Advances in Applied Mathematics 101 (2018) 281–319

Contents lists available at ScienceDirect

Advances in Applied Mathematics

www.elsevier.com/locate/yaama

Inverse Lyndon words and inverse Lyndon factorizations of words

Paola Bonizzoni [a], Clelia De Felice [b,*], Rocco Zaccagnino [b], Rosalba Zizza [b]

[a] *Dipartimento di Informatica Sistemistica e Comunicazione, Università degli Studi di Milano Bicocca, Viale Sarca 336, 20126 Milano, Italy*
[b] *Dipartimento di Informatica, Università degli Studi di Salerno, via Giovanni Paolo II 132, 84084 Fisciano (SA), Italy*

ARTICLE INFO

Article history:
Received 31 July 2018
Received in revised form 2 August 2018
Accepted 12 August 2018
Available online 30 August 2018

MSC:
68R15
68W32

Keywords:
Lyndon words
Lyndon factorization
Combinatorial algorithms on words
DNA sequences

ABSTRACT

Motivated by applications to string processing, we introduce variants of the Lyndon factorization called inverse Lyndon factorizations. Their factors, named inverse Lyndon words, are in a class that strictly contains anti-Lyndon words, that is Lyndon words with respect to the inverse lexicographic order. The Lyndon factorization of a nonempty word w is unique but w may have several inverse Lyndon factorizations. We prove that any nonempty word w admits a canonical inverse Lyndon factorization, named ICFL(w), that maintains the main properties of the Lyndon factorization of w: it can be computed in linear time, it is uniquely determined, and it preserves a compatibility property for sorting suffixes. In particular, the compatibility property of ICFL(w) is a consequence of another result: any factor in ICFL(w) is a concatenation of consecutive factors of the Lyndon factorization of w with respect to the inverse lexicographic order.

* Corresponding author.
E-mail addresses: bonizzoni@disco.unimib.it (P. Bonizzoni), cdefelice@unisa.it (C. De Felice), zaccagnino@dia.unisa.it (R. Zaccagnino), rzizza@unisa.it (R. Zizza).

https://doi.org/10.1016/j.aam.2018.08.005
0196-8858/

1. Introduction

Lyndon words were introduced in [34], as *standard lexicographic sequences*, and then used in the context of the free groups in [8]. A Lyndon word is a word which is strictly smaller than each of its proper cyclic shifts for the lexicographical ordering. A famous theorem concerning Lyndon words asserts that any nonempty word factorizes uniquely into a nonincreasing product of Lyndon words, called its Lyndon factorization. This theorem, that can be recovered from results in [8], provides an example of a factorization of a free monoid, as defined in [42] (see also [4,32]). Moreover, there are several results which give relations between Lyndon words, codes and combinatorics of words [3]. More recently these words found a renewed theoretical interest and several variants of them have been studied [7,16]. A related field studies the combinatorial and algorithmic properties of *necklaces*, that are powers of Lyndon words, and their prefixes or *prenecklaces* [6].

The Lyndon factorization has recently revealed to be a useful tool also in string processing algorithms [2,38] with a potential that has not been completely explored and understood. This is due also to the fact that it can be efficiently computed. Linear-time algorithms for computing this factorization can be found in [17,18] whereas an $\mathcal{O}(\lg n)$-time parallel algorithm has been proposed in [1,14]. A connection between the Lyndon factorization and the Lempel–Ziv (LZ) factorization has been given in [26], where it is shown that in general the size of the LZ factorization is larger than the size of the Lyndon factorization, and in any case the size of the Lyndon factorization cannot be larger than a factor of 2 with respect to the size of LZ.

A Lyndon word is lexicographically smaller than all its proper nonempty suffixes. This explains why the Lyndon factorization has become of particular interest also in suffix sorting problems. The suffix array (SA) of a word w is the lexicographically ordered list of the starting positions of the suffixes of w. The connection between Lyndon factorizations and suffix arrays has been pointed out in [25], where the authors show a method to construct the Lyndon factorization of a text from its SA. Conversely, the computation of the SA of a text from its Lyndon factorization has been proposed in [5] and then explored in [36,37].

The algorithm proposed in [36,37] is based on the following interesting combinatorial result, proved in the same papers: if u is a concatenation of consecutive Lyndon factors of $w = xuy$, then the position of a nonempty suffix u_i in the ordered list of suffixes of u (called *local* suffixes) is the same as the position of the nonempty suffix $u_i y$ in the ordered list of the suffixes of w (called *global* suffixes). In turn, this result suggests a divide and conquer strategy for the sorting of the suffixes of a word $w = w_1 w_2$: we order the nonempty suffixes of w_1 and the nonempty suffixes of w_2 independently (or in parallel) and then we merge the resulting lists (see Section 2.4 for further details).

Relations between Lyndon words and the Burrows–Wheeler Transform (BWT) have been discovered first in [11,35] and, more recently, in [29] and in [23]. Variants of the BWT

proposed in the previous papers are based on combinatorial results proved in [20] (see [39] for further details and [19] for more recent related results). In particular, variants of the BWT that are based on factoring the input word into Lyndon words lead to techniques for saving space [15]. These novel transforms may be useful to compress collections of words as those generated by sequencing technologies, called *reads*. Indeed, the BWT has produced extremely important tools to deal with these data [30]. Reads are words that have a highly repetitive structure, since they are shifted subwords of a common genomic sequence. An application to collections of reads of a factorization is of interest mainly if it allows to capture common factors of the input reads. In turn, applications of the Lyndon factorization are of practical interest when we can take advantage of the decomposition of the word, in particular when the word is factorized into a certain number of factors with a reasonable length. For example a word a^k is factorized into factors of size 1, while we may have Lyndon words of huge size.

In this paper we face the following main theoretical question, raised by the above discussion: *can we define variants of the Lyndon factorization of a word, which maintain some useful properties of the Lyndon factorization, but having factors that generalize Lyndon words?*

Following this direction, we introduce the notion of an *inverse Lyndon word*, that is a word greater than any of its proper nonempty suffixes (Section 3). We compare this notion with that of Lyndon words with respect to the inverse lexicographic order (or anti-Lyndon words [21]) and with the notion of strict sesquipowers of Lyndon words [17]. We show that the set of the inverse Lyndon words is equal to the set of the sesquipowers of anti-Lyndon words and also to the set of the anti-prenecklaces (prenecklaces with respect to the inverse lexicographic order). Consequently, the set of the inverse Lyndon words strictly contains the class of the anti-Lyndon words. We also compare this notion with that of Nyldon words [7,24] and generalized Lyndon words [16,41].

A consequent question is whether it is possible to define a unique factorization of a word w into inverse Lyndon words that can be computed in linear time as the Lyndon factorization. Then we give the definition of an inverse Lyndon factorization of a word, whose factors are inverse Lyndon words (Section 4). The Lyndon factorization of a nonempty word w is unique but w may have several inverse Lyndon factorizations. As a main result, we define a canonical inverse Lyndon factorization of a nonempty word w, denoted by ICFL(w). We prove that ICFL(w) can be still computed in linear time and it is uniquely determined. Moreover, if w is a Lyndon word different from a letter, then ICFL(w) has at least two factors and a converse holds for an inverse Lyndon word and its Lyndon factorization. Finally, we prove that ICFL(w) belongs to a special class of inverse Lyndon factorizations, called *groupings* (see Section 7). In a grouping each of its factors is a concatenation of consecutive factors of the Lyndon factorization of w with respect to the inverse lexicographic order. Hence the compatibility property proved in [36] applies also to ICFL(w), with respect to the inverse lexicographic order. Moreover this result allows us to partially compare ICFL with the classical Lyndon factorization with respect to the inverse lexicographic order.

In order to answer the above question, we propose to combine the Lyndon factorization of a word w with ICFL(w). We test our proposal by running an experimental analysis over two biological datasets.[1]

Experiments confirm that we obtain a factorization of intermediate size between that of LZ and that of the Lyndon factorization.

The paper is organized as follows. In Section 2, we gathered the basic definitions and known results we need. Inverse Lyndon words are discussed in Section 3. Inverse Lyndon factorizations and ICFL(w) are presented in Section 4. More precisely, for the construction of ICFL(w) we need a special prefix of w, defined in Section 4.2, whereas we give the recursive definition of the factorization in Section 4.3. A linear-time algorithm for computing ICFL(w) is presented in Section 6. This algorithm uses two subroutines described in Section 5. We introduce groupings in Section 7 and we prove that ICFL(w) falls in this class of factorizations in Section 7.2.

2. Preliminaries

For the material in this section see [4,9,32,33,40].

2.1. Words

Let Σ^* be the *free monoid* generated by a finite alphabet Σ and let $\Sigma^+ = \Sigma^* \setminus 1$, where 1 is the empty word. For a set X, Card(X) denotes the cardinality of X. For a word $w \in \Sigma^*$, we denote by $|w|$ its *length*. A word $x \in \Sigma^*$ is a *factor* of $w \in \Sigma^*$ if there are $u_1, u_2 \in \Sigma^*$ such that $w = u_1 x u_2$. If $u_1 = 1$ (resp. $u_2 = 1$), then x is a *prefix* (resp. *suffix*) of w. A factor (resp. prefix, suffix) x of w is *proper* if $x \neq w$. Two words x, y are *incomparable* for the prefix order, denoted as $x \bowtie y$, if neither x is a prefix of y nor y is a prefix of x. Otherwise, x, y are *comparable* for the prefix order. We write $x \leq_p y$ if x is a prefix of y and $x \geq_p y$ if y is a prefix of x.

We recall that two words x, y are called *conjugate* if there exist words u, v such that $x = uv, y = vu$. The conjugacy relation is an equivalence relation. A conjugacy class is a class of this equivalence relation. The following is Proposition 1.3.4 in [31].

Proposition 2.1. *Two words $x, y \in \Sigma^+$ are conjugate if and only if there exists $r \in \Sigma^*$ such that*

$$xr = ry \tag{2.1}$$

More precisely, equality (2.1) holds if and only if there exist $u, v \in \Sigma^*$ such that

$$x = uv, \quad y = vu, \quad r \in u(vu)^*. \tag{2.2}$$

[1] http://www.di.unisa.it/professori/zizza/EXP/experiments.pdf.

A *sesquipower* of a word x is a word $w = x^n p$ where p is a proper prefix of x and $n \geq 1$. A nonempty word w is *unbordered* if no proper nonempty prefix of w is a suffix of w. Otherwise, w is *bordered*. A nonempty word w is *primitive* if $w = x^k$ implies $k = 1$. An unbordered word is primitive. Let r, w be nonempty words over Σ. We say that two occurrences of r as a factor of w *overlap* if $w = xrz = x'rz'$ with $|x'| < |x| < |x'r|$. Therefore r is bordered. The following lemma will be used in Section 5.

Lemma 2.1. *Let $x, y, w, r \in \Sigma^+$ be such that $w = xr = ry$, with $|x| < |r|$, i.e., r occurs twice in w and these two occurrences of r in w overlap. Then there exists $r' \in \Sigma^+$ such that $w = x'r' = r'y'$, with $|r'| < |x'|$, and y', y start with the same letter.*

Proof:
Let $x, y, r \in \Sigma^+$ be as in the statement. By Proposition 2.1, there are $u, v \in \Sigma^*$ and $n \in \mathbb{N}$ such that

$$x = uv, \quad y = vu, \quad r = u(vu)^n. \tag{2.3}$$

Set

$$r' = \begin{cases} u & \text{if } u \neq 1 \\ v & \text{if } u = 1 \end{cases} \tag{2.4}$$

and

$$x' = \begin{cases} x(uv)^n = (uv)^{n+1} & \text{if } u \neq 1 \\ v^n & \text{if } u = 1 \end{cases} \quad y' = \begin{cases} (vu)^n y = (vu)^{n+1} & \text{if } u \neq 1 \\ v^n & \text{if } u = 1 \end{cases} \tag{2.5}$$

By using Eqs. (2.3)–(2.5), we can easily see that $x'r' = r'y' = ry = xr$. Moreover, since $|x| < |r|$, if $u \neq 1$, then $n \geq 1$ and if $u = 1$, then $n \geq 2$. Hence, in both cases, $|r'| < |x'|$. Finally, y is a prefix of y', therefore they start with the same letter. ∎

2.2. Lexicographic order and Lyndon words

Definition 2.1. Let $(\Sigma, <)$ be a totally ordered alphabet. The *lexicographic* (or *alphabetic order*) $\prec$ on $(\Sigma^*, <)$ is defined by setting $x \prec y$ if

- x is a proper prefix of y, or
- $x = ras$, $y = rbt$, $a < b$, for $a, b \in \Sigma$ and $r, s, t \in \Sigma^*$.

For two nonempty words x, y, we write $x \ll y$ if $x \prec y$ and x is not a proper prefix of y [2]. We also write $y \succ x$ if $x \prec y$. Basic properties of the lexicographic order are recalled below.

Lemma 2.2. *For $x, y \in \Sigma^*$, the following properties hold.*

(1) *$x \prec y$ if and only if $zx \prec zy$, for every word z.*
(2) *If $x \ll y$, then $xu \ll yv$ for all words u, v.*
(3) *If $x \prec y \prec xz$ for a word z, then $y = xy'$ for some word y' such that $y' \prec z$.*

Definition 2.2. A Lyndon word $w \in \Sigma^+$ is a word which is primitive and the smallest one in its conjugacy class for the lexicographic order.

Example 2.1. Let $\Sigma = \{a, b\}$ with $a < b$. The words a, b, $aaab$, $abbb$, $aabab$ and $aababaabb$ are Lyndon words. On the contrary, $abab$, aba and $abaab$ are not Lyndon words. Indeed, $abab$ is a non-primitive word, $aab \prec aba$ and $aabab \prec abaab$.

Lyndon words are also called *prime words* and their prefixes are also called *preprime words* in [27]. Some properties of Lyndon words are recalled below.

Proposition 2.2. *Each Lyndon word w is unbordered.*

Proposition 2.3. *A word $w \in \Sigma^+$ is a Lyndon word if and only if $w \prec s$, for each nonempty proper suffix s of w.*

A class of conjugacy is also called a *necklace* and often identified with the minimal word for the lexicographic order in it. We will adopt this terminology. Then a word is a necklace if and only if it is a power of a Lyndon word. A *prenecklace* is a prefix of a necklace. Then clearly any nonempty prenecklace w has the form $w = (uv)^k u$, where uv is a Lyndon word, $u \in \Sigma^*$, $v \in \Sigma^+$, $k \geq 1$, that is, w is a sesquipower of a Lyndon word uv. The following result has been proved in [17].

Proposition 2.4. *A word is a nonempty preprime word if and only if it is a sesquipower of a Lyndon word distinct of the maximal letter.*

The proof of Proposition 2.4 uses the following result which characterizes, for a given nonempty prenecklace w and a letter b, whether wb is still a prenecklace or not and, in the first case, whether wb is a Lyndon word or not [17, Lemma 1.6].

Theorem 2.1. *Let $w = (uav')^k u$ be a nonempty prenecklace, where uav' is a Lyndon word, $u, v' \in \Sigma^*$, $a \in \Sigma$, $k \geq 1$. For any $b \in \Sigma$, the word wb is a prenecklace if and only if $b \geq a$. Moreover $wb \in L$ if and only if $b > a$.*

A direct consequence of Theorem 2.1 is reported below (see [6, Theorem 2.1] which states both Theorem 2.1 and Corollary 2.1).

Corollary 2.1. *Let $w = (uav')^k u$ be a nonempty prenecklace, where uav' is a Lyndon word, $u, v' \in \Sigma^*$, $a \in \Sigma$, $k \geq 1$. Let $b \in \Sigma$ with $b \geq a$ and let y be the longest Lyndon*

prefix of wb. Then

$$y = \begin{cases} uav' & \text{if } b = a \\ wb & \text{if } b > a \end{cases}$$

2.3. The Lyndon factorization

A family $(X_i)_{i\in I}$ of subsets of Σ^+, indexed by a totally ordered set I, is a *factorization of the free monoid* Σ^* if each word $w \in \Sigma^*$ has a unique factorization $w = x_1 \cdots x_n$, with $n \geq 0$, $x_i \in X_{j_i}$ and $j_1 \geq j_2 \geq \ldots \geq j_n$ [4]. A factorization $(X_i)_{i\in I}$ is called *complete* if each X_i is reduced to a singleton x_i [4]. In the following $L = L_{(\Sigma^*,<)}$ will be the set of Lyndon words, totally ordered by the relation $\prec$ on $(\Sigma^*, <)$. The following theorem shows that the family $(\ell)_{\ell\in L}$ is a complete factorization of Σ^*.

