On Extended Special Factors of a Word

Panagiotis Charalampopoulos[1⋆], Maxime Crochemore[1,2], and Solon P. Pissis[1]

[1] Department of Informatics, King's College London, London, UK
[panagiotis.charalampopoulos,maxime.crochemore,solon.pissis]@kcl.ac.uk
[2] Université Paris-Est, Marne-la-Vallée, France

Abstract. An *extended special* factor of a word x is a factor of x whose longest infix can be extended by at least two distinct letters to the left or to the right and still occur in x. It is called *extended bispecial* if it can be extended in *both* directions and still occur in x. Let $\rho(n)$ be the maximum number of extended bispecial factors over all words of length n. Almirantis et al have shown that $2n - 6 \leq \rho(n) \leq 3n - 4$ [WABI 2017]. In this article, we show that there is no constant $c < 3$ such that $\rho(n) \leq cn$. We then exploit the connection between extended special factors and minimal absent words to construct a data structure for computing minimal absent words of a specific length in optimal time for integer alphabets generalising a result by Fujishige et al [MFCS 2016]. As an application of our data structure, we show how to compare two words over an integer alphabet in optimal time improving on another result by Charalampopoulos et al [Inf. Comput. 2018].

Keywords: special factors · minimal absent words · string algorithms.

1 Introduction

We begin with basic definitions and notation, generally following [14]. Let $x = x[0]x[1]\dots x[n-1]$ be a *word* of *length* $n = |x|$ over a finite ordered *alphabet* Σ of size σ, i.e. $\sigma = |\Sigma|$. In particular, we consider the case of an *integer alphabet*; in this case each letter is replaced by its rank such that the resulting word consists of integers in the range $\{1, \dots, n\}$. In what follows we assume without loss of generality that $\Sigma = \{0, 1, \dots, \sigma - 1\}$. We also define Σ_x to be the alphabet of word x and $\sigma_x = |\Sigma_x|$. For two positions i and j on x, we denote by $x[i\,..\,j] = x[i]\dots x[j]$ the *factor* (sometimes called *subword*) of x that starts at position i and ends at position j (it is empty if $j < i$), and by ε the *empty word*, word of length 0. We recall that a *prefix* of x is a factor that starts at position 0 ($x[0\,..\,j]$) and a *suffix* is a factor that ends at position $n-1$ ($x[i\,..\,n-1]$). A factor of x is called *proper* if it is not x itself. If a word y is both a proper prefix and a proper suffix of a non-empty word x, then y is called a *border* of x. A factor $x[i\,..\,j]$ of x that is neither a prefix nor a suffix of x is called an *infix* of x.

⋆ Partially supported by a Studentship from the Faculty of Natural and Mathematical Sciences at King's College London and an A. G. Leventis Foundation Educational Grant.

Let $w = w[0\,..\,m-1]$ be a word, $0 < m \leq n$. We say that there exists an *occurrence* of w in x, or, more simply, that w *occurs in* x, if w is a factor of x. Every occurrence of w can be characterised by a starting position in x. Thus we say that w occurs at (starting) position i in x when $w = x[i\,..\,i+m-1]$.

A factor $u \neq \varepsilon$ of a word x is called *bispecial* if there exist $a, b, c, d \in \Sigma$ with $a \neq b$ and $c \neq d$ such that au, bu, uc and ud occur in x. The notion of special factors has been extensively studied in literature, mainly in the case of infinite words or infinite languages [19, 20, 18, 4, 8–10]. We extend this definition here as follows. We call *extended left-special* the factors ayb, where $a, b \in \Sigma$, $y \neq \varepsilon$ is a factor of x and cy occurs in x for some $c \in \Sigma \setminus \{a\}$. Similarly, we call *extended right-special* the factors ayb, where $a, b \in \Sigma$, $y \neq \varepsilon$ is a factor and yd occurs in x for some $d \in \Sigma \setminus \{b\}$. Factors that are both extended left-special and extended right-special are called *extended bispecial*. The following result is known.

Lemma 1 ([2]). *For any word x of length n the number of extended right-special factors is no more than $3n - 2 - 2\sigma_x$.*

By symmetry the same bound holds for extended left-special factors. It also holds for extended bispecial factors, since these are a subset of extended right-special factors. In [2], the authors provide a word with a linear number of extended bispecial factors: $\mathtt{ba}^{n-2}\mathtt{b}$ which has $2n-6$ of them. Let $\rho(n)$ be the maximum number of extended bispecial factors over all words of length n.

