Minimal Absent Words on Run-Length Encoded Strings

Tooru Akagi
Department of Informatics, Kyushu University, Fukuoka, Japan

Kouta Okabe
Department of Information Science and Technology, Kyushu University, Fukuoka, Japan

Takuya Mieno[1]
Faculty of Information Science and Technology, Hokkaido University, Sapporo, Japan

Yuto Nakashima
Department of Informatics, Kyushu University, Fukuoka, Japan

Shunsuke Inenaga[2]
Department of Informatics, Kyushu University, Fukuoka, Japan
PRESTO, Japan Science and Technology Agency, Kawaguchi, Japan

Abstract

A string w is called a *minimal absent word* for another string T if w does not occur (as a substring) in T and all proper substrings of w occur in T. State-of-the-art data structures for reporting the set $\mathsf{MAW}(T)$ of MAWs from a given string T of length n require $O(n)$ space, can be built in $O(n)$ time, and can report all MAWs in $O(|\mathsf{MAW}(T)|)$ time upon a query. This paper initiates the problem of computing MAWs from a compressed representation of a string. In particular, we focus on the most basic compressed representation of a string, *run-length encoding* (*RLE*), which represents each maximal run of the same characters a by a^p where p is the length of the run. Let m be the RLE-size of string T. After categorizing the MAWs into five disjoint sets $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$, $\mathcal{M}_5$ using RLE, we present matching upper and lower bounds for the number of MAWs in $\mathcal{M}_i$ for $i = 1, 2, 4, 5$ in terms of RLE-size m, except for $\mathcal{M}_3$ whose size is unbounded by m. We then present a compact $O(m)$-space data structure that can report all MAWs in optimal $O(|\mathsf{MAW}(T)|)$ time.

2012 ACM Subject Classification Theory of computation → Pattern matching

Keywords and phrases string algorithms, combinatorics on words, minimal absent words, run-length encoding

Digital Object Identifier 10.4230/LIPIcs.CPM.2022.27

Funding *Takuya Mieno*: JSPS KAKENHI Grant Number JP20J11983
Yuto Nakashima: JSPS KAKENHI Grant Number JP18K18002, JP21K17705
Shunsuke Inenaga: JST PRESTO Grant Number JPMJPR1922

Acknowledgements We thank the anonymous referees for their comments.

1 Introduction

An *absent word* (a.k.a. *a forbidden word*) for a string T is a non-empty string that is *not* a substring of T. An absent word X for T is said to be a *minimal absent word* (*MAW*) for T if all proper substrings of X occur in T. MAWs are combinatorial string objects, and their interesting mathematical properties have extensively been studied in the literature

[1] Current affiliation: University of Electro-Communications, Japan (`tmieno@uec.ac.jp`)
[2] Corresponding author

(see [5, 14, 16, 13, 23, 1] and references therein). MAWs also enjoy several applications including phylogeny [8], data compression [12, 15, 3], musical information retrieval [11], and bioinformatics [2, 9, 24, 21].

Thus, given a string T of length n over an alphabet of size σ, computing the set $\mathsf{MAW}(T)$ of all MAWs for T is an interesting and important problem: Crochemore et al. [14] presented the first efficient data structure of $O(n)$ space which outputs all MAWs in $\mathsf{MAW}(T)$ in $O(\sigma n)$ time and $O(n)$ working space. Since the number $|\mathsf{MAW}(T)|$ of MAWs for T can be as large as $O(\sigma n)$ and there exist strings S for which $|\mathsf{MAW}(S)| \in \Omega(\sigma|S|)$ [14], Crochemore et al.'s algorithm [14] runs in optimal time in the worst case. Later, Fujishige et al. [19] presented an improved data structure of $O(n)$ space, which can report all MAWs in $O(n + |\mathsf{MAW}(T)|)$ time and $O(n)$ working space. Fujishige et al.'s algorithm [19] can easily be modified so it uses $O(|\mathsf{MAW}(T)|)$ time for reporting all MAWs, by explicitly storing all MAWs when $|\mathsf{MAW}(T)| \in O(n)$. The key tool used in these two algorithms is an $O(n)$-size automaton called the *DAWG* [7], which accepts all substrings of T. The DAWG for string T can be built in $O(n \log \sigma)$ time for general ordered alphabets [7], or in $O(n)$ time for integer alphabets of size polynomial in n [19]. There also exist other efficient algorithms for computing MAWs with other string data structures such as suffix arrays and Burrows-Wheeler transforms [6, 4]. MAWs in other settings have also been studied in the literature, including length specified versions [10], the sliding window versions [13, 23, 1], circular string versions [18], and labeled tree versions [17].

In this paper, we initiate the study of computing MAWs for *compressed* strings. As the first step of this line of research, we consider strings which are compactly represented by *run-length encoding* (*RLE*). Let m be the size of the RLE of an input string T. We first categorize the elements of $\mathsf{MAW}(T)$ into five disjoint subsets $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$, and $\mathcal{M}_5$, by considering how the MAWs can be related to the boundaries of maximal character runs in T (Section 2). In Section 3 and Section 4, we present matching upper bounds and lower bounds for their sizes $|\mathcal{M}_i|$ $(i = 1, 2, 4, 5)$ in terms of the RLE size m or the number σ'_T of distinct characters occurring in T. Notice that $\sigma'_T \leq m$ always holds. The exception is $\mathcal{M}_3$, which can contain $\Omega(n)$ MAWs regardless of the RLE size m. Still, in Section 5 we propose our RLE-compressed $O(m)$-space data structure that can enumerate all MAWs for T in output-sensitive $O(|\mathsf{MAW}(T)|)$ time. Since $m \leq n$ always holds, our result is an improvement over Crochemore et al.'s and Fujishige et al.'s results both of which require $O(n)$ space to store representations of all MAWs. Charalampopoulos et al. [10] showed how one can use *extended bispecial factors* of T to represent all MAWs for T in $O(n)$ space, and to output all MAWs in optimal $O(|\mathsf{MAW}(T)|)$ time upon a query. While the way how we characterize the MAWs may be seen as the RLE version of their method based on the extended bispecial factors, our $O(m)$-space data structure cannot be obtained by a straightforward extension from [10], since there exists a family of strings over a constant-size alphabet for which the RLE-size is $m \in O(1)$ but $|\mathsf{MAW}(T)| \in \Omega(n)$. We note that, by the use of *truncated RLE suffix arrays* [25], our $O(m)$-space data structure can be built in $O(m \log m)$ time with $O(m)$ working space (the details of the construction will be presented in the full version of this paper).

2 Preliminaries

2.1 Strings

Let Σ be an ordered alphabet. An element of Σ is called a character. An element of Σ^* is called a string. The length of a string T is denoted by $|T|$. The empty string ε is the string of length 0. If $T = xyz$, then x, y, and z are called a *prefix*, *substring*, and *suffix* of T, respectively. They are called a *proper prefix*, *proper substring*, and *proper suffix* of T if $x \neq T$, $y \neq T$, and $z \neq T$, respectively. For any $1 \leq i \leq |T|$, the i-th character of T is denoted by $T[i]$. For any $1 \leq i \leq j \leq |T|$, $T[i..j]$ denotes the substring of T starting at i and ending at j. For any $i \leq |T|$ and $1 \leq j$, let $T[..i] = T[1..i]$ and $T[j..] = T[j..|T|]$. We say that a string w *occurs* in a string T if w is a substring of T. Note that by definition, the empty string ε is a substring of any string T and hence ε always occurs in T.

Let $\#_T w$ denote the number of occurrences of a string w in a string T. We will abbreviate it to $\#w$ when no confusion occurs.

2.2 Run length encoding (RLE) and bridges

The *run-length encoding* $\mathsf{rle}(T)$ of string T is a compact representation of T such that each maximal run of the same characters in T is represented by a pair of the character and the length of the maximal run. More formally, $\mathsf{rle}(T) = a_1^{p_1} \cdots a_m^{p_m}$ encodes each substring $T[i..i+p-1]$ by a^p if $T[j] = a \in \Sigma$ for every $i \leq j \leq i+p-1$, $T[i-1] \neq T[i]$, and $T[i+p-1] \neq T[i+p]$. Each a^p in $\mathsf{rle}(T)$ is called a (character) *run*, and p is called the exponent of this run. The j-th maximal run in $\mathsf{rle}(T)$ is denoted by r_j, namely $\mathsf{rle}(T) = r_1 \cdots r_m$. The *size* of $\mathsf{rle}(T)$, denoted $R(T)$, is the number of maximal character runs in $\mathsf{rle}(T)$. E.g., for a string $T = \mathtt{aaccccccccbbabbbb}$ of length 18, $\mathsf{rle}(T) = \mathtt{a}^2\mathtt{c}^7\mathtt{b}^2\mathtt{a}^1\mathtt{b}^4$ and $R(T) = 5$.

Our model of computation is a standard word RAM with machine word size $\Omega(\log |T|)$, and the space requirements of our data structures will be measured by the number of words (not bits). Thus, $\mathsf{rle}(T)$ of size m can be stored in $O(m)$ space.

