Algorithms for Galois Words: Detection, Factorization, and Rotation

Diptarama Hendrian ✉  🇯🇵
Tokyo Medical and Dental University, Japan

Dominik Köppl ✉  🇯🇵
University of Yamanashi, Japan

Ryo Yoshinaka ✉  🇯🇵
Tohoku University, Japan

Ayumi Shinohara ✉  🇯🇵
Tohoku University, Japan

Abstract

Lyndon words are extensively studied in combinatorics on words – they play a crucial role on upper bounding the number of runs a word can have [Bannai+, SIAM J. Comput.’17]. We can determine Lyndon words, factorize a word into Lyndon words in lexicographically non-increasing order, and find the Lyndon rotation of a word, all in linear time within constant additional working space. A recent research interest emerged from the question of what happens when we change the lexicographic order, which is at the heart of the definition of Lyndon words. In particular, the alternating order, where the order of all odd positions becomes reversed, has been recently proposed. While a Lyndon word is, among all its cyclic rotations, the smallest one with respect to the lexicographic order, a Galois word exhibits the same property by exchanging the lexicographic order with the alternating order. Unfortunately, this exchange has a large impact on the properties Galois words exhibit, which makes it a nontrivial task to translate results from Lyndon words to Galois words. Up until now, it has only been conjectured that linear-time algorithms with constant additional working space in the spirit of Duval’s algorithm are possible for computing the Galois factorization or the Galois rotation.

Here, we affirm this conjecture as follows. Given a word $T$ of length $n$, we can determine whether $T$ is a Galois word, in $O(n)$ time with constant additional working space. Within the same complexities, we can also determine the Galois rotation of $T$, and compute the Galois factorization of $T$ online. The last result settles Open Problem 1 in [Dolce et al., TCS 2019] for Galois words.

1 Introduction

A Galois word is a word that is strictly smaller than all its cyclic rotations with respect to the so-called alternating order, where symbols at odd positions are compared in the usual lexicographic order, but symbols at the remaining positions in the opposite order. While $aab$ is clearly the smallest word among all its cyclic rotations $aba$ and $baa$ under the lexicographic order, $aab$ is larger than $aba$ under the alternating order because the $b$ in the second position is smaller than $a$. In fact, $aba$ is a Galois word. Readers familiar with Lyndon words may
identify \texttt{aab} to be, nevertheless, a Lyndon word because it is strictly smaller than all its cyclic rotations with respect to the lexicographic order. While the definition of Lyndon and Galois words only differ by the used order, the combinatorial differences are astonishing. For instance, on the one hand, Lyndon words cannot have proper borders, i.e., factors appearing both as a prefix \textit{and} as a suffix (but shorter than the word itself). On the other hand, Galois words such as \texttt{aba} can have proper borders of odd lengths [16, Proposition 3.1].

The name \textit{Galois word} has been coined by Reutenauer [16], who introduced these words and derived the naming by a bijection of Galois words and homographic classes of Galois numbers. In the same paper [16], Reutenauer defined a unique factorization of a generalization of Lyndon words, a class of words covering Galois words. Here, we call this factorization \textit{Galois factorization} since we only cover Galois words within the scope of this paper. The Galois factorization is a factorization of a word into a sequence of non-increasing Galois words. Later, Dolce et al. [5] could characterize the first factor of the Galois factorization [5, Theorem 33], and also provide a characterization of Galois words by their prefixes [5, Theorem 32]. However, Dolce et al. left it as an open problem ([5, Open Problem 1]) to find a computation algorithm similar to Duval’s algorithm [7] computing the Lyndon factorization. In this paper, we solve this problem by introducing a factorization algorithm (Algorithm 2 and Theorem 32) in the spirit of Duval’s algorithm, computing the Galois factorization in linear time with constant additional working space online.

Asides from the above results, we are only aware of the following two articles dealing with Galois words. First, Dolce et al. [6] studied generalized Lyndon-words, among them also Galois words, with respect to infinite orders. Finally, Burcroff and Winsor [3] gave a characterization of infinite generalized Lyndon words, and properties of how finite generalized Lyndon words can be infinitely extended.

2 Related Work

While we covered, to the best of our knowledge, all published results explicitly dealing with Galois words above, Galois words have a strong relation with Lyndon words and the alternating order.

\textbf{Lyndon.} Regarding the former, an exhaustive list of results would go beyond the scope of this paper. We nevertheless highlight that the Lyndon factorization (the aforementioned factorization when outputting factors that are Lyndon words) can be computed in linear time with constant additional space with Duval’s algorithm [7]. The same algorithm allows us to judge whether a word is Lyndon. Shiloach [17] gave a linear-time algorithm computing the Lyndon rotation of a primitive word $T$, i.e., its cyclic rotation that is Lyndon, in constant additional working space.

\textbf{Alternating Order.} Regarding the latter, much work focused on implementing a Burrows–Wheeler transform (BWT) [4] based on the alternating order. While the classic BWT sorts all cyclic rotations of a given input word in lexicographic order, the alternating BWT [11] sorts the cyclic rotations with respect to the alternating order. Gessel et al. [11] not only introduced this BWT variant, but also gave an inversion to restore the original word. Subsequently, Giancarlo et al. [12] gave a linear-time algorithm for computing the alternating BWT. To this end, they compute the Galois rotation of the input word $T$, i.e., the cyclic rotation of $T$ that is Galois. However, their algorithm computing the Galois rotation needs an auxiliary integer array of length $n$. Compared to space-efficient algorithms computing the classic
BWT (e.g. [15] with linear time and space linear in the bit size of the input text), this is a major bottleneck, but a necessary precomputation step of their algorithm constructing the alternating BWT. Giancarlo et al. [12] also showed how to invert the alternating BWT in linear time. In a follow-up [13], Giancarlo et al. put their discovered properties of the alternating BWT into larger context by covering BWT variants based on a generalized ordering. In that article, they also showed that the alternating BWT can be turned into a compressed self-index that supports pattern counting queries in time linear in the pattern length. The space of the index can be related with the number of symbol runs even when augmented for queries to locate all pattern occurrences in the text, by adapting r-indexing [10] techniques to the alternating BWT.

Our Contribution. This paper makes three contributions to the research on Galois words.

First, we give an algorithm (Theorem 25 and Algorithm 1) in Section 5 that checks, for a given word of length \( n \), whether this word is Galois, in \( O(n) \) time with constant additional working space. Second, we give an algorithm (Theorem 32 and Algorithm 2) in Section 6 that computes the Galois factorization in \( O(n) \) time with constant additional working space online. Finally, we show how to find the Galois rotation (Theorem 36 and Algorithm 3) in Section 7 that in \( O(n) \) time with constant additional working space online, paving the way for constructing the alternating BWT in \( o(n) \) working space.

We stress that, having an efficient Galois factorization algorithm allows us to merge indexing techniques for the alternate BWT with the bijective BWT [14, 1] to give rise to a BWT-variant that indexes Galois words, whose indexing capabilities are left as future work.

3 Preliminaries

Let \( \Sigma \) be a set of symbols called an alphabet. The set of words over \( \Sigma \) is denoted by \( \Sigma^* \). The empty word is denoted by \( \varepsilon \). The length of a word \( W \in \Sigma^* \) is denoted by \( |W| \). The \( i \)-th symbol of a word \( W \) is denoted by \( W[i] \) for \( 1 \leq i \leq |W| \) and the factor of \( W \) that begins at position \( i \) and ends at position \( j \) is \( W[i..j] \) for \( 1 \leq i \leq j \leq |W| \). We define \( W[i..j] = \varepsilon \) if \( i > j \).

