EXPLORATIONS IN BRAID THEORY: AN OVERVIEW

SYLVIA ANDREAE

Contents

1. Introduction and Acknowledgments

This paper is an overview of some of the material I learned about braid theory during my independent study of the subject. It covers the basics of braid theory, including the traditional Artin and the more recent Birman, Ko, Lee presentations of the braid group. This is followed by an overview of the word and conjugacy problems for braids. Applications are given, including how the word and conjugacy problems relate to cryptography with braids. Finally, there is a tangent into a braid related "puzzle" on a collection of braids I call closed-ended twist braids. I would like to thank Dr. Boden for his time at weekly discussions about braid theory. His guidance was most helpful for my understanding of the topic.

2. Braids and the Artin presentation

While braids have been around for as long as humans have woven strands together, the mathematical study of braids, or braid theory, was first developed by Emil Artin in 1947 [1]. Artin defined braids as curves (strands) in 3-space that travel from the points $(l, 0, 1)$ to $(m, 0, 0)$ with l,$m \in \mathbf{N}$, possibly different, and only one curve at each point. Further, each curve crosses any horizontal plane only once (ie. they do not double back on themselves). Examples of a braid and a non-braid are in Figure 1.

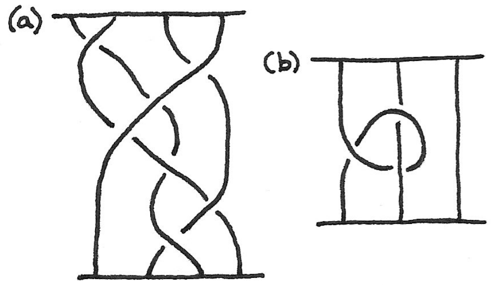


FIGURE 1. (a) A braid. (b) Something that is not a braid because one of the strands doubles back on itself.

The set of braids with n strands is denoted B_n and is a group with the operation of concatenation (connecting them end to end). Associativity can be seen in Figure 2(a), with the concatenation $(ab)c = a(bc)$. The braid of n strands where each strand goes from $(i, 0, 1)$ to $(i, 0, 0)$ without crossing is the identity (See Figure 2(b)). The inverse of a braid is given by its vertical mirror image, as in Figure 2(c).

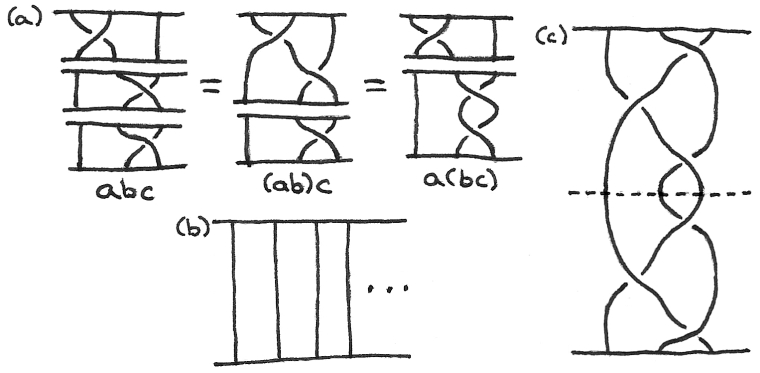


FIGURE 2. The structure of the braid group.

Artin defined generators of the braid group and relations between the generators. This allows every braid to be written out as a braid word, or series, of generators. The generators σ_i and σ_i^{-1} are shown in Figure 3. Note that the generators of B_n consist of σ_1 through σ_{n-1} and their inverses.

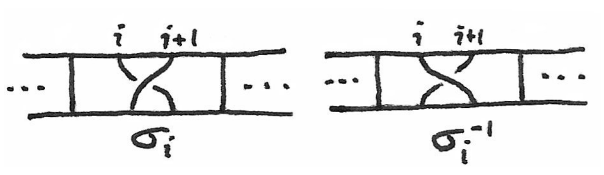


FIGURE 3. The Artin generators.

For this presentation, three more properties must be satisfied by the braid in order for it to be transcribed as at word.

1. No strands can be tangent to each other at any point.
2. Only two strands can cross at any point.
3. No two crossings can occur at the same horizontal level.

