Discrete Mathematics and Theoretical Computer Science **1**, 1997, 101–114

Finely homogeneous computations in free Lie algebras

Philippe Andary

Université de Rouen, Faculté des Sciences, Laboratoire d'Informatique de Rouen, F-76821 Mont-Saint-Aignan Cédex, France
E-Mail: andary@dir.univ-rouen.fr

We first give a fast algorithm to compute the maximal Lyndon word (with respect to lexicographic order) of $Ly_\alpha(A)$ for every given multidegree α in $\mathbb{N}^k$. We then give an algorithm to compute all the words living in $Ly_\alpha(A)$ for any given α in $\mathbb{N}^k$. The best known method for generating Lyndon words is that of Duval [1], which gives a way to go from every Lyndon word of length n to its successor (with respect to lexicographic order by length), in space and worst case time complexity $O(n)$. Finally, we give a simple algorithm which uses Duval's method (the one above) to compute the next standard bracketing of a Lyndon word for lexicographic order by length. We can find an interesting application of this algorithm in control theory, where one wants to compute within the command Lie algebra of a dynamical system (letters are actually vector fields).

Keywords: Lie algebras, finely homogeneous computations

1 Introduction

Let $A = \{a_1, a_2, \ldots, a_k\}$ be a set with k elements, and $\mathbb{Q}\langle A\rangle$ the associative (non-commutative) algebra on A. Defining a *Lie bracket* ($[x, y] = xy - yx$) on this $\mathbb{Q}$-module turns it into a Lie algebra, and we will denote by $\mathcal{L}(A)$ its Lie subalgebra generated by A (i.e. $\mathcal{L}(A)$ is the *free Lie algebra* on A and $\mathbb{Q}\langle A\rangle$ its *enveloping algebra*). A will now be called an *alphabet*, whose elements are the *letters*, and A^* is the *free monoid* (the set of all *words*) over A.

We know that $\mathcal{L}(A)$ is a graded $\mathbb{Q}$-module

$$\mathcal{L}(A) = \bigoplus_{n\geq 1} \mathcal{L}_n(A) \tag{1}$$

where $\mathcal{L}_n(A)$ is the submodule of $\mathcal{L}(A)$ generated by homogeneous components with degree n. Furthermore, $\mathcal{L}_n(A)$ is the direct sum of finely homogeneous submodules $\mathcal{L}_\alpha(A)$ with multidegree $\alpha \in \mathbb{N}^k$ such that $|\alpha| = n$. Hence we have

$$\mathcal{L}(A) = \bigoplus_{\alpha\in\mathbb{N}^k} \mathcal{L}_\alpha(A) \tag{2}$$

All the classical monomial basis of $\mathcal{L}(A)$ are finely homogeneous (see Reutenauer [2], for example, or Viennot [3]), i.e. their components are finely homogeneous Lie monomials, thus every computation on Lie polynomials (the elements of $\mathcal{L}(A)$) is nothing but a computation in some fine homogeneity class.

We want to emphasize the use of a particular basis of $\mathcal{L}(A)$, the Lyndon basis, introduced by Chen *et al.* in the late 1950s [4] (the interested reader can find all the details of its construction in Lothaire [5]). For this purpose, we choose a total order $a_1 < a_2 < \ldots < a_k$ on A, which induces a *lexicographic order* on A^*. A word ℓ is a *Lyndon word* if it is primitive and minimal in its conjuguacy class. This means that ℓ cannot be written as u^n for given u in A^* and $n \geq 2$, and $\ell \leq vu$ whenever $\ell = uv$. We denote by $Ly(A)$ the *set of Lyndon words* over A.

For every Lyndon word ℓ we define its *(right) standard factorization* (ℓ', ℓ'') as the unique couple of Lyndon words such that $\ell = \ell'\ell''$ and ℓ'' is of maximal length. Moreover, the recursive Lie bracketing of the standard factorization will be called the *(right) standard bracketing* of ℓ, denoted by $[\ell]$.

It would be convenient to define the *left standard factorization* (u, v) of ℓ as the unique couple of Lyndon words such that $\ell = uv$ and u is of maximal length. Then the recursive Lie bracketing of the left standard factorization will be called the *left standard bracketing* of ℓ, denoted by (ℓ).

Note that A is included in $Ly(A)$ and that $[a] = a = (a)$ for every letter a. As an exemple, let $A = \{a, b\}$ and $\ell = aaabab$, then ℓ is Lyndon because it is primitive and $\ell < s$ for each of its proper suffix s. The standard bracketings of ℓ are

$$\begin{array}{rcl} [\ell] & = & [a, [[a, [a, b]], [a, b]]] \\ (\ell) & = & [[a, [a, [a, b]]], [a, b]] \end{array}$$

It is well known that $Ly(A)$ is a factorization of A^* and that $[Ly(A)]$ is a basis of $\mathcal{L}(A)$ (called the *Lyndon basis* of $\mathcal{L}(A)$).