A word \( B \) is a border of \( W \) if \( B \) is a prefix and a suffix of \( W \). We say a border \( B \) of \( W \) is proper if \( B \neq W \). For a word \( T \) we call an integer \( p \in [1..|T|] \) a period of \( T \) if \( T[i+p] = T[i] \) for all \( i \in [1..|T| - p] \). In particular, \(|T| \) is always a period of \( T \). Let \( \text{Per}_o(W) \) and \( \text{Per}_e(W) \) be the shortest odd and even period of \( W \) if any, respectively. We set \( \text{Per}_o(W) = |W| + 1 \) or \( \text{Per}_e(W) = |W| + 1 \) if \( W \) does not have an odd or an even period, respectively. Since the length of a word itself is a period, a word of odd length always has an odd period and a word of even length always has an even period. For a rational number \( \alpha \), let \( W^\alpha \) be the word obtained by concatenating \( W \) \( \alpha \) times. Let \( W^\omega \) be the infinite repetition of \( W \). We call a word \( V \in \Sigma^* \) primitive if the fact \( V = U^k \) for some word \( U \in \Sigma^* \) and an integer \( k \geq 1 \) implies \( V = U \) and \( k = 1 \). We say that two words \( X \) and \( Y \) have the same length-parity if \( |X| \mod 2 = |Y| \mod 2 \), i.e., their lengths are both either odd or even.

We define the standard lexicographic order over words with \( \prec_{\text{lex}} \). We define the alternating order on words as follows: Given two distinct words \( S \) and \( T \) such that \( S^\omega \neq T^\omega \), with the first mismatching symbol pair at a position \( j \), i.e., \( S^\omega[1..j-1] = T^\omega[1..j-1] \) and \( S^\omega[j] \neq T^\omega[j] \), we write \( S \prec_{\text{alt}} T \) if either (a) \( j \) is odd and \( S^\omega[j] < T^\omega[j] \), or (b) \( j \) is even and \( S^\omega[j] > T^\omega[j] \).

In addition we denote by \( S =_{\text{alt}} T \) if \( S^\omega = T^\omega \). For instance, \( \text{aba} \prec_{\text{alt}} \text{aab} \) but \( \text{aab} \prec_{\text{alt}} \text{aba} \); \( \text{b} \prec_{\text{lex}} \text{bba} \) but \( \text{bba} \prec_{\text{alt}} \text{b} \); \( \text{aba} =_{\text{alt}} \text{ababa} \). We define \( \varepsilon \succ_{\text{alt}} X \) for all \( X \in \Sigma^+ \). We denote by \( S \preceq_{\text{alt}} T \) if either \( S \prec_{\text{alt}} T \) or \( S =_{\text{alt}} T \). We further write \( S \sqsubseteq_{\text{alt}} T \) if \( S \prec_{\text{alt}} T \) but neither \( S \) is a prefix of \( T \) nor vice versa.
We introduce $S \sqsubseteq_{alt} T$ for the following reason: For two words $S$ and $T$ with $S \prec_{alt} T$, it is generally not true that $SU \prec_{alt} TU$ for all words $U$ (e.g., $ab \prec_{alt} aba$ but $abac \prec_{alt} abc$). However, for $\sqsubseteq_{alt}$ we have:

**Fact 1.** For two words $S$ and $T$ with $S \sqsubseteq_{alt} T$, it holds that $SU \sqsubseteq_{alt} TU$ for all words $U$.

We also make use of the following additional facts:

**Fact 2.** For two words $S$ and $T$ with $S^\omega = T^\omega$, there exists a primitive word $U$ integers $a$ and $b$ such that $S = U^a$ and $T = U^b$.

**Fact 3.** $T$ is non-primitive if and only if $|T|/p$ is an integer of at least two for $p$ being $T$'s smallest period. If $\text{Per}_o(T) = |T|$, then $T$ is primitive.

**Example 4.** We cannot switch $\text{Per}_o$ with $\text{Per}_e$ in Fact 3. A counter-example is the non-primitive word $T_1 = abaaba$, for which we have $\text{Per}_e(T_1) = 3$ but $\text{Per}_e(T_1) = 6$. Also, for $T_2 = aa$ we have $\text{Per}_e(T_2) = 1$ but $\text{Per}_e(T_2) = 2$.

The following property holds for any two periods of a word.

**Lemma 5** (\[9\]). Let $p$ and $q$ be periods of a word $T$. If $p + q - r \leq |T|$, then $r$ is also a period of $T$, where $r$ is the greatest common divisor of $p$ and $q$.

A word is called Galois if it is, among all its cyclic rotations, the smallest with respect to $\prec_{alt}$. By definition, a Galois word has to be primitive (otherwise, it has an identical cyclic rotation that is not strictly larger). The following properties hold of Galois words.

**Lemma 6** ([5, Theorem 14]). A primitive word $T$ is Galois if and only if $T$ is smaller than all its suffixes, with respect to $\prec_{alt}$.

**Lemma 7** ([5, Theorem 32]). A word $T$ is Galois if and only if for any factorization $T = UV$ with $U, V \in \Sigma^+$, one of the following condition holds: (1) $U \prec_{alt} T$ if $|U|$ is even; (2) $U \succ_{alt} T$ if $|U|$ is odd.

**Lemma 8** ([5, Lemma 35], [16, Proposition 3.1]). If a Galois word $T$ has a proper border $B$, then the length of $B$ is odd.

For example $aba$ and $abba$ are Galois words with a proper border. Unlike for Lyndon words (cf. [7, Proposition 1.3]), it does not hold that, if $U$ and $V$ are Galois words then $UV$ is Galois if $U \prec_{alt} V$. For instance, $aba \prec_{alt} c$ but $abac$ is not Galois because $ac \prec_{alt} abac$.

## 4 Characteristics of Pre-Galois Words

In this section, we define pre-Galois words and study their properties. The observations we make here will lead us to helpful tools that we will leverage for proposing the three algorithms in the subsequent sections, namely 1. determining Galois words, 2. computing the Galois factorization, and 3. computing the Galois rotation of a word.

**Definition 9** (Pre-Galois word). A word $T$ is a pre-Galois word if every proper suffix $S$ of $T$ satisfies one or both of the following conditions: (1) $S$ is a prefix of $T$; (2) $S \succ_{alt} T$.

In particular, a Galois word is pre-Galois. However, the converse is in general not true; for example $T = abab$ is pre-Galois but not Galois because $ab \prec_{alt} abaab$. In what follows, we introduce a basic property of pre-Galois words.
Lemma 10. Let $U$ be a word that is not pre-Galois. Then, for any word $V$, $UV$ is not pre-Galois. The contraposition is that any prefix of a pre-Galois word is pre-Galois.

Proof. By definition there exists a proper suffix $S$ of $U$ such that $S \sqsubseteq_{alt} U$. By Fact 1, $S \cdot V \sqsubseteq_{alt} U \cdot V$ holds.

Next, we study properties of periods of pre-Galois words.

Lemma 11. Let $T$ be a pre-Galois word that has an odd period. Let $p_o = \text{Per}_o(T)$ be the shortest odd period of $T$. Then $T[1..p_o]$ is Galois.

Proof. Let $U = T[1..p_o]$ and $T = UV$. By Lemma 10, $U$ is pre-Galois. Assume $U$ is not Galois. Then there exists a proper suffix $S$ of $U$ such that $S$ is a prefix of $U$ and $S \sqsubseteq_{alt} U$.