The first two can be satisfied by spreading strands out horizontally, and the third merely requires one crossing to be shifted up or down slightly in relation to another. With these satisfied, the braid is transcribed from top to bottom, recording each crossing as the appropriate σ_i or σ_{i-1}. It is worth noting that not all papers are consistent about braid notation. Some papers draw braids from left to right [6], or with the left strand over the right being the positive generator [3]. Here I will follow the majority of papers and draw braids from top to bottom, with a positive generator having the right strand cross over the left.

There are three basic relations for braids with this presentation: the trivial relation, far commutativity, and the braid relation. The trivial relation uses the inverse element to cancel out a positive element as seen in Figure 4(a).

$$\sigma_i\sigma_i^{-1} = \sigma_i^{-1}\sigma_i \qquad \text{(trivial relation)}$$

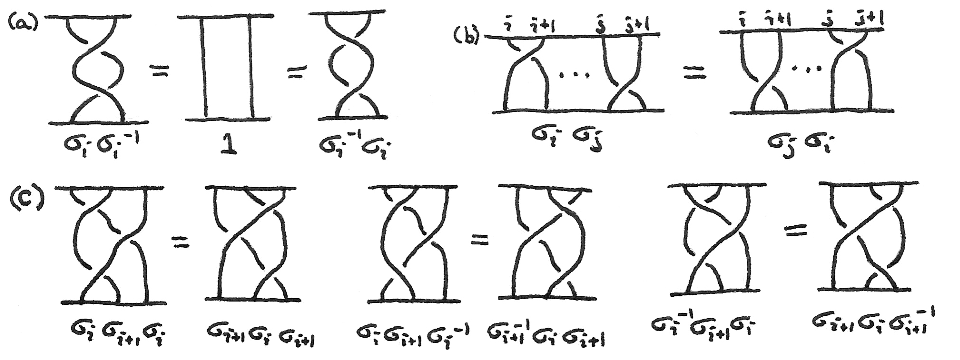
Far commutativity is the property that two generators that are sufficiently far away from each other (i.e. don't share a strand) can be commuted as in Figure 4(b).

$$\sigma_i\sigma_j = \sigma_j\sigma_i \qquad \text{when } |i-j| \leq 2 \qquad \text{(far commutativity)}$$

Note that braid elements in general are not commutative, so the braid group is not abelian. The braid relation has multiple forms. One is

$$\sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} \qquad \text{(braid relation)}$$

and others that involve inverse generators are shown in Figure 4(c). Three other relations can be made by the inverses of the braid relations shown. Artin proved that these basic relations are enough to characterize the entire braid group B_n.



(a)
=
=
$\sigma_i\sigma_i^{-1}$
1
$\sigma_i^{-1}\sigma_i$
(b)
i $i+1$ j $j+1$
=
i $i+1$ j $j+1$
$\sigma_i\sigma_j$
$\sigma_j\sigma_i$
(c)
=
=
=
$\sigma_i\sigma_{i+1}\sigma_i$
$\sigma_{i+1}\sigma_i\sigma_{i+1}$
$\sigma_i\sigma_{i+1}\sigma_i^{-1}$
$\sigma_{i+1}^{-1}\sigma_i\sigma_{i+1}$
$\sigma_i^{-1}\sigma_{i+1}\sigma_i$
$\sigma_{i+1}\sigma_i\sigma_{i+1}^{-1}$

FIGURE 4. The Artin relations.

3. The Birman, Ko, and Lee Presentation

In 1998, Joan Birman, Ki Hyoung Ko, and Sang Jin Lee published a new presentation of the braid group B_n using a larger set of generators called *band generators* [2]. Like Artin's generators, each band generator is a crossing of two strands, but instead of restricting the crossings to adjacent strands, arbitrary strands can be crossed over each other in front of all strands in between. A band generator crossing strands t and s with $t > s$, can be represented using Artin generators as follows

$$a_{t,s} = (\sigma_{t-1}\sigma_{t-2}\dots\sigma_{s+1})\sigma_s(\sigma_{s+1}^{-1}\sigma_{s+2}^{-1}\dots\sigma_{t-1}^{-1})$$

and is depicted in Figure 5. Note that for adjacent strands $s-1$ and s, $a_{s,s-1} = \sigma_s$, so the Artin generators are a subset of the band generators.

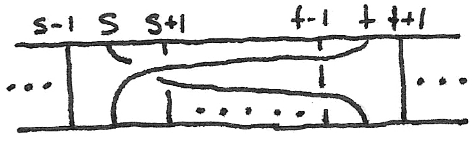


FIGURE 5. A band generator.

