

Smallest Suffixient Sets: Effectiveness, Resilience, and Calculation

Hiroto Fujimaru^{a,*}, Gonzalo Navarro^{b,c,*}, Giuseppe Romana^{d,*}, Cristian Urbina^{c,e,*}

^a*Department of Information Science and Technology, Kyushu University, Fukuoka, Japan*

^b*Department of Computer Science, University of Chile, Santiago, Chile*

^c*Center for Biotechnology and Bioengineering (CeBiB), Santiago, Chile*

^d*Department of Mathematics and Computer Science, University of
Palermo, Palermo, Italy*

^e*Faculty of Mathematics, Informatics and Mechanics, University of
Warsaw, Warsaw, Poland*

Abstract

A suffixient set is a novel combinatorial object that captures the essential information of repetitive strings in a way that, provided with a random access mechanism, supports various forms of pattern matching. In this paper, we study the size χ of the smallest suffixient set as a repetitiveness measure.

First, we study its sensitivity to various string operations. We show that χ cannot increase by more than 2 after appending or prepending a character to the string. As a consequence, we are able to give simple linear-time online algorithms to compute smallest suffixient sets. We also show that, although reversing the string can increase χ by an arbitrary $O(n)$ value, it always holds $\chi(T)/\chi(T^R) \leq 2$. We also prove lower and upper bounds for the additive or multiplicative increase of χ after applying arbitrary edit operations, or rotating the text. In particular, we show that the additive increase can be as large as $\Omega(\sqrt{n})$ for all those operations.

Secondly, we place χ among known repetitiveness measures. In particular, we show $\chi \leq 2r$ (where r is the number of runs in the Burrows-Wheeler Transform of the string), that there are string families where $\chi = o(v)$ (where

*Corresponding author

Email addresses: fujimaru.hiroto.134@s.kyushu-u.ac.jp (Hiroto Fujimaru), gnavarro@dcc.uchile.cl (Gonzalo Navarro), giuseppe.romana01@unipa.it (Giuseppe Romana), c.urbina-gallegos@uw.edu.pl (Cristian Urbina)

v is the size of the smallest lexicographic parse of the string), and that χ is uncomparable to almost all reachable measures based on copy-paste mechanisms. In passing, we give precise bounds for χ for some relevant string families, for example $\chi \leq \sigma + 2$ on episturmian words over alphabets of size σ (e.g., $\chi \leq 4$ on Fibonacci strings, for which we precisely characterize the only two smallest suffixient sets).

Keywords: Repetitive sequences, Burrows-Wheeler transform, Suffixient sets, Text compressibility

1. Introduction

The study of repetitive string collections has recently attracted considerable interest from the stringology community, triggered by practical challenges such as representing huge collections of similar strings in a way that they can be searched and mined directly in highly compressed form [1, 2]. An example is the *European '1+ Million Genomes' Initiative*¹, which aims at sequencing over a million human genomes: while this data requires around 750TB of storage in raw form (using 2 bits per base), the high similarity between human genomes would allow storing it in queriable form using two orders of magnitude less space.

An important aspect of this research is to understand how to measure repetitiveness, especially when those measures reflect the size of compressed representations that offer different access and search functionalities on the collection. Various repetitiveness measures have been proposed, from abstract lower bounds to those related to specific text compressors and indices; a relatively up-to-date survey is maintained [3]. Understanding how these measures relate to each other sheds light on what search functionality is obtained at what space cost.

A relevant measure recently proposed is the size χ of the smallest *suffixient set* of the text collection [4], whose precise definition will be given later. Within $O(\chi)$ size, plus a random-access mechanism on the string, it is possible to support some text search functionalities, such as finding one occurrence of a pattern, or finding its maximal exact matches (MEMs), which is of central use on various bioinformatic applications [5].

¹<https://digital-strategy.ec.europa.eu/en/policies/1-million-genomes>

While there has been some work already on how to build minimal suffixient sets and how to index and search a string within their size, less is known about that size, χ , as a measure of repetitiveness. It is only known [4] that $\gamma \leq \chi \leq 2\bar{r}$ on every string family, where γ is the size of the smallest *string attractor* of the collection (a measure that lower bounds most repetitiveness measures) [6] and $\bar{r}$ is the number of equal-letter runs of the Burrows-Wheeler Transform (BWT) [7] of the reversed string. Very recently, it has been shown that $s = O(\chi)$, where s is the size of the smallest Substring Equation System (SES) that uniquely describes the string [8].

In this paper we better characterize χ as a repetitiveness measure. First, we study how it behaves when the string (of length n) undergoes updates, showing in particular that it grows by $O(1)$ when appending or prepending symbols, but that it can grow additively by $\Omega(\sqrt{n})$ upon arbitrary edit operations or rotations, and by $\Omega(n)$ when reversing the string. However, for this last operation, we show that χ cannot grow by a factor larger than 2. Second, we show that $\chi \leq 2r$ on every string family, where r is the number of equal-letter runs of the BWT of the string. We also show that there are string families where $\chi = o(v)$, where v is the size of the smallest lexicographic parse [9] (an alternative to the size of the Lempel-Ziv parse [10], which behaves similarly). In particular, this holds on the Fibonacci strings, where we fully characterize the only 2 smallest suffixient sets of size 4, and further prove that $\chi \leq \sigma + 2$ on all substrings of episturmian words over an alphabet of size σ . Since $v = O(r)$ on all string families, this settles χ as a strictly smaller measure than r , which is a more natural characterization than in terms of the reverse string. We also show that χ is incomparable with most “copy-paste” based measures [1], as there are families where it is strictly smaller and others where it is strictly larger than any of those measures.

This result relates to the important question of whether a measure μ is *reachable* (i.e., one can represent the string within $O(\mu)$ space), *accessible* (i.e., one can access any string position from an $O(\mu)$ -size representation, in $o(n)$ time), or *searchable* (i.e., one can search for patterns in $o(n)$ time within space $O(\mu)$). Measure r is, curiously, the only one to date being reachable and searchable, but not known to be accessible. Now χ emerges as a measure smaller than r , which can search if provided with a mechanism to efficiently access substrings (r does not need access to support searches). It has been recently shown that χ is reachable, since $s = O(\chi)$ is reachable [8], but its accessibility status remains unknown.

Our final contribution are new, extremely simple, *online* algorithms to compute smallest suffixient sets (and thus χ), scanning the text left to right or right to left. While there already exist efficient algorithms to do this [11], our new algorithm can, at any point, exhibit a smallest suffixient set for the string it has just consumed. Just as online suffix tree constructions [12, 13], our algorithm uses $O(n)$ space and worst-case time in the transdichotomous RAM model over polynomial-size integer alphabets. The best previous algorithm obtains the same result [11], but it is not online and starts from the suffix array and other components of the suffix tree, whereas ours starts from the text and builds the suffix tree at the same time. All linear-time suffix tree construction algorithms run under the same model of computation.

A preliminary version of this paper appeared in the proceedings of the conference SPIRE 2025 [14]. In this extended version, we improved several of our previous results, establish new ones, and present the new online construction algorithms. We also show extended proofs, new figures, and more.

2. Preliminaries

An *ordered alphabet* $\Sigma = \{a_1, \dots, a_\sigma\}$ is a finite set of symbols equipped with a total order $<$ such that $a_1 < a_2 < \dots < a_\sigma$. When $\sigma = 2$, we assume $\Sigma = \{\mathbf{a}, \mathbf{b}\}$ with $\mathbf{a} < \mathbf{b}$. The special symbol $\$$, if it appears, is always assumed to be the smallest of the alphabet.

A *string* $w[1..n]$ (or simply w if it is clear from the context) of *length* $|w| = n$ over the alphabet Σ is a sequence $w[1]w[2]\dots w[n]$ of symbols where $w[i] \in \Sigma$ for all $i \in [1, n]$. The *empty string* of length 0 is denoted by ϵ . We denote by Σ^* the set of all strings over Σ . Additionally, we let $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$ and $\Sigma^k = \{w \in \Sigma^* \mid |w| = k\}$. We denote by $w[i..j]$ the substring $w[i]w[i+1]\dots w[j]$. If $x = x[1..n]$ and $y = y[1..m]$ are strings, we define the *concatenation operation* applied on x and y , as the string obtained by juxtaposing these two strings, that is, $x \cdot y = x[1]x[2]\dots x[n]y[1]\dots y[m] = xy$. A string x is a *substring* of w if $w = yxz$ for some $y, z \in \Sigma^*$. A string x is a *prefix* of w if $w = xy$ for some $y \in \Sigma^*$. Analogously, x is a *suffix* of w if $w = yx$ for some $y \in \Sigma^*$. We say that substrings, prefixes, and suffixes are *non-trivial* if they are different from w and ϵ . The set of substrings of w is denoted by $\mathcal{F}_w$. We also let $\mathcal{F}_w(k) = \mathcal{F}_w \cap \Sigma^k$. The *reverse* of a finite string w is the string $w^R = w[n] \cdot w[n-1] \dots w[1]$. We denote by $\mathcal{R}(w)$ the multiset of rotations of $w[1..n]$, that is, $\mathcal{R}(w) = \{w[i+1..n]w[1..i] \mid i \in [1..n]\}$. The *Burrows-Wheeler transform* (BWT) of a string w , denoted $\text{BWT}(w)$, is

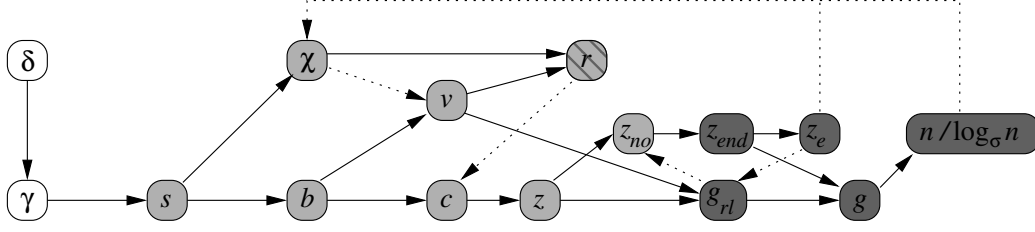


Figure 1: Relations between relevant repetitiveness measures and how our results place χ among them. An arrow $\mu_1 \rightarrow \mu_2$ means that $\mu_1 = O(\mu_2)$ for all strings and, save for $s \rightarrow b$, $c \rightarrow z$, $z_{no} \rightarrow z_{end}$, and $z_{end} \rightarrow z_e$, a string family where $\mu_1 = o(\mu_2)$ is known. The dotted arrows mark only this last condition, so they are not transitive. Measures in light gray nodes are known to be reachable; those in dark gray are accessible and searchable; and r is hatched because it is searchable but not known to be accessible.

the transformation of w obtained by collecting the last symbol of all rotations in $\mathcal{R}(w)$ in lexicographic order. The *BWT matrix* $B(w)$ of w is the $(n \times n)$ -matrix where the i -th row is the i -th rotation of w in lexicographic order.

A *right-infinite string* $\mathbf{w}$ —we use **boldface** to emphasize its infinite length— over Σ is any infinite sequence $\mathbb{Z}^+ \rightarrow \Sigma$. The set of all infinite strings over Σ is denoted Σ^ω . A substring of $\mathbf{w}$ is the finite string $\mathbf{w}[i..j]$ for any $1 \leq i \leq j$. A prefix of $\mathbf{w}$ is a finite substring of the form $\mathbf{w}[1..n]$ for some $n \geq 0$. The *substring complexity function* $P_{\mathbf{w}}(k) : \mathbb{Z}^+ \cup \{0\} \rightarrow \mathbb{Z}^+$ counts the number of distinct substrings of length k in $\mathbf{w}$, for any $k \in \mathbb{Z}^+ \cup \{0\}$, that is, $P_{\mathbf{w}}(k) = |\mathcal{F}_{\mathbf{w}}(k)|$. For a finite string $w[1..n]$, the domain of P_w is restricted to $[0..n]$.