2.3 Bridges

A string $w \in \Sigma^*$ of length $|w| \geq 2$ is said to be a *bridge* if $w[1] \neq w[2]$ and $w[|w|-1] \neq w[|w|]$. In other words, both of the first run and the last run in $\mathsf{rle}(w)$ are of length 1. A substring of T that is a bridge is called a bridge substring of T. Let B_ℓ denote the set of bridge substrings w of T with $R(w) = \ell$. Further let $\mathcal{B} = \bigcup_\ell B_\ell$ be the set of all bridge substrings of T. For example, for the same string $T = \mathtt{aaccccccccbbabbbb}$ as the above one, the substring $\mathtt{ac}^7\mathtt{b}^2\mathtt{a}$ of T is a bridge, and $B_4 = \{\mathtt{ac}^7\mathtt{b}^2\mathtt{a}, \mathtt{cb}^2\mathtt{a}^1\mathtt{b}\}$. For a string w with $R(w) \geq 3$, we can obtain a bridge substring of w by removing the first and the last runs of w and then *shrinking* the runs at both ends so that their exponents are 1. We denote by $\mathsf{shk}(w)$ such shrunk bridge. For convenience, let $\mathsf{shk}(w) = \varepsilon$ if $R(w) \leq 2$. Also, for every $k \geq 2$, we denote $\mathsf{shk}^k(w) = \mathsf{shk}(\mathsf{shk}^{k-1}(w))$. For example, consider the same T as the above again, $\mathsf{shk}(T) = \mathtt{accccccccbbab}$, $\mathsf{shk}^2(w) = \mathtt{cbba}$, $\mathsf{shk}^3(w) = \mathtt{b}$, and $\mathsf{shk}^k(w) = \varepsilon$ for any $k \geq 4$.

2.4 Minimal absent words (MAWs)

A string $w \in \Sigma^*$ is called an *absent word* for a string T if w does not occur in T, namely if $\#w = 0$. An absent word w for T is called a *minimal absent word* or *MAW* for T if all proper substrings of w occur in T. We denote by $\mathsf{MAW}(T)$ the set of all MAWs for T. An alternative definition of MAWs is such that a string aub of length at least two with $a, b \in \Sigma$ and $u \in \Sigma^*$ is a MAW of T if $\#(aub) = 0$, $\#(au) \geq 1$ and $\#(ub) \geq 1$. For a MAW of length 1 (namely a character not occurring in T), we use a convention that $u = \varepsilon$ and a and b are united into a single character.

The MAWs in $\mathsf{MAW}(T)$ are partitioned into the following five disjoint subsets $\mathcal{M}_i$ ($1 \leq i \leq 5$) based on their RLE sizes $R(aub)$:

- $\mathcal{M}_1 = \{aub \in \mathsf{MAW}(T) \mid R(aub) = 1\}$;
- $\mathcal{M}_2 = \{aub \in \mathsf{MAW}(T) \mid R(aub) = 2, u = \varepsilon\}$;
- $\mathcal{M}_3 = \{aub \in \mathsf{MAW}(T) \mid R(aub) = 3, a \neq u[1]$ and $b \neq u[|u|]\}$;
- $\mathcal{M}_4 = \{aub \in \mathsf{MAW}(T) \mid R(aub) \geq 4, a \neq u[1]$ and $b \neq u[|u|]\}$;
- $\mathcal{M}_5 = \{aub \in \mathsf{MAW}(T) \mid R(aub) \geq 2, a = u[1]$ or $b = u[|u|]\}$.

For $1 \leq i \leq 5$, a MAW aub in $\mathcal{M}_i$ is called of *type* i.

In the rest of this paper, we will consider an arbitrarily fixed string T of length n. For convenience, we assume that $n \geq 3$ and that there are special terminal symbols $T[1] = T[n] = \$ \notin \Sigma$ not occurring inside T. Since $\$ \notin \Sigma$, we do not consider any MAW containing \$ for T in our arguments to follow (recall that a MAW must be an element of Σ^*). In addition, since \$ does not occur elsewhere in T, $\mathsf{MAW}(T) = \mathsf{MAW}(T[2..n-1])$ holds.

▶ **Example 1.** Consider $T = \$\mathtt{b}^2\mathtt{ac}^3\mathtt{ba}^2\$ = \$\mathtt{bbacccbaa}\$$. All MAWs in $\mathsf{MAW}(T)$ are divided into the following five types: $\mathcal{M}_1 = \{\mathtt{aaa}, \mathtt{bbb}, \mathtt{cccc}\}$; $\mathcal{M}_2 = \{\mathtt{ca}, \mathtt{bc}\}$; $\mathcal{M}_3 = \{\mathtt{acb}, \mathtt{accb}\}$; $\mathcal{M}_4 = \{\mathtt{cbac}\}$; $\mathcal{M}_5 = \{\mathtt{bbaa}\}$.

Let Σ' denote the set of characters occurring in T except for \$. Let $\sigma' = |\Sigma'|$ be the number of distinct characters occurring in $T[2..n-1]$.

3 Upper bounds on the number of MAWs for RLE strings

In this section, we present upper bounds for the number of MAWs in a string T that is represented by its RLE $\mathsf{rle}(T)$ of size $R(T) = m$.

3.1 Upper bounds for the number of MAWs of type 1, 2, 3, 5

We first consider the number of MAWs except for those of type 4.

▶ **Lemma 2.** $|\mathcal{M}_1| = \sigma$.

Proof. By the definition of $\mathcal{M}_1$, any MAW in $\mathcal{M}_1$ is of the form a^k. For any character $\alpha \in \Sigma'$ that occurs in T, let $aub = \alpha^{p+1}$ such that α^p is the *longest* maximal run of α in T. Clearly $\alpha^p = au = ub$ occurs in T and α^{p+1} does not occur in T. Since $R(aub) = R(\alpha^{p+1}) = 1$, $\alpha^{p+1} \in \mathcal{M}_1$ and it is the unique MAW of type 1 consisting of α's. For any character $\beta \in \Sigma \setminus \Sigma'$ that does not occur in T, clearly β is a MAW of T and $\beta \in \mathcal{M}_1$ since $R(\beta) = 1$. In total, we obtain $|\mathcal{M}_1| = \sigma$. ◀

Note that this upper bound for $|\mathcal{M}_1|$ is tight for any string T and alphabet Σ of size σ.

▶ **Lemma 3.** $|\mathcal{M}_2| \in O((\sigma')^2)$.

Proof. Any MAW in $\mathcal{M}_2$ is of the form ab with $a, b \in \Sigma$ and $a \neq b$. By the definition of MAWs, ab can be a MAW for T only if both a and b occur in T, which implies that $a, b \in \Sigma'$. The number of such combinations of a and b is $\sigma'(\sigma' - 1)$. ◀

Since $\sigma' \leq m$ always holds, we have that $|\mathcal{M}_2| \in O(m^2)$. Later we will show that this upper bound for $|\mathcal{M}_2|$ is asymptotically tight.

▶ **Lemma 4.** $|\mathcal{M}_3|$ *is unbounded by* m.

Proof. Consider a string $T = ac^{n-2}b$, where $a \neq c$ and $c \neq b$. Then ac^kb for each $1 \leq k \leq n-3$ is a MAW of T and $R(ac^kb) = 3$. Since they are the only type 3 MAWs of T, we have that $|\mathcal{M}_3| = n - 3$. Clearly, the original length n of T cannot be bounded by $m = R(T) = 3$. ◀

Although the number of MAWs of type 3 is unbounded by m, later we will present an $O(m)$-space data structure that can enumerate all elements in $\mathcal{M}_3$ in output-sensitive time.

▶ **Lemma 5.** $|\mathcal{M}_5| \in O(m)$.

Proof. Any MAW $aub \in \mathcal{M}_5$ can be represented by $a^{i+1}vb$ or avb^{i+1} with maximal integer $i \geq 1$, where $a^iv = u$ in the former and $vb^i = u$ in the latter. Let us consider the case of $a^{i+1}vb$ as the case of avb^{i+1} is symmetric. Then ca^ivb with some character $c \neq a$ must occur in T. Let k be the beginning position of an occurrence of ca^ivb in T. Then, $T[k+1..k+i] = a^i$ is a maximal run of a.

Now consider any distinct MAW $a^{i+1}v'b' \in \mathcal{M}_5 \setminus \{a^{i+1}vb\}$ with $v'b' \neq vb$. Again, $c'a^iv'b'$ with some character $c' \neq a$ must occur in T. Suppose on the contrary that $c'a^iv'b'$ has an occurrence beginning at the same position k as ca^ivb. This implies that $c' = c$, and both a^ivb and $a^iv'b'$ are prefixes of $T[k+1..|T|]$.

- If $|a^ivb| < |a^iv'b'|$, then a^iv' contains a^ivb as a substring. Since $a^{i+1}v'$ occurs in T, $a^{i+1}vb$ must also occur in T. Hence $a^{i+1}vb$ is not a MAW for T, a contradiction.
- If $|a^ivb| > |a^iv'b'|$, then a^iv contains $a^iv'b'$ as a substring. Thus $a^{i+1}vb$ is an absent word for T but it is not minimal. Hence $a^{i+1}vb$ is not a MAW for T, a contradiction.
- If $|a^ivb| = |a^iv'b'|$, then this contradicts that $a^iub \neq a^iu'b'$.

Hence, at most two element of $\mathcal{M}_5$ can be associated with a position k in T such that $T[k] \neq T[k+1]$. The number of such positions does not exceed $2m$. ◀

3.2 Upper bound for the number of MAWs of type 4

In the rest of this section, we show an upper bound of the number of MAWs of type 4. Namely, we prove the following lemma.

▶ **Lemma 6.** $|\mathcal{M}_4| \in O(m^2)$.

