can be constructed in polynomial time.

2. Preliminaries

We complete the introduction of notions and notation given in the corresponding section of [11]. For further information on the basic background see, for instance, [1, 3].

In this paper, $\overline{\Omega}_A\mathsf{V}$ denotes the *free pro-*V *monoid on* A. The *natural interpretation* of $u \in \overline{\Omega}_A\mathsf{V}$ in a pro-V monoid S is the mapping $u_S : S^A \to S$ which associates to each function $\mu : A \to S$ the element $\hat{\mu}(u) \in S$. For $u \in \overline{\Omega}_A\mathsf{V}$, the sequence $(u^{n!-1})_n$ converges and we denote the limit by $u^{\omega-1}$. Similarly, u^ω denotes the limit of the sequence $(u^{n!})_n$, which is the unique idempotent in the closed subsemigroup generated by u. The elements of $\overline{\Omega}_A\mathsf{V}$ are called *implicit operations over* V or *pseudowords over* V. Usually, the first name is used when these elements are viewed, via their natural interpretation, as operations on finite semigroups, and the second name is used when the elements are viewed as combinatorial entities generalizing finite words. Recall that, if $\iota : A \to \overline{\Omega}_A\mathsf{V}$ is the natural generating function, then the submonoid generated by $\iota(A)$ is a dense submonoid of $\overline{\Omega}_A\mathsf{V}$.

An *implicit signature* is a set of implicit operations containing the monoid multiplication, $\_\cdot\_$. The *canonical signature* $\kappa = \{\_\cdot\_, \_^{\omega-1}\}$ consists of the monoid multiplication and the unary $(\omega-1)$-power. A *κ-term on the set* A is an element of the unary semigroup T_A^κ freely generated by A , and $\Omega_A^\kappa\mathsf{V}$ is the κ-submonoid of the pro-V monoid freely generated by A, whose elements are called *κ-words* or *κ-terms over* V. A *κ-identity over* V is an equality $u = v$, with u and v κ-words over V. The *κ-word problem for* V consists in deciding if two κ-terms of T_A^κ have the same image under the natural homomorphism into the free pro-V monoid, $\iota : T_A^\kappa \to \overline{\Omega}_A\mathsf{V}$. The signature $\omega = \{\_\cdot\_, \_^\omega\}$ is also of interest. Since, in an aperiodic monoid, any κ-term coincides with the ω-term obtained by replacing all $(\omega-1)$-powers by ω-powers, we can work and formulate the results in terms of the signature ω, which we do from hereon.

In Section 5, we adopt the simplified notation of McCammond [10] using the curved parentheses to represent the ω-power, and so, the ω-terms are seen as words on the extended alphabet $A \cup \{(,)\}$.

Given $w \in \overline{\Omega}_A\mathsf{DA}\backslash\{1\}$, we consider the central basic factorization of w, under the conditions described by Almeida [2], as the tuple $(\alpha, a, \gamma, b, \beta) \in \overline{\Omega}_A\mathsf{DA} \times A \times \overline{\Omega}_A\mathsf{DA} \times A \times \overline{\Omega}_A\mathsf{DA}$ or as the triple $(\alpha, a, \beta) \in \overline{\Omega}_A\mathsf{DA} \times A \times \overline{\Omega}_A\mathsf{DA}$ satisfying one of the following conditions:

(i) **standard form:** $w = \alpha a\gamma b\beta$ with $a, b \in A$, $\alpha, \beta, \gamma \in \overline{\Omega}_A\mathsf{DA}$, $a \notin c(\alpha)$, $b \notin c(\beta)$ and $c(\alpha a) = c(b\beta) = c(w)$;
(ii) **overlapped form:** $w = \alpha b\gamma a\beta$ with $a, b \in A$, $\alpha, \beta, \gamma \in \overline{\Omega}_A\mathsf{DA}$, $a \notin c(\alpha b\gamma)$, $b \notin c(\gamma a\beta)$ and $c(\alpha b\gamma a) = c(b\gamma a\beta) = c(w)$;
(iii) **degenerate form:** $w = \alpha a\beta$ with $a \in A$, $\alpha, \beta \in \overline{\Omega}_A\mathsf{DA}$, $a \notin c(\alpha)$, $a \notin c(\beta)$ and $c(\alpha a) = c(a\beta) = c(w)$.

Almeida proved that this factorization exists and is unique and we denote it by $\mathsf{CBF}(w)$. We iterate this factorization by applying it to the central factor γ until it becomes 1 or the central basic factorization is of the degenerate form. We denote this iterated central basic factorization (called of type 2 in [11]) by $\mathsf{I_2CBF}(w)$ and it has one of the following forms: $\mathsf{I_2CBF}(w) = \alpha_1 a_1 \cdots \alpha_n a_n b_n \beta_n \cdots b_1 \beta_1$, $\mathsf{I_2CBF}(w) = \alpha_1 a_1 \cdots \alpha_n a_n \beta_n \cdots b_1 \beta_1$ or $\mathsf{I_2CBF}(w) = \alpha_1 a_1 \cdots \ \cdots b_1 \beta_1$.

To solve the word problem over DA, we use a result from [11] which states that two pseudowords have the same DA-*quasi-ternary* tree if and only if they are equal over DA. As these DA-trees may be infinite, and, as such, they may not be calculated in full form, we use the improvement of this representation which consists of representing the implicit operations on DA by means of DA-automata.

Briefly, the tree $t(w) \in T_2(A)$ which represents the pseudoword $w \in \overline{\Omega}_A\mathsf{DA}$ is constructed recursively as follows: it has a root corresponding to the pseudoword w and, assuming that $\mathsf{CBF}(w) = \alpha a \gamma b \beta$, the root is labeled by the pair (a, b) and it has three sons, corresponding to the pseudowords α, γ and β, with edges labeled 0, 1 and 2, respectively. If $\mathsf{CBF}(v) = \alpha a \beta$ is degenerate, for some vertex corresponding to a pseudoword v, then this vertex is labeled a and it has only two sons with edges labeled 0 and 2, respectively. Any tree of $T_2(A)$ is a DA-automaton, $t(w) = (V, \to, q, F, \lambda)$, where q is the root, F is the set of vertices corresponding to the empty word and λ is the state labeling function. The *wrapped* DA-*automaton* of w, $\mathcal{A}(w)$, is the automaton obtained from $t(w)$ by identifying states corresponding to the same pseudoword. Moving the label of each state and adding it to the labels of the edges starting in such state, we obtain an automaton that recognizes the language associated to w, $\mathcal{L}(w)$ (see [11]), and which is minimal for the condition $\mathcal{L}(\mathcal{A}) = \mathcal{L}(w)$. We end this section with the following powerful result from [11]:

Proposition 2.1. *Let $v, w \in \overline{\Omega}_A\mathsf{DA}$. Then $\mathsf{DA} \models v = w$ if and only if $\mathcal{L}(v) = \mathcal{L}(w)$.*

3. Factors of a pseudoword over DA

In this paper, we prove that the word problem can be effectively solved when we work with ω-terms over DA. We start by considering certain types of factors of a pseudoword $w \in \overline{\Omega}_A\mathsf{DA}$.

Let $w \in \overline{\Omega}_A\mathsf{DA}$. We define certain sets of factors of w: $\mathcal{F}(w)$, which consists of the so called DA-*factors* of w; $\mathcal{R}(w)$, consisting of the *relative remainders* of w; and $\mathcal{S}(w)$, the set of the *absolute remainders* of w.

We define $f_\delta(w)$, $l_{(\delta,0)}(w)$ and $l_{(\delta,2)}(w)$ by induction on the length of $\delta \in \{0,1,2\}^*$ as follows:

$$\begin{aligned} f_\varepsilon(w) &= w \\ (f_{\delta 0}(w), l_{(\delta,0)}(w), f_{\delta 1}(w), l_{(\delta,2)}(w), f_{\delta 2}(w)) &\stackrel{def}{=} \mathsf{CBF}(f_\delta(w)) \\ \text{or } (f_{\delta 0}(w), l_{(\delta,0)}(w), f_{\delta 2}(w)) &\stackrel{def}{=} \mathsf{CBF}(f_\delta(w)) \end{aligned}$$

depending on whether the central basic factorization of $f_\delta(w)$ is of the standard or of the overlapped form, or if it is of the degenerate form. The set of DA-factors of w is

$$\mathcal{F}(w) = \{f_\delta(w) \mid \delta \in \{0,1,2\}^* \text{ and } f_\delta(w) \text{ is defined}\} \subseteq \overline{\Omega}_A\mathsf{DA}.$$

It consists of the set of images under $\rho : T_2(A) \mapsto \overline{\Omega}_A\mathsf{DA}$ of the subtrees $t(w)$, which correspond to some factor of the form α_δ, β_δ or γ_δ of the iterated factorization of some factor of the iterated central basic factorization of type 2 of w.

The set of relative remainders of w is the set $\mathcal{R}(w)$ of elements of $\mathcal{F}(w)$, which consists of the images under ρ of the subtrees attached to a vertex which is a son from a central branch of a given vertex. These subtrees are the trees corresponding to the factors γ_δ of some iterated factorization of a factor of the iterated central basic factorization of type 2 of w. Formally, we write

$$\mathcal{R}(w) = \{f_\delta(w) \mid \delta \in \{0,1,2\}^*1 \text{ and } f_\delta(w) \text{ is defined}\} = f_1(\mathcal{F}(w)).$$

Let $u, v \in \overline{\Omega}_A\mathsf{DA}$ be such that u is a prefix of v. We use the notation $u^{-1}v$ to represent any suffix of v such that $v = u \cdot u^{-1}v$ in $\overline{\Omega}_A\mathsf{DA}$. Similarly, if u is a suffix of v, we use vu^{-1} to denote any prefix of v such that $v = vu^{-1} \cdot u$ in $\overline{\Omega}_A\mathsf{DA}$. We define the set of absolute remainders of w, $\mathcal{S}(w)$, to be the smallest subset containing w and satisfying to the following conditions:

(i) $u \in \mathcal{S}(w) \Rightarrow f_0(u) \in \mathcal{S}(w)$;
(ii) $u \in \mathcal{S}(w) \Rightarrow f_2(u) \in \mathcal{S}(w)$;
(iii) $u, v \in \mathcal{S}(w)$, $a \in A$ and $\mathbf{o}(ua)$ is an initial segment of $\mathbf{o}(v)$ implies that $(ua)^{-1}v \subseteq \mathcal{S}(w)$;
(iv) $u, v \in \mathcal{S}(w)$, $a \in A$ and $\mathbf{o}(au)$ is a final segment of $\mathbf{o}(v)$ implies that $v(au)^{-1} \subseteq \mathcal{S}(w)$.