Basic relations for the positive band generators are

$a_{t,s}a_{r,q} = a_{r,q}a_{t,s}$ if $(t-r)(t-q)(s-r)(s-q) > 0$ (far and nested commutativity)

$a_{t,s}a_{s,r} = a_{t,r}a_{t,s} = a_{s,r}a_{t,r}$ if $1 \le r < s < t \le n$ (partial commutativity)

and generate the same braid group as the Artin generators, as proved in [2]. Visual representations of the band generator relations are shown in Figure 6.

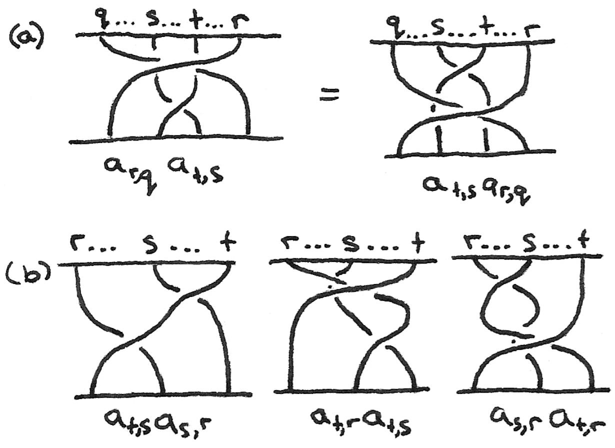


FIGURE 6. (a) Nested commutativity. (b) Partial commutativity.

It is useful to denote braids containing only positive generators as *positive braids.* The set of positive braids, B_n^+, is not a group since only the identity has an inverse. Just as there are more band generators than Artin generators, the set of positive braids in the band generators is larger than the set of positive braids in the Artin generators.

4. The Symmetric Group and Braids

There is a map from the braid group B_n to the symmetric group S_n given by the permutation of a braid's strands. Specifically, the starting point of each strand is permuted to its end point. This map is not one-to-one, since many different braids map on to the same element of S_n. For example, in B_3, the braid $\sigma_1\sigma_2^{-1}$ maps to the k-cycle (132), with strand 1 going to strand 3, 3 going to 2 and 2 going to 1. But the braid $\sigma_2\sigma_2\sigma_1\sigma_2$ goes to the same permutation, as shown in Figure 7.

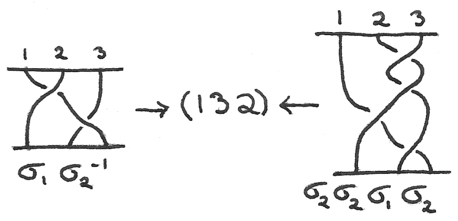


Figure 7. Two braids mapping to the same permutation.

The braids that are in the kernel of this map, being those with strands starting and ending in the same order, are called pure braids and form a subgroup of B_n denoted by P_n.

5. The Word Problem

A series of braid generators is called a braid word. Braid words are not unique since they can be changed by the group relations without changing the braid element. Artin proposed the word problem for braids: find an algorithm to determine whether two given braid words represent the same braid. While Artin gave a solution, an improved solution was later proposed by Garside in [5].

It might be natural to think that a simple way to see if two words are the same is to invert one word, concatenate the two, and determine whether the result is the identity braid. Unfortunately, it turns out that this procedure is not very efficient. Therefore, most solutions for the word problem rest on finding a normal form for braids, and then simply comparing the two normalized braids letter by letter.

5.1. **Greedy Normal Form.**

Garside's normal braid form, a common variation of which is called the *greedy normal form*, uses a *fundamental braid* Δ_n that looks like a half twist on n strands, and is defined inductively:

$$\Delta_1 = 1, \qquad \Delta_{n+1} = \Delta_n(\sigma_n\sigma_{n-1}\dots\sigma_1)$$

Δ_4 is shown in Figure 8 as an example.

The fundamental braid has the special property that it can be written beginning (or ending) with any generator while remaining a positive braid. A proof of this can be found in [2].

The general idea behind Garside's normal form is that the inverse of the fundamental braid is used to "twist" all negative generators to the front of the braid, and then write the remaining positive braid in a consistent way. Notationally, we have a braid b of B_n with a normal form $b = \Delta_n^k c$, where k is an integer (either 0 or negative), and c is a positive braid. Garside noted that the positive braids do

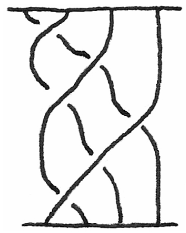


FIGURE 8. The fundamental braid Δ_4.

not change in length under the relations, and so there are only a finite number of ways that a positive braid can be written.