2.1. Measures of repetitiveness

In this work we will relate χ , in asymptotic terms, with several well-established measures of repetitiveness [1, 3]: $\delta = \max_{k \in [0..n]} (\mathcal{F}_w(k)/k)$ (a measure of substring complexity), γ (the smallest string attractor), s (the smallest Substring Equation System), b (the size of the smallest bidirectional macro scheme), z (the size of the Lempel-Ziv parse), z_{no} (the same without allowing phrases to overlap their sources), z_e (the size of the greedy LZ-End parse), z_{end} (the size of the minimal LZ-End parse), v (the size of the smallest lexicographic parse), r (the number of equal-letter runs in the BWT of the string), g (the size of the smallest context-free grammar generating only the string), g_{rl} (the same allowing run-length rules), and c (the size of the smallest collage system generating only the string). Except for δ , γ , and r , these measures are said to be *copy-paste* because they refer to a way of

cutting the sequence into chunks that can be copied from elsewhere in the same sequence. Indeed, δ and γ are lower-bound measures, the former known to be unreachable and the latter not known to date to be reachable; all the others are. The smallest measures known to be accessible (and searchable) are z_{end} and g_{rl} , and r is searchable but not known to be accessible.

The known relations between those measures are summarized in Fig. 1, where we have added the results we obtain in this paper with respect to χ .

2.2. Edit operations and sensitivity functions

The so-called *edit operations* are *insertion*, *substitution* and *deletion* of a single character on a string. We denote $\mathbf{ins}(w)$, $\mathbf{sub}(w)$, and $\mathbf{del}(w)$ the sets of strings that can be obtained by applying an insertion, a substitution, and a deletion to w respectively. In addition, we let $\mathbf{prepend}(w)$ and $\mathbf{append}(w)$ be $\mathbf{ins}(w)$ restricted to the insertion being made at the beginning and the end of the string, respectively, $\mathbf{rot}(w) = \mathcal{R}(w)$, and $\mathbf{rev}(w) = w^R$.

A repetitiveness measure μ is *monotone* or *non-decreasing* to the insertion of a single character if $\mu(w') - \mu(w) \geq 0$ for any w and $w' \in \mathbf{ins}(w)$. More generally, the *additive sensitivity* and *multiplicative sensitivity* functions of a repetitiveness measure μ to the insertion of a single character are the maximum possible values of $\mu(w') - \mu(w)$ and $\mu(w')/\mu(w)$, respectively. Formally, the additive sensitivity of a measure of repetitiveness μ to a string operation ρ is defined as a function $AS_{\mu,\rho} : \mathbb{Z}^+ \rightarrow \mathbb{R}$, where $AS_{\mu,\rho}(n) = \max_{w \in \Sigma^n} \max_{w' \in \rho(w)} \mu(w') - \mu(w)$, that is, the maximum achievable difference among all the strings. Similarly, the multiplicative sensitivity is defined as $MS_{\mu,\rho}(n) = \max_{w \in \Sigma^n} \max_{w' \in \rho(w)} \mu(w')/\mu(w)$.

We define the concept of monotonicity and sensitivity functions for the remaining string operations analogously.

3. Suffixient Sets and the Measure χ

In this section we define the central combinatorial objects and measures we analyse on this work. Note that some of our definitions are slightly different from their original formulation [5, 11], because we do not always assume that all strings are $\$$ -terminated.

Definition 1 (Right-maximal Substrings and Right-extensions [5, 11]).

Let $w \in \Sigma^*$. A substring $x \in \Sigma^*$ of w is *right-maximal* if there exist at least two distinct symbols $a, b \in \Sigma$ such that both xa and xb are substrings of

w . For any right-maximal substring x of w , the substrings xa with $a \in \Sigma$ are called *right-extensions*. We denote the set of right-extensions in w by $E_r(w) = \{xa \mid \exists b : b \neq a, xa \in \mathcal{F}_w, xb \in \mathcal{F}_w\}$.

We distinguish a special class of right-extensions that are not suffixes of any other right-extension.

Definition 2 (Supermaximal Extensions [5, 11]). The set of *supermaximal extensions* of w is $\mathcal{S}_r(w) = \{x \in E_r(w) \mid \forall y \in E_r(w), y = zx \Rightarrow z = \varepsilon\}$. Moreover, we let $\mathbf{sre}(w) = |\mathcal{S}_r(w)|$.

We now define suffixient sets for strings not necessarily $\$$ -terminated; we introduce later the special terminator $\$$.

Definition 3 (Suffixient Set [5, 11]). Let $w[1..n] \in \Sigma^*$. A set $S \subseteq [1..n]$ is a *suffixient set* for w if for every right-extension $x \in E_r(w)$ there exists $j \in S$ such that x is a suffix of $w[1..j]$.

Intuitively, a suffixient set is a collection of positions of $[1..|w|]$ capturing all the right-extensions appearing in w . The smallest suffixient sets, which are suffixient sets of minimum size, have also been characterized in terms of supermaximal right-extensions. The next definition simplifies the original one [5, 11].

Definition 4 (Smallest Suffixient Set). Let $w[1..n] \in \Sigma^*$. A suffixient set $S \subseteq [1..n]$ is a *smallest suffixient set* for w if there is a bijection $\text{pos} : \mathcal{S}_r(w) \rightarrow S$ such that every $x \in \mathcal{S}_r(w)$ is a suffix of $w[1..\text{pos}(x)]$.

In its original formulation, the measure χ is defined over $\$$ -terminated strings. Here, we define $\chi(w)$ with the $\$$ being implicit, not being part of w .

Definition 5 (Measure χ [5, 11]). Let $w \in \Sigma^*$ and assume $\$ \notin \mathcal{F}_w$. Then, $\chi(w) = |\mathcal{S}|$, where $\mathcal{S}$ is a smallest suffixient set for $w\$$.

One can see from the above definitions that χ is well-defined because $\chi(w) = \mathbf{sre}(w\$)$. We will use this relation to prove results on χ via $\mathbf{sre}$.

4. Additive Sensitivity of χ to String Operations

The sensitivity to string operations has been studied for many repetitiveness measures [15, 16, 17, 18, 19, 20, 21, 22]. A robust repetitiveness measure should not change much upon small changes in the sequence. For instance, b , z , and g can increase only by a multiplicative constant after an edit operation [15], and they can increase only by a constant additive factor when prepending or appending a character. On the other hand, r can increase by a $\Theta(\log n)$ factor when appending a character [19, Prop. 37]. Other results have been obtained concerning more complex string operations, like reversing a string [18], or applying a string morphism [16, 17].

In this section we study how **sre** and χ behave in this respect, focusing on additive sensitivity; the next section deals with multiplicative sensitivity. Overall, we obtain the following results on the additive sensitivity of χ ; the proof is given at the end of the section.

Theorem 1. *The following bounds on the additive sensitivity of measure χ to string operations hold:*

1. $AS_{\chi,\rho}(n) = \Theta(1)$ for $\rho \in \{\text{append}, \text{prepend}\}$;
2. $AS_{\chi,\rho}(n) = \Omega(\sqrt{n})$ for $\rho \in \{\text{ins}, \text{del}, \text{sub}, \text{rot}\}$;
3. $AS_{\chi,\text{rev}}(n) = \Theta(n)$.

4.1. Appending and prepending symbols

We first show that **sre**(w) and $\chi(w)$ increase or decrease by only additive constants when adding or removing symbols at the extremes of w . We start by proving the following useful lemma.

Lemma 1. *If $E_r(w_1) \subseteq E_r(w_2)$, then $\text{sre}(w_1) \leq \text{sre}(w_2)$.*

Proof. Let $x, y \in \mathcal{S}_r(w_1)$ with $x \neq y$. Because $x \in E_r(w_2)$, there exists $z \in \mathcal{S}_r(w_2)$ with x a suffix of z . Because y is not a suffix of x and vice versa, y cannot be a suffix of z . Therefore, the map $x \mapsto z$ with $x \in \mathcal{S}_r(w_1)$, $z \in \mathcal{S}_r(w_2)$, and $z = z'x$ for some $z' \in \Sigma^*$ is injective and then $\text{sre}(w_1) \leq \text{sre}(w_2)$. $\square$

We now prove that **sre**(w) grows only by $O(1)$ when prepending or appending characters.

Lemma 2. *Let $w \in \Sigma^*$, and $c \in \Sigma$. It holds $\text{sre}(w) \leq \text{sre}(wc) \leq \text{sre}(w) + 2$.*

Proof. The lower bound follows from Lemma 1. For the upper bound, we analyse the new right-extensions that may arise due to appending c to w . For any fixed suffix xc of wc :

1. if xc appears in w , or if xa does not appear in w for any $a \neq c$, or both, then xc induces no new right-extensions in wc ;
2. if for some $a \neq b$, xa and xb were both substrings of w , and xc was not, then xc is a new right-extension of wc ;
3. if x is always followed by $a \neq c$ in w (hence, xa is not a right-extension of w), then both xa and xc are new right-extensions of wc .

Cases 1 and 2 induce at most one new supermaximal right-extension in total for all possible xc , namely the longest right-extension in wc that is a suffix of wc . For Case 3, consider all the increasing-length suffixes $x_1c, x_2c, \dots, x_tc$ of wc that became right-extensions together with $x_1a, x_2a, \dots, x_ta$. Since the latter form a chain of suffixes of x_ta , we only have one possible new supermaximal right-extension ending with a , namely x_ta . Observe that the chain of suffixes $x_1a, x_2a, \dots, x_ta$ is unique: if the suffix x is always followed by a , any suffix y of x is either right-maximal in w (and y falls within Case 2), or it is always followed by an a (because x is always followed by an a), i.e. $y = x_i$ for some $i \in [1 \dots t]$. $\square$

Lemma 3. *Let $w \in \Sigma^*$ and $c \in \Sigma$. It holds $\mathbf{sre}(w) \leq \mathbf{sre}(cw) \leq \mathbf{sre}(w) + 2$.*

Proof. The lower bound follows from Lemma 1. For the upper bound, let cxa be the shortest prefix of cw that is not a right-extension of w , but is a right-extension of cw (if it exists). This implies that there exists $b \neq a$ such that $cx b \in \mathcal{F}_w$ and $cxa \notin \mathcal{F}_w$ (otherwise, cxa would be a right-extension of w), so no prefix of cw of length $|cxa|$ or more is right-maximal, and thus no prefix longer than cxa can be a right-extension. By the minimality of cxa , all prefixes of cw shorter than cxa either are not right-extensions of cw , or are right-extensions of both w and cw . Therefore, cxa together with some $cx b$ appearing in w , are the only possible new right-extensions in cw with respect to w . $\square$

Lemmas 2 and 3 yield remarkably simple algorithms to compute smallest suffixient sets, which we detail in Section 7.

By letting $c = \$ \notin \mathcal{F}_w$ in Lemma 2, we relate χ to $\mathbf{sre}$ (note that χ is always at least $\mathbf{sre} + 1$ because of the new supermaximal extension ending

with $\$$). This makes clear the relation between Combinatorics on words [23] with suffixient sets, via the common notion of *right-special factors* (which we call here right-maximal substrings).

Corollary 1. *Let $w \in \Sigma^*$. It holds $\mathbf{sre}(w) + 1 \leq \chi(w) \leq \mathbf{sre}(w) + 2$.*

Note that, while the value $\mathbf{sre}(w)$ is non-decreasing after appending a character, this is not the case for the measure χ .

Lemma 4. *The measure χ is not monotone to appending a character.*

Proof. Let $w = \text{abaab}$. It holds $\mathcal{S}_r(w\$) = \{\text{aa}, \text{ab}, \text{ab}\$, \text{aba}\}$ and $\mathcal{S}_r(wa\$) = \mathcal{S}_r(\text{abaaba}\$) = \{\text{ab}, \text{aba}\$, \text{abaa}\}$. Hence, $\chi(w) = 4$ and $\chi(wa) = 3$. $\square$

4.2. Sensitivity to edit operations and rotations

We first prove some asymptotic relations between the additive sensitivities of edit operations and rotations.

Lemma 5. *$AS_{\chi,\rho}(n) = \Theta(AS_{\chi,\text{rot}}(n))$ for $\rho \in \{\text{del}, \text{sub}\}$, and $AS_{\chi,\text{ins}}(n) = O(AS_{\chi,\text{rot}}(n))$.*

Proof. We start proving $AS_{\chi,\rho}(n) = O(AS_{\chi,\text{rot}}(n))$, with $\rho = \text{ins}$. Let $|xy| = n$ and let us insert $c \in \Sigma$ between x and y . Let us also shorten $R(n) = AS_{\chi,\text{rot}}(n)$. Then it holds that

$$\begin{aligned} \mathbf{sre}(xyc) &\leq \mathbf{sre}(yxc) + R(n+1) \leq \mathbf{sre}(yx) + 2 + R(n+1) \\ &\leq \mathbf{sre}(xy) + R(n) + 2 + R(n+1), \end{aligned}$$

where the second inequality stems from Lemma 2. Since $1 \leq R(n) = O(n)$ because $\mathbf{sre} = O(n)$, it follows that $\mathbf{sre}(xyc) = \mathbf{sre}(xy) + O(AS_{\chi,\text{rot}}(n))$.