Since $p_o$ is odd and the shortest odd period of $T$, $U$ is primitive according to Fact 3, and we obtain two observations: First, by Fact 2, if $S^e = U^\omega$ then $S = U$, a contradiction to $S$ being proper. Hence, $S \not\sqsubseteq_{alt} U$ must hold.

Second, there exists a rational number $\alpha \geq 1$ and a symbol $c \in \Sigma$ such that $S^\alpha c$ is a prefix of $U$, $S^\alpha[1..|S^\alpha c|] \sqsubseteq_{alt} S^\alpha c$, and $|S^\alpha c| < |U|$. See the left of Figure 1 for a sketch. By definition, we have $S^{\alpha - 1} = U[1..|S^{\alpha - 1}|]$. If $|S|$ is odd, we have $T[|S| + 1..|T|] \sqsubseteq_{alt} T$, which implies that $T$ is not pre-Galois. Otherwise, if $|S|$ is even, $|U| + |S^{\alpha - 1}c| \leq |T|$ since $p_o$ is the shortest odd period of $T$ and therefore $|T| \geq 2|U|$ with $|U| \geq |S^{\alpha - 1}c|$. Here, we have $T[|U| - |S| + 1..|U| + |S^{\alpha - 1}c|] = S^\alpha[1..|S^\alpha c|]$. Therefore, $T[|U| - |S| + 1..|U| + |S^{\alpha - 1}c|] \sqsubseteq_{alt} T[1..|S^\alpha c|]$, which implies $T$ is not pre-Galois.

A similar property also holds for even periods.

Lemma 12. Let $T$ be a pre-Galois word that has an even period. Let $p_e = \text{Per}_e(T)$ be the shortest even period of $T$. Then $T[1..p_e]$ is Galois if primitive.

Proof. We follow the proof of Lemma 11 by replacing $U$ there with $T[1..p_e]$. We also give there $p_e$ the role of $p_o$. We can do that because we assume that $U$ is primitive, so we obtain a proper border $S$ of $U$ like in the previous proof.

We are in particular interested in prefixes of pre-Galois words that are Galois. To formalize this idea, we define Galois roots of a pre-Galois word as follows.

Definition 13 (Galois root). Let $P$ be a prefix of a pre-Galois word $T$. We call $P$ a Galois root of $T$ if $|P|$ is a period of $T$ and $P$ is Galois.
In addition to our aforementioned example $T = \text{abaab}$, $\text{aba}$ is a Galois root of $T$. Also, the words $T[1..p_2]$ in Lemma 11 and $T[1..p_3]$ Lemma 12, are Galois roots of $T$ if they are, respectively, primitive. Note that a pre-Galois word $T$ has at least one Galois root, namely $T$’s prefix of length equal to $T$’s shortest period. While a pre-Lyndon word has exactly one Lyndon root, a pre-Galois word can have two different Galois roots:

▶ **Lemma 14.** A pre-Galois word $T$ can have at most two Galois roots, and their lengths have different parities.

**Proof.** Assume that there are two Galois roots $G_1$ and $G_2$ with the same length-parity. Then the length difference of $G_1$ and $G_2$ is even. Without loss of generality, suppose that $|G_1| < |G_2|$. Then the suffix $X = G_2||G_1| + 1.$ of $G_2$ is also a prefix of $G_2$ since $T = G_2\|G_1$ for rational numbers $\alpha$ and $\beta$. Hence, $X$ is a border of $G_2$ with even length, which is impossible due to Lemma 8. See the right of Figure 1 for a sketch. ◀

In what follows, we name the odd-length and the even-length Galois root, if they exist, by $G_o$ and $G_e$, respectively. By Lemma 14, they are well-defined. For example, consider $T = \text{aba}$. The two prefixes $ab$ and $\text{aba}$ are both Galois, for which $T = (\text{ab})^{3/2} = (\text{aba})^1$.

From Lemma 12, $T[1..p_e]$ is Galois only when it is primitive. Next, we consider the case where $T[1..p_o]$ is not primitive.

▶ **Lemma 15.** Let $T$ be an even-length pre-Galois word with no even-length Galois root, i.e., $T[1..p_e]$ is not primitive, where $p_e = \text{Per}_e(T)$. Then there exists $G_o = T[1..p_o]$ such that $T = G_o^2G'_o$ with $k \geq 2$, where $p_o = \text{Per}_o(T)$ and $G'_o$ is a prefix of $G_o$.

**Proof.** Since $|T|$ is even, $T$ has a period of even length. Let $p_e$ be the shortest even period of $T$. By Lemma 12, $U = T[1..p_e]$ is Galois if $U$ is primitive. Since $U$ is not primitive and $p_e$ is the smallest even period of $T$, we have $\text{Per}_o(T) = p_o = p_e/2$. Thus, there is an odd-length prefix $G_o = T[1..p_o]$ of $U$ such that $U = G_o^2$. ◀

▶ **Lemma 16.** Let $G_o$ be an odd-length Galois root of a pre-Galois word $T = G_o^kG'_o$ with $k \geq 2$ and $G'_o$ is a prefix of $G_o$. Then $T$ has no even-length Galois root.

**Proof.** Since $T = G_o^kG'_o$, $2|G_o|$ is a period of $T$. Assume that $T$ has a shorter even period $p_e < 2|G_o|$. By Lemma 8, $G_o$ does not have a proper border of even length. Because the two conditions (a) $G_oG_o$ has even length and (b) $p_e \in [|G_o| + 1..2|G_o| - 1]$ would imply that $G_oG_o$ (and thus $G_o$ due to the length of $p_e$) has a border of even length, $p_e$ must be less than $|G_o|$. However, by the periodicity lemma (Lemma 5), there exists an odd period shorter than $G_o$, which contradicts that $|G_o|$ is the shortest odd-length Galois period of $T$. ◀

▶ **Example 17.** Let $T = \text{aba}$ be an even-length Galois word. $T$ has the odd-length Galois root $\text{aba}$. By appending $b$ to $T$, we obtain $\text{abaab}$, which is pre-Galois with no even-length Galois root. $T \cdot b$ can be written as $(\text{aba})^{5/3}$, a fractional repetition of $\text{aba}$.

By Lemmas 15 and 16, if $T$ has an even period, $T[1..p_e]$ is either $G_o$ or $G_o^2$.

## 5 Determining Galois Words

The algorithm we propose checks if a word $T$ is Galois by reading $T$ from left to right. For that, we want to maintain the Galois roots of the prefix of $T$ read so far. To this end, we study the Galois roots of $T' = T \cdot z$, i.e., when appending a symbol $z$ to $T$. Our main observation can be stated as follows:
Theorem 18. Let $T$ be a pre-Galois word, $p_o = \text{Per}_o(T)$, and $p_e = \text{Per}_e(T)$. Given a symbol $z$, the extension $T' = T \cdot z$ is a pre-Galois word if and only if both conditions $T'[1..|T| - p_o + 1] \preceq_a \text{Alt} T'[p_o + 1..|T| + 1]$ and $T'[1..|T| - p_e + 1] \preceq_a \text{Alt} T'[p_e + 1..|T| + 1]$ hold.

In what follows, we break down the statement of this theorem into two lemmas for each direction. First, we consider the case where $T'$ cannot be a pre-Galois word.

Lemma 19. Let $T'$ be a pre-Galois word, $p_o = \text{Per}_o(T)$, and $p_e = \text{Per}_e(T)$. Consider a symbol $z$ and the extension $T' = T \cdot z$, such that either $T'[1..|T| - p_o + 1] \preceq_a T'[p_o + 1..|T| + 1]$ or $T'[1..|T| - p_e + 1] \preceq_a T'[p_e + 1..|T| + 1]$. Then the extension $T'$ is not a pre-Galois word.

