2007−09

EFFICIENT SOLUTIONS TO THE BRAID ISOTOPY PROBLEM

PATRICK DEHORNOY

Abstract. We describe the most efficient solutions to the word problem of Artin's braid group known so far, *i.e.*, in other words, the most efficient solutions to the braid isotopy problem, including the Dynnikov method, which could be especially suitable for cryptographical applications. Most results appear in literature; however, some results about the greedy normal form and the symmetric normal form and their connection with grid diagrams may have never been stated explicitly.

Because they are both not too simple and not too complicated, Artin's braid groups B_n have been and remain one of the most natural and promising platform groups for non-commutative group-based cryptography [2, 22, 13]. More precisely, braid groups are not too simple in that they lead to problems with presumably difficult instances, typically the conjugacy problem and the related conjugacy and multiple conjugacy search problems, and they are not too complicated in that there exist efficient solutions to the word problem, a preliminary requirement when one aims at practically computing in a group. It turns out that, since the founding paper [3] appeared in 1947, the word problem of braid groups—which is also the braid isotopy problem—has received a number of solutions: braid groups might even be the groups for which the number of known solutions to the word problem is currently the highest one.

In this paper, we review some of these solutions, namely those that, at the moment, appear as the most efficient ones for practical implementation, and, therefore, the most promising ones for cryptographical applications. What makes the subject specially interesting is that these solutions relie on deep underlying structures that explain their efficiency. Five solutions are described, and they come in two families, namely those based on a normal form, and those that use no normal form. In the first family, we consider the so-called greedy normal form, both in its non-symmetric and symmetric versions. In the second family, we consider the so-called subword reversing method, which, like the greedy normal forms, has a quadratic complexity, the handle reduction method, whose complexity remains unknown but which is very efficient in practice, and Dynnikov's coordinization method, which relies on an entirely different, geometric approach, and might turn to be very efficient. In this description, we deliberately discard lots of alternative solutions which are intrinsically exponential in complexity.

The paper is organized as follows. In Section 1, after setting the background, we describe the greedy normal form and the symmetric normal form, without providing explicit rules to compute them. All results in this section are standard. In Section 2, we introduce grid diagrams and explain—and prove—how to use such diagrams to

1991 *Mathematics Subject Classification.* 20F36, 94A60.

Key words and phrases. braid group, isotopy problem, word problem, greedy normal form, subword reversing, handle reduction, Dynnikov's coordinates.

compute the normal forms of Section 1. Though more or less equivalent to that of [19, Chapter 9], this approach is less standard, and, to the best of our knowledge, the results may have never been stated in the form given here. Finally, in Section 3, we describe the subword reversing, handle reduction, and Dynnikov coordinates methods. The results here already appeared in literature, but Dynnikov's approach, which appears in [15, Chapter 7], has not yet become classical. Also, the formulae of Section 3.3 have been optimized to make implementation easy.

1. Solutions based on a normal form

This section deals about solutions to the braid word problem that consist in defining for each braid x a unique distinguished representative called the normal form of x. When this is done, in order to compute with braids, it is in practice sufficient to work with normal representatives. There exist excellent normal forms for braids, namely those connected with the so-called *greedy normal form* based on Garside's theory [20]. Here we describe them, successively in a non-symmetric and a symmetric version.

1.1. **Braid groups.**

We first recall a few basic definitions and general results about braid groups.

1.1.1. *Presentation.* Artin's braid groups are infinite non-commutative groups. They appear in several *a priori* unrelated frameworks, and they admit many equivalent definitions. In our case, it will be convenient to introduce them by means of explicit presentations.

Definition 1.1. For $n \geqslant 2$, the braid group B_n is defined by the presentation

$$\langle \sigma_1, ..., \sigma_{n-1} \,;\, \sigma_i\sigma_j = \sigma_j\sigma_i \text{ for } |i-j| \geqslant 2,\ \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j \text{ for } |i-j| = 1\rangle. \tag{1.1}$$

An element of B_n will be called an *n-braid*. For each n, the identity mapping on $\{\sigma_1, ..., \sigma_{n-1}\}$ induces an embedding of B_n into B_{n+1}, so that we can consider an n-braid as a particular $(n+1)$-braid. Note that B_2 is an infinite cyclic group, *i.e.*, is isomorphic to the group $\mathbb{Z}$ of integers. For $n \geqslant 3$, the group B_n is not commutative: the center of B_n is the cyclic subgroup generated by the element Δ_n^2, where Δ_n is introduced in (1.2) below.

When a group is specified using a presentation, each element of the group is an equivalence class of words with respect to the congruence generated by the relations of the presentation. In the sequel, a word on the letters $\sigma_1^{\pm 1}$, ..., $\sigma_{n-1}^{\pm 1}$ will be called an *n-braid word*. So, every n-braid is an equivalence class of n-braid words under the congruence $\equiv$ generated by the relations of (1.1). If the braid x is the equivalence class of the word w, we say that w is a representative of x, and we write $x = \overline{w}$.

1.1.2. *The word problem.* Using ε for the empty word, the *word problem* of (1.1) is the algorithmic problem:

> Given one braid word w, does $w \equiv \varepsilon$ hold, *i.e.*, does w represent the unit braid 1?

This is the problem we investigate in the sequel. Because B_n is a group, the above one parameter problem is equivalent to the two parameter problem:

> Given two braid words w, w', does $w \equiv w'$ hold, *i.e.*, do w and w' represent the same braid?

Indeed, $w \equiv w'$ is equivalent to $w^{-1}w' \equiv \varepsilon$, where w^{-1} is the word obtained from w by reversing the order of the letters and exchanging σ_i and σ_i^{-1} everywhere.

1.1.3. *Geometric interpretation.* The elements of B_n can be interpreted as geometric n strand braids [5, 21, 15]. To this end, one associates with every braid word the plane diagram obtained by concatenating the elementary diagrams of Figure 1 corresponding to the successive letters.

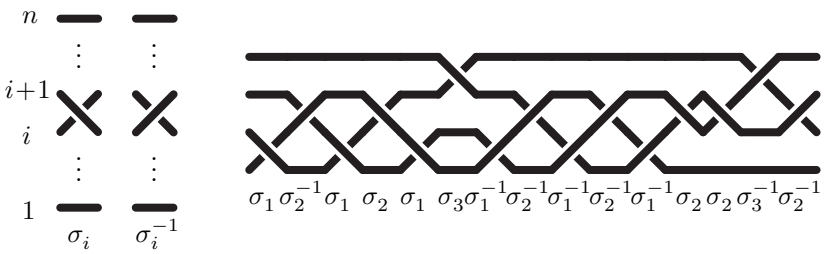


FIGURE 1. The n strand braid diagrams associated with σ_i and σ_i^{-1}, and the 4 strand braid diagram associated with the 4-braid word $\sigma_1\sigma_2^{-1}\sigma_1\sigma_2\sigma_1\sigma_3\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1}\sigma_2\sigma_2\sigma_3^{-1}\sigma_2^{-1}$—also denoted aBabacABABAbbCB in the sequel: one concatenates the successive 4 strand diagrams corresponding to the successive letters of the word.

A braid diagram can be seen as a plane projection of a three-dimensional figure consisting on n disjoint curves connecting the points $(1,0,0),...,(n,0,0)$ to the points $(1,0,1),...,(n,0,1)$ in $\mathbb{R}^3$. Then the relations of (1.1) correspond to ambient isotopy, *i.e.*, to continuously moving the curves without moving their ends and without allowing them to intersect. It is easy to check that each relation in (1.1) corresponds to such an isotopy; the converse implication, *i.e.*, the fact that the projections of isotopic 3D figures can always be encoded in words connected by (1.1) was proved by E. Artin in [3]. Thus the braid word problem for the presentation (1.1) is also the *braid isotopy problem*—thus similar to the (much more difficult) knot isotopy problem.

1.1.4. *Positive braids.* A braid word is said to be *positive* if it contains no letter σ_i^{-1}. A braid is said to be *positive* if it can be represented by at least one positive word. Positive n-braids form a submonoid denoted B_n^+ of the group B_n. Garside's theory [20] implies that B_n^+ admits, as a monoid, the presentation (1.1) and that B_n is a group of fractions of B_n^+, *i.e.*, every braid in B_n can be expressed as $y^{-1}x$ with x, y in B_n^+.

For x, y in B_n^+, we say that x is a *left divisor* of y, or, equivalently, that y is a *right multiple* of x, denoted $x \preccurlyeq_L y$, if $y = xz$ holds for some z in B_n^+. Right divisors and left multiples are defined symmetrically. With respect to left divisibility (and to right divisibility as well), B_n^+ has the structure of a lattice: any two positive n-braids x, y admit a greatest common left divisor $\gcd_L(x,y)$, and a least common right multiple.

1.1.5. *Permutation of a braid.* The geometric interpretation makes it clear that mapping σ_i to the transposition that exchanges i and $i+1$ induces a surjective homomorphism of the braid group B_n onto the symmetric group S_n. Under this homomorphism, here denoted π, a braid x is mapped to the permutation f of $\{1,...,n\}$

such that the strand that finishes at position i in any braid diagram representing x begins at position $f(i)$.

1.1.6. *Simple braids.* A special rôle is played by the positive n-braid Δ_n inductively defined by

$$\Delta_1 = 1, \qquad \Delta_n = \sigma_1\sigma_2\dots\sigma_{n-1}\,\Delta_{n-1}. \tag{1.2}$$

In B_n^+, the left and the right divisors of Δ_n coincide, they include each of σ_1, ..., σ_{n-1}, and they make a finite sublattice of B_n^+ with $n!$ elements. These divisors of Δ_n are called *simple* braids. Geometrically, simple braids are those positive braids that can be represented by a braid diagram in which any two strands cross at most once. Moreover, the restriction of the projection π to simple braids is a bijection: for each permutation f in S_n, there exists exactly one simple braid s satisfying $\pi(s) = f$. This simple braid will be denoted by $\widehat{f}$.

1.2. **The greedy normal form.** The seminal results of F.A. Garside [20] subsequently developed in [26, 1, 18] imply that braid groups can be equipped with a remarkable normal form, the so-called greedy normal form. The latter is excellent both in theory and in practice as it provides a bi-automatic structure, and it is easily computable.

1.2.1. *Description.* The greedy normal form exists in several variants, in particular left and symmetric right versions. Here we shall consider the left versions only. Again, there exist two different versions. Both consist in expressing an arbitrary braid as a quotient of two positive braids, *i.e.*, as a fraction. In the version considered in this section, all denominators have some special form, namely they are powers of the element Δ_n. By contrast, in the version considered in Section 1.3 below, the numerator and the denominator of fractions play symmetric rôles.

Definition 1.2. (*i*) A sequence of simple braids $(s_1, ..., s_p)$ is said to be *normal* if, for each $k < p$, every σ_i that is a left divisor of s_{k+1} is a right divisor of s_k.

(*ii*) A sequence of permutations $(f_1, ..., f_p)$ is said to be *normal* if, for each $k < p$, every recoil of f_{k+1} is a descent of f_k, *i.e.*, if $f_{k+1}^{-1}(i) > f_{k+1}^{-1}(i+1)$ implies $f_k(i) > f_k(i+1)$.

The connection between (*i*) and (*ii*) in Definition 1.2 is that, if s is a simple n-braid and f is the associated permutation, then σ_i is a left (*resp.* right) divisor of s if and only if we have $f^{-1}(i) > f^{-1}(i+1)$ (*resp.* $f(i) > f(i+1)$). So a sequence of simple braids $(s_1, ..., s_p)$ is normal if and only if the associated sequence of permutations $(\pi(s_1), ..., \pi(s_p))$ is normal.

We denote by ω_n the flip permutation of $\{1, ..., n\}$ defined by $\omega_n(i) = n - i + 1$.