The paper is organized as follows. First, we give a fast algorithm to compute the maximal Lyndon word (with respect to lexicographic order) of $Ly_\alpha(A)$ for every given multidegree α in $\mathbb{N}^k$. We will see that the letters of a Lyndon word which is maximal in its fine homogeneity class are as much 'regularly distributed' as possible.

In the second part, we give an algorithm to compute all the words living in $Ly_\alpha(A)$ for any given α in $\mathbb{N}^k$. The best known method for generating Lyndon words is that of Duval [1], which gives a way to go from every Lyndon word of length n to its successor (with respect to lexicographic order by length), in space and worst case time complexity $O(n)$. It is easy to see that in the special case where $A = \{a, b\}$ and $\ell = a^{k+3}ba^{k-1}b$ ($k > 0$), there are $2^{k-1} - 1$ Lyndon words of length $2k + 4$ between ℓ and its successor in the same fine homogeneity class (namely $a^{k+2}ba^kb$). Hence, this method has an exponential worst case time complexity for our purpose.

Finally, we give a simple algorithm which uses Duval's method (the one above) to compute the next standard bracketing of a Lyndon word for lexicographic order by length. We can find an interesting application of this algorithm in control theory, where one wants to compute within the command Lie algebra of a dynamical system (letters actually are vector fields). Standard bracketing are very expansive to compute in this context, and that is why we want to generate them as quickly as possible in the lexicographic order by length.

2 Maximal Finely Homogeneous Lyndon Words

2.1 Introduction

Given α in $(\mathbb{N}^*)^k$, the question is to find a description of the maximal finely homogeneous Lyndon word with respect to lexicographic order

$$\mu_\alpha(A) = \max\{Ly_\alpha(A)\} \tag{3}$$

We will merely denote it by μ_α when no confusion is possible.

Before answering our question, let us give a short description of the problem's background. Let ℓ be a Lyndon word with fine homogeneity $\alpha \in (\mathbb{N}^*)^k$; it is well known [5] that

$$[\ell] = \ell + \sum_{\ell < w} * w \tag{4}$$

Of course, if $\ell = \mu_\alpha$ the only Lyndon word appearing in $[\ell]$ is ℓ itself, but the reciprocal may, or may not, be true. Anyhow, this led us to an algorithmic answer to our former question, when $A = \{a, b\}$ in a first time, then for any finite alphabet. Just for fun now, we know that the reciprocal is not true and we were able to find a lot of counter examples (with the help of computers) for which

$$\mathrm{supp}([\ell]) \cap Ly_\alpha(A) = \{\ell\} \tag{5}$$

For example, each of the following words $aabbabaabbababababaabbabab$, $aabbababababaabbababababab$ and $aabbababababababababababab$ are in $Ly_{(12,12)}(\{a, b\})$ and verify equation (5).

2.2 The Binary Case

Let us consider the case $k = 2$, thus $A = \{a, b\}$. We have the following:

Theorem 1 *Given any couple (n, m) of positive integers, the* MFHLynd *function below returns the maximal Lyndon word in $Ly_{(n,m)}(\{a, b\})$ with respect to lexicographic order:*

```
function MFHLynd((n, m), {a, b})
# Input  : (n, m) is a couple of positive integers.
#          {a, b} is the alphabet.
# Output : max{Ly_(n,m)({a, b})}.
begin
  if (n = 1) or (m = 1)
  then Return(a^n b^m)
  else if n ≥ m
       then q := n div m
            r := n  mod m
            if r = 0
            then Return(a^(q+1) b a^(q-1) b (a^q b)^(m-2))
            else Return(MFHLynd((r, m), {a, a^q b}))
```

```
        else q := m div n
             r := m  mod n
             if r = 0
             then Return(ab^{q-1}ab^{q+1}(ab^q)^{n-2})
             else Return(MFHLynd((n − r, r), {ab^q, ab^{q+1}}))
end
```

Proof. Let (n, m) be any couple of positive integers. We will denote by ν the result of the MFHLynd function. As this algorithm is recursive, we denote by n_i, m_i and A_i the instances of formal parameters during the i-th function call ($n_1 = n$, $m_1 = m$ and $A_1 = \{a, b\}$). Note that the algorithm will terminate in a finite number of steps, say p, since through each step i the sum $n_i + m_i$ will strictly decrease. Furthermore, let us denote by μ_i the word $\mu_{(n_i,m_i)}$ at each stage i.

If $p = 1$ then n is a multiple of m, or m is a multiple of n, and we obviously have $\nu = \mu_1 = \mu_{(n,m)}(\{a, b\})$.