Firstly, we explain a way to characterize MAWs of type 4. For any string $w \in \Sigma^*$ and integer $t > 0$, let $\mathsf{Exp}^t(w)$ be the set of bridges such that $\mathsf{Exp}^t(w) = \{w' \in \mathcal{B} \mid \mathsf{shk}^t(w') = w\}$. Namely, $\mathsf{Exp}^t(w)$ is the *inverse image* of $\mathsf{shk}^t(w') = w$ for bridge substrings w' of T. We use $\mathsf{Exp}(w)$ to denote $\mathsf{Exp}^1(w)$. Figure 1 gives an example for $\mathsf{Exp}^t(w)$ ($\mathsf{Exp}^t_+(w)$ in the figure will be defined later). Any MAW z in $\mathcal{M}_4$ is of the form $a\alpha^iu\beta^jb$ with $a, b, \alpha, \beta \in \Sigma, u \in \Sigma^*$, and positive integers i, j where a, α^i, β^j, b are the first, the second, the second last, and the last run of z, respectively. By the definition of MAWs, both the suffix $\alpha^iu\beta^jb$ and the prefix $a\alpha^iu\beta^j$ of z occur in T. From this fact, we can obtain the following observations.

▶ **Observation 7.** *Each MAW $z \in \mathcal{M}_4$ corresponds to a pair of distinct bridges $(w_1, w_2) \in \mathsf{Exp}(\mathsf{shk}(z)) \times \mathsf{Exp}(\mathsf{shk}(z))$. Formally, for each MAW $z = a\alpha^iu\beta^jb \in \mathcal{M}_4$, there exist characters $a_1, b_1 \in \Sigma \cup \{\$\}$ and integers $i_1 \geq i, j_1 \geq j$ such that $w_1 = a_1\alpha^{i_1}u\beta^jb, w_2 = a\alpha^iu\beta^{j_1}b_1 \in \mathsf{Exp}(\mathsf{shk}(z))$ and $w_1 \neq w_2$ (since these two occur in T but z does not occur in T).*

This observation gives a main idea of our characterization which is stated in the following lemma.

▶ **Lemma 8.** *For any bridge w, $|\{z \mid \mathsf{shk}(z) = w, z \in \mathcal{M}_4\}| \leq |\mathsf{Exp}(w)|(|\mathsf{Exp}(w)| - 1)$.*

Proof. Let $\mathcal{M}_4(w) = \{z \mid \mathsf{shk}(z) = w, z \in \mathcal{M}_4\}$. By Observation 7, each $z \in \mathcal{M}_4(w)$ corresponds to a pair $(w_1, w_2) \in \mathsf{Exp}(\mathsf{shk}(z)) \times \mathsf{Exp}(\mathsf{shk}(z))$ where $w_1 \neq w_2$. Let $z_1 = a_1\alpha^{i_1}u\beta^{j_1}b_1, z_2 = a_2\alpha^{i_2}u\beta^{j_2}b_2$ be distinct MAWs in $\mathcal{M}_4(w)$ where $\mathsf{shk}(z_1) = \mathsf{shk}(z_2) = w$. Assume towards a contradiction that z_1 and z_2 correspond to $(a'\alpha^{i'}u\beta^jb, a\alpha^iu\beta^{j'}b') \in$

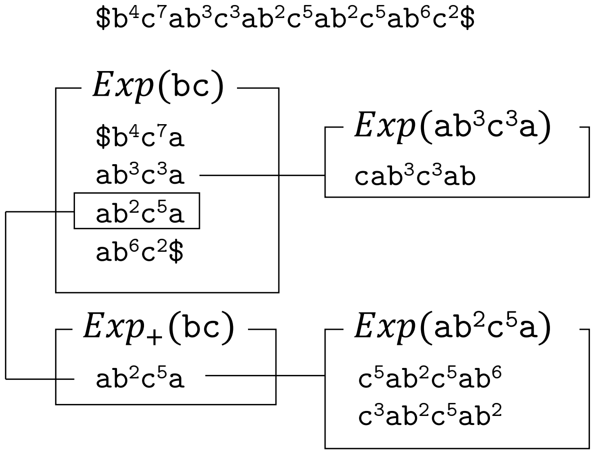

Figure 1 The bridge $w_1 = \mathtt{ab}^2\mathtt{c}^5\mathtt{a} \in \mathsf{Exp}(\mathtt{bc})$ is an element of $\mathsf{Exp}_+(\mathtt{bc})$ since $|\mathsf{Exp}(w_1)| \geq 2$. On the other hand, the bridge $w_2 = \mathtt{ab}^3\mathtt{c}^3\mathtt{a} \in \mathsf{Exp}(\mathtt{bc})$ is not an element of $\mathsf{Exp}_+(\mathtt{bc})$ since $|\mathsf{Exp}(w_2)| < 2$.

$\mathsf{Exp}(w) \times \mathsf{Exp}(w)$. This implies that, by Observation 7, $i = i_1 = i_2, j = j_1 = j_2, a = a_1 = a_2, b = b_1 = b_2$. Thus $z_1 = z_2$ holds, a contradiction. Hence, for any distinct MAWs $z_1, z_2 \in \mathcal{M}_4(w)$, z_1 and z_2 correspond to distinct elements of $\mathsf{Exp}(\mathsf{shk}(z)) \times \mathsf{Exp}(\mathsf{shk}(z))$. Since the number of elements (w_1, w_2) in $\mathsf{Exp}(\mathsf{shk}(z)) \times \mathsf{Exp}(\mathsf{shk}(z))$ such that $w_1 \neq w_2$ is $|\mathsf{Exp}(w)|(|\mathsf{Exp}(w)| - 1)$, this lemma holds. ◀

Since each MAW z corresponds to an element $(w_1, w_2) \in \mathsf{Exp}(\mathsf{shk}(z)) \times \mathsf{Exp}(\mathsf{shk}(z))$ such that $w_1 \neq w_2$, it is enough for the bound to sum up all $|\mathsf{Exp}(w)|^2$ such that $|\mathsf{Exp}(w)| \geq 2$ holds. Let $\mathcal{W}$ be the set of bridges w such that $|\mathsf{Exp}(w)| \geq 2$ or $w \in B_2 \cup B_3$. Let $\mathcal{X} = \sum_{w \in \mathcal{W}} |\mathsf{Exp}(w)|$. For considering such $\mathsf{Exp}(w)$, we also define a subset $\mathsf{Exp}_+^t(w)$ of $\mathsf{Exp}^t(w)$ as follows: For any string (bridge) w and integer $t > 0$,

$$\mathsf{Exp}_+^t(w) = \{w' \mid w' \in \mathsf{Exp}^t(w), |\mathsf{Exp}(w')| \geq 2\}.$$

We also use $\mathsf{Exp}_+(w)$ to denote $\mathsf{Exp}_+^1(w)$. Figure 2 shows an illustration for $\mathsf{Exp}^i(w), \mathsf{Exp}_+^i(w), \mathcal{W}$, and $\mathcal{X}$. We give the following lemma that explains relations between $\mathsf{Exp}^i(w), \mathsf{Exp}_+^i(w)$, and $\mathcal{X}$.

▶ **Lemma 9.**

$$\mathcal{X} = \sum_{w \in B_2 \cup B_3} \left(|\mathsf{Exp}(w)| + \sum_{i=1}^{\lfloor m/2 \rfloor - 1} \sum_{z \in \mathsf{Exp}_+^i(w)} |\mathsf{Exp}(z)| \right).$$

Proof. Let z_{even} be a bridge where $R(z_{\mathsf{even}}) = 2i + 2$ for some $i \geq 1$. Notice that $\mathsf{shk}(z_{\mathsf{even}}) = c_1c_2 \in B_2$ for some distinct characters c_1, c_2. By the definition of $\mathsf{Exp}_+^i(\cdot)$, if $|\mathsf{Exp}(z_{\mathsf{even}})| \geq 2$, then $z_{\mathsf{even}} \in \mathsf{Exp}_+^i(c_1c_2)$. Let z_{odd} be a bridge where $R(z_{\mathsf{odd}}) = 2i + 3$ for some $i \geq 1$. Notice that $\mathsf{shk}(z_{\mathsf{odd}}) = c_1c_2^kc_3 \in B_3$ for some characters c_1, c_2, c_3 and an integer $k \geq 1$. By the definition of $\mathsf{Exp}_+^i(\cdot)$, if $|\mathsf{Exp}(z_{\mathsf{odd}})| \geq 2$, then $z_{\mathsf{odd}} \in \mathsf{Exp}_+^i(c_1c_2^kc_3)$. Therefore the statement holds. ◀

This implies that $|\mathcal{M}_4| \leq \sum_{w \in \mathcal{W}} |\mathsf{Exp}(w)|^2 \leq \mathcal{X}^2$. Thus, if $\mathcal{X} \in O(m)$, $|\mathcal{M}_4| \in O(m^2)$.