Recall that $\mathbf{o}(u)$ is the reduced $*$-labeled linear ordering representing $u \in \overline{\Omega}_A\mathsf{DA}$, notation introduced in [11]. See Rosenstein [12] for the basics on linear orderings. Let us see the relation that exists between the elements of $\mathcal{S}(w)$ and the closed intervals of $\mathbf{o}(w)$, starting by observing some auxiliary results.

Lemma 3.1. *Given $w \in \overline{\Omega}_A\mathsf{DA}$, we have $\mathcal{F}(w) \subseteq \mathcal{S}(w)$.*

Proof. We obviously have $w \in \mathcal{S}(w)$. By conditions (i) and (ii) we have, respectively, the elements $f_0(w)$ and $f_2(w)$ in $S(w)$. By [11, Lemma 4.16], it follows that, for each $a \in A$, there exist the smallest position of $\mathbf{o}(w)$ labeled a, $p_a^{o(w)}$, and the largest position of $\mathbf{o}(w)$ labeled a, $\bar{p}_a^{o(w)}$. Therefore, and since A is finite, there exist $p^{o(w)} = \max\{p_a^{o(w)} | a \in A\}$ and $\bar{p}^{o(w)} = \min\{\bar{p}_a^{o(w)} | a \in A\}$. By definition of $f_0(w)$ and $f_2(w)$, it follows that $\mathbf{o}(f_0(w)l(p^{o(w)}))$ is an initial segment of $\mathbf{o}(w)$ and $\mathbf{o}(l(\bar{p}^{o(w)})f_2(w))$ is a final segment of $\mathbf{o}(w)$. By conditions (iii) and (iv), $f_1(w) \in S(w)$. Proceeding inductively on the factors $f_0(w)$, $f_1(w)$ and $f_2(w)$, we deduce that all the elements of $\mathcal{F}(w)$ are in $\mathcal{S}(w)$. □

Lemma 3.2. *Let $w \in \overline{\Omega}_A\mathsf{DA}$. For each position p in $\mathbf{o}(w)$, there exists a closed interval $\mathbf{o}' \subseteq \mathbf{o}(w)$ such that:*

(i) *$\mathbf{o}' \simeq \mathbf{o}(f_\delta(w))$ with $f_\delta(w) \in \mathcal{F}(w)$;*
(ii) *$p = p^{o'}$ or $p = \bar{p}^{o'}$.*

Proof. We proceed by induction on the content of w, $c(w)$. If $|c(w)| = 1$, suppose that $c(w) = \{a\}$, then $w = a^n$, with n finite, or $w = a^\omega$. If $w = a^n$, with n finite, then $\mathbf{o}(w) = \mathbf{n}$. In this case, $\mathbf{o}' = \mathbf{o}(f_{1^{p-1}}(w))$, if $p \leq \lceil n/2 \rceil$, or $\mathbf{o}' = \mathbf{o}(f_{1^{n-p}}(w))$, if $p > \lceil n/2 \rceil$ satisfies the desired conditions (it is enough to observe that in the iterated central basic factorization of w the factors α_i and β_i are all empty and each letter a at a given position is a distinguished label in a position $p^{o'}$ or $\bar{p}^{o'}$ of some iteration). In the case where $w = a^\omega$, we have $\mathbf{o}(w) = \omega + \omega^*$. If p is a position in ω, then we set $\mathbf{o}' = \mathbf{o}(f_{1^{p-1}}(w))$. Otherwise, we set $\mathbf{o}' = \mathbf{o}(f_{1^{q-1}}(w))$, where q is the positive integer corresponding to the position p in $\omega + \omega^*$ when we

count from right to left. In any case, the chosen orderings satisfy the conditions (i) and (ii).

Now, suppose that $|c(w)| > 1$. We consider the iterated central basic factorization of type 2 of w, $\mathsf{I}_2\mathsf{CBF}(w)$. Then, by [11, Theorem 4.5] and by the analogue version of [11, Theorem 4.18] for $T_2(A)$, we have $\mathbf{o}(w) = \mathbf{o_1} + \mathbf{1} + \mathbf{o_2} + \mathbf{1} + \cdots + \cdots + \mathbf{1} + \bar{\mathbf{o}}_\mathbf{2} + \mathbf{1} + \bar{\mathbf{o}}_\mathbf{1}$, for some orderings $\mathbf{o_i}$ and $\bar{\mathbf{o}}_\mathbf{i}$. If $p \in \mathbf{o}(w)$ corresponds to any position labeled a_i or b_i of $\mathsf{I}_2\mathsf{CBF}(w)$, then $\mathbf{o}' = \mathbf{o}(f_{1^{i-1}}(w))$ satisfies the desired conditions. Otherwise, p is a position in $\mathbf{o_i}$ or $\bar{\mathbf{o}}_\mathbf{i}$, for some i. Since the content of the pseudoword represented by this ordering is strictly contained in $c(w)$, the result follows by induction. □

Given $f_\delta(w) \in \mathcal{F}(w)$, with $\delta \in \{0,1,2\}^*$, we define the *depth* of $f_\delta(w)$, $\mathfrak{d}(f_\delta(w))$, as the length of the word $\delta \in \{0,1,2\}^*$.

Lemma 3.3. *Let $w \in \overline{\Omega}_A\mathsf{DA}$. Given $f_\delta(w) \in \mathcal{F}(w)$, with $\delta \in \{0,1,2\}^*$, there exist $k \geq 0$, $f_{\delta_1}, f_{\delta_2}, \ldots, f_{\delta_k} \in \mathcal{F}(w)$ and $a_{\delta_1}, a_{\delta_2}, \ldots, a_{\delta_k} \in A$ such that $\mathbf{o}(f_{\delta_1}a_{\delta_1}f_{\delta_2}a_{\delta_2}\cdots f_{\delta_k}a_{\delta_k}f_\delta(w))$ is an initial segment of $\mathbf{o}(w)$.*

Proof. We proceed by induction on $\mathfrak{d}(f_\delta(w))$. The case where $\mathfrak{d}(f_\delta(w)) = 0$, i.e., $f_\delta(w) = f_\varepsilon(w)$, is trivial since $f_\varepsilon(w) = w$. Let $f_\delta(w) \in \mathcal{F}(w)$, with $\delta \in \{0,1,2\}^*$, be such that $\mathfrak{d}(f_\delta(w)) = 1$. Three cases can occur: $f_\delta(w) = f_0(w)$, $f_\delta(w) = f_1(w)$ or $f_\delta(w) = f_2(w)$. It follows, respectively, that $\mathbf{o}(f_0(w))$, $\mathbf{o}(f_0 l(p^{o(w)})f_1(w))$ and $\mathbf{o}(f_0 l(p^{o(w)})f_1 l(\bar{p}^{o(w)})f_2(w))$ are initial segments of $\mathbf{o}(w)$. Now, suppose that $\mathfrak{d}(f_\delta(w)) = n > 1$. Let η be the prefix of δ with length $|\delta| - 1$. By induction hypothesis, there exist $f_{\eta_1}, f_{\eta_2}, \ldots, f_{\eta_m} \in \mathcal{F}(w)$ and $a_{\eta_1}, a_{\eta_2}, \ldots, a_{\eta_m} \in A$ such that $\mathbf{o}(f_{\eta_1}a_{\eta_1}f_{\eta_2}a_{\eta_2}\cdots f_{\eta_m}a_{\eta_m}f_\eta(w))$ is an initial segment of $\mathbf{o}(w)$. By definition of $f_\delta(w) \in \mathcal{F}(w)$, it follows that $f_\delta(w)$ is a factor of $f_\eta(w)$ if and only if η is a prefix of δ. Consider the factors $f_{\eta 0}, f_{\eta 1}, f_{\eta 2} \in \mathcal{F}(w)$. Note that one of them is the factor $f_\delta(w)$. We have, respectively, $\mathbf{o}(f_{\eta_1}a_{\eta_1}f_{\eta_2}a_{\eta_2}\cdots f_{\eta_m}a_{\eta_m}f_{\eta 0}(w))$, $\mathbf{o}(f_{\eta_1}a_{\eta_1}f_{\eta_2}a_{\eta_2}\cdots f_{\eta_m}a_{\eta_m}f_{\eta 0}l(p^{o(\eta)})f_{\eta 1}(w))$ and $\mathbf{o}(f_{\eta_1}a_{\eta_1}f_{\eta_2}a_{\eta_2}\cdots f_{\eta_m}a_{\eta_m}f_{\eta 0} \cdot l(p^{o(\eta)})f_{\eta 1}l(\bar{p}^{o(\eta)})f_{\eta 2}(w))$ as initial segments of $\mathbf{o}(w)$, in the cases where $f_\delta(w) = f_{\eta 0}(w)$, $f_\delta(w) = f_{\eta 1}(w)$ and $f_\delta(w) = f_{\eta 2}(w)$, respectively. □

Lemma 3.4. *Let $w \in \overline{\Omega}_A\mathsf{DA}$. We have:*

(1) $u \in \mathcal{S}(w) \Rightarrow \exists\, p, q \in \mathbf{o}(w) : \mathbf{o}(u) \simeq [p, q]$;
(2) $p, q \in \mathbf{o}(w), p < q \Rightarrow \exists\, u \in \mathcal{S}(w) : \mathbf{o}(u) \simeq [p, q]$.