The greedy normal form improves on this by providing a more consistent way to write c as a product of *positive permutation braids*. The positive permutation braids are the set of all positive braids l such that $\Delta_n = lr$ for r also a positive braid. The set of all l is known as a *left gcd of* Δ_n. The set of right gcds of Δ_n, all r, is actually the same set. We already saw that all positive generators are positive permutation braids, since the Δ_n can be written starting with any positive generator followed by a positive braid, but there are many others. Another way to think of a positive permutation braid is as a braid that can be written with each strand crossing any other strand no more than once. Note that Δ_n in particular has this property.

While positive permutation braids do not have a unique word representation, they do have a unique permutation. So any two positive permutation braids that have the same permutation are isomorphic. This was first shown by E.S. Elrifai.

To return to the consistent notation of a positive braid, Thurston showed that a positive braid can be written uniquely as a product of positive permutation braids, with each successive permutation braid being as large as possible without introducing negative generators. This, then, gives a unique form for a braid up to isomorphisms of permutation braids.

It should be noted that the normal form of a braid does not necessarily have the shortest word length, and in many cases is much longer than the initial braid word. Finding an algorithm to give a shorter canonical word for a given braid in Artin's generators is explored in [6] with some success.

5.2. **Normal Form for Band Generators.** A similar approach can be taken with Birman, Ko, and Lee's band generators. In this case, a different fundamental word is used

$$\delta_n = a_{n,n-1}a_{n-1,n-2}\dots a_{2,1} = \sigma_{n-2}\dots\sigma_1$$

that looks like the right hand strand crossing over all other strands to the left, as shown in Figure 9. As with Δ_n, δ_n can also be written starting or ending with any generator in the braid group. A proof of this was developed by Dr. Boden and myself and is included as Appendix 1. While by far not the first proof (see [2] for another), the inductive nature of this one makes it quite tidy.

Similar to the greedy normal form, each braid b of B_n can be written $b = \delta_n^k c$, where k is an integer and c in this case is a braid in the positive band generators made of a product of *canonical factors*. Canonical factors are the band generator equivalent of positive permutation braids. These canonical factors are based on the

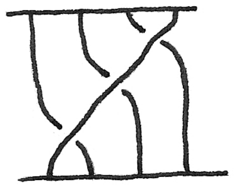


FIGURE 9. The fundamental braid δ_4.

concept of parallel descending cycles. A descending cycle is intuitively a series of generators $a_{t_j,t_{j-1}}a_{t_{j-1},t_{j-2}}\dots a_{t_2,t_1}$ with $t_j > t_{j-1} > \dots > t_1$. Parallel descending cycles are those that are either nested or entirely disjoint, so the words can commute by the far and nested commutativity property. Figure 10 shows an example and a non-example of parallel descending cycles. A canonical factor is a braid that can be represented as a product of parallel descending cycles. It is shown in [2] that a positive braid can be uniquely decomposed into a product of canonical factors, thereby solving the word problem in the band generators.

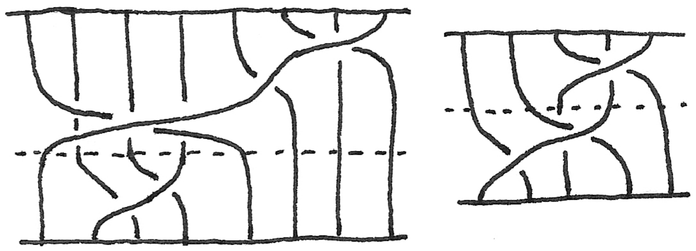


FIGURE 10. The left braid consists of two parallel descending cycles because the descending cycles commute. The right braid has two descending cycles that intertwine, and so are not parallel.

Solutions to the word problem in both presentations are solved in polynomial time with respect to the word length and the size of the braid group B_n. However, a braid represented in the band generators is sometimes a shorter word than the same braid represented in the Artin generators (at worst the words are the same length), and so using band generators can be computationally faster when the word length is reduced.

