Generating random braids

Volker Gebhardt,* Juan González-Meneses†

June 30, 2012

Abstract

We present an algorithm to generate positive braids of a given length as words in Artin generators with a uniform probability. The complexity of this algorithm is polynomial in the number of strands and in the length of the generated braids.

As a byproduct, we describe a finite state automaton accepting the language of lexicographically minimal representatives of positive braids that has the minimal possible number of states, and we prove that its number of states is exponential in the number of strands.

1 Introduction

Recently, many research papers have appeared that describe effective computations with braids on n strands. In many cases, the results of performing a certain computation for a set of 'random' or 'pseudo-random' braids of a given length are reported. Usually, the authors generate so-called *positive braids*, that is, elements of the submonoid B_n^+ of the braid group B_n generated (as a monoid) by the standard Artin generators $\{\sigma_1, \ldots, \sigma_{n-1}\}$. Methods commonly used to generate such positive braids are the following:

1. In order to generate a positive braid of length k, choose k times an element of $\{\sigma_1, \ldots, \sigma_{n-1}\}$ with uniform probability, and form their product.

2. In order to generate a positive braid of canonical length k, choose k times a simple braid with uniform probability on the (finite) set of simple braids and compute their product; if the obtained braid has canonical length smaller than k, discard it and try again.

However, none of these procedures generates braids with a uniform distribution; some braids are far more likely to appear than others. For instance, if one uses the first procedure to produce positive braids of length 6 in B_4, the probability

of obtaining the braid $(\sigma_1)^6$ is 3^{-6}, as there is only one way to write this braid as a positive word in Artin generators. On the other hand, the probability of obtaining the braid $\Delta = \sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1$ is $16 \cdot 3^{-6}$, as there are 16 distinct ways to write $\Delta \in B_4$ as a positive word in Artin generators [8]. That is, the above procedure is 16 times more likely to generate Δ than to generate $(\sigma_1)^6$. This bias becomes more dramatic as n and the length of the braids involved increase.

In this paper we shall give a procedure to generate random positive braids of given length with a uniform probability. That is, given n and k as input, the algorithm generates a positive braid in B_n of length k, in such a way that the probability of obtaining any given braid is $1/x_{n,k}$, where $x_{n,k}$ is the number of positive braids of length k in B_n.

The structure of the paper is as follows. Section 2 describes the basic idea of generating uniformly random positive braids via lexicographically minimal representative words (lex-representatives, for short). In Section 3, we count the braids in B_n^+ of a given length k using a result by Bronfman. In Section 4, we develop a description of lex-representatives, which will be used in Section 6 to count the lex-representatives that start with a given prefix, completing the description of our algorithm.

It is known that the set of lex-representatives is a regular language. In Section 5, we show that our description of lex-representatives yields an acceptor for this regular language that has the minimal number of states, and that the number of states is exponential in n. This shows, in particular, that using standard language theoretical techniques to generate uniformly random positive braids is not efficient. Finally, in Section 7, we analyse the complexity of our algorithm and give timing results.

2   Structure of the algorithm

The basic idea of our algorithm to produce random positive braids is the following. Let $\mathcal{A}_n = \{\sigma_1, \ldots, \sigma_{n-1}\}$ be the set of standard generators (or *atoms*) in B_n^+. Let $\mathcal{A}_n^*$ be the free monoid generated by $\mathcal{A}_n$. We know there is a morphism of monoids $b : \mathcal{A}_n^* \to B_n^+$ which sends each element in $\mathcal{A}_n^*$ (a word in $\sigma_1, \ldots, \sigma_{n-1}$) to the positive braid it represents. As we saw above, the map b is not injective, so we are going to define a section of b. For that purpose, we will order the elements in $\mathcal{A}_n^*$ having the same length using $<_{\text{lex}}$, which is the lexicographical order in which $\sigma_1 < \sigma_2 < \cdots < \sigma_{n-1}$.

Definition 2.1. *Given $\beta \in B_n^+$, we define the* **lex-representative** *of β to be*

$$\omega(\beta) = \min_{<_{\text{lex}}}\{b^{-1}(\beta)\} \in \mathcal{A}_n^*.$$

In other words, $\omega(\beta)$ is the smallest positive word, with respect to $<_{\text{lex}}$, that represents β. Notice that this is well defined as B_n^+ is a homogeneous monoid (all words representing a given element have the same length). For instance $\omega(\sigma_3\sigma_1) = \sigma_1\sigma_3$, $\omega(\sigma_2\sigma_1\sigma_2) = \sigma_1\sigma_2\sigma_1$ and for $\Delta \in B_4$, $\omega(\Delta) = \sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1$. Here the arguments of ω are braids in B_n^+, whereas the images are words in $\mathcal{A}_n^*$. It is clear that $\omega : B_n^+ \to \mathcal{A}_n^*$ is a section of b, so it is injective. Moreover, as B_n^+ is homogeneous, we can define $(B_n^+)_k$ to be the set of positive braids of length k, and $L_{n,k}$ to be the set of lex-representatives of length k (L stands for language). Then $\omega : (B_n^+)_k \to L_{n,k}$ is a bijection, whose inverse is $b_{|_{L_{n,k}}}$.

Our algorithm will produce a random element of $L_{n,k}$ with uniform probability. By the above arguments, this is equivalent to producing a random positive braid of length k with uniform probability on $(B_n^+)_k$. There are three main steps:

1. Determine the size of $L_{n,k}$, that is, compute $x_{n,k} = \left|(B_n^+)_k\right|$.

2. Choose a random integer r between 1 and $x_{n,k}$.

3. Find the r-th word in $L_{n,k}$, where $L_{n,k}$ is ordered by $<_{\text{lex}}$.

The second step poses no difficulty (assuming we know how to generate random integers), so our task is to be able to perform steps 1 and 3 in polynomial time.

It is useful to identify the lex-representatives of braids in B_n^+ as the vertices of a rooted tree, with the elements of $L_{n,k}$ being the vertices at depth k and edges given by the prefix partial order in the monoid $\mathcal{A}_n^*$. (That is, the root corresponds to the trivial word ϵ, and given a word $w \in L_{n,s}$ and $\sigma_i \in \mathcal{A}_n$ such that $w\sigma_i \in L_{n,s+1}$, there is an edge labelled σ_i joining w to $w\sigma_i$.)

Figure 1 shows the tree of lex-representatives in B_4^+ truncated at depth 3; the elements of $L_{4,3}$ correspond to the leaves of the truncated tree. We have $|L_{4,0}| = 1$, $|L_{4,1}| = 3$, $|L_{4,2}| = 8$ and $|L_{4,3}| = 19$. Notice that there are no vertices corresponding to the words $\sigma_3\sigma_1$ and $\sigma_2\sigma_1\sigma_2$, as these words are not lex-representatives. The fact that there are 19 leaves means that there are 19 positive braids in B_4^+ of length 3. Hence, in this example, our algorithm will choose a random number r between 1 and 19, and it will look for the r-th leaf of the rooted tree, counting from left to right, as leaves are ordered lexicographically by construction.

Of course, it is not efficient at all to try to compute the whole set $L_{n,k}$, as it has exponential size with respect to k. We shall overcome this problem by counting suitable subsets of leaves of the truncated rooted tree, as follows.

Definition 2.2. *Given a word $w \in \mathcal{A}_n^*$ and an integer $m \in \{0, \ldots, n-1\}$, we define $x_{n,k}(w, m)$ to be the number of words in $L_{n,k}$ of the form ww', where $w' \in \mathcal{A}_n^*$ does not start with $\sigma_1, \ldots, \sigma_m$.*

Obviously, $x_{n,k}(w, m) = 0$ if w is not a lex-representative, or if $|w| > k$. Moreover, if w is a lex-representative, then $x_{n,k}(w, n-1) = 1$ if $k = |w|$ and $x_{n,k}(w, n-1) = 0$ otherwise. We will show in Section 3 how to compute $x_{n,k}(\epsilon, 0) = x_{n,k}$, and in Section 6 how to compute $x_{n,k}(\epsilon, m)$ for $m \geq 1$, and also $x_{n,k}(w, m)$ where $w = w'\sigma_j$ and $m \geq j - 1$, in polynomial time and space
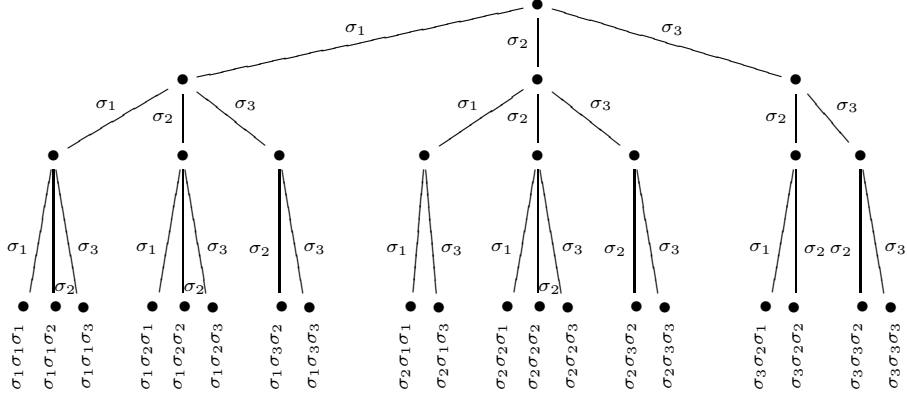
Figure 1: The rooted tree corresponding to $L_{4,3}$

with respect to n and k. If we assume this to be known, we can explain our main algorithm with some more detail:

Theorem 2.3. *Suppose the time and space required to compute $x_{n,k}(w, m)$, where $m \geq 0$ if $w = \epsilon$ and $m \geq j - 1$ if $w = w'\sigma_j$, is polynomial in n and k. Then there is an algorithm generating a random element of $\left(B_n^+\right)_k$, with uniform probability, whose time and space complexity is polynomial in n and k.*

Proof. In order to generate a random element of $\left(B_n^+\right)_k$, we compute $x_{n,k}$ and choose a random integer r between 1 and $x_{n,k}$. It just remains to determine the r-th element $w_{(r)}$ of the chain $L_{n,k}$ with respect to $<_{\text{lex}}$. We will find the word $w_{(r)}$ letter by letter. (Pseudocode is given in Algorithm 1.)