Theorem 1.3. [19, Chapter 9] (*i*) *Every braid z in B_n admits a unique decomposition of the form $\Delta_n^m s_1 ... s_p$ with m in $\mathbb{Z}$ and $(s_1, ..., s_p)$ a normal sequence of simple braids satisfying $s_1 \neq \Delta_n$ and $s_p \neq 1$.*

(*ii*) *Every braid z in B_n admits a unique decomposition of the form $\Delta_n^m \widehat{f}_1 ... \widehat{f}_p$ with m in $\mathbb{Z}$ and $(f_1, ..., f_p)$ a normal sequence of permutations satisfying $f_1 \neq \omega_n$ and $f_p \neq$ id.*

In the situation of Theorem 1.3(*i*), the sequence $(m; s_1, ..., s_p)$ is called the *greedy normal form* of z—or the n-greedy normal form of z if we wish to insist on the braid index n. As simple braids are in one-to-one correspondence with permutations,

and by the remark above, the braid form and the permutation form of the greedy normal form are equivalent. So there is no problem in also calling the sequence $(m; f_1, ..., f_p)$ of Theorem 1.3(*ii*) the greedy normal form of z.

Clearly, $(0; \emptyset)$ is the greedy normal form of 1, and the uniqueness of the greedy normal form implies the following solution to the braid isotopy problem—but a solution that remains uneffective as long as we give no method for computing from an arbitrary braid word w the greedy normal form of $\overline{w}$, *i.e.*, until Section 2 below:

Corollary 1.4. *A braid word w represents 1 in the braid group if and only if the greedy normal form of $\overline{w}$ is $(0; \emptyset)$. Two braid words w, w' represent the same braid in B_n if and only if the greedy normal forms of the braids $\overline{w}$ and $\overline{w'}$ coincide.*

Example 1.5. In order to obtain shorter notation, we shall in the sequel use $\mathtt{a}, \mathtt{b}, \mathtt{c}...$ for $\sigma_1, \sigma_2, \sigma_3...$, and, symmetrically, $\mathtt{A}, \mathtt{B}...$ for $\sigma_1^{-1}, \sigma_2^{-1}...$ (as in the caption of Figure 1). Then, a typical greedy normal form for a 4-braid is the sequence

$(-2; \mathtt{ac}, \mathtt{abcb}, \mathtt{bcba}, \mathtt{a})$,

i.e., equivalently, using $(f(1), ..., f(n))$ to specify a permutation f of $\{1, ..., n\}$,

$(-2; (2,1,4,3), (2,4,3,1), (4,1,3,2), (2,1,3,4))$,

consisting of an integer and four simple 4-braids, or, equivalently, four permutations of $\{1, ..., 4\}$: for instance, $(2,1,4,3)$ is the permutation associated with $\mathtt{ac}$, *i.e.*, with $\sigma_1\sigma_3$. To check that we have a greedy normal form, we observe for instance that the descents of $(2,1,4,3)$ are 1 and 3, while the recoils of $(2,4,3,1)$, *i.e.*, the descents of $(4,1,3,2)$, are 1 and 3 as well, so the normality condition is satisfied between $(2,1,4,3)$ and $(2,4,3,1)$. The other verifications are similar. So, the above sequences are two versions of the greedy normal form of the 4-braid represented by $\Delta_4^{-2}.\mathtt{ac.abcb.bcba.a}$, *i.e.*, by

$w = \mathtt{ABACBA.ABACBA.ac.abcb.bcba.a}$.

As the above normal form is not $(0, \emptyset)$, we deduce from Corollary 1.4 that w does not represent 1 in B_4.

1.2.2. *Explanation.* The existence and uniqueness of the greedy normal form follows from two results:

(*i*) For every braid z in B_n, there exist $m \in \mathbb{Z}$ and $x \in B_n^+$ satisfying $\Delta_n^m x = z$, the decomposition being unique provided $\Delta_n \not\preccurlyeq_L x$ is required;

(*ii*) For every positive braid x, there exists a unique normal sequence $(s_1, ..., s_p)$ of simple n-braids with $s_p \neq 1$ satisfying $s_1...s_p = x$.

The proof of (*i*) is an easy induction on the length of a braid word representing z once one knows, for each i, the relations $\sigma_i \preccurlyeq_L \Delta_n$ and $\Delta_n\sigma_i = \sigma_{n-i}\Delta_n$, which imply that, by multiplying an n-braid by a sufficient large power of Δ_n, one can always obtain a positive braid.

As for (*ii*), the existence of left gcd's in the monoid B_n^+ implies that each positive n-braid x can be expressed as $x = s_1x'$ with $s_1 = \gcd_L(x, \Delta_n)$, and $x \neq 1$ implies $s_1 \neq 1$. By iterating the process, thus writing $x' = s_2x''$, *etc.*, one eventually obtains a decomposition $x = s_1...s_p$. By construction, the sequence $(s_1, ..., s_p)$ consists of divisors of Δ_n, *i.e.*, of simple n-braids, and, for each $k < p$, one has $s_k = \gcd_L(s_ks_{k+1}...s_p, \Delta_n)$, hence, *a fortiori*, $s_k = \gcd_L(s_ks_{k+1}, \Delta_n)$. The point is that the latter relations are equivalent to (s_k, s_{k+1}) being normal for each k, and, therefore, to $(s_1, ..., s_p)$ being normal. The uniqueness comes from the fact that, if $(s_1, ..., s_p)$ is a normal sequence, then, necessarily, one has $s_1 = \gcd_L(s_1...s_p, \Delta_n)$.

1.2.3. *Discussion.* What is missing in the above description of the greedy normal form is an algorithm for computing the (unique) normal form of a braid z starting from an arbitrary representative of z. Clearly, the existence of such an algorithm is a necessary condition for using the normal form in practice. Such an algorithm will be provided in Section 2 below, and discussing the practical implementation of the greedy normal form will be possible only then.

Actually, the point is not necessarily to find the normal form equivalent to an arbitrary word, but rather to find the normal form of the product or the quotient of two normal forms. Indeed, whenever one chooses to work with normal forms, one may forget about non-normal words provided one is able to perform the basic operations inside the family of normal forms. Of course, as the generator σ_i is itself a braid, with normal form $(0;\sigma_i)$, an algorithm computing the product of two normal forms will in particular determine the product of a normal form by σ_i and, therefore, inductively determine the normal form of any product of $\sigma_i^{\pm1}$'s, but the general philosophy is not exactly that of a normalizing algorithm.

Note that, while the permutation variant of the greedy normal form is non-ambiguously defined, the braid word variant is not: for instance, the first braid factor in Example 1.5, namely $\mathtt{ac}$, is uniquely defined as a simple braid, but it can be represented by two different positive braid words, namely $\mathtt{ac}$ and $\mathtt{ca}$. So the braid word form becomes unique only when a distinguished word representative has been chosen for every simple braid. This explains why the permutation form is often more convenient.

1.3. **The symmetric normal form.** In the greedy normal form where the denominator is always a power of Δ_n. The symmetric normal form is a variant in which the numerator and the denominator play symmetric rôles.

1.3.1. *Description.* The symmetric normal form appeals to the same notion of a normal sequence of simple braids as the greedy normal form.

Theorem 1.6. [19, Chapter 9] (i) *Every braid z admits a unique decomposition as $t_q^{-1}\dots t_1^{-1}\,s_1\dots s_p$ with $(s_1,\dots,s_p)$, $(t_1,\dots,t_q)$ two normal sequences of simple braids satisfying $s_p\neq1$, $t_q\neq1$, and $\gcd_L(s_1,t_1)=1$.*

(ii) *Every braid z admits a unique decomposition as $\widehat{g_q}^{-1}\dots\widehat{g_1}^{-1}\,\widehat{f_1}\dots\widehat{f_p}$ with $(f_1,\dots,f_p)$, $(g_1,\dots,g_q)$ two normal sequences of permutations satisfying $f_p,g_q\neq\mathrm{id}$ and such that $f_1^{-1}(i)>f_1^{-1}(i+1)$ implies $g_1^{-1}(i)\leqslant g_1^{-1}(i+1)$.*

In the situation of Theorem 1.6(i), the double sequence $(t_1,\dots,t_q;s_1,\dots,s_p)$ is called the *symmetric normal form* of z. As for the greedy normal form, both versions of the symmetric normal forms are equivalent, and the double sequence of permutations $(g_1,\dots,g_q;f_1,\dots,f_p)$ of Theorem 1.6(ii) is also called the symmetric normal form of z. As $(\emptyset;\emptyset)$ is the symmetric normal form of 1, the uniqueness of the symmetric normal form provides the following solution to the braid isotopy problem:

Corollary 1.7. *A braid word w represents* 1 *in the braid group if and only if the symmetric normal form of $\overline{w}$ is $(\emptyset;\emptyset)$. Two braid words w,w' represent the same braid in B_n if and only if the symmetric normal forms of $\overline{w}$ and $\overline{w'}$ coincide.*

Example 1.8. A typical symmetric normal form is

$(\mathtt{ab},\mathtt{bacb};\mathtt{bcba},\mathtt{a})$,

i.e., equivalently,

$((2,3,1,4),(3,4,1,2);(4,1,3,2),(2,1,3,4))$,

For instance, the normality condition between the first factors of the two sequences holds as 1 is the only recoil of $(2,3,1,4)$, while 2 and 3 are those of $(4,1,3,2)$. In other words, the simple braids `ab` and `bcba` admit no nontrivial common left divisor. Thus the above expressions specify the symmetric normal form of the braid represented by

$w = \texttt{BCAB.BA.bcba.a}$,

which we shall see below coincides with the braid of Example 1.5—and of Figure 1. As the above normal form is not $(\emptyset,\emptyset)$, we deduce that w does not represent 1 in B_4.

1.3.2. *Explanation.* As B_n is a group of fractions for the monoid B_n^+, every element of B_n can be expressed as a fraction $y^{-1}x$ with x, y in B_n^+, and the decomposition is unique if, in addition, $\gcd_L(x,y)$ is required to be 1. The symmetric normal form is obtained by taking the greedy normal forms of the positive braids x, y so obtained, with the only difference that the Δ_n factors are not separated. The specific properties of normal sequences imply that $\gcd_L(x,y) = 1$ is equivalent to $\gcd_L(s_1,t_1) = 1$, where s_1 and t_1 respectively are the first factors in the normal forms of x and y.

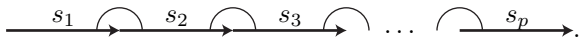
1.3.3. *Discussion.* As in Section 1.2, our description will be complete only when we give algorithms for computing the symmetric normal form of a product or of a quotient. This will be done in Section 2.

2. Grid properties of the normal form

The interest of the greedy and symmetric normal forms of braids lies in the existence of simple computing rules for determining the normal form of a product or of a quotient. Here we shall solve the following two problems, which then easily leads to complete algorithms for all problems connected with the normal forms:

> Starting with the greedy normal forms of two positive braids x, y, find the greedy normal form of the product yx and, assuming that y is a right divisor of x, the greedy normal form of the quotient xy^{-1}.

The solutions we shall describe involve grid diagrams that visualize the properties of normal sequences. To this end, it will be convenient to associate with every braid x an arrow labelled x, so that a relation of the form $xy = z$ corresponds to a commutative diagram for the associated arrows. We shall indicate that a sequence $(s_1, ..., s_p)$ is normal by drawing an arc connecting the final end of each arrow with the initial end of the next one, on the shape of



2.1. **Prerequisites.** The only properties of simple braids needed in the forthcoming proofs are summarized in the following lemma, which is standard:

Lemma 2.1. [7] *For x in B_n^+, let $\alpha(x) := \gcd_L(x, \Delta_n)$. Then, for all x, y, we have:*

$$\alpha(x) \preccurlyeq_L x, \quad \text{and } \alpha(x) = x \text{ if } x \text{ is simple}, \tag{2.1}$$

$$x \preccurlyeq_L y \text{ implies } \alpha(x) \preccurlyeq_L \alpha(y), \tag{2.2}$$

$$\alpha(xy) = \alpha(x\alpha(y)). \tag{2.3}$$

Then the definition of normal sequences immediately rewrites in terms of the function α as follows—and this is the technical form we shall use in the sequel:

Lemma 2.2. *A sequence of simple braid $(s_1, ..., s_p)$ is normal if and only if, for each $k < p$, we have $s_k = \alpha(s_k s_{k+1})$.*

In the sequel, we mostly deal with positive braids. When we say that $(s_1, ..., s_p)$ is the normal form of a positive n-braid x, we mean that $(s_1, ..., s_p)$ is normal and that $x = s_1 ... s_p$. Equivalently, this means that the greedy normal form of x is $(m; s_{m+1}, ..., s_p)$, where m is the number of initial k's such that s_k equals Δ_n.