Proof. We treat here only the case involving $p_o$ because the other case involving $p_e$ can be proved similarly. If $p_o = |T| + 1$, $T'[1..|T| - p_o + 1] = T'[p_o + 1..|T| + 1] = \varepsilon$. Thus, this case does not meet the requirements of the lemma statement. It remains to consider $p_o \leq |T|$. For that, suppose $T'[1..|T| - p_o + 1] \preceq_a T'[p_o + 1..|T| + 1]$. Since $T'[1..|T| - p_o + 1] \neq T'[p_o + 1..|T| + 1]$, $T'[p_o + 1..|T| + 1]$ not a prefix of $T'$. Thus, $T'$ is not pre-Galois.

For all other cases, we show that $T'$ is a pre-Galois word.

Lemma 20. Let $T'$ be a pre-Galois word, $p_o = \text{Per}_o(T)$, and $p_e = \text{Per}_e(T)$. Consider a symbol $z$ and the extension $T' = T \cdot z$, such that $T'[1..|T| - p_o + 1] \preceq_a T'[p_o + 1..|T| + 1]$ and $T'[1..|T| - p_e + 1] \preceq_a T'[p_e + 1..|T| + 1]$. Then the extension $T'$ is a pre-Galois word.

Proof. We prove the contraposition, i.e., if $T'$ is not a pre-Galois word, either $T'[1..|T| - p_o + 1] \not\preceq_a T'[p_o + 1..|T| + 1]$ or $T'[1..|T| - p_e + 1] \not\preceq_a T'[p_e + 1..|T| + 1]$ holds.

Suppose $T'$ is not a pre-Galois word. Since $T$ is pre-Galois and $T'$ is not pre-Galois, there exists a suffix $S$ of $T$ such that $S$ is a prefix of $T$ but $S \cdot z$ is not a prefix of $T'$ and $S \cdot z \npreceq_a T'$, i.e., $S = T'[1..|S|]$ and $S \cdot z \nsubseteq T'[1..|S| + 1]$. In what follows we consider three cases. The first case is that $S$ is the empty word. But then $z \npreceq_a T'[1]$, and therefore $T'$ is not pre-Galois. The other cases concern the length-parity of $T$ and $S$.

Consider $T$ and $S$ have the same length-parity. Since $p_e$ is even, $T'[1..|T| - p_e]$ and $S$ also have the same length-parity. Since $S$ is a proper border of $T$, $p_e \leq |T| - |S|$. Here we have $S = T'[1..|S|] = T'[|T| - p_e - |S| + 1..|T| - p_e]$. Let $x = T'[|S| + 1]$ and $y = T'[|T| - p_e + 1]$. See Figure 2 for a sketch. Since $|T|$ is pre-Galois, we have $S \cdot y \succeq_a T \cdot x$. Moreover, $S \cdot x \succeq_a T \cdot z$. Thus we have $S \cdot y \succeq_a S \cdot z$. Since $T'[1..|T| - p_e]$ and $|S|$ have the same parity, we have $T'[1..|T| - p_e + 1] \succeq_a T'[p_e + 1..|T| + 1]$.

Consider $T$ and $S$ have different length-parity. Since $p_o$ is odd, $T'[1..|T| - p_o]$ and $|S|$ have the same parity. Note that $S$ is a proper border of $T$ thus $p_o \leq |T| - |S|$. Here we have $S = T'[1..|S|] = T'[|T| - p_o - |S| + 1..|T| - p_o]$. Let $x = T'[|S| + 1]$ and $y = T'[|T| - p_o + 1]$. Since $|T|$ is pre-Galois, we have $S \cdot y \succeq_a S \cdot x$. Moreover $S \cdot x \succeq_a T \cdot z$ holds by $T'[1..|S| + 1] \succeq_a T \cdot z$. Thus we have $S \cdot y \succeq_a T \cdot z$. Since $T'[1..|T| - p_o]$ and $|S|$ have the same parity, we have $T'[1..|T| - p_o + 1] \succeq_a T'[p_o + 1..|T| + 1]$. ▶
Next we show how periods change when we append a symbol to a pre-Galois word $T$. Here, we focus on $p_o$ first. The cases for $p_e$ can be proven in a similar way. The claim of the first lemma follows by definition.

**Lemma 21.** Let $T$ be a pre-Galois word and $p_o = \text{Per}_o(T)$. Consider a symbol $z$ and the extension $T' = T \cdot z$, such that $z = T'[|T| - p_o + 1]$. Then the extension $T'$ has $\text{Per}_o(T') = p_o$.

**Lemma 22.** Let $T$ be a pre-Galois word and $p_o = \text{Per}_o(T)$. Consider a symbol $z$ and the extension $T' = T \cdot z$ with $T'[1..|T| - p_o + 1] \prec_{alt} T'[p_o + 1..|T| + 1]$. Then,

$$\text{Per}_o(T') = \begin{cases} |T'| & \text{if } |T'| \text{ is odd}, \\ |T'| + 1 & \text{otherwise}. \end{cases}$$

**Proof.** If $|T|$ has no odd period, i.e., $p_o = |T| + 1 = |T|$, then $|T|$ is even. Thus, $|T'|$ is odd and $\text{Per}_o(T') = |T'|$. Otherwise, suppose that $|T|$ has an odd period. Assume $T'$ has odd period $p_o' < |T'|$. Thus, we have $T'[1..|T| - p_o' + 1] = T'[p_o' + 1..|T| + 1]$. Let $S = T'[1..|T| - p_o]$, $y = T'[|T| - p_o + 1]$, and $q = |T'|[|T| - p_o + 1]$. Since $T$ is pre-Galois, we have $S \cdot y \preceq_{alt} S \cdot z$. However, by $T'[1..|T| - p_o + 1] \prec_{alt} S \cdot y$, which is a contradiction. Therefore, $T'$ has no odd period $p_o'$ with $p_o' < |T'|$, which implies $\text{Per}_o(T') = |T'|$ if $|T'|$ is odd or $\text{Per}_o(T') = |T'| + 1$ if $|T'|$ is even.

In a similar way, we can show the following lemmas.

**Lemma 23.** Let $T$ be a pre-Galois word and $p_e = \text{Per}_e(T)$. Consider a symbol $z$ and the extension $T' = T \cdot z$, such that $z = T'[|T| - p_e + 1]$. Then the extension $T'$ has $\text{Per}_e(T') = p_e$.

**Lemma 24.** Let $T$ be a pre-Galois word and $p_e = \text{Per}_e(T)$. Consider a symbol $z$ and the extension $T'[1..|T| - p_e + 1] \prec_{alt} T'[p_e + 1..|T| + 1]$. Then,

$$\text{Per}_e(T') = \begin{cases} |T'| & \text{if } |T'| \text{ is even}, \\ |T'| + 1 & \text{otherwise}. \end{cases}$$

With Algorithm 1 we give algorithmic instructions in how to verify whether an input word $T$ is Galois. For each position in $T$, the algorithm performs a constant number of symbol comparisons on $T$. Storing only the two periods $p_e$ and $p_o$ of the processed prefix up so far, it thus runs in linear time with a constant number of words extra to the input word $T$. We obtain the following theorem.

**Theorem 25.** Given a word $T$, we can verify whether $T$ is Galois in $O(|T|)$ time with $O(1)$ working space.

## 6 Computing the Galois Factorization Online

In this section we present an online algorithm for computing the Galois factorization of a given word. We first start with a formal definition of the Galois factorization, introduce a key property called $\text{SPref}(T)$, and then show how to compute $\text{SPref}(T)$. The Galois factorization of a word $T$ is defined as follows.