Suppose that we have computed a prefix w of $w_{(r)}$ and know that there are ν words in $L_{n,k}$ of the form ww', where $w' \in \mathcal{A}_n^*$ and $w_{(r)} <_{\text{lex}} ww'$. (Initially, $w = \epsilon$ and $\nu = x_{n,k} - r$.) If $w \neq \epsilon$, let σ_j be the last letter of w; otherwise let $j = 0$. Let σ_i be the next letter of $w_{(r)}$. Observe that $i \geq j - 1$, since otherwise $w_{(r)}$ would not be lexicographically minimal. By Definition 2.2, we have $m \geq i$ if and only if $x_{n,k}(w, m) \leq \nu$. Hence, we have $i = \min S$, where $S = \left\{m \in \{j - 1, \ldots, n - 1\} \mid x_{n,k}(w, m) \leq \nu\right\} \ni n - 1$. We can determine i using binary search in at most $\lceil \log_2(n - 1) \rceil$ steps, each step requiring the computation of one number of the form $x_{n,k}(w, m)$. We then replace w by $w\sigma_i$ and ν by $\nu - x_{n,k}(w, i)$, and proceed to the next letter of $w_{(r)}$. (While this does not affect the complexity, we remark that $x_{n,k}(w, i)$ is known from the binary search, so updating ν just requires one subtraction.)

As $w_{(r)}$ has length k, at most $k\lceil \log_2(n - 1) \rceil$ steps are required to find $w_{(r)}$; since each step is polynomial in n and k, the claim follows. $\square$

Example 2.4. *Suppose we want to generate a random braid of length 3 in B_4^+ (see Figure 1). Compute $x_{4,3} = 19$. Choose a random number r between 1 and*

4

19. Say $r = 16$, so $\nu = 3$. Now compute $x_{4,3}(\epsilon, 2) = 4 > 3 = \nu$. Thus, the first letter of $w_{(16)}$ is σ_3. Since $x_{4,3}(\epsilon, 3) = 0$, the value of ν remains unchanged.

Now we find the second letter of $w_{(16)}$, which must belong to the set $\{\sigma_2, \sigma_3\}$ (as the first letter is σ_3). We compute $x_{4,3}(\sigma_3, 2) = 2 \leq 3 = \nu$. Hence, the second letter of $w_{(16)}$ is σ_2, and we must replace ν by $\nu - x_{4,3}(\sigma_3, 2) = 1$.

Finally, as the second letter of $w_{(16)}$ is σ_2, the third one could be any letter in $\{\sigma_1, \sigma_2, \sigma_3\}$. We compute $x_{4,3}(\sigma_3\sigma_2, 2) = 0 \leq 1 = \nu$, then $x_{4,3}(\sigma_3\sigma_2, 1) = 1 \leq 1 = \nu$, and conclude that the third letter of $w_{(16)}$ is σ_1. Thus, $w_{(16)} = \sigma_3\sigma_2\sigma_1$.

Algorithm 1 gives pseudocode for finding $w_{(r)}$, assuming we know how to compute $x_{n,k}(w, m)$ as in the statement of Theorem 2.3. It contains references to two algorithms which will be introduced in Section 3 respectively Section 6. In Section 3 we will use a known result [2] to compute $x_{n,k}(\epsilon, 0) = x_{n,k}$ (line 1 of Algorithm 1). An algorithm for computing all other required instances of $x_{n,k}(w, m)$ (line 8 of Algorithm 1) will be given in Section 6.

Algorithm 1 Producing a uniformly random braid in B_n^+ of length k

---

Input: Integers $n \geq 2$ and $k \geq 0$.
Output: A braid in B_n^+ of length k.

1: Compute $x_{n,k}$. [Section 3; equations (2) and (3)]
2: Choose a random integer $r \in \{1, \ldots, x_{n,k}\}$ with uniform probability.
3: $w := \varepsilon$; $\nu := x_{n,k} - r$; $a := 0$
4: **for** $l := 1$ to k **do**
5:     $a := \max\{a - 1, 1\}$; $b := n - 1$; $\mu := 0$ /* $\mu = x_{n,k}(w, b)$ at all times */
6:     **while** $a < b$ **do**
7:         $m := \lfloor \frac{a+b}{2} \rfloor$
8:         Compute $x_{n,k}(w, m)$. [Algorithm 2]
9:         **if** $x_{n,k}(w, m) \leq \nu$ **then**
10:           $b := m$; $\mu := x_{n,k}(w, m)$
11:         **else**
12:           $a := m + 1$
13:     $w := w\sigma_a$; $\nu := \nu - \mu$ /* at this point $a = b$, hence $\mu = x_{n,k}(w, a)$ */
14: **return** $b(w)$

---

3   Counting all positive braids of a given length

In this section we use a formula given by Bronfman [2] to describe how to compute the number of elements of $(B_n^+)_k$ in time respectively space that is polynomial in n and k. To our knowledge, the paper [2] was never published, but one can find proofs of its main result in [1] and also in [7]. Bronfman gave a recurrence relation for the growth function of the monoid B_n^+. More precisely, if $x_{n,k}$ is the number of elements of length k in B_n^+, then the formal power series

$$G_n(t) = \sum_{k \geq 0} x_{n,k}\, t^k$$

is the growth function of B_n^+. Deligne [4] showed that this function was rational, namely the inverse of a polynomial $H_n(t)$. The following recurrence relation to compute this polynomial was given in [2]:

$$H_n(t) = \sum_{i=1}^{n} (-1)^{i+1} t^{\binom{i}{2}} H_{n-i}(t) \,, \tag{1}$$

where $H_0(t) = H_1(t) = 1$. In particular, $H_n(t)$ is a polynomial of degree $\binom{n}{2}$ such that $H_n(0) = 1$. Denoting the coefficient of t^m in $H_i(t)$ by $h_{i,m}$, it follows from $G_n(t) H_n(t) = 1$ that for any $j > 0$ one has

$$x_{n,j} = -\left(x_{n,j-1} h_{n,1} + x_{n,j-2} h_{n,2} + \cdots + x_{n,0} h_{n,j} \right) \,, \tag{2}$$

where the sum has at most $\min\{j, \binom{n}{2}\}$ terms, since $h_{n,m} = 0$ if $m > \binom{n}{2}$. Therefore, we can iteratively compute the numbers $x_{n,0}, x_{n,1}, \ldots, x_{n,k}$, knowing the coefficients $h_{n,1}, \ldots, h_{n,k}$. From (1), the latter can in turn be calculated with the recurrence relation

$$h_{m,j} = \sum_{i=1}^{m} (-1)^{i+1} h_{m-i,j-\binom{i}{2}} \,, \tag{3}$$

where we define $h_{m-i,j-\binom{i}{2}} = 0$ if $j - \binom{i}{2} < 0$. Note that in order to compute $x_{n,k}$, only the coefficients $h_{m,j}$ for $0 \leq m \leq n$ and $0 \leq j \leq k$ are required. In particular, time and space required to compute $x_{n,k}$ are polynomial in k and n.

The coefficients $h_{m,j}$ as well as the integers $x_{m,j}$ for $0 \leq m \leq n$ and $0 \leq j \leq k$ will be used later in our algorithm generating random elements of B_n^+. Note that coefficients $h_{m,j}$ that have already been computed do not change if n and k are changed; one merely might have to compute additional coefficients. Thus, any values of $h_{m,j}$ and $x_{m,j}$ that are computed can be stored once and for all; their computation should be thought of as a precomputation.

4 Forbidden prefixes

It only remains to compute $x_{n,k}(w, m)$, where $m \geq 1$ if $w = \epsilon$, and $m \geq i - 1$ if $w = w' \sigma_i$. (Recall Definition 2.2.) This calculation, which will be done in Section 6, relies on a detailed description of the words w' that can be appended to w so that the concatenation ww' is still a lex-representative in terms of what we call *forbidden prefixes*.

Denote by

$$L_n = \bigcup_{k \geq 0} L_{n,k} = \bigcup_{k \geq 0} \omega\big((B_n^+)_k\big) \subseteq \mathcal{A}_n^*$$

the set of words in $\{\sigma_1, \ldots, \sigma_{n-1}\}$ that are lex-representatives. To simplify notation, let $M = B_n^+$ and $M_k = (B_n^+)_k$ for $k \in \mathbb{N}$. We denote the length of a word $w \in \mathcal{A}_n^*$ by $|w|$, and the length of a braid $x \in M$ by $|x|$.

Definition 4.1. *For $w \in L_n$, we define the set of* **forbidden prefixes after** w *as $F_n(w) = \{\alpha \in M \mid w\omega(\alpha) \notin L_n\}$.*

For a braid $\alpha \in M$, we define the set of *multiples* of α as $\alpha M = \{\alpha\beta \mid \beta \in M\}$. The following is an important property of the sets of forbidden prefixes.

Lemma 4.2. *Let $w \in L_n$. If $\alpha \in F_n(w)$, then $\alpha M \subseteq F_n(w)$.*

Proof. For $u \in \mathcal{A}_n^*$ and $k \in \mathbb{N}$, let $u_{|_k}$ denote the initial subword of length k of u. As $\alpha \in F_n(w)$, we have $\omega(b(w)\alpha) <_{\text{lex}} w\omega(\alpha)$, and thus $\omega(b(w)\alpha)_{|_{|w|}} <_{\text{lex}} w$. For any $\beta \in M$, the latter implies $\omega(b(w)\alpha\beta) \leq_{\text{lex}} \omega(b(w)\alpha)\omega(\beta) <_{\text{lex}} w\omega(\alpha\beta)$, proving the claim. $\qquad\square$

As M is a cancellative monoid, it admits a well-defined partial order $\preccurlyeq$, where $a \preccurlyeq b$ if there exists $c \in M$ such that $ac = b$; in this case we say that a is a *prefix* of b. It is well known that $(M, \preccurlyeq)$ is Noetherian, that is, that there are no infinite descending chains with respect to $\preccurlyeq$ in M. We can then define a special subset of the forbidden prefixes.