2.2. **Grid properties for the quotient.** We start with the problem of finding the normal form of xy^{-1} from those of x and y, where x and y belong to B_n^+. In principle, the computation makes sense only if y happens to be a right divisor of x. Actually, in any case, the positive braids x, y admit a left lcm in B_n^+, and what we shall do is to compute the normal form of this left lcm, denoted $\mathrm{lcm}_L(x, y)$, and of the associated left complements defined as follows:

Definition 2.3. For x, y in B_n^+, the unique x' in B_n^+ satisfying $\mathrm{lcm}_L(x, y) = x'y$ is called the *left complement* of x in y, and denoted by x/y.

If y happens to be a right divisor of x, *i.e.*, if $y/x = 1$ holds, then we have $\mathrm{lcm}_L(x, y) = y$, and $x/y = xy^{-1}$. Thus an algorithm computing the left complement is in particular an algorithm computing the right quotient when it exists.

A standard observation is that simple braids are closed under left lcm and left complement. Indeed, assume that s, t are simple, and let $u = \mathrm{lcm}_L(s, t) = s't = t's$, *i.e.*, $s' = s/t$ and $t' = t/s$. Then, s and t are right divisors of Δ_n, hence u is also a right divisor of Δ_n, and it is therefore a simple braid. Then, s' and t' are simple as well, as every left divisor of a simple braid is a simple braid. Moreover, it is straightforward to check that $s't$ being the left lcm of s and t is equivalent to s' and t' admitting no common left divisor except 1, *i.e.*, to $\mathrm{gcd}_L(s', t') = 1$.

Definition 2.4. A commutative diagram consisting of four simple braids s, t, s', t' satisfying $s't = t's$ and $\mathrm{gcd}_L(s', t') = 1$ is called a *C-tile* (like "complement tile"), and it is represented as in Figure 2.

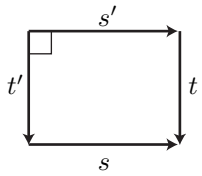


FIGURE 2. A C-tile: a commutative diagram involving four simple braids such that the two initial ones have no nontrivial common left divisor—here indicated by a perpendicularity sign.

We establish normality conditions for diagrams involving the above C-tiles.

Lemma 2.5. *(Figure 3) Assume that $s_1, s_2, s_1', s_2', t_0, t_1, t_2$ are simple braids satisfying $t_2 s_1 = s_1' t_1$, $t_1 s_2 = s_2' t_0$, $\mathrm{gcd}_L(s_2', t_1) = 1$, and that (s_1, s_2) is normal. Then (s_1', s_2') is normal as well.*

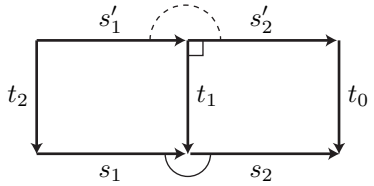


FIGURE 3. A grid property involving C-tiles: assuming that the right rectangle is a C-tile and the bottom sequence (s_1, s_2) is normal, the top sequence (s'_1, s'_2) is normal as well.

Proof. With the notation of Figure 3, we compute:

$$\begin{aligned}
\alpha(s'_1 s'_2) &\preccurlyeq_L \alpha(s'_1 s'_2 t_0) && \text{by (2.2)}\\
&= \alpha(t_2 s_1 s_2) && \text{by commutativity}\\
&= \alpha(t_2 \alpha(s_1 s_2)) && \text{by (2.3)}\\
&= \alpha(t_2 s_1) && \text{by the hypothesis that } (s_1, s_2) \text{ is normal}\\
&= \alpha(s'_1 t_1) && \text{by commutativity}\\
&\preccurlyeq_L s'_1 t_1 && \text{by (2.1).}
\end{aligned}$$

On the other hand, by (2.1) again, we have $\alpha(s'_1 s'_2) \preccurlyeq_L s'_1 s'_2$, hence

$$\alpha(s'_1 s'_2) \preccurlyeq_L \gcd_L(s'_1 s'_2, s'_1 t_1) = s'_1 \cdot \gcd_L(s'_2, t_1) = s'_1,$$

using the hypothesis $\gcd_L(s'_2, t_1) = 1$. It follows that (s'_1, s'_2) is normal. □

Similarly, we have the following result involving the diagonals of C-tiles, *i.e.*, the left lcm's of the corresponding simple braids.

Lemma 2.6. *(Figure 4) Assume that (s_1, s_2) and (t_1, t_2) are normal sequences of simple braids. Let u_1, u_2 be as in Figure 4. Then (u_1, u_2), (u_1, s'_2), and (u_1, t'_2) are normal as well.*

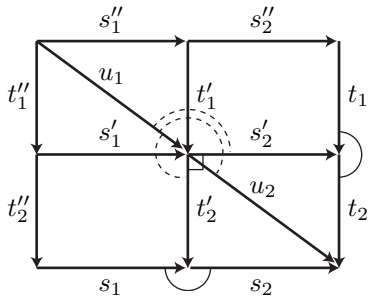


FIGURE 4. Another grid property involving C-tiles: assuming that the bottom right rectangle is a C-tile and the bottom and right sequences (s_1, s_2) and (t_1, t_2) are normal, the diagonal sequence (u_1, u_2) is normal as well, and so are the diagonal-then-horizontal sequence (u_1, s'_2) and the diagonal-then-vertical sequence (u_1, t'_2).

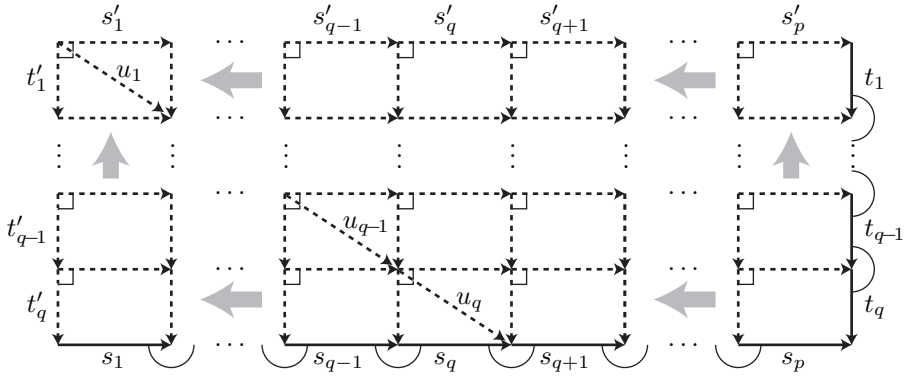


FIGURE 5. Construction of the left lcm and the left complements by the grid method: we start from the bottom and right sides, and fill the grid with C-tiles, from right to left and from bottom to top; then every row, every column, and every diagonal in the grid consist of normal sequences.

Proof. With the notation of Figure 4, we compute

$$\begin{aligned}
\alpha(u_1u_2) &= \alpha(t''_2t''_1s_1s_2) && \text{by commutativity}\\
&= \alpha(t''_2t''_1\alpha(s_1s_2)) && \text{by (2.3)}\\
&= \alpha(t''_2t''_1s_1) && \text{by the hypothesis that } (s_1,s_2) \text{ is normal}\\
&= \alpha(u_1t'_2) && \text{by commutativity}\\
&\preccurlyeq_L u_1t'_2 && \text{by (2.1).}
\end{aligned}$$

Similarly, we obtain $\alpha(u_1u_2) \preccurlyeq_L u_1s'_2$, and we deduce

$$\alpha(u_1u_2) \preccurlyeq_L \gcd_L(u_1t'_2, u_1s'_2) = u_1 \cdot \gcd_L(s'_2,t'_2) = u_1$$

from the hypothesis $\gcd_L(s'_2,t'_2) = 1$. So (u_1,u_2) is normal. As s'_2 is a left divisor of u_2, the normality of (u_1,u_2) implies that of (u_1,s'_2), and, similarly, that of (u_1,t'_2). □

We are ready to establish:

Proposition 2.7. (*i*) *Assume that* $(s_1,...,s_p)$ *and* $(t_1,...,t_q)$ *are normal sequences of simple braids. Let* $\mathcal{D}$ *be the grid diagram obtained by starting from the right column* $(t_1,...,t_q)$ *and the bottom row* $(s_1,...,s_p)$ *and filling the diagram with* C*-tiles from right to left, and from bottom to top, as shown in Figure 5. Then every path in* $\mathcal{D}$ *consisting of diagonal arrows followed by horizontal arrows, as well as every path consisting of diagonal arrows followed by horizontal arrows corresponds to a normal sequence.*

(*ii*) *Let* $x = s_1...s_p$ *and* $y = t_1...t_q$. *Let* $(s'_1,...,s'_p)$ *be the top row of* $\mathcal{D}$, *let* $(t'_1,...,t'_q)$ *be its left column, and, assuming* $p \geqslant q$, *let* $(u_1,...,u_q)$ *be the diagonal from the top-left corner. Then* $(s'_1,...,s'_p)$ *is the normal form of* x/y, *while* $(t'_1,...,t'_q)$ *is the normal form of* y/x, *and* $(u_1,...,u_q,s_{q+1},...,s_p)$ *is the normal form of the left lcm of* x *and* y.

Proof. By construction, the diagram $\mathcal{D}$ is commutative. Let $z = s'_1...s'_pt_1...t_q$. Then z is a common left multiple of x and y. Moreover, an easy induction shows that every common left multiple z' of x and y has to be a left multiple of z: indeed,

z', being a common left multiple of s_p and t_q, has to be a left multiple of the braid represented by the diagonal of the bottom-right rectangle in $\mathcal{D}$, and we argue similarly for each of the pq rectangles in $\mathcal{D}$. Hence z is the left lcm of x and y, and, therefore, we have $s'_1...s'_p = x/y$ and $t'_1...t'_q = y/x$.

Then, by repeatedly applying Lemma 2.5, we obtain that every row and every column in $\mathcal{D}$ is a normal sequence. As for the diagonals, and the diagonals followed by rows or columns, we similarly apply Lemma 2.6. □

As a straightforward application, we obtain the following computing rule for determining the normal form of xu^{-1} from that of x.

Corollary 2.8. *Assume that $(s_1, ..., s_p)$ is the normal form of a positive braid x, and that u is a simple braid that divides x on the right. Then the normal form of xu^{-1} is the sequence $(s'_1, ..., s'_p)$ determined by the grid diagram of Figure 6.*

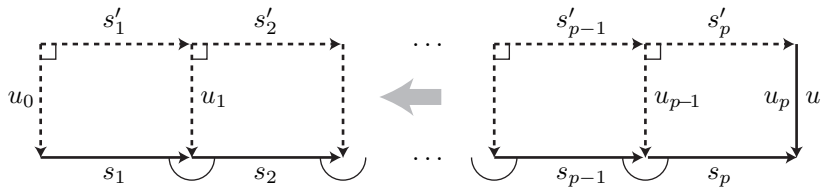


FIGURE 6. Normal form of xu^{-1} from that of x: start from the plain arrows and fill the diagram with C-tiles, from right to left; the expected normal form is the sequence $(s'_1, ..., s'_p)$ read on the top line; the hypothesis that u is a right divisor of x guarantees that u_0 is 1—without any hypothesis, $(s'_1, ..., s'_p)$ is the normal form of x/u, and u_0 is then u/x.