**Definition 26.** (Galois factorization). A factorization $T = G_1 \cdot G_2 \cdots G_k$ is the Galois factorization of $T$ if $G_i$ is Galois for $1 \leq i \leq k$ and $G_1 \succeq_{alt} G_2 \succeq_{alt} \cdots \succeq_{alt} G_k$ holds.

It is known that every word admits just one Galois factorization (see [16, Théorème 2.1] or [5]). We denote the Galois factorization $T = G_1 \cdot G_2 \cdots G_k$ of $T$ by $\text{GF}(T) = (G_1, G_2, \ldots, G_k)$. The Galois factorization has the following property.
Then we have 

or (b)

of a word (time). Then, for any word 

where the integer \(z\) symbol

Lemma 28. Let \(\text{GF}(T) = (G_1, G_2, \ldots, G_k)\) be the Galois factorization of a word \(T\) of length \(n\). Let \(P\) be the shortest non-empty prefix of \(T\) such that 

\[
P \not\equiv_{alt} T \text{ if } |P| \text{ is even and } P \equiv_{alt} T \text{ if } |P| \text{ is odd. (1)}
\]

Then we have 

\[
P = \begin{cases} 
G_1^m & \text{if } |G_1| \text{ is odd, } m \text{ is even, and } m < k, \\
G_1 & \text{otherwise,}
\end{cases}
\]

where the integer \(m\) is the multiplicity of \(G_1\), i.e., \(G_i = G_1\) for \(i \leq m\), but \(G_{m+1} \neq G_1\).

We denote such \(P\) in Lemma 27 by \(\text{SPref}(T)\). If we can compute \(\text{SPref}(T)\) for any word \(T\), we can compute the Galois factorization of \(T\) by recursively computing \(\text{SPref}(T)\) from the suffix remaining when removing the prefix \(\text{SPref}(T)\). To this end, we present a way to compute \(\text{SPref}(T)\) by using the periods of \(T\).

Lemma 28. Let \(T\) be a pre-Galois word, \(p_o = \text{Per}_o(T)\), and \(p_c = \text{Per}_c(T)\). Consider a symbol \(z\) and the extension \(T' = T \cdot z\), such that (a) \(T'[1..|T| - p_o + 1] \not\equiv_{alt} T'[p_o + 1..|T| + 1] \) or (b) \(T'[1..|T| - p_c + 1] \not\equiv_{alt} T'[p_c + 1..|T| + 1] \) hold (both (a) and (b) can hold at the same time). Then, for any word \(S\), we have \(\text{SPref}(T'S) = T'[1..p]\) such that 

\[
p = \begin{cases} 
\min\{p_o, p_c\} & \text{if } T'[1..|T| - p_o + 1] \not\equiv_{alt} T'[p_o + 1..|T| + 1] \\
& \text{and } T'[1..|T| - p_c + 1] \not\equiv_{alt} T'[p_c + 1..|T| + 1], \text{ (a) and (b))} \\
p_o & \text{if } T'[1..|T| - p_c + 1] \equiv_{alt} T'[p_c + 1..|T| + 1], \text{ (b) but not (a))} \\
p_c & \text{otherwise. (a) but not (b))}
\end{cases}
\]

Algorithm 1 Determining whether a word is Galois, see Theorem 25.

1. **Function isGalois(T)** // Assume \(|T| \geq 2\), otherwise always true
2. \(p_o = 1; p_c = 2;\)
3. for \(i\) from 2 to \(|T|\) do // Loop-Invariant: \(T[1..i-1]\) is pre-Galois
   4. if \(i\) is odd then
      5. if \(p_c < i\) then
         6. if \(T[i] < T[i - p_c]\) then return False; // Lemma 19
         7. else if \(T[i] > T[i - p_c]\) then \(p_c = i + 1;\) // Lemma 24
      8. if \(p_o < i\) then
         9. if \(T[i] < T[i - p_o]\) then \(p_o = i;\) // Lemma 22
        10. else if \(T[i] > T[i - p_o]\) then return False; // Lemma 19
    11. else
        12. if \(p_o = |T|\) then // Is \(T\) primitive?
           13. return True // \(T\) is Galois by Lemma 11
        14. else if \(p_o = |T|\) and \(p_c \neq 2p_o\) then
           15. return True // \(T\) is pre-Galois but not primitive (hence not Galois)

   return False // \(T\) is pre-Galois but not primitive (hence not Galois)
Figure 3 Sketch of the proof of Lemma 28 (left) and of Lemma 29 (right).

**Proof.** Let $W = T^s S$ for some arbitrary word $W \in \Sigma^*$. Suppose $T'[1..|T| - p_o + 1] \succ_T T'[p_o + 1..|T| + 1]$ holds. Let $x = T'[|T| - p_o + 1]$. Since $T$ and $T'[1..|T| - p_o]$ have different length-parities, we have $T \cdot x \prec_T T \cdot z$, which implies $T'[1..p_o] \prec_T W$. See the left of Figure 3 for a sketch. Similarly, we can show that $T'[1..p_o] \prec_T W$ if $T'[1..|T| - p_o + 1] \succ_T T'[p_o + 1..|T| + 1]$.

Next, we show the minimality of $p$. Consider a prefix $X$ such that $|X| < p$. Let $T[1..q]$ be the longest prefix of $T$ such that $|X|$ is its period. Then, we have $T[1..q - |X|] = T'[|X| + 1..q]$ and $T[1..q - |X| + 1] \neq T'[|X| + 1..q + 1]$. Since $T$ is pre-Galois, we have $T[1..q - |X| + 1] \prec_T T'[|X| + 1..q + 1]$. If $|X|$ is odd, $X \succ_T T[1..q + 1]$, otherwise if $|X|$ is even, $X \prec_T T[1..q + 1]$, which does not satisfy the condition of Equation (1) in Lemma 27.

By using the property shown in Lemma 28, we can compute $\text{SPref}(W)$ by computing $\text{SPref}(W)$ recursively. For example, given $W = UV$ with $U = \text{SPref}(W)$, after computing $\text{SPref}(W)$ to get $U$, we recurse on the remaining suffix $V$ and compute $\text{SPref}(V)$ to get the next factor. We can modify Algorithm 1 to output $\text{SPref}(W)$ in $O(\ell)$ time, where $\ell$ is the length of the longest pre-Galois prefix of $W$. However, it takes time if we compute $\text{GF}(W)$ by the recursive procedure, especially when $\ell$ is much larger than $|\text{SPref}(W)|$. To tackle this problem, we use the following property.

**Lemma 29.** Let $T$ be a pre-Galois word and $p_e = \text{Per}_e(T)$. Consider a symbol $z$ and the extension $T' = T \cdot z$, such that $T'[1..|T| - p_e + 1] \succ_T T'[p_e + 1..|T| + 1]$. Let $\text{SPref}(T') = T'[1..p_e]$. If $p = p_e$ and $|T| \geq 2p$, we have $\text{SPref}(T'[p + 1..|T|]) = T'[p + 1..2p] = T'[1..p_e]$.