Theorem 2.2. *Any word $w \in \Sigma^+$ can be written in a unique way as a nonincreasing product $w = \ell_1\ell_2\cdots\ell_h$ of Lyndon words, i.e., in the form*

$$w = \ell_1\ell_2\cdots\ell_h, \text{ with } \ell_j \in L \text{ and } \ell_1 \succeq \ell_2 \succeq \ldots \succeq \ell_h \tag{2.6}$$

The sequence $\mathrm{CFL}(w) = (\ell_1, \ldots, \ell_h)$ in Eq. (2.6) is called the *Lyndon decomposition* (or *Lyndon factorization*) of w. It is denoted by $\mathrm{CFL}(w)$ because Theorem 2.2 is usually credited to Chen, Fox and Lyndon [8]. Uniqueness of the above factorization is a consequence of the following result, proved in [17].

Lemma 2.3. *Let $w \in \Sigma^+$ and let $\mathrm{CFL}(w) = (\ell_1, \ldots, \ell_h)$. Then the following properties hold:*

(i) *ℓ_h is the nonempty suffix of w which is the smallest with respect to the lexicographic order.*
(ii) *ℓ_h is the longest suffix of w which is a Lyndon word.*
(iii) *ℓ_1 is the longest prefix of w which is a Lyndon word.*

Therefore, given $w \in \Sigma^+$, if ℓ_1 is its longest prefix which is a Lyndon word and $w = \ell_1 w'$, then $\mathrm{CFL}(w) = (\ell_1, \mathrm{CFL}(w'))$. As a consequence of Theorem 2.2, for any word w there is a factorization

$$w = \ell_1^{n_1} \cdots \ell_r^{n_r}$$

where $r > 0$, $n_1, \ldots, n_r \geq 1$, and $\ell_1 \succ \ldots \succ \ell_r$ are Lyndon words, also named *Lyndon factors* of w. There is a linear time algorithm to compute the pair (ℓ_1, n_1) and thus, by iteration, the Lyndon factorization of w. It is due to Fredricksen and Maiorana [18] and it is also reported in [33]. It can also be used to compute the Lyndon word in the conjugacy class of a primitive word in linear time [33]. Linear time algorithms may also be found in [17] and in the more recent paper [22].

2.4. Sorting suffixes of a text

In [36,37], the authors found interesting relations between the sorting of the suffixes of a word w and that of its factors. Let $w, x, u, y \in \Sigma^*$, and let u be a nonempty factor of $w = xuy$. Let $first(u)$ and $last(u)$ denote the position of the first and the last symbol of u in w, respectively. If $w = a_1 \cdots a_n$, $a_i \in \Sigma$, $1 \leq i \leq j \leq n$, then we also set $w[i, j] = a_i \cdots a_j$. A *local suffix* of w is a suffix of a factor of w, specifically $suf_u(i) = w[i, last(u)]$ denotes the *local suffix* of w at the position i with respect to u, $i \geq first(u)$. The corresponding *global suffix* $suf_u(i)y$ of w at the position i is denoted by $suf_w(i) = w[i, last(w)]$ (or simply $suf(i)$ when it is understood). We say that $suf_u(i)y$ is *associated* with $suf_u(i)$.

Definition 2.3. [36,37] Let $w \in \Sigma^+$ and let u be a nonempty factor of w. We say that the sorting of the nonempty local suffixes of w with respect to u is compatible with the sorting of the corresponding nonempty global suffixes of w if for all i, j with $first(u) \leq i < j \leq last(u)$,

$$suf_u(i) \prec suf_u(j) \iff suf(i) \prec suf(j).$$

Let $u = \ell_r \cdots \ell_s$ be a concatenation of consecutive Lyndon factors of w. Let $\mathcal{L}_{loc}(u, w) = (s_1, \ldots, s_t)$ be the ordered list of the nonempty local suffixes of w with respect to u and let $\mathcal{L}_{glob}(u, w) = (s'_1, \ldots, s'_t)$ be the ordered list of the corresponding nonempty global suffixes of w. We name it the *global list associated* with $\mathcal{L}_{loc}(u, w)$. The following result proved in [36,37] shows that each s'_i in $\mathcal{L}_{glob}(u, w)$ is associated with s_i.

Theorem 2.3. *Let $w \in \Sigma^+$ and let $\mathrm{CFL}(w) = (\ell_1, \ldots, \ell_h)$ be its Lyndon factorization. Then, for any r, s, $1 \leq r \leq s \leq h$, the sorting of the nonempty local suffixes of w with respect to $u = \ell_r \cdots \ell_s$ is compatible with the sorting of the corresponding nonempty global suffixes of w.*

Remark 2.1. The compatibility property stated in Theorem 2.3 does not hold for the empty suffix of a word. Indeed, let $\Sigma = \{a, b, c, d\}$ with $a < b < c < d$. Let $w = bbcbcacad$, thus $\mathrm{CFL}(w) = (bbc, bc, acad)$. Consider the local suffixes $1, bbc$ of w with respect to $u = bbc$ and the corresponding global suffixes $bcacad, w$. We have $1 \prec bbc$ but $w \prec bcacad$.

If $\mathcal{L}_1$ and $\mathcal{L}_2$ are two sorted lists of elements taken from any totally ordered set, then the result of the operation $\mathrm{merge}(\mathcal{L}_1, \mathcal{L}_2)$ is a single sorted list containing the elements of $\mathcal{L}_1$ and $\mathcal{L}_2$. Theorem 2.3 could be considered in a merge sort algorithm for the sorting of the suffixes of w, as suggested in [36,37]. Starting with the list $\mathrm{CFL}(w) = (\ell_1, \ldots, \ell_h)$, it could operate as follows.

– Divide the sequence into two subsequences $(\ell_1, \ldots, \ell_r)$, $(\ell_{r+1}, \ldots, \ell_h)$, where $r = \lceil h/2 \rceil$

- Let $\mathcal{L}_1$ be the list of the nonempty suffixes of $u = \ell_1 \cdots \ell_r$, let $\mathcal{L}_2$ be the list of the nonempty suffixes of $y = \ell_{r+1} \cdots \ell_h$. Sort the two subsequences $\mathcal{L}_1, \mathcal{L}_2$ recursively using merge sort, thus obtaining $\mathcal{L}_{loc}(u, w), \mathcal{L}_{loc}(y, w)$
- Merge the two subsequences $\mathcal{L}_{glob}(u, w), \mathcal{L}_{glob}(y, w)$ to produce $\mathcal{L}_{glob}(w, w)$.

Notice that in the third step we change $\mathcal{L}_{loc}(u, w), \mathcal{L}_{loc}(y, w)$ into $\mathcal{L}_{glob}(u, w), \mathcal{L}_{glob}(y, w)$. Indeed, as pointed out in Example 2.2, two problems arise if we considered the local lists in the merge step. First, there could exist two local suffixes s, s', with corresponding global suffixes $sz, s'z'$, such that $s \prec s'$ but $s'z' \prec sz$. Furthermore, an element s could occur twice in merge$(\mathcal{L}_{loc}(u, w), \mathcal{L}_{loc}(y, w))$. In this case, when we produce the global list $\mathcal{L}_{glob}(w, w)$, we have to change the second occurrence of s into sy.

Example 2.2. Let $\Sigma = \{a, b, c, d\}$ with $a < b < c < d$. Let $w = bbcbcacad$, thus CFL$(w) = (bbc, bc, acad)$. Then, $\mathcal{L}_{loc}(bbc, w) = (bbc, bc, c)$, $\mathcal{L}_{loc}(bc, w) = (bc, c)$, and $\mathcal{L}_{loc}(acad, w) = (acad, ad, cad, d)$. Consider the local suffix c in $\mathcal{L}_{loc}(bbc, w)$, the local suffix cad in $\mathcal{L}_{loc}(acad, w)$ and the corresponding associated global suffixes $cbcacad$ and cad. We see that $c \prec cad$ but $cbcacad \succ cad$. Let $u = bbcbc$, $y = acad$, then

$$\begin{aligned}
\mathcal{L}_{loc}(bbcbc, w) &= \mathcal{L}_{glob}(bbcbc, u) \\
&= \text{merge}(\mathcal{L}_{glob}(bbc, u), \mathcal{L}_{glob}(bc, u)) \\
&= \text{merge}((bbcbc, bcbc, cbc), (bc, c)) = (bbcbc, bc, bcbc, c, cbc) \\
\mathcal{L}_{glob}(w, w) &= \text{merge}(\mathcal{L}_{glob}(u, w), \mathcal{L}_{glob}(y, w)) \\
&= \text{merge}((bbcbcacad, bcacad, bcbcacad, cacad, cbcacad), (acad, ad, cad, d)) \\
&= (acad, ad, bbcbcacad, bcacad, bcbcacad, cacad, cad, cbcacad, d)
\end{aligned}$$

If in the third step we merged $\mathcal{L}_{loc}(bbc, u), \mathcal{L}_{loc}(bc, u)$, we would obtain (bbc, bc, bc, c, c). Then, in order to obtain $\mathcal{L}_{loc}(bbcbc, w)$, we have to change bbc into $bbcbc$, the second occurrence of bc into $bcbc$ and the second occurrence of c into cbc.

2.5. Inverse lexicographic order and anti-Lyndon words

We also need the following well-known definition.

Definition 2.4. Let $(\Sigma, <)$ be a totally ordered alphabet. Let $<_{in}$ be the inverse of $<$, defined by

$$\forall a, b \in \Sigma \quad b <_{in} a \Leftrightarrow a < b$$

The inverse lexicographic or inverse alphabetic order, denoted $\prec_{in}$, on $(\Sigma^*, <)$ is the lexicographic order on $(\Sigma^*, <_{in})$.

Example 2.3. Let $\Sigma = \{a, b, c, d\}$ with $a < b < c < d$. Then $dab \prec dabd$ and $dabda \prec dac$. We have $d <_{in} c <_{in} b <_{in} a$. Therefore $dab \prec_{in} dabd$ and $dac \prec_{in} dabda$.

The following proposition justifies the adopted terminology.

Proposition 2.5. *Let $(\Sigma, <)$ be a totally ordered alphabet. For all $x, y \in \Sigma^*$ such that $x \bowtie y$,*

$$y \prec_{in} x \Leftrightarrow x \prec y.$$

Moreover, in this case $x \ll y$.

Proof:
Let $x, y \in \Sigma^*$ such that $x \bowtie y$. Assume $y \prec_{in} x$. Thus, by Definition 2.4, there are $a, b \in \Sigma$, with $b <_{in} a$, and $r, s, t \in \Sigma^*$ such that $y = rbs$, $x = rat$. Hence $a < b$ and $x \prec y$, by Definition 2.1. Conversely, if $x \prec y$ we have $y \prec_{in} x$ by a similar argument. The second part of the statement follows by Definition 2.1. ■

From now on, $L_{in} = L_{(\Sigma^*, <_{in})}$ denotes the set of the Lyndon words on Σ^* with respect to the inverse lexicographic order. A word $w \in L_{in}$ will be named an *anti-Lyndon word*. Correspondingly, an *anti-prenecklace* will be a prefix of an *anti-necklace*, which in turn will be a necklace with respect to the inverse lexicographic order. The following proposition characterizes $L_{in} = L_{(\Sigma^*, <_{in})}$.

Proposition 2.6. *A word $w \in \Sigma^+$ is in L_{in} if and only if w is primitive and $w \succ vu$, for each $u, v \in \Sigma^+$ such that $w = uv$.*

Proof:
By Definition 2.2, if $w \in L_{in}$, then w is nonempty and primitive. Moreover, if $w = uv$, with $u, v \neq 1$, then $w \prec_{in} vu$. Since $uv \bowtie vu$, by Proposition 2.5 one has $vu \prec w$, i.e., $w \succ vu$. A similar argument shows that if w is a primitive nonempty word and $w \succ vu$, for each $u, v \in \Sigma^+$ such that $w = uv$, then $w \in L_{in}$. ■

We state below a slightly modified dual version of Proposition 2.3.

Proposition 2.7. *A word $w \in \Sigma^+$ is in L_{in} if and only if w is unbordered and $w \succ v$, for each proper nonempty suffix v.*

Proof:
Let $w \in L_{in} = L_{(\Sigma^*, <_{in})}$. Therefore, w is nonempty and unbordered (Definition 2.2, Proposition 2.2). Moreover, if $w = uv$, with $u, v \neq 1$, then $w \prec_{in} v$, by Proposition 2.3. In addition, $w \bowtie v$ since w is unbordered and $|v| < |w|$. Consequently, $w \succ v$ (Proposition 2.5).

Conversely, let $w = uv$ be a nonempty unbordered word such that $w \succ v$, for each proper nonempty suffix v. Thus, $v \bowtie w$, hence $w \prec_{in} v$ (Proposition 2.5). By Proposition 2.3, the word w is in L_{in}. ■

The following result gives more precise relations between words in L_{in} and their proper nonempty suffixes.

Proposition 2.8. *If v is a proper nonempty suffix of $w \in L_{in}$, then $v \ll w$.*

Proof:
Let v be a proper nonempty suffix of $w \in L_{in}$. By Proposition 2.7, we have $v \prec w$. Moreover, since w is unbordered, we have $v \bowtie w$, hence $v \ll w$. ■

3. Inverse Lyndon words

As mentioned in Section 1, the Lyndon factorization of a word may generate very long or too short factors, thus becoming unsatisfactory with respect to a parallel strategy. We face this problem in Section 4, where we introduce another factorization which maintains the main properties of the Lyndon factorization but that allows us to overcome the glitch. This factorization is based on the notion of inverse Lyndon words, given below.

Definition 3.1. A word $w \in \Sigma^+$ is an inverse Lyndon word if $s \prec w$, for each nonempty proper suffix s of w.

Example 3.1. The words a, b, $aaaaa$, $bbba$, $baaab$, $bbaba$ and $bbababbaa$ are inverse Lyndon words on $\{a, b\}$, with $a < b$. On the contrary, $aaba$ is not an inverse Lyndon word since $aaba \prec ba$. Analogously, $aabba \prec ba$ and thus $aabba$ is not an inverse Lyndon word.

The set of inverse Lyndon words properly contains the set of the anti-Lyndon words. Indeed, by Proposition 2.7, anti-Lyndon words are inverse Lyndon words but there are inverse Lyndon words which are not anti-Lyndon words. For instance consider $\Sigma = \{a, b\}$, with $a < b$. The word bab is an inverse Lyndon word but it is not unbordered, thus it is not an anti-Lyndon word. The following result shows that the set of the inverse Lyndon words and that of the nonempty anti-prenecklaces are equal.

Proposition 3.1. *A word $w \in \Sigma^+$ is an inverse Lyndon word if and only if w is a nonempty anti-prenecklace.*

Proof:
Let $w \in \Sigma^+$ be an inverse Lyndon word. The first letter of w is an anti-Lyndon word, thus let p be the longest nonempty prefix of w which is an anti-prenecklace. By Theorem 2.1, if p were distinct from w, then we would have $p = (uav')^k u$, $w = (uav')^k ubt$, where $uav' \in L_{in}$, $u, v', t \in \Sigma^*$, $a, b \in \Sigma$, $a < b$, $k \geq 1$. Thus, $w \ll ubt$, in contradiction with Definition 3.1. Therefore, w is a nonempty anti-prenecklace. Conversely, let w be a nonempty anti-prenecklace. If $w = a^n$, where a is the minimal letter in Σ, then clearly w is an inverse Lyndon word. Otherwise, by Proposition 2.4, there is a word t such that $wt \in L_{in}$. If there existed a nonempty proper suffix s of w such that $w \prec s$, we clearly

would have $w \ll s$. Hence, by item (2) in Lemma 2.2, $wt \ll st$, where st is a nonempty proper suffix of $wt \in L_{in}$. This is in contradiction with Proposition 2.8, thus w is an inverse Lyndon word. ■

Some useful properties of the inverse Lyndon words are stated below. They are direct consequences of Definitions 2.1, 3.1 and Proposition 3.1.

Lemma 3.1. *If $w \in \Sigma^+$ is not an inverse Lyndon word, then there exists a nonempty proper suffix s of w such that $w \ll s$.*

Next lemma states that the set of the inverse Lyndon words (with the empty word) is a *prefix-closed* set, that is, it contains the prefixes of its elements.

Lemma 3.2. *Any nonempty prefix of an inverse Lyndon word is an inverse Lyndon word.*

Lemma 3.3 is needed for the definition of our new factorization of a word.