Proof. 1. By definition of $f_0(w)$, $f_1(w)$ and $f_2(w)$ and also by definition of $p^{o(w)}$ and $\bar{p}^{o(w)}$, it follows that $f_0(w) \simeq [\min o(w), p^{o(w)}[$, $f_1(w) \simeq]p^{o(w)}, \bar{p}^{o(w)}[$ and $f_2(w) \simeq]\bar{p}^{o(w)}, \max o(w)]$. Note that the predecessors and the successors of $p^{o(w)}$ and $\bar{p}^{o(w)}$ exist in any $*$-labeled linear ordering. Applying [11, Lemma 4.16] to each interval isomorphic to the elements $f_0(w)$, $f_1(w)$ and $f_2(w)$, respectively, and proceeding inductively, we deduce that all elements of $\mathcal{F}(w)$ are isomorphic to closed intervals of $\mathbf{o}(w)$. Let $u \in \mathcal{S}(w)$ and $a \in A$ be such that $\mathbf{o}(ua)$ is an initial segment of $\mathbf{o}(w)$. Then $\mathbf{o}((ua)^{-1}w)$ is a reduced $*$-labeled linear ordering, by [11, Lemma 4.13], because it is a closed interval on $\mathbf{o}(w)$. Hence there exist $p, q \in \mathbf{o}(w)$ such that $\mathbf{o}((ua)^{-1}w) \simeq [p, q]$ (in this case $q = \max o(w)$). We obtain a similar result using the condition (iv). Proceeding inductively, we conclude that all elements of $\mathcal{S}(w)$ are isomorphic to some closed interval of $\mathbf{o}(w)$.

2. Let $p, q \in \mathbf{o}(w)$ be such that $p < q$ and consider the closed interval $[p, q]$. By [11, Lemma 4.13], $[p, q]$ is a reduced $*$-linear ordering. We want to prove

that it is isomorphic to the $*$-linear ordering corresponding to an element of $\mathcal{S}(w)$. Let $p' = \text{predecessor}(p)$ and $q' = \text{successor}(q)$. Consider the interval $[p', q']$. By Lemma 3.2, there exists $f_\delta(w) \in \mathcal{F}(w)$ such that $p' = p^{o(f_\delta(w))}$ or $p' = \bar{p}^{o(f_\delta(w))}$. If $p' = p^{o(f_\delta(w))}$, we choose the factor $f_{\delta 0}$, and if $p' = \bar{p}^{o(f_\delta(w))}$, we choose the factor $f_{\delta 0} l(p^{o(f_\delta(w))}) f_{\delta 1}$. Let $f_{\delta_1}, \ldots, f_{\delta_k} \in \mathcal{F}(w)$ and $a_{\delta_1}, \ldots, a_{\delta_k} \in A$ be such that $\mathbf{o}(f_{\delta_1} a_{\delta_1} \cdots f_{\delta_k} a_{\delta_k} f_\delta(w))$ is an initial segment of $\mathbf{o}(w)$, as we had shown in Lemma 3.3. Then either $\mathbf{o}(f_{\delta_1} a_{\delta_1} \cdots f_{\delta_k} a_{\delta_k} f_{\delta 0} l(p^{o(f_\delta(w))})(w)) \simeq [\min o(w), p']$ or $\mathbf{o}(f_{\delta_1} a_{\delta_1} \cdots f_{\delta_k} a_{\delta_k} f_{\delta 0} l(p^{o(f_\delta(w))}) f_{\delta 1} l(\bar{p}^{o(f_\delta(w))})(w)) \simeq [\min o(w), p']$, depending on the case. By condition (iii) applied either $k+1$ or $k+2$ times and, depending on the case, using the factors f_{δ_i} of this initial segment, $f_{\delta 0}$ and $f_{\delta 1}$, the letters a_{δ_i}, $l(p^{o(f_\delta(w))})$ and $l(\bar{p}^{o(f_\delta(w))})$ and the pseudoword w, we obtain a pseudoword $v \in \mathcal{S}(w)$ such that $\mathbf{o}(v) \simeq]p', \max o(w)] = [p, \max o(w)]$. We proceed similarly with q' and using condition (iv) and the pseudoword v. It follows that there exists $u \in \mathcal{S}(w)$ is such that $\mathbf{o}(u) \simeq [p, q]$. □

We conclude, by Lemma 3.4, that the elements of $\mathcal{S}(w)$ correspond to the closed intervals of $\mathbf{o}(w)$. Let $u \in \mathcal{S}(w)$ and let $p, q \in \mathbf{o}(w)$ be such that $\mathbf{o}(u) \simeq [p, q]$ as we have seen in the previous lemma. Let $f_\delta(w), f_\eta(w) \in \mathcal{F}(w)$, with $\delta, \eta \in \{0, 1, 2\}^*$, satisfy the conditions of Lemma 3.2, respectively, to p and q. We call p and q the *borders* of u and $|\delta|$ and $|\eta|$ are, respectively, the *depth* of each border.

4. Characterizations of ω-terms over DA

We solve the word problem for ω-terms over DA. For this purpose, we present, in this section, several characterizations of an ω-term over DA. We start by observing that the factors involved in the central basic factorization of an ω-term over DA are also ω-terms over DA. As a tool to be used in inductive processes that follow, we define, inductively, the length of an ω-term by $|a| = 1$, with $a \in A$, $|uv| = |u| + |v|$ and $|u^\omega| = |u| + 1$.

Lemma 4.1. *Let $w \in \Omega_A^\omega \mathsf{DA} \backslash \{1\}$ and let $(\alpha, a, \gamma, b, \beta)$ (respectively, (α, a, β)) be the central basic factorization of w. Then α, γ and β (respectively, α and β) are also ω-terms over* DA.

Proof. We proceed by induction on $(c(w), |w|)$, where the pairs are ordered lexicographically. The case $w \in A$ is trivial. Suppose that $w = x^\omega$ with $x \in \Omega_A^\omega \mathsf{DA} \backslash \{1\}$ and that the factors involved in the central basic factorization of x, $\mathsf{CBF}(x) = (\alpha, a, \gamma, b, \beta)$ (respectively, $\mathsf{CBF}(x) = (\alpha, a, \beta)$ in the degenerate case), are ω-terms over DA. Then the central basic factorization of w is of one of the following forms: $(\alpha, a, \gamma b \beta w^2 \alpha a \gamma, b, \beta)$, in the standard case (note that $w = x x^{\omega-2} x = x(x^{\omega-1})^2 x = x(x^\omega)^2 x = x w^2 x$), $(\alpha a \gamma, b, \beta w^2 \alpha, a, \gamma b \beta)$, in the overlapped case, and $(\alpha, a, \beta w^2 \alpha, a, \beta)$, in the degenerate case. In any case, the factors involved are also ω-terms over DA.

Now, suppose that $w = xy$, where the factors involved in the central basic factorization of x and y are ω-terms over DA. Let $\mathsf{CBF}(x) = (\alpha_x, a_x, \gamma_x, b_x, \beta_x)$ or $\mathsf{CBF}(x) = (\alpha_x, a_x, \beta_x)$, and $\mathsf{CBF}(y) = (\alpha_y, a_y, \gamma_y, b_y, \beta_y)$ or $\mathsf{CBF}(y) = (\alpha_y, a_y, \beta_y)$, be the central basic factorizations of x and y, respectively, depending on the type of factorization. Several cases can occur:

(i) Suppose that $c(x) = c(y) = c(w)$. Then the central basic factorization of w is $(\alpha_x, a_x, \gamma_x b_x \beta_x \alpha_y a_y \gamma_y, b_y, \beta_y)$, or $(\alpha_x a_x \gamma_x, b_x, \beta_x \alpha_y a_y \gamma_y, b_y, \beta_y)$, or $(\alpha_x, a_x, \gamma_x b_x \beta_x \alpha_y,$

$a_y, \gamma_y b_y \beta_y)$, or $(\alpha_x a_x \gamma_x, b_x, \beta_x \alpha_y, a_y, \gamma_y b_y \beta_y)$, depending on whether the central basic factorizations of x and y are, respectively, both of the standard form, $\mathsf{CBF}(x)$ is of the standard form and $\mathsf{CBF}(y)$ is of the overlapped form, $\mathsf{CBF}(x)$ is of the overlapped form and the $\mathsf{CBF}(y)$ is of the standard form, or both of the factorizations are of the overlapped form. In the cases where at least one of the central basic factorizations of x and y is degenerate, we also have analogous central basic factorizations of w. In fact, in the case where $\mathsf{CBF}(x) = (\alpha_x, a_x, \beta_x)$, we have $\mathsf{CBF}(w) = (\alpha_x, a_x, \beta_x \alpha_y a_y \gamma_y, b_y, \beta_y)$, $\mathsf{CBF}(w) = (\alpha_x, a_x, \beta_x \alpha_y, a_y, \gamma_y b_y \beta_y)$ or $\mathsf{CBF}(w) = (\alpha_x, a_x, \beta_x \alpha_y, a_y, \beta_y)$, depending on whether the central basic factorization of y is standard, overlapped or degenerate. In any case, the factors involved are finite products of ω-terms and, therefore, they are ω-terms.

(ii) Now, we suppose that $c(x) \neq c(w)$ and $c(y) = c(w)$. We also suppose that the central basic factorization of y is of the standard form, $\mathsf{CBF}(y) = (\delta_{y_k}, a_{y_0}, \gamma_y, b_y, \beta_y)$, where $k = |c(y)| - 1$. Let $(\delta_{y_{(k-1)}}, a_{y_1}, \alpha_{y_1})$ be the left basic factorization of δ_{y_k}, as defined in [5]. Since $c(\delta_{y_{(k-1)}}) \subsetneqq c(\delta_{y_k}) \subsetneqq c(y)$, we repeat the process a finite number of times until we obtain the factorization $y = \delta_{y_0} a_{y_k} \cdots a_{y_1} \alpha_{y_1} a_{y_0} \gamma_y b_y \beta_y$. Remember that the factors involved in this factorization are also ω-terms, by [5, Lemma 2.2] and by induction hypothesis. Let i be maximum such that $c(w) = c(x \cdot \delta_{y_0} a_{y_k} \cdots a_{y_i} \alpha_{y_i} a_{y_{(i-1)}})$. Then we have $\mathsf{CBF}(w) = (x \cdot \delta_{y_0} a_{y_k} \cdots a_{y_i} \alpha_{y_i}, a_{y_{(i-1)}}, \alpha_{y_{(i-1)}} \cdots a_{y_0} \gamma_y, b_y, \beta_y)$, where all the factors involved are ω-terms. In the case where the central basic factorization of y is degenerate, we use the same argument and we obtain $\mathsf{CBF}(w) = (x \cdot \delta_{y_0} a_{y_k} \cdots a_{y_i} \alpha_{y_i}, a_{y_{(i-1)}}, \alpha_{y_{(i-1)}} \cdots a_{y_1} \alpha_{y_1}, a_{y_0}, \beta_y)$. Let us see the case where the central basic factorization of y is of the overlapped form, $\mathsf{CBF}(y) = (\alpha_y, a_y, \gamma_y, b_y, \beta_y)$. If $c(x\alpha_y) = c(w)$ then, by a similar argument to the one used in the previous case, we obtain $\mathsf{CBF}(w) = (x \cdot \delta_{y_0} a_{y_k} \cdots a_{y_i} \alpha_{y_i}, a_{y_{(i-1)}}, \alpha_{y_{(i-1)}} \cdots \alpha_{y_1}, a_y, \gamma_y b_y \beta_y)$. If $c(x\alpha_y) \neq c(w)$ and $c(x\alpha_y a_y) = c(w)$, then $\mathsf{CBF}(w) = (x\alpha_y, a_y, \gamma_y b_y \beta_y)$. In the case where $c(x\alpha_y a_y) \neq c(w)$, we use a similar argument for γ_y and we obtain $\mathsf{CBF}(w) = (x\alpha_y, a_y, \delta_{y_0} b_{y_k} \cdots b_{y_i} \gamma_{y_i}, b_{y_{(i-1)}}, \gamma_{y_{(i-1)}} \cdots \gamma_{y_1} b_y \beta_y)$. We obtain the dual result for the case where $c(y) \neq c(w)$ and $c(x) = c(w)$.