So let us consider the case $p > 1$, i.e. either $n = q_1 m + r_1$ with $q_1 \geq 1$ and $0 < r_1 < m$, or $m = q_1 n + r_1$ with $q_1 \geq 1$ and $0 < r_1 < n$. Then (n_2, m_2) is (r_1, m) or $(n - r_1, r_1)$, depending on the value of n and m, and A_2 is $\{a, a^{q_1}b\}$ or $\{ab^{q_1}, ab^{q_1+1}\}$ accordingly. Of course, $\mu_2 \leq \mu_1$ because $Ly(A_2) \subset Ly(A_1)$, and if we prove that

$$\mu_1 \in A_2^* \tag{6}$$

the previous inequality turns to an equality. Now the end of the proof is trivial: we have $\nu = \mu_p$, but relation (6) implies

$$\mu_1 = \mu_2 = \ldots = \mu_p \tag{7}$$

Finally, for the proof to be complete, we have to justify relation (6). For this purpose, we will first suppose that $n = q_1 m + r_1$ (thus $A_2 = \{a, a^{q_1}b\}$). We can write

$$\mu_1 = a^{q_1+\varepsilon_1}b \ldots a^{q_1+\varepsilon_m}b \tag{8}$$

and one can easily see that $\varepsilon_1 = 1$, $\varepsilon_2 \in \{0, 1\}$ and $-q_1 \leq \varepsilon_i \leq 1$ for $3 \leq i \leq m$ (since $\mu_1 \in Ly_{(n,m)}(\{a, b\})$). Next, the existence of $h = \min\{i : \varepsilon_i < 0\}$ would imply the existence of a Lyndon word ℓ in $Ly_{(n,m)}(\{a, b\})$ which would be greater than μ_1. Indeed, let us write $\mu_1 = wv_1v_2 \ldots v_t$, where $w = a^{q_1+1}b \ldots a^{q_1+\varepsilon_{h-1}}ba^{q_1-x}b$ ($x = |\varepsilon_h|$) and the v_i's are defined by the left standard bracketing of μ_1

$$(\mu_1) = [[\ldots[[(w), (v_1)], (v_2)], \ldots], (v_t)] \tag{9}$$

Furthermore, we will consider

$$w' = a^{q_1+1}b \ldots a^{q_1+\varepsilon_{h-1}-1}b \tag{10}$$

$$w'' = a^{q_1-x+1}b \tag{11}$$

Some v_i's are greater than w', some are not, but they are all greater than w. So we can split those v_i between w and w' in two words, and apply the same transformation as in $w \to (w', w'')$. Now construct

a sequence of Lyndon words, with those v_i that are greater than w' on the one hand, and the splitted ones (v'_i, v''_i) on the other. Then concatenating this reordered sequence will give us a new Lyndon word which is greater than μ_1 (because the former begins with w', and the latter with w). But this fact contradicts the maximality of μ_1, so h doesn't exist.

Now, the second case $m > n$, m not a multiple of n, is not substantially different from the previous one. Let us denote by A^0 the alphabet A equipped with the reverse order $b < a$. Then $\varphi : A \to A^0$ such that $\varphi(a) = b$ and $\varphi(b) = a$ extends to an isomorphism from A^* onto $(A^0)^*$. Moreover, φ is order preserving, and we have clearly

$$\mu_{(n,m)}(\{a,b\}) = \big(\varphi(\mu_{(m,n)}(\{a,b\}))\big)^{\sim} \tag{12}$$

where $(u_1 \dots u_k)^{\sim} = u_k \dots u_1$ is the mirror image of the word u. □

The analysis of this algorithm is trivial, since its recurrence tree is structurally equivalent to that of Euclide's algorithm for computing the greatest common divisor of n and m. So we have

Theorem 2 *For given positive integers (n, m), the worst case time complexity for* `MFHLynd` *function is* $O(\log(\max(n, m)))$.

Proof. See Knuth [6] for instance, *Theorem 4.5.3-F* and *Corollary 4.5.3-L*, p. 343. □

2.3 The General Case

In a second time, we were able to generalize the function `MFHLynd` to any finite alphabet A in the following manner:

```
function MFHLynd(α, A)
# Input  : α = (α_1, ..., α_k) is a k-tuple of positive integers.
#          A = {a_1, ..., a_k} is the (ordered) alphabet.
# Output : max{Ly_α(A)}.
begin
  n := α_1
  m := Σ_{i=2}^{k} α_i
  if (n = 1) or (m = 1)
  then Return(a_1^n a_k^{α_k} ... a_2^{α_2})
  else if n ≥ m
       then q := n div m
            r := n  mod m
            if r = 0
            then if k = 2
                 then Return(a_1^{q+1} a_2 a_1^{q-1} a_2 (a_1^q a_2)^{m-2})
                 else Return(MFHLynd((α_2, ..., α_k), {a_1^q a_2, ..., a_1^q a_k}))
            else Return(MFHLynd((r, α_2, ..., α_k), {a_1, a_1^q a_2, ..., a_1^q a_k}))
       else h := n
            for i decreasing from k downto 2 do
```