Lemma 3.3. *If $w \in \Sigma^+$ is not an inverse Lyndon word, then there exists a nonempty proper prefix p of $w = ps$ such that p is an inverse Lyndon word, $p \ll s$ and $w \ll s$.*

Proof:
Let $w \in \Sigma^+$ be a word which is not an inverse Lyndon word. The first letter of w is an anti-Lyndon word, thus let p be the longest nonempty prefix of w which is an anti-prenecklace. By Proposition 3.1, p is an inverse Lyndon word and p is a nonempty proper prefix of w. By Theorem 2.1, $p = (uav')^k u$, $w = (uav')^k ubt$, where $uav' \in L_{in}$, $u, v', t \in \Sigma^*$, $a, b \in \Sigma$, $a < b$, $k \geq 1$. Therefore $p \ll s = ubt$ and $w \ll s$. ■

Corollary 3.1. *A word $w \in \Sigma^+$ is not an inverse Lyndon word if and only if there are words $r, u, t \in \Sigma^*$ and letters $a, b \in \Sigma$, with $a < b$ such that $w = raurbt$.*

Proof:
Let $r, u, t \in \Sigma^*$ and $a, b \in \Sigma$, with $a < b$. By Definition 2.1 $w = raurbt \ll rbt$, hence w is not an inverse Lyndon word. The other direction follows by Definition 2.1 and Lemma 3.3. ■

Another class of words is that of the *Nyldon words*, defined in [24] and subsequently investigated in [7]. This class $\mathcal{N}$ of words may be recursively defined as follows: letters are in $\mathcal{N}$; a word of length at least two belongs to $\mathcal{N}$ if and only if it cannot be factorized into a (lexicographically) nondecreasing sequence of shorter words of $\mathcal{N}$. Nyldon words are inverse Lyndon words [7, Theorem 12], but there are inverse Lyndon words which are not Nyldon words. For instance, let $\Sigma = \{a, b\}$, with $a < b$. For any $n \geq 2$, the word a^n is an inverse Lyndon word which is not a Nyldon word. Indeed a^n can be factorized into the nondecreasing sequence $(w_1, \ldots, w_n)$ of shorter words $w_i = a$ of $\mathcal{N}$, $1 \leq i \leq n$.

Finally, *generalized Lyndon words*, introduced in [41] and subsequently developed in [16], deals with *generalized lexicographic orders*. These words are not inverse Lyndon words. When the inverse lexicographic order on $(\Sigma, <)$ is taken as a generalized order, the generalized Lyndon words are the anti-Lyndon words.

4. Variants of the Lyndon factorization

4.1. Inverse Lyndon factorizations

We give below the notion of an *inverse Lyndon factorization*.

Definition 4.1. Let $w \in \Sigma^+$. A sequence $(m_1, \ldots, m_k)$ of words over Σ is an inverse Lyndon factorization of w if it satisfies the following conditions

(1) $w = m_1 \cdots m_k$,
(2) for any $j \in \{1, \ldots, k\}$, the word m_j is an inverse Lyndon word,
(3) $m_1 \ll m_2 \ll \ldots \ll m_k$.

Example 4.1 shows that a word may have different inverse Lyndon factorizations even with a different number of factors.

Example 4.1. Let $\Sigma = \{a, b, c, d\}$ with $a < b < c < d$, let $w = dabadabdabdadac \in \Sigma^+$. The two sequences $(daba, dabdab, dadac)$, $(dabadab, dabda, dac)$ are both inverse Lyndon factorizations of w. Indeed,

$$w = (daba)(dabdab)(dadac) = (dabadab)(dabda)(dac).$$

Moreover, $daba, dabdab, dadac, dabadab, dabda, dac$ are all inverse Lyndon words. Furthermore,

$$daba \ll dabdab \ll dadac, \quad dabadab \ll dabda \ll dac.$$

As another example, consider the following two factorizations of $dabdadacddbdc$

$$(dab)(dadacd)(db)(dc) = (dabda)(dac)(ddbdc)$$

It is easy to see that the two sequences $(dab, dadacd, db, dc)$, $(dabda, dac, ddbdc)$ are both inverse Lyndon factorizations of $dabdadacddbdc$. The first factorization has four factors whereas the second one has three factors.

In Section 2.3 we have given the definition of a complete factorization $(x_i)_{i \in I}$ of the free monoid Σ^*. By a result of Schützenberger, if $(x_i)_{i \in I}$ is a complete factorization of Σ^*, then the set $X = \{x_i \mid i \in I\}$ is a set of representatives of the primitive conjugacy classes (see [4, Corollary 8.1.7]). In particular, any x_i is a primitive word. The fact that a

word w may have several different inverse Lyndon factorizations is a consequence of this result since an inverse Lyndon word is not necessarily primitive (take $baba$, with $a < b$, for instance).

However, we focus on a special canonical inverse Lyndon factorization, denoted by ICFL(w), which maintains three important features of CFL(w): it is uniquely determined (Proposition 4.4), it can be computed in linear time (Section 6) and it maintains the compatibility property of the nonempty suffixes with respect to the inverse lexicographic order (Theorem 7.2). We give the definition of ICFL(w) in Section 4.3, where we also compare ICFL(w) with other similar factorizations, namely the Lyndon factorization with respect to the inverse order and the Nyldon factorization (for the former, see also Section 7, Corollary 7.1). The definition of ICFL(w) is based on the definition of the *bounded right extension* of a prefix of a word, defined in Section 4.2.

4.2. The bounded right extension

The bounded right extension, abbreviated *bre*, of a prefix of a word w, defined below, allows us to define the first factor in the inverse Lyndon factorization ICFL(w).

Definition 4.2. Let $w \in \Sigma^+$, let p be an inverse Lyndon word which is a nonempty proper prefix of $w = pv$. The bounded right extension $\overline{p}_w$ of p (relatively to w), denoted by $\overline{p}$ when it is understood, is a nonempty prefix of v such that:

(1) $\overline{p}$ is an inverse Lyndon word,
(2) pz' is an inverse Lyndon word, for each proper nonempty prefix z' of $\overline{p}$,
(3) $p\overline{p}$ is not an inverse Lyndon word,
(4) $p \ll \overline{p}$.

Moreover, we set

$$\mathrm{Pref}_{bre}(w) = \{(p, \overline{p}) \mid p \text{ is an inverse Lyndon word which is a nonempty proper prefix of } w\}.$$

Notice that, given a word w, a nonempty proper prefix p of $w = pv$ can have at most one bounded right extension $\overline{p}_w$. Indeed, two different prefixes p_1, p_2 of v are comparable for the prefix order, say $p_1 \leq_p p_2$. If $p_2 = \overline{p}_w$, then pp_1 is an inverse Lyndon word and p_1 cannot be a bounded right extension of p. Analogously, if $p_1 = \overline{p}_w$, then p_2 cannot be a bounded right extension of p because p_2 does not satisfy condition (2) in Definition 4.2. Furthermore, the bounded right extension $\overline{p}_w$ of p might not exist. For instance, let $\Sigma = \{a, b, c\}$ with $a < b < c$. For the prefix ba of $baababc$, $\overline{ba}$ does not exist since any nonempty prefix p' of $ababc$ starts with a, thus $p' \ll ba$. On the contrary, for the prefix baa of $baababc$, we have $bab = \overline{baa}$. As another example, for the prefix bab of $babc$ we have $c = \overline{bab}$ but for the prefix ba of the same word $babc$, $\overline{ba}$ does not exist.

It is clear that if w is a letter, then $\mathrm{Pref}_{bre}(w)$ is empty. More precise results will be proved below. We will see that the set $\mathrm{Pref}_{bre}(w)$ is either empty or it is a singleton. In other words, given a word w, either there is no prefix of w which has a bounded right extension or this prefix is unique. $\mathrm{Pref}_{bre}(w)$ is empty if and only if w is an inverse Lyndon word. These results will be proved through Propositions 4.1, 4.2 and 4.3. For a word w which is not an inverse Lyndon word, Proposition 4.1 characterizes the pair $(p, \overline{p})$ in $\mathrm{Pref}_{bre}(w)$ through two simple conditions which allow us to compute this pair (see Section 5). Lemma 4.1 is a preliminary step.

Lemma 4.1. *Let $w \in \Sigma^+$ and let $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$. Then, there are $r, s, t \in \Sigma^*$, $a, b \in \Sigma$, with $a < b$, such that $p = ras$ and $\overline{p} = rb$.*

Proof:
Let $w \in \Sigma^+$ and let $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$. By Definition 4.2, $p \ll \overline{p}$, hence $p = ras$, $\overline{p} = rbt$, with $r, s, t \in \Sigma^*$, $a, b \in \Sigma$, $a < b$. Moreover $t = 1$, otherwise for any proper prefix t' of t, the word $z' = rbt'$ would be a proper prefix of $\overline{p}$ such that $p \ll z'$, thus $pz' \ll z'$ and pz' would not be an inverse Lyndon word, in contradiction with Definition 4.2. ∎

Proposition 4.1. *Let $w \in \Sigma^+$ be a word which is not an inverse Lyndon word. Let z be the shortest nonempty prefix of w which is not an inverse Lyndon word. Then,*

(1) *$z = p\overline{p}$, with $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$.*
(2) *$p = rau$ and $\overline{p} = rb$, where $r, u \in \Sigma^*$, $a, b \in \Sigma$ and r is the shortest prefix of $p\overline{p}$ such that $p\overline{p} = raurb$, with $a < b$.*

Proof:
Let $w \in \Sigma^+$ be a word which is not an inverse Lyndon word. Let z be the shortest nonempty prefix of w which is not an inverse Lyndon word. Thus, $z = yb$, where y is an inverse Lyndon word. Since y is an inverse Lyndon word and $z = yb$ is not an inverse Lyndon word, by Proposition 3.1 and Theorem 2.1, there are $r, u \in \Sigma^*$, $a, b \in \Sigma$, $a < b$, such that $z = raurb$.

Assume that r is the shortest prefix of z such that the above condition holds. Set $p = rau$ and $\overline{p} = rb$. We claim that $(p, \overline{p}) = (rau, rb) \in \mathrm{Pref}_{bre}(w)$. By Lemma 3.2, any nonempty prefix of $y = raur$ is an inverse Lyndon word. Moreover $p = rau \ll rb = \overline{p}$. It remains to prove that rb is an inverse Lyndon word.

If rb were not an inverse Lyndon word, then $r \neq 1$ (otherwise, $\overline{p} = b$ would be an inverse Lyndon word). Moreover, by Lemma 3.3, there would exist a nonempty proper prefix q of $\overline{p} = rb = qt$ such that $q \ll t$. Notice that q would be a prefix of $r = qq'$, thus $z = p\overline{p} = pqt = rauqt = qq'auqt$. Therefore, by item (2) in Lemma 2.2 and $q \ll t$, we get $pq \ll t$, hence $pq = r'cu'$, $t = r'b$, with $r', u' \in \Sigma^*$, $c \in \Sigma$, $c < b$. By $qq'b = rb = qt$ and $t = r'b$, we get $q' = r'$, hence $z = p\overline{p} = pqt = r'cu'r'b = q'cu'q'b$. Of course

$|q'| < |r| = |qq'|$, because q is nonempty, and this contradicts our assumption on the length of r. ■

Proposition 4.2. *Let $w \in \Sigma^+$ be a nonempty word. The following statements are equivalent.*

(1) $\mathrm{Pref}_{bre}(w) = \emptyset$.
(2) *w is an inverse Lyndon word.*
(3) *Any nonempty prefix of w is an inverse Lyndon word.*

Proof:
To prove (1) ⇒ (2), we use a proof by contraposition. If w is not an inverse Lyndon word, then $\mathrm{Pref}_{bre}(w) \neq \emptyset$ by Proposition 4.1. (2) ⇒ (3) is a direct consequence of Lemma 3.2. To prove (3) ⇒ (1), we use again a proof by contraposition. If $\mathrm{Pref}_{bre}(w) \neq \emptyset$, then let $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$. The word $p\overline{p}$ is a nonempty prefix of w which is not an inverse Lyndon word. ■

Proposition 4.3. *If a word $w \in \Sigma^+$ is not an inverse Lyndon word, then* $\mathrm{Card}(\mathrm{Pref}_{bre}(w)) = 1$.

Proof:
Let $w \in \Sigma^+$ be a word which is not an inverse Lyndon word. By Proposition 4.1, the set $\mathrm{Pref}_{bre}(w)$ is nonempty. More precisely, for the shortest nonempty prefix z of w which is not an inverse Lyndon word, we have $z = p\overline{p}$, with $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$ and there are $r, u \in \Sigma^*$, $a, b \in \Sigma$, $a < b$, such that $p = rau$, $\overline{p} = rb$. Moreover, r is the shortest prefix of $p\overline{p}$ such that $p\overline{p} = raurb$, with $a < b$. By contradiction, let $(q, \overline{q}) \in \mathrm{Pref}_{bre}(w)$, with $p \neq q$. Since $p\overline{p}$ and $q\overline{q}$ are both prefixes of w, they are comparable for the prefix order and one of the following three cases holds.

(1) $p\overline{p}$ is a proper prefix of $q\overline{q}$,
(2) $q\overline{q}$ is a proper prefix of $p\overline{p}$,
(3) $p\overline{p} = q\overline{q}$.

In case (1), either $p\overline{p}$ is a prefix of q or $p\overline{p} = qz'$, for a proper prefix z' of $\overline{q}$. Since $p\overline{p}$ is not an inverse Lyndon word, both cases are impossible (the first contradicts Lemma 3.2, the second Definition 4.2). We may exclude case (2) by a similar reasoning, thus assume that $p\overline{p} = q\overline{q}$. By Lemma 4.1, there are $r', s' \in \Sigma^*$, $a', b' \in \Sigma$, with $a' < b'$, such that $q = r'a's'$, $\overline{q} = r'b'$. Hence

$$b = b', \quad p\overline{p} = q\overline{q} = rasrb = r'a's'r'b, \quad a < b,\ a' < b. \tag{4.1}$$

Since r' satisfies item (2) in Proposition 4.1, we have $|r| = |r'|$. Thus, by Eq. (4.1), $r = r'$ which implies $\overline{p} = \overline{q}$ and $p = q$. ■

Example 4.2. Let $\Sigma = \{a, b, c, d\}$ with $a < b < c < d$, let $w = cbabcbad \in \Sigma^+$. The word w is not an inverse Lyndon word but any nonempty proper prefix of w is an inverse Lyndon word. By Proposition 4.1, item (1), we have $w = p\overline{p}$, with $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$. We have $cbab \ll cbad$ but cba does not satisfy item (2) in Proposition 4.1. Notice that $cbad$ is not an inverse Lyndon word. Since $cbabcba \ll d$, the (shortest) prefix r of $w = p\overline{p}$ satisfying item (2) in Proposition 4.1 is the empty word. Consequently, $(p, \overline{p}) = (cbabcba, d)$.

Example 4.3. Let $\Sigma = \{a, b, c, d\}$ with $a < b < c < d$, let $v = dabdabdadac$. We can check that $dabdabdad$ is the shortest nonempty prefix of v which is not an inverse Lyndon word, hence $dabdabdad = p\overline{p}$, with $(p, \overline{p}) \in \mathrm{Pref}_{bre}(v)$, according to item (1) in Proposition 4.1. The prefix r of $p\overline{p}$ satisfying item (2) in Proposition 4.1 is da, thus $(p, \overline{p}) = (dabdab, dad) \in \mathrm{Pref}_{bre}(v)$. As another example, consider $w = dabadabdabdadac$. We can check that $dabadabd$ is the shortest nonempty prefix of w which is not an inverse Lyndon word, hence $dabadabd = q\overline{q}$, with $(q, \overline{q}) \in \mathrm{Pref}_{bre}(w)$, according to item (1) in Proposition 4.1. The prefix r of $q\overline{q}$ satisfying item (2) in Lemma 4.1 is dab, thus $(q, \overline{q}) = (daba, dabd) \in \mathrm{Pref}_{bre}(w)$.