6. The Conjugacy Problem

The conjugacy problem asks whether two given words are conjugate, i.e. whether there is a braid α such that $b = \alpha c \alpha^{-1}$ for braids b and c. While many solutions have been discovered for the word problem other than the ones presented here, very few have been found for the conjugacy problem. Further, the common algorithm to solve the conjugacy problem is rather inefficient. This algorithm is once again based on work by Garside that has been improved upon by others.

Given a braid b in greedy normal form, there is a finite (though large) set of conjugates called the *summit set*, S_b. Garside showed that the summit sets of two braids are either identical or disjoint, and braids with identical summit sets are conjugate. So if one knows the entire set S_b, and an element from S_c is also in S_b, then b is conjugate to c. If an element of S_c is not in S_b, then the braids are not conjugate

Unfortunately, the summit set for a given braid can be unpredictably large, and so calculating the entire set is unwieldy. Improvements in Garside's algorithm have been made in refining the number of elements required for comparison. One improvement by ElRifai and Morton was to reduce the problem to finding a subset of the summit set called the *super summit set*, and this has recently been reduced further by Gebhardt to an even smaller subset called the *ultra summit set*. Both of these subsets have the same property of being identical or disjoint for conjugate braids, and so are more efficient to use for the algorithm. However, these advances still do not reduce the algorithm to shorter than exponential in both word length and braid group size.

There is an equivalent to the super summit set in the band generators that can be used to determine conjugacy. It is conjectured that this solution may be polynomial in time (i.e. far more efficient than the same algorithm with the Artin generators), but as far as I know, this has not been proved.

7. Applications

Braid theory has applications in other areas of mathematics, as well other disciplines. While certainly not an exhaustive list, some of these applications include connections to knots and links, cryptography, and quantum physics.

7.1. **Knots and Links.** Links can be made from braids by extending one end of the braid around to attach to the top, closing the strands into one or more loops. Knots are links with only one component, or loop. In particular, all braids that map to a single k-cycle form knots, while those that map to multiple k-cycles form loops. It is clear that every braid can be made into a link by closing its ends, but it is less clear that every link can be oriented such that it is the closure of a braid. The latter is known as Alexander's Braiding Theorem, and is proved in [8]. Some simple examples transforming knots into braids are shown in Figure 11.

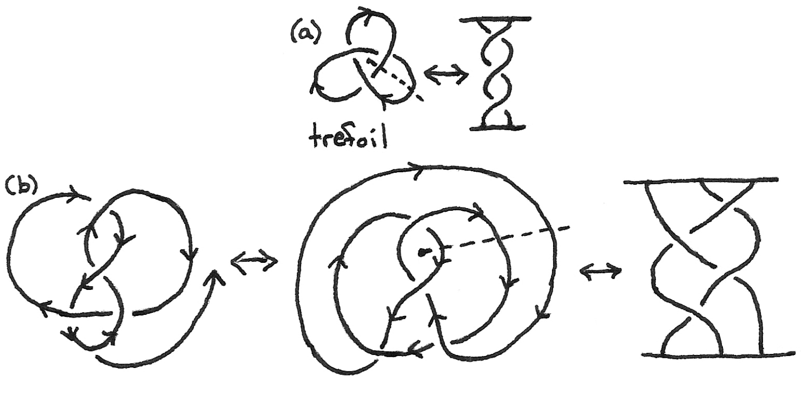


Figure 11. (a) A trefoil and associated braid. (b) A more complicated knot that requires rearrangement before the associated braid can be found.

7.2. **Cryptography.**

Public key cryptography is used in situations where one wants to send information securely across an insecure line. Classic public key systems rely on functions that are easy to calculate, with inverses that are difficult to calculate. The public key, k, is announced, and the message m is encoded via some function $f(k,m)$. Only by knowing the private key k' to perform f^{-1} can m be easily recovered. While knowing k and $f(k,m)$ should theoretically be enough to calculate k', and thus m, in practicality the computational difficulty is set well beyond current capabilities.

A standard method uses the difficulty of factoring products of large primes. Two large primes, p and q, are chosen secretly by person A. Their product, $n = pq$, is announced publicly, and is used by person B to encode a message m in such a way that m cannot be recovered without factoring n. While both finding large primes and multiplying them together is relatively easy, factoring very large numbers (hundreds of digits long) requires an impractically slow brute force method. This scheme was developed by Rivest, Shamir, and Adleman, and so is known as RSA.