We can also observe that $\sum_{i=1}^{\lfloor m/2 \rfloor - 1} \sum_{z \in \mathsf{Exp}_+^i(w)} |\mathsf{Exp}(z)|$ is the sum of the number of children of black nodes (which have more than a single child) in the tree for w. The number of leaves of the tree is an upper bound for the sum. It is also clear that $|\mathsf{Exp}(w)|$ can be

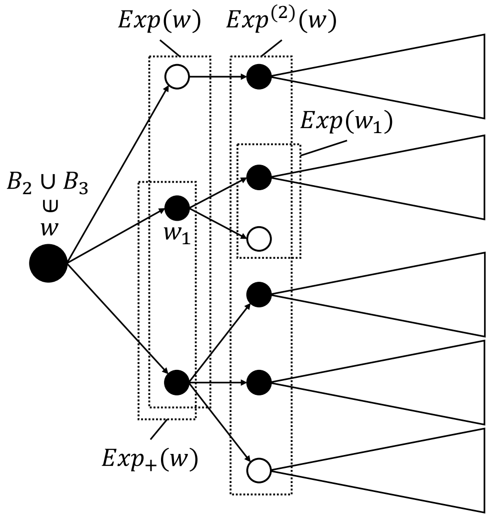

Figure 2 This tree shows an illustration for $\mathsf{Exp}^i(w), \mathsf{Exp}^i_+(w), \mathcal{W}$, and $\mathcal{X}$. The root node represents a bridge $w \in B_2 \cup B_3$. The set of children of the root corresponds to $\mathsf{Exp}(w)$, namely, each child x represents a bridge such that $\mathsf{shk}(x) = w$. Each black node represents a bridge x such that $|\mathsf{Exp}(x)| \geq 2$ (i.e., each black node has at least two children) or the root. Let $W(w)$ be the set of nodes consisting of all the black nodes in the tree rooted at a bridge $w \in B_2 \cup B_3$. Then $\mathcal{W}$ is the union of $W(w)$ for all $w \in B_2 \cup B_3$, and $\mathcal{X}$ is the total number of children of black nodes in $\mathcal{W}$.

bounded by the number of leaves of the tree (In Appendix we give a more mathematical description for the above discussion as Observation 23 and Proposition 24). Consequently, we obtain $|\mathcal{X}| \in O(m)$ as in Lemma 10.

▶ **Lemma 10.** $|\mathcal{X}| \in O(m)$.

Proof. By Lemma 9 and the above discussion, we have

$$\begin{aligned}
\mathcal{X} &= \sum_{w \in B_2 \cup B_3} \left(|\mathsf{Exp}(w)| + \sum_{i=1}^{\lfloor m/2 \rfloor - 1} \sum_{z \in \mathsf{Exp}^i_+(w)} |\mathsf{Exp}(z)| \right) \\
&\leq \sum_{w \in B_2 \cup B_3} 2\#w \\
&\leq 2\left((m-1) + (m-2)\right) \in O(m).
\end{aligned}$$ ◀

We are ready to prove Lemma 6:

Proof of Lemma 6. $|\mathcal{M}_4| \leq \sum_{w \in \mathcal{W}} |\mathsf{Exp}(w)|^2 \leq |\mathcal{X}|^2 \leq (2(2m-3))^2 \in O(m^2)$. ◀

4 Lower bounds on the number of MAWs for RLE strings

In the previous section, we showed a tight bound $|\mathcal{M}_1| = \sigma$, and showed that $|\mathcal{M}_3|$ is unbounded by the RLE size m. In this section, we give tight lower bounds for the sizes of $\mathcal{M}_2$, $\mathcal{M}_3$, and $\mathcal{M}_5$ which asymptotically match the upper bounds given in the previous section. Throughout this section, we omit the terminal \$ at either end of T, since our lower bound instances do not need them.

▶ **Lemma 11.** *There exists a string T such that $|\mathcal{M}_2| = \sigma'(\sigma' - 2) + 1$.*

Proof. Let $T = \mathtt{123} \cdots \sigma'$, where all characters in T are mutually distinct. Any bigram occurring in T is of the form $i(i+1)$ with $1 \leq i < \sigma'$. Thus, for each $1 \leq i < \sigma'$, bigram $i \cdot j$ with any $j \in \{1, \ldots, i-1, i+2, \ldots, \sigma'\}$ is a type-2 MAW for T, and bigram $\sigma' \cdot j$ is a type-2 MAW for T. Namely, the set $\mathcal{M}_2$ of type-2 MAWs for T is:

$$\mathcal{M}_2 = \left\{ \begin{array}{l} \mathtt{13}, \ldots, \mathtt{1}\sigma', \\ \mathtt{21}, \mathtt{24}, \ldots, \mathtt{2}\sigma', \\ \mathtt{31}, \mathtt{32}, \mathtt{35}, \ldots, \mathtt{3}\sigma', \\ \ldots, \\ (\sigma'-1)\mathtt{1}, \ldots, (\sigma'-1)(\sigma'-2), \\ \sigma'\mathtt{1}, \ldots, \sigma'(\sigma'-1) \end{array} \right\}.$$

Thus we have $|\mathcal{M}_2| = \sigma'(\sigma'-2)+1$ for this string T. ◀

Since $\sigma' = m$ for the string T of Lemma 11, we obtain a tight lower bound $|\mathcal{M}_2| \in \Omega(m^2)$ in terms of m. The string $T = \mathtt{123} \cdots \sigma'$ can easily be generalized so that $m < n$, where $n = |T|$. For instance, consider $T' = \mathtt{1}^{p_1}\mathtt{2}^{p_2}\mathtt{3}^{p_3} \cdots \sigma'^{p_{\sigma'}}$ with $p_i > 1$ for each i. The set of type-2 MAWs for T' is equal to that for T.

▶ **Lemma 12.** *There exists a string T with $R(T) = m$ such that $|\mathcal{M}_4| \in \Omega(m^2)$.*

Proof. Consider string $T = \mathtt{abc}^p \cdot \mathtt{ab}^2\mathtt{c}^{p-1} \cdot \mathtt{ab}^3\mathtt{c}^{p-2} \cdot \mathtt{ab}^4\mathtt{c}^{p-3} \cdots \mathtt{ab}^{p-1}\mathtt{c}^2 \cdot \mathtt{ab}^p\mathtt{c} \cdot \mathtt{a}$, where $\mathtt{a}$, $\mathtt{b}$, and $\mathtt{c}$ are mutually distinct characters. Then the set of type-4 MAWs for T is a superset of the following set:

$$\left\{ \begin{array}{l} \mathtt{abca}, \mathtt{abc}^2\mathtt{a}, \ldots, \mathtt{abc}^{p-1}\mathtt{a}, \\ \mathtt{ab}^2\mathtt{ca}, \mathtt{ab}^2\mathtt{c}^2\mathtt{a}, \ldots, \mathtt{ab}^2\mathtt{c}^{p-2}\mathtt{a}, \\ \mathtt{ab}^3\mathtt{ca}, \mathtt{ab}^3\mathtt{c}^2\mathtt{a}, \ldots, \mathtt{ab}^3\mathtt{c}^{p-3}\mathtt{a}, \\ \ldots, \\ \mathtt{ab}^{p-2}\mathtt{ca}, \mathtt{ab}^{p-2}\mathtt{c}^2\mathtt{a}, \\ \mathtt{ab}^{p-1}\mathtt{ca} \end{array} \right\}.$$

Since $m = 3p+1$, we have $|\mathcal{M}_4| > p(p-1)/2 \in \Omega(p^2) = \Omega(m^2)$. ◀

▶ **Lemma 13.** *There exists a string T with $R(T) = m$ such that $|\mathcal{M}_5| \in \Omega(m)$.*

Proof. Consider string $T = \mathtt{abc} \cdot \mathtt{ab}^2\mathtt{c}^2 \cdot \mathtt{ab}^3\mathtt{c}^3 \cdots \mathtt{ab}^p\mathtt{c}^p \cdot \mathtt{a}$, where $\mathtt{a}$, $\mathtt{b}$, and $\mathtt{c}$ are mutually distinct characters. Then the set of type-5 MAWs for T is a superset of the set

$$\{\mathtt{b}^{i+1}\mathtt{c}^i\mathtt{a} \mid 1 \leq i \leq p-1\}.$$

Since $m = 3p+1$, $|\mathcal{M}_5| > p-1 \in \Omega(p) = \Omega(m)$. ◀

5 Efficient representations of MAWs for RLE strings

Consider a string T that contains σ' distinct characters. In this section, we present compact data structures that can output every MAW for T upon query, using a total of $O(m)$ space, where $m = R(T)$ is the size of $\mathsf{rle}(T)$. We will prove the following theorem:

▶ **Theorem 14.** *There exists a data structure D of size $O(m)$ which can output all MAWs for string T in $O(|\mathsf{MAW}(T)|)$ time, where m is the RLE-size of T.*

In our representation of MAWs that follows, we store $\mathsf{rle}(T)$ explicitly with $O(m)$ space. The following is a general lemma that we can use when we output a MAW from our data structures.

▶ **Lemma 15.** *For each MAW $w \in \mathsf{MAW}(T)$, $\mathsf{rle}(w)$ of size $R(w)$ can be retrieved in $O(R(w))$ time from a tuple (a, i, s, t, b, j) and $\mathsf{rle}(T)$, where $a, b \in \Sigma$, $0 \le i, j \le |T|$, and $0 \le s, t \le m$.*

Proof. When $R(w) = 1$ (i.e. $w \in \mathcal{M}_1$), then since w is of the form a^i with $i \ge 1$, we can simply represent it by $(a, i, 0, 0, 0, 0)$.