Example 4.4. Let $\Sigma = \{a, b, c\}$ with $a < b < c$, let $v = cbabacbac$. We can check that $v = cbabacbac$ is the shortest nonempty prefix of v which is not an inverse Lyndon word, hence $v = cbabacbac = p\overline{p}$, with $(p, \overline{p}) \in \mathrm{Pref}_{bre}(v)$, according to item (1) in Proposition 4.1. The prefix r of $p\overline{p}$ satisfying item (2) in Lemma 4.1 is cba, thus $(p, \overline{p}) = (cbaba, cbac) \in \mathrm{Pref}_{bre}(v)$. As another example, consider $w = cbabacaacbabacbac$. We can check that $cbabacaacbabacb$ is the shortest nonempty prefix of w which is not an inverse Lyndon word, hence $cbabacaacbabacb = q\overline{q}$, with $(q, \overline{q}) \in \mathrm{Pref}_{bre}(w)$, according to item (1) in Proposition 4.1. The prefix r of $q\overline{q}$ satisfying item (2) in Lemma 4.1 is $cbabac$, thus $(q, \overline{q}) = (cbabacaa, cbabacb) \in \mathrm{Pref}_{bre}(w)$.

4.3. A canonical inverse Lyndon factorization: ICFL(w)

We give below the recursive definition of the canonical inverse Lyndon factorization ICFL(w).

Definition 4.3. Let $w \in \Sigma^+$.

(Basis Step) If w is an inverse Lyndon word, then $\mathrm{ICFL}(w) = (w)$.

(Recursive Step) If w is not an inverse Lyndon word, let $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$ and let $v \in \Sigma^*$ such that $w = pv$. Let $\mathrm{ICFL}(v) = (m'_1, \ldots, m'_k)$ and let $r \in \Sigma^*$ and $a, b \in \Sigma$ such that $p = rax$, $\overline{p} = rb$ with $a < b$.

$$\mathrm{ICFL}(w) = \begin{cases} (p, \mathrm{ICFL}(v)) & \text{if } \overline{p} = rb \leq_p m'_1 \\ (pm'_1, m'_2, \ldots, m'_k) & \text{if } m'_1 \leq_p r \end{cases}$$

Remark 4.1. With the same notations as in the recursive step of Definition 4.3, we notice that rb and m'_1 are both prefixes of the same word v. Thus they are comparable for the prefix order, that is, either $rb \leq_p m'_1$ or $m'_1 \leq_p r$.

Example 4.5. Let $\Sigma = \{a, b, c\}$ with $a < b < c$, let $v = cbabacbac$. Let us compute ICFL(v). As showed in Example 4.4, we have $(p, \bar{p}) = (cbaba, cbac) \in \text{Pref}_{bre}(v)$. Therefore, $cbaba, cbac$ are both inverse Lyndon words and we have ICFL($cbaba$) = ($cbaba$), ICFL($\bar{p}$) = ICFL($cbac$) = (m') = ($cbac$). Since $\bar{p} = cbac \leq_p m'$, the first case of Definition 4.3 applies, thus ICFL(v) = (p, ICFL($\bar{p}$)) = ($cbaba, cbac$) = (m'_1, m'_2). Now, let $w = cbabacaacbabacbac$ and let us compute ICFL(w). In Example 4.4 we showed that $(q, \bar{q}) = (cbabacaa, cbabacb) \in \text{Pref}_{bre}(w)$. Thus, $w = qv$, where $v = cbabacbac$ is the above considered word and $\bar{q} = r'b$, where $r' = cbabac$. Since $m'_1 = cbaba \leq_p cbabac$, we are in the second case of Definition 4.3, therefore ICFL(w) = (qm'_1, m'_2) = ($cbabacaacbaba, cbac$).

Example 4.6. Let $\Sigma = \{a, b, c, d\}$ with $a < b < c < d$, let $v = dabdabdadac$. Let us compute ICFL(v). As showed in Example 4.3, we have $(p, \bar{p}) = (dabdab, dad) \in \text{Pref}_{bre}(v)$. Hence, $v = pv'$, where $v' = dadac = \overline{p}ac$. On the other hand, we can easily see that $dadac$ is an inverse Lyndon word, hence ICFL($dadac$) = ($dadac$). Since $\overline{p} = dad \leq_p dadac$, by Definition 4.3 (first case), ICFL(v) = ($dabdab, dadac$). Now, let us compute ICFL(w), where $w = dabadabdabdadac$. In Example 4.3 we showed that $(q, \bar{q}) = (daba, dabd) \in \text{Pref}_{bre}(w)$. Thus, $w = qv$, where v is the above considered word. Since $\overline{q} = dabd \leq_p dabdab$, by Definition 4.3 (first case), ICFL(w) = ($daba, dabdab, dadac$). In Example 4.1 we showed two different inverse Lyndon factorizations of $w' = dabdadacddbdc$. Both are different from ICFL(w') = ($dab, dadac, ddbdc$).

As proved below, ICFL(w) is uniquely determined.

Proposition 4.4. *For any word $w \in \Sigma^+$, there is a unique sequence $(m_1, \ldots, m_k)$ of words over Σ such that* ICFL(w) = $(m_1, \ldots, m_k)$.

Proof:
The proof is by induction on $|w|$. If w is a letter, then the statement is proved since w is an inverse Lyndon word and ICFL(w) = (w) (Definition 4.3). Thus, assume $|w| > 1$. If w is an inverse Lyndon word we are done since, by Definition 4.3, ICFL(w) = (w). Otherwise, w is not an inverse Lyndon word and there is a unique pair $(p, \overline{p})$ in $\text{Pref}_{bre}(w)$ (Proposition 4.3). Let $v \in \Sigma^+$ such that $w = pv$. Since $|v| < |w|$, by induction hypothesis there is a unique sequence $(m'_1, \ldots, m'_{k'})$ of words such that ICFL(v) = $(m'_1, \ldots, m'_{k'})$. Let $r \in \Sigma^*$ and $a, b \in \Sigma$ such that $p = rax$, $\overline{p} = rb$ with $a < b$. By the recursive step of Definition 4.3, we have

$$\text{ICFL}(w) = \begin{cases} (p, \text{ICFL}(v)) & \text{if } rb \leq_p m'_1 \\ (pm'_1, m'_2, \ldots, m'_{k'}) & \text{if } m'_1 \leq_p r \end{cases}$$

In both cases, by the above arguments, the sequence ICFL(w) is uniquely determined. ■

As a main result, we now prove that ICFL(w) is an inverse Lyndon factorization of w.

Lemma 4.2. *For any $w \in \Sigma^+$, the sequence* ICFL(w) $= (m_1, \ldots, m_k)$ *is an inverse Lyndon factorization of w, that is, $w = m_1 \cdots m_k$, $m_1 \ll \ldots \ll m_k$ and each m_i is an inverse Lyndon word.*

Proof:
The proof is by induction on $|w|$. If w is a letter, then the statement is proved since w is an inverse Lyndon word and ICFL(w) $= (w)$ (Definition 4.3). Thus, assume $|w| > 1$. If $k = 1$ we are done since, by Definition 4.3, w is an inverse Lyndon word and ICFL(w) $= (w)$. Otherwise, w is not an inverse Lyndon word. Let $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$ and let $v \in \Sigma^+$ such that $w = pv$. Let ICFL(v) $= (m'_1, \ldots, m'_{k'})$ and let $r, x \in \Sigma^*$ and $a, b \in \Sigma$ be such that $p = rax$, $\overline{p} = rb$ with $a < b$. By the recursive step of Definition 4.3, we have

$$\mathrm{ICFL}(w) = \begin{cases} (p, \mathrm{ICFL}(v)) & \text{if } rb \leq_p m'_1 \\ (pm'_1, m'_2, \ldots, m'_{k'}) & \text{if } m'_1 \leq_p r \end{cases}$$

Moreover, since $|v| < |w|$, by induction hypothesis, $v = m'_1 \cdots m'_{k'}$, $m'_1 \ll \ldots \ll m'_{k'}$ and each m'_i is an inverse Lyndon word. Therefore, $w = pv = pm'_1 \cdots m'_{k'}$.

If $rb \leq_p m'_1$, then there is $z \in \Sigma^*$ such that $m'_1 = rbz$ and ICFL(w) $= (p, m'_1, \ldots, m'_{k'})$. By Definition 4.2, p is an inverse Lyndon word and so are all the words in ICFL(w). Furthermore, $p \ll m'_1 \ll \ldots \ll m'_{k'}$ and the proof is ended. Otherwise, there is $z \in \Sigma^*$ such that $r = m'_1 z$, hence $p = rax = m'_1 zax$ and ICFL(w) $= (pm'_1, \ldots, m'_{k'})$. The word pm'_1 is a proper prefix of $p\overline{p}$, thus, by Definition 4.2, pm'_1 is an inverse Lyndon word and so are all words in ICFL(w). Finally, by $m'_1 \ll m'_2 \ll \ldots \ll m'_{k'}$ we have $pm'_1 = m'_1 zaxm'_1 \ll m'_2 \ll \ldots \ll m'_{k'}$ (Lemma 2.2). ■

We end this section with the following result showing that inverse Lyndon factorizations of Lyndon words are not trivial.

Proposition 4.5. *Any $w \in \Sigma^+$ has an inverse Lyndon factorization $(m_1, \ldots, m_k)$. Moreover, if w is a Lyndon word which is not a letter, then $k > 1$.*

Proof:
The first part of the statement follows by Proposition 4.4 and Lemma 4.2. Assume that w is a Lyndon word which is not a letter and let $(m_1, \ldots, m_k)$ be one of its inverse Lyndon factorizations. By Proposition 2.3, there is a proper nonempty suffix v of w such that $w \prec v$. Hence, w is not an inverse Lyndon word, which yields $w \neq m_1$ and consequently $k > 1$. ■

Of course a converse of Proposition 4.5 can also be stated.

Proposition 4.6. *Let $w \in \Sigma^+$ and let* $\mathrm{CFL}(w) = (\ell_1, \ldots, \ell_h)$. *If w is an inverse Lyndon word which is not a letter, then $h > 1$.*

Proof:
Let w be an inverse Lyndon word which is not a letter and let $\mathrm{CFL}(w) = (\ell_1, \ldots, \ell_h)$. By Definition 3.1, there is a proper nonempty suffix v of w such that $v \prec w$. Hence, by Proposition 2.3, w is not a Lyndon word, which yields $w \neq \ell_1$ and consequently $h > 1$. ■

As the following example shows, $\mathrm{ICFL}(w)$ is in general different from the Lyndon factorization $\mathrm{CFL}_{in}(w)$ with respect to the inverse order, even if we concatenate consecutive equal factors. $\mathrm{ICFL}(w)$ is also different from the Nyldon factorization $\mathrm{Nyl}(w)$ of w, defined in [7] as the unique nondecreasing sequence of Nyldon words that factorizes w.

Example 4.7. Let $\Sigma = \{a, b\}$ with $a < b$. We have $\mathrm{ICFL}(aab) = (aa, b)$ and $\mathrm{CFL}_{in}(aab) = \mathrm{Nyl}(aab) = (a, a, b)$ [7, Theorem 13]. As another example, $\mathrm{CFL}_{in}(bab) = (ba, b)$ whereas $\mathrm{ICFL}(bab) = \mathrm{Nyl}(bab) = (bab)$. Finally, consider $w = cbabcbad$ over $\{a, b, c, d\}$ with $a < b < c < d$. We have $\mathrm{CFL}_{in}(w) = (cbab, cba, d)$ and $\mathrm{ICFL}(w) = (cbabcba, d)$.

More precise relations between $\mathrm{ICFL}(w)$ and $\mathrm{CFL}_{in}(w)$ are pointed out in Section 7. In particular, Corollary 7.1 deals with a special case in which the two factorizations are equal.

5. An algorithm for finding the bounded right extension

In Section 6 we will give a linear time recursive algorithm, called **Compute-ICFL**, that computes $\mathrm{ICFL}(w)$, for a given nonempty word w. By Definition 4.3, we know that the computation of $\mathrm{ICFL}(w)$, when w is not an inverse Lyndon word, is based on that of the pair $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$. In this section we give algorithms to compute the above pair $(p, \overline{p})$. By Proposition 4.1, we are faced with the problems of

(1) stating whether w is an inverse Lyndon word;
(2) if not, finding the shortest prefix x of w such that $x = raurb$, where $r, u \in \Sigma^*$, $a, b \in \Sigma$, $a < b$. Therefore, $x = p\bar{p}$.
(3) Finding the shortest r such that $x = raurb$, with r, u, a, b as in (2). Thus $p = rau$, $\bar{p} = rb$.

The first two tasks are carried out by algorithm **Find-prefix**, the third is performed by algorithm **Find-bre**. Algorithm **Compute-ICFL** uses algorithm **Find-bre** as a subroutine. In turn, the latter uses the output of algorithm **Find-prefix**. Algorithm **Find-bre** also calls a procedure to compute the well known *failure function* of the Knuth-Morris-Pratt matching algorithm [28].

Firstly, we give a high-level description of algorithm **Find-prefix** followed by its pseudocode (Section 5.1) and some loop invariants to prove its correctness (Section 5.2). Next, in Section 5.3, we recall the definition of the failure function and some known results concerning this function which will be useful later. Finally, we give a high-level description of algorithm **Find-bre** followed by its pseudocode (Section 5.4) and we end the section with the proof of its correctness through some loop invariants (Section 5.5).

5.1. Description of Find-prefix

Algorithm **Find-prefix** is very similar to Duval's algorithm [17,13]. We first briefly discuss why the former is a variant of the latter. Given w, algorithm **Find-prefix** looks for its shortest prefix w' which is not an inverse Lyndon word, if such shortest prefix w' exists. **Find-prefix** outputs $w\$$ if w' does not exist, i.e., if w is an inverse Lyndon word. Assume that w' exists. By Proposition 3.1, Theorem 2.1 and Corollary 2.1, we know that $w' = (xav')^k xb$, where $xav' \in L_{in}$ is a Lyndon word with respect to the inverse order, $x, v' \in \Sigma^*$, $a, b \in \Sigma$, $k \geq 1$. Moreover, $(xav')^k x$ is the longest prefix of w which is a necklace, thus the word xav' is the longest prefix of w which is in L_{in}. Duval's algorithm looks for the longest Lyndon prefix of w by looking for the longest prefix of w which is a necklace. Therefore, Duval's algorithm and **Find-prefix** work in the same way and differ only with respect to their outputs: the former outputs k times the word xav' (and then it recursively applies to the rest of the word), whereas the latter outputs $w' = (xav')^k xb = raurb$, where $r, u \in \Sigma^*$.

Let us describe the high-level structure of algorithm **Find-prefix**. We refer to Duval's algorithm as presented in [22]. As usual, the word is represented as an array $w[1..n]$ containing the sequence of the letters in w, with $n = |w|$. The algorithm uses two indices i, j to scan the word w. Initially, these indices denote the position of the first letter of a candidate common prefix r.

A while-loop is used to compare the two letters $w[i]$ and $w[j]$. While j is incremented at each iteration, i is incremented only when $w[i] = w[j]$. Notice that the algorithm does not test if eventually the two occurrences of r overlap. Lemma 2.1 allows us to avoid this test.

If $w[i] > w[j]$, then the algorithm resets i to the first position of w and examines a new candidate common prefix r, whose position of the first letter in the second occurrence of r is indicated by the value of j at this time.

The loop condition is false when $w[i] < w[j]$ or when j denotes the last letter of w. In the first case, our search has been successful and the algorithm returns $raurb$. If j denotes the last letter of w (and $w[i] \geq w[j]$), then w has no prefix $raurb$, with r, u, a, b as above, and the algorithm returns $w\$$, where $\$$ is a letter such that $\$ \notin \Sigma$. In this case, by Corollary 3.1, we know that w is an inverse Lyndon word.

In conclusion, **Find-prefix** allows us to state whether w is an inverse Lyndon word or not and, in the latter case, it finds the prefix x of w such that $x = p\bar{p}$, with $(p, \bar{p}) \in$

$\mathrm{Pref}_{bre}(w)$. Algorithm 1 describes the procedure **Find-prefix**. It is understood that the empty array, namely $w[j+1..n]$ with $j = n$, represents the empty word.

Algorithm 1: Find-prefix.

```
   Input : A string w
   Output: A pair of strings (x, y), where x = w$, y = 1 if w is an inverse Lyndon word, xy = w, x = pp̄,
           with (p, p̄) ∈ Pref_bre(w), otherwise.
 1 if |w| = 1 then
 2 |   return (w$, 1);
 3 i ← 1;
 4 j ← 2;
 5 while j < |w| and w[j] ≤ w[i] do
 6 |   if w[j] < w[i] then
 7 |   |   i ← 1;
 8 |   else
 9 |   |   i ← i + 1
10 |   j ← j + 1;
11 if j = |w| then
12 |   if w[j] ≤ w[i] then
13 |   |   return (w$,1);
14 return (w[1..j], w[j + 1..|w|]);
```

We notice that we can reach line (11) in two different cases: either when $x = w$ or when w is an inverse Lyndon word. We distinguish these two cases: in the former case the output is $(w, 1)$ and in the latter case the output is $(w\$, 1)$ (see example below).