$q_i := \alpha_i \operatorname{div} h$
$r_i := \alpha_i \mod h$
$h := h - r_i$
`if` $(\forall i, r_i = 0)$
`then Return(`$a_1 a_k^{q_k} \ldots a_2^{q_2-1} a_1 a_k^{q_k} \ldots a_2^{q_2+1} (a_1 a_k^{q_k} \ldots a_2^{q_2})^{n-2}$`)`
`else` $t := 0$
`for` i `increasing from` 2 `to` k `do`
`if` $r_i > 0$
`then` $t := t + 1$
$i_t := i$
`Return(MFHLynd(`$(n - \sum_{j=1}^{t} r_{i_j}, r_{i_1}, \ldots, r_{i_t})$`,`
$\{a_1 a_k^{q_k} \ldots a_2^{q_2}, a_1 a_k^{q_k} \ldots a_{i_1-1}^{q_{i_1-1}} a_{i_1}^{q_{i_1}+1}, \ldots, a_1 a_k^{q_k} \ldots a_{i_t-1}^{q_{i_t-1}} a_{i_t}^{q_{i_t}+1}\}$`))`
`end`

We can reasonably expect the complexity of such an algorithm to be a function in the size of the alphabet and in the maximum of the components α_i, but unfortunately, we were not able to analyse the behaviour of `MFHLynd` function in this case deeply enough.

3 Fine Homogeneous Generation of Lyndon Words

3.1 Introduction

Now the problem is to find all words in $Ly_\alpha(A)$, for every α in $(\mathbb{N}^*)^k$. We will need the following well known reducing property (see, for example, Duchamp and Krob [7] for a partially commutative version of this proposition):

Proposition 1 *Let $k > 1$, $A = \{a_1 < \ldots < a_k\}$ and $A' = (A - a_k)a_k^*$. Then $Ly(A) = Ly(A') \cup \{a_k\}$.*

Let us see, with an example, how we will dissect Lyndon words to solve our problem. Suppose $\alpha = (3,3)$ and $A = \{a,b\}$ ($k = 2$). By proposition 1, if $A' = ab^*$ then $Ly(A) = Ly(A') \cup b$. Denoting by B the minimal subset of A' such that $Ly_\alpha(A) \subset B^*$, we mean the code

$$B = \{a, ab, abb, abbb\} \tag{13}$$

We want to find out all Lyndon words on B that are in $Ly_\alpha(A)$; we know their fine homogeneity, but we don't know how they look like: there can be either two a and one $abbb$, or one a, one ab and one abb. (Note that we cannot take three times ab, since Lyndon words are primitive.)

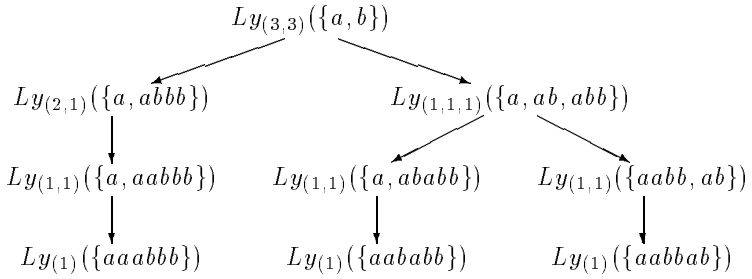
Hence, our initial problem breaks down into two 'smaller' subproblems – find all of the words in $Ly_{(2,1)}(\{a, abbb\})$ and $Ly_{(1,1,1)}(\{a, ab, abb\})$. These subproblems are smaller insofar as the length of the involved Lyndon words are smaller over the new alphabets than over the previous one (here, we have to compare $\{a, abbb\}$ and $\{a, ab, abb\}$ with $\{a, b\}$). It is still necessary to detail the way we compute $Ly_{(1,1,1)}(\{a, ab, abb\})$. Here we have $\alpha = (1,1,1)$, $A = \{a, ab, abb\}$ ($k = 3$) and B is the code

$$B = \{a, aabb, ab, ababb\} \tag{14}$$

Every word in $Ly_\alpha(A)$ is a Lyndon word over B, with either one $aabb$ and one ab, or one a and one $ababb$. Thus

$$Ly_{(1,1,1)}(\{a, ab, abb\}) = Ly_{(1,1)}(\{aabb, ab\}) \cup Ly_{(1,1)}(\{a, ababb\}) \tag{15}$$

Clearly, iterating this process will give, in a finite number of steps, every Lyndon word on $\{a, b\}$ with fine homogeneity $(3, 3)$. In fact, our algorithm will ultimately construct all alphabets with only one letter which is a word in $Ly_{(3,3)}(\{a, b\})$. For example, the subproblems' decomposition tree in this case is



and finally, $Ly_{(3,3)}(\{a, b\}) = \{aaabbb, aababb, aabbab\}$.