For $\rho = \text{del}$, we instead delete d from xdy , so we have

$$\mathbf{sre}(xy) \leq \mathbf{sre}(yx) + R(n) \leq \mathbf{sre}(yxd) + R(n) \leq \mathbf{sre}(xdy) + R(n+1) + R(n),$$

where again the second inequality stems from Lemma 2. By combining both equations above, we get a bound for $\rho = \text{sub}$, with $n = |xyc|$:

$$\mathbf{sre}(xyc) \leq \mathbf{sre}(yx) + 2 + R(n) \leq \mathbf{sre}(yxd) + 4 + R(n) \leq \mathbf{sre}(xdy) + 4 + 2R(n).$$

In the other direction, for $AS_{\chi,\text{rot}}(n) = O(AS_{\chi,\text{del}}(n))$, let $\$$ be a symbol not in xy and $D(n) = AS_{\chi,\text{del}}(n)$. Then

$$\mathbf{sre}(yx) \leq \mathbf{sre}(y\$x) + D(n+1) \leq \mathbf{sre}(xy\$) + D(n+1) \leq \mathbf{sre}(xy) + 2 + D(n+1).$$

The second inequality stems from $\$$ being unique in $y\$x$, so no right-maximal string can contain it. Thus, $E_r(y\$x) = E_r(y\$) \cup E_r(x) \cup U \subseteq E_r(xy\$)$, where $U = \{ua, ub \mid a \neq b \wedge ua \in \mathcal{F}_x \wedge ub \in \mathcal{F}_{y\$}\}$. By Lemma 1, this implies $\mathbf{sre}(y\$x) \leq \mathbf{sre}(xy\$)$. For substitutions, let $S(n) = AS_{\chi, \text{sub}}(n)$, and consider strings x and yc , with $c \in \Sigma$ and $n = |ycx|$:

$$\begin{aligned} \mathbf{sre}(ycx) &\leq \mathbf{sre}(y\$x) + S(n) \leq \mathbf{sre}(xy\$) + S(n) \\ &\leq \mathbf{sre}(xy) + 2 + S(n) \leq \mathbf{sre}(xyc) + 2 + S(n). \end{aligned} \quad \square$$

We now show that $\mathbf{sre}$ can grow by $\Omega(\sqrt{n})$ upon arbitrary edits and rotations.

Lemma 6. *Let $w_m = \mathbf{ab}^{2m}\mathbf{ab}^{2m+2} \prod_{k=1}^{m-1} u_k$, where $u_k = (\mathbf{ab}^k\mathbf{ab}^{2m-k})^2$ for some $m \geq 3$. Then, $\mathbf{sre}(w_m) = 6m + 4$ holds.*

Proof. Let us consider the prefix $v = \mathbf{ab}^{2m}\mathbf{ab}^{2m+2}$ of w . The substrings $\mathbf{ab}^{2m}$ and $\mathbf{b}^{2m+1}$ appear twice in v , once followed by $\mathbf{a}$ and once followed by $\mathbf{b}$. For each $k \in [1, m-1]$: the substring $\mathbf{b}^k\mathbf{ab}^{2m-k}$ appears three times, once in v , and twice in u_k (note that one occurrence of $\mathbf{b}^{m-1}\mathbf{ab}^{m+1}$ is a suffix of w_m); the substring $\mathbf{b}^{2m-k+1}\mathbf{ab}^k$ appears once in v and twice across $u_k u_{k+1}$; the substring $\mathbf{b}^{2m-k}\mathbf{ab}^k\mathbf{ab}^{2m-k}\mathbf{ab}^k$ appears twice across $u_{k-1}u_k u_{k+1}$.

By carefully analyzing the right-extensions of these factors, one can verify that the following substrings are supermaximal right-extensions of w_m :

1. $\mathbf{ab}^{2m}\mathbf{a}$ and $\mathbf{ab}^{2m+1}$,
2. $\mathbf{b}^k\mathbf{ab}^{2m-k}\mathbf{a}$ and $\mathbf{b}^k\mathbf{ab}^{2m-k+1}$ (for $k \in [1, m-1]$),
3. $\mathbf{b}^{2m-k+1}\mathbf{ab}^k\mathbf{a}$ and $\mathbf{b}^{2m-k+1}\mathbf{ab}^{k+1}$ (for $k \in [1, m-1]$),
4. $\mathbf{b}^{2m-k}\mathbf{ab}^k\mathbf{ab}^{2m-k}\mathbf{ab}^k\mathbf{a}$ and $\mathbf{b}^{2m-k}\mathbf{ab}^k\mathbf{ab}^{2m-k}\mathbf{ab}^{k+1}$ (for $k \in [1, m-2]$),
5. $\mathbf{b}^{2m+1}\mathbf{a}$ and $\mathbf{b}^{2m+2}$.

We now prove that there are no other super-maximal right-extensions in w . Let us consider the other right-extensions (also see Figure 2). For any other right-extensions x ending in $\mathbf{a}$, we have that it is a suffix of one of the supermaximal right-extensions, since every occurrence of $\mathbf{a}$, except the first and last $\mathbf{a}$, serves as the ending position of a supermaximal right-extension listed above; the longest right-extensions ending in correspondence of the first and last $\mathbf{a}$ are $\mathbf{a}$ and $\mathbf{b}^{m+1}\mathbf{ab}^{m-1}\mathbf{a}$ respectively, and both are suffixes of $\mathbf{b}^{m+2}\mathbf{ab}^{m-1}\mathbf{a}$ (case 3. with $k = m-1$). For any other right-extension ending in $\mathbf{b}$: by construction, any substring including $\mathbf{a}$ at least twice is not a

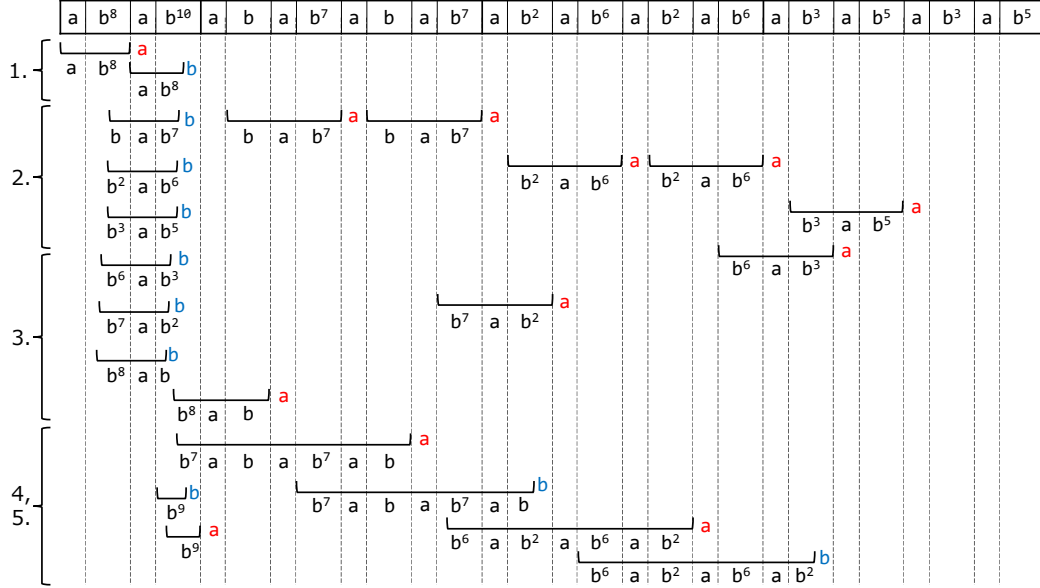


Figure 2: The supermaximal right-extension of the string w_4 shown in Example 1. Note that the ending positions of those ending within b^{10} are all different.

right-extension, with the exception of $b^{2m-k}ab^k ab^{2m-k}ab^{k+1}$ for $k \in [1, m-2]$ (included in case 4.); all right-extensions containing only one a have the form $b^i ab^j$, for any $i \geq 0, j \geq 1$ such that $i + j \leq 2m + 1$ and $j \neq m$, and each is a suffix of one of the supermaximal right-extensions included in either cases 2. or 3.; finally, the right-extensions that do not contain a 's consist of runs of b 's, which are all suffixes of b^{2m+2} (included in 5.). Therefore, there are no other supermaximal extensions, and $\text{sre}(w_m) = 6m - 4$. $\square$

Example 1. Let $w_4 = ab^8ab^{10}abab^7abab^7ab^2ab^6ab^2ab^6ab^3ab^5ab^3ab^5$ (also see Figure 2). It can be verified that the supermaximal right-extensions of w_4 are:

1. ab^8a and ab^9 ,
2. bab^7a and bab^8 ; b^2ab^6a and b^2ab^7 ; b^3ab^5a and b^3ab^6 ,
3. b^8aba and b^8ab^2 ; b^7ab^2a and b^7ab^3 ; b^6ab^3a and b^6ab^4 ,
4. b^7abab^7aba and $b^7abab^7ab^2$; $b^6ab^2ab^6ab^2a$ and $b^6ab^2ab^6ab^3$,
5. b^9a and b^{10} .

One can see that $\text{sre}(w_4) = 20$, as stated in Lemma 6.

We next provide the lower-bound families for edit operations and rotations. When convenient, we describe the witness string via the reverse transformation; the labels (Ins), (Del), (Sub), and (Rot) refer to the operations whose sensitivity is being witnessed.

Lemma 7. *Let $w_m = \mathbf{ab}^{2m} \mathbf{ab}^{2m+2} \prod_{k=1}^{m-1} u_k$, where $u_k = (\mathbf{ab}^k \mathbf{ab}^{2m-k})^2$ for some $m \geq 3$. It holds:*

- (Ins) *If $w'_m = \mathbf{ab}^{4m+2} \prod_{k=1}^{m-1} u_k$, then $\mathbf{sre}(w'_m) = 4m - 2$;*
- (Sub) *If $w'_m = \mathbf{ab}^{4m+3} \prod_{k=1}^{m-1} u_k$, then $\mathbf{sre}(w'_m) = 4m - 2$;*
- (Del) *If $w'_m = \mathbf{ab}^{2m} \mathbf{a} \mathbf{ab}^{2m+2} \prod_{k=1}^{m-1} u_k$, then $\mathbf{sre}(w'_m) = 4m + 2$;*
- (Rot) *If $w'_m = \mathbf{ab}^{2m+2} \prod_{k=1}^{m-1} u_k \cdot \mathbf{ab}^{2m}$, then $\mathbf{sre}(w'_m) = 4m - 2$.*

Proof. We prove the claims separately.

Insertions. Let us consider the string w'_m obtained by removing the second $\mathbf{a}$ from w_m . Then, the prefix $v = \mathbf{ab}^{2m} \mathbf{ab}^{2m+2}$ of w_m is changed to $v' = \mathbf{ab}^{4m+2}$. The substring $\mathbf{b}^{2m-k+1} \mathbf{ab}^k$ occurs only once for each $k \in [1..m-1]$, and any substring $\mathbf{b}^i \mathbf{ab}^j$ of w_m with $i+j > 2m$ does not occur in w'_m . Instead, $\mathbf{b}^{4m+1}$ occurs twice in v' , and $\mathbf{b}^k \mathbf{ab}^{2m-k-1} \mathbf{a}$ and $\mathbf{b}^k \mathbf{ab}^{2m-k}$ become new supermaximal right-extensions for each $k \in [1, m-2]$, occurring twice in u_{k+1} and u_k , respectively. Also, $\mathbf{ab}^{2m-1}$ appears once in v' and twice in u_1 , and $\mathbf{b}^{m-1} \mathbf{ab}^{m-1}$ can be found three times in u_{m-1} . Wrapping up and analyzing its right-extensions, one can verify that the supermaximal right-extensions of w'_m and their ending positions are the following:

1. $\mathbf{ab}^{2m-1} \mathbf{a}$ (ending at $(2m+3)$ in v'), and $\mathbf{ab}^{2m}$ (ending at $(2m+3)$ in u_1 and 1 in u_2),
2. $\mathbf{b}^k \mathbf{ab}^{2m-k-1} \mathbf{a}$ (ending at $(2m+3)$ in u_{k+1} and 1 in u_{k+2}) and $\mathbf{b}^k \mathbf{ab}^{2m-k}$ (ending at $(2m+2)$ and $(4m+4)$ in u_k) for $k \in [1, m-2]$,
3. $\mathbf{b}^{2m-k} \mathbf{ab}^k \mathbf{ab}^{2m-k} \mathbf{ab}^k \mathbf{a}$ (ending at $(2m+k+4)$ in u_k) and $\mathbf{b}^{2m-k} \mathbf{ab}^k \mathbf{ab}^{2m-k} \mathbf{ab}^{k+1}$ (ending at $(k+2)$ in u_{k+1}) for $k \in [1, m-2]$,
4. $\mathbf{b}^{4m+1} \mathbf{a}$ (ending at $(4m+4)$ in v') and $\mathbf{b}^{4m+2}$ (ending at $(4m+3)$ in v'),
5. $\mathbf{b}^{m-1} \mathbf{ab}^{m-1} \mathbf{a}$ (ending at $(2m+1)$ in u_{m-1}) and $\mathbf{b}^{m-1} \mathbf{ab}^m$ (ending at $(m+1)$ and $(3m+3)$ in u_{m-1}).