2.3. **Grid properties for the product.** We turn to the product, with the question of determining the normal form of yx from the normal forms of x and y. The method is similar to that of the previous section, with another type of basic tile.

For all simple braids t_1, t_2, there exist simple braids s_1, s_2 satisfying $s_1s_2 = t_1t_2$ and (s_1, s_2) is normal. Indeed, let $s_1 = \alpha(t_1t_2)$, and s_2 satisfying $t_1t_2 = s_1s_2$. By (2.1), $t_1 \preccurlyeq_L t_1t_2$ implies $t_1 = \alpha(t_1) \preccurlyeq_L \alpha(t_1t_2) = s_1$, so we have $t_2 = us_2$ for some u, which forces s_2 to be simple. In other words, the normal form of the product of two simple braids must consist of at most two simple braids.

Definition 2.9. A commutative diagram consisting of four simple braids s_1, s_2, t_1, t_2 satisfying $s_1s_2 = t_1t_2$ and such that (s_1, s_2) is normal is called a *P-tile* (like "product tile"), and it is represented as in Figure 7.

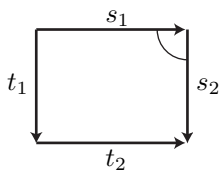


FIGURE 7. A P-tile: a commutative diagram involving four simple braids such that two of them make a normal sequence.

As was done above with C-tiles, we establish normality results in diagrams involving P-tiles.

Lemma 2.10. *(Figure 8) Assume that $s_1, s_2, s'_1, s'_2, t_0, t_1, t_2$ be simple braids satisfying $t_0 s_1 = s'_1 t_1$, $t_1 s_2 = s'_2 t_2$, and such that (s_1, s_2) and (s'_1, t_1) are normal. Then (s'_1, s'_2) is normal as well.*

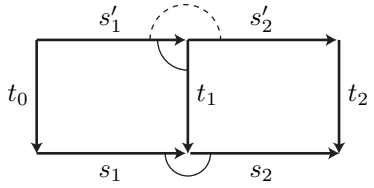


FIGURE 8. A grid property involving P-tiles: assuming that the left rectangle is a P-tile and the bottom sequence (s_1, s_2) is normal, the top sequence (s'_1, s'_2) is normal as well.

Proof. (ii) With the notation of Figure 8, we compute:

$$\begin{aligned}
\alpha(s'_1 s'_2) &\preccurlyeq_{\mathrm{L}} \alpha(s'_1 s'_2 t_2) && \text{by (2.2)}\\
&= \alpha(t_0 s_1 s_2) && \text{by commutativity}\\
&= \alpha(t_0 \alpha(s_1 s_2)) && \text{by (2.3)}\\
&= \alpha(t_0 s_1) && \text{by the hypothesis that } (s_1, s_2) \text{ is normal}\\
&= \alpha(s'_1 t_1) && \text{by commutativity}\\
&= s'_1 && \text{by the hypothesis that } (s'_1, t_1) \text{ is normal,}
\end{aligned}$$

so (s'_1, s'_2) is normal. □

To go further, we need to use the duality with respect to Δ_n. For each simple n-braid s, there exists a unique simple n-braid s^* satisfying $ss^* = \Delta_n$. Then, we have $s\Delta_n = ss^*s^{**} = \Delta_n s^{**}$, hence s^{**} is the conjugate $\Delta_n^{-1} s \Delta_n$, which is easily seen to be the image of s under the flip automorphism ϕ_n that exchanges σ_i and σ_{n-i} for $1 \leqslant i < n$.

Lemma 2.11. *Assume that s, t are simple braids. Then (s, t) is normal if and only if $\gcd_{\mathrm{L}}(s^*, t) = 1$ holds.*

Proof. We have

$$\alpha(st) = \gcd_{\mathrm{L}}(st, \Delta_n) = \gcd_{\mathrm{L}}(st, ss^*) = s \cdot \gcd_{\mathrm{L}}(t, s^*),$$

hence $\alpha(st) = s$ is equivalent to $\gcd_{\mathrm{L}}(t, s^*) = 1$. □

Lemma 2.12. *(Figure 9) Assume that $s_0, s_1, s_2, t_1, t_2, t'_1, t'_2$ are simple braids satisfying $t_1 s_1 = s_0 t'_1$, $t_2 s_2 = s_1 t'_2$, and such that (t_1, t_2) and (s_1, t'_2) are normal. Then (t'_1, t'_2) is normal as well.*

Proof. Introduce s_0^*, s_1^* and s_2^*. Then the diagram of Figure 10 is commutative. As ϕ_n is an automorphism of B_n^+—and of the group B_n too—the hypothesis that (t_1, t_2) is normal implies that $(\phi_n(t_1), \phi_n(t_2))$ is normal as well.

Now, by Lemma 2.11, the hypothesis that (s_1, t'_2) is normal gives $\gcd_{\mathrm{L}}(s_1^*, t'_2) = 1$. By Lemma 2.5, the latter condition together with the normality of $(\phi_n(t_1), \phi_n(t_2))$ implies that (t'_1, t'_2) is normal. □

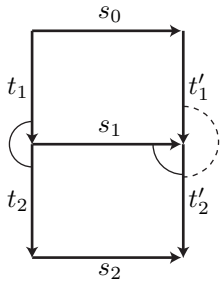


FIGURE 9. Another grid property involving P-tiles: assuming that the bottom rectangle is a P-tile and the left sequence (t_1, t_2) is normal, the right sequence (t'_1, t'_2) is normal as well.

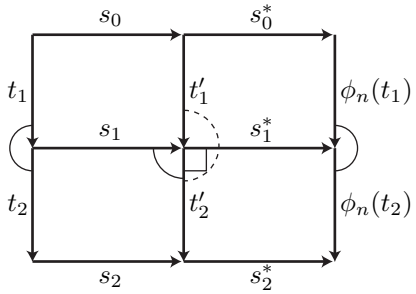


FIGURE 10. Proof of Lemma 2.12: one introduces the dual of the horizontal arrows; the hypothesis that the left column (t_1, t_2) is normal implies that the right column $(\phi_n(t_1), \phi_n(t_2))$ is normal as well, and, then, we apply Lemma 2.5 to come back to (t'_1, t'_2).

We deduce

Proposition 2.13. *(i) Assume that $(s_1, ..., s_p)$ and $(t_1, ..., t_q)$ are normal sequences of simple braids. Let $\mathcal{D}$ be the grid diagram obtained by starting from the left column $(t_1, ..., t_q)$ and the bottom row $(s_1, ..., s_p)$ and filling the diagram with P-tiles from left to right, and from bottom to top, as shown in Figure 11. Then every path in $\mathcal{D}$ consisting of horizontal arrows followed by vertical arrows corresponds to a normal sequence.*

(ii) Let $x = s_1...s_p$ and $y = t_1...t_q$. Let $(s'_1, ..., s'_p)$ be the top row of $\mathcal{D}$ and $(t'_1, ..., t'_q)$ is its right column. Then $(s'_1, ..., s'_p, t'_1, ..., t'_q)$ is the normal form of yx.

Proof. By hypothesis, the bottom row is normal, hence, by Lemma 2.10, we inductively deduce that the kth row from the bottom is normal. Similarly, by hypothesis, the left column is normal, hence, by Lemma 2.12, we inductively deduce that kth column from the left is normal. Finally, the sequence $(s'_1, ..., s'_p, t'_1, ..., t'_q)$ is normal, as it is the concatenation of two normal sequences and, moreover, (s'_p, t'_1) is normal by construction. As, by construction, the diagram $\mathcal{D}$ is commutative, the product of the latter sequence is also the product of the left column and the bottom row, *i.e.*, it is the braid yx. □

As in the case of the quotient, we deduce in particular rules for computing the normal form of ux and xu from that of x when u is a simple braid.

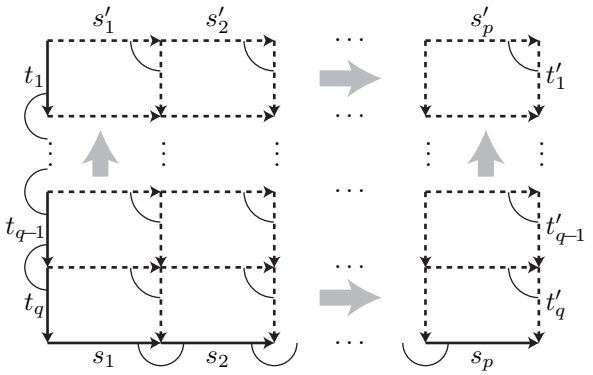


FIGURE 11. Computation of the normal form of a product by means of a grid: we start from the left and the bottom sides, and fill the diagram using P-tiles, from left to right and from bottom to top; then the top row and the right column are normal sequences as well, and so is the sequence obtained by concatenating them.

Corollary 2.14. *Assume that $(s_1, ..., s_p)$ is the normal form of the positive braid x, and that u is a simple braid. Then the normal forms of ux and xu are the sequences $(s'_1, ..., s'_p, u'_p)$ and $(u''_0, s''_1, ..., s''_p)$ determined by the grid diagrams of Figure 12.*

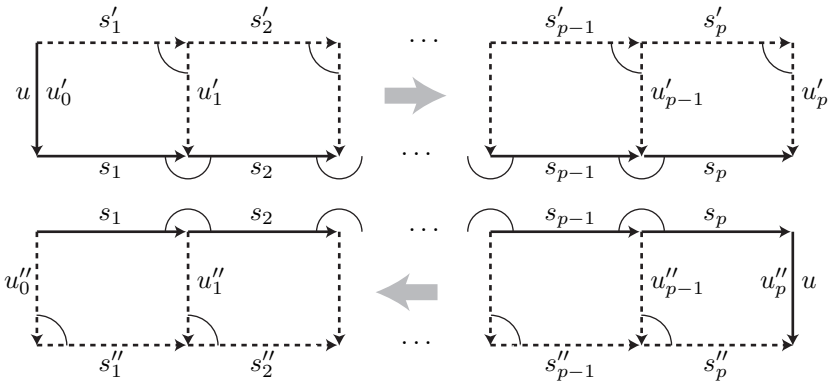


FIGURE 12. Construction of the normal from of ux (above) and xu (below) from the normal form $(s_1, ..., s_p)$ of x: start with $u'_0 := u$ (*resp.* with $u''_p := u$) and fill the diagram with P-tiles from left to right (*resp.* from right to left); then $(s'_1, ..., s'_p, u'_p)$ (*resp.* $(u''_0, s''_1, ..., s''_p)$) is the expected normal form—up to removing u'_p or s''_p if the latter happen to be 1.

2.4. **Application to computing normal forms.** We can now easily provide algorithms that compute the greedy normal form and the symmetric normal form of a braid starting from an arbitrary word that represents it. Clearly, the point is, starting from the greedy (*resp.* symmetric) normal form of a braid z, to be able to determine the greedy (*resp.* symmetric) normal form of $zu^{\pm 1}$ when u is a simple braid, hence in particular a generator σ_i. The solutions come as easy applications of the results of the previous sections.

2.4.1. *Computing the greedy normal form.* We recall that, for s a simple n-braid, s^* denotes the unique (simple) braid satisfying $ss^* = \Delta_n$. Symmetrically, we denote by ${}^*s$ the unique (simple) braid satisfying ${}^*ss = \Delta_n$. Note that, by construction, $({}^*s)^* = s$ holds for each simple braid s.

Proposition 2.15. *Assume that the n-greedy normal form of z is $(m; s_1, ..., s_p)$ and u is a simple n-braid.*

(i) Let s'_p, u_{p-1}, ..., s'_1, u_0 be determined by filling the top diagram of Figure 13. Then the greedy normal form of zu is $(m+1; s'_1, ..., s'_p)$ if $u_0 = \Delta_n$ holds, and $(m; u_0, s'_1, ..., s'_p)$ otherwise.