When $R(w) \ge 2$, then let $w = aub$. When $aub \in \mathcal{M}_2$, then $w = ab$ and thus it can be simply represented by $(a, 1, 0, 0, b, 1)$. When $aub \in \mathcal{M}_3 \cup \mathcal{M}_4$, then $a \ne u[1]$ and $b \ne u[|u|]$. Hence it can be represented by $(a, 1, s, t, b, 1)$ where $r_s \cdots r_t = \mathsf{rle}(u)$. When $aub \in \mathcal{M}_5$, then $a = u[1]$ or $u[|u|] = b$. Let i, j be the maximal integers such that $a^i u' b^j = aub$. We can represent it by (a, i, s, t, b, j) with $r_s \cdots r_t = \mathsf{rle}(u')$. ◀

For ease of discussion, in what follows, we will identify each MAW w with its corresponding tuple (a, i, s, t, b, j) which takes $O(1)$ space.

5.1 Representation for $\mathcal{M}_1$

We have shown that $|\mathcal{M}_1| = \sigma$ (Lemma 2), however, σ can be larger than σ' and m. However, a simple representation for $\mathcal{M}_1$ exists, as follows:

▶ **Lemma 16.** *There exists a data structure D_1 of $O(\sigma') \subseteq O(m)$ space that can output each MAW in $\mathcal{M}_1$ in $O(1)$ time.*

Proof. For ease of explanation, assume that the string T is over the integer alphabet $\Sigma = \{1, \ldots, \sigma\}$ and let $\Sigma' = \{c_1, \ldots, c_{\sigma'}\} \subseteq \{1, \ldots, \sigma\}$. Let $M = \langle c_1^{p_1}, \ldots, c_{\sigma'}^{p_{\sigma'}} \rangle$ be the list of type-1 MAWs in $\mathcal{M}_1$ that are runs of characters in Σ', sorted in the lexicographical order of the characters, i.e. $1 \le c_1 < \cdots < c_{\sigma'} \le \sigma$. We store M explicitly in $O(\sigma')$ space. When we output each MAW in $\mathcal{M}_1$, we test the numbers (i.e. characters) in $\Sigma = \{1, \ldots, \sigma\}$ incrementally, and scan M in parallel: For each $c = 1, \ldots, \sigma$ in increasing order, if $c^p \in M$ with some $p > 1$ then we output c^p, and otherwise we output c. ◀

5.2 Representation for $\mathcal{M}_2$

Recall that $|\mathcal{M}_2| \in O(\sigma'^2) \subseteq O(m^2)$ and this bound is tight in the worst case. Therefore we cannot store all elements of $\mathcal{M}_2$ explicitly, as our goal is an $O(m)$-space representation of MAWs. Nevertheless, the following lemma holds:

▶ **Lemma 17.** *There exists a data structure D_2 of $O(m)$ space that can output each MAW in $\mathcal{M}_2$ in $O(1)$ amortized time.*

Proof. If $|\mathcal{M}_2| \in O(m)$, then we explicitly store all elements of $\mathcal{M}_2$.

If $|\mathcal{M}_2| \in \Omega(m)$, then let D_2 be the trie that represents all bigrams that occur in T. See Figure 3 for a concrete example of D_2. Note that for any pair $a, b \in \Sigma'$ of *distinct* characters both occurring in T, ab is either in D_2 or in $\mathcal{M}_2$. Since the number of such pairs a, b is $\sigma'(\sigma' - 1)$, we have that $\sigma'^2 = \Theta(|\mathsf{D}_2| + |\mathcal{M}_2|)$, where $|\mathsf{D}_2|$ denotes the size of the trie D_2. Since $|\mathsf{D}_2| < m$, we have $\sigma'^2 = O(|\mathcal{M}_2| + m)$. Suppose that the character labels of the out-going edges of each node in D_2 are lexicographically sorted. When we output each element in $\mathcal{M}_2$, we test every bigram ab such that $a \ne b$ and $a, b \in \Sigma'$ in the lexicographical order, and traverse D_2 in parallel in a depth-first manner. We output ab if it is not in the trie D_2. This takes $O(\sigma'^2 + |\mathsf{D}_2|) \subseteq O(|\mathcal{M}_2| + m) = O(|\mathcal{M}_2|)$ time, since $|\mathcal{M}_2| \in \Omega(m)$. ◀

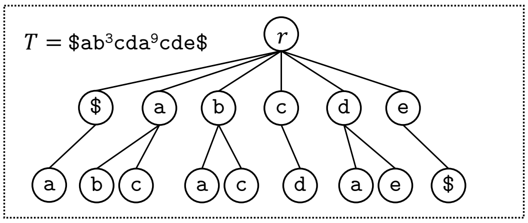

Figure 3 The trie D_2 for string $T = \$\mathtt{ab}^3\mathtt{cda}^9\mathtt{cde}\$$. A bigram ab with $a \neq b$, $a, b \in \Sigma'$ is in $\mathcal{M}_2$ iff ab is not in this trie D_2. For instance, ae and db are MAWs of T.

5.3 Representation for $\mathcal{M}_3$

Recall that the number of MAWs of type 3 in $\mathcal{M}_3$ is unbounded by the RLE size m (Lemma 4). Nevertheless, we show that there exists a compact $O(m)$-space data structure that can report each MAW in $\mathcal{M}_3$ in $O(1)$ time.

Notice that, by definition, a MAW aub of type 3 is a bridge and therefore, it is of the form ac^kb with $c \in \Sigma'_T \setminus \{a, b\}$ and $k \geq 1$.

We begin with some observations. For a triple (a, c, b) of characters with $a \neq c$ and $b \neq c$, let us consider the ordered set $\mathcal{BS}_{acb}(T)$ of bridge substrings of T which are of the form $ac^\ell b$ ($\ell \geq 1$), where the elements in $\mathcal{BS}_{acb}(T)$ are sorted in increasing order of ℓ. Let $\ell_{\max} = \max\{\ell \mid ac^\ell b \in \mathcal{BS}_{acb}(T)\}$. Then, for any $1 \leq k < \ell_{\max}$, $ac^kb \in \mathcal{M}_3$ iff $ac^kb \notin \mathcal{BS}_{acb}(T)$. For instance, consider string $T = \mathtt{ac}^3\mathtt{bac}^9\mathtt{bac}^5\mathtt{bc}^4\mathtt{e}$ for which $\mathcal{BS}_{\mathtt{acb}}(T) = \{\mathtt{ac}^3\mathtt{b}, \mathtt{ac}^5\mathtt{b}, \mathtt{ac}^9\mathtt{b}\}$. Then, $\{\mathtt{ac}^1\mathtt{b}, \mathtt{ac}^2\mathtt{b}, \mathtt{ac}^4\mathtt{b}, \mathtt{ac}^6\mathtt{b}, \mathtt{ac}^7\mathtt{b}, \mathtt{ac}^8\mathtt{b}\}$ is the subset of type-3 MAWs of T of the form $\mathtt{ac}^k\mathtt{b}$. We remark that the above strategy that is based on bridge substrings of the string is not enough to enumerate all elements of $\mathcal{M}_3$, since e.g. $\mathtt{ac}^3\mathtt{e}$ and $\mathtt{bc}^2\mathtt{b}$ are also type-3 MAWs in this running example. This leads us to define the notion of *combined bridges*: A bridge $ac^\ell b$ is a combined bridge of T if (1) $ac^\ell b$ is not a bridge substring of T, (2) ac^ib' and $a'c^jb$ are bridge substrings of T with $b' \neq b$ and $a' \neq a$, and (3) $\ell = \min\{i, j\}$. Let $\mathcal{CB}_c(T)$ denote the set of combined bridges of T with middle character c.

▶ **Observation 18.** *A bridge ac^kb is in $\mathcal{M}_3$ iff $ac^kb \notin \mathcal{BS}_{acb}(T)$ and either (i) $ac^{k'}b \in \mathcal{BS}_{acb}(T)$ with $k' > k$ or (ii) $ac^{k'}b \in \mathcal{CB}_c(T)$ with $k' \geq k$.*

The type-3 MAWs $\mathtt{ac}^3\mathtt{e}$ and $\mathtt{bc}^2\mathtt{b}$ in the running example belong to Case (ii), since $\mathtt{ac}^3\mathtt{e}$ is in $\mathcal{CB}_{\mathtt{c}}(T)$ and $\mathtt{bc}^3\mathtt{b}$ is in $\mathcal{CB}_{\mathtt{c}}(T)$, respectively.

Observation 18 leads us to the following idea: For each character $c \in \Sigma'_T$, let $\mathcal{BS}_c(T) = \bigcup_{a,b \in \Sigma'} \mathcal{BS}_{acb}(T)$ be the ordered set of bridge substrings z of T with $R(z) = 3$ whose middle characters are all c. We suppose that the elements of $\mathcal{BS}_c(T)$ are sorted in increasing order of the exponents ℓ of the middle character c. See Figure 4 for a concrete example for $\mathcal{BS}_c(T)$.

Given $\mathcal{BS}_c(T)$, we can enumerate all type-3 MAWs in $\mathcal{M}_2$ by incrementally constructing a trie T_c of bigrams. Initially, T_c is a trie only with the root. The algorithm has two stages:

First Stage: The first stage deals with Case (i) of Observation 18. We perform a linear scan over $\mathcal{BS}_c(T)$. When we encounter a bridge substring $ac^\ell b$ from $\mathcal{BS}_c(T)$, we traverse the trie T_c with the corresponding bigram ab.