Thus, we have $\mathbf{sre}(w'_m) = 4m - 2$.

Substitutions. Assuming that the string w'_m is obtained by replacing the second $\mathbf{a}$ in w_m with $\mathbf{b}$, we obtain a string similar to the one we consider in case **Insertions**. Considering its structure, we have that the supermaximal right-extensions of w'_m are the following:

1. $\mathbf{ab}^{2m-1}\mathbf{a}$ and $\mathbf{ab}^{2m}$,
2. $\mathbf{b}^k\mathbf{ab}^{2m-k-1}\mathbf{a}$ and $\mathbf{b}^k\mathbf{ab}^{2m-k}$ (for $k \in [1, m-2]$),
3. $\mathbf{b}^{2m-k}\mathbf{ab}^k\mathbf{ab}^{2m-k}\mathbf{ab}^k\mathbf{a}$ and $\mathbf{b}^{2m-k}\mathbf{ab}^k\mathbf{ab}^{2m-k}\mathbf{ab}^{k+1}$ (for $k \in [1, m-2]$),
4. $\mathbf{b}^{4m+2}\mathbf{a}$ and $\mathbf{b}^{4m+3}$,
5. $\mathbf{b}^{m-1}\mathbf{ab}^{m-1}\mathbf{a}$ and $\mathbf{b}^{m-1}\mathbf{ab}^m$.

Thus, we have $\mathbf{sre}(w'_m) = 4m - 2$.

Deletions. Let us consider the string w'_m obtained by adding $\mathbf{a}$ just following the second $\mathbf{a}$ of w_m . Same as above, $\mathbf{b}^{2m-k+1}\mathbf{ab}^k$ occurs only once for each $k \in [1, m-1]$, and any other occurrences of $\mathbf{b}^i\mathbf{ab}^j$ disappears when $i + j > 2m$. This is almost as the previous case, but we have to consider the right-extension of $\mathbf{b}^{2m}\mathbf{a}$ found twice, once in the prefix $v' = \mathbf{ab}^{2m}\mathbf{aab}^{2m+2}$ and once across $v'u_1$, whereas $\mathbf{ab}^{2m}\mathbf{a}$ and $\mathbf{ab}^{2m+1}$ still remain supermaximal right-extensions. We then have that the elements of $\mathcal{S}_r(w'_m)$ are the following:

1. $\mathbf{ab}^{2m}\mathbf{a}$ and $\mathbf{ab}^{2m+1}$,
2. $\mathbf{ab}^{2m-1}\mathbf{a}$ and $\mathbf{ab}^{2m}$,
3. $\mathbf{b}^k\mathbf{ab}^{2m-k-1}\mathbf{a}$ and $\mathbf{b}^k\mathbf{ab}^{2m-k}$ (for $k \in [1, m-2]$),
4. $\mathbf{b}^{2m-k}\mathbf{ab}^k\mathbf{ab}^{2m-k}\mathbf{ab}^k\mathbf{a}$ and $\mathbf{b}^{2m-k}\mathbf{ab}^k\mathbf{ab}^{2m-k}\mathbf{ab}^{k+1}$ (for $k \in [1, m-2]$),
5. $\mathbf{b}^{2m}\mathbf{aa}$ and $\mathbf{b}^{2m}\mathbf{ab}$,
6. $\mathbf{b}^{2m+1}\mathbf{a}$ and $\mathbf{b}^{2m+2}$,
7. $\mathbf{b}^{m-1}\mathbf{ab}^{m-1}\mathbf{a}$ and $\mathbf{b}^{m-1}\mathbf{ab}^m$.

Thus, we have $\mathbf{sre}(w'_m) = 4m + 2$.

Rotations. Lastly, we consider the string w'_m that is a rotation of w_m beginning in the second $\mathbf{a}$. Note that u_m is now followed by $\mathbf{ab}^{2m}$. Hence, with respect to w_m , the substring $\mathbf{b}^{m+1}\mathbf{ab}^{m-1}\mathbf{ab}^{m+1}\mathbf{ab}^{m-1}$ occurs twice in w'_m . Thus, we have that the elements of $\mathcal{S}_r(w'_m)$ are the following:

1. $\mathbf{b}^{2m+1}\mathbf{a}$ and $\mathbf{b}^{2m+2}$,
2. $\mathbf{b}^k\mathbf{ab}^{2m-k-1}\mathbf{a}$ and $\mathbf{b}^k\mathbf{ab}^{2m-k}$ (for $k \in [1, m-1]$),
3. $\mathbf{b}^{2m-k}\mathbf{ab}^k\mathbf{ab}^{2m-k}\mathbf{ab}^k\mathbf{a}$ and $\mathbf{b}^{2m-k}\mathbf{ab}^k\mathbf{ab}^{2m-k}\mathbf{ab}^{k+1}$ (for $k \in [1, m-1]$).

Thus, we have $\mathbf{sre}(w'_m) = 4m - 2$. □

Corollary 2. *There exists a string family where $\mathbf{sre}(w) - \mathbf{sre}(w') \in \Omega(\sqrt{n})$.*

Proof. The strings w_m in Lemmas 6 and 7 have length $\Theta(m^2)$. Since $\mathbf{sre}(w_m) = 6m + 4$ and $\mathbf{sre}(w'_m) \leq 4m + 2$ for the string w'_m that results from applying any edit operation or rotation on w_m , the result follows. □

4.3. Sensitivity to string reversal

We now show that $\mathbf{sre}$ can grow by $\Omega(n)$ upon string reversals. We remind that the maximum additive growth is always $O(n)$ because $\chi \leq n$.

Lemma 8. *There exists a string family where $\mathbf{sre}(w^R) - \mathbf{sre}(w) = n/5 - 1$.*

Proof. Such a family is composed of the strings $w_k = \prod_{i=1}^k \mathbf{ca}_i \#_i \mathbf{a}_i \$_i$ on the alphabet $\Sigma = \bigcup_{i=1}^k \{\mathbf{a}_i, \#_i, \$_i\} \cup \{\mathbf{c}\}$. The size of w_k is $n = 5k$.

Observe that any substring of w_k containing $\#_i$ or $\$_i$ is not right-maximal because those symbols occur once. Moreover, every substring of length 2 of w_k is also unique and not right-maximal. Therefore, all the supermaximal extensions in w_k and w_k^R have length at most 2.

One can verify that the set of supermaximal extensions of w_k is $\mathcal{S}_r(w_k) = \bigcup_{i=1}^k \{\mathbf{a}_i \#_i, \mathbf{a}_i \$_i, \mathbf{ca}_i\} \cup \{\mathbf{c}\}$. Thus, $\mathbf{sre}(w_k) = 3k + 1$.

On the reversed string, one can verify that the set of supermaximal extensions of w_k^R is $\mathcal{S}_r(w_k^R) = \bigcup_{i=1}^k \{\mathbf{a}_i \mathbf{c}, \mathbf{a}_i \#_i, \mathbf{a}_i\} \cup \bigcup_{i=1}^{k-1} \{\mathbf{c} \$_i\} \cup \{\$_k\}$. Thus, $\mathbf{sre}(w_k^R) = 4k$, and the claim follows. $\square$

Proof (of Theorem 1). We have obtained along the section those results in terms of $\mathbf{sre}$, which by Corollary 1 can be written in terms of χ . Claim 1 follows by Lemmas 2 and 3. Claim 2 follows by Lemma 7, where $n = |w_m| = \Theta(m^2)$ and $AS_{\chi, \rho}(n) = \Omega(m) = \Omega(\sqrt{n})$, for all $\rho \in \{\mathbf{ins}, \mathbf{del}, \mathbf{sub}, \mathbf{rot}\}$. The $\Omega(n)$ part of Claim 3 follows by Lemma 8, where $n = |w_k| = 5k = \Theta(k)$ and $AS_{\chi, \mathbf{rev}}(n) = \Omega(k) = \Omega(n)$; the $O(n)$ part is a consequence of χ being $O(n)$. $\square$

5. Multiplicative Sensitivity of χ to String Operations

We now focus our attention to multiplicative sensitivity. We first show that χ has a multiplicative sensitivity to reversals that is a constant strictly larger than 1. Then we show a relation between the multiplicative sensitivities of the other operations. We finally express both additive and multiplicative sensitivities in terms of the substring complexity δ .

5.1. Sensitivity to edits and rotations

A simple consequence of previous results is that the multiplicative sensitivity of edits and rotations is at least $3/2 - o(1)$.

Corollary 3. *There exists a string family where $\lim_{n \rightarrow \infty} \mathbf{sre}(w)/\mathbf{sre}(w') = 3/2$ upon edit operations or rotations.*

Proof. The strings w_m in Lemmas 6 and 7 satisfy $\mathbf{sre}(w_m) = 6m + 4$, whereas $\mathbf{sre}(w'_m) \leq 4m + 2$ holds for the string w'_m that results from applying any edit operation or rotation on w_m . The result follows. $\square$

We conjecture that the multiplicative sensitivity of all those operations is indeed constant. The following results relate them all in this sense.

Lemma 9. *For any $\text{op}_1, \text{op}_2 \in \{\text{ins}, \text{del}, \text{sub}, \text{rot}\}$, $\mathbf{sre}(\text{op}_1(w))/\mathbf{sre}(w) = \Theta(1)$ if and only if $\mathbf{sre}(\text{op}_2(w))/\mathbf{sre}(w) = \Theta(1)$.*

Proof. Suppose $\mathbf{sre}(w')/\mathbf{sre}(w) = \Theta(1)$ for any $w = xy$ and $w' = yx$ a rotation of w . Then, writing $X \approx Y$ for $X = \Theta(Y)$, it holds

$$\mathbf{sre}(xcy) \approx \mathbf{sre}(yxc) \approx \mathbf{sre}(yx) \approx \mathbf{sre}(xy)$$

and

$$\mathbf{sre}(xcy) \approx \mathbf{sre}(yxc) \approx \mathbf{sre}(yxd) \approx \mathbf{sre}(xdy),$$

which implies $\mathbf{sre}(w')/\mathbf{sre}(w) = \Theta(1)$ for all edit operations. This holds because appending symbols change $\mathbf{sre}$ by $O(1)$ only.

Now assume $\mathbf{sre}(w')/\mathbf{sre}(w) = \Theta(1)$ for any $w = xy$ and $w' = xcy$ (insertion) or vice versa (deletion). Then, using a $\$$ as in Lemma 5,

$$\mathbf{sre}(xy) \approx \mathbf{sre}(x\$y) \leq \mathbf{sre}(yx\$) = O(\mathbf{sre}(yx)),$$

$$\mathbf{sre}(yx) \approx \mathbf{sre}(y\$x) \leq \mathbf{sre}(xy\$) = O(\mathbf{sre}(xy)),$$

and thus $\mathbf{sre}(xy) \approx \mathbf{sre}(yx)$. Finally, assume $\mathbf{sre}(w')/\mathbf{sre}(w) = \Theta(1)$ for any $w = xcy$ and $w' = xdy$ (substitution). Then

$$\mathbf{sre}(xcyd) \approx \mathbf{sre}(x\$yd) \leq \mathbf{sre}(ydx\$) = O(\mathbf{sre}(ydx)),$$

$$\mathbf{sre}(ydx) \approx \mathbf{sre}(y\$xc) \leq \mathbf{sre}(xcy\$) = O(\mathbf{sre}(xcy)),$$

thus $\mathbf{sre}(xcyd) \approx \mathbf{sre}(ydx)$, which includes every rotation. $\square$

5.2. Sensitivity to string reversals

The proof of Lemma 8 immediately yields a lower bound of $4/3 - o(1)$.