Theorem 2 ([2]). $2n - 6 \leq \rho(n) \leq 3n - 4$.

The main algorithm presented in [2] computes statistically overabundant words of a word over an integer alphabet in linear time, by first computing all extended right-special factors of the word and then filtering out some of them based on a simple computation. We can easily adapt the algorithm to compute the extended left-special factors; the extended bispecial factors can be then retrieved easily within the same complexity. We thus know the following.

Theorem 3 ([2]). *Given a word of length n over an integer alphabet all extended left-, right-special and bispecial factors can be computed in $\mathcal{O}(n)$ time.*

2 A Lower Bound on Extended Bispecial Factors

In this section, we improve the lower bound of Theorem 2.

Definition 4. *A word x over an alphabet Σ of size σ is a de Bruijn sequence of order k if and only if all words of length k over Σ occur exactly once in x.*

By definition, a de Bruijn sequence of order k has length $\sigma^k + k - 1$.

Theorem 5. *There is no constant $c < 3$ such that $\rho(n) \leq cn$.*

Proof. In a de Bruijn sequence of order k all words over Σ of lengths 3 to k (inclusive) are extended bispecial factors. In addition, by the definition of de Bruijn sequences, the $\sigma^k - 1$ subwords of x of length $k+1$ are all distinct and each of them is an extended bispecial factor as its longest infix is of length $k-1$ and hence it can be extended by all letters in Σ in any direction and still occur in x. We thus have at least

$$\sigma^3 + \ldots + \sigma^k + \sigma^k - 1 = \sigma^k - 1 + \sigma^3 \cdot \sum_{i=0}^{k-3} \sigma^i = \sigma^k - 1 + \sigma^3 \cdot \frac{\sigma^{k-2} - 1}{\sigma - 1}$$

extended bispecial factors. By letting $\sigma = 2$, the above formula becomes $2^k - 1 + 2(2^k - 4) = 3 \cdot 2^k - 9$. We now look at the ratio of the number of bispecial factors over the length of the sequence as k increases and have that

$$\lim_{k \to \infty} \frac{3 \cdot 2^k - 9}{2^k + k - 1} = 3$$

by L'Hôpital's rule. □

3 Minimal Absent Words via Extended Special Factors

The word y is an *absent word* of x if it does not occur in x. The absent word y of x is *minimal* if and only if all its proper factors occur in x. The set of all minimal absent words for a word x is denoted by $\mathcal{M}_x$. The set of all minimal absent words of length ℓ for a word x is denoted by $\mathcal{M}_x^\ell$. For example, if $x = \texttt{abaab}$, then $\mathcal{M}_x = \{\texttt{aaa}, \texttt{aaba}, \texttt{bab}, \texttt{bb}\}$ and $\mathcal{M}_x^3 = \{\texttt{aaa}, \texttt{bab}\}$. If we suppose that all the letters of Σ appear in x and $|x| = n$, the length of a minimal absent word of x lies between 2 and $n+1$. It can be equal to $n+1$ if and only if x is of the form a^n, $a \in \Sigma$. So, if x contains occurrences of at least two different letters, the length of any minimal absent word of x is upper bounded by n. In what follows, we perform the computations considering all minimal absent words of length at least 3; the ones of length 2 can be handled separately in the same manner.

The upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$ and it is tight for integer alphabets [12]; in fact, for large alphabets, such as when $\sigma \geq \sqrt{n}$, this bound is also tight even for minimal absent words of the same length [1].

In many real-world applications of minimal absent words, such as in sequence comparison [12], data compression [17], on-line pattern matching [15], and identifying pathogen-specific signatures [24], only a subset of minimal absent words may be considered, and, in particular, the minimal absent words of length (at most) ℓ. State-of-the-art algorithms compute all minimal absent words of x in $\mathcal{O}(\sigma n)$ time [16, 3] or in $\mathcal{O}(n + |\mathcal{M}_x|)$ time [23]. There also exist space-efficient data structures based on the Burrows-Wheeler transform of the input that can be applied for this computation [6, 5]. In the worst case, the number of minimal absent words of x is $\Theta(\sigma n)$ and we would thus need $\Omega(\sigma n)$ space to represent them explicitly.

3.1 The Data Structure

We next present an alphabet-independent data structure that stores information related to extended special factors. It allows for *counting* and *reporting* minimal absent word queries in optimal time. Specifically, we show the following result.