1. If ab is not in the current trie, then ac^kb for all $1 \leq k < \ell$ are MAWs in $\mathcal{M}_3$. After reporting all these MAWs, we create a node v representing ab and store ℓ.
2. If ab is already in the current trie, then the value $\hat{\ell}$ stored in the node v which represents ab is less than ℓ. Then, ac^kb for all $\hat{\ell} < k < \ell$ are MAWs in $\mathcal{M}_3$. After reporting all these MAWs, we update the value in v with ℓ.

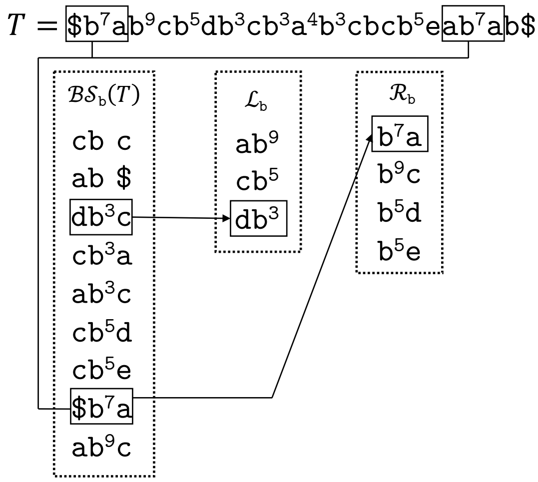

Figure 4 $\mathcal{BS}_{\texttt{b}}$, $\mathcal{L}_{\texttt{b}}$, and $\mathcal{R}_{\texttt{b}}$ for string $T = \texttt{ab}^7\texttt{ab}^9\texttt{cb}^5\texttt{db}^3\texttt{cb}^3\texttt{a}^4\texttt{b}^3\texttt{cbcb}^5\texttt{eab}^7\texttt{abc}$ and character $\texttt{b}$.

The final trie $\mathcal{T}_c$ after the first stage will be unchanged in the following second stage.

Second Stage: The second stage deals with Case (ii) of Observation 18. For each character $a \in \Sigma'_T \setminus \{c\}$, we store the left component ac^i of a bridge substring such that i is the largest exponent of the bridge substrings beginning with ac. Let $\mathcal{L}_c$ be the set of ac^i's for all characters $a \in \Sigma'_T \setminus \{c\}$. Similarly, let $\mathcal{R}_c$ be the set of the right components $c^j b$ for all characters $b \in \Sigma'_T \setminus \{c\}$, where j is the largest exponent of the bridge substrings ending with cb. See Figure 4 for a concrete example for $\mathcal{L}_c$ and $\mathcal{R}_c$.

For each pair of $ac^i \in \mathcal{L}_c$ and $c^j b \in \mathcal{R}_c$, let $ac^\ell b$ be the combined bridge with $\ell = \min\{i, j\}$.

1. If ab is not in the trie T_c, then $ac^k b$ for all $1 \leq k \leq \ell$ are MAWs in $\mathcal{M}_3$.
2. If ab is in the trie T_c, then let $\hat{\ell}$ be the value stored in the node that represents ab.
   - **a.** If $\hat{\ell} < \ell$, then $ac^k b$ for all $\hat{\ell} < k \leq \ell$ are MAWs in $\mathcal{M}_3$.
   - **b.** If $\hat{\ell} \geq \ell$, then we do nothing.

We have the following lemma:

▶ **Lemma 19.** *There exists a data structure D_3 of $O(m)$ space that can output each MAW in $\mathcal{M}_3$ in amortized $O(1)$ time.*

Proof. Analogously to the case of $\mathcal{M}_2$, if $|\mathcal{M}_2| \in O(m)$, then we can explicitly store all type-3 MAWs in $O(m)$ space.

In what follows, we consider the case where $|\mathcal{M}_2| \in \Omega(m)$. For each character $c \in \Sigma'_T$, we perform the above algorithm on $\mathcal{BS}_c(T)$. The correctness of the algorithm follows from Observation 18. Since $\sum_{c \in \Sigma'_T} |\mathcal{BS}_c(T)| \in O(m)$, the total space requirement of the data structure for all characters in Σ'_T is $O(m)$. Let us consider the time complexity. The first stage takes $O(m + f) \subseteq O(|\mathcal{M}_3|)$ time, where f is the number of MAWs reported in the first stage for all characters in Σ'_T. The second stage takes $O(|\mathcal{L}_c| \cdot |\mathcal{R}_c|)$ time for each $c \in \Sigma'_T$. For each combined bridge $ac^\ell b$ created from $\mathcal{L}_c$ and $\mathcal{R}_c$, when it falls into Case 1 or Case 2-a, then at least one MAW is reported. When it falls into Case 2-b, then no MAW is reported. However, in Case 2-b, there has to be a MAW $ac^k b$ that was reported in the first stage. Since we test at most one combined bridge for each pair of characters a, b, a MAW $ac^k b$ reported in the first stage is charged at most once. Therefore, the second stage takes a total of $O(\sum_{c \in \Sigma'_T} |\mathcal{L}_c| \cdot |\mathcal{R}_c|) \subseteq O(|\mathcal{M}_3|)$ time. ◀

5.4 Representation for $\mathcal{M}_4$

Recall that $|\mathcal{M}_4| \in O(m^2)$ and this bound is tight in the worst case. Therefore we cannot store all elements of $\mathcal{M}_4$ explicitly, as our goal is an $O(m)$-space representation of MAWs. Nevertheless, the following lemma holds:

▶ **Lemma 20.** *There exists a data structure* D_4 *of* $O(m)$ *space that can output each MAW in* $\mathcal{M}_4$ *in* $O(1)$ *amortized time.*

Our data structure D_4 is based on the discussion in Section 3.2. We consider the following bipartite graph $G_w = (V_L \cup V_R, E)$ for any bridge $w \in \mathcal{W}$. We can identify each bridge $a\alpha^i u\beta^j b \in \mathsf{Exp}(w)$ by representing the bridge as a 4-tuple (a, i, j, b). Let F_w be the set of 4-tuples which represents all elements in $\mathsf{Exp}(w)$. Two disjoint sets V_L, V_R of vertices and set E of edges are defined as follows:

$$\begin{aligned} V_L &= \{(a, i) \mid \exists(a, i, j, b) \in F_w\}, \\ V_R &= \{(j, b) \mid \exists(a, i, j, b) \in F_w\}, \\ E &= \{((a, i), (j, b)) \mid \exists(a, i, j, b) \in F_w\}. \end{aligned}$$

V_L (resp. V_R) represents the set of the left (resp. right) parts of bridges in $\mathcal{W}$. For each edge in E represents a bridge in $\mathcal{W}$. This implies that $|E| = |\mathsf{Exp}(w)|$. Assume that all vertices in V_L (resp. V_R) are sorted in non-decreasing order w.r.t. the value i (resp. j) which represents the exponent of corresponding run. For any $k \in [1, |V_L|]$ and $k' \in [1, |V_R|]$, $v_L(k) = (\mathsf{c}_L(k), \mathsf{e}_L(k))$ denotes the k-th vertex in V_L, and $v_R(k') = (\mathsf{c}_R(k'), \mathsf{e}_R(k'))$ denotes the k'-th vertex in V_R. For any vertex $v_L(k) \in V_L$ and $v_R(k') \in V_R$, we also define

$$E^{LR}_{max}(k) = \max\{\mathsf{e}_R(i) \mid \exists(v_L(k), v_R(i)) \in E\},$$

$$E^{RL}_{max}(k') = \max\{\mathsf{e}_L(i) \mid \exists(v_R(i), v_R(k)) \in E\}.$$

Figure 5 gives an illustration for this graph. Due to Observation 7, each MAW z of type 4 corresponds to an element of $\mathsf{Exp}(w) \times \mathsf{Exp}(w)$ where $z^{(1)} = w$. By this idea, we detect each MAW as a pair of vertices in $V_L \times V_R$ which is not an edge in E. The following lemma explains all MAWs which can be represented by the graph.

▶ **Lemma 21.** *For any vertices* $v_L(k) \in V_L$ *and* $v_R(k') \in V_R$ *of* $G_{\alpha u\beta}$*, the string* $\mathsf{c}_L(k)\alpha^{\mathsf{e}_L(k)}u\beta^{\mathsf{e}_R(k')}\mathsf{c}_R(k')$ *is a MAW iff the following three conditions hold (see also Figure 6 for an illustration):*

- $(v_L(k), v_R(k')) \notin E$,
- $E^{LR}_{max}(k) \geq \mathsf{e}_R(k')$, *and*
- $E^{RL}_{max}(k') \geq \mathsf{e}_L(k)$.

Proof. If $(v_L(k), v_R(k')) \notin E$, $\mathsf{c}_L(k)\alpha^{\mathsf{e}_L(k)}u\beta^{\mathsf{e}_R(k')}\mathsf{c}_R(k')$ is an absent word. $E^{LR}_{max}(k) \geq \mathsf{e}_R(k')$ and $E^{RL}_{max}(k') \geq \mathsf{e}_L(k)$ implies that $\mathsf{c}_L(k)\alpha^{\mathsf{e}_L(k)}u\beta^{\mathsf{e}_R(k')}$ and $\alpha^{\mathsf{e}_L(k)}u\beta^{\mathsf{e}_R(k')}\mathsf{c}_R(k')$ occur in the string. Thus $\mathsf{c}_L(k)\alpha^{\mathsf{e}_L(k)}u\beta^{\mathsf{e}_R(k')}\mathsf{c}_R(k')$ is a MAW.