Definition 4.3. *For $w \in L_n$, the set of* **minimal forbidden prefixes after** w, *denoted $F_n^{\min}(w)$, is the set of minimal elements, with respect to $\preccurlyeq$, in $F_n(w)$.*

From Lemma 4.2 and the Noetherianity of M, one has $F_n(w) = \bigcup_{\alpha \in F_n^{\min}(w)} \alpha M$.

The rest of this section will be devoted to the study of $F_n^{\min}(w)$. We will show that it is a finite set, we will describe its elements, and we will show how to compute them in linear time with respect to n and $|w|$.

First we recall a particularly useful property of the prefix order $\preccurlyeq$ of M: It is a lattice order [5], that is, for any $a, b \in M$ there are a greatest common prefix $a \wedge b$ and a least common multiple $a \vee b$ with respect to $\preccurlyeq$. In the particular case of Artin generators, we have $\sigma_i \vee \sigma_j = \sigma_i\sigma_j = \sigma_j\sigma_i$ if $|i - j| > 1$, and $\sigma_i \vee \sigma_j = \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j$ if $|i - j| = 1$.

Given two elements $a, b \in M$, there are positive elements $a\backslash b$ and $b\backslash a$, such that $a \vee b = a(a\backslash b) = b(b\backslash a)$. As the monoid M embeds in the group B_n, we can write $a\backslash b = a^{-1}(a \vee b)$ and $b\backslash a = b^{-1}(a \vee b)$. Notice that $\sigma_j\backslash\sigma_i = \sigma_i$ if $|i - j| > 1$ and $\sigma_j\backslash\sigma_i = \sigma_i\sigma_j$ if $|i - j| = 1$.

We shall frequently use the following property:

Lemma 4.4. *If $a, b, c \in M$, then $a \preccurlyeq bc$ is equivalent to $b\backslash a \preccurlyeq c$.*

Proof. As $b \preccurlyeq bc$, it follows that $a \preccurlyeq bc$ if and only if $a \vee b \preccurlyeq bc$; the latter is equivalent to $b^{-1}(a \vee b) \preccurlyeq c$, so the claim is shown. $\qquad\square$

The following two results form the basis for an inductive description of $F_n^{\min}(w)$.

Proposition 4.5.

(a) $F_n(\epsilon) = F_n^{\min}(\epsilon) = \emptyset$.

(b) $F_n^{\min}(\sigma_j) = \{\sigma_1, \ldots, \sigma_{j-2}, \sigma_{j-1}\sigma_j\}$.

(Here and in the sequel, we use the convention that words involving indices less than 1 are ignored. That is, $F_n^{\min}(\sigma_1) = \emptyset$ and $F_n^{\min}(\sigma_2) = \{\sigma_1\sigma_2\}$.)

Proof. It is obvious that $\{\sigma_1, \ldots, \sigma_{j-2}, \sigma_{j-1}\sigma_j\} \subseteq F_n(\sigma_j)$. Indeed, $\sigma_1, \ldots, \sigma_{j-2}$ are atoms and $\sigma_{j-1} \notin F_n(\sigma_j)$, so $\{\sigma_1, \ldots, \sigma_{j-2}, \sigma_{j-1}\sigma_j\} \subseteq F_n^{\min}(\sigma_j)$.

Conversely, if $\alpha \in F_n(\sigma_j)$ then $\omega(\sigma_j\alpha) <_{\mathrm{lex}} \sigma_j\omega(\alpha)$, and hence $\omega(\sigma_j\alpha)_{|_1} <_{\mathrm{lex}} \sigma_j$. This means that $\sigma_i \preccurlyeq \sigma_j\alpha$ for some $i < j$, whence α admits a prefix from the set $\{\sigma_j\backslash\sigma_i \mid i = 1, \ldots, j-1\} = \{\sigma_1, \ldots, \sigma_{j-2}, \sigma_{j-1}\sigma_j\}$.

This proves Claim (b). Claim (a) is trivial. $\square$

Proposition 4.6. *Let $w \in L_n$ with $|w| \geq 1$. If $w = v\sigma_j$, then $F_n^{\min}(w)$ is the set of $\preccurlyeq$-minimal elements in*

$$\{\sigma_1, \ldots, \sigma_{j-2}, \sigma_{j-1}\sigma_j\} \cup \{\sigma_j\backslash\beta \mid \beta \in F_n^{\min}(v)\}.$$

Proof. The case $|w| = 1$ is clear from Proposition 4.5, as $F_n^{\min}(\epsilon) = \emptyset$. Suppose that $|w| > 1$ and let α be a forbidden prefix after $w = v\sigma_j$, that is, $v\sigma_j\omega(\alpha) \notin L_n$. We will distinguish two cases: either $\sigma_j\omega(\alpha)$ is a lex-representative, or not.

If $\sigma_j\omega(\alpha)$ is not a lex-representative, then α is a forbidden prefix after σ_j. By Proposition 4.5, in this case α admits a prefix $\alpha' \in \{\sigma_1, \ldots, \sigma_{j-2}, \sigma_{j-1}\sigma_j\}$.

If $\sigma_j\omega(\alpha)$ is a lex-representative, then $\sigma_j\alpha$ is a forbidden prefix after v, hence there is $\beta \in F_n^{\min}(v)$ such that $\beta \preccurlyeq \sigma_j\alpha$, that is, $\sigma_j\backslash\beta \preccurlyeq \alpha$.

We have then shown that if α is a forbidden prefix after w, then α admits a prefix $\alpha' \in \{\sigma_1, \ldots, \sigma_{j-2}, \sigma_{j-1}\sigma_j\} \cup \{\sigma_j\backslash\beta \mid \beta \in F_n^{\min}(v)\}$. Since the elements of the latter set are forbidden prefixes after w by construction, the claim follows. $\square$

The following lemma will give us control over the elements of the form $\sigma_j\backslash\beta$.

Lemma 4.7. *For $i \in \{2, \ldots, n-1\}$ and $j \in \{1, \ldots, n-1\}$, one has:*

$$\sigma_j\backslash\sigma_{i-1}\sigma_i = \begin{cases} (\sigma_{i-1}\sigma_i)(\sigma_{i-2}\sigma_{i-1}) & \text{if } j = i-2 \\ \sigma_i & \text{if } j = i-1 \\ \sigma_{i-1}\sigma_i\sigma_{i+1} & \text{if } j = i+1 \\ \sigma_{i-1}\sigma_i & \text{otherwise} \end{cases}$$

For $j \in \{1, \ldots, n-1\}$ and $1 \leq m \leq i \leq n-1$, one has:

$$\sigma_j\backslash\sigma_i\sigma_{i-1}\cdots\sigma_m = \begin{cases} \sigma_i\sigma_{i-1}\cdots\sigma_m & \text{if } j \neq m-1, i, i+1 \\ \sigma_i\sigma_{i-1}\cdots\sigma_{m-1} & \text{if } j = m-1 \\ \sigma_{i-1}\sigma_{i-2}\cdots\sigma_m & \text{if } j = i \\ (\sigma_i\sigma_{i+1})(\sigma_{i-1}\sigma_i)\cdots(\sigma_m\sigma_{m+1}) & \text{if } j = i+1 \end{cases}$$

Proof. For $x \in M$ let $\pi_x \in \Sigma_n$ denote the permutation that x induces on the n strands. A so-called *permutation braid* is a braid in which any two strands

8

cross at most once. If x is a permutation braid, then x is uniquely determined by π_x. Moreover, one has $\sigma_i \preccurlyeq x$ if and only if $\pi_x(i) > \pi_x(i+1)$, that is, if and only if strands i and $i+1$ cross in x. In particular, the least common multiple of two permutation braids x and y can be computed easily from π_x and π_y [5]. All the braids occurring in the statement of the lemma are permutation braids and the claimed equalities are readily checked. $\square$

Example 4.8. *Consider $\sigma_4\sigma_3\sigma_2\sigma_2\sigma_1 \in B_5$. From Proposition 4.5 one has $F_n^{\min}(\sigma_4) = \{\sigma_1, \sigma_2, \sigma_3\sigma_4\}$. Using Proposition 4.6 and Lemma 4.7 repeatedly, one then obtains:*

$$F_n^{\min}(\sigma_4\sigma_3) = \min_{\preccurlyeq} \left\{\sigma_1, \sigma_2\sigma_3, \sigma_3\backslash\sigma_1, \sigma_3\backslash\sigma_2, \sigma_3\backslash\sigma_3\sigma_4\right\} = \{\sigma_1, \sigma_2\sigma_3, \sigma_4\}$$

$$F_n^{\min}(\sigma_4\sigma_3\sigma_2) = \min_{\preccurlyeq} \left\{\sigma_1\sigma_2, \sigma_2\backslash\sigma_1, \sigma_2\backslash\sigma_2\sigma_3, \sigma_2\backslash\sigma_4\right\} = \{\sigma_1\sigma_2, \sigma_3, \sigma_4\}$$

$$F_n^{\min}(\sigma_4\sigma_3\sigma_2\sigma_2) = \min_{\preccurlyeq} \left\{\sigma_1\sigma_2, \sigma_2\backslash\sigma_1\sigma_2, \sigma_2\backslash\sigma_3, \sigma_2\backslash\sigma_4\right\} = \{\sigma_1\sigma_2, \sigma_3\sigma_2, \sigma_4\}$$

$$F_n^{\min}(\sigma_4\sigma_3\sigma_2\sigma_2\sigma_1) = \min_{\preccurlyeq} \left\{\sigma_1\backslash\sigma_1\sigma_2, \sigma_1\backslash\sigma_3\sigma_2, \sigma_1\backslash\sigma_4\right\} = \{\sigma_2, \sigma_3\sigma_2\sigma_1, \sigma_4\}$$

Definition 4.9. Let $[n] = \{1, \ldots, n-1\}$.