Another scheme, known as the Diffie-Hellman key exchange, has the goal of creating a secret key to be known by only by two people over an insecure line. This also depends on functions with inverses that are are difficult to compute. Here there is a public key k, a function f, and two a private keys a and b known only to person A and person B respectively. A calculates and sends $f(k,a)$ to B and B sends $f(k,b)$ to A. A then calculates $f(f(k,b),a)$ and B calculates $f(f(k,a),b)$, which gives the same result as A has. For Diffie-Hellman key exchange, the function f is exponentiation on a cyclic group. Exponentiation is easy to compute, but the inverse, discrete log, is as difficult to compute as large factorizations, so a cannot be recovered by knowing only k and $f(k,a)$.

Public key cryptography schemes using braid theory depend on the difficulty of solving the conjugacy problem for arbitrary braids. One example is the Ko et al scheme [7]. Persons A and B choose private keys a and b respectively from B_n, and a public key p is chosen from B_{2n+1}. Person A augments a with enough identity strings on the right to make it an element of B_{2n+1}, then calculates apa^{-1} and sends it to B. B then computes b', which is b augmented by identity strings on the left, sending σ_i to σ_{i+n+1}. Note that a and b' then commute by far commutativity since the closest crossings are two strands apart. $b'p(b')^{-1}$ is then communicated to A. A computes $a(b'p(b')^{-1})a^{-1}$ and B computes $b'(apa^{-1})(b')^{-1}$, which is the same as $a(b'p(b')^{-1})a^{-1}$ since a and b' commute. Thus this braid is known to both A and B but not known publicly, so it can be used as a private key.

This scheme depends on the assumption that a is not deducible given the public knowledge of braids p and apa^{-1} (and similarly for b'). However, if the conjugacy problem can indeed be solved efficiently, as has been conjectured, a could indeed be calculated from this information. Thus such an algorithm would be an effective attack on this and most other braid based cryptographic systems. Dehornoy in [4] gives an overview of some other braid based cryptographic schemes, as well as some possible ways to prevent such attacks. However, it seems likely that braid-based cryptography will not become a practical alternative to the standard RSA and Diffie-Hellman systems.

7.3. **Quantum Physics.**

Braids can be used as the quantum equivalent to permutations. For this use, one can think of braids as a time series of n "particles" descending through a tube while moving around each other. Thus a braid not only keeps track of which particles end up where, but also which ones crossed in front of or behind other ones. The basic relations to braids also apply to some groups used in quantum physics.

8. Closed-Ended Twist Braids

In this section I look at a problem tangentially related to braid theory that was of personal interest to myself, as it was proposed to me by my dad a number of years ago. The terminology given here was developed by Dr. Boden and myself.

Closed-ended twist braids are formed from three ribbons attached at both ends. A simple way to form this is to cut two slits lengthwise in a strip of paper, making a trivial three strand braid. However, since the strands are ribbons, not only can strands cross, but strands can also have full twists in either direction.

One "non-braid" move for this object is done by inserting one end through a slit either from the back or from the front. This crosses two of the strands around each other twice and gives them each a full twist, while adding a full twist in the other direction to the third strand. Other possible moves include adding a half twist to the whole braid, and adding trivial crossings $\sigma_1\sigma_1^{-1}$ or $\sigma_2\sigma_2^{-1}$ to strands.

Even when the twists are ignored, not all elements of B_3 can be made with a closed-ended twist braid because of the restrictive nature of the moves. One way to see this is to note that all closed-ended twist braids are either pure braids, or can be made a pure braid by adding a half twist. But even all elements of P_3 cannot be made since all crossings are added in pairs, so elements of P_3 with an odd number of crossings, notably the generators, cannot be made.

But surprisingly, with only these moves, the braid $\sigma_1\sigma_2\sigma_1\sigma_2\sigma_1\sigma_2$ (a standard hair braid design) can be made, with no remaining twists. See Figure 12 for an illustration of the braid. This was first shown to me by my dad, but I could not figure out how to create it myself until recently. While it is possible to write out a series of moves that should result in canceled twists and the correct braid, in practice it tends to rapidly get messy. Thus the most reliable way to create the braid is to force the desired braid at the top, while removing the bottom inverse braid crossings by inserting the end through the appropriate slit.