(iii) Finally, we can verify the case where $c(x) \neq c(w)$ and $c(y) \neq c(w)$ using, again, an argument similar to that given for (ii). □

We say that an ω-term is *reduced* if it has no subterm of the form $r^\omega s t^\omega$, with $c(s) \subseteq c(r) = c(t)$, and no subterm of the form $(rs^\omega z^\omega t)^\omega$, with r and t pseudowords which may be empty and with $c(t) \cup c(r) \subseteq c(s) = c(z)$. Recall that, in a pro-$\mathsf{DA}$ monoid, $r^\omega s t^\omega = r^\omega t^\omega$, if $c(s) \subseteq c(r) = c(t)$ (see [1, Lemma 8.1.4 and Theorem 8.1.7]).

Lemma 4.2. *Let w be an ω-term which defines an idempotent in $\overline{\Omega}_A\mathsf{DA}$. Then we have one of the following conditions:*

(i) *There exist ω-terms x, y, z, t such that $\mathsf{DA} \models w = xy^\omega z^\omega t$, $c(y) = c(z) = c(w)$, $|x| + |y| + |z| + |t| < |w|$ and x and t satisfy one of the following conditions: they do not define idempotents over DA or $c(s) \subsetneqq c(w)$ for both $s = x$ and $s = t$;*

(ii) *There exist ω-terms x, y, z such that $\mathsf{DA} \models w = xy^\omega z$, $c(y) = c(w)$, $|x| + |y| + |z| < |w|$ and x and z satisfy one of the following conditions: they do not define idempotents over DA or $c(s) \subsetneqq c(w)$ for both $s = x$ and $s = z$.*

We also have that $xy^\omega z^\omega t$ (respectively, $xy^\omega z$) is reduced.

Proof. We begin by noting that the substitutions $r^\omega s t^\omega \to r^\omega t^\omega$, if $c(s) \subseteq c(r) = c(t)$, and $(rs^\omega y z^\omega t)^\omega \to rs^\omega z^\omega t$, if $c(ryt) \subseteq c(s) = c(z)$, do not change the value of an ω-term over DA. Moreover, the length of the terms decrease when we apply these substitutions. Let v be a reduced ω-term obtained from w by applying these substitutions. Since w is idempotent over DA, v is also idempotent. Moreover, $|v| \leq |w|$. We write $v = x_1 \cdots x_r$, where each x_i is a letter or a term of the form y_i^ω. By [11, Corollary 3.17], there exists x_i such that $c(x_i) = c(v)$ and $x_i = y_i^\omega$. Suppose that there exists another factor x_j with $c(x_j) = c(v)$ and $x_j = y_j^\omega$, for some y_j. Considering the fact that v is reduced, the factors x_i and x_j must be consecutive and, therefore, $v = xy^\omega z^\omega t$, with x and t not satisfying one of the conditions $c(s) = c(v) = c(w)$ or $s = y_i^\omega$, with $s = x$ or $s = t$. Thus, either x is not an idempotent, or x is an idempotent and $c(x) \subsetneq c(v)$, and similarly for t. Now, suppose that no other x_j is such that $c(x_j) = c(v)$ and $x_j = y_j^\omega$. Then $v = x_1 \cdots x_{i-1} x_i x_{i+1} \cdots x_r = xy^\omega z$ for some x, y, z, where x and z are not idempotents or, if any of them is idempotent, then it has strictly smaller content than v. We also have $|x|+|y|+|z|+|t| < |w|$ in the first case, and $|x|+|y|+|z| < |w|$ in the second case. □

We are now ready to present some characterizations of the ω-terms over DA. The following is a sort of periodicity theorem for DA.

Theorem 4.3. *Let $w \in \overline{\Omega}_A\mathsf{DA}$. The following conditions are equivalent:*

(a) $\mathcal{L}(w)$ *is rational.*
(b) $\mathcal{A}(w)$ *is finite.*
(c) *The set $\{\rho(t(w)_v) \mid v \in V\}$ is finite, where $t(w) = \langle V, \to, q, F, \lambda\rangle$.*
(d) $\mathcal{F}(w)$ *is finite.*
(e) $\mathcal{R}(w)$ *is finite.*
(f) $\mathcal{S}(w)$ *is finite.*
(g) $w \in \Omega_A^\omega\mathsf{DA}$.

Proof. $(a) \Leftrightarrow (b)$: Given $\mathcal{A}(w)$, which is finite, we construct a finite automaton recognizing $\mathcal{L}(w)$, by replacing the label of each edge in $\mathcal{A}(w)$ by the pair whose first component is the label that the edge has in $\mathcal{A}(w)$ and the second component is the label of the initial vertex of the edge in $\mathcal{A}(w)$. For the direct implication we do the converse: given the minimal automaton that recognizes the language $\mathcal{L}(w)$ (and it is unique by [11, Lemma 4.10]), we construct the automaton $\mathcal{A}(w)$ whose states are labeled with the second component of the label of the edges that starts from that state.

$(b) \Leftrightarrow (c)$: Note that, by definition, there exists a bijection between the set of states in $\mathcal{A}(w)$ and the pseudowords $\rho(t(w)_v)$, with $v \in V$. Hence, the result follows.

$(c) \Leftrightarrow (d)$: Applying [11, Lemma 4.9] to $t(w)$, we have that the set of vertices, $\{\rho(t(w)_v) \mid v \in V\}$, is in bijection with $\mathcal{F}(w)$.

$(d) \Rightarrow (e)$: It is obvious, because $\mathcal{R}(w) \subseteq \mathcal{F}(w)$.

$(e) \Rightarrow (f)$: Suppose that $\mathcal{R}(w)$ is finite. To show that $\mathcal{S}(w)$ is also finite, we proceed by induction on $|A|$, where the case $|A| = 0$ is trivial. Now, suppose that $|A| \geq 1$. Let $\mathcal{S}_n(w) = \{u \in \mathcal{S}(w) \mid$ the borders of u have depth not exceeding $n\}$.

Then, we have

$$\begin{array}{rcl}\mathcal{S}_{n+1}(\mathcal{R}(w)) & \subseteq & \mathcal{S}_n[f_0(\mathcal{R}(w))]\cdot A\cdot f_1(\mathcal{R}(w))\cdot A\cdot \mathcal{S}_n[f_2(\mathcal{R}(w))]\\ & & \cup \mathcal{S}_n[f_0(\mathcal{R}(w))]\cdot A\cdot \mathcal{S}_n[f_1(\mathcal{R}(w))]\\ & & \cup \mathcal{S}_n[f_1(\mathcal{R}(w))]\cdot A\cdot \mathcal{S}_n[f_2(\mathcal{R}(w))]\\ & & \cup \mathcal{S}_n[f_0(\mathcal{R}(w))]\cup \mathcal{S}_n[f_1(\mathcal{R}(w))]\cup \mathcal{S}_n[f_2(\mathcal{R}(w))]\\ & \subseteq & \mathcal{S}[f_0(\mathcal{R}(w))]\cdot A\cdot \mathcal{R}(w)\cdot A\cdot \mathcal{S}[f_2(\mathcal{R}(w))]\\ & & \cup \mathcal{S}[f_0(\mathcal{R}(w))]\cdot A\cdot \mathcal{S}_n(\mathcal{R}(w))\\ & & \cup \mathcal{S}_n(\mathcal{R}(w))\cdot A\cdot \mathcal{S}[f_2(\mathcal{R}(w))]\\ & & \cup \mathcal{S}[f_0(\mathcal{R}(w))]\cup \mathcal{S}_n(\mathcal{R}(w))\cup \mathcal{S}[f_2(\mathcal{R}(w))].\end{array}$$

By induction on n and by definition of $\mathcal{S}(w)$, we obtain

$$\begin{array}{rcl}\mathcal{S}_{n+1}(\mathcal{R}(w)) & \subseteq & \bigcup_{i=0}^{n}(\mathcal{S}[f_0(\mathcal{R}(w))]\cdot A)^i\cdot\\ & & (\mathcal{S}[f_0(\mathcal{R}(w))]\cdot A\cdot \mathcal{R}(w)\cdot A\cdot \mathcal{S}[f_2(\mathcal{R}(w))]\cup \mathcal{S}[f_0(\mathcal{R}(w))]\cdot A\cup \mathcal{R}(w))\\ & & \bigcup (\mathcal{S}[f_0(\mathcal{R}(w))]\cdot A\cdot \mathcal{R}(w)\cdot A\cdot \mathcal{S}[f_2(\mathcal{R}(w))]\cup A\cdot \mathcal{S}[f_2(\mathcal{R}(w))]\cup \mathcal{R}(w))\cdot\\ & & \bigcup_{i=0}^{n}(A\cdot \mathcal{S}[f_2(\mathcal{R}(w))])^i\\ & & \bigcup \mathcal{S}[f_0(\mathcal{R}(w))]\cup \mathcal{R}(w)\cup \mathcal{S}[f_2(\mathcal{R}(w))]\end{array}$$

for all n and, therefore, $\mathcal{S}(\mathcal{R}(w))$ is contained in the union of these sets. We also have

$$\begin{array}{rcl}\mathcal{S}(w) & \subseteq & \{w\}\cup \mathcal{S}(f_0(w))\cdot A\cdot \mathcal{R}(w)\cdot A\cdot \mathcal{S}(f_2(w))\\ & & \cup \mathcal{S}(f_0(w))\cdot A\cdot \mathcal{S}(\mathcal{R}(w))\\ & & \cup \mathcal{S}(\mathcal{R}(w))\cdot A\cdot \mathcal{S}(f_2(w))\\ & & \cup \mathcal{S}(f_0(w))\cup \mathcal{S}(\mathcal{R}(w))\cup \mathcal{S}(f_2(w)).\end{array}$$