(*a*) *A function $f : [n] \to \{-1, 0\} \cup [n]$ is called **admissible**, if it satisfies $f(i) \leq i$ for all $i \in [n]$, and $f(1) \neq -1$.*

(*b*) *For an admissible function f, we define*

$$F_f = \left\{\sigma_i\sigma_{i-1}\cdots\sigma_{f(i)} \mid i \in f^{-1}([n])\right\} \cup \left\{\sigma_{i-1}\sigma_i \mid i \in f^{-1}(-1)\right\} \subset M.$$

We now show that for any $w \in L_n$, there is an admissible function f_w such that $F_n^{\min}(w) = F_{f_w}$, and we express $f_{(w\sigma_j)}$, for $w\sigma_j \in L_n$, in terms of f_w.

Proposition 4.10.

(*a*) *$F_n^{\min}(\epsilon) = F_f$, where $f : [n] \to \{-1, 0\} \cup [n]$ given by $f(i) = 0$ for all i is admissible.*

(*b*) *$F_n^{\min}(\sigma_j) = F_{g_j}$, where $g_j : [n] \to \{-1, 0\} \cup [n]$ given by*

$$g_j(i) = \begin{cases} i & \text{if } 1 \leq i \leq j-2 \\ -1 & \text{if } i = j > 1 \\ 0 & \text{otherwise} \end{cases}$$

is admissible.

Proof. Claim (*a*) is trivial. Claim (*b*) is just a restatement of Proposition 4.5. $\square$

9

Proposition 4.11. *Let $w \in L_n$ and let f be admissible with $F_n^{\min}(w) = F_f$. Then $w\sigma_j \in L_n$ if and only if $f(j) \neq j$. Moreover, in this case, $F_n^{\min}(w\sigma_j) = F_g$, for the admissible function g given as follows:*

$$\text{for } i < j - 1: \quad g(i) = i$$

$$g(j-1) = \begin{cases} f(j) & \text{if } f(j) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$g(j) = \begin{cases} 0 & \text{if } j = 1 \text{ or } f(j) = j - 1 \\ -1 & \text{otherwise} \end{cases}$$

$$\text{for } i > j: \quad g(i) = \begin{cases} i & \text{if } f(i) = -1 \\ j & \text{if } f(i) = j + 1 \\ f(i) & \text{otherwise} \end{cases}$$

Proof. One has $w\sigma_j \in L_n$ if and only if $\sigma_j \notin F_n(w)$; the latter is the case if and only if $f(j) \neq j$, as σ_j is an atom. Now assume $w\sigma_j \in L_n$, and thus $f(j) \neq j$. Then it is clear that g is admissible, as $f(j) \leq j - 1$.

Recall from Proposition 4.6 that $F_n^{\min}(w\sigma_j)$ is the set of minimal elements in $\{\sigma_1, \ldots, \sigma_{j-2}, \sigma_{j-1}\sigma_j\} \cup \{\sigma_j \backslash \beta \mid \beta \in F_n^{\min}(w)\}$. This implies that $\sigma_1, \ldots, \sigma_{j-2}$ belong to $F_n^{\min}(w\sigma_j)$, as they are atoms, so $g(i) = i$ for $i < j - 1$. Also, $F_n^{\min}(w\sigma_j)$ contains $\sigma_{j-1}\sigma_j$, unless $j = 1$ or $\sigma_{j-1} \in F_n^{\min}(w\sigma_j)$. We have $\sigma_{j-1} \in F_n^{\min}(w\sigma_j)$ if and only if $\sigma_j \backslash \beta = \sigma_{j-1}$ for some $\beta \in F_n^{\min}(w)$. By Lemma 4.7, the latter is equivalent to $\beta = \sigma_j \sigma_{j-1}$, and hence to $f(j) = j - 1$.

It follows by induction on $|w|$ that one can have $f(i) = -1$ only if σ_i is the last letter of w. In that case, $\sigma_1, \ldots, \sigma_{i-2}$ are forbidden prefixes after w, hence $j \geq i - 1$. Therefore, if $i > j$ we can have $f(i) = -1$ only if $i = j + 1$. As $\sigma_j \backslash \sigma_j \sigma_{j+1} = \sigma_{j+1}$, this is why in this case we have $g(i) = i$.

The other values of g follow directly from Proposition 4.6, using the identities from Lemma 4.7 and discarding any elements which are not minimal (specifically, any multiples of $\sigma_1, \ldots, \sigma_{j-2}$ and $\sigma_{j-1}\sigma_j$). $\qquad\square$

Corollary 4.12. *If $w \in L_n$, then there exists a unique admissible function f_w, such that $F_n^{\min}(w) = F_{f_w}$. In particular, $|F_n^{\min}(w)| \leq n - 1$.*

Proof. If it exists, f_w is uniquely determined by $F_n^{\min}(w)$ and thus by w. The existence follows by induction on $|w|$, using Propositions 4.10 and 4.11. $\qquad\square$

Example 4.13. *Consider again $\sigma_4\sigma_3\sigma_2\sigma_2\sigma_1 \in B_5$. To shorten notation, we identify an admissible function f with the sequence $[f(1), \ldots, f(n-1)]$. From Proposition 4.10 one has $f_{\sigma_4} = [1, 2, 0, -1]$. Repeated application of Proposition 4.11 then yields $f_{\sigma_4\sigma_3} = [1, 0, -1, 4]$, $f_{\sigma_4\sigma_3\sigma_2} = [0, -1, 3, 4]$, $f_{\sigma_4\sigma_3\sigma_2\sigma_2} = [0, -1, 2, 4]$, and $f_{\sigma_4\sigma_3\sigma_2\sigma_2\sigma_1} = [0, 2, 1, 4]$. Observe that the sets of minimal*

forbidden prefixes described by these functions are exactly those computed in Example 4.8.

Given $w \in L_n$, the set $F_n^{\min}(w)$ can be computed efficiently using Propositions 4.10 and 4.11; the time required is obviously linear in n and $|w|$.

There are many admissible functions, but the ones corresponding to minimal sets of forbidden prefixes are very special: It can be shown that they are exactly those satisfying the conditions in the following corollary. However, proving that these conditions are sufficient is quite technical; as we do not use this fact in the sequel, we only show that they are necessary.

Corollary 4.14. *Let $w = w'\sigma_j \in L_n$. Let $i_1 < i_2 < \cdots < i_r$ be all the indices greater than j such that $0 < f_w(i_t) < i_t$ $(t = 1, \ldots, r)$. Denote $m = f_w(j-1)$.*

(a) $f_w(i) = i$ for $i < j - 1$.

(b) $f_w(i) \in \{0, \ldots, i\}$ for all $i \neq j$.

(c) $f_w(j) = \begin{cases} 0 & \text{if either } j = 1 \text{ or } f_w(j-1) = j - 1 \\ -1 & \text{otherwise} \end{cases}$

(d) If $f_w(i) = 0$ for some $i > j$, then $f_w(\ell) = 0$ for $\ell = i, \ldots, n - 1$.

(e) If $r \geq 1$, then $0 < f_w(i_r) \leq \cdots \leq f_w(i_1) \leq j$.

(f) If $r \geq 1$ and $m > 0$, then either $f_w(i_1) = j$ or $f_w(i_1) \leq m$.

(g) If $r \geq 2$ and $m > 0$, then $0 < f_w(i_r) \leq \cdots \leq f_w(i_2) \leq m$.

Proof. If $w' = \epsilon$, that is $w = \sigma_j$, the results holds trivially by Proposition 4.10, so let $w' = w''\sigma_k$ and assume the result for $f_{w'}$. To shorten notation, let $f = f_{w'}$ and $g = f_w$. Since $w'\sigma_j \in L_n$, we have $f(j) \neq j$ and thus $j \geq k - 1$. Proposition 4.11 immediately yields Claims (a), (b) and (c).

For Claim (d) assume that $g(i) = 0$ for some $i > j$. By Proposition 4.11 this happens only if $g(i) = f(i) = 0$, and if $i > k$ this implies that $g(r) = f(r) = 0$ for $r = i, \ldots, n - 1$. As $i > j \geq k - 1$, the only remaining case is $i = k = j + 1$ and $f(k) = 0$. In this case Claim (c) applied to f gives $f(k-1) = k - 1$, that is $f(j) = j$, which is a contradiction. Thus Claim (d) holds.

Now suppose $r \geq 1$. We have $i_1 > j \geq k - 1$, and $i_1 = k$ would imply $f(i_1) \in \{0, -1\}$ and then, with Proposition 4.11, $g(i_1) \in \{0, i_1\}$; the latter is a contradiction, so $k < i_1 < \cdots < i_r$. For $t = 1, \ldots, r$, Proposition 4.11 also yields that either $g(i_t) = f(i_t)$, or $f(i_t) = j + 1$ and $g(i_t) = j$. Since $f(i_t) \leq i_t$ and $j < i_1 < i_2 < \cdots < i_r$, the latter implies that either $0 < f(i_t) < i_t$, or $t = 1$ and $i_1 = j + 1 = f(i_1) = g(i_1) + 1$. In any case, applying Claim (e) to f and using $k \leq j + 1$, we obtain $0 < f(i_r) \leq \cdots \leq f(i_1) \leq j + 1$. Again using Proposition 4.11 then yields $0 < g(i_r) \leq \cdots \leq g(i_1) \leq j$, showing Claim (e).