Corollary 4. *There exists a string family where $\lim_{n \rightarrow \infty} \mathbf{sre}(w^R)/\mathbf{sre}(w) = 4/3$.*

Proof. For the family described in the proof of Lemma 8, it holds that $\mathbf{sre}(w_k) = 3k + 1$ and $\mathbf{sre}(w_k^R) = 4k$. $\square$

We now prove that χ at most doubles when we reverse the string, by studying how $\mathbf{sre}$ behaves. For simplicity of analysis, we assume that w contains a special symbol $\$$ at the beginning and at the end (this works because it is not hard to see that $\chi(w) = \mathbf{sre}(w\$) = \mathbf{sre}(\$w\$)$ for every w). We do this to (i) ensure all right-maximal substrings of w have at least two left-extensions, and hence, right-extensions have at least one left-extension, and (ii) provide the needed symmetry to prove Lemma 11 below.

In the following, we denote the set of symbols preceding a substring x (or *left-extensions* of x) of w by $L_x(w)$, and we use $l_x(w)$ to denote an arbitrary element of $L_x(w)$ (we drop w if it is clear from the context). Please note that left-extensions are not analogous to right-extensions. A right-extension is a string xa such that xb also occurs for some $b \neq a$, while a left-extension of x is just any symbol preceding x . A left-extension can be unique, whereas right-extensions come (at least) in pairs. We will rely on the following observation.

Observation 1. If xa is a supermaximal right-extension, then $L_{xa} \cap L_{xb} = \emptyset$ for any $b \neq a$.

Proof. Otherwise, xa is not supermaximal, because there exists $c \in L_{xa} \cap L_{xb}$, for which $cxa \in \mathcal{F}_w$ and $cxb \in \mathcal{F}_w$, and thus xa is a proper suffix of the right-extension cxa of the right-maximal string cx . $\square$

Let us introduce a special tree of supermaximal extensions, which will be helpful to prove our result.

Definition 6. The *prefix supermaximal right-extension tree* (PSR tree) of w is a compacted trie of all the supermaximal right-extensions of w . Since some of those strings can be prefixes of others, internal nodes of the trie may also represent strings, and their corresponding nodes are not compacted. Distinct leaves do not represent prefixes of each other.

The following result is key to our proof.

Lemma 10. *Let w be a string starting with a unique symbol $\$$. If x is a right-maximal substring of w , and xa is a supermaximal right-extension of x in w , then x^R is a right-maximal substring of w^R .*

Proof. Because xa is a right-extension in w , there exists $xb \in \mathcal{F}_w$ with $b \neq a$. Any symbol preceding xa cannot precede xb , otherwise, xa would not be supermaximal. Since w starts with the unique symbol $\$$, if $x \neq \varepsilon$, then xa and xb have occurrences that are not prefixes of w , and hence, they are preceded by different symbols. It follows x^R is right-maximal in w^R . $\square$

Lemma 11. *It always holds that $\mathbf{sre}(w^R)/2 \leq \mathbf{sre}(w) \leq 2 \cdot \mathbf{sre}(w^R)$.*

Proof. We define an injection λ from the set of leaves and unary internal nodes (those with exactly one child) of the PSR tree $T(w)$ of w into $\mathcal{S}_r(w^R)$. We will identify tree nodes with the string they represent.

- **Leaf nodes:** If xa is (the supermaximal right-extension represented by) a leaf node, we let $\lambda(xa) = ux^R \cdot l_{xa}$ where $l_{xa} \in L_{xa}(w)$ (which exists because w starts with $\$$, as discussed), and u is chosen such that $ux^R \cdot l_{xa} \in \mathcal{S}_r(w^R)$. At least one such u must exist because $x^R \cdot l_{xa} \in \mathcal{F}_{w^R}$ and x^R is right-maximal in w^R by Lemma 10.
- **Unary internal nodes:** If xa a unary internal node with child $yb = xazb$, consider a right-extension $xazc$ of xaz , where $c \neq b$ (one must exist because xaz is right-maximal). We then let $\lambda(xa) = uz^R ax^R \cdot l_{yc} \in \mathcal{S}_r(w^R)$ where $l_{yc} \in L_{yc}(w) \subseteq L_{xa}(w)$. Note L_{yc} is not empty because, as in the previous point, $xazc$ is a supermaximal right-extension, and u exists because $z^R ax^R$ is right-maximal in w^R by Lemma 10.

We now prove that λ is an injection. Consider two leaves xa and yb of $T(w)$:

1. If $x[i] \neq y[i]$ for some i , then $\lambda(xa) = ux^R \cdot l_{xa}$ and $\lambda(yb) = vy^R \cdot l_{yb}$, aligned from the right, differ at their position $\lambda(xa)[|\lambda(xa)| - i] \neq \lambda(yb)[|\lambda(yb)| - i]$, and therefore are different.
2. If x is a proper prefix of y , then y has to start with xc for some $c \neq a$ (otherwise xa would be internal). Because xa is a supermaximal right-extension, the set of left-extensions L_{xa} and $L_{yb} \subseteq L_{xc}$ are disjoint by Observation 1. Therefore, the selected supermaximal extensions are of the form $\lambda(xa) = ux^R \cdot l_{xa}$ and $\lambda(yb) = vy^R \cdot l_{xc}$ with $l_{xa} \neq l_{xc}$, thus, distinct.
3. Similarly, if $x = y$, the two supermaximal extensions xa and xb have disjoint sets of left-extensions L_{xa} and L_{xb} , and hence $\lambda(xa) \neq \lambda(xb)$ because they end in different characters.

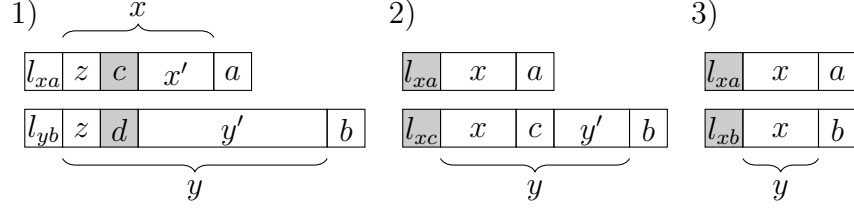


Figure 3: Cases 1), 2), and 3) of Lemma 11, where xa and yb are leaves of the PSR tree. Note that the suffix $x^R \cdot l_{xa}$ and $x^R \cdot l_{yb}$ of $\lambda(xa)$ and $\lambda(yb)$ are not a suffix one of the other because the grayed symbols are different.

For a graphic depiction of these three cases see Figure 3.

In the other cases, one of the two nodes is a (unary) internal node. Assume, without loss of generality, that $|x| \leq |y|$. By the same argument of point 1 above, $\lambda(xa)$ cannot be equal to some $\lambda(yb)$ if x is not a prefix of y . It follows that xa is a unary internal PSR node that is an ancestor of $yb = xazb$:

4. Let yb be the child of xa . Let $c \neq b$ be such that $xazc$ is a right-extension in w , and $\lambda(xa) = uz^Rax^R \cdot l_{yc}$ with $l_{yc} \in L_{yc}$. Note that $\lambda(xa)$ differs from $\lambda(yb)$ because $\lambda(yb)$ finishes with some $l_{yb} \in L_{yb}$ and, by Observation 1, $L_{yb} \cap L_{yc} = \emptyset$.
5. If yb is a farther descendant of xa , then xaz has children by c and by some $d \neq c$ where $xazd$ prefixes y . The string $xazd$ is a supermaximal right-extension because there is a node for it in the tree, hence, by Observation 1, it holds $L_{xazc} \cap L_{xazd} = \emptyset$. As $L_{yb} \subseteq L_{xazd}$, it follows $L_{xazc} \cap L_{yb} = \emptyset$. Therefore, $\lambda(xa)$ and $\lambda(yb)$ end in different symbols.

For a graphic depiction of these two cases see Figure 4.

We are then injecting k nodes into $\mathcal{S}_r(w^R)$, where $k \geq \mathbf{sre}(w)/2$ (with equality when T is a full binary tree). Thus, $\mathbf{sre}(w)/2 \leq \mathbf{sre}(w^R)$. Since the reversal operation is an involution, we have similarly that $\mathbf{sre}(w^R)/2 \leq \mathbf{sre}(w)$. $\square$

5.3. Sensitivity as a function of δ

Finally, we give upper bounds on the sensitivity of χ to string operations in terms of measure δ .

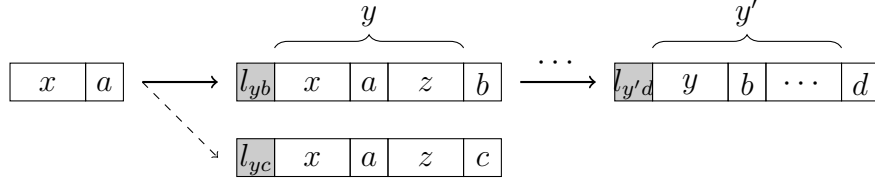


Figure 4: Case 4) and 5) of Lemma 11, where xa is an internal node of the PSR tree, yb is the only child of xa , and $y'd$ is a farther descendant of xa (if it exists). Note $xazc$ is a right-extension, but not supermaximal (otherwise, xa would not be unary). In this case, $\lambda(xa)$ is suffixed by $y^R \cdot l_{yc}$, whereas $\lambda(yb)$ and $\lambda(y'd)$ are suffixed by $y^R \cdot l_{yb}$ and $y^R \cdot l_{y'd}$, respectively. Note how yb and all its descendants yield $\lambda(\cdot)$ values differing from $\lambda(xa)$ in their last symbol.

Lemma 12. *Let $w \in \Sigma^*$ and $w' \in \text{ins}(w) \cup \text{del}(w) \cup \text{sub}(w) \cup \text{rot}(w) \cup \text{rev}(w)$. For $\delta = \delta(w)$, it holds*

$$\begin{aligned} \chi(w') - \chi(w) &= O(\delta \max(1, \log(n/(\delta \log \delta))) \log \delta) \quad \text{and} \\ \chi(w') / \chi(w) &= O(\max(1, \log(n/(\delta \log \delta))) \log \delta). \end{aligned}$$

Proof. It holds $\delta \leq \chi \leq 2\bar{r}$ [5] and $r = O(\delta \max(1, \log(n/(\delta \log \delta))) \log \delta)$ [24]. Since the multiplicative sensitivity of δ to any of the string operations is $O(1)$ [15], for any $w \in \Sigma^*$ it holds $\bar{r}(w) = r(w^R) = O(\delta \max(1, \log(n/(\delta \log \delta))) \log \delta)$. The thesis follows by considering the worst case, that is, $\chi(w) = \Theta(\delta)$ and $\chi(w') = \Theta(\delta \max(1, \log(n/(\delta \log \delta))) \log \delta)$. $\square$

6. Relating χ to Other Repetitiveness Measures

Previous work [4] established that $\gamma \leq \chi \leq 2\bar{r}$ on every string family. In this section we obtain the more natural upper bound $\chi \leq 2r$, and that χ can be asymptotically strictly smaller, $\chi = o(r)$, on some string families (we actually prove $\chi = o(v)$). We also show that χ is incomparable with all the copy-paste measures except s and b , in the sense that there are string families where χ is asymptotically strictly smaller than the others, and vice versa. We recall that $s = O(\chi)$ and that the relation between b and χ remains unknown.

6.1. Proving $\chi \leq 2r$

We first prove that χ is asymptotically upper-bounded by the number r of runs in the BWT of the sequence. As for the measure χ , we assume

that the BWT is computed after appending the $\$$ symbol. Observe that the precise bound of the following lemma has been recently proved to be tight for binary strings, while on σ -ary alphabets, with $\sigma > 2$, there exists a family of strings that approaches it as σ grows [25].

Lemma 13. *It always holds that $\chi \leq 2r$.*

Proof. Let x_i denotes the i th rotation of $w\$$ in lexicographic order, for each $i \in [1 \dots |w|+1]$, and let u_i be the longest common prefix between the rotations x_i, x_{i+1} , for each $i \in [1 \dots |w|]$ (note this implies that u_i is right-maximal). We further define $s : [1 \dots n+1] \rightarrow [0 \dots n]$ as $s(i) = j$ if $x_i = w[j+1 \dots |w|]\$w[1 \dots j]$, that is, the number of cyclic shifts to the right required to transform x_i into $w\$$.² As the symbol $\$$ occurs only once in $w\$$, the function s is bijective.