Example 5.1. Let $\Sigma = \{a, b, c\}$ with $a < b < c$. Algorithm **Find-prefix**(bac) returns $(bac, 1)$. Indeed, $|w| = 3$ and $w[2] < w[1]$ in the first iteration of the while-loop. Then, $i = 1$ (line (7)), $j = 3$ (line (10)), and we reach line (11) with $j = 3 = |w|$. Since $w[j] = w[3] = c > w[1] = w[i]$, the algorithm returns $(w[1..j], w[j+1..|w|]) = (bac, 1)$.

Consider now the inverse Lyndon word $w = bab$. Algorithm **Find-prefix**(bab) returns $(bab\$, 1)$. Indeed, $|w| = 3$ and, as before, $w[2] < w[1]$ in the first iteration of the while-loop. Then, $i = 1$ (line (7)), $j = 3$ (line (10)), and we reach line (11) with $j = 3 = |w|$. Since $w[j] = w[3] = b = w[1] = w[i]$, the algorithm returns $(bab\$, 1)$. The same argument applies when $w = baa$.

Example 5.2. Let $\Sigma = \{a, b\}$ with $a < b$. Algorithm **Find-prefix**$(bbabbabbb)$ outputs $(bbabbabbb, 1)$.

5.2. Correctness of Find-prefix

We will not prove formally the correctness of **Find-prefix** since it follows from the correctness of Duval's algorithm. However in this section we indicate what are the main properties useful to prove it. To begin, notice that each time around the while-loop of lines (5) to (10), j is increased by 1 at line (10). Therefore, when j becomes equal to $|w|$, if we do not break out of the while-loop earlier, the loop condition $j < |w|$ will be false and the loop will terminate. In order to prove that **Find-prefix** does what it is intended

to do, we define the following loop-invariant statement, where we use k to stand for one of the values that the variable j assumes and h_k for the corresponding variable i, as we go around the loop. Finally h'_k depends on h_k and k.

$S(k)$: If we reach the loop test "$j < |w|$ and $w[j] \leq w[i]$" with the variable i having the value h_k and the variable j having the value k, then

(a) $1 \leq h_k < k$. Set $h'_k = k - h_k + 1$. We have $1 < h'_k \leq |w|$ and the following conditions hold
  (a1) $w[1..h_k-1] = w[h'_k..k-1]$, that is, $w[1..h_k-1]$ is a proper prefix of $w[1..k-1]$ and $w[1..h_k-1]$ is also a suffix of $w[1..k-1]$.
  (a2) For any t', $1 < t' < h'_k$, $w[t'..k-1]$ is not a prefix of $w[1..k-1]$.

(b) $w[1..k-1]$ has no prefix with the form $raurb$, $r, u \in \Sigma^*$, $a, b \in \Sigma$, $a < b$. Therefore, by Corollary 3.1, $w[1..k-1]$ is an inverse Lyndon word.

Remark 5.1. Notice that $w[t'..k-1]$ is nonempty, for any t', $1 < t' < h'_k$. Indeed, $h'_k = k - h_k + 1 \leq k$. Thus, since $t' \leq h'_k - 1 \leq k - 1$, we have $|w[t'..k-1]| \geq |w[h'_k - 1..k-1]| \geq |w[k-1..k-1]| = |w[k-1]| = 1$.

Loosely speaking, at the beginning of each iteration of the loop of lines (5)–(10), indices i, j store the end of a candidate common prefix r (item (a1)) and we do not yet find a prefix with the required form in the examined part of the array (items (a2), (b)). Note that the examined part may be a single letter and r could be the empty word.

One can prove that $S(k)$ is true, for any $k \geq 2$, by (complete) induction on k. The invariant provides a useful property to prove correctness when the loop terminates.

Proposition 5.1. *Algorithm* ***Find-prefix*** *allows us to state whether w is an inverse Lyndon word or not and, in the latter case, it finds the shortest prefix x of w such that $x = raurb$ where $r, u \in \Sigma^*$, $a, b \in \Sigma$, $a < b$. More precisely, if w is an inverse Lyndon word, then* ***Find-prefix*** *outputs $(w\$, 1)$; if w is not an inverse Lyndon word, then* ***Find-prefix*** *outputs $(p\bar{p}, y)$, where $w = p\bar{p}y$ and $(p, \bar{p}) \in \mathrm{Pref}_{bre}(w)$.*

5.3. The failure function

We recall that, given a nonempty word w, a *border* of w is a word which is both a proper prefix and a suffix of w [12]. The longest proper prefix of w which is a suffix of w is also called *the border* of w [33,12]. It is also known that, given w represented by the array $w[1..n]$, the *failure function* for w, is the function $f : \{1, \ldots, n\} \to \{0, \ldots, n-1\}$ such that

$$f(i) = \max\{k \mid k < i \text{ and } w[1..k] \text{ is a suffix of } w[1..i]\}.$$

That is, $f(i)$ is the length of the border of $w[1..i]$. Proposition 1.5 in [12], reported in Lemma 5.1, states that by iterating the failure function f, we can enumerate the lengths of all the borders of a prefix $w[1..i]$ in decreasing order.

Definition 5.1. For a positive integer i, we set

$$f^{(0)}(i) = i$$
$$f^{(\ell)}(i) = f(f^{(\ell-1)}(i)), \text{ for } \ell \geq 1$$
$$f^*(i) = \{i, f(i), f^{(2)}(i), \ldots, f^{(m)}(i)\},$$

where it is understood that the sequence in $f^*(i)$ stops when $f^{(m)}(i) = 0$ is reached.

Lemma 5.1. *Let w be a word of length n with failure function f. For any i, $1 \leq i \leq n$, $f^*(i)$ lists the sequence of the lengths of all the borders of $w[1..i]$ in decreasing order. That is,*

(1) *a word x is a border of $w[1..i]$ if and only if there is k, $0 \leq k \leq \mathrm{Card}(f^*(i)) - 1$, such that $x = w[1..f^{(k)}(i)]$.*
(2) *If $k < k'$, then $|w[1..f^{(k')}(i)]| < |w[1..f^{(k)}(i)]|$ and $w[1..f^{(k')}(i)]$ is a border of $w[1..f^{(k)}(i)]$.*

It is known that there is an algorithm that outputs the array $\mathcal{F}$ of $n = |w|$ integers such that $\mathcal{F}[i] = f(i)$ is the length of the border of $w[1..i]$ in time $\mathcal{O}(n)$ (for a description of this procedure see [10], where it is called **Compute-Prefix-Function**, or [33], where it is called **Border**, or [12], where it is called **Borders**). Algorithm **Find-bre** needs the lengths of the borders of a specific word w. It is clear that the array $\mathcal{F}$ determines the set $f^*(n) \setminus \{n\} = \{f(n), f^{(2)}(n), \ldots, f^{(m)}(n)\}$ of these lengths. Indeed, $\mathcal{F}[n] = f(n)$ and, if $f^k(n) = \mathcal{F}[i] = f(i) = j$, then $f^{k+1}(n) = f(f^k(n)) = f(j) = \mathcal{F}[j]$.

Example 5.3. Let us consider again $w = bbabbabbb$ as in Example 5.2. We have that $\mathcal{F} = [0, 1, 0, 1, 2, 3, 4, 5, 2]$.

5.4. Description of Find-bre

In this section we present **Find-bre**, which applies to the output (x, y) of **Find-prefix**(w) when w is not an inverse Lyndon word. In this case, $x = p\overline{p}$, where $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$. As already said, the task of **Find-bre**(x, y) is to find the shortest r such that $x = raurb$, where $r, u \in \Sigma^*$, $a, b \in \Sigma$, $a < b$. Hence, by Proposition 4.1, we have $p = rau$ and $\bar{p} = rb$. Therefore, **Find-bre**(x, y) computes the prefix p and its bounded right extension $\overline{p}$ and outputs the quadruple $(p, \overline{p}, y, |r|)$. **Find-bre** uses the array $\mathcal{F}$ computed by **Border**$(raur)$. Algorithm 2 describes the procedure **Find-bre**.

Algorithm 2: Find-bre.

Input : A pair of strings (x, y), where $w = xy$ is not an inverse Lyndon word, $x = p\bar{p} = raurb$, with $(p, \bar{p}) \in \mathrm{Pref}_{bre}(w)$, $n = |raur| = |x| - 1$. The array $\mathcal{F}$ computed by **Border**$(raur)$.
Output: A quadruple (x_1, x_2, y, x_3), where $(x_1, x_2, y, x_3) = (rau, rb, y, |r|)$.

```
1 i ← n;
2 LAST ← n + 1;
3 while i > 0 do
4 |   if w[f(i) + 1] < b then
5 |   |   LAST = f(i);
6 |   i ← f(i);
7 return (w[1..n − LAST], w[n − LAST +1, n]b, y, LAST) ;
```

Example 5.4. Let $\Sigma = \{a, b\}$ with $a < b$, let $w = bbabbabbb$. In Example 5.2 we noticed that **Find-prefix**$(bbabbabbb)$ outputs $(bbabbabbb, 1)$. We can check that **Find-bre**$(bbabbabbb, 1)$ returns $(bbabba, bbb, 1, 2)$.

5.5. Correctness of Find-bre

In this section we prove that **Find-bre** does what it is claimed to do. To begin, notice that each time around the while-loop of lines (3) to (6), i decreases since $f(i) < i$. Thus when i becomes zero, the loop condition $i > 0$ will be false and the loop will terminate. Then, consider the following loop-invariant statement.

$S(t)$: If we reach the loop test "$i > 0$" with the variable i having the value h and the variable LAST having the value k, after t iterations of the while-loop, then

- $h = f^t(n)$.
- $k \geq h$. Precisely, if $w[h + 1] < w[n + 1]$, then $k = h$ else $k > h$.

Proposition 5.2. *For any $t \geq 0$, $S(t)$ is true.*

Proof:

(Basis) Let us prove that $S(t)$ is true for $t = 0$. We reach the test after 0 iterations of the while-loop only when we enter the loop from the outside. Prior to the loop, lines (1) and (2) set i to $n = f^0(n)$ and LAST to $n + 1$. Hence, we reach the test after 0 iterations of the while-loop with $h = f^0(n)$ and $k > h$. Thus, since $w[h + 1] = w[n + 1]$, clearly $S(0)$ is true.

(Induction) We suppose that $S(t)$ is true and prove that $S(t + 1)$ is true. Therefore, after t iterations of the while-loop, we reach the loop test "$i > 0$", with the variable i having the value $h = f^t(n)$. We may assume $h > 0$ (otherwise we break out of the while-loop after t iterations or earlier and $S(t+1)$ is clearly true, since it is a conditional expression with a false antecedent). Let us consider what happens when we run the $(t + 1)$th iteration of the while-loop and we execute the body of the while-loop with i having the value h and LAST having the value k.

The variable i assumes value $f(h) = f^{t+1}(n)$ on line (6). Moreover, if $w[f(h) + 1] = w[f^{t+1}(n) + 1] < w[n + 1]$, then LAST assumes value $f^{t+1}(n)$ (lines (4)–(5)). Otherwise LAST remains unchanged, hence $k \geq h = f^t(n) > f^{t+1}(n)$ (induction hypothesis). In both cases, $S(t + 1)$ is true. ■

Proposition 5.3. *Let $w \in \Sigma^+$ be a word which is not an inverse Lyndon word. Algorithm* ***Find-bre****, applied to the output (x, y) of* ***Find-prefix***(w)*, outputs the quadruple $(p, \overline{p}, y, |r|)$, where $w = xy = p\overline{p}y$, $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$, $p = rau$, $\overline{p} = rb$.*

Proof:
Let $w \in \Sigma^+$ be a word which is not an inverse Lyndon word. By Proposition 5.1, if (x, y) is the output of **Find-prefix**(w), then $x = p\overline{p}$, where $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$, and $w = p\overline{p}y$. Let z be the prefix of x such that $|z| = |x| - 1 = n$. By Proposition 4.1, it suffices to prove that **Find-bre** outputs $(rau, rb, y, |r|)$, where r is the shortest prefix and suffix of z such that $zb = x = p\overline{p} = raurb$, with $r, u \in \Sigma^*$, $a, b \in \Sigma$, $a < b$.

Of course after $m = \mathrm{Card}(f^*(n)) - 1$ iterations, the while-loop of lines (3) to (6) terminates. Recall also that the sequence $f^*(n)$ is strictly decreasing (Lemma 5.1). Let r be the shortest prefix and suffix of z such that $z = raur$ and $w[|r|+1] = a < b = w[n+1]$. By Lemma 5.1, there is $q \geq 0$ such that $|r| = f^q(n)$. Moreover, at line (6), the integer t such that $i = f^t(n)$ is incremented by 1 each time around the loop. Therefore, after q iterations of the while-loop we have $i = |r| = f^q(n)$. By Proposition 5.2, $S(q)$ is true. Thus, $i = |r| = f^q(n) = \mathrm{LAST}$ since $w[|r| + 1] = w[i + 1] = a < b = w[n + 1]$. This value of LAST remains unchanged until we break out of the while-loop since, otherwise, for $s > q$, we would have $w[f^s(n) + 1] = c < w[n + 1]$ and there would exist a shorter word $r' = w[1..f^s(n)]$ such that $z = r'u'r'$, where u' starts with c, a contradiction. Finally, **Find-bre** outputs $(w[1..n - \mathrm{LAST}], w[n - \mathrm{LAST} + 1..n]b, y, \mathrm{LAST})$. Now $w[n - \mathrm{LAST} + 1..n]$ is the suffix of z of length $\mathrm{LAST} = |r|$, hence $w[n - \mathrm{LAST} + 1..n] = r$. Of course $w[1..n - \mathrm{LAST}] = rau$. ∎

6. Computing ICFL in linear time

In this section we give a linear time algorithm, called **Compute-ICFL** to compute $\mathrm{ICFL}(w)$. For the sake of simplicity we present a recursive version of **Compute-ICFL**. In this case the correctness of the algorithm easily follows from the definition of ICFL. The output of **Compute-ICFL** is represented as a list denoted by *list*.

Let us describe the high-level structure of algorithm **Compute-ICFL**(w). The algorithm firstly calls **Find-prefix**(w) (line (1)), that, in view of Proposition 5.1, allows us to state whether w is an inverse Lyndon word or not.

If w is an inverse Lyndon word, then **Find-prefix**(w) returns $(x, y) = (w\$, 1)$ and **Compute-ICFL** stops and returns (w) (lines (2)–(3)), according to Definition 4.3. If w is not an inverse Lyndon word, then **Find-prefix**(w) returns the pair (x, y) such that $w = xy = p\overline{p}y$, $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$, and **Compute-ICFL** calls **Find-bre**(x, y) (line (4)). In turn, **Find-bre**(x, y) returns a quadruple $(x_1, x_2, y, \mathrm{LAST})$, where $x_1 = p$, $x_2 = \overline{p}$ and $\mathrm{LAST} = |r|$. Next, **Compute-ICFL** recursively calls itself on x_2y (line (5)) and returns $list = \mathrm{ICFL}(x_2y) = \mathrm{ICFL}(\bar{p}y)$. Let $z = m'_1$ be the first element of *list* (line (6)). According to Definition 4.3, we have to test whether $x_2 = \overline{p} = rb \leq_p m'_1 = z$, that is if $|z| > |r| = \mathrm{LAST}$. This is done on line (7). If $|z| > |r| = \mathrm{LAST}$, then we add $x_1 = p$ at

the first position of *list* (line (8)), otherwise we replace $z = m'_1$ in *list* with $x_1 z = pm'_1$ (line (10)). In both cases, **Compute-ICFL** returns *list* = ICFL(w).

It is worth of noting that there is no preprocessing of w for computing the failure function used by **Find-bre**(x, y). Each call to **Find-bre**(x, y) calls **Border**(x'), where (x, y) is the output of **Find-prefix**(w) and x' is the prefix of x of length $|x| - 1$.