Theorem 6. *Given a word x of length n over an integer alphabet, we can construct in $\mathcal{O}(n)$ time an $\mathcal{O}(n)$-sized data structure that outputs, for a given on-line query ℓ, $\mathcal{M}_x^\ell$ in $\mathcal{O}(1+|\mathcal{M}_x^\ell|)$ time or $|\mathcal{M}_x^\ell|$ in $\mathcal{O}(1)$ time.*

Let us start with a simple but crucial lemma. It unveils the connection between extended special factors and minimal absent words (see also [4], Sect. 2).

Lemma 7. *Given a minimal absent word awb of x, where $a, b \in \Sigma$ and $w \in \Sigma^*$, either (i) w occurs as an infix of x and any word cwd, $c, d \in \Sigma$, that occurs in x is an extended left- or right-special factor of x; or (ii) wb is a prefix of x, aw is a suffix of x and w occurs only twice in x.*

Proof. If w occurs as infix of x at position i, then $x[i-1\,..\,i+|w|] \neq awb$ and since aw and wb occur in x, $x[i-1\,..\,i+|w|]$ is an extended left- or right-special factor; this is case (i). If w does not occur as infix in x, we are at case (ii). □

Proposition 8 ([22]). *In a word x of length n there is at most one minimal absent word awb of type (ii) (Lemma 7) and we can compute it in $\mathcal{O}(n)$ time.*

Proof. The word w must be a border of x that does not occur elsewhere in x[3]; this can only be the longest border u of x: any other border of x is also a border of u [22]. We locate u and check if it has another occurrence in x in $\mathcal{O}(n)$ time [14], thus retrieving this minimal absent word of type (ii), if there is one. □

Main Idea. For each word w that is the longest infix of a minimal absent word, we compute the letters that precede it in x, the ones that succeed it and the pairs of letters (a, b) such that awb occurs in x. The total size of these sets is $\mathcal{O}(n)$ by Lemmas 1 and 7. If the minimal absent words with longest infix w are no more than twice the number of factors of the form awb of x we pre-compute them in $\mathcal{O}(n)$ time in total; otherwise we off-load the computation to the query.

Construction. Since word x is stored in internal memory, in what follows, we assume a constant-sized representation of arbitrary-length factors and minimal absent words of x. We first compute all extended left- and right-special factors in $\mathcal{O}(n)$ time using Theorem 3. We form their union U, assign their longest infix w as their representative, group the elements of U based on their representatives' length, and sort them lexicographically based on the representatives in each group. The sorting can be done in $\mathcal{O}(n)$ time, for all groups together, using standard tools that exploit longest common prefix information [14] and radix sort. We then identify in $\mathcal{O}(n)$ time the prefixes P_x (resp. suffixes S_x) of x of

[3] In this case, x is called *closed*. Such words are an object of combinatorial interest [21].

the form wb (resp. aw), where $cwd \in U$, $a, b, c, d \in \Sigma$ and $w \in \Sigma^*$, that do not occur elsewhere in x. This can also be implemented using longest common prefix information [14]. We assign the longest proper prefix (resp. suffix) of each element of P_x (resp. S_x) as its representative.

We then group the elements of $V = U \cup S_x \cup P_x$ based on their representatives' length and store them in each such group based on the representatives. We do this by inferring the representatives' lexicographical order in $\mathcal{O}(n)$ time by using the same tools. The size of V is $\mathcal{O}(n)$ by Lemma 1.

For each such representative w, we construct the following sets:

- $B(w) = \{(\alpha, \beta) | \alpha, \beta \in \Sigma \cup \{\varepsilon\}$ and $\alpha w \beta \in V$ with representative $w\}$;
- $B'(w) = \{(\alpha, \beta) | (\alpha, \beta) \in B(w), \alpha \neq \varepsilon, \beta \neq \varepsilon\}$;
- $L(w) = \{\alpha | \alpha \in \Sigma, (\alpha, \beta) \in B(w)\}$ and $R(w) = \{\beta | \beta \in \Sigma, (\alpha, \beta) \in B(w)\}$.