Note that each right-extension of $w\$$ can be written as $u_i c$, for some $i \in [1 \dots |w|]$ and $c \in \Sigma$. Consider now the set

$$S = \bigcup_{i \in [1 \dots |w|]} \{s(i) + |u_i| + 1, s(i+1) + |u_i| + 1\},$$

that is, the set of positions where the occurrences of the right-extensions $u_i c_1$ and $u_i c_2$ end in $w\$$, where $u_i c_1$ and $u_i c_2$ are the prefixes of x_i and x_{i+1} , respectively, for some $c_1, c_2 \in \Sigma$ such that $c_1 < c_2$. It follows by construction that the set S is a suffixient set of $w\$$.

We now show that $|S| \leq 2r$. Let us factorize each pair of consecutive rotations in the BWT-matrix as $x_i = u_i v_i c_i$ and $x_{i+1} = u_i v'_i c_{i+1}$. Observe that $v_i, v'_i \neq \varepsilon$ [16, Corollary 8], $v_i[1] \neq v'_i[1]$, and $c_i = \text{BWT}(w\$)[i]$ for all $i \in [1 \dots |w|+1]$. A well-known property of the BWT-matrix is that if $c_i = c_{i+1} = c \in \Sigma$, then there exists $j \in [1 \dots |w|]$ such that $x_j = c u_i v_i$ and $x_{j+1} = c u_i v'_i$ [7]. As a consequence, one has that $s(j) + |u_j| + 1 = (s(i) - 1) + (|u_i| + 1) + 1 = s(i) + |u_i| + 1$ and $s(j+1) + |u_j| + 1 = (s(i+1) - 1) + (|u_i| + 1) + 1 = s(i+1) + |u_i| + 1$, and the procedure can be reiterated as long as x_j and x_{j+1} end with the same symbol. See Figure 5 for an illustration. It follows that the same set can be written as

$$S = \{s(i) + |u_i| + 1, s(i+1) + |u_i| + 1 \mid i \in [1 \dots |w|] \wedge \text{BWT}[i] \neq \text{BWT}[i+1]\},$$

that is, the size of S is at most twice the number of equal-letter runs in $\text{BWT}(w\$)$, and the thesis follows. $\square$

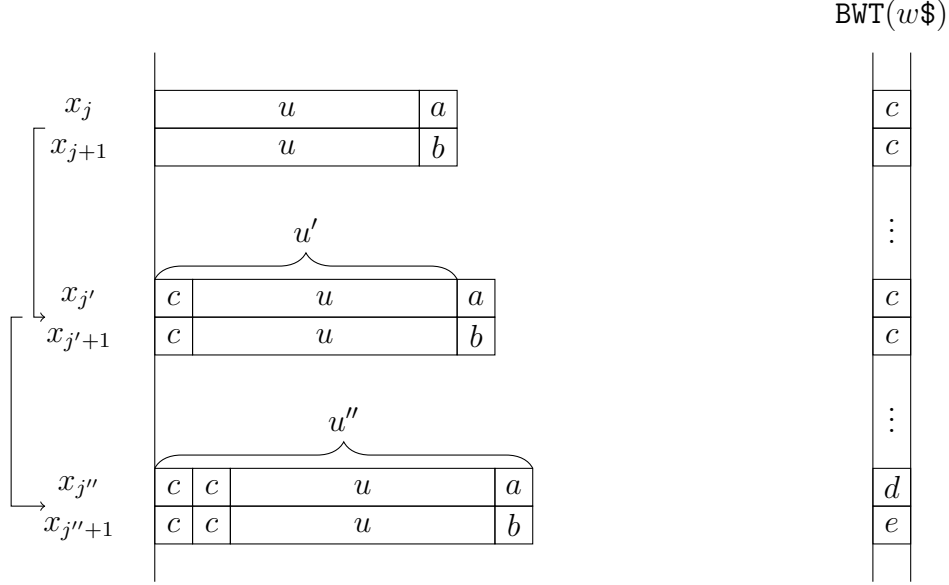


Figure 5: Depiction of the proof of Lemma 13.

In some fields like Combinatorics on Words, the BWT is used without appending the dollar symbol at the end of the string. We call this variant the *circular BWT* and denote its number of runs by r_c . This small change can have a great impact in the value of the measure: r and r_c can differ by a $\Omega(\log n)$ factor [19]. We show that χ is also smaller than the variant r_c .

Lemma 14. *It always holds that $\chi \leq 2r_c + 2$.*

Proof. Consider this time the BWT-matrix of w instead of $w\$$. Observe that all the right-extensions of $w\$$ still appear as $u_i a$ for some $u_i = lcp(x_i, x_{i+1})$ and symbol $a \in \Sigma$, with some exceptions:

1. the right-extensions ending with $\$$, and
2. the right-extensions za ending with some symbol a such that only za and $z\$$ are right-extensions.

The right-extensions in Case 1 and Case 2 form two suffix chains, and

²The function s mimics the well-known Suffix Array [26], here omitted for simplicity of exposition.

hence, the set

$$S = \bigcup_{i \in [1..|w|-1]} \{s(i) + |u_i| + 1, s(i+1) + |u_i| + 1\},$$

is missing positions for at most 2 supermaximal right-extensions. Thus, $\chi \leq 2r_c + 2$. $\square$

6.2. A family with $\chi = o(v)$ (and thus $o(r)$)

We will now show that $\chi = o(v)$ on the so-called Fibonacci words, which also implies $\chi = o(r)$ in that string family because $v = O(r)$ [9]. Combined with Lemma 13, this implies that χ is a strictly smaller measure than r . In contrast, χ is incomparable with v , as we show later. On our way, we obtain some relevant byproducts about the structure of suffixient sets on Fibonacci, and more generally, episturmian words.

Definition 7 ([27, 28]). An infinite string $\mathbf{w}$ is *episturmian* if it has at most one right-maximal substring of each length and its set of substrings is closed under reversal, that is, $\mathcal{F}_{\mathbf{w}} = \mathcal{F}_{\mathbf{w}}^R$. It is *standard episturmian* (or *epistandard*) if, in addition, all the right-maximal substrings of $\mathbf{w}$ are of the form $\mathbf{w}[1..i]^R$ with $i \geq 0$, that is, they are the reverse of some prefix of $\mathbf{w}$.

Lemma 15. *Let $\mathbf{w} \in \Sigma^\omega$ be an episturmian word with $|\Sigma| = \sigma \geq 2$. Then, $\mathbf{sre}(\mathbf{w}[i..j]) \leq \sigma$ for any $i, j \geq 0$.*

Proof. Let $\mathbf{w}$ be an epistandard word. The right-extensions $x_1, x_2, \dots$ ending with $a \in \Sigma$ form a *suffix-chain* where each x_i is a suffix of x_{i+1} . This is because the right-maximal substrings are the reverses of the prefixes of the word, so they are all suffixes of the longest one. If we add the same letter to a subset of them to obtain right-extensions, they still form a suffix-chain. There is one of those suffix-chains for each character $a \in \Sigma$.

Let $\mathbf{w}$ be episturmian but not necessarily epistandard. There exists some epistandard word $\mathbf{s}$ with the same set of substrings, i.e., $\mathcal{F}_{\mathbf{w}} = \mathcal{F}_{\mathbf{s}}$ [27]. Therefore, for any episturmian word $\mathbf{w}$, there exist exactly σ suffix-chains of right-extensions.

When considering substrings of $\mathbf{w}$, the supermaximal right-extension in $\mathbf{w}[i..j]$ ending with $a \in \Sigma$ is the longest right-extension of $\mathbf{w}$ ending with a that remains a right-extension in $\mathbf{w}[i..j]$. It follows that for any substring $\mathbf{w}[i..j]$ of any episturmian word $\mathbf{w}$, it holds $\mathbf{sre}(\mathbf{w}[i..j]) \leq \sigma$. $\square$

Combining this result with Corollary 1, we obtain the following bound.

Corollary 5. *For any episturmian word $\mathbf{w} \in \Sigma^\omega$ it holds $\chi(\mathbf{w}[i..j]) \leq \sigma + 2$.*

The next lemma precisely characterizes the suffixient sets of Fibonacci words, a particular case of epistandard words that will be useful to relate χ with v .

Definition 8. Let $F_1 = \mathbf{b}$, $F_2 = \mathbf{a}$, and $F_k = F_{k-1}F_{k-2}$ for $k \geq 3$ be the Fibonacci family of strings. Their lengths, $f_k = |F_k|$, form the Fibonacci sequence.

Lemma 16. *Every Fibonacci word $F_k\$$ has a suffixient set of size at most 4. For $k \geq 6$, the only smallest suffixient sets for $F_k\$$ are $\{f_k + 1, f_k - 1, f_{k-1} - 1, p\}$, where $p \in \{f_{k-2} + 1, 2f_{k-2} + 1\}$.*

Proof. The upper bound of 4 stems directly from Corollary 5, because the infinite Fibonacci word is binary epistandard. For $k \geq 3$, there exist strings H_k such that $F_k = F_{k-1}F_{k-2} = H_kcd$ and $F_{k-2}F_{k-1} = H_kdc$, for $cd = \mathbf{ab}$ or $cd = \mathbf{ba}$ depending on the parity of k [29]. Let us call $F'_k = H_kdc = F_{k-2}F_{k-1}$, that is, F_k with the last two letters exchanged; thus $F_k = F_{k-1}F_{k-2} = F_{k-2}F'_k$.

Note that $F_{k-1} = H_{k-1}dc$ prefixes F_k . On the other hand, we can write $F_k = F_{k-1}F_{k-2} = F_{k-2}F_{k-3}F_{k-2} = F_{k-2}F'_{k-1} = F_{k-2}H_{k-1}cd$. Therefore, string H_{k-1} is right-maximal in F_k . Its extensions, $H_{k-1}d$ and $H_{k-1}c$, are supermaximal because there are no other occurrences of H_{k-1} in F_k : (i) H_{k-1} cannot occur starting at positions $f_{k-2} + 2$ or $f_{k-2} + 3$ because it occurs at $f_{k-2} + 1$, so H_{k-1} should match itself with an offset of 1 or 2, which is impossible because it prefixes F_{k-1} and all F_{k-1} for $k-1 \geq 5$ start with $\mathbf{abaab}$; (ii) H_{k-1} cannot occur starting at positions 2 to f_{k-2} because its prefix F_{k-2} should occur inside the prefix $F_{k-2}F_{k-2}$ of $F_k = F_{k-2}F'_{k-1} = F_{k-2}F_{k-2}F'_{k-3}$, and so F_{k-2} should equal a rotation of itself, which is impossible [30, Cor. 3.2]. The two positions following H_{k-1} , $f_{k-1} - 1$ and $f_k - 1$, then appear in any suffixient set.

On the other hand, F_{k-2} is followed by $\$$ in $F_k\$$, and it also prefixes $F_k = F_{k-2}F'_{k-1}$, therefore F_{k-2} is right-maximal. The first occurrence is preceded by F_{k-1} , and hence by c , and the second by no symbol. F_{k-2} also occurs in F_k at position $f_{k-2} + 1$, as seen above, preceded by F_{k-2} and thus by d . There are no other occurrences of F_{k-2} in F_k because (i) it cannot occur starting at positions 2 to f_{k-2} by the same reason as point (ii) of the previous

paragraph; (ii) it cannot appear starting at positions $f_{k-2} + 2$ to $f_{k-1} - 2$ because $F_k = F_{k-2}F_{k-2}F'_{k-3}$ and $F'_{k-3}[1, f_{k-3} - 2] = F_{k-3}[1, f_{k-3} - 2] = F_{k-2}[1..f_{k-3} - 2]$, thus such an occurrence would also match a rotation of F_{k-2} , which is impossible as noted above; (iii) it cannot appear starting at positions $f_{k-1} - 1$ or f_{k-1} because, since it matches at position $f_{k-1} + 1$, F_{k-2} would match itself with an offset of 1 or 2, which is impossible as noted in point (i) of the previous paragraph. The right-extensions of F_{k-2} are then supermaximal. The one followed by $\$$ occurs ending at position $f_k + 1$. The other two are followed by $\mathbf{a}$ because they are followed by F_{k-2} and by F'_{k-3} and all F_k for $k \geq 2$ start with $\mathbf{a}$. We can then choose either ending position for a suffixient set, $f_{k-2} + 1$ or $2f_{k-2} + 1$. $\square$

Corollary 6. *There exist string families where $\chi = o(v)$.*

Proof. It follows from Lemma 16 and the fact that $v = \Omega(\log n)$ on the odd Fibonacci words [9, Thm. 28]. $\square$

6.3. Uncomparability of χ with copy-paste measures

Finally, we show that χ is incomparable with most copy-paste measures. This follows from χ being $\Theta(n)$ on de Bruijn sequences and $O(1)$ on Fibonacci strings.