Algorithm 3: Compute-ICFL.

```
   Input  : A word w.
   Output: ICFL(w)
 1 (x, y) ← Find-prefix(w);
 2 if  x = w$ then
 3 |   return (w);
 4 (x_1, x_2, y, LAST) ← Find-bre(x, y);
 5 list ← Compute-ICFL(x_2 y);
 6 z ← the first element of list;
 7 if |z| > LAST then
 8 |   add x_1 in front of the list;
 9 else
10 |   replace z in list with x_1 z;
11 return list;
```

Example 6.1. Let $\Sigma = \{a, b, c\}$ with $a < b < c$, let $w = cbabacaacbabacbac$, already considered in Example 4.5. Suppose that we call **Compute-ICFL** on w and on the empty list *list*. The three tables below illustrate the sequence of calls made to **Compute-ICFL**, **Find-prefix** and **Find-bre** if we read the first column downward. For instance, since **Find-prefix**$(cbac)$ returns $(cbac\$, 1)$, **Compute-ICFL**$(cbac)$ returns $(cbac)$ without invoking itself again and the recursion stops. **Compute-ICFL**$(cbabacbac)$ calls **Find-prefix** on $cbabacbac$, **Find-bre** on $(cbabacbac, 1)$ and then **Compute-ICFL**$(cbac)$ which returns $(cbac)$. Since $|z| = |cbac| = 4 >$ LAST $= 3$, **Compute-ICFL**$(cbabacbac)$ returns $(cbaba, cbac)$. Finally, **Compute-ICFL**(w) calls **Find-prefix** on w, **Find-bre** on $(cbabacaacbabacb, ac)$ and then **Compute-ICFL**$(cbabacbac)$ which returns $(cbaba, cbac)$. Since $|z| = |cbaba| = 5 \leq$ LAST $= 6$, **Compute-ICFL**(w) replaces $cbaba$ with the concatenation of $cbabacaa$ and $cbaba$ and returns *list* = $(cbabacaacbaba, cbac)$ = ICFL(w).

CALL	RETURN
Compute-ICFL$(cbabacaacbabacbac)$	$(cbabacaacbaba, cbac)$
Compute-ICFL$(cbabacbac)$	$(cbaba, cbac)$
Compute-ICFL$(cbac)$	$(cbac)$

CALL	RETURN
Find-prefix$(cbabacaacbabacbac)$	$(cbabacaacbabacb, ac)$
Find-prefix$(cbabacbac)$	$(cbabacbac, 1)$
Find-prefix$(cbac)$	$(cbac\$, 1)$

CALL	RETURN
Find-bre$(cbabacaacbabacb, ac)$	$(cbabacaa, cbabacb, ac, 6)$
Find-bre$(cbabacbac, 1)$	$(cbaba, cbac, 1, 3)$

6.1. Performance of **Compute-ICFL**

Let us compute the running time $T(n)$ of **Compute-ICFL** when w has length n. We can check that the running time of **Find-prefix**(w) is $\mathcal{O}(|x|)$, when the output of the procedure is (x, y). Then, **Compute-ICFL** calls the procedure **Find-bre**(x, y), where $x = p\overline{p}$ and $(p, \overline{p}) \in \mathrm{Pref}_{bre}(w)$. By Lemma 4.1, we know that $|\overline{p}| \leq |p|$, and so the running time of **Find-bre** is $\mathcal{O}(|p| + |\overline{p}|) = \mathcal{O}(|p|)$. Let $\mathrm{ICFL}(w) = (m_1, \ldots, m_k)$ and let n_j be the length of m_j, $1 \leq j \leq k$. The recurrence for $T(n)$ is defined as $T(n) = T(n-n_1)+\mathcal{O}(n_1)$, where $T(n_k)$ is $\mathcal{O}(n_k)$, because there is no recursive call in this case. It is easy to see that the solution to this recurrence is $T(n) = \sum_{j=1}^{k} \mathcal{O}(n_j) = \mathcal{O}(n)$, since $w = m_1 \cdots m_k$.

7. Groupings

Let $(\Sigma, <)$ be a totally ordered alphabet. As we know, for any word $w \in \Sigma^+$, there are three sequences of words associated with w: the Lyndon factorization of w with respect to the order $\prec$, denoted $\mathrm{CFL}(w)$, the Lyndon factorization of w with respect to the inverse lexicographic order $\prec_{in}$, denoted $\mathrm{CFL}_{in}(w)$ and the inverse Lyndon factorization $\mathrm{ICFL}(w)$ of w. In this section, we compare $\mathrm{CFL}_{in}(w)$ and $\mathrm{ICFL}(w)$.

Precisely, starting with $\mathrm{CFL}_{in}(w)$, we define a family of inverse Lyndon factorizations of w, called *groupings* of $\mathrm{CFL}_{in}(w)$ (Section 7.1). We prove that $\mathrm{ICFL}(w)$ is a grouping of $\mathrm{CFL}_{in}(w)$ in Section 7.2. We also prove that Theorem 2.3 may be generalized to groupings when we refer to the sorting with respect to the inverse lexicographic order (Section 7.3).

7.1. A family of inverse Lyndon factorizations of words

Groupings of $\mathrm{CFL}_{in}(w)$ are special inverse Lyndon factorizations. They are constructed in a very natural way. We first give some needed definitions and results.

Definition 7.1. Let $w \in \Sigma^+$, let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$ and let $1 \leq r < s \leq h$. We say that $\ell_r, \ell_{r+1} \ldots, \ell_s$ is a non-increasing maximal chain for the prefix order in $\mathrm{CFL}_{in}(w)$, abbreviated $\mathcal{PMC}$, if $\ell_r \geq_p \ell_{r+1} \geq_p \ldots \geq_p \ell_s$. Moreover, if $r > 1$, then $\ell_{r-1} \not\geq_p \ell_r$, if $s < h$, then $\ell_s \not\geq_p \ell_{s+1}$. Two $\mathcal{PMC}$ $\mathcal{C}_1 = \ell_r, \ell_{r+1} \ldots, \ell_s$, $\mathcal{C}_2 = \ell_{r'}, \ell_{r'+1} \ldots, \ell_{s'}$ are consecutive if $r' = s + 1$ (or $r = s' + 1$).

Lemma 7.1. *Let x, y be nonempty words such that $x \succeq_{in} y$. Then either $x \geq_p y$ or $x \ll y$.*

Proof:
Let x, y be nonempty words such that $x \succeq_{in} y$. Therefore x is not a proper prefix of y. If y is not a prefix of x, then $x \bowtie y$ and $y \prec_{in} x$. Hence, by Proposition 2.5, we have $x \ll y$. ■

The following is a direct consequence of Lemma 7.1.

Proposition 7.1. *Let $w \in \Sigma^+$, let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$. Then*

$$(\ell_1, \ldots, \ell_h) = (\mathcal{C}_1, \ldots, \mathcal{C}_t),$$

where any $\mathcal{C}_j$, $1 \leq j \leq t$, is a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$. Moreover, $\mathcal{C}_j$, $\mathcal{C}_{j+1}$ are consecutive and $\ell \ll \ell'$, where ℓ is the last word in $\mathcal{C}_j$ and ℓ' is the first word in $\mathcal{C}_{j+1}$, for $1 \leq j \leq t-1$.

Proof:
Let $w \in \Sigma^+$ and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$, i.e.,

$$w = \ell_1 \cdots \ell_h, \text{ with } \ell_j \in L_{in} \text{ and } \ell_1 \succeq_{in} \ldots \succeq_{in} \ell_h \tag{7.1}$$

By Lemma 7.1, each symbol $\succeq_{in}$ in Eq. (7.1) may be replaced either by $\geq_p$ or by $\ll$. Therefore, the conclusion follows. ■

The definition of a grouping of $\mathrm{CFL}_{in}(w)$ is given below in two steps. We first define the grouping of a $\mathcal{PMC}$. Then a grouping of $\mathrm{CFL}_{in}(w)$ is obtained by changing each $\mathcal{PMC}$ with one of its groupings.

Definition 7.2. Let $\ell_1, \ldots, \ell_h$ be words in L_{in} such that ℓ_i is a prefix of ℓ_{i-1}, $1 < i \leq h$. We say that $(m_1, \ldots, m_k)$ is a grouping of $(\ell_1, \ldots, \ell_h)$ if the following conditions are satisfied.

(1) m_j is an inverse Lyndon word,
(2) $\ell_1 \cdots \ell_h = m_1 \cdots m_k$. More precisely, there are $i_0, i_1, \ldots, i_k$, $i_0 = 0$, $1 \leq i_j \leq h$, $i_k = h$, such that $m_j = \ell_{i_{j-1}+1} \cdots \ell_{i_j}$, $1 \leq j \leq k$,
(3) $m_1 \ll \ldots \ll m_k$.

We now extend Definition 7.2 to $\mathrm{CFL}_{in}(w)$.

Definition 7.3. Let $w \in \Sigma^+$ and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$. We say that $(m_1, \ldots, m_k)$ is a grouping of $\mathrm{CFL}_{in}(w)$ if it can be obtained by replacing any $\mathcal{PMC}$ $\mathcal{C}$ in $\mathrm{CFL}_{in}(w)$ by a grouping of $\mathcal{C}$.

Proposition 7.2 shows that groupings of $\mathrm{CFL}_{in}(w)$ are inverse Lyndon factorizations of w. However, as Example 7.1 shows, there are inverse Lyndon factorizations which are not groupings.

Example 7.1. Let $w = dabadabdabdadac \in \{a, b, c, d\}^+$ with $a < b < c < d$. Thus, $d <_{in} c <_{in} b <_{in} a$ and $\mathrm{CFL}_{in}(w) = (daba, dab, dab, dadac)$. The two sequences $(daba, dabdab, dadac)$, $(dabadab, dabda, dac)$ are both inverse Lyndon factorizations of w and $\mathrm{ICFL}(w) = (daba, dabdab, dadac)$ (Examples 4.1, 4.6). However, $\mathrm{ICFL}(w)$ is a grouping of $\mathrm{CFL}_{in}(w)$, whereas $(dabadab, dabda, dac)$ is not a grouping of $\mathrm{CFL}_{in}(w)$.

In Example 4.1 we also considered the following two inverse Lyndon factorizations of $z = dabdadacddbdc$

$$(dab)(dadacd)(db)(dc) = (dabda)(dac)(ddbdc)$$

Both of them are not groupings since $\mathrm{CFL}_{in}(z) = (dab, dadac, ddbdc)$. Notice that $\mathrm{CFL}_{in}(z) = \mathrm{ICFL}(z)$ (see Corollary 7.1).

Proposition 7.2. *Let $w \in \Sigma^+$. If $(m_1, \ldots, m_k)$ is a grouping of $\mathrm{CFL}_{in}(w)$, then $(m_1, \ldots, m_k)$ is an inverse Lyndon factorization of w.*

Proof:
Let $w \in \Sigma^+$ and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$. Therefore, by Proposition 7.1, we have

$$(\ell_1, \ldots, \ell_h) = (\mathcal{C}_1, \ldots, \mathcal{C}_t),$$

where any $\mathcal{C}_j$, $1 \leq j \leq t$, is a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$. Moreover, $\mathcal{C}_j$, $\mathcal{C}_{j+1}$ are consecutive and $\ell \ll \ell'$, where ℓ is the last word in $\mathcal{C}_j$ and ℓ' is the first word in $\mathcal{C}_{j+1}$, for $1 \leq j \leq t-1$.

Now let $(m_1, \ldots, m_k)$ be a grouping of $\mathrm{CFL}_{in}(w)$. Any m_j is an inverse Lyndon word since it is an element of a grouping of a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$. Then, let $\mathcal{S}_j$ be the product of the words in $\mathcal{C}_j$, $1 \leq j \leq t$. It is clear that

$$w = \ell_1 \cdots \ell_h = \mathcal{S}_1 \cdots \mathcal{S}_t = m_1 \cdots m_k.$$

Finally, set $\mathcal{C}_j = (\ell_i, \ldots, \ell_g)$ and $\mathcal{C}_{j+1} = (\ell_{g+1}, \ldots, \ell_f)$, for $1 \leq j \leq t-1$. Thus $\ell_g \ll \ell_{g+1}$. It suffices to show that if $(m_r, \ldots, m_s)$ is the grouping of $\mathcal{C}_j$ that replaces $\mathcal{C}_j$ and $(m_{s+1}, \ldots, m_v)$ is the grouping of $\mathcal{C}_{j+1}$ that replaces $\mathcal{C}_{j+1}$, then $m_s \ll m_{s+1}$. But this is clear since ℓ_g is a suffix of m_s, hence it is also a prefix of m_s ($\mathcal{C}_j$ is a $\mathcal{PMC}$) and ℓ_{g+1} is a prefix of m_{s+1}. Therefore, by item (2) in Lemma 2.2, $\ell_g \ll \ell_{g+1}$ implies $m_s \ll m_{s+1}$. ■

7.2. $\mathrm{ICFL}(w)$ *is a grouping of* $\mathrm{CFL}_{in}(w)$

Let us outline how we prove below that $\mathrm{ICFL}(w)$ is a grouping of $\mathrm{CFL}_{in}(w)$. Let $\mathrm{ICFL}(w) = (m_1, \ldots, m_k)$, let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$ and let $\ell_1, \ldots, \ell_q$ be a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$, $1 \leq q \leq k$. The proof will be divided into four steps.

(1) We prove that m_1 cannot be a proper prefix of ℓ_1 (Proposition 7.4).
(2) We prove that if m_1 is a prefix of $\ell_1 \cdots \ell_q$, then $m_1 = \ell_1 \cdots \ell_{q'}$, for some q', $1 \leq q' \leq q$ (Proposition 7.5).
(3) We prove that $\ell_1 \cdots \ell_q$ cannot be a proper prefix of m_1 (Proposition 7.6).
(4) We complete the proof by induction on $|w|$ (Proposition 7.7).

We say that a sequence of nonempty words $(m_1, \ldots, m_k)$ is a *factorization* of w if $w = m_1 \cdots m_k$. It is worth of noting that steps (1) and (2) are proved under the more general hypothesis that $(m_1, \ldots, m_k)$ is a factorization of w such that $m_1 \ll m_2$ and, for step (2), where m_1 is an inverse Lyndon word. Proposition 7.3 and Corollary 7.1 deal with two extremal cases where there is only one grouping of $\mathrm{CFL}_{in}(w)$, namely $\mathrm{ICFL}(w)$.

Proposition 7.3. *Let $(\Sigma, <)$ be a totally ordered alphabet. Let $w \in \Sigma^+$ and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$. If w is an inverse Lyndon word, then either w is unbordered or $\ell_1, \ldots, \ell_h$ is a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$. In both cases $\mathrm{ICFL}(w) = (w)$ is the unique grouping of $\mathrm{CFL}_{in}(w)$.*

Proof:
Let $w \in \Sigma^+$ be an inverse Lyndon word and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$. We know that $\mathrm{ICFL}(w) = (w)$ (Definition 4.3). By Proposition 2.7, if w is unbordered, then w is an anti-Lyndon word. Thus, by item (iii) in Lemma 2.3, $\mathrm{CFL}_{in}(w) = (w)$ and of course this is the unique grouping of $\mathrm{CFL}_{in}(w)$.

Otherwise, w is bordered and, again by Proposition 2.7, $h > 1$. By contradiction assume that $\ell_1, \ldots, \ell_h$ is not a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$. By Lemma 7.1, there would be a smallest q, $1 \leq q \leq h-1$ such that

$$\ell_1 \geq_p \ldots \geq_p \ell_q \ll \ell_{q+1}$$

Hence, since ℓ_q is a prefix of w, by item (2) in Lemma 2.2, we would have $w \ll \ell_{q+1} \cdots \ell_h$, which is a contradiction since w is an inverse Lyndon word and $\ell_{q+1} \cdots \ell_h$ is a proper nonempty suffix of w. Thus, $\ell_1, \ldots, \ell_h$ is a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$ and $\mathrm{ICFL}(w) = (w)$ is a grouping of $\mathrm{CFL}_{in}(w)$. By contradiction, assume that $(m_1, \ldots, m_k)$, $k \geq 2$, is another grouping of $\mathrm{CFL}_{in}(w)$. Therefore, $m_1 \ll m_2$ and by item (2) in Lemma 2.2, we would have $w \ll m_2 \cdots m_k$, which is a contradiction since w is an inverse Lyndon word and $m_2 \cdots m_k$ is a proper nonempty suffix of w. ∎

Proposition 7.4. *Let $(\Sigma, <)$ be a totally ordered alphabet. Let $w \in \Sigma^+$ and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$. For any factorization $(m_1, \ldots, m_k)$ of w, with $k > 1$, if $m_1 \ll m_2$, then m_1 cannot be a proper prefix of ℓ_1.*

Proof:
Let $(m_1, \ldots, m_k)$ be a factorization of w such that $m_1 \ll m_2$. By contradiction assume that m_1 is a proper prefix of ℓ_1. Therefore, there are two nonempty words x, y such that $m_1 = x$, $\ell_1 = xy$, and $v \in \Sigma^*$ such that $m_2 \cdots m_k = yv$. On one hand, by Proposition 2.8 we have $y \ll \ell_1$. On the other hand, $x = m_1 \ll m_2$, thus $x = m_1 \ll m_2 \cdots m_k = yv$, by item (2) in Lemma 2.2. Consequently, $x = ras$, $yv = rbt$, with $a < b$. Hence either y is a prefix of r, which is a contradiction since $\ell_1 = xy = rasy$ is unbordered (Proposition 2.7), or rb is a prefix of y, which is once again a contradiction since we would have $x \ll y$ and consequently, by item (2) in Lemma 2.2, $\ell_1 \ll y$. ∎

Proposition 7.5. *Let $(\Sigma, <)$ be a totally ordered alphabet. Let $w \in \Sigma^+$ and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$. Let $(m_1, \ldots, m_k)$ be a factorization of w such that $k > 1$, m_1 is an inverse Lyndon word and $m_1 \ll m_2$. Let $\ell_1, \ldots, \ell_q$ be a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$, $1 \leq q \leq h$. If m_1 is a prefix of $\ell_1 \cdots \ell_q$, then $m_1 = \ell_1 \cdots \ell_{q'}$, for some q', $1 \leq q' \leq q$.*

Proof:
Let $w \in \Sigma^+$ and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$. Let $(m_1, \ldots, m_k)$ be a factorization of w such that $k > 1$, m_1 is an inverse Lyndon word and $m_1 \ll m_2$. Let $\ell_1, \ldots, \ell_q$ be a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$, $1 \leq q \leq h$.