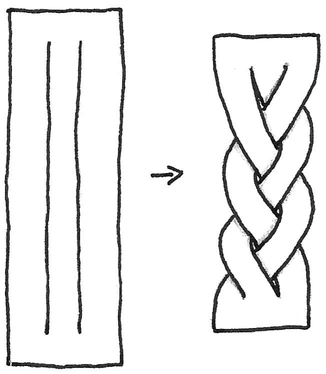


Figure 12. The identity closed-ended twist braid, and the hair braid design that can be made without tearing the shape.

It should be noted that a similar flat braid is possible with five, or any odd number of strands in a closed-ended twist braid, but gets increasingly complicated as the strand number increases.

References

[1] E. Artin, *Theory of braids*, Annals of Mathematics, 48 (1988), 101-126.

[2] J. Birman, K.H. Ko, and S.J. Lee, *A new approach to the word and conjugacy problems in the braid groups*, Annals of Mathematics, Advances in Mathematics, 139-2 (1998), 322-335.

[3] R. Corran, *Solving the word problem in the singular braid monoid*, Australian Mathematics Society Gazette 26 (1999), 27-33.

[4] P. Dehornoy, *Braid-based cryptography*, Contemporary Mathematics, 360 (2004), 5-33.

[5] F.A. Garside, *The braid group and other groups*, The Quarterly Journal of Mathematics. Oxford, 20-78 (1969), 235-254.

[6] M. Hock, *Braid compression*, currently available at http://www.math.wisc.edu/∼boston/hock.pdf.

[7] K.H. Ko, S.J. Lee, J.H. Cheon, J.W. Han, J.S. Kang, and C. Park, *New Public-Key Cryptosystem Using Braid Groups*, Crypto 2000, Springer Lecture Notes in Computer Science, 1880 (2000), 166-184.

[8] V.V. Prasolov and A.B. Sossinsky, *Knots, links, braids and 3-manifolds*, Translations of Mathematical Monographs, 154 (1997).

9. Appendix 1: Positive Representations of δ_n

Birman, Ko, and Lee defined a new presentation of the braid group with generators $a_{t,s}$ and fundamental word δ_n given in the standard generators by

$$\begin{aligned} a_{t,s} &= (\sigma_{t-1}\sigma_{t-2}\dots\sigma_{s+1})\sigma_s(\sigma_{s+1}^{-1}\dots\sigma_{t-2}^{-1}\sigma_{t-1}^{-1}) \\ \delta_n &= a_{n,(n-1)}a_{(n-1),(n-2)}\dots a_{2,1} = \sigma_{n-1}\sigma_{n-2}\dots\sigma_1 \end{aligned}$$

The defining relations for the generators $a_{t,s}$ are far commutativity and 'partial' commutativity when the generators share a strand as follows

$$a_{t,s}a_{r,q} = a_{r,q}a_{t,s} \tag{1}$$

if $a_{t,s}$ and $a_{r,q}$ are either nested or entirely separate, and

$$a_{t,s}a_{s,r} = a_{t,r}a_{t,s} = a_{s,r}a_{t,r} \tag{2}$$

if $n > t > s > r > 1$

As with Garside's Δ_n, δ_n is also positively equivalent to a word that begins with any generator $a_{t,s}$. This can be shown using induction on δ_n. For the first step, $\delta_2 = a_{2,1}$, starts with the only possible braid generator. So, for $n > 2$, assume δ_n can be written beginning with any generator followed by some positive word w, then there are three cases. First, if the desired generator is of the form $a_{t,s}$, with $1 \le s < t \le n-1$, then

$$\begin{aligned} \delta_{n+1} &= a_{(n+1),n}\delta_n \\ &= a_{(n+1),n}a_{t,s}w \\ &= a_{t,s}a_{(n+1),n}w \end{aligned}$$

where w is a positive word. Second, if the desired generator is of the form $a_{n,s}$ with $1 \le s \le n-1$, then

$$\begin{aligned} \delta_{n+1} &= a_{(n+1),n}\delta_n \\ &= a_{(n+1),n}a_{n,s}w \\ &= a_{n,s}a_{(n+1),s}w \end{aligned}$$

Finally, if the desired generator is of the form $a_{(n+1),s}$ with $1 \le s \le n-1$ (note that for $s = n$ δ_n already starts with the generator), then

$$\begin{aligned} \delta_{n+1} &= a_{(n+1),n}\delta_n \\ &= a_{(n+1),n}a_{n,s}w \\ &= a_{(n+1),s}a_{(n+1),n}w \end{aligned}$$