Suppose $m = g(j-1) > 0$. If $i_1 = j + 1 = f(i_1) = g(i_1) + 1$, then Claim (f) holds. By the preceding paragraph, we may thus assume that $0 < f(i_t) < i_t$

11

for $1 \le t \le r$. By Proposition 4.11, $m > 0$ implies $f(j) > 0$ and $m = f(j)$. In particular, $j \ne k$ (as $f(k) \in \{0, -1\}$), whence we have either $j = k - 1$ or $j > k$. In the former case, $m = f(k-1)$, so we can apply the result to f and obtain that $f(i_t) \le m$ for $2 \le t \le r$, and either $f(i_1) = k = j + 1$ or $f(i_1) \le m$; as $m \le j + 1$, Proposition 4.11 then yields $g(i_t) = f(i_t) \le m$ for $2 \le t \le r$, and either $g(i_1) = j$ or $g(i_1) = f(i_1) \le m$, showing Claims (f) and (g) in this case. On the other hand, if $j > k$ we just need to notice that $0 < m = f(j) < j$ and apply Claim (e) to f, which yields $f(i_t) \le f(j) = m < j$, whence $g(i_t) = f(i_t) \le m$. So Claims (f) and (g) hold. $\qquad\square$

One can show that the conditions of Corollary 4.14 are sufficient for f to be the defining function of some $F_n^{\min}(w)$ by constructing, for every f satisfying the conditions, a word w_f such that $F_n^{\min}(w_f) = F_f$. As this construction for general f is very technical, we do not describe it here. Instead, we only consider some special cases; enough to show that the number of possible sets $F_n^{\min}(w)$ is exponential in n.

Proposition 4.15. *For* $1 \le i, j \le n - 1$ *define* $[i, j] \in \mathcal{A}^*$ *as the product* $\sigma_i \sigma_{i \pm 1} \cdots \sigma_j$. *To shorten notation, we identify an admissible function* f *with the sequence* $\big[f(1), \ldots, f(n-1)\big]$.

(a) One has $[n - 1, 1] \in L_n$ *and*

$$f_{[n-1,j]} = \begin{cases} [1, \ldots, j-2, 0, -1, j+1, \ldots, n-1] & \text{if } n - 1 \ge j > 1 \\ [0, 2, \ldots, n-1] & \text{if } j = 1 \end{cases}$$

(b) If $i \in [n]$ *and* $w \in L_n$ *such that* $f_w = \big[0, 2, 3, \ldots, i, m_{i+1}, \ldots, m_{n-1}\big]$, *with* $m_k \in \{1, k\}$ *for* $k = i + 1, \ldots, n - 1$, *then* $w \cdot [1, i - 1] \in L_n$ *and*

$$f_{w \cdot [1,j]} = \begin{cases} [1, \ldots, j-1, 0, j, j+2, \ldots, i, m_{i+1}, \ldots, m_{n-1}] & \text{if } 1 \le j < i - 1 \\ [1, \ldots, i-2, 0, i-1, m_{i+1}, \ldots, m_{n-1}] & \text{if } j = i - 1 \end{cases}$$

(c) If $i \in [n]$ *and* $w \in L_n$ *such that* $f_w = \big[1, \ldots, i-2, 0, i-1, m_{i+1}, \ldots, m_{n-1}\big]$, *with* $m_k \in \{1, k\}$ *for* $k = i + 1, \ldots, n - 1$, *then* $w \cdot [i - 1, 1] \in L_n$ *and*

$$f_{w \cdot [i-1,j]} = \begin{cases} [1, \ldots, j-2, 0, -1, j+1, \ldots, i-1, j, m_{i+1}, \ldots, m_{n-1}] \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } i - 1 \ge j > 1 \\ [0, 2, 3, \ldots, i-1, 1, m_{i+1}, \ldots, m_{n-1}] \qquad \text{if } j = 1 \end{cases}$$

Proof. The claims easily follow by induction on j, using Proposition 4.11. $\qquad\square$

Corollary 4.16. *If* $S = \{i_1, \ldots, i_r\}$ *is a subset of* $\{1, \ldots, n-2\}$, *possibly empty, with* $i_1 > \cdots > i_r$, *then* $w_S = [n-1, 1] \cdot [1, i_1] \cdot [i_1, 1] \cdots \cdot [1, i_r] \cdot [i_r, 1] \in L_n$, *and*

$$f_{w_S}(j) = \begin{cases} 0 & \text{if } j = 1 \\ 1 & \text{if } j - 1 \in S \\ j & \text{otherwise} . \end{cases}$$

12

Proof. The claim follows from Proposition 4.15 by induction on r. $\qquad\square$

Corollary 4.17. *Let $n > 1$. The set $\{F_n^{\min}(w) \mid w \in L_n\}$ is finite, but it has at least 2^{n-2} elements.*

Proof. The set is finite as each $F_n^{\min}(w)$ is equal to F_f for some admissible function f. It has at least 2^{n-2} elements by Corollary 4.16 $\qquad\square$

Before using, in Section 6, our description of forbidden prefixes to compute the numbers $x_{n,k}(w, m)$ required by Algorithm 1, we show in the next section that minimal forbidden prefixes yield a finite state automaton accepting the language of lex-representative words, that has the minimal possible number of states.

5 A minimal finite state automaton accepting L_n

The sets of minimal forbidden prefixes after a given word $w \in L_n$ provide a very natural way to construct a finite state automaton that accepts the language L_n of lex-representatives of braids in M. Indeed, we will see that this finite state automaton is minimal, in the sense that it has the minimal possible number of states.

Recall that a finite state automaton is a quintuple $\Gamma = (SS, \mathcal{A}, \mu, Y, S_0)$ where SS is a finite set, $\mathcal{A}$ is the alphabet, $\mu: SS \times \mathcal{A} \to SS$ is the transition function, $Y \subseteq SS$ is the set of accepted states and $S_0 \in SS$ is the initial state [5]. We extend μ to a function $SS \times \mathcal{A}^* \to SS$, also denoted by μ, in the natural way.

Definition 5.1. *Denote $\Gamma_n = (SS, \mathcal{A}, \mu, Y, S_0)$, where $\mathcal{A} = \{\sigma_1, \ldots, \sigma_{n-1}\}$, $SS = \{S \subset \mathcal{A}^* \mid S = F_n^{\min}(w) \text{ for some } w \in L_n\} \cup \{\mathcal{A}\}$, $S_0 = \emptyset = F_n^{\min}(\epsilon)$, $Y = SS\backslash\{\mathcal{A}\}$, and $\mu: SS \times \mathcal{A} \to SS$ is given by*

$$(S, \sigma_i) \mapsto \begin{cases} \mathcal{A} & \text{if } \sigma_i \in S \\ \min_{\preccurlyeq}\left(\{\sigma_1, \ldots, \sigma_{i-2}, \sigma_{i-1}\sigma_i\} \cup \{\sigma_i\backslash\beta \mid \beta \in S\}\right) & \text{otherwise}. \end{cases}$$

Proposition 5.2. *Γ_n is a finite state automaton accepting the language L_n. Moreover, any finite state automaton accepting the language L_n has at least as many states as Γ_n.*

Proof. By Corollary 4.12, every state $S = F_n^{\min}(w) \in SS$ is uniquely determined by an admissible function. As the number of admissible functions is finite, Γ_n is indeed a finite state automaton.

If $w \in L_n$, then it follows from Proposition 4.6 by induction on $|w|$ that $\mu(S_0, w) = F_n^{\min}(w) \in Y$, so w is accepted. Conversely, if $w \notin L_n$, let $w = v\sigma_i v'$, where v is the initial subword of maximal length of w that lies in L_n. As above, we have $\mu(S_0, v) = F_n^{\min}(v)$. Moreover, since $v\sigma_i \notin L_n$, we have $\sigma_i \in F_n(v)$ and thus $\sigma_i \in F_n^{\min}(v) = \mu(S_0, v)$, as σ_i is an atom. Then, by the definition of μ, we have $\mu(S_0, v\sigma_i) = \mathcal{A}$, and further (by induction on $|v'|$) $\mu(S_0, w) = \mu(S_0, v\sigma_i v') = \mathcal{A}$, so w is not accepted.

13

In order to show that Γ_n has the minimal possible number of states, assume that Γ' is a finite state automaton accepting the language L_n. Let $w_1, w_2 \in L_n$ be such that $F_n^{\min}(w_1) \neq F_n^{\min}(w_2)$. This implies $F_n(w_1) \neq F_n(w_2)$, so by symmetry, assume that there exists some $\alpha \in F_n(w_1) \backslash F_n(w_2)$, and consider the word $w = \omega(\alpha) \in L_n$. As α is forbidden after w_1, we have $w_1 w \notin L_n$, so reading w starting in the state corresponding to w_1 one ends at a fail state. However, as α is not forbidden after w_2, one has $w_2 w \in L_n$, so reading w starting in the state corresponding to w_2 one ends at an accepted state; in particular, the states corresponding to w_1 and w_2 in Γ' must be distinct. As Γ' has at least one fail state, the number of states of Γ' is at least the number of states of Γ_n. $\square$

Example 5.3. *For $n = 3$, the automaton Γ_n has 5 accepted states; it is represented in Figure 2. Recall that the initial state corresponds to the empty set, and notice that we did not represent the fail state: as usual, arrows which are not drawn lead to the fail state $\{\sigma_1, \sigma_2\}$.*
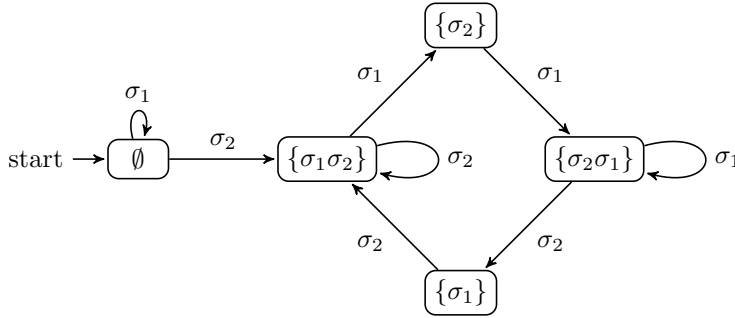

Figure 2: Finite state automaton accepting lex-representatives in B_3^+.