Considering the last two inclusions, it is enough to show that the following sets are finite: $\mathcal{S}(f_0(w))$, $\mathcal{S}(f_2(w))$, $\mathcal{S}[f_0(\mathcal{R}(w))]$ and $\mathcal{S}[f_2(\mathcal{R}(w))]$. Let $u\in\{f_0(w),\ f_2(w)\}\cup f_0(\mathcal{R}(w))\cup f_2(\mathcal{R}(w))$. Since $c(f_0(v)), c(f_2(v))\subsetneq c(v)$, for all $v\neq 1$, it follows that $c(u)\subsetneq c(w)$. Moreover, since

$$\mathcal{R}(\mathcal{F}(w)) = f_1[\mathcal{F}(\mathcal{F}(w))] = f_1(\mathcal{F}(w)) = \mathcal{R}(w),$$

we have, in particular, $\mathcal{R}(u)\subseteq\mathcal{R}(w)$ and, therefore, $\mathcal{R}(u)$ is finite. Applying the induction hypothesis to u, which has a smaller content, we conclude that $\mathcal{S}(u)$ is finite. Hence $\mathcal{S}(w)$ is finite.

$(f)\Rightarrow(g)$: Let $\mathcal{S}(w)$ be finite. We proceed by induction on $|c(w)|$ to show that w is an ω-term. If $c(w)=\{a\}$, then $w=a^n$, with n finite, or $w=a^\omega$ and, therefore, it is an ω-term. Now, suppose that $|c(w)|\geq 1$. Let $w=\overrightarrow{\prod}_{i=0}^{\llbracket w\rrbracket-1}(\alpha_i a_i)\cdot\overleftarrow{\prod}_{i=0}^{\llbracket w\rrbracket-1}(b_i\beta_i)$ be the iterated central basic factorization of w. Suppose that $\llbracket w\rrbracket$ is finite. Recall that $\llbracket w\rrbracket$ is the number of iterations until we obtain the iterated central basic factorization of w. Note that $\mathcal{S}(\alpha_i)$, $\mathcal{S}(\beta_i)\subseteq\mathcal{S}(w)$, for all i, because $\alpha_i,\beta_i\subseteq\mathcal{S}(w)$. Since, by induction hypothesis, $\mathcal{S}(w)$ is finite, then $\mathcal{S}(\alpha_i)$ and $\mathcal{S}(\beta_i)$ are also finite, for all i. Moreover, $c(\alpha_i)$, $c(\beta_i)\subsetneq c(w)$, for all i. It follows, by induction hypothesis, that α_i and β_i are ω-terms, for all i. Hence, w is an ω-term.

Now, suppose that $\llbracket w\rrbracket$ is infinite. Let $u_{l,k}=\overrightarrow{\prod}_{i=l}^{l+k-1}(\alpha_i a_i)$, $v_{l,k}=\overleftarrow{\prod}_{i=l}^{l+k-1}(b_i\beta_i)$ and $w_{l,k}=\overrightarrow{\prod}_{i\geq l}(\alpha_i a_i)\cdot\overleftarrow{\prod}_{i\geq k}(b_i\beta_i)$, with $k>l\geq 0$. We have $w=u_{0,i}\cdot\alpha_i a_i\cdot w_{i+1,i+1}\cdot b_i\beta_i\cdot v_{0,i}$. By definition, $w_{i,i}=f_1{}^i(w)\in\mathcal{S}(w)$. Let N be an integer satisfying the condition of [11, Lemma 3.10], i.e., if $i,j,k\geq N$, then $c(w_{i,i})=c(w_{j,j})$ and $c(\alpha_k a_k)=c(b_k\beta_k)$. Since $\mathcal{S}(w)$ is finite, there exist $l\geq N$ and $k>0$ such that $w_{l+k,l+k}=w_{l,l}=u_{l,l+k}\cdot w_{l+k,l+k}\cdot v_{l,l+k}$ and, therefore, $w_{l,l}=u_{l,l+k}^\omega\cdot w_{l,l}\cdot v_{l,l+k}^\omega$. Since $c(w_{l,l})\subseteq c(u_{l,l+k})=c(v_{l,l+k})$, we have, by [11, Corollary 3.7], $w_{l,l}=u_{l,l+k}^\omega\cdot v_{l,l+k}^\omega$. Hence $w=u_{0,l}w_{l,l}v_{0,l}=u_{0,l}u_{l,l+k}^\omega v_{l,l+k}^\omega v_{0,l}$ which is an ω-term.

$(g) \Rightarrow (d)$: Let $w \in \Omega_A^\omega \mathsf{DA}$. We proceed by induction on $(|c(w)|, |w|)$ where the pairs are ordered lexicographically.

If $c(w) = a$, then or $w = a^n$ is a word and $\mathcal{F}(w) = \{1, a^2, a^4, \ldots, a^n\}$ or $\mathcal{F}(w) = \{1, a, a^3, \ldots, a^n\}$, depending on whether n is even or odd, or $w = a^\omega$ and we have $\mathcal{F}(w) = \{1, a^\omega\}$. In any case, $\mathcal{F}(w)$ is finite.

If $|c(w)| > 1$, we start by showing that the set $f_{1^*}(w)$ is finite. Let $w = \overrightarrow{\prod}_{i=0}^{\lVert w \rVert - 1}(\alpha_i a_i) \cdot \overleftarrow{\prod}_{i=0}^{\lVert w \rVert - 1}(b_i \beta_i)$ be the iterated central basic factorization of w. If $\|w\|$ is finite, where $\|w\|$ denotes the largest integer n such that $c(\alpha_n a_n) = c(b_n \beta_n) = c(w)$ with $\alpha_n a_n$ and $b_n \beta_n$ disjoint (notation introduced in [11]), then we can write $w = \alpha_0 a_0 \cdots \alpha_k a_k \gamma_k b_k \beta_k \cdots b_0 \beta_0$, with $a_i, b_i \in A$, $c(\alpha_i), c(\beta_i) \subsetneq c(w)$, for all i, and $c(\gamma_k) \subsetneq c(w)$. By Lemma 4.1, these factors are also ω-terms. Since $c(\gamma_k) \subsetneq c(w)$, it follows, by induction on $|c(w)|$, that $f_{1^*}(\gamma_k)$ is finite. Since $f_{1^*}(w) = f_{1^*}(\gamma_k) \cup \{\alpha_i a_i \cdots \gamma_k \cdots b_i \beta_i \mid i \leq k\}$, it follows that $f_{1^*}(w)$ is finite.

If $\|w\|$ is infinite, then, by [11, Proposition 3.15], w is idempotent. By Lemma 4.2, we can write w in one of the following forms: $w = xy^\omega z$, with $|x| + |y| + |z| < |w|$, or $w = xy^\omega z^\omega t$, with $|x| + |y| + |z| + |t| < |w|$. Suppose that we have the first case. Since $c(y) \subseteq c(w)$ and $|y| < |w|$, by induction hypothesis applied to y, $\mathcal{F}(y)$ is finite. Since $(d) \Rightarrow (f)$, it follows that $\mathcal{S}(y)$ is also finite. Similarly, the sets $\mathcal{S}(x)$ and $\mathcal{S}(z)$ are finite. Hence we have

$$f_{1^*}(w) = f_{1^*}(xy^\omega z) \subseteq \mathcal{S}(x)y^\omega \mathcal{S}(z) \cup \mathcal{S}(x)y^\omega \mathcal{S}(y) \cup \mathcal{S}(y)y^\omega \mathcal{S}(z) \cup \mathcal{S}(y)y^\omega \mathcal{S}(y).$$

It follows that $f_{1^*}(w)$ is finite. The second case is similar.

Let $l \geq N$ and $k > 0$ be such that $f_{1^{l+k}}(w) = f_{1^l}(w)$, where N satisfies the condition of [11, Lemma 3.10]. Then the following equalities are satisfied by DA:

$$\begin{aligned} f_{1^l}(w) &= \alpha_l a_l \cdots \alpha_{l+k-1} a_{l+k-1} f_{1^{l+k}}(w) b_{l+k-1} \beta_{l+k-1} \cdots b_l \beta_l \\ &= \alpha_l a_l \cdots \alpha_{l+k-1} a_{l+k-1} f_{1^l}(w) b_{l+k-1} \beta_{l+k-1} \cdots b_l \beta_l \\ &= (\alpha_l a_l \cdots \alpha_{l+k-1} a_{l+k-1})^\omega f_{1^l}(w) (b_{l+k-1} \beta_{l+k-1} \cdots b_l \beta_l)^\omega \\ &= (\alpha_l a_l \cdots \alpha_{l+k-1} a_{l+k-1})^\omega (b_{l+k-1} \beta_{l+k-1} \cdots b_l \beta_l)^\omega \end{aligned}$$

where the last equality follows from [11, Corollary 3.7]. It follows that

$$w = \alpha_0 a_0 \cdots \alpha_{l-1} a_{l-1} (\alpha_l a_l \cdots \alpha_{l+k-1} a_{l+k-1})^\omega (b_{l+k-1} \beta_{l+k-1} \cdots b_l \beta_l)^\omega b_{l-1} \beta_{l-1} \cdots b_0 \beta_0.$$

Note that $f_{1^*0}(w) \subseteq W_0 = \{\alpha_0, \ldots, \alpha_{l+k-1}\}$ and $f_{1^*2}(w) \subseteq W_2 = \{\beta_0, \ldots, \beta_{l+k-1}\}$. Hence we have $\mathcal{F}(w) = f_{1^*}(w) \cup \mathcal{F}(f_{1^*0}(w)) \cup \mathcal{F}(f_{1^*2}(w)) \subseteq f_{1^*}(w) \cup \mathcal{F}(W_0) \cup \mathcal{F}(W_2)$. Since W_0 and W_2 are finite sets of ω-terms on a smaller alphabet than $c(w)$, we have, by induction hypothesis, that $\mathcal{F}(W_0)$ and $\mathcal{F}(W_2)$ are finite. The implication $(g) \Rightarrow (d)$ follows and this completes the proof of the theorem. □

5. An algorithm to compute the minimal DA-automaton

Given two ω-terms on an alphabet A, we wish to show that it is possible to decide if they coincide over all elements of DA. By Theorem 4.3, we know that, if w is an ω-term over DA, then the wrapped DA-automaton (which is minimal) that represents w is finite. Moreover, by Proposition 4.11, Lemma 4.10, and Corollary 4.8 from [11], and by definition of minimal DA-automaton, two ω-terms coincide over DA if and only if their wrapped DA-automata are isomorphic.