On the other hand, if $(v_L(k), v_R(k')) \in E$, $\mathsf{c}_L(k)\alpha^{\mathsf{e}_L(k)}u\beta^{\mathsf{e}_R(k')}\mathsf{c}_R(k')$ occurs in the text. $E^{LR}_{max}(k) < \mathsf{e}_R(k')$ implies that $\mathsf{c}_L(k)\alpha^{\mathsf{e}_L(k)}u\beta^{\mathsf{e}_R(k')}$ does not occur in the string. $E^{RL}_{max}(k') < \mathsf{e}_L(k)$ implies that $\alpha^{\mathsf{e}_L(k)}u\beta^{\mathsf{e}_R(k')}\mathsf{c}_R(k'))$ does not occur in the string. Thus all three conditions hold if $\mathsf{c}_L(k)\alpha^{\mathsf{e}_L(k)}u\beta^{\mathsf{e}_R(k')}\mathsf{c}_R(k')$ is a MAW. ◀

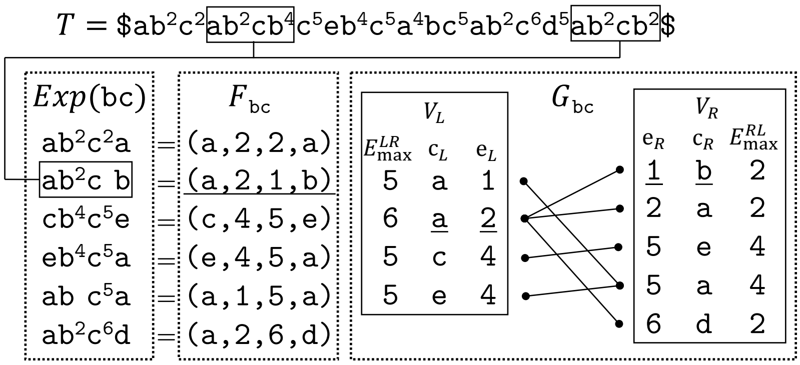

Figure 5 This figure shows $G_{\mathtt{bc}}$ for $T = \$\mathtt{ab}^2\mathtt{c}^2\mathtt{ab}^2\mathtt{cb}^4\mathtt{c}^5\mathtt{eb}^4\mathtt{c}^5\mathtt{a}^4\mathtt{bc}^5\mathtt{ab}^2\mathtt{c}^6\mathtt{d}^5\mathtt{ab}^2\mathtt{cb}^2\$$. For a bridge bc, $\mathsf{Exp}(\mathtt{bc})$ has 6 bridges. $F_{\mathtt{bc}}$ contains 6 tuples which represents all bridges in $\mathsf{Exp}(\mathtt{bc})$. For instance, a bridge $\mathtt{ab}^2\mathtt{cb} = (\mathtt{a}, 2, 1, \mathtt{b})$ where the first character is a, the exponent of the second run is 2, the exponent of the second last run is 1, and the last character is b. V_L is the set of pairs by the left-half of elements in $F_{\mathtt{bc}}$. In this example, V_L has 4 vertices $\{(\mathtt{a}, 1), (\mathtt{a}, 2), (\mathtt{c}, 4), (\mathtt{e}, 4)\}$ which are sorted in non-decreasing order of the second key (representing its exponent). V_R is the symmetric set for the right parts. Each bridge corresponds to an edge. For example, the second bridge $\mathtt{ab}^2\mathtt{cb}$ in the figure corresponds to the edge from the second vertex $(\mathtt{a}, 2)$ in V_L to the first vertex $(1, \mathtt{b})$ in V_R. Since the number of bridges in $\mathsf{Exp}(\mathtt{bc})(F_{\mathtt{bc}})$ is 6, the graph has 6 edges.

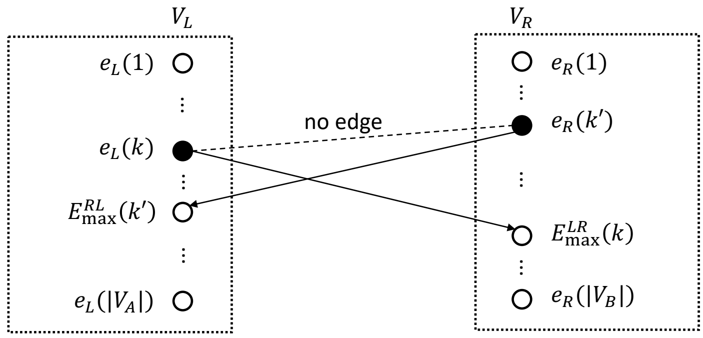

Figure 6 This is an illustration for Lemma 21. For the k-th vertex $v_L(k) \in V_L$ and k'-th vertex $v_R(k') \in V_R$, this graph satisfies the three conditions of the lemma.

Proof of Lemma 20. Let x be the number of outputs. If $x < m$, we can just store all the MAWs themselves. Assume that $x \in \Omega(m)$.

For all bridge $w = \alpha u \beta \in \mathcal{W}$, G_w represents all MAWs which correspond to elements in $\mathsf{Exp}(w) \times \mathsf{Exp}(w)$. Our data structure D_4 consists of G_w for any $w \in \mathcal{W}$. It is clear that G_w can be stored in $O(|\mathsf{Exp}(w)|)$ space. This implies that the size of D_4 is linear in $\mathcal{X}$, namely, D_4 can be stored in $O(m)$ space (Lemma 10).

We can output all MAWs which are represented by G_w based on Lemma 21 (see Algorithm 1). For the k-th vertex $v_L(k)$, C represents all vertices $v_R(k')$ in V_B such that $(v_L(k), v_R(k')) \notin E$ and $E^{RL}_{max}(k') \geq \mathsf{e}_L(k)$ (the first and third condition in Lemma 21). For each vertex in C, if $E^{LR}_{max}(k) \geq \mathsf{e}_R(k')$ (the second condition in Lemma 21), the algorithm outputs a MAW $\mathsf{c}_L(k)\alpha^{\mathsf{e}_L(k)}u\beta^{\mathsf{e}_R(k')}\mathsf{c}_R(k')$. Then the running time of our algorithm is $O(x + \sum_{w \in \mathcal{W}} |G_w|) \subseteq O(x + m) = O(x)$, since $x \in \Omega(m)$. ◀

Algorithm 1 Compute all MAWs in $\mathcal{M}_4$.

Input: bipartite graph $G_{\alpha u \beta} = (V_L, V_R, E)$
Output: all MAWs in $\mathcal{M}_4$ that are associated by $\alpha u \beta$, $a\alpha^{k_1} u \beta^{k_2} b$ for $a, b \in \Sigma, k_1, k_2 \in \mathbb{N}$

$C_R \leftarrow V_R$
for each $v_L(k) \in V_L$ **do**
  $C = \{v_R(k') \in C_R \mid \mathsf{e}_R(k') \le E^{LR}_{max}(k)\} \setminus \{v \mid (v_L(k), v) \in E\}$
  for each $v_R(k') \in C$ **do**
    if $E^{RL}_{max}(k') \ge \mathsf{e}_L(v_L(k))$ **then**
      output $\mathsf{c}_L(k)\alpha^{\mathsf{e}_L(k)} u \beta^{\mathsf{e}_R(k')} \mathsf{c}_R(k')$
    else
      $C_R \leftarrow C_R \setminus \{v_R(k')\}$
    end if
  end for
end for

5.5 Representation for $\mathcal{M}_5$

▶ **Lemma 22.** *There exists a data structure of size $O(m)$ that outputs each element of $\mathcal{M}_5$ in $O(1)$ time.*

Proof. By Lemma 5, $|\mathcal{M}_5| \in O(m)$. Recall that an element of M_5 can be as long as $O(n)$. However, using Lemma 15 we can represent and store all elements in $\mathcal{M}_5$ in a total of $O(m)$ space. It is trivial that each stored element can be output in $O(1)$ time. ◀

6 Conclusions and open questions

Minimal absent words (MAWs) are combinatorial string objects that can be used in applications such as data compression (anti-dictionaries) and bioinformatics. In this paper, we considered MAWs for a string T that is described by its run-length encoding (RLE) $\mathsf{rle}(T)$ of size m. We first analyzed the number of MAWs for a string T in terms of its RLE size m, by dividing the set $\mathsf{MAW}(T)$ of all MAWs for T into five disjoint types. Albeit the number of MAWs of some types is superlinear in m, we devised a compact $O(m)$-space representation for $\mathsf{MAW}(T)$ that can output all MAWs in output-sensitive $O(|\mathsf{MAW}(T)|)$ time.

We would like to remark that our $O(m)$-space representation can be built in $O(m \log m)$ time with $O(m)$ space, with the help of the *truncated RLE suffix array* (*tRLESA*) data structure [25]. A suffix s of T is called a tRLE suffix of T if $s = ar_i \cdots r_m$ where the first a is the last character in the previous run r_{i-1}. $\mathsf{tRLESA}(T)$ for $\mathsf{rle}(T) = r_1 \cdots r_m$ is an integer array of length m such that $\mathsf{tRLESA}(T)[i] = k$ iff $ar_i \cdots r_m$ is the k-th lexicographically smallest tRLE suffix for T. tRLESA occupies $O(m)$ space, and can be built in $O(m \log m)$ time with $O(m)$ working space [25]. The details for our tRLESA-based construction algorithm for our $O(m)$-space MAW representation will appear in the full version of this paper.