Example 5.4. *For $n = 4$, the automaton Γ_n has 18 accepted states; it is represented in Figure 3. As above, the initial state corresponds to the empty set, and we did not represent the fail state $\{\sigma_1, \sigma_2, \sigma_3\}$.*

Proposition 5.5. *Any finite state automaton accepting the language L_n has at least 2^{n-2} states.*

Proof. By Proposition 5.2 it is sufficient to show that Γ_n has at least 2^{n-2} states. The latter follows from Corollary 4.17. $\square$

Remark 5.6. We mentioned that the admissible functions that correspond to sets of minimal forbidden prefixes are precisely those satisfying the conditions of Corollary 4.14. This allows to give an exact expression for the number of states of Γ_n. The arguments are extremely technical, however, and as we do not use this result in the sequel, we skip the details here.

Just to give an idea, the following table contains the number of accepted states of Γ_n, for $n = 3, \ldots, 20$.
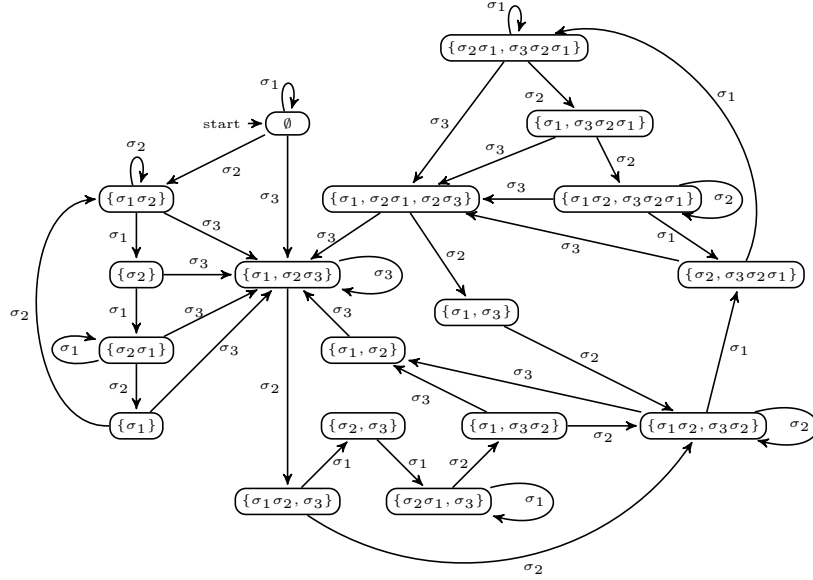
Figure 3: Finite state automaton accepting lex-representatives in B_4^+.

n	3	4	5	6	7	8	9	10	11	12	13
$|Y|$	5	18	56	161	443	1190	3156	8315	21835	57246	149970

n	14	15	16	17	18	19	20
$|Y|$	392743	1028351	2692416	7049018	18454775	48315461	126491780

Remark 5.7. There are generic methods to generate uniformly random words of a regular language, such as the *recursive method* [10, 6]. These algorithms have a precomputation phase, in which a (minimal) acceptor for the language in question is computed.

The results of this section show that the number of states of a minimal acceptor for L_n is exponential in n. Thus, generating uniformly random words of L_n by generic language-theoretic methods has a time complexity respectively a space complexity that is exponential in n. While efficient for small values of n, such approaches are not feasible for larger values of n, as the above figures show.

6 Counting braids with suitable prefixes

In this section we finally give a method to compute $x_{n,k}(w, m)$, where $m \geq 1$ if $w = \epsilon$, and $m \geq j - 1$ if $w = w'\sigma_j$. We assume that we have already computed $h_{s,t}$ for $0 \leq s \leq n$ and $0 \leq t \leq k$, as well as $x_{n,s}$ for $0 \leq s \leq k$. (See Section 3.) Recall from Definition 2.2 that $x_{n,k}(w, m)$ is the number of lex-representatives of length k of the form wv where $v \in L_n$ does not start with σ_i for $i \leq m$.

15

As the relations of M are homogeneous, this amounts to counting the braids x of length $k - |w|$ for which $w\omega(x) \in L_n$ and $\omega(x)$ does not start with σ_i for $i \leq m$. Knowing the number of braids x of length $k - |w|$, it suffices to count those braids x of length $k - |w|$ for which $w\omega(x) \notin L_n$, or $\omega(x)$ starts with σ_i for $i \leq m$. If $m = 0$, the second condition is never satisfied and we simply need to count $F_n(w) \cap M_{k-|w|}$. If $m > 0$, however, we have to add $\sigma_1, \ldots, \sigma_m$ to the set of forbidden prefixes.

Definition 6.1. Let $w \in L_n$ and $m \in \{0, \ldots, n-1\}$. We define $F_n^{\min}(w, m)$ to be the set of minimal elements, with respect to $\preccurlyeq$, in $\{\sigma_1, \ldots, \sigma_m\} \cup F_n^{\min}(w)$.

Notice that $F_n^{\min}(w, m)$ is obtained from $F_n^{\min}(w)$ by removing all elements that start with $\sigma_1, \ldots, \sigma_m$, and adding the elements $\sigma_1, \ldots, \sigma_m$. If $m \geq j - 1$, where σ_j is the last letter of w, this simplifies the description of the set $F_n^{\min}(w, m)$:

Proposition 6.2. Let $w \in L_n$, let $j = 1$ if $w = \epsilon$, or $w = w'\sigma_j$ otherwise, and let $m \geq j - 1$. There is an admissible function f such that $F_n^{\min}(w, m) = F_f$. Moreover, the following hold:

(a) $f(r) = r$ for $r = 1, \ldots, m$ and $f(r) \geq 0$ for $r = m + 1, \ldots, n - 1$.

(b) $f(j) = 0$ if $m = j - 1$ and $f(j) = j$ if $m \geq j$.

(c) If $i \in \{j + 1, \ldots, n - 1\}$ and $f(i) = 0$, then $f(r) = 0$ for $r = i, \ldots, n - 1$.

(d) If $i \in [n]$ and $0 < f(i) < i$, then $f(i) \leq j$.

(e) If $i, i' \in [n]$, such that $0 < f(i) < i$ and $0 < f(i') < i'$, then $i' > i$ implies $f(i) \geq f(i')$.

In particular, all elements of $F_n^{\min}(w, m)$ are of the form $\sigma_r \sigma_{r-1} \cdots \sigma_{f(r)}$ with $f(r) \leq r$, and those elements that are longer than one letter are nested: If $f(i_t) < i_t$ $(1 \leq t \leq s)$ for $j < i_1 < \cdots < i_s$ then $j \geq f(i_1) \geq f(i_2) \geq \cdots \geq f(i_s)$.

Proof. This follows immediately from Corollary 4.14 together with the fact that $\sigma_i \in F_n^{\min}(w, m)$ for $i = 1, \ldots, m$. $\qquad\square$

We are now in a position to calculate $x_{n,k}(w, m)$. For a non-empty finite set $S = \{a_1, \ldots, a_r\} \subset M$, we define $\bigvee S = a_1 \vee \cdots \vee a_r$. We also define $\bigvee \emptyset = 1$. Recall that $M_k = \{x \in M \mid |x| = k\}$; so $x_{n,k} = |M_k|$ and $x_{n,k} = 0$ if $k < 0$.

Proposition 6.3. Let $w \in L_n$ and $m \in \{1, \ldots, n-1\}$. One has

$$x_{n,k}(w, m) = \sum_{S \subseteq F_n^{\min}(w,m)} (-1)^{|S|} x_{n,k-|w|-|\bigvee S|} = \sum_{l=0}^{k-|w|} \left(x_{n,k-|w|-l} \cdot T_l(w, m) \right),$$

where

$$T_l(w, m) = \sum_{\substack{S \subseteq F_n^{\min}(w,m) \\ |\bigvee S| = l}} (-1)^{|S|}.$$

16

Proof. The second equality is just a reordering of terms. For $\alpha \in M$, let $(\alpha M)_k = \alpha M \cap M_k$. As M is a cancellative monoid with homogeneous relations $(\alpha M)_k = \alpha M_{k-|\alpha|} = \{\alpha\beta \mid \beta \in M_{k-|\alpha|}\}$, that is, $|(\alpha M)_k| = x_{n,k-|\alpha|}$. The braids x for which $w\omega(x) \notin L_n$ or $\omega(x)$ starts with σ_i for $i \leq m$ are exactly those that admit a prefix $\alpha \in F_n^{\min}(w, m)$. Moreover, $|w\omega(x)| = |w| + |x|$. Hence

$$x_{n,k}(w,m) \;\; = \;\; x_{n,k-|w|} \;\; - \;\; \left| \bigcup_{\alpha \in F_n^{\min}(w,m)} (\alpha M)_{k-|w|} \right| .$$

By the inclusion-exclusion principle,

$$\left| \bigcup_{\alpha \in F_n^{\min}(w,m)} (\alpha M)_{k-|w|} \right| \;\; = \;\; \sum_{\emptyset \neq S \subseteq F_n^{\min}(w,m)} (-1)^{|S|-1} \left| \bigcap_{\alpha \in S} (\alpha M)_{k-|w|} \right| .$$

Noting that $\bigvee \emptyset = 1 \in M$, whence $|\bigvee \emptyset| = 0$, the claim then follows from

$$\left| \bigcap_{\alpha \in S} (\alpha M)_{k-|w|} \right| \;\; = \;\; \left| \left(\bigcap_{\alpha \in S} \alpha M \right) \cap M_{k-|w|} \right| \;\; = \;\; \left| ((\bigvee S) M) \cap M_{k-|w|} \right|$$

$$= \;\; \left| (\bigvee S) M_{k-|w|-|\bigvee S|} \right| \;\; = \;\; x_{n,k-|w|-|\bigvee S|} . \qquad \square$$

In order to obtain a sub-exponential algorithm, we need to compute $T_l(w, m)$ without explicitly summing over all subsets of $F_n^{\min}(w, m)$. To this end, we use some special properties of the braid monoid. By Proposition 6.2, the elements of $F_n^{\min}(w, m)$ are permutation braids, thus so is the least common multiple of any collection of them. The crucial point is that, under certain conditions, we only need to know the final position of some strands in a permutation braid, in order to describe sufficiently its least common multiple with another permutation braid. Specifically, we have the following lemma, which can be readily verified working with the induced permutations [5].