(ii) Let s'_p, u_{p-1}, ..., s'_1, u_0 be determined by filling the bottom diagram of Figure 13. Then the greedy normal form of zu^{-1} is $(m; s'_1, ..., s'_p)$ if $u_0 = 1$ holds, and $(m-1; {}^*u_0, s'_1, ..., s'_p)$ otherwise.

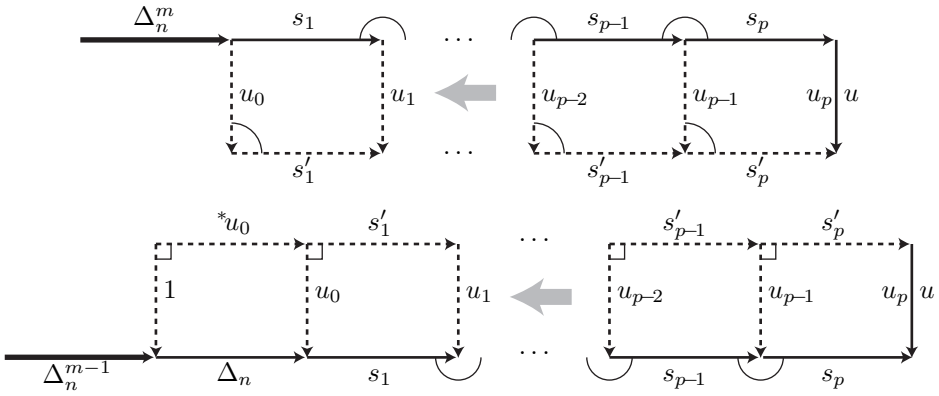


FIGURE 13. Greedy normal form of zu (top) and zu^{-1} (bottom) from the greedy normal form $(m; s_1, ..., s_p)$ of z: starting from $u_p := u$, fill the diagram using P-tiles (*resp.* C-tiles) from right to left, and adapt at the left end to guarantee the connection with the Δ_n^m-factor, namely include u_0 in the latter if needed (top), or factorize $\Delta_n u_0^{-1}$ into ${}^*u_0$ (bottom).

Proof. (i) By commutativity of the diagram, we have $zu = \Delta_n^m u_0 s'_1 ... s'_p$, so the point is to check that the sequence $(m; u_0, s'_1, ..., s'_p)$, or $(m+1; s'_1, ..., s'_p)$, is normal. Now Corollary 2.14 guarantees that $(u_0, s'_1, ..., s'_p)$ is normal. According to whether u_0 equals Δ_n or not, we integrate the factor u_0 in Δ_n^m, and we obtain a greedy normal form, hence the greedy normal form of zu by uniqueness.

(ii) By commutativity, we have $zu^{-1} = \Delta_n^m u_0^{-1} s'_1 ... s'_p = \Delta_n^{m-1} {}^*u_0 s'_1 ... s'_p$, and the point is to check that the expected sequences are greedy normal forms. Corollary 2.8 guarantees that $(s'_1, ..., s'_p)$ is normal. So, if $u_0 = 1$ holds, $(m; s'_1, ..., s'_p)$ is a greedy normal form, hence it is the greedy normal form of zu^{-1}; otherwise, we observe that $(m-1; {}^*u_0, s'_1, ..., s'_p)$ is a greedy normal form, as $u_0 \neq 1$ implies ${}^*u_0 \neq \Delta_n$, and, by Lemma 2.11, the hypothesis $\gcd_L(u_0, s'_1) = 1$ is equivalent to $({}^*u_0, s'_1)$ being normal since $u_0 = ({}^*u_0)^*$ holds. □

Example 2.16. Let w_0 be the braid word `aBabacABABAbbCB`—the randomly chosen 4-braid word illustrated in Figure 1, which will be repeatedly considered in the sequel. Starting with $(0; \emptyset)$, which is the greedy normal form of 1, and applying the

algorithm of Proposition 2.15 to the successive letters of w, we obtain the 4-greedy normal form of the prefixes of w, namely (in braid words form):

$0 : \varepsilon \rightsquigarrow (0; \emptyset)$
$1 : \mathtt{a} \rightsquigarrow (0; \mathtt{a})$
$2 : \mathtt{aB} \rightsquigarrow (-1; \mathtt{abcb}, \mathtt{ba})$
$3 : \mathtt{aBa} \rightsquigarrow (-1; \mathtt{abcb}, \mathtt{ba}, \mathtt{a})$
$4 : \mathtt{aBab} \rightsquigarrow (-1; \mathtt{abcb}, \mathtt{ba}, \mathtt{ab})$
$5 : \mathtt{aBaba} \rightsquigarrow (0; \mathtt{a}, \mathtt{ab})$
$6 : \mathtt{aBabac} \rightsquigarrow (0; \mathtt{a}, \mathtt{abc})$
...
$14 : \mathtt{aBabacABABAbbC} \rightsquigarrow (-2; \mathtt{ac}, \mathtt{abcb}, \mathtt{bcba}, \mathtt{ab})$
$15 : \mathtt{aBabacABABAbbCB} \rightsquigarrow (-2; \mathtt{ac}, \mathtt{abcb}, \mathtt{bcba}, \mathtt{a})$.

Thus the greedy normal form of the braid represented by w is the sequence

$(-2; \mathtt{ac}, \mathtt{abcb}, \mathtt{bcba}, \mathtt{a})$,

i.e., in permutation form,

$(-2; (2,1,4,3), (2,4,3,1), (4,1,3,2), (2,1,3,4))$

—this is the greedy normal form of Example 1.5. As the initial word w_0 contains 15 letters, computing its greedy normal form entails 15 applications of Proposition 2.15. However, one can speed up the process by gathering adjacent letters that together represent a simple braid: for instance, abac represents a simple braid, so steps 3 to 6 above can be gathered into a single step corresponding to multiplying by the simple braid abac. By Corollary 1.4, we deduce that the braid word w_0 does not represent 1 in B_4.

As for complexity analysis, two cases are to be considered. If the braid index n is fixed, and, for practical implementations, has a small value, say $n \leqslant 6$, then one can precompute the tables of the binary operations $(s,t) \mapsto \alpha(st)$, $(s,t) \mapsto \alpha(st)^{-1}st$, and $(s,t) \mapsto \gcd_{\scriptscriptstyle L}(s,t)$, in which case the determination of the greedy normal form for a braid word of length ℓ has complexity $O(\ell^2)$, and is easy and quick in practice. Otherwise, there are too many simple n-braids to store all results, and one has to compute the values of $\alpha(st)$ and $\gcd_{\scriptscriptstyle L}(s,t)$ locally. As explained in [19, Chapter 9], this is essentially a sorting process, and, therefore, each such computation can be done in time $O(n \log n)$, resulting in a global time complexity $O(\ell^2 n \log n)$ for the computation of the normal form for an n-braid word of length ℓ.

2.4.2. *Computing the symmetric normal form.* We now consider the symmetric normal form of Section 1.3, and show how to compute the symmetric normal form of zu and zu^{-1} form that of z by using convenient grid diagrams. The method is similar to that for the greedy normal, but a bit more care is needed for the transition between the numerator and the denominator in the case of a product.

Proposition 2.17. *Assume that the symmetric normal form of z is the double sequence $(t_1, ..., t_q; s_1, ..., s_p)$, and u is a simple braid.*

(i) Let s'_p, u_{p-1}, ..., s'_1, u_0, v_0, v_1, t'_2, v_2, ... , t'_q, v_q be determined by filling the top diagram of Figure 14. Then the symmetric normal form of zu is $(t'_2, ..., t'_q, v_q; v_0^*, s'_1, ..., s'_p)$,

(ii) Let s'_p, u_{p-1}, ..., s'_1, $u_0 = v_0$, v_1, t'_2, ... , t'_q, v_q are determined by filling the bottom diagram of Figure 14. Then the symmetric normal form of zu^{-1} is $(t'_1, ..., t'_q, v_q; s'_1, ..., s'_p)$,

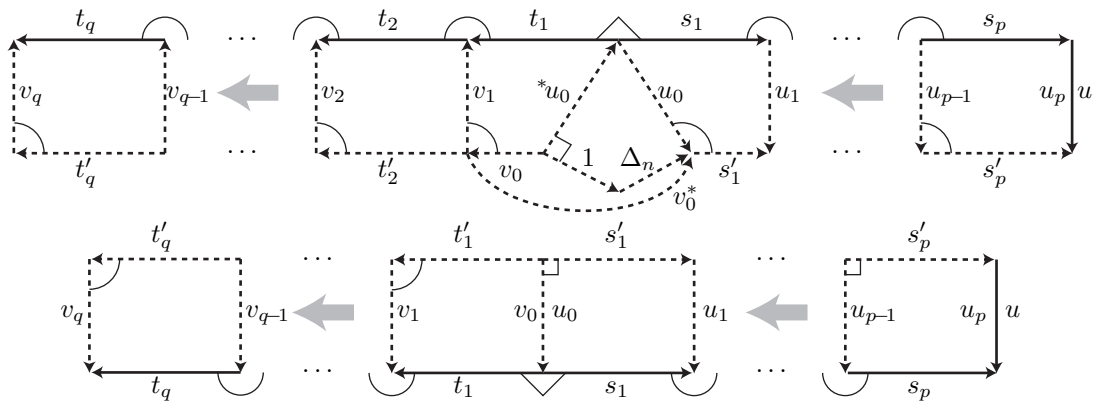


FIGURE 14. Symmetric normal form of zu (top) and zu^{-1} (bottom) from the symmetric normal form $(t_1, ..., t_q; s_1, ..., s_p)$ of z: start from $u_p := u$, and fill the diagram from right to left using P-tiles (*resp.* C-tiles, then P-tiles); in the case of the product, the transition is as follows: having got u_0, we find ${}^*u_0$ using a C-tile, then let (v_0, v_1) be the normal form of ${}^*u_0 t_1$ using a P-tile, and continue with v_1, t_2, *etc.*.; in the case of the quotient, we simply put $v_0 := u_0$, and continue with v_0, t_1, *etc.*

Proof. (*i*) By commutativity of the diagram, we have $zu = v_q^{-1} t_q'^{-1} ... t_2'^{-1} v_0^* s_1' ... s_p'$, and the point is to show that the double sequence $(t_2', ..., t_q', v_q; v_0^*, s_1', .., s_p')$ is a symmetric normal form. Corollary 2.14 guarantees that the sequences $(u_0, s_1', ..., s_p')$ and $(v_0, t_2', ..., t_q', v_q)$ are normal. There remain two points to check, namely that $\gcd_L(t_2', v_0^*)$ is 1, and that (v_0^*, s_1') is normal. For the first relation, Lemma 2.10 implies that (v_0, t_2') is normal, which is equivalent to $\gcd_L(t_2', v_0^*) = 1$ by Lemma 2.11.

As for the second relation, *i.e.*, for the normality of (v_0^*, s_1'), by Lemma 2.11 again, it is equivalent to $\gcd_L(v_0^{**}, s_1') = 1$, hence to $\gcd_L(v_0, \phi_n^{-1}(s_1')) = 1$. Now we have $v_0 v_1 = {}^*u_0 t_1$, and $s_1 u_1 = u_0 s_1'$, hence

$$ {}^*u_0 s_1 u_1 = \Delta_n s_1' = \phi_n^{-1}(s_1') \Delta_n = \phi_n^{-1}(s_1') {}^*u_1 u_1,$$

hence $\phi_n^{-1}(s_1') {}^*u_1 = {}^*u_0 s_1$. Assume s is a simple left divisor of v_0 and $\phi_n^{-1}(s_1')$. Then, *a fortiori*, we have $s \preccurlyeq_L v_0 v_1$ and $s \preccurlyeq_L \phi_n^{-1}(s_1') {}^*u_1$, hence, by the above computations, $s \preccurlyeq_L {}^*u_0 t_1$ and $s \preccurlyeq_L {}^*u_0 s_1$, and, therefore, $s \preccurlyeq_L {}^*u_0$ as, by hypothesis, $\gcd_L(s_1, t_1) = 1$ holds. Now, by hypothesis, (u_0, s_1') is normal, hence, by Lemma 2.11, we have $\gcd_L(u_0^*, s_1') = 1$, and therefore $\gcd_L(\phi_n^{-1}(u_0^*), \phi_n^{-1}(s_1')) = 1$, *i.e.*, $\gcd_L({}^*u_0, \phi_n^{-1}(s_1')) = 1$. So we must have $s = 1$, implying $\gcd_L(v_0, \phi_n^{-1}(s_1')) = 1$, which was seen above to be equivalent to (v_0^*, s_1') being normal. So the proof is complete.