3.2 Translation in Terms of Partition

Looking closely at our process gives the feeling that breaking down our problem into smaller ones is exactly the same as finding a set of partitions under some constraints. For example, in the first stage of our previous computation we have searched for the set of elements x in $\mathbb{N}^4$ such that $x_1 + x_2 + x_3 + x_4 = 3$, verifying the constraint $x_1 + 2x_2 + 3x_3 + 4x_4 = 6$. We have accepted $(2, 0, 0, 1)$ and $(1, 1, 1, 0)$, refused $(0, 3, 0, 0)$ (since Lyndon words are primitive), and thereby found all the solutions in x. Next, during the decomposition of $Ly_{(1,1,1)}(\{a, ab, abb\})$, we have searched for the set of elements (x, x') in $\left(\mathbb{N}^2\right)^2$ verifying

$$\begin{aligned} x_1 + x_2 &= 1 \\ x'_1 + x'_2 &= 1 \\ x_1 + 2x_2 + x'_1 + 2x'_2 &= 3 \end{aligned}$$

Here, again, we have found all the solutions in (x, x'), namely $((1, 0), (0, 1))$ and $((0, 1), (1, 0))$.

For a good formulation of the problem in terms of composition and partition, we will need some notations and definitions. For every x in $\mathbb{N}^p$, we define

$$|x| = x_1 + x_2 + \ldots + x_p \tag{16}$$

and

$$||x|| = x_1 + 2x_2 + \ldots + px_p \tag{17}$$

A *composition* of the positive integer n into p parts is a p-tuple x in $\mathbb{N}^p$ such that $|x| = n$. Furthermore, when $||x|| = n + \lambda$ we say that x is constrained by λ. We will denote by $\chi(n,p)$ (resp. $\chi_\lambda(n,p)$) the set of all compositions of n into p parts (resp. the set of all compositions constrained by λ).

A *partition* of the positive integer n into p parts is a p-tuple y in $(\mathbb{N}^*)^p$ such that $|y| = n$ and $y_1 \geq \ldots \geq y_p \geq 1$. We will denote by $\pi(n,p)$ the set of all partitions of n into p parts, and by $\pi_\lambda(n,p)$ the set of all partitions of n into p parts whose value does not exceed λ (thus $\lambda \geq y_1$).

For consistency with our previous notations, we will call $\bar{x} = (x_{i_1}, \ldots, x_{i_t})$ the composition x without any 0's (thus, for all $1 \leq j \leq t$, $x_{i_j} > 0$ and every other x_i are zero), and we will denote by B_x the subset of elements of B at the i_j-th place for lexicographic order ($1 \leq j \leq t$).

Here is a trivial property which will be helpful for computing the compositions, since we know a constant worst case time complexity algorithm to derive the successor of a partition (see Nijenhuis and Wilf [8]).

Proposition 2 *For given n, p and λ, every partition in $\pi_p(n+\lambda, n)$ is equivalent to a unique composition in $\chi_\lambda(n,p)$.*

Proof. The proof is immediate if we see that, given a partition y in $\pi_p(n+\lambda, n)$, the corresponding composition x in $\chi_\lambda(n,p)$ is such that x_i is nothing but the number of parts (in y) equal to i (for all i in $[1,p]$). □

As an example, all the compositions constrained by 3, of 3 into no more than 4 parts are $(2,0,0,1)$, $(1,1,1,0)$ and $(0,3,0,0)$. Proposition 2 says that they are equivalent to the partitions of 6 into 3 parts not greater than 4, namely $(4,1,1)$, $(3,2,1)$ and $(2,2,2)$; which will become evident with the notation $1^2 4$, 123 and 2^3. For every partition y in this last form, which is equivalent to a composition x, we will write $\bar{y}$ the sequence of non zero exponents (for example, if $y = 123$ then $\bar{y} = (1,1,1)$); so that $\bar{x} = \bar{y}$.

3.3 The Algorithm

Now we can state

Theorem 3 *Let $k > 1$, $A = \{a_1 < \ldots < a_k\}$ and $\alpha = (\alpha_1, \ldots, \alpha_k) \in (\mathbb{N}^*)^k$. Set $B := \{a_i a_k{}^j : 1 \leq i \leq k-1,\ 0 \leq j \leq \alpha_k\}$, $\Phi := \chi(\alpha_k, k)$ and for each ϕ in Φ:*

$$X_\phi := \chi_{\phi_1}(\alpha_1, \alpha_k + 1) \times \ldots \times \chi_{\phi_{k-1}}(\alpha_{k-1}, \alpha_k + 1).$$

Then we have

$$Ly_\alpha(A) = \bigcup_{\phi \in \Phi} \bigcup_{x \in X_\phi} Ly_x(B) \tag{18}$$

Proof. On the one hand, we will study the case $k = 2$, so we consider $A = \{a, b\}$, $\alpha = (n, m)$ and $B = \{ab^j : 0 \leq j \leq m\}$. There is $m+1$ elements in B, and to each composition $x \in \mathbb{N}^{m+1}$, we associate the fine homogeneous class of words in B^*

$$F_x = \{w \in B^* : |w|_{ab^j} = x_j,\ 0 \leq j \leq m\}. \tag{19}$$

Since F_x is never empty, there is at least one Lyndon word in F_x (note that, by Proposition 1, $Ly(B) \subset Ly(A)$) and we want to determine those x such that F_x contains a Lyndon word in $Ly_\alpha(A)$ – thus every

Lyndon word in F_x will belong to $Ly_\alpha(A)$. Of course, $Ly_\alpha(A)$ will be the union set of $Ly_x(B)$ over all these x. But it is clear that the only criteria for $x \in \mathbb{N}^{m+1}$ to be choosen are $|x| = n$ and $||x|| = n + m$. So x must be a composition of n into no more than $m + 1$ parts, constrained by m, and the theorem is proved when $k = 2$ (with $\Phi = \{(m)\}$ and $X_{(m)} = \chi_m(n, m + 1)$).