We also construct these sets for the single minimal absent word of type (ii) if there is one. By definition, the minimal absent words whose longest infix is w are the ones of the form $\alpha w \beta$, where $\alpha \in L(w), \beta \in R(w)$, and $(\alpha, \beta) \notin B'(w)$. We lexicographically sort the elements in $L(w)$, $R(w)$ and $B'(w)$, for all w together, in $\mathcal{O}(n)$ time using radix sort. Then if

$$|B'(w)| \geq |L(w)| \cdot |R(w)| - |B'(w)| \iff |L(w)| \cdot |R(w)| \leq 2|B'(w)|,$$

we pre-compute all minimal absent words with longest infix w in $\mathcal{O}(|B'(w)|)$ time by generating all possible awb, $a \in L(w), b \in R(w)$ in lexicographical order, filtering out awb such that $(a, b) \in B'(w)$ by scanning $B'(w)$ at the same time. We store these words in the linked list $\Lambda_1(|w|)$.

Otherwise, if $|L(w)| \cdot |R(w)| > 2|B'(w)|$, we store $L(w), R(w)$ and $B'(w)$ as an element in the linked list $\Lambda_2(|w|)$. This requires $\mathcal{O}(n)$ time in total by Lemma 1; and the total size of Λ_1 and Λ_2 is $\mathcal{O}(n)$.

By definition, the number of minimal absent words whose longest infix is w is $|L(w)| \cdot |R(w)| - |B'(w)|$. We can thus maintain this information per length in an integer array C initialised to zeros, by adding this number to $C[|w|]$, for all representatives w. This requires $\mathcal{O}(n)$ time in total and the array is of size $\mathcal{O}(n)$.

Querying. For a *reporting* on-line query ℓ, we can output $\mathcal{M}_x^\ell$ in $\mathcal{O}(1+|\mathcal{M}_x^\ell|)$ time as follows. We locate the elements in V with representatives of length $\ell - 2$. For the representatives for which we have already pre-computed the minimal absent words we output them from $\Lambda_1(\ell - 2)$; for the rest, we perform the computation described above for each w based on the sets $L(w), R(w)$ and $B'(w)$, which are stored in $\Lambda_2(\ell - 2)$. For a *counting* on-line query ℓ, we output $|\mathcal{M}_x^\ell| = C[\ell - 2]$.

Lemma 7 guarantees the correctness of the algorithm and we thus arrive at Theorem 6. If we apply Theorem 6 for pre-processing and then query for $\ell = 2, \ldots, n + 1$, we obtain the respective result of [23], which is based on constructing the directed acyclic word graph for x [7, 13] and on refining the algorithm of [16].

Corollary 9 ([23]). *Given a word x of length n over an integer alphabet, $\mathcal{M}_x$ can be computed in the optimal $\mathcal{O}(n + |\mathcal{M}_x|)$ time.*

3.2 Sequence Comparison

In [11], the authors introduced a measure of similarity between two words x and y based on the notion of minimal absent words. Let $\mathcal{M}_x^\ell$ (resp. $\mathcal{M}_y^\ell$) denote the set of minimal absent words of length at most ℓ of x (resp. y). The authors made use of a length-weighted index to provide a measure of similarity between x and y, using their sample sets $\mathcal{M}_x^\ell$ and $\mathcal{M}_y^\ell$, by considering the length of each member in the symmetric difference $\mathcal{M}_x^\ell \triangle \mathcal{M}_y^\ell$ of the sample sets. In [12] the authors considered a more general measure of similarity for two words x and y. It is based on the set $\mathcal{M}_x \triangle \mathcal{M}_y$, and is defined by

$$\mathsf{LW}(x,y) = \sum_{w \in \mathcal{M}_x \triangle \mathcal{M}_y} \frac{1}{|w|^2},$$

so without any restriction on the lengths of minimal absent words. The smaller the value of $\mathsf{LW}(x,y)$, the more similar we assume x and y to be; in fact, $\mathsf{LW}(x,y)$ is a metric on Σ^* [12]. Note that $\mathsf{LW}(x,y)$ is affected by both the cardinality of $\mathcal{M}_x \triangle \mathcal{M}_y$ and the lengths of its elements; longer words in $\mathcal{M}_x \triangle \mathcal{M}_y$ contribute less in the value of $\mathsf{LW}(x,y)$ than shorter ones. Hence, intuitively, the shorter the words in $\mathcal{M}_x \triangle \mathcal{M}_y$, the more dissimilar x and y are.

One of the main results of [12] is that $\mathsf{LW}(x,y)$ can be computed in $\mathcal{O}(\sigma(|x|+|y|))$ time. In what follows, we improve this result for integer alphabets by avoiding to compute the minimal absent words explicitly. We rather exploit the connection between minimal absent words and extended special factors, and thus remove the dependency on the alphabet size — a somewhat surprising result.