(*ii*) The argument for the quotient is similar. By commutativity of the diagram, we have $zu^{-1} = v_q^{-1} t_q'^{-1} ... t_1'^{-1} s_1' ... s_p'$, and, once again, the point is to check that the double sequence $(t_1', ..., t_q', v_q; s_1', ..., s_p')$ is a symmetric normal form. Corollary 2.8 implies that $(s_1', ..., s_p')$ is normal, and Corollary 2.14 implies that $(t_1', ..., t_q', v_q)$ is normal, so the only remaining point to check is $\gcd_L(s_1', t_1') = 1$. Now assume that s is a simple left divisor of s_1' and t_1'. Then, *a fortiori*, we have $s \preccurlyeq_L s_1' u_1 = u_0 s_1$ and $s \preccurlyeq_L t_1' v_1 = u_0 t_1$. As $\gcd_L(s_1, t_1) = 1$ holds by hypothesis, we deduce $s \preccurlyeq_L u_0$, and, finally, $s = 1$ as, by construction, we have $\gcd_L(u_0, s_1') = 1$. Hence $\gcd_L(s_1', t_1') = 1$ holds. □

Example 2.18. Let w_0 be the braid word aBabacABABAbbCB again. Applying the algorithm of Proposition 2.17 to the successive letters of w leads to the symmetric normal forms (here in braid word form):

$0 : \varepsilon \rightsquigarrow (\emptyset; \emptyset)$
$1 : \mathtt{a} \rightsquigarrow (\emptyset; \mathtt{a})$
$2 : \mathtt{aB} \rightsquigarrow (\mathtt{ab}; \mathtt{ba})$
$3 : \mathtt{aBa} \rightsquigarrow (\mathtt{ab}; \mathtt{ba}, \mathtt{a})$
$4 : \mathtt{aBab} \rightsquigarrow (\mathtt{ab}; \mathtt{ba}, \mathtt{ab})$
$5 : \mathtt{aBaba} \rightsquigarrow (\emptyset; \mathtt{a}, \mathtt{ab})$
$6 : \mathtt{aBabac} \rightsquigarrow (\emptyset; \mathtt{a}, \mathtt{abc})$
...
$14 : \mathtt{aBabacABABAbbC} \rightsquigarrow (\mathtt{ab}, \mathtt{bacb}; \mathtt{bcba}, \mathtt{ab})$
$15 : \mathtt{aBabacABABAbbCB} \rightsquigarrow (\mathtt{ab}, \mathtt{bacb}; \mathtt{bcba}, \mathtt{a})$.

Thus the symmetric normal form of the braid represented by w_0 is the double sequence $(\mathtt{ab}, \mathtt{bacb}; \mathtt{bcba}, \mathtt{a})$, *i.e.*, in permutation form,

$((2,3,1,4),(3,4,1,2);(4,1,3,2),(2,1,3,4))$

—this is the symmetric normal form of Example 1.8. As in the case of the greedy normal form, we observe that the steps corresponding to a single simple factor can be gathered. We deduce from Corollary 1.7 that w_0 does not represent 1 in B_4.

The complexity analysis is the same as in the case of the greedy normal form: for a fixed braid index n, the algorithm of Proposition 2.17 is quadratic in the length of the initial word; when n is not fixed, a multiplicative factor $n \log n$ has to be inserted.

Remark. The diagrams of Figures 13 and 14 make it clear that the greedy and symmetric normal forms satisfy the fellow traveler property of [19] and therefore are connected with an automatic structure on the braid group B_n.

Also, let us point here that the greedy and symmetric normal forms exist for a class of structures that is much wider than braid groups, namely the so-called Garside groups of [16, 10], and even more: thin groups of fractions of [11], Garside categories of [23, 17, 4]—all eligible for the grid properties described above.

Finally, we mention the existence of alternative normal forms [6, 14] whose computation has the same complexity as the greedy and symmetric normal forms, but which rely on different bases and are connected with the braid order alluded to in Section 3.2 below. So far, there seems to be no reason to expect these normal forms to be more suitable for practical applications than those described above.

3. Direct solutions

Using a normal form is not the only way for solving the braid isotopy problem. Besides the solutions of Section 1, there exist alternative solutions directly deciding whether a given braid word w represents the trivial braid 1 or not.

Using such solutions entails working with arbitray braid words. As discussed in [13], this option makes the computation of product and inverse obvious (merely concatenating or reversing words), at the expense of making equivalence not obvious, whereas the option of using a normal form and restricting to normal words makes equivalence obvious (a mere equality), but makes the algebraic operations of product and inverse less obvious, as normalization processes such as those described in Propositions 2.13 and 2.7 are needed.

Here we describe three direct solutions to the braid isotopy problem, namely two syntactic solutions based on some word rewrite systems, and one geometric method due to I. Dynnikov which consists in attributing integral coordinates to every braid.

3.1. **Subword reversing.** Subword reversing—also simply called reversing in literature—is a simple syntactic transformation that consists in reversing some order of letters in some specific subwords so as to push the positive letters σ_i in one direction and the negative letters σ_i^{-1} in the other direction, until a word of the form "all positive, then all negative" is obtained. This leads to solutions of the word problem that admit a quadratic complexity. The underlying theory is Garside theory again.

3.1.1. *Description.* Redressing a braid word consists in looking for the subwords of the form $\sigma_i^{-1}\sigma_j$, *i.e.*, one negative letter followed by a positive letter, and transforming them into equivalent patterns consisting of positive letters followed by negative letters, *i.e.*, replacing negative–positive subwords with equivalent positive–negative words.

Definition 3.1. [8, 12] Assume that w, w' are braid words. We say that w is *right reversible* to w' in one step if w' is obtained from w either by deleting some subword of the form $\sigma_i^{-1}\sigma_i$, or by replacing some subword of the form $\sigma_i^{-1}\sigma_j$ with $|i-j| \geqslant 2$ by $\sigma_j\sigma_i^{-1}$, or by replacing some subword of the form $\sigma_i^{-1}\sigma_j$ with $|i-j| = 1$ by $\sigma_j\sigma_i\sigma_j^{-1}\sigma_i^{-1}$. We say that $w \curvearrowright w'$ holds if there is a finite sequence $w_0 = w$, ..., $w_N = w'$ such that w_k is right reversible to w_{k+1} in one step for each k.

By construction, the words that are terminal for subword reversing are the words that include no subword of the form $\sigma_i^{-1}\sigma_j$, *i.e.*, the words of the form $w'w''^{-1}$ with w', w'' positive braid words.

Theorem 3.2. [8] *For each braid word w, there exist two unique positive braid words w', w'' such that $w \curvearrowright w'w''^{-1}$ holds.*

A new solution to the braid isotopy problem follows:

Corollary 3.3. *A braid word w represents 1 in the braid group if and only if, denoting by w' and w'' the positive words for which $w \curvearrowright w'w''^{-1}$ holds, we have $w''^{-1}w' \curvearrowright \varepsilon$, where ε denotes the empty word.*

Example 3.4. Let us start again with the braid word $w_0 = \texttt{aBabacABABAbbCB}$ of Examples 2.16 and 2.18. Owing to Corollary 3.3, we decide whether w is trivial or not by reversing w to a word of the form $w'w''^{-1}$ with w', w'' positive words, then reversing $w''^{-1}w'$ again, and looking whether we finally obtain the empty word. In the current case, selecting at each step the leftmost pattern of the form $\sigma_i^{-1}\sigma_j$ (underlined in the words below), the successive words are as follows

0 : a<u>Ba</u>bacABABAbbCB
1 : aabA<u>Bb</u>acABABAbbCB
2 : aab<u>Aa</u>cABABAbbCB
3 : aabcABAB<u>Ab</u>bCB
4 : aabcABA<u>Bb</u>aBAbCB
...
11 : aabcbABBACB.

At this point, we switch the positive and negative subword, and reverse again:

12 : ABBAC<u>Ba</u>abcb

13 : ABBACabABabcb
14 : ABBAaCbABabcb
...
37 : cbaaBbcBCABBA.
38 : cbaacBCABBA.

The latter word is not empty: we conclude that w_0 does not represent 1 in B_4.

3.1.2. *Explanation.* Clearly, subword reversing replaces a braid word with an equivalent braid word, and it is easy to show that, if w reverses to a word of the form $w'w''^{-1}$ with w', w'' positive, then w' and w'' are uniquely determined. Now the problem is that, as Example 3.4 shows, reversing may increase the length of the words, and it is not obvious that the process terminates. The point is that, if w has the form $w_1^{-1}w_2$ where w_1, w_2 are positive words representing simple braids, then w reverses to some word $w'w''^{-1}$ where w' and w'' are positive and again represent simple braids. Thus, with respect to an enhanced alphabet containing all positive braid words representing simple braids, the length does not increase under reversing, and this leads to the termination result. Actually, what reversing does is to compute the right lcm in the braid monoid, and so the properties behind subword reversing are the Garside theory.

3.1.3. *Discussion.* The advantage of the reversing method is the simplicity of its implementation: there is a unique syntactic operation, involving length 2 subwords of the considered word only. From that point of view, the method is more easily implemented than the greedy or symmetric normal form. At a theoretical level, both methods are essentially equivalent: there exist positive constants C, C' such that, for each n-braid word w, if $N_1(w)$ (*resp.* $N_2(w)$) denotes the number of braid relations needed to put w into a greedy normal form (*resp.* to reverse w), then we have $CN_1(w) \leqslant N_2(w) \leqslant C'N_1(w)$. However, by representing simple braids by permutations and using fast sorting algorithms to compute the normal form as explained in [19, Chapter 9], one presumably obtains a more efficient algorithm.

According to Theorem 3.2, the reversing method starting from a braid word w yields a final braid word w' such that $w \equiv \varepsilon$ is equivalent to $w' = \varepsilon$. However, in general, $w' \equiv w$ fails: because of the exchange of the negative and positive factors between the two passes, w' is only equivalent to a conjugate of w. Now, it is easy to describe a variant of the reversing method that avoids the median conjugation. Indeed, let *left subword reversing*, denoted $w \curvearrowright w'$, be the symmetric counterpart to (right) reversing consisting in replacing each pattern $\sigma_i\sigma_j^{-1}$ with an equivalent negative–positive word. Formally, we may define $w \curvearrowright w'$ to mean $\widetilde{w} \curvearrowright \widetilde{w'}$, where $\widetilde{w}$ is the word obtained from w by reversing the order of the letters.

Corollary 3.5. [8] *A braid word w represents 1 in the braid group if and only if, denoting by w' and w'' the positive words for which $w \curvearrowright w'w''^{-1}$ holds, we have $w'w''^{-1} \curvearrowright \varepsilon$.*

So the method is the same as in Corollary 3.3, with the only difference that we do not change the word obtained at the end the first reversing pass, but instead continue with left reversing. The criterion remains the same, namely that the initial word w is trivial if and only if the final word w'' is empty. The advantage of this variant is that w'' is equivalent to w, and, moreover, it gives a fractionary decomposition of $\overline{w}$ which is geodesic, *i.e.*, has minimal length among all fractionary

expressions of $\overline{w}$. This implies that the negative and the positive parts of w'' must be equivalent to the two components of the symmetric normal form of $\overline{w}$, although they need not be in normal form in general. Assuming that an algorithm for computing the normal form of a positive braid is available, one obtains in this way an alternative way for computing the symmetric normal form of an arbitrary braid that is more simply implemented than the method of Proposition 2.17: perform double subword reversing, then put the numerator and the denominator in normal form.