On the other hand, we will now consider the case $k > 2$. Then we have $B = \{a_i a_k{}^j : 1 \leq i \leq k-1,\ 0 \leq j \leq \alpha_k\}$ and $|B| = k(\alpha_k + 1)$. Here, the problem is to find all $x = (x^{(1)}, \ldots, x^{(k-1)})$ in $\left(\mathbb{N}^{\alpha_k+1}\right)^{k-1}$ such that

$$\left\{ \begin{array}{rcl} |x^{(i)}| & = & \alpha_i \quad (1 \leq i \leq k-1) \\ \sum_{i=1}^{k-1} ||x^{(i)}|| & = & |\alpha| \end{array} \right. \tag{20}$$

Thus we must have, for all i, the inequalities

$$\alpha_i \leq ||x^{(i)}|| \leq \alpha_i + \alpha_k \tag{21}$$

and it is clear now that the set of solutions for system (20) is exactly the union set over $\phi \in \chi(\alpha_k, k)$ of the $|B|$-tuple x of (non-negative) integers verifying

$$\left\{ \begin{array}{rcl} |x^{(i)}| & = & \alpha_i \\ ||x^{(i)}|| & = & \alpha_i + \phi_i \end{array} \right. \tag{22}$$

for all $1 \leq i \leq k-1$. □

From Theorem 3 and Proposition 2, we can deduce the function `FHGLynd`, which takes as input an alphabet A and a multidegree α, and outputs the set of elements in $Ly_\alpha(A)$

```
function FHGLynd(α, A)
# Input  : α = (α_1, ..., α_k) is a k-tuple of positive integers.
#          A = {a_1, ..., a_k} is the (ordered) alphabet.
# Output : Ly_α(A).
begin
  if (k = 2) and ((α_1 = 1) or (α_2 = 1))
  then Return({a_1^α1 a_2^α2})
  else B := {a_i a_k^j : 1 ≤ i ≤ k-1,  0 ≤ j ≤ α_k}
       XPhi := {}
       Lyn := {}
       for φ in χ(α_k, k) do
           X := π_{α_k+1}(α_1 + φ_1, α_1)
           for i increasing from 2 to k-1 do
               X := X × π_{α_k+1}(α_i + φ_i, α_i)
           XPhi := XPhi ∪ X
       for x in XPhi do
           Lyn := Lyn ∪ FHGLynd(x̄, B_x)
       Return(Lyn)
end
```

3.4 Tests on the Execution Time

Except for the FHGLynd function, the only algorithm at hand for computing the homogeneity class of Lyndon words could be one derived from Duval's method. As we will see in the next section, this method allows us to go from one Lyndon word to the (lexicographically) next one with the same length. Thus, a new function, say FHGL, is born: given an alphabet A with k letters and a homogeneity vector α (with $n = |\alpha|$), construct the set $Ly_\alpha(A)$ by iterative application of Duval's method, picking up those words whose homogeneity is α. However, we have to bear in mind that this is an exponential algorithm, since the number of Lyndon words whose length is n is $O(\frac{k^n}{n})$. We wanted to get a feeling on the complexities of both methods. For this purpose, both algorithms have been implemented in Maple V.3 on an SS10 workstation under Solaris 2.3 operating system.

The first time we found out that, for the generation of a given homogeneity class of Lyndon words, the FHGLynd function is more efficient than FHGL.

The second time, we decided, given a finite alphabet A of size k and an integer n, to generate all the words in $Ly_n(A)$, first in lexicographic order by Duval's method (algorithm D), and then by homogeneity classes with our function (algorithm A). Although the average running time of A per homogeneity class becomes gradually smaller than the running time of D while k and/or n grow, algorithm A does not seem to be linear in n, since algorithm D does, and their execution time ratio increases.

Hence, it is obvious that these two algorithms are devoted to different problems: FHGL is well suited for lexicographic enumeration of Lyndon words of given length, while FHGLynd is just right for homogeneity generation of Lyndon words.

4 Generating the Lyndon Basis

4.1 Introduction

We want here to generate the standard bracketing of Lyndon words for lexicographic order by length. We know an efficient algorithm, due to Duval [1], which gives the next Lyndon word for lexicographic order by length with a constant average time complexity [9]. The idea is quite simple: adapt Duval's method to obtain the index of the standard factorization in addition to the Lyndon word, of course, preserving time complexity; then a convenient tree data structure could be used to recursively store all the standard factorizations.