In this section, the aim is to construct the minimal DA-automaton of an ω-term. For that purpose, we present an algorithm which constructs a finite DA-automaton of an ω-term and, using existing tools, this automaton may be efficiently minimized.

5.1. **The main function.** Let w be an ω-term and let $\bar{w} = \mathbf{word}(w)$ be a well-parenthesized word on the alphabet $A \cup \{(,)\}$, which results from replacing the ω-powers of w by a pair of parentheses. In the automaton that we want to construct, each state represents a word $\bar{u} = \mathbf{word}(u)$ that corresponds to an ω-term u which defines a DA-factor of the ω-word defined by the initial ω-term w. The automaton has as initial state the vertex corresponding to the ω-term w. For each state $\bar{u}$, the *sons* of $\bar{u}$, which are the terminal states from an edge whose initial state corresponds to $\bar{u}$, represent the words which define the factors of the central basic factorization of u.

For better understanding the algorithm, we present the programming of some routines. The complete programming of the algorithm in Python may be found in http://cmup.fc.up.pt/cmup/amoura/DAautomaton_complete.py.

The main routine, called DAautomaton and described in Algorithm 1, constructs the automaton $\mathcal{A} = (V, E, \iota, e, \nu)$ by a recursive process. Initially, the automaton is presented as follows: the set of states, V, consists of the initial state ι, which corresponds to the word $\bar{w}$, and of the final state e, which corresponds to the word ε, the set of transitions is empty and the labeling relation has only the pair (e, ε).

```
1  def DAautomaton(input):
2      e = ''
3      iota = input
4      V = [e,input]
5      E = []
6      nu = [[e,'epsilon']]
7      V0 = []
8      V1 = [input]
9      while V1 != []:
10         for w in V1:
11             ll = LeftLabel(w)
12             rl = RightLabel(w)
13             F = Factorization(w,ll,rl)
14             nu = nu+[F[0]]
15             desc = F[1]
16             if len(desc) == 3:
17                 for j in range(3):
18                     E += [[w,j,desc[j]]]
19                     if desc[j] not in V:
20                         V += [desc[j]]
21                         V0 += [desc[j]]
22             else:
23                 E += [[w,0,desc[0]]]
24                 E += [[w,2,desc[1]]]
25                 for i in range(2):
26                     if desc[i] not in V:
27                         V += [desc[i]]
28                         V0 += [desc[i]]
29         V1 = V0
30         V0 = []
31     A = [V,E,iota,e,nu]
32     return A
```

ALGORITHM 1

Let V_0 and V_1 be, respectively, the set of states which were not yet processed and the set of states which will be processed in the following step (which corresponds

to run the **while** cycle once). Initially, V_0 is the empty set and V_1 consists of the initial state. The algorithm stops when these sets are both empty.

The process consists in the computation that we proceed to describe. Given a state of V_1, which corresponds to an ω-term u, we calculate the positions ll and rl in $\bar{u}$ of the labels of the central basic factorization of u. For that, we use two functions called LeftLabel and RightLabel, respectively.

We apply to this ω-term u and its label-positions the function Factorization, that it is described in detail in 5.2. The function computes the label of the state and keeps it in the list ν. It also produces the sons of this state. Then the main routine tests if each one of the sons is already in V. If it is not, it is added to V and to V_0 to be processed later. A transition is created that goes from the state that we are processing to the state corresponding to each son and labeled by the order of such son (i.e., 0, 1 or 2).

When $V_0 = \emptyset = V_1$, the routine stops. This means that all the elements have already been processed and all the states corresponding to DA-factors of the initial ω-term are already in the set of states of the automaton. Hence the DA-automaton, that we denote by $\mathcal{G}(w)$, is constructed.

5.2. **The factorization of an ω-term.** It is the function Factorization, described in Algorithm 2, that analyzes a state corresponding to a DA-factor of the initial ω-term. It takes as input the word that corresponds to the state that we are processing, $\bar{u}$, and the positions ll and rl, corresponding to the left label and to the right label of the central basic factorization of this DA-factor. It uses the function Parenthesis to compute the image of the partial function which associates to each position in $\bar{u}$ whose letter is a parenthesis the position corresponding to its matching pair. So that this information will be easily found, the function Parenthesis creates a list of length equal to $|\bar{u}|$ and puts the value -1 on the entries corresponding to the positions of $\bar{u}$ whose letter belongs to A.

```
1  def Factorization(w,ll,rl):
2      m = -1
3      P = Parenthesis(w)
4      for i in range(len(P)):
5          if i < ll < P[i] and i < rl < P[i]:
6              m = i
7              break
8      if ll < rl or m != -1:
9          nu = [w,w[ll]+w[rl]]
10         desc = [S0forget(w,P,ll),S1remind(w,P,ll,rl,m),S2forget(w,P,rl
               )]
11     elif ll > rl:
12         nu = [w,w[rl]+w[ll]]
13         desc = [S0remind(w,P,rl),S1forget(w,P,rl,ll),S2remind(w,P,ll)]
14     else:
15         nu = [w,w[ll]]
16         desc = [S0forget(w,P,ll),S2forget(w,P,rl)]
17     return [nu,desc]
```

Algorithm 2

The function Factorization verifies if the labels LeftLabel and RightLabel are inside a same ω-power and keeps the information, in a variable m, of the position where the largest ω-power that contains these labels begins. Then, it compares the

values ll and rl, corresponding to the positions of the labels in the word $\bar{u}$. With this data, it determines the type of the central basic factorization. We have the following cases: if $ll < rl$ or $m \neq -1$, then the central basic factorization is of the standard form; if $ll > rl$ and $m = -1$, then the central basic factorization is of the overlapped form; if $ll = rl$ and $m = -1$, then the central basic factorization is degenerate. In the first case, we use the functions S0forget, S1remind and S2forget to construct the sons, while in the second and third cases we use, respectively, the functions S0remind, S1forget and S2remind, and the functions S0forget and S2forget. These functions are presented in the next subsection.

5.3. **The computation of the sons of an ω-term.** We present the functions that compute the sons of any state of the automaton. The functions consist on the construction of words from the word corresponding to the state that is being processed.

The functions whose name includes the word *forget* consider the subword of the initial word ending at $ll - 1$, between $ll + 1$ and $rl - 1$, or starting at $rl + 1$, depending on whether we are computing the son of the transition 0, 1 or 2, respectively, and consisting of all letters in A and all the matching parentheses in the considered interval. We show, for example, the function S0forget in Algorithm 3, which constructs the son of w from the transition labeled by 0.

```
def S0forget(w,P,ll):
    w0 = ''
    for i in range(ll):
        if w[i] != '(' or P[i] < ll:
            w0 += w[i]
    return w0
```

Algorithm 3

The functions whose name includes the word *remind* construct a word from the initial word considering all the ω-powers where the labels are inserted. We describe in detail the most intricate one, the function S1remind, presented in Algorithm 4.

```
def S1remind(w,P,ll,rl,m):
    w1 = ''
    if m == -1:
        for i in range(ll+1,rl):
            if (w[i] != '(' and w[i] != ')') or \
               (w[i] == ')' and P[i] > ll) or \
               (w[i] == '(' and P[i] < rl):
                w1 += w[i]
            elif w[i] == ')' and P[i] < ll:
                for l in range(P[i],i+1):
                    w1 += w[l]
            else:
                for l in range (i,P[i]+1):
                    w1 += w[l]
    else:
        M = P[m]
        for i in range(ll+1,M):
            if w[i] != ')' or P[i] > ll:
                w1 += w[i]
            else:
                for l in range(P[i],i+1):
```

```
                    w1 += w[l]
        for i in range(m,M+1):
            w1 += w[i]
        for i in range(m+1,rl):
            if w[i] != '(' or P[i] < rl:
                w1 += w[i]
            else:
                for l in range(i,P[i]+1):
                    w1 += w[l]
    return w1
```

ALGORITHM 4

Firstly, the routine verifies the value of the parameter m. If it is different from -1, it means that the labels are in a same ω-power and the value of m is the position where the largest ω-power containing both labels begins. The son consists of the concatenation of the suffix of this ω-term beginning in the left label, with the respective ω-term and the prefix of it ending in the right label. Moreover, all the ω-powers containing one of the labels are concatenated as they are read. If the parameter m is equal to -1, meaning that the labels are not in a same ω-power, the routine constructs the son just reading the word from left to right and concatenating all the ω-powers containing one of the labels.

We finish with an example of a DA-automaton $\mathcal{G}(w)$ constructed by the described algorithm:

Example 5.1. Let $w = (ab^\omega caa^\omega)^\omega$ and $\bar{w} = (a(b)ca(a))$. We have LeftLabel $= c$ in the position $ll = 5$ and RightLabel $= b$ in the position $rl = 3$. As these labels are in the same ω-power, corresponding to the interval $[0, 10]$, it follows that the first occurrence of c appears before the last occurrence of b, when we read from left to right. Thus the central basic factorization of w is standard. The sons are calculated with the functions S0forget, S1remind and S2forget, respectively, and correspond to the following words: $\bar{w}_0 = a(b)$, $\bar{w}_1 = a(a)(a(b)ca(a))a(b)$ and $\bar{w}_2 = ca(a)$. The other states are constructed recursively. The DA-automaton $\mathcal{G}(w)$ associated to the ω-term $w = (ab^\omega caa^\omega)^\omega$ is described in Figure 1.