By contradiction, assume that there are two nonempty words x, y such that $m_1 = \ell_1 \cdots \ell_{j-1} x$, $xy = \ell_j$, $1 \leq j \leq q$ (where it is understood that $m_1 = x$ when $j = 1$). By Proposition 7.4, we have $j > 1$. Moreover there is $v \in \Sigma^*$ such that $m_2 \cdots m_k = yv$.

Since $m_1 \ll m_2$, we have $m_1 = \ell_1 \cdots \ell_{j-1} x \ll m_2 \cdots m_k = yv$. Hence, there are words $r, s, t \in \Sigma^*$ and letters $a, b \in \Sigma$, with $a < b$ such that $m_1 = ras$, $m_2 \cdots m_k = yv = rbt$. If $|r| < |y|$, then rb is a prefix of y, therefore it is a factor of ℓ_j. In turn, ℓ_j is a prefix of ℓ_{j-1}, thus there is $\gamma \in \Sigma^*$ such that $rb\gamma$ is a proper nonempty suffix of m_1. Then, $m_1 = ras \ll rb\gamma$, a contradiction, since m_1 is an inverse Lyndon word. Hence $|r| \geq |y|$, i.e., y is a prefix of r and consequently it is a prefix of m_1. The word $\ell_j = xy$ is also a prefix of m_1, thus y and $\ell_j = xy$ are comparable for the prefix order. Therefore y is both a nonempty proper prefix and a suffix of ℓ_j, i.e., $\ell_j = xy$ is bordered, a contradiction since $\ell_j \in L_{in}$ (see Proposition 2.7). ■

Proposition 7.6. *Let $(\Sigma, <)$ be a totally ordered alphabet. Let $w \in \Sigma^+$, let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$ and let $\mathrm{ICFL}(w) = (m_1, \ldots, m_k)$. Let $\ell_1, \ldots, \ell_q$ be a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$, $1 \leq q \leq h$. Then $m_1 = \ell_1 \cdots \ell_{q'}$, for some q', $1 \leq q' \leq q$.*

Proof:
Let $w \in \Sigma^+$, let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$, and let $\mathrm{ICFL}(w) = (m_1, \ldots, m_k)$. We prove the statement by induction on $|w|$. If $|w| = 1$, then w is an inverse Lyndon word and we are done, by Proposition 7.3. Hence assume $|w| > 1$. If w is an inverse Lyndon word, then the proof is ended, once again by Proposition 7.3. Therefore, assume that w is not an inverse Lyndon word.

Let $\ell_1, \ldots, \ell_q$ be a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$, $1 \leq q \leq h$. Since m_1 and $\ell_1 \cdots \ell_q$ are both prefixes of w, m_1 and $\ell_1 \cdots \ell_q$ are comparable for the prefix order. By contradiction, assume that m_1 violates the statement. Therefore, by Propositions 7.4 and 7.5, m_1 is not a prefix of $\ell_1 \cdots \ell_q$. Hence, $\ell_1 \cdots \ell_q$ is a proper prefix of m_1, i.e., there are two words x, y and j, with $q < j \leq h$, such that $m_1 = \ell_1 \cdots \ell_{j-1} x$, $xy = \ell_j$. Moreover, $\ell_q \ll \ell_{q+1}$ and, if $j - 1 = q$, then $x \neq 1$. We must have $j - 1 = q$ (thus $x \neq 1$ also) and $y \neq 1$. Indeed, otherwise $j - 1 > q$ or $x = \ell_j$, $j \geq q + 1$. In both cases, $\ell_{q+1} \cdots \ell_{j-1} x$ would be a proper nonempty suffix of m_1 and ℓ_q is a prefix of ℓ_1, hence ℓ_q is a prefix of m_1. By item (2) in Lemma 2.2 applied to $\ell_q \ll \ell_{q+1}$, we would have $m_1 \ll \ell_{q+1} \cdots \ell_{j-1} x$, a

contradiction since m_1 is an inverse Lyndon word. In conclusion, there are words x, y, y', with $x \neq 1$ and $y \neq 1$, such that

$$m_1 = \ell_1 \cdots \ell_q x,\ \ell_1 \geq_p \ldots \geq_p \ell_q \ll \ell_{q+1} = xy,\ m_2 \cdots m_k = yy' \tag{7.2}$$

Let $(p, \bar{p}) \in \mathrm{Pref}_{bre}(w)$. Let $v \in \Sigma^+$ be such that $w = pv$ and let $\mathrm{ICFL}(v) = (m'_1, \ldots, m'_{k'})$. By Definition 4.3, one of the following two cases holds

(1) $m_1 = p$
(2) $m_1 = pm'_1$.

In both cases, p is a prefix of $\ell_1 \cdots \ell_q x$, therefore $|p| \leq |\ell_1 \cdots \ell_q x|$. We claim that we also have $|p| > |\ell_1 \cdots \ell_q|$. By Eq. (7.2), this is clearly true in case (1), hence assume $m_1 = pm'_1$. Since $p \ll \bar{p}$, by item (2) in Lemma 2.2, we have $p \ll v$. Thus, Propositions 7.4 and 7.5 apply to the factorization (p, v) of w. Therefore, if $|p| \leq |\ell_1 \cdots \ell_q|$, then $p = \ell_1 \cdots \ell_j$, with $j \leq q$. Hence, $v = \ell_{j+1} \cdots \ell_h$ and, by Theorem 2.2, $\mathrm{CFL}_{in}(v) = (\ell_{j+1}, \ldots, \ell_h)$. On the other hand, by Eq. (7.2), we would have $m'_1 = \ell_{j+1} \cdots \ell_q x$, $\ell_{q+1} = xy$, $\ell_q \ll \ell_{q+1}$, and $x \neq 1$, in contradiction with induction hypothesis applied to v. In conclusion, in both cases (1) and (2), we have $|\ell_1 \cdots \ell_q| < |p| \leq |\ell_1 \cdots \ell_q x|$, thus there are words x_1, x_2 such that

$$p = \ell_1 \cdots \ell_q x_1,\ x = x_1 x_2,\ x_1 \neq 1,\ x_2 = \begin{cases} 1 & \text{if } m_1 = p, \\ m'_1 & \text{if } m_1 = pm'_1 \end{cases} \tag{7.3}$$

Then, by Definitions 4.2 and 4.3, there are words $r, s, t \in \Sigma^*$ and letters $a, b \in \Sigma$, with $a < b$ such that

$$p = ras, \quad \bar{p} = rb, \quad v = x_2 m_2 \cdots m_k = rbt \tag{7.4}$$

Since $\ell_q \ll \ell_{q+1}$, there are words $z, f, g \in \Sigma^*$ and letters $c, d \in \Sigma$, with $c < d$ such that

$$\ell_q = zcf, \quad \ell_{q+1} = xy = zdg \tag{7.5}$$

Observe that x and zd are both prefixes of ℓ_{q+1}, hence they are comparable for the prefix order. If zd would be a prefix of x, then for a word γ, $zd\gamma$ would be a proper nonempty suffix of m_1 (see Eq. (7.2)) and ℓ_q would be a prefix of ℓ_1, thus of m_1 such that $\ell_q \ll zd\gamma$ (see Eq. (7.5)). By item (2) in Lemma 2.2 applied to $\ell_q \ll zd\gamma$, we would have $m_1 \ll zd\gamma$, a contradiction since m_1 is an inverse Lyndon word. Therefore, x is a proper prefix of zd, i.e., there is $z' \in \Sigma^*$ such that

$$z = xz' = x_1 x_2 z'. \tag{7.6}$$

Eqs. (7.5) and (7.6) yield $xy = zdg = xz'dg$, therefore z' is a prefix of y and thus $x_2 z'$ is a prefix of $x_2 m_2 \cdots m_k$ (see Eq. (7.2)). On the other hand rb is also a prefix of $x_2 m_2 \cdots m_k$

(see Eq. (7.4)). Hence, rb and x_2z' are comparable for the prefix order and one of the following two cases is satisfied

(i) rb is a prefix of x_2z',
(ii) x_2z' is a prefix of r.

Assume that case (i) holds. In this case, since x_2z' is a suffix of z (Eq. (7.6)) and z is a prefix of ℓ_q (Eq. (7.5)), the word rb would be a factor of ℓ_q. Thus, by Eq. (7.2), there would be $\gamma \in \Sigma^*$ such that $rb\gamma$ would be a proper nonempty suffix of m_1. Since $ras = p$ (Eq. (7.4)), the word ras would be a prefix of m_1 and we would have $m_1 \ll rb\gamma$, a contradiction since m_1 is an inverse Lyndon word. Case (ii) also leads to a contradiction. Indeed, set $p' = \ell_1 \cdots \ell_q$. Then, by Eqs. (7.3) and (7.6), $px_2z' = p'x_1x_2z' = p'z$. If x_2z' is a prefix of r, then $px_2z' = p'z$ is a prefix of pr, thus $px_2z' = p'z$ is a proper prefix of $p\bar{p}$ (Eq. (7.4)), hence $px_2z' = p'z$ is an inverse Lyndon word, as do all nonempty prefixes of $p'z$, by Definition 4.2. Moreover, by Eq. (7.5), zc is a prefix of ℓ_q, thus of p'. Therefore, we have $p' = zcf'$, for a word f', and $p'zd$ is not an inverse Lyndon word since $p'zd \ll zd$. On the contrary, zd is an inverse Lyndon word, because it is a prefix of ℓ_{q+1} (Eq. 7.5). Finally, $p' = zcf' \ll zd$. By Definition 4.2, the pair $(p', zd) \in \mathrm{Pref}_{bre}(w)$. Since $x_1 \neq 1$, we have $p' \neq p = p'x_1$, in contradiction with Proposition 4.3. ■

Proposition 7.7. *Let $(\Sigma, <)$ be a totally ordered alphabet. For any $w \in \Sigma^+$, $\mathrm{ICFL}(w)$ is a grouping of $\mathrm{CFL}_{in}(w)$.*

Proof:
Let $w \in \Sigma^+$, let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$, and let $\mathrm{ICFL}(w) = (m_1, \ldots, m_k)$. The proof is by induction on $|w|$. If $|w| = 1$, then w is an inverse Lyndon word and we are done, by Proposition 7.3. Hence assume $|w| > 1$.

If w is an inverse Lyndon word, once again by Proposition 7.3, $\mathrm{ICFL}(w) = (w) = (\ell_1 \cdots \ell_h)$ is a grouping of $\mathrm{CFL}_{in}(w)$. Therefore, assume that w is not an inverse Lyndon word. By Propositions 7.4-7.6, there is j, $1 \leq j \leq h$ such that $m_1 = \ell_1 \cdots \ell_j$, where $\ell_1 \geq_p \ldots \geq_p \ell_j$.

Let $(p, \bar{p}) \in \mathrm{Pref}(w)_{bre}$. Let $\mathrm{ICFL}(v) = (m'_1, \ldots, m'_{k'})$, where $v \in \Sigma^+$ is such that $w = pv$. By induction hypothesis, $(m'_1, \ldots, m'_{k'})$ is a grouping of $\mathrm{CFL}_{in}(v)$. By Definition 4.3, one of the following two cases holds

(1) $m_1 = p$, $(m_2, \ldots, m_k) = (m'_1, \ldots, m'_{k'})$, i.e., $k = k' + 1$, $m_j = m'_{j-1}$, $2 \leq j \leq k$.
(2) $m_1 = pm'_1$, $(m_2, \ldots, m_k) = (m'_2, \ldots, m'_{k'})$, i.e., $k = k'$, $m_j = m'_j$, $2 \leq j \leq k$.

(Case (1)). In this case, since $p = m_1 = \ell_1 \cdots \ell_j$, we have $v = \ell_{j+1} \cdots \ell_h$, where any ℓ_g is in L_{in} and $\ell_{j+1} \succeq_{in} \ldots \succeq_{in} \ell_h$. By Theorem 2.2, $\mathrm{CFL}_{in}(v) = (\ell_{j+1}, \ldots, \ell_h)$. Consequently, if $(\ell_{j+1}, \ldots, \ell_h) = (\mathcal{C}_1, \ldots, \mathcal{C}_t)$, where any $\mathcal{C}_j$, $1 \leq j \leq t$, is a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(v)$, then $(\ell_1, \ldots, \ell_h) = (\mathcal{C}, \mathcal{C}_1, \ldots, \mathcal{C}_t)$ where $\mathcal{C} = \ell_1, \ldots, \ell_j$, any $\mathcal{C}_j$, $2 \leq j \leq$

t, is a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$ and either $\mathcal{C}, \mathcal{C}_1$ represents two $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$ or it represents a single $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$. In both cases, since $m_1 = \ell_1 \cdots \ell_j$ and given that $(m_2, \ldots, m_k) = (m'_1, \ldots, m'_{k'})$ is a grouping of $\mathrm{CFL}_{in}(v)$, we conclude that $\mathrm{ICFL}(w)$ is a grouping of $\mathrm{CFL}_{in}(w)$.