Theorem 10. *Given two words x and y over an integer alphabet, $\mathsf{LW}(x,y)$ can be computed in the optimal $\mathcal{O}(|x|+|y|)$ time.*

Proof. It suffices to compute the size of the set $\mathcal{M}_x^\ell \triangle \mathcal{M}_y^\ell$, for all $2 \le \ell \le n+1$. We will do that by computing the number of words $awb \in \mathcal{M}_x^\ell \triangle \mathcal{M}_y^\ell$, $a, b \in \Sigma$ for each w that is the longest infix of some minimal absent word of x or of y.

Let us denote by $\mathcal{M}_{z,w}$ the minimal absent words of z whose longest infix is w. By definition we have that

$$\mathcal{M}_{x,w} \triangle \mathcal{M}_{y,w} = (\mathcal{M}_{x,w} \cup \mathcal{M}_{y,w}) \setminus (\mathcal{M}_{x,w} \cap \mathcal{M}_{y,w}).$$

This implies

$$|\mathcal{M}_{x,w} \triangle \mathcal{M}_{y,w}| = |\mathcal{M}_{x,w} \cup \mathcal{M}_{y,w}| - |\mathcal{M}_{x,w} \cap \mathcal{M}_{y,w}| = \\ |\mathcal{M}_{x,w}| + |\mathcal{M}_{y,w}| - 2|\mathcal{M}_{x,w} \cap \mathcal{M}_{y,w}|.$$

We further denote the sets $L(w)$, $R(w)$, $B'(w)$ for word z by $L_z(w)$, $R_z(w)$, $B'_z(w)$. By the definition of minimal absent words, we have that

$$|\mathcal{M}_{x,w}| = |L_x(w)| \cdot |R_x(w)| - |B'_x(w)|.$$

This can be computed, for all w, in $\mathcal{O}(|x|)$ time by applying the data structure of Theorem 6. We obtain $|\mathcal{M}_{y,w}|$ analogously. We thus only need to compute:

$$|\mathcal{M}_{x,w} \cap \mathcal{M}_{y,w}| =$$
$$|\{(a,b)|(a,b) \in (L_x(w) \times R_x(w)) \cap (L_y(w) \times R_y(w)), (a,b) \notin B'_x(w) \cup B'_y(w)\}| =$$
$$|\{(a,b)|(a,b) \in (L_x(w) \cap L_y(w)) \times (R_x(w) \cap R_y(w)), (a,b) \notin B'_x(w) \cup B'_y(w)\}|.$$

The quantities $|(L_x(w) \cap L_y(w)) \times (R_x(w) \cap R_y(w))|$ can be computed in $\mathcal{O}(|x| + |y|)$ time, for all w, since we store the elements of the sets sorted. We can then check for each $(\alpha, \beta) \in B'_x(w) \cup B'_y(w)$ whether it occurs in $(L_x(w) \cap L_y(w)) \times (R_x(w) \cap R_y(w))$, for all w, within the same complexity as follows. Since all our sets are sorted, we can check whether $\alpha \in L_x(w) \cap L_y(w)$ in time linear in the total size of $B'_x(w)$, $B'_y(w)$, $L_x(w)$ and $L_y(w)$, for all pairs (α, β); if so, we keep (α, β). After we do this for all w, we (globally) sort the surviving pairs based on their second element — using integer identifiers for representatives w so that we can regroup them — and conclude in an analogous manner as before by employing $R_x(w) \cap R_y(w)$. The result then follows from Theorem 3. □

Acknowledgements

The authors would like to acknowledge the financial support towards travel and subsistence from the Laboratoire d'Informatique Gaspard-Monge at the Université Paris-Est, where part of this work has been conducted.

References

1. Y. Almirantis, P. Charalampopoulos, J. Gao, C. S. Iliopoulos, M. Mohamed, S. P. Pissis, and D. Polychronopoulos. On avoided words, absent words, and their application to biological sequence analysis. *Algorithms for Molecular Biology*, 12(1):5:1–5:12, 2017.
2. Y. Almirantis, P. Charalampopoulos, J. Gao, C. S. Iliopoulos, M. Mohamed, S. P. Pissis, and D. Polychronopoulos. Optimal Computation of Overabundant Words. In R. Schwartz and K. Reinert, editors, *17th International Workshop on Algorithms in Bioinformatics (WABI 2017)*, volume 88 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
3. C. Barton, A. Héliou, L. Mouchard, and S. P. Pissis. Linear-time computation of minimal absent words using suffix array. *BMC Bioinformatics*, 15:388, 2014.
4. M. Béal, F. Mignosi, and A. Restivo. Minimal forbidden words and symbolic dynamics. In C. Puech and R. Reischuk, editors, *STACS 96, 13th Annual Symposium on Theoretical Aspects of Computer Science, Grenoble, France, February 22-24, 1996, Proceedings*, volume 1046 of *Lecture Notes in Computer Science*, pages 555–566. Springer, 1996.
5. D. Belazzougui and F. Cunial. A framework for space-efficient string kernels. *Algorithmica*, 79(3):857–883, 2017.