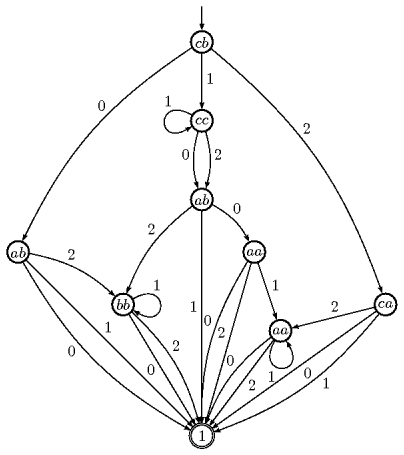

FIGURE 1. The DA-automaton $\mathcal{G}(w)$ associated to the ω-term $w = (ab^\omega caa^\omega)^\omega$.

5.4. **The complexity of the algorithm.** As explained in the previous subsections, the algorithm constructs, in each step, the factors of the central basic factorization of the ω-term that we are considering. However, nothing so far guarantees that the algorithm stops and, consequently, that the automaton $\mathcal{G}(w)$ is finite. This is what we propose to prove in this subsection together with the study of the complexity of the algorithm.

Let $\bar{w} = \mathbf{word}(w)$ be the input and let $|\bar{w}| = n$. For $l < n$, k_l is the number of pairs of parentheses containing the position l, $K = \max_{l<n} k_l$, $\mathfrak{l}_{(i,j)}$ is the length of the subword bounded by the pair of parentheses (i, j), with $0 \leq i, j < n$, i.e., the length of the subword corresponding to the ω-power (i, j), Φ_l is the sum of the lengths of the subwords corresponding to the ω-powers containing the position l and $\Phi = \max_{l<n} \Phi_l$.

Lemma 5.2. *Let $w \in T^{\omega}_A$. The length of an ω-term representing a* DA*-factor of the ω-word $\iota(w)$ is bounded above by $n + 2\Phi$.*

Proof. We give an upper bound for the length of the words corresponding to each vertex of $\mathcal{G}(w)$, using the parameters defined above.

We start by observing that the functions whose name includes *forget* create a word with length strictly smaller than the length of the input given to that function. So, it is enough to verify what happens when we apply to a word a function whose name includes *remind*. Consider the function S0remind and suppose that $\bar{w}_0 = \text{S0remind}(\bar{w})$. Note that, in this case, the central basic factorization of w is overlapped. Let rl be the position of the right label. Then we have $|\bar{w}_0| = rl + \sum_{i<rl<j}(\mathfrak{l}_{(i,j)} - 1) = rl + \Phi_{rl} - k_{rl} < n + \Phi$, because we insert in the prefix of the word $\bar{w}$ ending in rl the subwords corresponding to the ω-powers containing rl. Similarly, for $\bar{w}_2 = \text{S2remind}(\bar{w})$, we have $|\bar{w}_2| = ll + \Phi_{ll} - k_{ll} < n + \Phi$. If the central basic factorization of w is standard, we have $\bar{w}_1 = \text{S1remind}(\bar{w})$. It follows that $|\bar{w}_1| \leq (rl - ll - 1) + \sum_{i<ll<j}(\mathfrak{l}_{(i,j)} - 1) + \sum_{i<rl<j}(\mathfrak{l}_{(i,j)} - 1) = (rl - ll - 1) + (\Phi_{ll} - k_{ll}) + (\Phi_{rl} - k_{rl}) < n + 2\Phi$.

In the following iterations, we have the same procedure. When we cut the word to create the three sons, the functions *remind* add the subwords corresponding to the ω-powers containing the position where we cut. Note that, when this cut is done in a factor which had been added previously to the subword that issued from $\bar{w}$, the number of pairs of parentheses containing this position decreases and we have just those corresponding to the ω-powers which had not been added (when we read from the center to the borders). It follows that, in any depth that we are working, $|\bar{u}| < n + 2\Phi$, where $\bar{u}$ is a word corresponding to a state of the automaton. □

We note that, in the above proof, we could use the number $(2K + 1)|\bar{w}|$ as an upper bound of $|\bar{u}|$. However, the upper bound that we have considered is smaller and easily computable. As the length of a word corresponding to a state of the automaton is bounded above and A is a finite alphabet, it follows that V, the set of states of the automaton, is finite. Hence $\mathcal{G}(w)$ is finite.

Corollary 5.3. *The automaton $\mathcal{G}(w)$ produced by the algorithm is finite.*

Although the previous lemma tells us that the number of states of $\mathcal{G}(w)$ is finite, we need to find a smaller upper bound for this number so we can show that the complexity of this construction is polynomial.

We consider the following sets:

$$Q(\bar{w}) = \{(i,j,p_i,p_j) \mid -1 \leq i,j \leq |\bar{w}|,\ \lambda(i),\lambda(j) \notin \{(,)\},\ 0 \leq p_i \leq k_i,\ 0 \leq p_j \leq k_j\}$$

and

$$T(\bar{w}) = \{\bar{w}_{(i,j,p_i,p_j)} \mid (i,j,p_i,p_j) \in Q(\bar{w})\}$$

where k_i and k_j are the numbers of pairs of parentheses containing the positions i and j, respectively, and $\bar{w}_{(i,j,p_i,p_j)}$ is the word obtained from $\bar{w}$ beginning at the position $i+1$, ending at the position $j-1$, reading, from left to right, the first p_i ω-powers containing i and reading, from right to left, the first p_j ω-powers containing the position j. We also use these parentheses as *bridges* to go from a higher position to a lower position (or the dual, when we read from right to left) and this is done at the largest ω-power containing both positions and that is read in any of the ways. If there is no ω-power to be read, then we go from a higher to a lower position by the smaller ω-power containing both positions i and j. The following example should help to understand this definition.

Example 5.4. Let $\bar{w} = a(b(cb)ab)a$. We have, for example, the following elements of $T(\bar{w})$:

$$\begin{aligned}
\bar{w}_{(-1,7,0,0)} &= ab(cb)\\
\bar{w}_{(-1,7,0,1)} &= a(b(cb)ab)b(cb)\\
\bar{w}_{(-1,5,0,1)} &= ab(cb)c\\
\bar{w}_{(-1,5,0,2)} &= a(b(cb)ab)b(cb)c\\
\bar{w}_{(5,4,0,0)} &= \varepsilon\\
\bar{w}_{(5,4,1,0)} &= (cb)\\
\bar{w}_{(5,4,1,2)} &= (cb)ab(b(cb)ab)b(cb)\\
\bar{w}_{(5,4,2,2)} &= (cb)ab(b(cb)ab)(b(cb)ab)b(cb).
\end{aligned}$$

Let $\Lambda_{\bar{w}} : Q(\bar{w}) \to T(\bar{w})$ be the function that maps each tuple $(i,j,p_i,p_j) \in Q(\bar{w})$ to the word $\bar{w}_{(i,j,p_i,p_j)} \in T(\bar{w})$.

Proposition 5.5. *The function $\Lambda_{\bar{w}} : Q(\bar{w}) \to T(\bar{w})$ has in its image all words corresponding to the states of $\mathcal{G}(w)$.*

Proof. Let $\bar{u}$ be a word corresponding to a state of $\mathcal{G}(w)$. Then $\bar{u}$ is a son of a word $\bar{v}$ and, therefore, $\bar{u}$ begins and ends, respectively, at positions i and j corresponding to the left and the right labels of the central basic factorization of $\bar{v}$ ($\bar{u} = \bar{v}_1$), or $i+1$ is the initial position of $\bar{v}$ and j is the position corresponding to one of the labels ($\bar{u} = \bar{v}_0$), or the dual ($\bar{u} = \bar{v}_2$). The numbers p_i and p_j correspond to the ω-powers containing i and j, respectively, that are considered when we read from i to j and from j to i, respectively. Note that the order in which these ω-powers appear, when we read from the borders to the center, is from that of the smallest length to that of the largest length. It follows that $\bar{u} = \bar{w}_{(i,j,p_i,p_j)}$ for the values i, j, p_i and p_j chosen above. □

We note that $\Lambda_{\bar{w}}$ is not an injective function. For example, the empty word is the image of all pairs of the form $(i,i+1,0,0)$, $-1 \leq i < |\bar{w}|$. Moreover, the elements $\bar{w}_{(i,j,p_i,p_j)}$ and $\bar{w}_{(i,j,p_i-1,p_j+1)}$ may have the same image under $\Lambda_{\bar{w}}$. This follows from the fact that the p_i-th ω-power when we read from the left coincides with the (p_j+1)-th ω-power when we read from the right. By Proposition 5.5, we have the following:

Corollary 5.6. *The number of states of $\mathcal{G}(w)$ is at most $(|\bar{w}|+2)^2(K+1)^2$.*

Now, we are ready to determine the complexity of our algorithm. The main function that constructs the automaton consists of a routine that processes each state of the automaton once. For each element of V, it tests if this state has already been processed, involving $\mathcal{O}(|V|) \leq \mathcal{O}(|\bar{w}|^2K^2)$ steps. Then, it computes the left and the right labels with the respective functions. These functions read each letter of the word and, whenever a new letter is found, it is kept in the variable *ll* (respectively, *rl*). The complexity of these functions is $\mathcal{O}(|A|(|\bar{w}| + \Phi))$. Afterwards, the algorithm constructs the sons of the state that is being processed using the function Factorization. This function uses the function Parenthesis and the functions to compute the sons. The function Parenthesis reads the word and computes a list with the positions of the pairs of matching parentheses, with complexity $\mathcal{O}(|\bar{w}|+\Phi)$. The functions which construct the sons read the word corresponding to the state and the ω-powers that will be considered in the new word. So, the complexity of that is $\mathcal{O}(|\bar{w}| + \Phi)$. It follows that the complexity of the function Factorization is $\mathcal{O}(|\bar{w}| + \Phi)$. Hence, the complexity of the algorithm is:

$$|V| \cdot \mathcal{O}(|V| + 2|A|(|\bar{w}| + \Phi) + 3(|\bar{w}| + \Phi)) \leq \mathcal{O}(|\bar{w}|^4K^4). \tag{1}$$

We have proved the following theorem:

Theorem 5.7. *The algorithm that constructs the automaton $\mathcal{G}(w)$, described in the previous subsections, has complexity not exceeding $\mathcal{O}(|\bar{w}|^4K^4)$.*

We have already observed, after Lemma 5.2, that $(2K + 1)|\bar{w}|$ is a higher upper bound for the length of a word corresponding to a state than the upper bound established in the proof of the lemma. However, we use this number make it easier to prove the inequality 1.