Example 3.6. Starting with `aBabacABABAbbCB` once more, the first 11 steps of the subword reversing algorithm are as in Example 3.4, but, then, we appeal to left reversing, and the sequel is different. We find:

0 : a<u>Ba</u>bacABABAbbCB
...
11 : aabc<u>bA</u>BBACB
12 : aab<u>cA</u>BabBBACB
13 : aa<u>bA</u>cBabBBACB
14 : a<u>aA</u>BabcBabBBACB
...
39 : BACBBAcbaac<u>bB</u>.
40 : BACBBAcbaac.

Once again, the latter word is not empty, and we conclude that w_0 does not represent 1 in B_4. In addition, we obtain that `BACBBAcbaac` is a shortest expression of $\overline{w_0}$ as a negative–positive fraction, which is coherent with Example 1.8, where it was shown that the symmetric normal form of $\overline{w_0}$ is $(\mathtt{ab}, \mathtt{bacb}, \mathtt{bcba}, \mathtt{a})$: indeed, $(\mathtt{ab}, \mathtt{bacb})$ is the normal form of `abbcab`, and $(\mathtt{bcba}, \mathtt{a})$ is the normal form of `cbaac`.

As a final remark, let us observe that, because reversing is efficient at computing lcm's and complements in the braid monoid, it also provides an easy way to compute gcd's and, from there, normal forms. Though not as efficient as those based on quick sorting, the algorithms based on subword reversing are more easily implemented than the latter, and they are convenient for small and medium size braid words.

3.2. **Handle reduction.** Handle reduction is another syntactic braid word transformation which, like subword reversing, consists in iterating some basic word transformation and concluding that the initial braid word represents 1 if and only if the final word is empty. The basic transformation step is more complicated than the one involved in reversing but the number of steps is much lower, and the method turns out to be extremaly efficient in practice. The underlying structure behind handle reduction is a linear ordering of braids, which pilots the reduction process and heuristically explains its efficiency.

3.2.1. *Description.* Handle reduction is an extension of free reduction. The latter consists in iteratively deleting patterns of the form xx^{-1} or $x^{-1}x$. Handle reduction involves not only patterns of the form $\sigma_i\sigma_i^{-1}$ or $\sigma_i^{-1}\sigma_i$, but also more general patterns of the form $\sigma_i \dots \sigma_i^{-1}$ or $\sigma_i^{-1} \dots \sigma_i$ with intermediate letters between the letters σ_i and σ_i^{-1}.

Definition 3.7. (*i*) A *σ_i-handle* is a braid word of the form

$$w = \sigma_i^e\, w_0\, \sigma_{i+1}^d\, w_1\, \sigma_{i+1}^d \dots \sigma_{i+1}^d\, w_m\, \sigma_i^{-e}, \tag{3.1}$$

with $e, d = \pm 1$, $m \geqslant 0$, and $w_0, ..., w_m$ containing no $\sigma_j^{\pm 1}$ with $j \leqslant i+1$. Then the *reduct* of w is defined to be

$$w' = w_0\, \sigma_{i+1}^{-e} \sigma_i^d \sigma_{i+1}^e\, w_1\, \sigma_{i+1}^{-e} \sigma_i^d \sigma_{i+1}^e \dots \sigma_{i+1}^{-e} \sigma_i^d \sigma_{i+1}^e\, w_m, \tag{3.2}$$

i.e., we delete the initial and final letters $\sigma_i^{\pm 1}$, and we replace each letter $\sigma_{i+1}^{\pm 1}$ with $\sigma_{i+1}^{-e} \sigma_i^{\pm 1} \sigma_{i+1}^e$.

(*ii*) We say that a braid word w is *reduced* if it contains no σ_i-handle, where σ_i is the generator with minimal index occurring in w.

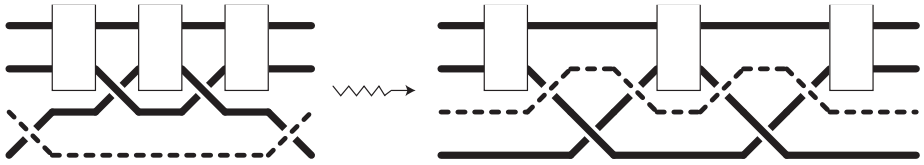


FIGURE 15. Reduction of a handle, here a σ_1-handle: the dashed strand has the shape of a handle, and reduction consists in pushing that strand so that it skirts above the next crossings instead of below.

A braid word of the form $\sigma_i \sigma_i^{-1}$ or $\sigma_i^{-1} \sigma_i$ is a handle, and reducing it means deleting it, so handle reduction generalizes free reduction. As illustrated in Figure 15, reducing a handle yields an equivalent braid word. So, as in the case of free reduction, if there is a reduction sequence from a braid word w to the empty word, *i.e.*, a sequence $w = w_0, ..., w_N = \varepsilon$ such that, for each k, the word w_{k+1} is obtained from w_k by replacing some handle of w_k by its reduct, then w represents 1 in the braid group. The point is that the converse is also true.

Theorem 3.8. [9] *For every braid word w, every sequence of handle reductions from w leads in finitely many steps to a reduced word w'. Moreover, a reduced word w' represents* 1 *if and only if it is empty.*

We obtain a new solution to the braid isotopy problem:

Corollary 3.9. *A braid word w represents* 1 *in the braid group if and only if any sequence of handle reductions starting from w terminates with the empty word.*

Example 3.10. Consider $w_0 =$ `aBabacABABAbbCB` again. Choosing to reduce the leftmost handle at each step and underlying it, we succesively obtain:

0 : `aBabacABABAbbCB`
1 : `aBabcBABAbbCB`
2 : `aBaCbcABAbbCB`
3 : `aBCBabcBAbbCB`
4 : `aBCBaCbcAbbCB`
5 : `aBCBCBabcbbCB`.

The latter word contains no σ_1-handle, so it is reduced, and it is not empty, so we conclude that w_0 does not represent 1 in B_4.

3.2.2. *Explanation.* Two different structures lie behind handle reduction, namely Garside's theory that was already involved on the previous solutions, and, in addition, some linear order that is compatible with left multiplication on B_n. It can be seen that each handle reduction is essentially a composition of reversing steps,

but the main difference with the algorithms of Sections 1 and 3.1 is that, here, we do not perform all reversing steps systematically, but only some of them according to a general strategy provided by the underlying braid order. This should make it natural why the handle reduction method is, in practice, much more efficient than the reversing method and than the greedy and symmetric normal form methods (5 *vs.* 40 steps in our example).

3.2.3. *Discussion.* A braid word may contain many handles, so building an actual algorithm requires to fix a strategy prescribing in which order the handles will be reduced. In Example 3.10, we chose to reduce the leftmost handle, but more efficient strategies exist. As can be expected, the most efficient ones use a divide-and-conquer trick. Although the only upper bound for space and time complexity proved so far is exponential, handle reduction is extremely efficient in practice, as show the statistics of [13]. Also, reduction being a local procedure, the amount of memory needed to implement it is what is needed to just store the braid under reduction. So, using arbitrary words together with handle reduction instead of normal words could be specially interesting when the computing resources are limited.

It was mentioned above that the symmetric normal form, as well as the subword reversing method, yield fractionary expressions of minimal length. Such fractionary expressions need not be expressions of minimal length: there may be shorter expressions that are not fractions, *i.e.*, in which all negative letters are not gathered in one block and all positive letters in another block. The general problem of finding the shortest expression of a braid is difficult: its B_∞ version is known to be NP-complete [25]. Although no actual result is proved, it has been observed that handle reduction is good at providing short expressions. Actually, the definition of handle reduction is not symmetric and the left side plays a distinguished rôle. One can compensate this lack of symmetry by defining *iterated handle reduction* as follows: starting with w, we reduce w to w', then flip w' using ϕ_n, reduce it, and flip the result. Equivalently, after one handle reduction, one performs the symmetric operation in which the right side is distinguished. By doing so, and possibly iterating the process, one empirically obtains short expressions. It is conjectured by the author, as well as by A. Miasnikov and A. Ushakov, that there might exist a constant C such that applying the previous process to any braid word w leads to a final word of length at most $C\ell_{\min}(\overline{w})$, where, for x a braid, $\ell_{\min}(x)$ denotes the length of the shortest word representing x.

Example 3.11. Starting once more with $w_0 = \texttt{aBabacABABAbbCB}$, iterated handle reduction from w_0 leads in 3 steps to the word `acBCCBa`, which happens to be a geodesic representative of the braid $\overline{w_0}$.

3.3. **Dynnikov coordinates.** We conclude with still another solution to the braid isotopy problem, namely the one provided by the so-called Dynnikov coordinates. This solution relies on a completely different approach steming from geometry and deep results by L. Mosher about the existence of an automatic structure for all mapping class groups [24].

3.3.1. *Description.* The principle consists in associating with every n-braid word a sequence of $2n$ integers, that can be thought of as coordinates for the braid. This coordinization is faithful in that two braid words receive the same coordinates if and only if they represent the same braid.

Definition 3.12. (*i*) For x in $\mathbb{Z}$, write x^+ for $\max(0,x)$, and x^- for $\min(x,0)$. Let $F, \overline{F} : \mathbb{Z}^4 \to \mathbb{Z}^4$ be defined by $F = (F_1, ..., F_4)$, $\overline{F} = (\overline{F}_1, ..., \overline{F}_4)$ with

$$\begin{cases} F_1(x_1,y_1,x_2,y_2) &:= x_1 + y_1^+ + (y_2^+ - z_1)^+, \\ F_2(x_1,y_1,x_2,y_2) &:= y_2 - z_1^+, \\ F_3(x_1,y_1,x_2,y_2) &:= x_2 + y_2^- + (y_1^- + z_1)^-, \\ F_4(x_1,y_1,x_2,y_2) &:= y_1 + z_1^+, \\ \overline{F}_1(x_1,y_1,x_2,y_2) &:= x_1 - y_1^+ - (y_2^+ + z_2)^+, \\ \overline{F}_2(x_1,y_1,x_2,y_2) &:= y_2 + z_2^-, \\ \overline{F}_3(x_1,y_1,x_2,y_2) &:= x_2 - y_2^- - (y_1^- - z_2)^-, \\ \overline{F}_4(x_1,y_1,x_2,y_2) &:= y_1 - z_2^-, \end{cases} \tag{3.3}$$

where we put $z_1 := x_1 - y_1^- - x_2 + y_2^+$ and $z_2 := x_1 + y_1^- - x_2 - y_2^+$.

(*ii*) For $(a_1, b_1, ..., a_n, b_n)$ in $\mathbb{Z}^{2n}$, put

$$(a_1, b_1, ..., a_n, b_n) \cdot \sigma_i^e = (a'_1, b'_1, ..., a'_n, b'_n) \tag{3.4}$$

with $a'_k = a_k$ and $b'_k = b_k$ for $k \neq i, i+1$, and

$$(a'_i, b'_i, a'_{i+1}, b'_{i+1}) = \begin{cases} F(a_i, b_i, a_{i+1}, b_{i+1}) & \text{for } e = +1, \\ \overline{F}(a_i, b_i, a_{i+1}, b_{i+1}) & \text{for } e = -1. \end{cases}$$

Then, for w an n-braid word, we recursively define

$$(a_1, b_1, ..., a_n, b_n) \cdot w = \begin{cases} (a_1, b_1, ..., a_n, b_n) & \text{for } w = \varepsilon, \\ ((a_1, b_1, ..., a_n, b_n) \cdot w') \cdot \sigma_i^e & \text{for } w = w'\sigma_i^e. \end{cases}$$

The *Dynnikov coordinates* of w are defined to be the sequence $(0,1,0,1,...,0,1) \cdot w$.