4.2 Duval's Theorem

We will write '$\leq$' and '$\preceq$' for lexicographic order and lexicographic order by length, respectively. These total orders on A^* are defined as follows:

$$u \leq v \quad \text{iff} \quad \begin{cases} & v \in uA^*, \\ \text{or} & \\ & (\exists r,s,t \in A^*)(\exists a,b \in A)(u = ras, v = rbt, a < b) \end{cases}$$

$$u \preceq v \quad \text{iff} \quad \begin{cases} & |u| < |v|, \\ \text{or} & \\ & |u| = |v|, u \leq v \end{cases}$$

For all Lyndon word ℓ, we will denote by $S(\ell)$, when it makes sense, its *successor*, for lexicographic order, which is not in ℓA^*. It is computed as follows: remove all terminal maximal letters of ℓ, then

replace the last non-maximal letter by its successor in A. Notice that $S(\ell)$ has always a smaller size than ℓ. For example, on $A = \{a, b, c\}$, $S(aaacc)$ is aab (not $aabab$!), and $S(aaabb)$ is $aaabc$. Duval proves that

Theorem 4 *If ℓ is a Lyndon word with length n, which is not maximal in its fine homogeneous class ; if w is the following Lyndon word of ℓ for '$\preceq$', then we have:*

$$w = \begin{cases} S(\ell) & \text{if } |S(\ell)| = n \\ {u_1}^{q_1} \dots {u_t}^{q_t} & \text{otherwise} \end{cases} \tag{23}$$

where the u_i are Lyndon words derived from prefixes of ℓ.

More precisely, the u_i are defined as follows (where $u[1..j]$ is the prefix of u, whose length is j):

$$\begin{array}{llll} u_1 = S(\ell), & d_1 = |u_1|, & q_1 = (n-1) \operatorname{div} d_1, & r_1 = 1 + (n-1) \bmod d_1, \\ u_2 = S(u_1[1..r_1]), & d_2 = |u_2|, & q_2 = r_1 \operatorname{div} d_2, & r_2 = r_1 \bmod d_2 > 0, \\ & & \dots & \\ u_t = S(u_1[1..r_{t-1}]), & d_t = |u_t|, & q_t = r_{t-1} \operatorname{div} d_t, & r_t = r_{t-1} \bmod d_t = 0. \end{array}$$

Let us have a feeling on the behaviour of the underlying algorithm on the Lyndon word $\ell = a^3bab^{10}$ (over $A = \{a, b\}$). The first prefix we compute is $(a^3b^2)^2$, since $S(\ell) = a^3b^2$, and because r_1 must be positive. Then we get $(a^3b^2)^2a^2b$ since $S(a^3b^2) = a^2b$, and finally, we compute $S(a^2) = ab$ giving $w = (a^3b^2)^2a^2bab$.

This algorithm is $O(|\ell|)$ in worst case time complexity, and Berstel and Pocchiola [9] have proved it to be optimal in average time complexity (actually, they give the asymptotic bound $(k+1)/k$, where k is the cardinal of A).

4.3 Adapted Duval's Algorithm

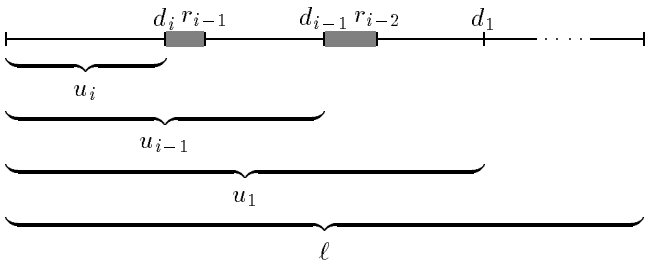
We have seen that Duval constructs the next Lyndon word by computing successive prefixes, thus we had the natural idea to use the left standard factorization instead of the right one, as usual. So, if we denote by $i(u)$ the index of the left standard factorization of a Lyndon word u, and with the notations of Theorem 4, we have

Theorem 5 *If $w = S(\ell)$, then $i(w) = i(\ell)$; otherwise if $w = {u_1}^{q_1}u_2$, then $i(w) = d_1$, else $w = {u_1}^{q_1} \dots {u_t}^{q_t}$ and $i(w) = n - d_t$.*

Proof. First, we have to notice that, for all $1 < i \leq t$, the relation $u_i = S(u_1[1..r_{i-1}])$ implies the existence of the integer d_i in $[\![1\,,\, r_{i-1}]\!]$ and the letter $\lambda_i = \operatorname{succ}(u_{i-1}[d_i])$ such that (set $u_0 = \ell$)

$$u_i = u_{i-1}[1..d_i - 1]\lambda_i \tag{24}$$

So we have



where grayed zones means the suppression of maximal letters.

Now we can carry through the proof. If $w = S(\ell)$ this is because $|S(\ell)| = n$, that is $S(\ell) = \ell_1 \dots \ell_{n-1}\mathrm{succ}(\ell_n)$ with $\ell_n < z$. In this case $i(w)$ is obviously $i(\ell)$.