Probably, an improvement of the programming and the discovery of a smaller upper bound for the number of states of the automaton allow us to compute a smaller upper bound to the complexity of the computation of $\mathcal{G}(w)$. However, this upper bound can not be smaller than $\mathcal{O}(|\bar{w}|^2)$, as we can see by the following example:

Example 5.8. We consider the sequence of words $(\bar{w}_n)_{n\in\mathbb{N}}$ where $\bar{w}_n = (a_n(a_{n-1}(\cdots(a_1))))$, with $a_i \neq a_j$, if $i \neq j$. We have $|\bar{w}_n| = 3n$ and $|A_n| = n$, where A_n is the alphabet involved in $\bar{w}_n$. We compute the number of states of $\mathcal{G}(w_n)$, $|V_n|$, by recurrence.

For $n = 1$, $\bar{w}_1 = (a_1)$, and for $n = 2$, $\bar{w}_2 = (a_2(a_1))$, the words corresponding to the DA-factors are, respectively, (a_1) and ε, and $(a_2(a_1))$, a_2, $(a_1)(a_2(a_1))$, (a_1) and ε. Hence $\mathcal{G}(w_1)$ and $\mathcal{G}(w_2)$ have, respectively, 2 and 5 states.

Let $\bar{w}_n = (a_n(a_{n-1}(\cdots(a_1))))$, with $n \geq 3$. The central basic factorization of w_n produces the following sons: $a_na_{n-1}\cdots a_2$, $(a_1)(a_2(a_1))\cdots(a_n(a_{n-1}(\cdots(a_1)))) = \bar{w}_1\bar{w}_2\cdots\bar{w}_n$ and $(a_{n-1}(\cdots(a_1))) = \bar{w}_{n-1}$. Thus, the number of states of $\mathcal{G}(w_n)$ is the sum of the number of states of $\mathcal{G}(w_{n-1})$ with the other states corresponding to the DA-factors of w_n and that are not DA-factors of w_{n-1}. Let $\bar{w}_{n(0)} = a_na_{n-1}\cdots a_2$ and $\bar{w}_{n(1)} = (a_1)(a_2(a_1))\cdots(a_n(a_{n-1}(\cdots(a_1)))) = \bar{w}_1\bar{w}_2\cdots\bar{w}_n$ be, respectively, the sons of $\bar{w}_n$ by the edges labeled 0 and 1. The successive iterations of the central basic factorization of $\bar{w}_{n(0)}$ produce the factors $a_{n-1}\cdots a_3$, $a_{n-2}\cdots a_4$, ..., and $a_{\frac{n+3}{2}}a_{\frac{n+1}{2}}$ (respectively, $a_{\frac{n}{2}}$, if n is even). Note that these factors are not states of $\mathcal{G}(w_{n-1})$. Hence $\bar{w}_n$ has $\frac{n-1}{2}$ factors (respectively, $\frac{n}{2}$ factors, if n is even) which are descendants from the left edge of the state $\bar{w}_n$. On the other hand, the central

basic factorization of $\bar{w}_{n(1)}$ produces the factors $\bar{w}_1\bar{w}_2\cdots\bar{w}_{n-1}$, $\bar{w}_{n-1}\bar{w}_n$ and $\bar{w}_{n-1}$. Note that $\bar{w}_{n-1}\bar{w}_n$ is the only factor which is not a state of $\mathcal{G}(w_{n-1})$, since it has in its content the letter a_n. Moreover, the central basic factorization of this factor produces the factors $\bar{w}_{n-1}$ and $\bar{w}_{n-1}\bar{w}_n$, which were already counted. Thus, we count two new factors which are descendants from the central branch. We have the following recurrence formula for the number of states of $\mathcal{G}(w_n)$, with $n \geq 3$:

$$|V_n| = |V_{n-1}| + 3 + \left\lfloor \frac{n}{2} \right\rfloor$$

and, therefore, using basic calculus, we have, for $m \geq 1$,

$$|V_{2m+1}| = 9 + (m+8)(m-1)$$

and

$$|V_{2m}| = 5 + (m+7)(m-1).$$

Hence, the number of states of $\mathcal{G}(w_n)$ is $\Omega(|\bar{w}|^2)$.

Given an automaton $\mathcal{G}(w)$, we construct the finite automaton that recognizes $\mathcal{L}(w)$ by replacing the label of each edge of $\mathcal{G}(w)$ by the ordered pair whose first component is the label of the edge in $\mathcal{G}(w)$ and the second component is the label of the initial state of the edge in $\mathcal{G}(w)$. After that, we minimize the automaton. Brzozowski's Algorithm [7] and Hopcroft's Algorithm [8] to minimize a finite deterministic automaton are well known and they have exponential and $\mathcal{O}(lm\log m)$ complexity, respectively, where l is the cardinality of the alphabet and m is the number of states of the automaton. However, Almeida and Zeitoun [6] described an algorithm to minimize a finite deterministic automaton whose strongly connected non-trivial components are cycles, in time $\mathcal{O}(l+d)$, where d is the number of transitions of the automaton. Note that $\mathcal{G}(w)$ satisfies this condition, since in any cycle of $\mathcal{G}(w)$ the edges are labeled by $(1,x)$, with $x \in A \times A \cup A$ and there is only one edge going from each state with first component labeled 1. As the number of states of the automaton is bounded above by $(|\bar{w}|+2)^2(K+1)^2$, the number $3(|\bar{w}|+2)^2(K+1)^2$ is an upper bound for the number of transitions of the automaton. Furthermore, in 1971, Hopcroft and Karp [9] presented a linear algorithm for testing the equivalence of two finite deterministic automata without requiring previous minimization. So, we have established the following result:

Theorem 5.9. *The word problem for ω-terms over* DA *has a solution in polynomial time, not exceeding $\mathcal{O}((nK)^4)$, where n is the length of the word corresponding to the ω-term and K is the maximum depth of ω-powers.*

Example 5.10. The minimal DA-automaton of the ω-term $w = (ab^\omega caa^\omega)^\omega$ is represented in Figure 2. It follows from identifying states v_{120} and v_{1201} of the automaton $\mathcal{G}(w)$ presented in the Example 5.1. Note that the state v_{120} corresponds to the ω-term a^ω and the state v_{1201} corresponds to the ω-term aa^ω, which are equal over DA.

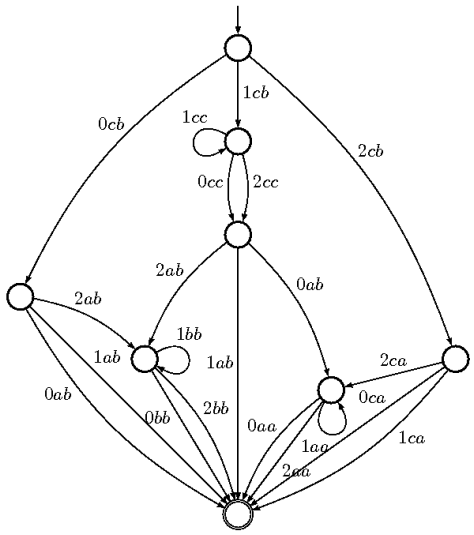

FIGURE 2. The minimal DA-automaton associated to $w = (ab^\omega caa^\omega)^\omega$.

ACKNOWLEDGMENTS

This work is part of the author's doctoral thesis, written under the supervision of Prof. Jorge Almeida, from whose advice the author has greatly benefited. This work was supported by the *Fundação para a Ciência e a Tecnologia (FCT)* through the *PhD grant* SFRH/BD/19720/2004, through the *Centro de Matemática da Universidade do Porto (CMUP)* and also through the project PTDC/MAT/65481/2006, which is partly funded by the European Community Fund FEDER.

REFERENCES

[1] J. Almeida, *Finite Semigroups and Universal Algebra*, World Scientific, Singapore, 1994. English translation.
[2] ————, *A syntactical proof of locality of* DA, Int. J. Algebra and Comp. **6** (1996) 165–177.
[3] ————, *Finite semigroups: an introduction to a unified theory of pseudovarieties*, in Semigroups, Algorithms, Automata and Languages, J.-E. P. G. M. S. Gomes and P. V. Silva, eds., Singapore, 2002, World Scientific, 3–64.
[4] J. Almeida and M. Zeitoun, *The equational theory of ω-terms for finite* R*-trivial semigroups*, in Proc. of the Workshop on Semigroups and Languages, Lisbon 2002, V. H. F. I. M. Araújo M. J. J. Branco and G. M. S. Gomes, eds., World Scientific, 2004, 1–22.
[5] ————, *An automata-theoretic approach to the word problem for ω-terms over* R, Theor. Comp. Sci. **370** (2007) 131–169.
[6] ————, *Description and analysis of a bottom-up DFA minimization algorithm*, Inform. Process. Lett. **107** (2008) 52–59.
[7] J. Brzozowski, *Canonical regular expressions and minimal state graphs for definitive events*, in Proc. Symp. Math. Theor. Automata, New York, 1963, 529–561.
[8] J. E. Hopcroft, *An $n \log n$ algorithm for minimizing states in a finite automaton*, in Theory of machines and computations (Proc. Internat. Sympos., Technion, Haifa, 1971), Z. Kohavi, ed., Academic Press, 1971, 189–196.
[9] J. E. Hopcroft and R. M. Karp, *A linear algorithm for testing equivalence of finite automata*, tech. rep., Cornell University, 1971.

[10] J. McCammond, *Normal forms for free aperiodic semigroups*, Int. J. Algebra and Comp. **11** (2001) 581–625.
[11] A. Moura, *Representations of the free profinite object over* DA, Tech. Rep. CMUP 2009-26, Univ. of Porto, 2009. `http://cmup.fc.up.pt/cmup/v2/include/filedb.php?id=275&table=publicacoes&field=file`.
[12] J. G. Rosenstein, *Linear Orderings*, Academic Press, New York, 1982.
[13] P. Tesson and D. Thérien, *Diamonds are forever: the variety* DA, in Semigroups, Algorithms, Automata and Languages, World Scientific, 2002, 475–499.

Instituto Superior de Engenharia do Porto/LEMA and Centro de Matemática da Universidade do Porto, Rua do Campo Alegre, 687, 4169-007 Porto, Portugal
E-mail address: `aim@isep.ipp.pt/amoura@fc.up.pt`