An interesting open question is whether there exist other compressed representations of MAWs, based on e.g. grammar-based compression [20], Lempel-Ziv 77 [26], and run-length Burrows-Wheeler transform [22].

References

1 Tooru Akagi, Yuki Kuhara, Takuya Mieno, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Combinatorics of minimal absent words for a sliding window. *CoRR*, abs/2105.08496, 2021. `arXiv:2105.08496`.

2 Yannis Almirantis, Panagiotis Charalampopoulos, Jia Gao, Costas S. Iliopoulos, Manal Mohamed, Solon P. Pissis, and Dimitris Polychronopoulos. On avoided words, absent words, and their application to biological sequence analysis. *Algorithms for Molecular Biology*, 12(1):5, 2017.

3 Lorraine A. K. Ayad, Golnaz Badkobeh, Gabriele Fici, Alice Héliou, and Solon P. Pissis. Constructing antidictionaries of long texts in output-sensitive space. *Theory Comput. Syst.*, 65(5):777–797, 2021.

4 Carl Barton, Alice Heliou, Laurent Mouchard, and Solon P. Pissis. Linear-time computation of minimal absent words using suffix array. *BMC Bioinformatics*, 15(1):388, 2014.

5 Marie Pierre Béal, Filippo Mignosi, and Antonio Restivo. Minimal forbidden words and symbolic dynamics. In *STACS 1996*, pages 555–566, 1996.

6 Djamal Belazzougui, Fabio Cunial, Juha Kärkkäinen, and Veli Mäkinen. Versatile succinct representations of the bidirectional Burrows-Wheeler transform. In *ESA 2013*, pages 133–144, 2013.

7 Anselm Blumer, J. Blumer, David Haussler, Andrzej Ehrenfeucht, M. T. Chen, and Joel I. Seiferas. The smallest automaton recognizing the subwords of a text. *Theor. Comput. Sci.*, 40:31–55, 1985.

8 Supaporn Chairungsee and Maxime Crochemore. Using minimal absent words to build phylogeny. *Theor. Comput. Sci.*, 450:109–116, 2012.

9 Panagiotis Charalampopoulos, Maxime Crochemore, Gabriele Fici, Robert Mercas, and Solon P. Pissis. Alignment-free sequence comparison using absent words. *Inf. Comput.*, 262:57–68, 2018.

10 Panagiotis Charalampopoulos, Maxime Crochemore, and Solon P Pissis. On extended special factors of a word. In *SPIRE 2018*, pages 131–138. Springer, 2018.

11 Tim Crawford, Golnaz Badkobeh, and David Lewis. Searching page-images of early music scanned with OMR: A scalable solution using minimal absent words. In *ISMIR 2018*, pages 233–239, 2018.

12 M. Crochemore, F. Mignosi, A. Restivo, and S. Salemi. Data compression using antidictionaries. *Proc. IEEE*, 88(11):1756–1768, 2000.

13 Maxime Crochemore, Alice Héliou, Gregory Kucherov, Laurent Mouchard, Solon P. Pissis, and Yann Ramusat. Absent words in a sliding window with applications. *Information and Computation*, 270:104461, 2020.

14 Maxime Crochemore, F. Mignosi, and A. Restivo. Automata and forbidden words. *Information Processing Letters*, 67(3):111–117, 1998.

15 Maxime Crochemore and Gonzalo Navarro. Improved antidictionary based compression. In *12th International Conference of the Chilean Computer Science Society, 2002. Proceedings.*, pages 7–13. IEEE, 2002.

16 Gabriele Fici. *Minimal forbidden words and applications*. PhD thesis, Università di Palermo and Université Paris-Est Marne-la-Vallée, 2006.

17 Gabriele Fici and Pawel Gawrychowski. Minimal absent words in rooted and unrooted trees. In *SPIRE 2019*, pages 152–161, 2019.

18 Gabriele Fici, Antonio Restivo, and Laura Rizzo. Minimal forbidden factors of circular words. *Theor. Comput. Sci.*, 792:144–153, 2019.

19 Yuta Fujishige, Yuki Tsujimaru, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Computing DAWGs and minimal absent words in linear time for integer alphabets. In *MFCS 2016*, volume 58, pages 38:1–38:14, 2016.

20 John C. Kieffer and En-Hui Yang. Grammar-based codes: A new class of universal lossless source codes. *IEEE Trans. Inf. Theory*, 46(3):737–754, 2000. doi:10.1109/18.841160.

21 Grigorios Koulouras and Martin C Frith. Significant non-existence of sequences in genomes and proteomes. *Nucleic acids research*, 49(6):3139–3155, 2021.

22 Veli Mäkinen and Gonzalo Navarro. Succinct suffix arrays based on run-length encoding. *Nord. J. Comput.*, 12(1):40–66, 2005.

23 Takuya Mieno, Yuki Kuhara, Tooru Akagi, Yuta Fujishige, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Minimal unique substrings and minimal absent words in a sliding window. In *SOFSEM 2020*, volume 12011 of *Lecture Notes in Computer Science*, pages 148–160. Springer, 2020.

24 Diogo Pratas and Jorge M Silva. Persistent minimal sequences of SARS-CoV-2. *Bioinformatics*, 36(21):5129–5132, 2020.

25 Yuya Tamakoshi, Keisuke Goto, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. An opportunistic text indexing structure based on run length encoding. In *CIAC 2015*, volume 9079 of *Lecture Notes in Computer Science*, pages 390–402. Springer, 2015.

26 J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory*, IT-23(3):337–349, 1977.

A Appendix

We give a supplemental proposition that can be useful for analyzing the upper bound on the number of MAWs of type 4.

We begin with the following observation:

▶ **Observation 23.** *For any bridge substring $w \in \Sigma^*$ of T,*

$$|\mathsf{Exp}(w)| = \#w - \sum_{z \in \mathsf{Exp}(w)} (\#z - 1) \leq \#w + |\mathsf{Exp}_{+}(w)| - \sum_{z \in \mathsf{Exp}_{+}(w)} \#z.$$

Note that $\sum_{z \in \mathsf{Exp}_{+}(w)} (\#z - 1) \leq \sum_{z \in \mathsf{Exp}(w)} (\#z - 1)$ since $\#z - 1 = 0$ when $z \in \mathsf{Exp}(w) \setminus \mathsf{Exp}_{+}(w)$. Below we present Proposition 24 which gives an upper bound for $\mathcal{X}$.

▶ **Proposition 24.** *For any bridge w and $t \geq 1$ such that $|\mathsf{Exp}(w)| \geq 2$,*

$$|\mathsf{Exp}(w)| + \sum_{i=1}^{t} \sum_{z \in \mathsf{Exp}^i_{+}(w)} |\mathsf{Exp}(z)| \quad \leq \quad \#w + \sum_{i=1}^{t} |\mathsf{Exp}^i_{+}(w)|. \tag{1}$$

Proof. We prove this lemma by induction on t. By Observation 23 and $|\mathsf{Exp}(w)| \leq \#w$ for any w, we have

$$|\mathsf{Exp}(w)| + \sum_{z \in \mathsf{Exp}_{+}(w)} |\mathsf{Exp}(z)| \leq (\#w + |\mathsf{Exp}_{+}(w)| - \sum_{z \in \mathsf{Exp}_{+}(w)} \#z) + \sum_{z \in \mathsf{Exp}_{+}(w)} \#z = \#w + |\mathsf{Exp}_{+}(w)|.$$

Thus, the statement holds for $t = 1$. Suppose that the statement holds for some $t' \geq 1$.

$$\begin{aligned}
& |\mathsf{Exp}(w)| + \sum_{i=1}^{t'+1} \sum_{z \in \mathsf{Exp}_+^i(w)} |\mathsf{Exp}(z)| \\
= \; & |\mathsf{Exp}(w)| + \sum_{w' \in \mathsf{Exp}_+(w)} \left(|\mathsf{Exp}(w')| + \sum_{i=1}^{t'} \sum_{z \in \mathsf{Exp}_+^i(w')} |\mathsf{Exp}(z)| \right) \\
\leq \; & |\mathsf{Exp}(w)| + \sum_{w' \in \mathsf{Exp}_+(w)} \left(\#w' + \sum_{i=1}^{t'} |\mathsf{Exp}_+^i(w')| \right) \text{ (by induction hypothesis)} \\
\leq \; & \left(\#w + |\mathsf{Exp}_+(w)| - \sum_{w' \in \mathsf{Exp}_+(w)} \#w' \right) + \sum_{w' \in \mathsf{Exp}_+(w)} \#w' + \sum_{w' \in \mathsf{Exp}_+(w)} \sum_{i=1}^{t'} |\mathsf{Exp}_+^i(w')| \\
& \text{(by Observation 23)} \\
= \; & \#w + |\mathsf{Exp}_+(w)| + \sum_{w' \in \mathsf{Exp}_+(w)} \sum_{i=1}^{t'} |\mathsf{Exp}_+^i(w')| \\
\leq \; & \#w + |\mathsf{Exp}_+(w)| + \sum_{i=2}^{t'+1} |\mathsf{Exp}_+^i(w)| \\
= \; & \#w + \sum_{i=1}^{t'+1} |\mathsf{Exp}_+^i(w)|
\end{aligned}$$

Thus, the statement holds for $t' + 1$. Therefore, the statement holds for any $t \geq 1$. ◀