Lemma 6.4. *Let* $x \in \langle \sigma_a, \sigma_{a+1}, \ldots, \sigma_{b-1} \rangle^+ \subset M$ *be a permutation braid and let* π_x *be the permutation induced by* x. *Let* $\pi_x(a) = a + r$ *and* $\pi_x(b) = b - s$.

(a) If $a > 1$, *then* $x' = \sigma_{a-1} \vee x = x(\sigma_{a-1}\sigma_a \cdots \sigma_{a-1+r})$. *Moreover,*

$$\pi_{x'}(a-1) = a + r \qquad and \qquad \pi_{x'}(b) = \begin{cases} b - s & \text{if } a + r < b - s \\ b - s - 1 & \text{if } a + r > b - s \end{cases} .$$

(b) If $b < n$, *then* $x' = \sigma_b \vee x = x(\sigma_b\sigma_{b-1} \cdots \sigma_{b-s})$. *Moreover,*

$$\pi_{x'}(a) = \begin{cases} a + r & \text{if } a + r < b - s \\ a + r + 1 & \text{if } a + r > b - s \end{cases} \qquad and \qquad \pi_{x'}(b+1) = b - s .$$

(c) If $b < n$ *then* $x' = (\sigma_b\sigma_{b-1} \cdots \sigma_a) \vee x = x(\sigma_b\sigma_{b-1} \cdots \sigma_a)$. *Moreover,*

$$\pi_{x'}(a) = a + r + 1 \qquad and \qquad \pi_{x'}(b+1) = a .$$

17

Evaluating $T_l(w, m)$

Let $w \in L_n$, let $j = 1$ if $w = \epsilon$, or $w = w'\sigma_j$ otherwise, and let $m \geq j - 1$.

We work with a 3-dimensional array T of size $(k - |w| + 1) \times n \times n$, which is transformed in at most $n - 1$ steps. At each step, we have two distinguished positions a and b with $1 \leq a \leq b \leq n$, defining $S_{a,b} = F_n^{\min}(w, m) \cap \langle \sigma_a, \sigma_{a+1}, \ldots, \sigma_{b-1} \rangle^+$, and for $l \in \{0, \ldots, k - |w|\}$ and $r, s \in \{0, \ldots, n - 1\}$, the entry $T_{l,r,s}$ of T is

$$T_{l,r,s} = \sum_{\substack{S \subseteq S_{a,b}, \, |\bigvee S| = l \\ \pi_{(\bigvee S)}(a) = a + r, \, \pi_{(\bigvee S)}(b) = b - s}} (-1)^{|S|} .$$

Initially, we take $a = b = j$, so $S_{a,b} = \emptyset$. Obviously, in this case all entries of T are 0, except $T_{0,0,0} = 1$, which corresponds to $\bigvee \emptyset = 1$.

At the final step, we will have $a = 1$ and $b = n$, so $S_{a,b} = F_n^{\min}(w, m)$, whence

$$T_l(w, m) = \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} T_{l,r,s} .$$

In each step we replace (a, b) by $(a', b') \in \{(a - 1, b), (a, b + 1)\}$: If there is some $\sigma_b \sigma_{b-1} \cdots \sigma_s \in F_n^{\min}(w, m)$ with $s < a$ or if $b = n$ then we set $(a', b') = (a - 1, b)$, otherwise we set $(a', b') = (a, b + 1)$. By Proposition 6.2, either $S_{a',b'} \backslash S_{a,b} = \emptyset$, or $S_{a',b'} \backslash S_{a,b} = \{\sigma_b\}$, or $S_{a',b'} \backslash S_{a,b} = \{\sigma_b \sigma_{b-1} \cdots \sigma_a\}$, or $S_{a',b'} \backslash S_{a,b} = \{\sigma_{a-1}\}$. Hence, if $S_{a',b'} \backslash S_{a,b} = \{x\} \neq \emptyset$ and $y \in S_{a,b}$, then according to Lemma 6.4, the length, the displacement of strand a', and the displacement of strand b' of the element $x \vee y$ only depend on x, and the length, the displacement of strand a, and the displacement of strand b of the element y. Therefore, the table T can be updated when replacing (a, b) by (a', b'): The elements contributing to $T_{l,r,s}$ in the old table will in the updated table contribute to $T_{l,0,s}$ if $(a', b') = (a - 1, b)$, respectively to $T_{l,r,0}$ if $(a', b') = (a, b + 1)$. If $S_{a',b'} \backslash S_{a,b} \neq \emptyset$, then in addition, their least common multiples with x will contribute to the entry of the updated table given by Lemma 6.4.

The arguments from this section yield Algorithm 2 computing $x_{n,k}(w, m)$.

7 Complexity and timing analysis

The analysis of the worst-case complexity of Algorithm 1 is relatively straightforward. We assume that the addition of two N-bit integers has cost $O(N)$, and that the multiplication of two N-bit integers, in the relevant range of N, has cost $O(N^\alpha)$ with $\alpha = \log_2 3 \approx 1.585$ (Karatsuba multiplication).

Proposition 7.1.

(a) *The worst-case complexity of Algorithm 2 is at most $O(n^4 k^{\alpha+1})$.*

(b) *The worst-case complexity of Algorithm 1 is at most $O(n^4 \ln n \, k^{\alpha+2})$.*

18

Algorithm 2 Computing $x_{n,k}(w,m)$

Input: Integers $n \geq 2$ and $k \geq 0$. A word $w = \sigma_{a_1} \cdots \sigma_{a_t} \in \mathcal{A}_n^*$. An integer $m \leq n-1$,
  where $m \geq 1$ if $w = \epsilon$ and $m \geq j - 1$ if $w = v\sigma_j$. The numbers $x_{n,0}, \ldots, x_{n,k}$.
Output: The number $x_{n,k}(w,m)$.

1: **if** $t > k$ **then**
2:   **return** 0
3: $F := \emptyset$
4: **for** $i := 1$ to t **do**
5:   **if** $\sigma_{a_i} \in F$ **then**
6:     **return** 0
7:   **else**
8:     $F := \min_{\preccurlyeq} \left(\{\sigma_1, \ldots, \sigma_{a_i-2}, \sigma_{a_i-1}\sigma_{a_i}\} \cup \{\sigma_{a_i} \backslash \beta \mid \beta \in F\} \right)$
9: $F := \min_{\preccurlyeq} \left(\{\sigma_1, \ldots, \sigma_m\} \cup F \right)$
10: $T_{l,r,s} := 0$ for all $(l,r,s) \in I = \{0, \ldots, k-t\} \times \{0, \ldots, n-1\} \times \{0, \ldots, n-1\}$
11: **if** $t = 0$ **then**
12:   $a := 1$; $b := 1$; $\alpha := 0$; $T_{0,0,0} := 1$
13: **else**
14:   $a := a_t$; $b := a_t$; $\alpha := 0$; $T_{0,0,0} := 1$
15: **while** $(a,b) \neq (1,n)$ **do**
16:   **if** $b = n$ or $\sigma_b \sigma_{b-1} \cdots \sigma_s \in F$ for some $s < a$ **then**
17:     $(a',b') := (a-1, b)$; $\alpha := \alpha + 1$
18:     **if** $F \cap \langle \sigma_{a'}, \ldots, \sigma_{b'-1} \rangle^+ = F \cap \langle \sigma_a, \ldots, \sigma_{b-1} \rangle^+$ **then**
19:       **for** $(l,r,s) \in I$ **do** $T'_{l,r,s} := \begin{cases} \sum_{u=0}^{b-a} T_{l,u,s} & \text{if } r = 0 \\ 0 & \text{otherwise} \end{cases}$
20:     **else if** $(F \cap \langle \sigma_{a'}, \ldots, \sigma_{b'-1} \rangle^+) \backslash (F \cap \langle \sigma_a, \ldots, \sigma_{b-1} \rangle^+) = \{\sigma_{a-1}\}$ **then**
21:       **for** $(l,r,s) \in I$ **do** $T'_{l,r,s} := \begin{cases} \sum_{u=0}^{b-a} T_{l,u,s} & \text{if } r = 0 \\ -T_{l-r,r-1,s} & \text{if } 1 \leq r \leq l \text{ and } r + s < \alpha \\ -T_{l-r,r-1,s-1} & \text{if } 1 \leq r \leq l \text{ and } r + s > \alpha \\ 0 & \text{otherwise} \end{cases}$
22:   **else**
23:     $(a',b') := (a, b+1)$; $\alpha := \alpha + 1$
24:     **if** $F \cap \langle \sigma_{a'}, \ldots, \sigma_{b'-1} \rangle^+ = F \cap \langle \sigma_a, \ldots, \sigma_{b-1} \rangle^+$ **then**
25:       **for** $(l,r,s) \in I$ **do** $T'_{l,r,s} := \begin{cases} \sum_{u=0}^{b-a} T_{l,r,u} & \text{if } s = 0 \\ 0 & \text{otherwise} \end{cases}$
26:     **else if** $(F \cap \langle \sigma_{a'}, \ldots, \sigma_{b'-1} \rangle^+) \backslash (F \cap \langle \sigma_a, \ldots, \sigma_{b-1} \rangle^+) = \{\sigma_b\}$ **then**
27:       **for** $(l,r,s) \in I$ **do** $T'_{l,r,s} := \begin{cases} \sum_{u=0}^{b-a} T_{l,r,u} & \text{if } s = 0 \\ -T_{l-s,r,s-1} & \text{if } 1 \leq s \leq l \text{ and } r + s < \alpha \\ -T_{l-s,r-1,s-1} & \text{if } 1 \leq s \leq l \text{ and } r + s > \alpha \\ 0 & \text{otherwise} \end{cases}$
28:     **else if** $(F \cap \langle \sigma_{a'}, \ldots, \sigma_{b'-1} \rangle^+) \backslash (F \cap \langle \sigma_a, \ldots, \sigma_{b-1} \rangle^+) = \{\sigma_b \cdots \sigma_a\}$ **then**
29:       **for** $(l,r,s) \in I$ **do** $T'_{l,r,s} := \begin{cases} \sum_{u=0}^{b-a} T_{l,r,u} & \text{if } s = 0 \\ -\sum_{u=0}^{b-a} T_{l-\alpha,r-1,u} & \text{if } s = \alpha, l \geq \alpha, r \geq 1 \\ 0 & \text{otherwise} \end{cases}$
30:   $(a,b) := (a',b')$; $T := T'$
31: **return** $\sum_{l=0}^{k-t} \left(\sum_{r=0}^{n-1} \sum_{s=0}^{n-1} T_{l,r,s} \right) x_{n,k-t-l}$

Proof. The absolute values of all entries of the array T are bounded by 2^n, so the entries of T have at most n bits. As it is known [9, Theorem 10] that the logarithmic volume $\lim_{k \to \infty} \frac{\log x_{n,k}}{k}$ of B_n^+ is bounded above by 4, the numbers $x_{n,j}$ for $0 \leq j \leq k$ have $O(k)$ bits.