Otherwise, there are two cases left. On the one hand, when $w = {u_1}^{q_1}u_2$ we must have $i(w) = d_1$ because of the preliminary remark. On the other hand, in the general case ($t > 2$, or $t = 2$ and $q_2 > 1$), the longest prefix of w which is Lyndon, is at least as long as ${u_1}^{q_1} \dots {u_{t-1}}^{q_{t-1}}{u_t}^{q_t-1}$, since $u_1 < u_2 < \dots < u_t$. But according to the preliminary remark, it cannot be longer than this word. Hence $i(w) = \sum_{i=1}^{t-1} q_i d_i + (q_t - 1)d_t = n - d_t$. □

As an example, let us consider $A = \{a, b\}$ and $n = 5$. Then we have the following table:

$(\ell, i(\ell))$	$w = {u_1}^{q_1} \dots {u_t}^{q_t}$	$i(w)$
$(aaaab, 1)$	$(aaab)(b)$	4
$(aaabb, 4)$	$(aab)(ab)$	3
$(aabab, 3)$	$(aabb)(b)$	4
$(aabbb, 4)$	$(ab)^2(b)$	2
$(ababb, 2)$	$(abb)(b)^2$	4

for the four first lines $i(w)$ is d_1, since $t = 2$ and $q_2 = 1$, but for the fifth it is $n - d_2 = 4$ since $q_2 > 1$.

Now, for the sake of completeness, let us give the slight improvement of Duval's algorithm

---

```
function NextLynd(ℓ,i(ℓ))
# Input  : ℓ is a Lyndon word which is not maximal in its fine
#          homogeneous class.
#          i(ℓ) is the index of the standard factorization of ℓ.
# Output : w, the successor of ℓ with same length, for lexicogra-
#          phic order by length ; or the empty word if ℓ is maximal
#          in its fine homogeneous class.
#          i(w), the index of the standard factorization of w ;
#          or 0 if ℓ is maximal in its fine homogeneous class.
begin
  let z be the maximal letter of A for lexicographic order
  n := |ℓ|
  w := ℓ
  k := n
  while w[k] = z do
        k := k − 1
  w[k] := succ(w[k])
  if w[1] = z
  then Return(ε,0)
  else t := 1
```

```
        i := 0
        d_1 := k
        while k < n − d_1 do
              for j increasing from 1 to d_1 do
                    w[k + j] := w[j]
              k := k + d_1
        while k ≠ n do
              t := t + 1
              i := k
              for j increasing from 1 to n − k do
                    w[k + j] := w[j]
              k := n
              while w[k] = z do
                    k := k − 1
              w[k] := succ(w[k])
              d_t := k − i
              if t = 2
              then q_2 := 1
                   while k ≤ n − d_2 do
                         for j increasing from 1 to d_2 do
                               w[k + j] := w[i + j]
                         k := k + d_2
                         q_2 := q_2 + 1
              else while k ≤ n − d_t do
                         for j increasing from 1 to d_t do
                               w[k + j] := w[i + j]
                         k := k + d_t
        if t = 1
        then Return(w, i(ℓ))
        else if t = 2 and q_2 = 1
             then Return(w, d_1)
             else Return(w, n − d_t)
end
```

---

where $u[j]$ means the j^{th} letter of u, and $\mathrm{succ}(\lambda)$ is the following letter of λ in A.

It is clear that the complexity is the same as for the original algorithm, since we have added only two tests in the function's body (and also the use of four new integer variables).

References

[1] Duval, J.-P. (1988). Génération d'une section des classes de conjuguaison et arbre des mots de Lyndon de longueur bornée. *Theor. Comput. Sci.* **60**, 255–283.

[2] Reutenauer, C. (1993). *Free Lie Algebras*. London Mathematical Society Monographs, new series, Vol. 7. Academic Press.

[3] Viennot, X. G. (1978). *Algèbres de Lie libres et monoïdes libres*. Lecture Notes in Mathematics **691**. Springer-Verlag.

[4] Chen, K. T., Fox, R. H. and Lyndon, R. C. (1958). Free differential calculus, IV. The quotient groups of the lower central series. *Ann. Mathematics* **68**, 81–95.

[5] Lothaire, M. (1983). Combinatorics on words. *Encyclopedia of Mathematics* **17**. Addison-Wesley.

[6] Knuth, D. E. (1981). *The Art of Computer Programming*, vol. 2*: Semi-numerical algorithms (2nd ed.)* Addison-Wesley.

[7] Duchamp, G. and Krob, D. (1994). Combinatorics in trace monoids II. In: Diekert and Rozenberg, editors, *The Book of Traces.*

[8] Nijenhuis, A. and Wilf, H. S. (1975). *Combinatorial Algorithms*. Academic Press.

[9] Berstel, J. and Pocchiola, M. (1992). Average cost of Duval's algorithm for generating Lyndon words. *Preprint no. 92-8*, Laboratoire d'Informatique de l'Ecole Normale Supérieure, Paris.