**Proof.** Since $T'[1..|T| - p_e]$ and $T'[p_e + 1..|T| - p_e]$ have the same length-parity, we have $T'[p_e + 1..|T| - p_e + 1] \succ_T T'[2p_e + 1..|T| + 1]$. Assume that $T[p_e + 1..]$ has a period $p' < p_e$ and $T'[p_e + 1..|T| - p' + 1] \succ_T T'[p_e + p' + 1..|T| + 1]$. Since $T$ is pre-Galois, $T'[1..p_e] = T'[p_e + 1..2p_e]$ has no even border. Thus $p'$ is odd. Let $S_1 = T'[p_e + 1..|T| - p_e]$ and $S_2 = T'[p_e + 1..|T| - p']$, $x = T'[|T| - p_e + 1]$, and $y = T'[|T| - p' + 1]$. By $T'[p_e + 1..|T| - p' + 1] \succ_T T'[p_e + p' + 1..|T| + 1]$, we have $S_2 \cdot y \succ_S S_2 \cdot z$. See the right of Figure 3 for a sketch. Since $S_1$ and $S_2$ have different parities, we have $S_1 \cdot y \prec_S S_1 \cdot z$. Moreover, by $T'[p_e + 1..|T| - p_e + 1] \succ_T T'[2p_e + 1..|T| + 1]$, we have $S_1 \cdot x \succ_S S_1 \cdot z$, which implies $S_1 \cdot x \succ_S S_1 \cdot y$. However, since $T[p_e + 1..]$ is pre-Galois, $S_1 \cdot x \preceq_S S_1 \cdot y$, which contradicts $S_1 \cdot x \succ_S S_1 \cdot y$. Therefore, $T[p_e + 1..]$ has no period $p'$ with $p' < p_e$.

Let $U = \text{SPref}(W)$, $|U|$ is even, and $W = U^k V$ for some $k \geq 2$. By Lemma 29, we know that $\text{SPref}(U^j V) = U$ for $1 \leq j < k$ without computing it explicitly. Next, we consider the case when $|\text{SPref}(W)|$ is odd.
Algorithm 2 Computing the Galois factorization, as claimed in Theorem 32.

1. Function GaloisFactorization(T)
2. Append $\$'$ to $T$;
3. Fact = ( ) empty list; $i = 0$;
4. while $i \leq |T|$ do
5. \hspace{1em} $p_o = 1$; $p_e = 2$;
6. \hspace{1em} for $j$ from 2 to $|T| - i$ do
7. \hspace{2em} \hspace{1em} $p = |T| + 2$; $p'_o = p_o$; $p'_o = p_o$;
8. \hspace{2em} \hspace{1em} if $j$ is odd then
9. \hspace{3em} \hspace{1em} if $T[i + j] < T[i + j - p_o]$ then $p = \min\{p, p_o\}$; // Lemma 28
10. \hspace{3em} \hspace{1em} else if $T[i + j] > T[i + j - p_o]$ then $p'_o = j + 1$; // Lemma 24
11. \hspace{2em} \hspace{1em} if $p_o < j$ then
12. \hspace{3em} \hspace{1em} if $T[i + j] < T[i + j - p_o]$ then $p'_o = j$; // Lemma 22
13. \hspace{3em} \hspace{1em} else if $T[i + j] > T[i + j - p_o]$ then $p = \min\{p, p_o\}$; // Lemma 28
14. \hspace{2em} \hspace{1em} else
15. \hspace{3em} \hspace{1em} if $T[i + j] < T[i + j - p_o]$ then $p'_e = j$; // Lemma 24
16. \hspace{3em} \hspace{1em} else if $T[i + j] > T[i + j - p_o]$ then $p'_e = j + 1$; // Lemma 22
17. \hspace{2em} \hspace{1em} if $p \neq |T| + 2$ then
18. \hspace{3em} \hspace{1em} while $j > p$ do
19. \hspace{4em} \hspace{1em} if $p = p_e$ and $p_e = 2p_o$ then // Lemma 29
20. \hspace{5em} \hspace{1em} Append $T[i..i + p - 1]$ and $T[i + p..i + 2p_o - 1]$ to Fact;
21. \hspace{5em} else
22. \hspace{6em} \hspace{1em} Append $T[i..i + p - 1]$ to Fact;
23. \hspace{6em} \hspace{1em} $i = i + p$; $j = j - p$;
24. \hspace{6em} $p = p_e$;
25. \hspace{4em} break;
26. \hspace{3em} \hspace{1em} $p_e = p'_e$; $p_o = p'_o$;
27. return Fact;

Lemma 30. Let $T$ be a pre-Galois word, $p_o = \text{Per}_o(T)$, and $p_e = \text{Per}_e(T)$. Consider a symbol $z$ and the extension $T' = T + z$, such that $T'[1..|T| - p_o + 1] \nless T'[|p_o + 1..|T| + 1]$. Let $P = T'[1..p] = SPref(T')$. If $p = p_o$ and $|T| \geq 3p$, we have $SPref(T'[p + 1..]) = T'[p + 1..3p] = T'[1..2p]$.

Proof. Since $T$ is pre-Galois, $T'[1..p_o] = T'[p_o + 1..2p_o]$ does not have an even border. Thus $\text{Per}_o(T[p_o + 1..]) = p_o$ and $\text{Per}_e(T[p_o + 1..]) = 2p_o$. Moreover, since $T'[1..|T| - p_o]$ and $T'[p_o + 1..|T| - p_o]$ have different parities, we have $T'[p_o + 1..|T| - p_o + 1] \nless T'[2p_o + 1..|T| + 1]$. Next, $T'[p_o + 1..|T| - 2p_o + 1] \nless T'[3p_o + 1..|T| + 1]$ holds, since $T'[1..|T| - p_o]$ and $T'[p_o + 1..|T| - 2p_o]$ have the same parity. Therefore, $SPref(T'[p_o + 1..]) = T'[p_o + 1..3p] = T'[1..2p]$.
Let $U = \text{SPref}(W)$, $|U|$ is odd, and $W = U^k V$ for some $k \geq 3$. By Lemmas 29 and 30, we know that $\text{SPref}(U^{k-2}V) = U^2$ for $0 \leq j < \lfloor k/2 \rfloor$ without computing it explicitly.

Lemmas 28, 29, and 30 are used to factorize a pre-Galois word when we extended it. However, we cannot use the Lemmas to factorize $T$ when $T$ is pre-Galois but not Galois and the input is terminated. Although we can factorize $T$ by finding $P = \text{SPref}(T)$ in Lemma 27, we need an additional procedure to find such $P$. To simplify our algorithm, we append a terminal symbol $\$$ that is smaller than all symbols of $\Sigma$. In particular, all other symbols in $W$ are different from $\$$.

Here we show that the appended $\$$ determines a Galois factor of length one, thus it does not affect the factorization result.

**Definition 33.** Galois rotation of a word.

$\text{GF}$ is a modification of Algorithm 2. We start with formally defining the Galois rotation of a word. A rotation $T = VU$ is a Galois rotation of $W = UV$ if $T$ is Galois.

**Lemma 31.** Consider a symbol $\$$ that does not appear in a word $T$ and $\$$ $\prec c$ for any $c \in \Sigma$. Then, $\text{GF}(T) = (G_1, G_2, \ldots, G_k)$ iff $\text{GF}(T \cdot \$$) = (G_1, G_2, \ldots, G_k, \$$).

**Proof.** Let $\text{GF}(T) = (G_1, G_2, \ldots, G_k)$. Here, $G_1 \succeq_{\text{alt}} G_2 \succeq_{\text{alt}} \ldots \succeq_{\text{alt}} G_k$. Since $\$$ $\prec c$ for any $c \in \Sigma$, we have $G_1 \succeq_{\text{alt}} G_2 \succeq_{\text{alt}} \ldots \succeq_{\text{alt}} G_k \succeq_{\text{alt}} \$$. Therefore, $\text{GF}(T \cdot \$$) = (G_1, G_2, \ldots, G_k, \$$)$. Similarly, let $\text{GF}(T \cdot \$$) = (G_1, G_2, \ldots, G_k, \$$)$. Here, $G_1 \succeq_{\text{alt}} G_2 \succeq_{\text{alt}} \ldots \succeq_{\text{alt}} G_k \succeq_{\text{alt}} \$$. Therefore, $\text{GF}(T) = (G_1, G_2, \ldots, G_k)$. ▶

This gives the final ingredient for introducing the algorithmic steps for computing the Galois factorization, which we present as pseudo code in Algorithm 2.