6. D. Belazzougui, F. Cunial, J. Kärkkäinen, and V. Mäkinen. Versatile succinct representations of the bidirectional Burrows-Wheeler transform. In H. L. Bodlaender and G. F. Italiano, editors, *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, volume 8125 of *Lecture Notes in Computer Science*, pages 133–144. Springer, 2013.
7. A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht, M. T. Chen, and J. I. Seiferas. The smallest automaton recognizing the subwords of a text. *Theor. Comput. Sci.*, 40:31–55, 1985.
8. A. Carpi and A. de Luca. Special factors, periodicity, and an application to Sturmian words. *Acta Inf.*, 36(12):983–1006, 2000.
9. A. Carpi and A. de Luca. Words and special factors. *Theor. Comput. Sci.*, 259(1-2):145–182, 2001.
10. J. Cassaigne, G. Fici, M. Sciortino, and L. Q. Zamboni. Cyclic complexity of words. *J. Comb. Theory, Ser. A*, 145:36–56, 2017.
11. S. Chairungsee and M. Crochemore. Using minimal absent words to build phylogeny. *Theoretical Computer Science*, 450:109–116, 2012.
12. P. Charalampopoulos, M. Crochemore, G. Fici, R. Mercaş, and S. P. Pissis. Alignment-free sequence comparison using absent words. *Information and Computation*, 2018. In press.
13. M. Crochemore. Transducers and repetitions. *Theor. Comput. Sci.*, 45(1):63–86, 1986.
14. M. Crochemore, C. Hancart, and T. Lecroq. *Algorithms on Strings*. Cambridge University Press, New York, NY, USA, 2007.
15. M. Crochemore, A. Héliou, G. Kucherov, L. Mouchard, S. P. Pissis, and Y. Ramusat. Minimal absent words in a sliding window and applications to on-line pattern matching. In R. Klasing and M. Zeitoun, editors, *Fundamentals of Computation Theory - 21st International Symposium, FCT 2017, Bordeaux, France, September 11-13, 2017, Proceedings*, volume 10472 of *Lecture Notes in Computer Science*, pages 164–176. Springer, 2017.
16. M. Crochemore, F. Mignosi, and A. Restivo. Automata and forbidden words. *Inf. Process. Lett.*, 67(3):111–117, 1998.
17. M. Crochemore and G. Navarro. Improved antidictionary based compression. In *22nd International Conference of the Chilean Computer Science Society (SCCC 2002), 6-8 November 2002, Copiapo, Chile*, pages 7–13. IEEE Computer Society, 2002.
18. A. de Luca and L. Mione. On bispecial factors of the Thue-Morse word. *Inf. Process. Lett.*, 49(4):179–183, 1994.
19. A. de Luca and S. Varricchio. On the factors of the Thue-Morse word on three symbols. *Inf. Process. Lett.*, 27(6):281–285, 1988.
20. A. de Luca and S. Varricchio. Some combinatorial properties of the Thue-Morse sequence and a problem in semigroups. *Theor. Comput. Sci.*, 63(3):333–348, 1989.
21. G. Fici. Open and closed words. *Bulletin of the EATCS*, 123, 2017.
22. G. Fici, F. Mignosi, A. Restivo, and M. Sciortino. Word assembly through minimal forbidden words. *Theor. Comput. Sci.*, 359(1-3):214–230, 2006.
23. Y. Fujishige, Y. Tsujimaru, S. Inenaga, H. Bannai, and M. Takeda. Computing DAWGs and minimal absent words in linear time for integer alphabets. In P. Faliszewski, A. Muscholl, and R. Niedermeier, editors, *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*, volume 58 of *LIPIcs*, pages 38:1–38:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

24. R. M. Silva, D. Pratas, L. Castro, A. J. Pinho, and P. J. S. G. Ferreira. Three minimal sequences found in ebola virus genomes and absent from human DNA. *Bioinformatics*, 31(15):2421–2425, 2015.