Consider Algorithm 2 and recall that we can describe forbidden prefixes by admissible functions as explained in Section 4. By Proposition 4.11, line 8, which is executed at most k times, and line 9 both have a cost of $O(n)$. Line 10 has cost $O(n^2 k)$. Now consider lines 19, 21, 25, 27 and 29: At most n executions of these occur. Each line involves $O(nk)$ sums of $O(n)$ terms, and possibly $O(n^2 k)$ assignments. Since the operands have at most $O(n)$ bits, the total cost is at most $O(n^4 k)$. Line 31 involves $O(n^2 k)$ additions with operands of size $O(n)$, and k multiplications and additions with operands of size $O(n+k)$, so the cost is at most $O(n^3 k + k(n+k)^\alpha) = O(n^3 k^{\alpha+1})$. All other lines have cost $O(1)$ and are executed at most n times, so Claim (a) is shown.

Consider Algorithm 1. The body of the for-loop (lines 5 to 13) is executed k times, the body of the while-loop (lines 7 to 12) at most $O(\ln n)$ times. The cost of the for-loop is dominated by the invocations of Algorithm 2; by Claim (a), the cost is at most $O(n^4 \ln n \; k^{\alpha+2})$. Line 2 has a cost of $O(k)$. From Section 3, equation (3), calculating $h_{m,j}$ involves m additions. We need to calculate $h_{m,j}$ for $0 \leq m \leq n$ and $0 \leq j \leq k$, so overall $O(n^2 k)$ additions are required; as it follows by induction on n that the operands have at most $O(n \ln n)$ bits, this has a cost of at most $O(n^3 \ln n \; k)$. Similarly, from equation (2), the calculation of $x_{n,j}$ involves $O(j)$ additions and multiplications; we need to calculate $x_{n,j}$ for $0 \leq j \leq k$, so overall this requires $O(k^2)$ additions and multiplications of operands with at most $O(n \ln n + k)$ bits, which has a cost of at most $O((n \ln n)^\alpha \; k^{\alpha+2})$. This shows Claim (b). $\square$

Remark 7.2. There are several optimisations that can be applied when implementing Algorithms 1 and 2. While they do not affect the estimate of the worst-case complexity, they have a significant impact on actual running times.

1. It is clearly not necessary to compute the forbidden prefixes from scratch in every invocation of Algorithm 2. If the forbidden prefixes of the generated part of the word are stored in Algorithm 1, only one invocation of line 8 of Algorithm 2 is needed.

2. It is possible to initialise the array T in Algorithm 2 directly for the values of a and b corresponding to the largest range that does not involve a forbidden prefix consisting of more than one letter, using the data computed in Section 3.

3. The array T in Algorithm 2 is in general very sparse, as its contents are often collapsed to the 2-dimensional subarrays given by $r = 0$ respectively $s = 0$. Keeping track of the values of r and s that actually can contain non-zero entries and only considering those in any subsequent summations greatly reduces actual running times; see Remark 7.3.

4. Since the least common multiple of all forbidden prefixes is a permutation braid, its length is bounded above by $\binom{n}{2}$. Similarly, $H_n(t)$ is a polynomial

of degree $\binom{n}{2}$, so $h_{n,j} = 0$ if $j > \binom{n}{2}$. It is therefore possible to restrict some summations to $\min\{k, \binom{n}{2}\}$ terms. It is clear from the proof of Proposition 7.1 that in this case (that is, n constant and $k > \binom{n}{2}$) the complexity in terms of k is reduced to $O(k^{\alpha+1})$.

Table 1 contains timing results for several values of n and k, using the implementation of the algorithm in the C-kernel of the computational algebra package MAGMA [3] by the first author. Computations were done with a development version of MAGMA V2.16 on a GNU / Linux system with an Intel E8400 64-bit CPU (core: 3 GHz, FSB: 1333 MHz) and a main memory bandwidth of 4.7 GB/s (X38 chipset, dual channel DDR2 RAM, memory bus: 1066 MHz). The minimum sample size was 100. The sample size was varied to achieve an approximate running time of 2.5 minutes per sample (where possible honouring the minimum sample size), and hence a sufficient degree of accuracy for all parameter values.

				n				
	4	8	16	32	64	128	256	512
4	1.09e0	1.33e0	1.76e0	2.07e0	2.45e0	2.95e0	3.58e0	8.03e0
8	4.62e0	6.39e0	5.79e0	6.10e0	1.26e1	2.19e1	3.13e1	4.84e1
16	2.20e1	5.10e1	4.56e1	7.52e1	9.73e1	1.35e2	2.31e2	2.42e2
32	1.07e2	3.81e2	4.63e2	6.70e2	8.81e2	1.01e3	1.15e3	1.79e3
k 64	4.84e2	2.07e3	4.55e3	6.72e3	5.72e3	7.94e3	7.61e3	7.72e3
128	2.01e3	9.55e3	3.30e4	5.24e4	5.70e4	1.03e5	1.13e5	1.04e5
256	8.42e3	4.27e4	1.90e5	4.17e5	6.06e5	1.20e6	1.81e6	2.06e6
512	4.36e4	1.78e5	9.70e5	3.27e6	5.85e6	1.23e7	2.56e7	3.29e7
1024	2.52e5	7.24e5	4.77e6	2.07e7	5.22e7	1.25e8	2.78e8	4.29e8

Table 1: Running times in μs per random braid for different values of n and k.

k	4	8	16	32	64	128	256	512	1024
e_n	0.35	0.50	0.48	0.48	0.47	0.75	1.08	1.36	1.59

Table 2: Observed exponent e_n of n for different values of k; see Remark 7.3.

n	4	8	16	32	64	128	256	512
e_k	2.30	2.41	2.80	3.02	3.08	3.19	3.27	3.21

Table 3: Observed exponent e_k of k for different values of n; see Remark 7.3.

Remark 7.3. Assuming that the observed running time t is approximated by a relation of the form $t = c\, n^{e_n} k^{e_k}$, the value of e_n for a fixed value of k can be obtained by applying regression analysis to the rows of Table 1. Similarly, the value of e_k for a fixed value of n can be obtained from the columns of Table 1. The observed values of e_n and e_k are given in Tables 2 and 3, respectively.

1. The observed values of e_k are close to $\alpha + 2 \approx 3.585$ for large values of n. For small values of n, where $k > \binom{n}{2}$ for most of the parameter range tested, the observed values of e_k are closer to $\alpha + 1 \approx 2.585$.

   This is exactly what should be expected from Proposition 7.1 and Remark 7.2 (4). That is, as far as k is concerned, the average case complexity is quite close to the worst-case complexity.

2. The observed values of e_n are significantly smaller than 4, that is, the average case complexity in n is significantly better than the worst-case bound of Proposition 7.1.

   This is not surprising in the light of Remark 7.2 (3): Our implementation exploits the fact that the array T is sparse and tends to contain non-zero entries only for relatively few values of the indices r and s. The experimental results suggest that the array T effectively is far from 3-dimensional, especially if k is relatively small. We expect that the theoretical growth rate in n will be achieved, if at all, only for extremely large values of k.

References

[1] M. Albenque, P. Nadeau. Growth function for a class of monoids. *FPSAC 2009*, Hagenberg, Austria. DMTCS proc. AK (2009) 25–38.

[2] A. Bronfman. Growth functions of a class of monoids. Preprint. 2001.

[3] W. Bosma, J. Cannon, C. Playoust. The MAGMA algebra system I: The user language. *J. Symbolic Comput.* **24** (1997) 235–265.

[4] P. Deligne. Les immeubles des groupes de tresses généralisés. *Invent. Math.* **17** (1972), 273–302.

[5] D. Epstein, J. Cannon, D. Holt, S. Levy, M. Paterson, W. Thurston. *Word processing in groups.* Jones and Bartlett Publishers, Boston, MA, 1992.

[6] P. Flajolet, P. Zimmermann, B. Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theor. Comp. Sci.* **132** (1994), 1–35.

[7] J. González-Meneses. On the growth function of braid monoids. In preparation.

[8] E. Stanley. On the number of Reduced Decompositions of Elements of Coxeter Groups. *Europ. J. of Combinatorics* **5** (1984), 359–372.

[9] A. M. Vershik, S. Nechaev, R. Bikbov. Statistical properties of locally free groups with applications to braid groups and growth of random heaps. *Commun. Math. Phys.* **212** (2000), 469–501.

[10] H. Wilf. A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects. *Adv. Math.* **24** (1977), 281–291.

Volker Gebhardt
School of Computing, Engineering and Mathematics
University of Western Sydney, Locked Bag 1797, Penrith NSW 2751, Australia.
E-mail: `v.gebhardt@uws.edu.au`

J. González-Meneses
Departamento de Álgebra, Facultad de Matemáticas, Instituto de Matemáticas
(IMUS), Universidad de Sevilla, Apdo. 1160, 41080 Sevilla, Spain.
E-mail: `meneses@us.es`