Theorem 3.13. [15, Propositions 8.5.3 and 8.5.4] *The Dynnikov coordinates of an n-braid word w characterize the braid $\overline{w}$ represented by w: the coordinates of w and w' are equal if and only if $\overline{w} = \overline{w'}$ holds.*

We deduce still another solution to the braid isotopy problem:

Corollary 3.14. *An n-strand braid word represents 1 in B_n if and only if its Dynnikov coordinates are $(0,1,0,1,...,0,1)$.*

Example 3.15. Consider $w_0 =$ `aBabacABABAbbCB` once more. To compute the Dynnikov coordinates of the 4-braid represented by w_0, we start with the sequence $(0,1,0,1,0,1,0,1)$, and we apply the formulae of (3.3) and (3.4) with the successive letters of w_0, *i.e.*, we compute the Dynnikov coordinates of the braids represented by the successive prefixes of w_0:

$0 : \varepsilon \rightsquigarrow (0,1,0,1,0,1,0,1)$

$1 :$ `a` $\rightsquigarrow (1,0,0,2,0,1,0,1)$

by applying (3.4) with $i = 1$ and $e = +1$ to the previous sequence

$2 :$ `aB` $\rightsquigarrow (1,0,-2,0,0,3,0,1)$

by applying (3.4) with $i = 2$ and $e = -1$ to the previous sequence, etc.

$3 :$ `aBa` $\rightsquigarrow (1,-3,-2,3,0,3,0,1)$

$4 :$ `aBab` $\rightsquigarrow (1,-3,3,2,0,4,0,1)$

$5 :$ `aBaba` $\rightsquigarrow (1,-1,3,0,0,4,0,1)$

...

$14 :$ `aBabacABABAbbC` $\rightsquigarrow (1,-7,5,-1,-7,4,0,8)$

$15 : \texttt{aBabacABABAbbCB} \rightsquigarrow (1, -7, -6, 4, 1, -1, 0, 8)$.
The latter coordinates are not $(0, 1, 0, 1, 0, 1, 0, 1)$, so we conclude that w_0 does not represent 1 in B_4.

3.3.2. *Explanation.* At first, the formulae (3.3) seem quite mysterious. Actually, there is no miracle here, but a very clever use of the simple formula

$$x + x' = \max(x_1 + x_3, x_2 + x_4) \tag{3.5}$$

that compares the number of intersections of a family of curves with two triangulations obtained one from the other by switching one diagonal in a quadrilateral (flip transformation), as shown in Figure 16.

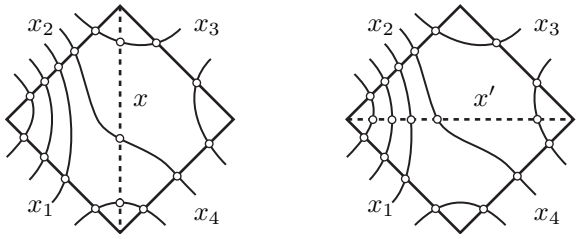
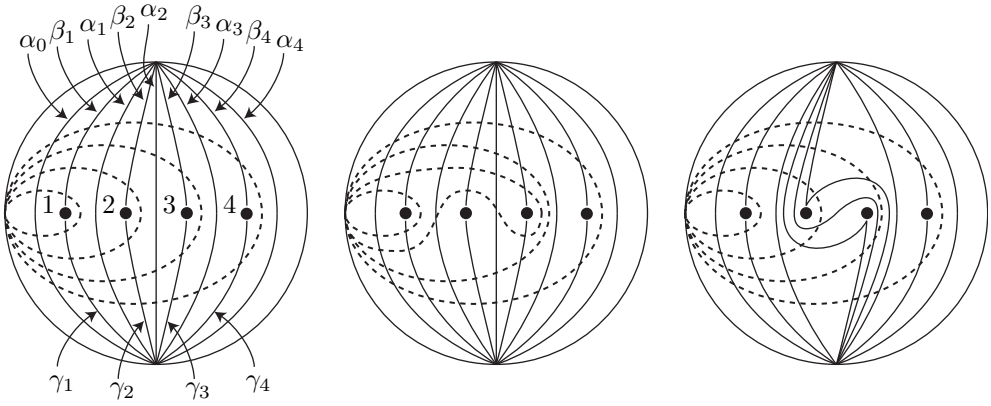


FIGURE 16. Action of a flip on intersections of a triangulation with a family of normal curves: we count how many curves intersect each edge, and compare the numbers when the diagonal is flipped; the connection between the new numbers and the old numbers is given by (3.5); in the current case, we have $x_1 = 4$, $x_2 = 5$, $x_3 = 2$, $x_4 = 4$, $x = 3$, $x' = 5$, and (3.5) corresponds to the equality $3 + 5 = \max(4 + 2, 5 + 3)$; the hypothesis that the curves are normal means that they are tangent to no edge of the triangulation, and they form no digon with them.

The framework consists in considering an n-braid as the isotopy class of a homeomorphism of a disk with n punctures [5, 21]: then the generator σ_i corresponds to the homeomorphism that exchanges the ith and $(i + 1)$st punctures by a half-turn. Dynnikov's idea is to let the braid act on a particular family of curves in the punctured disk (lamination), and to count their intersections with a fixed triangulation—see Figure 17. Applying (3.5) repeatedly leads to the mysterious formulae (3.3). Note that the Dynnikov formulae can be used as a black box: once the correct formulae have been guessed, one can check by a direct computation that they are compatible with the braid relations, *i.e.*, that applying $\sigma_1\sigma_2\sigma_1$ and $\sigma_2\sigma_1\sigma_1$ to any sequence of numbers leads to the same result, and then the faithfulness result of Theorem 3.13 follows from the remark that the sequence associated with a braid that can be expressed using a word that contains the letter σ_1 but not the letter σ_1^{-1} cannot be $(0, 1, 0, 1, 0, 1, ...)$ and the properties of the so-called Dehornoy ordering of braids—see [15] for details.

3.3.3. *Discussion.* The remarkable point about the Dynnikov coordinates is that they involve the semiring $(\mathbb{Z}, \max, +, 0)$, which explains their efficiency. Multiplying by one generator σ_i can only increase the size of the coordinates by one unit, whereas similar formulae in the ring $(\mathbb{Z}, +, \times, 1)$ would double the size in the worst case. It follows that the solution to the braid word problem given by Theorem 3.13 has a linear space complexity, and a quadratic time complexity.



initial configuration after applying σ_2 to L after applying σ_2^{-1} to T

FIGURE 17. Origin of the Dynnikov formulae: we count the intersections of a lamination L—here the four dotted curves—with the edges of a prescribed triangulation T—here the 13 plain arcs denoted $\alpha_0, ..., \alpha_4, \beta_1, ..., \beta_4, \gamma_1, ..., \gamma_4$—thus getting a sequence of natural numbers; when we apply a braid x, viewed as a homeomophism of the punctured disk, to L, the lamination changes, and so do the intersection numbers (central picture); now the new intersection numbers can be expressed in terms of the old ones and of the braid x; to see that, instead of applying x to the lamination L and keeping the triangulation T fixed, we can equivalently keep L unchanged and apply x^{-1} to T (right picture); then the transformation of the triangulation corresponding to applying σ_i is easily decomposed into the composition of four flips, and the formulae of (3.3) follow from (3.5) when one defines $a_i := (\beta_i - \gamma_i)/2$ and $b_i := (\alpha_i - \alpha_{i-1})/2$.

It is not so easy to compare the solution based on the Dynnikov coordinates with the other solutions to the braid isotopy problem, because its practical efficiency much depends on the way large integer arithmetic is implemented—large integers do appear when long braid words are considered, typically length $O(\ell)$ binary integers for a length ℓ braid word representing a pseudo-Anosov braid. However, the only arithmetic operations involved are addition and maximum, and both can be implemented very efficiently and easily. No statistical study has been completed so far, and it would be desirable to compare Dynnikov's method with handle reduction. The only weak point of the former is that, at the moment, it is purely incremental: the only available formulae correspond to multiplying by one single letter σ_i or σ_i^{-1}, so, in particular, no divide-and-conquer variant exists.

REFERENCES

[1] S.I. Adyan, *Fragments of the word Delta in a braid group*, Mat. Zam. Acad. Sci. SSSR **36-1** (1984) 25–34; translated Math. Notes of the Acad. Sci. USSR; 36-1 (1984) 505–510.
[2] I. Anshel, M. Anshel, & D. Goldfeld, *An algebraic method for public-key cryptography*, Math. Research Letters **6** (1999) 287–291.
[3] E. Artin, *Theory of Braids*, Ann. of Math. **48** (1947) 101–126.
[4] D. Bessis, *Garside categories, periodic loops and cyclic sets*, Preprint; math.GR/0610778.
[5] J. Birman, *Braids, Links, and Mapping Class Groups*, Annals of Math. Studies **82** Princeton Univ. Press (1975).
[6] S. Burckel, *The wellordering on positive braids*, J. Pure Appl. Algebra **120-1** (1997) 1–17.

[7] R. Charney, *Artin groups of finite type are biautomatic*, Math. Ann. **292-4** (1992) 671–683.
[8] P. Dehornoy, *Groups with a complemented presentation*, J. Pure Appl. Algebra **116** (1997) 115–137.
[9] P. Dehornoy, *A fast method for comparing braids*, Advances in Math. **125** (1997) 200–235.
[10] P. Dehornoy, *Groupes de Garside*, Ann. Scient. Ec. Norm. Sup. **35** (2002) 267–306.
[11] P. Dehornoy, *Thin groups of fractions*, Contemp. Math. **296** (2002) 96–129.
[12] P. Dehornoy, *Complete positive group presentations*, J. of Algebra **268** (2003) 156–197.
[13] P. Dehornoy, *Braid-based cryptography*, Contemp. Math. **360** (2004) 5–33.
[14] P. Dehornoy, *Alternating normal forms for braids and locally Garside monoids*, Preprint; math.GR/0702592.
[15] P. Dehornoy, I. Dynnikov, D. Rolfsen, B. Wiest, *Why are braids orderable?*, Panoramas & Synthèses vol. 14, Soc. Math. France (2002).
[16] P. Dehornoy & L. Paris, *Gaussian groups and Garside groups, two generalizations of Artin groups*, Proc. London Math. Soc. **79-3** (1999) 569–604.
[17] F. Digne & J. Michel, *Garside and locallly Garside categories*, Preprint; math.GR/0612652.
[18] E. A. Elrifai & H.R. Morton, *Algorithms for positive braids*, Quart. J. Math. Oxford **45-2** (1994) 479–497.
[19] D. Epstein, J. Cannon, D. Holt, S. Levy, M. Paterson & W. Thurston, *Word Processing in Groups*, Jones & Bartlett Publ. (1992).
[20] F. A. Garside, *The braid group and other groups*, Quart. J. Math. Oxford **20-78** (1969) 235–254.
[21] C. Kassel & V. Turaev, *Braid groups*, Springer (2007).
[22] K.H. Ko, S.J. Lee, J.H. Cheon, J.W. Han, J.S. Kang, & C. Park, *New public-key cryptosystem using braid groups*, Crypto 2000; Springer Lect. Notes in Comput. Sci., 1880 (2000) 166–184.
[23] D. Krammer, *A class of Garside groupoid structures on the pure braid group*, Trans. Amer. Math. Soc., to appear.
[24] L. Mosher, *Mapping class groups are automatic*, Ann. of Math. **142** (1995) 303–384.
[25] M.S. Paterson & A.A. Razborov, *The set of minimal braids is co-NP-complete*, J. Algorithms **12–3** (1991) 393–408.
[26] W. Thurston, *Finite state algorithms for the braid group*, Circulated notes (1988).

Laboratoire de Mathématiques Nicolas Oresme, UMR 6139 CNRS, Université de Caen BP 5186, 14032 Caen, France
E-mail address: dehornoy@math.unicaen.fr
URL: //www.math.unicaen.fr/∼dehornoy