**Theorem 32.** The Galois factorization of a word $T$ can be computed in $O(|T|)$ time and $O(1)$ additional working space, excluding output space.

**Proof.** The correctness of Algorithm 2 is proven by Lemmas 28, 29, 30, and 31. Next, we prove the time complexity of Algorithm 2. The time complexity of the algorithm is bounded by the number of iterations of the inner loop (Line 6). The algorithm increments $j$ until it finds a prefix to be factorized (Line 22). Here we show that $j \leq 3\ell + 1$, where $\ell$ is the length of the factorized prefix. Let $p = p_e$. If $j < 2p + 1$, it is clear that $j \leq 3\ell + 1$, where $\ell = p$. Otherwise, if $j \geq 2p + 1$, the algorithm factorizes the prefix recursively $k$ times, such that $kp \geq j$ and $(k + 1)p > j$. Thus, we have $\ell = kp$ which implies $j \leq 3\ell + 1$. On the other hand, let $p = p_e$. If $j < 3p + 1$, it is clear that $j \leq 3\ell + 1$, where $\ell = p$. Otherwise, if $j \geq 3p + 1$, the algorithm factorizes the prefix recursively $k$ times, such that $kp \geq j$ and $(k + 2)p > j$. Thus, we have $\ell = kp$ which implies $j \leq 3\ell + 1$. Therefore, the number of iterations of the inner loop is $O(|T|)$, since the total length of the factors is $|T|$. ▶

## 7 Computing Galois rotation

While we can infer the Lyndon rotation of a word $T$ from the Lyndon factorization of $T \cdot T$, the same kind of inference surprisingly does not work for Galois words [5, Example 41]. Here, we present an algorithm computing the Galois rotation, using constant additional working space. The algorithm is a modification of Algorithm 2. We start with formally defining the Galois rotation of a word.

**Definition 33.** Let $W$ be a primitive word. A rotation $T = VU$ is a Galois rotation of $W = UV$ if $T$ is Galois.
Lemma 35.

Algorithm 3  Computing the Galois rotation of $T$, as claimed in Theorem 36.

```plaintext
1 Function GaloisRotation($T$)
2     $i = 0$;
3     while $i \leq 3|T|$ do
4         $p_o = 1$; $p_e = 2$;
5         for $j$ from 2 to $3|T| - i$ do
6             $p = 3|T| + 2$; $p'_o = p_o$; $p'_e = p_o$;
7                 if $j$ is odd then
8                     if $p_e < j$ then
9                         if $T[i + j] < T[i + j - p_e]$ then $p = \min\{p, p_e\}$; // Lemma 28
10                            else if $T[i + j] > T[i + j - p_e]$ then $p'_e = j + 1$; // Lemma 24
11                            if $p_o < j$ then
12                                if $T[i + j] < T[i + j - p_o]$ then $p'_o = j$; // Lemma 22
13                                    else if $T[i + j] > T[i + j - p_o]$ then $p = \min\{p, p_o\}$; // Lemma 28
14                     else if $p_e < j$ then
15                         if $T[i + j] < T[i + j - p_e]$ then $p'_e = j$; // Lemma 24
16                             else if $T[i + j] > T[i + j - p_e]$ then $p = \min\{p, p_e\}$; // Lemma 28
17                             if $p_o < j$ then
18                                 if $T[i + j] < T[i + j - p_o]$ then $p = \min\{p, p_o\}$; // Lemma 28
19                                     else if $T[i + j] > T[i + j - p_o]$ then $p'_o = j + 1$; // Lemma 22
20                         if $p = 3|T| + 2$ then
21                             while $j > p$ do
22                                 $i = i + p$; $j = j - p$;
23                             break;
24                             $p_e = p'_e$; $p_o = p'_o$;
25                     if $p_o \geq |T|$ and $p_e \geq |T|$ then
26                         return $(i \mod |T|) + 1$;
```

To describe our algorithm computing Galois rotations, we study a property of the Galois factorization for repetitions of a Galois word.

Lemma 34. Let $T$ be a Galois word and $P = SPref(T^k)$ for some rational number $k \geq 2$. Then $|P| \geq |T|$.

Proof. Suppose that $|P| < |T|$ and $|P|$ is even. Since $T$ is primitive, $|P|$ is not a period of $T^2$ by Lemma 5. Thus, there exist a position $i < 2|T|$ such that $P^o[i] \neq T^2[i]$. Moreover, we have $P \succeq alt T^k$ by Lemma 27, which implies $P \succeq alt T^2$. However, we have $P \prec alt T^2 = alt T^2$ by Lemma 7, which is a contradiction. The case that $|P|$ is odd leads to a similar contradiction, and thus $|P| \geq |T|$ must hold.

We then use the above property to show the following lemma, which is the core of our algorithm.

Lemma 35. Let $W$ be the Galois rotation of a primitive word $T$. Given $TTT = UWWV$ with $|U| < |T|$, let $GF(U) = (G_1, G_2, \ldots, G_k)$ and $GF(WWV) = (H_1, H_2, \ldots, H_l)$. Then we have $GF(UWWV) = (G_1, G_2, \ldots, G_k, H_1, H_2, \ldots, H_l)$.
Proof. For \( U = \varepsilon \), the claim trivially holds with \( V = W \) and \( \text{GF}(UWVV) = \text{GF}(WWV) = (H_1, H_2, \ldots, H_l) \). In the rest of the proof we assume \( U \neq \varepsilon \). Let \( \text{GF}(U) = (G_1, G_2, \ldots, G_k) \) and \( \text{GF}(WWV) = (H_1, H_2, \ldots, H_l) \). Because \( W = V \cdot U, U \) (and in particular \( G_k \)) is a suffix of \( W \). By the definition of the Galois factorization, we have \( G_1 \succ W \altdot G_2 \succeq \altdot \cdots \succeq G_k \) and \( H_1 \succeq H_2 \altdot \cdots \succeq H_l \). By showing \( G_k \succ H_1 \), we obtain that the factorization \((G_1, G_2, \ldots, G_k, H_1, H_2, \ldots, H_l)\) of \( UWVV \) admits the properties of the Galois factorization, which proves the claim.

To that end, we first observe that \( G_k \) is a proper suffix of \( W \) and \( W \) is Galois, thus \( G_k \succeq W \). Next, if \( G_k \) is not a prefix of \( W \), there exists a position \( i \leq |G_k| < |W| \) such that \( G_k^v[i] \neq W[i] \). Otherwise if \( G_k \) is a prefix of \( W \), \( G_k \) is a border of \( W \) and \( |W| - |G_k| \) is a period of \( W \). Assuming \( |G_k| \) is a period of \( W \), the greatest common divisor \( \gcd \) of \( |G_k| \) and \( |W| - |G_k| \) is a period of \( G_k \) by the periodicity lemma (Lemma 5), and \( \gcd \) is a factor of \( |W| \). However this is impossible since \( W \) is primitive; thus \( |G_k| \) cannot be a period of \( W \). Hence, there exists a position \( i \leq |W| \) such that \( G_k^v[i] \neq W[i] \). We therefore know that the first \( k \) Galois factors in \( \text{GF}(U) \) and \( \text{GF}(UWVV) \) are the same since we cannot extend \( G_k \) further without losing the property to be Galois. Moreover, \( W \) is a prefix of \( H_1 \) by Lemma 34. Thus, there exists a position \( j \leq |W| \leq |H_1| \) such that \( G_k^v[j] \neq H_1[j] \), which implies \( G_k \succ H_1 \). Therefore, we have \( G_1 \altdot G_2 \altdot \cdots \altdot G_k \altdot H_1 \altdot H_2 \altdot \cdots \altdot H_l \), which implies \( \text{GF}(UWVV) = (G_1, G_2, \ldots, G_k, H_1, H_2, \ldots, H_l) \).