Definition 9. A *binary de Bruijn sequence of order $k > 0$* [31] contains every binary string in $\{\mathbf{a}, \mathbf{b}\}^k$ as a substring exactly once. The length of this sequence is $n = 2^k + (k - 1)$. The set of binary de Bruijn sequences of order k is $\mathbf{dB}(k)$.

Lemma 17. *It holds $\mathbf{sre}(w) = 2^k = \Omega(n)$ for any $w[1..n] \in \mathbf{dB}(k)$.*

Proof. Let $w[1..n]$ be a binary de Bruijn string of order k . By definition, w contains every binary string of length k as a substring exactly once. As all the possible pairs of strings $x\mathbf{a}$ and $x\mathbf{b}$ of length k appear in w , it follows that all the strings in $\mathcal{F}_w(k)$ are right-extensions. Moreover, each $x\mathbf{a}$ and $x\mathbf{b}$ of length k are supermaximal right-extensions: otherwise, there would exist some $c \in \{\mathbf{a}, \mathbf{b}\}$ such that cxa and cxb are both substrings of w , which raises a contradiction since the k -length string cx cannot appear twice in w . Moreover, there are no right-maximal strings of length k or greater; hence, there are no right-extensions of length greater than k . It follows that $\mathbf{sre}(w) = |\mathcal{F}_w(k)| = 2^k = \Omega(n)$. $\square$

Because $g = O(n/\log n)$ on de Bruijn sequences [9] and by Lemma 17, we have:

Corollary 7. *There exists a string family with $\chi = \Omega(g \log n)$.*

This result is particularly relevant because all the copy-paste based measures μ , with the exception of z_e , are $O(g)$. Corollary 7 then implies $\mu = o(\chi)$ on de Bruijn sequences for all these measures μ .

While it has been said that $z_e = O(n/\log n)$ on binary sequences as well [32], this referred to the version that adds to each phrase the next non-matching character. Because z_e is not an optimal parse, it is not obvious that this also holds for the version studied later in the literature, which does not add the next character. We then prove next that $z_e = o(\chi)$ holds on de Bruijn words.

Lemma 18. *There exists a string family with $\chi = \Omega\left(z_e \frac{\log n \log \log \log n}{(\log \log n)^2}\right)$.*

Proof. It always holds that $z_e = O\left(z \frac{\log^2(n/z)}{\log \log(n/z)}\right)$ [33]. In de Bruijn sequences it holds that $z = \Theta(n/\log n)$, so $n/z = \Theta(\log n)$. Therefore, $z_e = O\left(z \frac{(\log \log n)^2}{\log \log \log n}\right)$, and replacing $z = \Theta(n/\log n)$ we get $z_e = O\left(n \frac{(\log \log n)^2}{\log n \log \log \log n}\right)$. By Lemma 17, this yields $\chi = \Omega\left(z_e \frac{\log n \log \log \log n}{(\log \log n)^2}\right) = \omega(z_e)$ on de Bruijn sequences. $\square$

Corollary 8. *The measure χ is uncomparable to $\mu \in \{z, z_{no}, z_e, z_{end}, v, g, g_{rl}, c\}$.*

Proof. From Corollary 7 and Lemma 18, and that $z, z_{no}, z_{end}, v, g_{rl}$ and c are always $O(g)$, it follows that there are string families where $\mu = o(\chi)$, for any $\mu \in \{z, z_{no}, z_e, z_{end}, v, g, g_{rl}, c\}$. On the other hand, from Lemma 16 and Corollary 6, and that $c = \Omega(\log n)$ on Fibonacci words [9, Thm. 32] and $c = O(\mu)$ for any $\mu \in \{z, z_{no}, z_e, z_{end}, g_{rl}, g\}$ [9, Thm. 30], it follows that there are string families where $\chi = o(\mu)$, for any $\mu \in \{z, z_{no}, z_e, z_{end}, v, g, g_{rl}, c\}$. $\square$

7. Online Computation of Smallest Suffixient Sets

As an application of Lemmas 2 and 3, we show that Ukkonen's [13] and Weiner's [12] linear-time online constructions of suffix trees can be easily modified to compute smallest suffixient sets within the same space and time complexity.

The *suffix tree* of a text T is a tree of size $O(|T|)$ where edges are labeled by nonempty substrings of T (the label $T[i..j]$ is indicated by the pair (i, j)). Every internal node has at least two children, and the labels of edges to any two children must differ in their first symbol. If node x has a child node y by an edge labeled (i, j) , we say that y is a child *by label* $T[i]$. Each node x is said to be the *locus* of the string obtained by concatenating the labels of the edges that lead from the root to x . The *string depth* of a node x is the length of the string x is the locus of. The tree leaves are the loci of suffixes $T[i..|T|]$, whereas internal nodes are loci of strings that occur at least twice in T . Edges to leaves have labels of the form (i, ∞) , which means the second component is always the last position processed of T . Suffix tree nodes also have so-called *suffix links*, which lead from a node that is the locus of a string cu , for $c \in \Sigma$, to the locus of string u (which always exists: if cu occurs more than once, so does u). Finally, we extend the concept of suffix tree nodes to *implicit nodes*, which are virtual nodes with one child, assumed to exist along edges: if y is the child of x by an edge labeled (i, j) and x is the locus of string u , then $(x, (i, p))$, for $i \leq p < j$, is an implicit node that is the locus of $u \cdot T[i..p]$. Knowing the loci of which string they are, suffix links are also defined from implicit nodes (those can lead to explicit or to implicit nodes).

7.1. Ukkonen's left-to-right construction

Ukkonen's algorithm [13] builds the suffix tree of T such that, after having processed any prefix w of T , it has built the suffix tree of w . To prevent special cases, it assumes that the root is in fact a child of a special node $\perp$, with edges to the root labeled c for every $c \in \Sigma$; the suffix link of the root points to $\perp$. We do not aim at a full explanation of the algorithm, but just highlight some of its relevant properties. Let the prefix w of T be followed by $c \in \Sigma$. The algorithm maintains pointers to two (possibly implicit) nodes of the suffix tree of w :

- s : called the *active point* of w , is the locus of the longest suffix u of w that occurs at least twice in w ;
- s' : called the *end point* of w , is the locus of the longest suffix v of w such that vc occurs in w .

Note that this implies that the locus of vc is the active point of wc (i.e., the new s after processing c), and that, because v must be a suffix of u , there is a *chain* of consecutive suffix links from s to s' . The algorithm updates the

suffix tree nodes in that chain, from s to (but not including) s' , by adding a new leaf child labeled c to those nodes. Although updating the suffix tree of w to obtain that of wc may take non-constant time, the time does amortize to constant [13].

We will enhance the suffix tree nodes by adding a *mark* to the nodes that are loci of the supermaximal right-extensions of a smallest suffixient set of w . The length of the extension is the string depth of the marked node, and any suffix descending from the node serves as a starting position of such extension. This is the way in which we maintain a smallest suffixient set for the current prefix w of T . For example, we can easily maintain the marked nodes in a list in order to collect the set for any prefix in optimal time. To compute $\chi(T)$, we can run the algorithm on $T\$$ and then return the length of the list.

Note that the loci of right-extensions are either explicit nodes, or implicit nodes of the form $(x, (i, i))$, because they extend by one symbol a string that occurs more than once. We can then always associate the mark to the corresponding explicit child y of x , so as to consult and update it in constant time.

The first letter c of T is processed by adding a child leaf of the root by label c and setting s to the root. From there on, considering the cases of Lemma 2, we proceed as follows:

s has a child by label c : This means that $s = s'$ and Ukkonen's algorithm just descends by c to find the new s . We do not need any further action because we are in case 1 of the lemma.

s has two or more children, none by label c : This is case 2 of the lemma. Ukkonen's algorithm will create a new leaf child by label c of s , and of all the nodes in the suffix-link chain until it finds s' (i.e., the first node in the chain having a child by label c). We then (1) mark the (just created) child of s by label c , and (2) unmark, if it is marked, the (already existing) child of s' by label c . This is because uc is a new supermaximal right extension of the right-maximal string u and, in case (2), vc ceases to be a supermaximal right extension because it is a suffix of the new one we are adding, uc .

s has just one child, by label $a \neq c$: This is case 3 of the lemma. Ukkonen's algorithm proceeds exactly as in the previous case, finding the first s' with a child by label c in the suffix-link chain. We (1) mark the

two children of s (by labels a and c), (2) unmark, if it is marked, the child of s' by label c , and (3) unmark, if it is marked, the child of s'' by label a , where s'' is the first node in the suffix-link chain that has another child labeled $b \neq a$ (note that s'' must be on the chain from s to s' , because one possible choice is $b = c$ and $s'' = s'$). This is because ua and uc are new supermaximal right-extensions of u , and in case (2), vc is not anymore supermaximal, as before. In case (3), similarly, if s'' is the locus of z , then za is not anymore supermaximal. Note that the child by label a of s'' is the only locus of some za suffix of ua that could possibly be marked.

Ukkonen's algorithm is claimed to be $O(n)$ time, but this assumes that the alphabet is constant. If it is not, we may need more time to find the child by label c among its children, or determine it does not exist. If the alphabet is an integer range and of size polynomial in n , we can still retain $O(n)$ time in the transdichotomous RAM model of computation with computer word size $\omega = \Omega(\log n)$ using fusion trees, as the children operations can then be handled in time $O(\log_\omega |\Sigma|) = O(1)$ [34]. Otherwise, an extra factor of $O(\log |\Sigma|)$ appears.

Theorem 2. *There exists an algorithm to compute smallest suffixient sets that processes a text T left to right such that, for every n , after having processed the prefix $T[1..n]$ it has determined a smallest suffixient set of $T[1..n]$ using $O(n)$ space and $O(n)$ worst-case time. This can be used to compute $\chi(T)$ within $O(|T|)$ space and time.*

7.2. Weiner's right-to-left construction

Weiner's algorithm [12] process the text right-to-left, prepending a symbol at each step. Analogously to Ukkonen's algorithm, after having processed a suffix of T , it has built the suffix tree of that suffix. The algorithm is possibly less convenient than Ukkonen's, as it is more natural to process the text left to right, but its adaptation to compute **sre** using Lemma 3 needs diving less into its details. Further, it has the property that we immediately know χ for each suffix of T as soon as we process it.

The algorithm is actually defined for a $\$$ -terminated text, $T\$$. Assume we have already processed the suffix $w\$$ of $T\$$, and now prepend c to obtain the suffix tree of the suffix $cw\$$. The algorithm maintains a pointer to the deepest explicit node u in the path to the leaf that represents $w\$$. To go

from $w\$$ to $cw\$$, the algorithm finds, or creates, the deepest explicit node v in the path to $cw\$$, and adds as a child of v a new leaf that represents $cw\$$.

While the algorithm manages to find v in constant amortized time from u using (and maintaining) the so-called “Weiner links” (which are the inverse of the suffix links), let us reason as if we found v by descending from the root with the symbols of $cw\$$. We would descend by the path until we find an explicit or implicit node v that is the locus of a prefix cx of $cw\$$ from where no child descends by label a , where cxa also prefixes $cw\$$. Those are the strings cx and cxa alluded to in Lemma 3. The algorithm then:

- If v is implicit, converts it into an explicit node with (temporarily) a single child, which descends by label, say, b .
- Creates a new child of v by label $a \neq b$, which is a leaf representing $cw\$$.

By Lemma 3, cx is right-maximal in $cw\$$ (this is also clear because the explicit node v is its locus). To obtain $\mathcal{S}_r(cw\$)$, we need to add to $\mathcal{S}_r(w\$)$ at most two right-extensions: cxa , which is represented by the child of v by label a and, in case v was originally implicit, cxb , the other child of v . We maintain $\mathcal{S}_r$ by marking the edges leaving from the corresponding right-maximal strings. Consequently, we now mark the two edges that leave from v . On the other hand, if xa and/or xb were in $\mathcal{S}_r(w\$)$, they must be removed to form $\mathcal{S}_r(cw\$)$ as they are now suffixes of cxa and cxb . We thus unmark, if marked, the two children of u that descend by a and b (both of which exist).