(Case (2)). Set $\mathrm{CFL}_{in}(v) = (\ell'_1, \ldots, \ell'_{h'})$, and let j' be such that $m'_1 = \ell'_1 \cdots \ell'_{j'}$, $\ell'_1 \geq_p \ldots \geq_p \ell'_{j'}$. Then we have

$$(pm'_1)(m_2 \cdots m_k) = (\ell_1 \cdots \ell_j)(\ell_{j+1} \cdots \ell_h) = (p\ell'_1 \cdots \ell'_{j'})(\ell'_{j'+1} \cdots \ell'_{h'}),$$

which, along with $(pm'_1) = (\ell_1 \cdots \ell_j) = (p\ell'_1 \cdots \ell'_{j'})$, yields

$$(m_2 \cdots m_k) = (\ell_{j+1} \cdots \ell_h) = (\ell'_{j'+1} \cdots \ell'_{h'}) \tag{7.7}$$

All the words $\ell_g, \ell'_{g'}$ are in L_{in} and $\ell_{j+1} \succeq_{in} \ldots \succeq_{in} \ell_h$, $\ell'_{j'+1} \succeq_{in} \ldots \succeq_{in} \ell'_{h'}$. Hence, by Theorem 2.2, applied to $v' = m_2 \cdots m_k = m'_2 \cdots m'_{k'}$, Eq. (7.7) implies $(\ell_{j+1}, \ldots, \ell_h) = (\ell'_{j'+1}, \ldots, \ell'_{h'})$ and $\mathrm{CFL}_{in}(v') = (\ell_{j+1}, \ldots, \ell_h)$. Now, $(m'_1, \ldots, m'_{k'})$ is a grouping of $\mathrm{CFL}_{in}(v)$, where $v = m'_1 v'$ and $m'_1 = \ell'_1 \cdots \ell'_{j'}$. Thus, $(m_2, \ldots, m_h) = (m'_2, \ldots, m'_{k'})$ is a grouping of $(\ell_{j+1}, \ldots, \ell_h) = (\ell'_{j'+1}, \ldots, \ell'_{h'}) = \mathrm{CFL}_{in}(v')$. The rest of the proof runs as in Case (1). Namely, if $(\ell_{j+1}, \ldots, \ell_h) = (\mathcal{C}_1, \ldots, \mathcal{C}_t)$, where any $\mathcal{C}_j$, $1 \leq j \leq t$, is a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(v')$, then $(\ell_1, \ldots, \ell_h) = (\mathcal{C}, \mathcal{C}_1, \ldots, \mathcal{C}_t)$ where $\mathcal{C} = \ell_1, \ldots, \ell_j$, any $\mathcal{C}_j$, $2 \leq j \leq t$, is a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$ and either $\mathcal{C}, \mathcal{C}_1$ represents two $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$ or it represents a single $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$. In both cases, since $m_1 = \ell_1 \cdots \ell_j$ and given that $(m_2, \ldots, m_k) = (m'_2, \ldots, m'_{k'})$ is a grouping of $\mathrm{CFL}_{in}(v')$, we conclude that $\mathrm{ICFL}(w)$ is a grouping of $\mathrm{CFL}_{in}(w)$. ■

Corollary 7.1. *Let $(\Sigma, <)$ be a totally ordered alphabet. Let $w \in \Sigma^+$ and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$, with $h > 1$. If $\ell_1 \ll \ldots \ll \ell_h$, then $\mathrm{ICFL}(w) = \mathrm{CFL}_{in}(w)$ and this is the unique grouping of $\mathrm{CFL}_{in}(w)$.*

Proof:
Let $w \in \Sigma^+$ and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$, with $h > 1$. If $\ell_1 \ll \ldots \ll \ell_h$, then $\mathrm{CFL}_{in}(w)$ is the unique grouping of $\mathrm{CFL}_{in}(w)$ (Definition 7.3). Thus, by Proposition 7.7, $\mathrm{ICFL}(w) = \mathrm{CFL}_{in}(w)$. ■

The following example shows that there are words w such that $\mathrm{CFL}_{in}(w)$ has more than one grouping, thus there are groupings of $\mathrm{CFL}_{in}(w)$ different from $\mathrm{ICFL}(w)$.

Example 7.2. Let $w = dabadabdabdabdadac \in \{a, b, c, d\}^+$ with $a < b < c < d$. Therefore, $d <_{in} c <_{in} b <_{in} a$ and $\mathrm{CFL}_{in}(w) = (daba, dab, dab, dab, dadac)$. The two sequences $(dabadab, (dab)^2, dadac)$, $(daba, (dab)^3, dadac)$ are both groupings of $\mathrm{CFL}_{in}(w)$. Let us compute $\mathrm{ICFL}(w)$. The shortest prefix of w which is not an inverse Lyndon word is $dabadabd$ and $(daba, dabd) \in \mathrm{Pref}_{bre}(w)$ by Lemma 4.1. Set $w = dabay$, we

have to compute ICFL(y). The shortest prefix of $y = (dab)^3 dadac$ which is not an inverse Lyndon word is $(dab)^3 dad$ and $((dab)^3, dad) \in \mathrm{Pref}_{bre}(y)$ by Lemma 4.1. On the other hand, since $dadac$ is an inverse Lyndon word, we have $\mathrm{ICFL}(dadac) = (dadac)$ and thus $\mathrm{ICFL}(y) = ((dab)^3, dadac)$ (see Definition 4.3). Finally, by Definition 4.3, $\mathrm{ICFL}(w) = (daba, (dab)^3, dadac)$.

We do not know whether there exists an efficient algorithm to compute all groupings of $\mathrm{CFL}_{in}(w)$. Proposition 7.3 and Corollary 7.1 answer this question in two particular cases. The following compositional properties of the inverse Lyndon words, which can be stated as a direct consequence of Proposition 3.1, also deal with this open problem.

Proposition 7.8. *For any $w \in L_{in}$ and $h \geq 1$, the word w^h is an inverse Lyndon word on $(\Sigma^*, <)$.*

Proposition 7.9. *Let $\ell_1, \ell_2 \in L_{in}$. If ℓ_2 is a proper prefix of ℓ_1, then $\ell_1\ell_2$ is an inverse Lyndon word.*

Example 7.3. Let $\ell_1, \ldots, \ell_h$ be words in L_{in} which form a non-increasing chain $\ell_1 \geq_p \ldots \geq_p \ell_h$ for the prefix order, i.e., ℓ_i is a prefix of ℓ_{i-1}, $1 < i \leq h$. The word $\ell_1 \cdots \ell_h$ is not necessarily an inverse Lyndon word. As an example, let $\Sigma = \{a, b\}$ with $a < b$. The sequence baa, ba, b is such that $baa \geq_p ba \geq_p b$. The word $baabab$ is not an inverse Lyndon word since $baabab \ll bab$.

We end the section with Proposition 7.10. It shows that nonempty words which are both proper prefix and suffix of an inverse Lyndon word have a special form.

Proposition 7.10. *Let $(\Sigma, <)$ be a totally ordered alphabet. Let $w \in \Sigma^+$ be an inverse Lyndon word and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$. For any proper nonempty suffix u of w such that u is also a prefix of w, there is j_u such that $u = \ell_{j_u} \cdots \ell_h$, $2 \leq j_u \leq h$.*

Proof:
Let $(\Sigma, <)$ be a totally ordered alphabet. Let $w \in \Sigma^+$ be a bordered inverse Lyndon word and let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$. Let $u \in \Sigma^+$ be such that u is a proper nonempty suffix of w and u is also a prefix of w. By contradiction, assume that $u = s\ell_{j_u} \cdots \ell_h$, $2 \leq j_u \leq h$, where s is a nonempty proper suffix of ℓ_{j_u-1}. Since w is a bordered inverse Lyndon word, by Proposition 7.3, $\ell_1, \ldots, \ell_h$ is a $\mathcal{PMC}$ in $\mathrm{CFL}_{in}(w)$. Hence ℓ_{j_u-1} is a prefix of w. It follows that ℓ_{j_u-1} and s are comparable for the prefix order and consequently ℓ_{j_u-1} is bordered, in contradiction with $\ell_{j_u-1} \in L_{in}$. ■

7.3. Sorting suffixes in ICFL(w)

In this section we use the same notation and terminology as in Section 2.4. We prove that the same compatibility property proved in [37] holds for the sorting of the nonempty suffixes of a word w with respect to $\prec_{in}$ if we replace CFL(w) with ICFL(w).

Theorem 7.1. *Let w be a word and let $(m_1, \ldots, m_k)$ be a grouping of $\mathrm{CFL}_{in}(w)$. Then, for any r, s, $1 \leq r \leq s \leq k$, the sorting with respect to $\prec_{in}$ of the nonempty local suffixes of w with respect to $u = m_r \cdots m_s$ is compatible with the sorting with respect to $\prec_{in}$ of the corresponding nonempty global suffixes of w.*

Proof:
Let w and $(m_1, \ldots, m_k)$ be as in the statement. Let $\mathrm{CFL}_{in}(w) = (\ell_1, \ldots, \ell_h)$. Let $u = m_r \cdots m_s$, with $1 \leq r \leq s \leq k$. By Definitions 7.2, 7.3, any m_j is a concatenation of consecutive ℓ_q. Hence u is also a concatenation of consecutive ℓ_q. By Theorem 2.3, for all i, j with $first(u) \leq i < j \leq last(u)$, we have

$$suf_u(i) \prec_{in} suf_u(j) \iff suf(i) \prec_{in} suf(j). \quad \blacksquare$$

The following corollary is a direct consequence of Proposition 7.7 and Theorem 7.1.

Corollary 7.2. *Let w be a word and let $\mathrm{ICFL}(w) = (m_1, \ldots, m_k)$. Then, for any r, s, $1 \leq r \leq s \leq k$, the sorting with respect to $\prec_{in}$ of the nonempty local suffixes of w with respect to $u = m_r \cdots m_s$ is compatible with the sorting with respect to $\prec_{in}$ of the corresponding nonempty global suffixes of w.*

On the contrary, we give below a counterexample showing that the compatibility property of local and global nonempty suffixes does not hold in general for inverse Lyndon factorizations with respect to $\prec_{in}$ (and with respect to $\prec$).

Example 7.4. Let $(\Sigma, <)$ be as in Example 4.1 and let $w = daddbadc \in \Sigma^+$. Therefore, $d <_{in} c <_{in} b <_{in} a$. Consider the inverse Lyndon factorization (dad, dba, dc) of w, with $dad \ll dba \ll dc$ and the factor $u = daddba$. Consider the local suffixes $a, addba$ of u and the corresponding global suffixes adc and $addbadc$. We have that $addbadc \prec_{in} adc$ while $a \prec_{in} addba$. Consequently, in general, the compatibility property does not hold for inverse Lyndon factorizations with respect to $\prec_{in}$. It does not hold also with respect to $\prec$ and even for ICFL. Indeed, let $w = dabadabdabdabdadac \in \Sigma^+$. We know that $\mathrm{ICFL}(w) = (daba, (dab)^3, dadac)$ (see Example 7.2). For the local suffixes $dab, dabdab$ of $(dab)^3$ we have $dab \prec dabdab$ but for the corresponding global suffixes $dabdadac, dabdabdadac$ we have $dabdabdadac \prec dabdadac$.

References

[1] A. Apostolico, M. Crochemore, Fast parallel Lyndon factorization with applications, Math. Syst. Theory 28 (1995) 89–108.
[2] H. Bannai, I. Tomohiro, S. Inenaga, Y. Nakashima, M. Takeda, K. Tsuruta, A new characterization of maximal repetitions by Lyndon trees, in: P. Indyk (Ed.), 26th ACM-SIAM Symposium on Discrete Algorithms (SODA), San Diego, CA, USA, 2015, pp. 562–571.
[3] J. Berstel, D. Perrin, The origins of combinatorics on words, European J. Combin. 28 (2007) 996–1022.
[4] J. Berstel, D. Perrin, C. Reutenauer, Codes and Automata, Encyclopedia Math. Appl., vol. 129, Cambridge Univ. Press, Cambridge, 2009.
[5] S. Bonomo, S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, Suffixes, conjugates and Lyndon words, in: M.-P. Béal, O. Carton (Eds.), DLT 2013, in: Lecture Notes in Comp. Sci., vol. 7907, Springer, Berlin, 2013, pp. 131–142.
[6] K. Cattell, F. Ruskey, J. Sawada, M. Serra, C.R. Miers, Fast algorithms to generate necklaces, unlabeled necklaces, and irreducible polynomials over GF(2), J. Algorithms 37 (2000) 267–282.
[7] E. Charlier, M. Philibert, M. Stipulanti, Nyldon words, https://arxiv.org/abs/1804.09735, 2018.
[8] K.-T. Chen, R.H. Fox, R.C. Lyndon, Free differential calculus. IV: the quotient groups of the lower central series, Ann. of Math. 68 (1958) 81–95.
[9] C. Choffrut, J. Karhumäki, Combinatorics on words, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 1, Springer Verlag, Berlin, 1997, pp. 329–438.
[10] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, third edition, The MIT Press, Cambridge, Massachusetts, USA, 2009.
[11] M. Crochemore, J. Désarménien, D. Perrin, A note on the Burrows–Wheeler transformation, Theoret. Comput. Sci. 332 (2005) 567–572.
[12] M. Crochemore, C. Hancart, T. Lecroq, Algorithms on Strings, Cambridge Univ. Press, Cambridge, 2007.
[13] M. Crochemore, W. Rytter, Text Algorithms, Oxford University Press, Oxford, 1994.
[14] J.W. Daykin, C.S. Iliopoulos, W.F. Smyth, Parallel RAM algorithms for factorizing words, Theoret. Comput. Sci. 127 (1994) 53–67.
[15] J.W. Daykin, W.F. Smyth, A bijective variant of the Burrows–Wheeler Transform using V-order, Theoret. Comput. Sci. 531 (2014) 77–89.
[16] F. Dolce, A. Restivo, C. Reutenauer, On generalized Lyndon words, 2018, submitted, private communication.
[17] J.-P. Duval, Factorizing words over an ordered alphabet, J. Algorithms 4 (1983) 363–381.
[18] H. Fredricksen, J. Maiorana, Necklaces of beads in k colors and k-ary de Brujin sequences, Discrete Math. 23 (1978) 207–210.
[19] I.M. Gessel, A. Restivo, C. Reutenauer, A bijection between words and multisets of necklaces, European J. Combin. 33 (2012) 1537–1546.
[20] I.M. Gessel, C. Reutenauer, Counting permutations with given cycle structure and descent set, J. Combin. Theory Ser. A 64 (1993) 189–215.
[21] D.A. Gewurza, F. Merola, Numeration and enumeration, European J. Combin. 33 (2012) 1547–1556.
[22] S.S. Ghuman, E. Giaquinta, J. Tarhio, Alternative algorithms for Lyndon factorization, in: J. Holub, J. Zdárek (Eds.), Prague Stringology Conference (PSC) 2014, Czech Technical University in Prague, Czech Republic, 2014, pp. 169–178.
[23] J. Yossi Gil, D.A. Scott, A bijective string sorting transform, CoRR abs/1201.3077, 2012, https://arxiv.org/abs/1201.3077.
[24] D. Grinberg, "Nyldon words": understanding a class of words factorizing the free monoid increasingly, https://mathoverflow.net/questions/187451/, 2014.
[25] C. Hohlweg, C. Reutenauer, Lyndon words, permutations and trees, Theoret. Comput. Sci. 307 (2003) 173–178.
[26] J. Kärkkäinen, D. Kempa, Y. Nakashima, S.J. Puglisi, A.M. Shur, On the size of Lempel–Ziv and Lyndon factorizations, in: H. Vollmer, B. Vallée (Eds.), STACS 2017, Leibniz International Proceedings in Informatics (LIPIcs), vol. 66, Schloss Dagstuhl – Leibniz Center for Informatics, 2017, pp. 1–13.
[27] D.E. Knuth, The Art of Computer Programming, vol. 4A, Combinatorial Algorithms: Part I, Addison Wesley, 2012.
[28] D.E. Knuth, J.H. Morris Jr., V.R. Pratt, Fast pattern matching in strings, SIAM J. Comput. 6 (1977) 323–350.

[29] M. Kufleitner, On bijective variants of the Burrows–Wheeler transform, in: J. Holub, J. Zdárek (Eds.), Prague Stringology Conference (PSC) 2009, Czech Technical University in Prague, Czech Republic, 2009, pp. 65–79.
[30] H. Li, N. Homer, A survey of sequence alignment algorithms for next-generation sequencing, Brief. Bioinform. 11 (2010) 473–483.
[31] M. Lothaire, Combinatorics on Words, Cambridge Math. Lib., Cambridge Univ. Press, Cambridge, 1997.
[32] M. Lothaire, Algebraic Combinatorics on Words, Encyclopedia Math. Appl., vol. 90, Cambridge Univ. Press, Cambridge, 2002.
[33] M. Lothaire, Applied Combinatorics on Words, Encyclopedia Math. Appl., vol. 105, Cambridge Univ. Press, Cambridge, 2005.
[34] R.C. Lyndon, On Burnside problem I, Trans. Amer. Math. Soc. 77 (1954) 202–215.
[35] S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, An extension of the Burrows–Wheeler transform, Theoret. Comput. Sci. 387 (2007) 298–312.
[36] S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, Sorting suffixes of a text via its Lyndon factorization, in: J. Holub, J. Zdárek (Eds.), Prague Stringology Conference (PSC) 2013, Czech Technical University in Prague, Czech Republic, 2013, pp. 119–127.
[37] S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, Suffix array and Lyndon factorization of a text, J. Discrete Algorithms 28 (2014) 2–8.
[38] M. Mucha, Lyndon words and short superstrings, in: S. Khanna (Ed.), 24th ACM-SIAM Symposium on Discrete Algorithms (SODA), San Diego, CA, USA, 2013, pp. 958–972.
[39] D. Perrin, A. Restivo, Words, in: M. Bóna (Ed.), Handbook of Enumerative Combinatorics, in: Discrete Mathematics and Its Applications Series, Chapman and Hall/CRC, 2015, pp. 485–539.
[40] C. Reutenauer, Free Lie Algebras, London Mathematical Society Monographs, vol. 7, Oxford Science Publications, Oxford, 1993.
[41] C. Reutenauer, Mots de Lyndon généralisés, Sém. Lothar. Combin. 54 (2006) B54h.
[42] M.-P. Schützenberger, On a factorization of free monoids, Proc. Amer. Math. Soc. 16 (1965) 21–24.