With Lemma 35, we now have a tool to find the Galois rotation \( W \) of \( T \) in \( TTT \) by determining \( H_1 \) and knowing that \( W \) is a prefix of \( H_1 \). Since we can write \( TTT = UWVV \) with \( W = VU \) and \( |U| < |T| \), the goal is to determine \( U \). For \( U \), we know that all its Galois factors have even or odd periods shorter than \( |T| \), so it suffices to find the first Galois factor in \( TTT \) for which both periods are at least \( |T| \) long (cf. Lemma 34).

In detail, let \( G \) be the first factor of \( \text{GF}(TTT) \) with \( |G| \geq |T| \). From Lemma 35 we know that the Galois rotation \( W \) of a word \( T \) is the prefix of \( G \) whose length is \( |T| \), i.e. \( W = G[1..|T|] \). Algorithm 3 describes an algorithm to compute the Galois rotation of an input word \( T \). The algorithm scans \( TTT \) sequentially from the beginning, mimicking our Galois factorization algorithm, except that it does not output the factors. At the time where we set \( p_0 \geq |T| \) and \( p_e \geq |T| \), we know that the length of next factor we compute is \( |T| \) or longer. At that time, we can determine \( G \). To this end, we output the starting position \( i \) of \( G \) when reaching the condition that \( p_0 \geq |T| \) and \( p_e \geq |T| \). In a post-processing, we determine \( W = (TTT)[i..i+|T|-1] \), where \( W \) is the Galois rotation of \( T \). To keep the additional working space constant, we do not load three copies of \( T \) into memory, but use that fact that \( (TTT)[k] = T'[((k-1) \mod |T|) + 1] \) for \( k > |T| \) when processing the input \( TTT \).

\medskip

▸ Theorem 36. The Galois rotation of a word \( T \) can be computed in \( O(|T|) \) time and \( O(1) \) additional working space.

8 Experiments

We have implemented our algorithms in C++. The software is freely available by the link on the title page. For a short demonstration, we computed the Galois factors of files from the Canterbury, the Calgary [2] and the Pizza&CChili corpus [8], and depict the results in Table 1. We have omitted those files that contain a zero byte, which is prohibited in our implementation. The experiments run on WSL with Intel Core i7-10700K CPU. To compare the time with a standard Lyndon factorization algorithm, we used the implementation of Duval’s algorithm from https://github.com/cp-algorithms/cp-algorithms.
Table 1 Counting the number of Galois factors for various datasets. The alphabet size is denoted by \( \sigma \). Counts are listed in the \# columns, together with a time evaluation with Duval’s algorithm computing the Lyndon factorization. Upper part: The Canterbury and Calgary corpus datasets. Lower part: The Pizza&Chili corpus datasets. Note that we used different time units for the upper table (microseconds) and lower table (seconds).

<table>
<thead>
<tr>
<th>File</th>
<th>( \sigma )</th>
<th>Size [KB]</th>
<th>Galois #</th>
<th>Lyndon #</th>
<th>Time [( \mu )s]</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>alice29.txt</td>
<td>74</td>
<td>152</td>
<td>14</td>
<td>3070</td>
<td>3</td>
<td>192</td>
</tr>
<tr>
<td>asyoulik.txt</td>
<td>68</td>
<td>125</td>
<td>7</td>
<td>2435</td>
<td>2</td>
<td>134</td>
</tr>
<tr>
<td>Bib</td>
<td>81</td>
<td>111</td>
<td>25</td>
<td>2372</td>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>Book2</td>
<td>96</td>
<td>610</td>
<td>20</td>
<td>12182</td>
<td>27</td>
<td>555</td>
</tr>
<tr>
<td>cp.html</td>
<td>86</td>
<td>24</td>
<td>7</td>
<td>544</td>
<td>8</td>
<td>21</td>
</tr>
<tr>
<td>fields.c</td>
<td>90</td>
<td>11</td>
<td>18</td>
<td>237</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>Grammar.lsp</td>
<td>76</td>
<td>3</td>
<td>10</td>
<td>78</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>lcet10.txt</td>
<td>84</td>
<td>426</td>
<td>12</td>
<td>8599</td>
<td>6</td>
<td>438</td>
</tr>
<tr>
<td>News</td>
<td>98</td>
<td>377</td>
<td>24</td>
<td>7440</td>
<td>24</td>
<td>375</td>
</tr>
<tr>
<td>paper1</td>
<td>95</td>
<td>53</td>
<td>19</td>
<td>1016</td>
<td>9</td>
<td>40</td>
</tr>
<tr>
<td>paper2</td>
<td>91</td>
<td>82</td>
<td>14</td>
<td>1593</td>
<td>16</td>
<td>66</td>
</tr>
<tr>
<td>paper3</td>
<td>84</td>
<td>46</td>
<td>11</td>
<td>908</td>
<td>14</td>
<td>39</td>
</tr>
<tr>
<td>paper4</td>
<td>80</td>
<td>13</td>
<td>8</td>
<td>267</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>paper5</td>
<td>91</td>
<td>11</td>
<td>9</td>
<td>237</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>paper6</td>
<td>93</td>
<td>38</td>
<td>12</td>
<td>740</td>
<td>15</td>
<td>32</td>
</tr>
<tr>
<td>Plrabn12.txt</td>
<td>81</td>
<td>481</td>
<td>4</td>
<td>9801</td>
<td>6</td>
<td>559</td>
</tr>
<tr>
<td>progc</td>
<td>92</td>
<td>39</td>
<td>15</td>
<td>788</td>
<td>12</td>
<td>36</td>
</tr>
<tr>
<td>progl</td>
<td>87</td>
<td>71</td>
<td>84</td>
<td>1423</td>
<td>77</td>
<td>63</td>
</tr>
<tr>
<td>progp</td>
<td>89</td>
<td>49</td>
<td>14</td>
<td>944</td>
<td>12</td>
<td>38</td>
</tr>
<tr>
<td>xargs.1</td>
<td>74</td>
<td>4</td>
<td>6</td>
<td>88</td>
<td>9</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File</th>
<th>( \sigma )</th>
<th>Size [KB]</th>
<th>Galois #</th>
<th>Lyndon #</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>dblp.xml</td>
<td>97</td>
<td>296135</td>
<td>3</td>
<td>5.969</td>
<td>15</td>
</tr>
<tr>
<td>DNA</td>
<td>97</td>
<td>403927</td>
<td>26</td>
<td>7.799</td>
<td>18</td>
</tr>
<tr>
<td>PROTEINS</td>
<td>27</td>
<td>1184051</td>
<td>29</td>
<td>24384</td>
<td>30</td>
</tr>
<tr>
<td>SOURCES</td>
<td>230</td>
<td>210866</td>
<td>23</td>
<td>4.307</td>
<td>35</td>
</tr>
</tbody>
</table>

References


18:16 Algorithms for Galois Words: Detection, Factorization, and Rotation