A suffixient set of string positions can be inferred from the edges we have marked at each point.

As Ukkonen’s, Weiner’s algorithm is linear-time under the same assumptions of a transdichotomous RAM model with computer word size $\Omega(\log n)$; otherwise a factor of $O(\log |\Sigma|)$ multiplies the time complexity.

Theorem 3. *There exists an algorithm to compute smallest suffixient sets that processes a $T\$$ right to left such that, for every n , after having processed the suffix $w[1..n]\$$ of $T\$$ it has determined a smallest suffixient set of $w\$$, and its size $\chi(w)$, using $O(n)$ space and $O(n)$ worst-case time.*

8. Conclusions and Open Questions

We have contributed to the understanding of χ as a new measure of repetitiveness, better finding its place among more studied ones. Figure 1

shows the (now) known relations around χ (cf. [3]). We also proved various additive and multiplicative lower and upper bounds on the sensitivity of χ to various string operations: appends/prepends, general edits, rotations, and reversals. As a direct consequence of our results, we derive new simple linear-time online algorithms to compute smallest suffixient sets, and thus χ , as a modification of Ukkonen’s and Weiner’s suffix tree constructions [13, 12].

There are still many interesting open questions about χ . As a follow-up to our conference paper, it has been recently shown that our bound $\chi \leq 2r$ is tight [35]. It was also proved recently that χ is reachable [8], a major breakthrough. Nevertheless, it is still not known if $\chi = \Omega(b)$.

One consequence of Corollary 7 is that $\chi \notin O(g \log^k(n/g))$ for any $k \geq 0$. It could be the case, though, that $\chi \in O(\delta \log n)$, because the separation of χ and δ on de Bruijn sequences is a $\Theta(\log n)$ factor.

Regarding edit operations, we proved a lower bound of $\Omega(\sqrt{n})$ to their additive sensitivity. A trivial upper bound is $O(n)$ because $\chi \leq n$, but we conjecture that the sensitivity is indeed $\Theta(\sqrt{n})$. Proving or disproving this conjecture is an open problem (per our results, it suffices to prove the bound $O(\sqrt{n})$ for rotations, deletions, or substitutions). With respect to multiplicative sensitivity, experiments suggest that $\mathbf{sre}(w')/\mathbf{sre}(w)$ is $O(1)$ for all the string operations we considered. We proved such a multiplicative constant for reversals. Proving the same for insertions would imply the existence of a constant for rotation and vice versa. It is also open whether $r = O(\chi \log n)$. If this were true—and provided that χ has $O(1)$ multiplicative sensitivity to string operations—it would imply that r has $O(\log n)$ multiplicative sensitivity to these operations, making the already known lower bounds on multiplicative sensitivity [15, 18, 19] tight. If the conjecture were false, then χ could be considerably smaller than r in some string families. In the same line, observe that Lemma 11 implies that, either $r(w)/r(w^R) = \omega(\log n)$ and r can be $\omega(\chi \log n)$, or $r(w)/r(w^R) = O(\log n)$ and this known bound is tight.

Acknowledgements

We thank Davide Cenzato, Francisco Olivares, and Nicola Prezza, for useful discussions on suffixient sets, and for their code to compute smallest suffixient sets <https://github.com/regindex/suffixient> [11], which was helpful to propose and discard hypotheses on the behavior of χ , and to verify the implementation of our Ukkonen’s extension that computes them online. We also thank Gregory Kucherov and Dominik Köppl for pointing out a bug in an early version of our Weiner’s extension.

Disclosure of interests

The authors have no competing interests to declare that are relevant to the content of this article.

Funding

Open access funding provided by Università degli Studi di Palermo within the CRUI-CARE Agreement.

H.F. was supported by JST BOOST, Japan Grant Number JPMJBS2406.

G.N. and C.U. were partially funded by Basal Funds FB0001 and AFB240001, ANID, Chile; and Fondecyt grants 1230755 and 1260080, ANID, Chile.

G.R. was partially supported by the project “ACoMPA – Algorithmic and Combinatorial Methods for Pangenome Analysis” (CUP B73C24001050001) funded by NextGeneration EU programme PNRR MUR M4 C2 Inv. 1.5 - Project ECS00000017 Tuscany Health Ecosystem (Spoke 6), CUP Master B63C22000680007, and by the INdAM - GNCS Project CUP_E53C25002010001.

C.U. was supported by the Polish National Science Center, grant no. 2022/46/E/ST6/00463; ANID-Subdirección de Capital Humano/Doctorado Nacional/2021-21210580, ANID, Chile; and NIC Chile Doctoral Scholarship, NIC, Chile.

References

- [1] G. Navarro, Indexing highly repetitive string collections, part I: Repetitiveness measures, *ACM Computing Surveys* 54 (2) (2021) article 29. doi:10.1145/3434399.
- [2] G. Navarro, Indexing highly repetitive string collections, part II: Compressed indexes, *ACM Computing Surveys* 54 (2) (2021) article 26. doi:10.1145/3432999.
- [3] G. Navarro, Indexing highly repetitive string collections, *CoRR* 2004.02781 (2022). doi:10.48550/arXiv.2004.02781.
- [4] L. Depuydt, T. Gagie, B. Langmead, G. Manzini, N. Prezza, Suffixient sets, *CoRR* 2312.01359 (2023). doi:10.48550/arXiv.2312.01359.

- [5] D. Cenzato, L. Depuydt, T. Gagie, S.-H. Kim, G. Manzini, F. Olivares, N. Prezza, Suffixient arrays: a new efficient suffix array compression technique, CoRR 2407.18753 (2025). doi:10.48550/arXiv.2407.18753.
- [6] D. Kempa, N. Prezza, At the roots of dictionary compression: String attractors, in: Proc. 50th Annual ACM Symposium on the Theory of Computing (STOC 2018), ACM, 2018, pp. 827–840. doi:10.1145/3188745.3188814.
- [7] M. Burrows, D. Wheeler, A block sorting lossless data compression algorithm, Tech. Rep. 124, Digital Equipment Corporation (1994).
- [8] H. Shibata, H. Bannai, String representation in suffixient set size space, CoRR 2604.04377 (2026).
URL <https://arxiv.org/abs/2604.04377>
- [9] G. Navarro, C. Ochoa, N. Prezza, On the approximation ratio of ordered parsings, IEEE Transactions on Information Theory 67 (2) (2021) 1008–1026. doi:10.1109/TIT.2020.3042746.
- [10] A. Lempel, J. Ziv, On the complexity of finite sequences, IEEE Transactions on Information Theory 22 (1) (1976) 75–81. doi:10.1109/TIT.1976.1055501.
- [11] D. Cenzato, F. Olivares, N. Prezza, On computing the smallest suffixient set, in: Proc. 31st International Symposium on String Processing and Information Retrieval (SPIRE 2024), Vol. 14899 of Lecture Notes in Computer Science, Springer, 2024, pp. 73–87. doi:10.1007/978-3-031-72200-4_6.
- [12] P. Weiner, Linear pattern matching algorithm, in: Proc. 14th Annual IEEE Symposium on Switching and Automata Theory, 1973, pp. 1–11.
- [13] E. Ukkonen, On-line construction of suffix trees, Algorithmica 14 (3) (1995) 249–260.
- [14] G. Navarro, G. Romana, C. Urbina, Smallest suffixient sets as a repetitiveness measure, in: Proc. 32nd International Symposium on String

- Processing and Information Retrieval (SPIRE 2025), Vol. 16073 of Lecture Notes in Computer Science, Springer, 2025, pp. 217–232. doi: https://doi.org/10.1007/978-3-032-05228-5_18.
- [15] T. Akagi, M. Funakoshi, S. Inenaga, Sensitivity of string compressors and repetitiveness measures, *Information and Computation* 291 (2023) 104999. doi:10.1016/j.ic.2022.104999.
 - [16] G. Fici, G. Romana, M. Sciortino, C. Urbina, On the impact of morphisms on BWT-runs, in: *Proc. 34th Annual Symposium on Combinatorial Pattern Matching (CPM 2023)*, Vol. 259 of *Leibniz International Proceedings in Informatics*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 10:1–10:18. doi:10.4230/LIPIcs.CPM.2023.10.
 - [17] G. Fici, G. Romana, M. Sciortino, C. Urbina, Morphisms and BWT-Run Sensitivity, in: *50th International Symposium on Mathematical Foundations of Computer Science (MFCS 2025)*, Vol. 345 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, pp. 49:1–49:18. doi:10.4230/LIPIcs.MFCS.2025.49.
 - [18] S. Giuliani, S. Inenaga, Z. Lipták, N. Prezsa, M. Sciortino, A. Toffanello, Novel results on the number of runs of the Burrows-Wheeler Transform, in: *Proc. 47th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2021)*, Vol. 12607 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 249–262. doi:10.1007/978-3-030-67731-2_18.
 - [19] S. Giuliani, S. Inenaga, Z. Lipták, G. Romana, M. Sciortino, C. Urbina, Bit catastrophes for the Burrows-Wheeler transform, *Theory of Computing Systems* 69 (2) (2025) 19. doi:10.1007/s00224-024-10212-9.
 - [20] S. Mantaci, A. Restivo, G. Romana, G. Rosone, M. Sciortino, A combinatorial view on string attractors, *Theoretical Computer Science* 850 (2021) 236–248. doi:10.1016/j.tcs.2020.11.006.
 - [21] G. Navarro, F. Olivares, C. Urbina, Generalized straight-line programs, *Acta Informatica* 62 (1) (2025) 14. doi:10.1007/s00236-025-00481-3.

- [22] G. Navarro, C. Urbina, Repetitiveness measures based on string morphisms, *Theoretical Computer Science* 1043 (2025) 115259. doi:10.1016/j.tcs.2025.115259.
- [23] M. Lothaire, *Algebraic Combinatorics on Words*, Encyclopedia of Mathematics and its Applications, Cambridge University Press, New York, NY, USA, 2002. doi:10.1017/CB09781107326019.
- [24] D. Kempa, T. Kociumaka, Resolution of the Burrows-Wheeler transform conjecture, *Communications of the ACM* 65 (6) (2022) 91–98. doi:10.1145/3531445.
- [25] V. T. V. Date, L. M. Zatesko, On the near-tightness of $\chi \leq 2r$: a general σ -ary construction and a binary case via LFSRs, in: 17th Latin American Theoretical Informatics Symposium(LATIN), 2026, to appear.
- [26] U. Manber, E. W. Myers, Suffix arrays: A new method for on-line string searches, *SIAM Journal on Computing* 22 (5) (1993) 935–948. doi:10.1137/0222058.
- [27] X. Droubay, J. Justin, G. Pirillo, Episturmian words and some constructions of de Luca and Rauzy, *Theoretical Computer Science* 255 (1) (2001) 539–553. doi:10.1016/S0304-3975(99)00320-5.
- [28] A. Glen, J. Justin, Episturmian words: a survey, *RAIRO - Theoretical Informatics and Applications* 43 (3) (2009) 403–442. doi:10.1051/ita/2009003.
- [29] A. de Luca, A combinatorial property of the Fibonacci words, *Information Processing Letters* 12 (4) (1981) 193–195. doi:10.1016/0020-0190(81)90099-5.
- [30] X. Droubay, Palindromes in the Fibonacci word, *Information Processing Letters* 55 (4) (1995) 217–221. doi:10.1016/0020-0190(95)00080-V.
- [31] N. Bruijn, de, A combinatorial problem, *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam* 49 (7) (1946) 758–764.
- [32] S. Kreft, G. Navarro, On compressing and indexing repetitive sequences, *Theoretical Computer Science* 483 (2013) 115–133. doi:10.1016/j.tcs.2012.02.006.

- [33] P. Gawrychowski, M. Kosche, F. Manea, On the number of factors in the LZ-end factorization, in: Proc. 30th International Symposium on String Processing and Information Retrieval (SPIRE 2023), Vol. 14240 of Lecture Notes in Computer Science, Springer, 2023, pp. 253–259. doi:10.1007/978-3-031-43980-3_20.
- [34] M. Patrascu, M. Thorup, Dynamic integer sets with optimal rank, select, and predecessor search, in: Proc. 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS), 2014, pp. 166–175.
- [35] V. T. V. Date, L. M. Zatesko, On the near-tightness of $\chi \leq 2r$: A general σ -ary construction and a binary case via LFSRs, in: Proc. 17th Latin American Theoretical Informatics (LATIN), 2026, to appear.