

Combinatorial aspects of braids with applications to cryptography

by

Max Bennett

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics & Optimization

Waterloo, Ontario, Canada, 2015

© Max Bennett 2015

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis is a collection of different results on braids, and draws connections between them. We first introduce braids by showcasing a number of equivalent ways of describing what a braid is, and how those representations are related. Then, while uncovering enumerative properties of the positive braid monoid, we consider algorithms to compute the lcm of a set of braids. This leads to more than one elegant solution to the word problem. We explore some efficient algorithms which solve the word problem for braids, and then also explore the conjugacy problem and the cryptosystems that rely on the hardness of it in their proofs of security.

Acknowledgements

I would first of all like to thank my supervisor Ian Goulden for giving me the freedom to study something that I found interesting and enjoyable. I would also like to thank my readers Alfred Menezes and David Wagner for taking the time to read my thesis.

I also couldn't have done this without the wonderful friends I have made over my two years in Waterloo, both inside and outside of school. Thank you for that.

Dedication

To Jordana and Annie.

Table of Contents

List of Figures	viii
List of Algorithms	ix
1 Introduction	1
1.1 A brief history of braids in mathematics	1
1.2 Outline	3
2 Representations of braids	4
2.1 Geometric braids	4
2.2 Artin generators and the classical representation	8
2.3 The braid monoid	17
2.4 Birman, Ko and Lee's representation.	19
2.5 Pure braids.	22
2.6 The punctured disc.	24
3 Enumeration	27
3.1 Divisibility in the braid monoid.	27
3.2 Least common multiples via subword reversing.	32
3.3 How to count positive braids.	42

4	Isotopy Problem	46
4.1	Invariants and algorithms.	46
4.2	Artin’s algorithm.	48
4.3	Subword reversing.	50
4.4	The greedy normal form.	51
4.5	The left-normal form for the braid group.	64
5	Braid group cryptography	66
5.1	Security definitions.	67
5.2	Conjugacy problems.	69
5.3	Cryptosystems based on braid groups.	69
5.4	Attacks on braid group cryptography.	75
	APPENDICES	78
A	Computer programs for using braids.	79
A.1	Manual for braid-progs.py.	79
	References	81

List of Figures

2.1	Visualizations of braids.	5
2.2	The product of braids b_1 and b_2	6
2.3	The commutative product of b and ϵ is simply b	6
2.4	The mirror image of a braid is its inverse.	7
2.5	Associativity of the braid product.	7
2.6	A braid.	8
2.7	The important braids σ_i and σ_i^{-1}	9
2.11	A Δ -move on a braid.	11
2.15	A Δ -move can be broken up into smaller moves.	15
2.20	A braid diagram of the generator $\sigma_{2,5}$	20
2.27	The braid $\sigma_1\sigma_3\sigma_2\sigma_1$ and its associated permutation.	22
2.28	The pure braid $A_{i,j}$	23
2.32	Geometric braids and the mapping class group of D_n^-	26
3.4	A consistency check on 3.1.	28
3.5	A diagram of $\Delta_{2,5}$ in B_6	29
3.28	The reversing diagram for $u^{-1}v$ and $u^{-1}v'$	39
4.2	Equivalent braids imply equal permutations, but the converse is false.	47
4.4	An example of a 1-pure braid.	49
4.15	The permutation braid $\text{br}((1743)(26)(5))$	54

List of Algorithms

4.3	Naïve algorithm.	48
4.5	Artin’s algorithm	49
4.6	Subword reversing for positive braids.	50
4.8	Subword reversing for general braids.	51
4.29	Computing the greedy normal form.	63
5.2	Diffie-Hellman key exchange protocol.	67
5.8	Anshel-Anshel-Fisher-Goldfeld key-exchange	70
5.11	Ko et al.’s key-exchange protocol.	72
5.13	Ko et al.’s encryption protocol.	73
5.15	Sibert, Dehornoy, and Girault’s authentication scheme.	75
5.16	Garside’s solution to the conjugacy problem.	76

Chapter 1

Introduction

Almost everybody from around the world and from all time periods is familiar with braids in some capacity. Both men and women have been interweaving their hair into braids for both cultural and utilitarian reasons for thousands of years. Braids have a place in culture and fashion, but also mathematics.

1.1 A brief history of braids in mathematics

Despite the many mathematical properties that braids exhibit, braids were not formally considered by mathematicians [53] until the 18-th century, along with the dawn of knot theory. In 1771 Vandermonde [60] wrote in “Remarques sur les problèmes de situation”:

Whatever the twists and turns of a system of threads in space, one can always obtain an expression for the calculation of its dimensions, but this expression will be of little use in practice. The craftsman who fashions a braid, a net, or some knots will be concerned, not with questions of measurement, but with those of position: what he sees there is the manner in which the threads are interlaced.

CHAPTER 1. INTRODUCTION

Vandermonde is insightful with this observation. Over two centuries later, we are still only concerned with the position of the crossings in a braid and not with the exact position of the strands. Around the same time as Vandermonde there is evidence of Gauss having interest in braids [53] although Gauss thought of braids as a way of coding a knot.

The first time mathematics saw braids in a formal sense was in 1900 when Hurwitz described an action (now called the Hurwitz action) that a braid would have on finite sequences of a free group. The first major investigation of braids, however was by Emil Artin in 1925 [7] when he published a paper in German called *Theorie der Zöpfe* (Theory of Braids). It described braids in great detail and introduced the important braid σ_i , and the well studied braid relations. In 1947 he published a paper in English called Theory of Braids [8] where he disregards a lot of the proofs in [7] because of the nature of braid projections and diagrams - the basis of those proofs. In [8], he gives more rigorous proofs using a well defined coordinate system, not that different than a configuration space. Not long after in 1950, Artin published a more streamlined note in English called The Theory of Braids [6] (note the definite article “the” in the title) that gave a very concise and easy to understand overview of braids. These three similarly entitled articles are why many refer to the generators of the braid group as Artin generators.

Artin [6] introduced the idea of joining the ends of a braid together to obtain a link - much like the way Gauss thought of braids. He realized that two “connected braids” are the same if and only if they are conjugate in the braid group. He did not know if the conjugacy problem was solvable yet.

In 1965, Garside studied braids for his PhD thesis, and published his findings in a paper called The Braid Group and Other Groups [34] in 1969. Here he showed that the conjugacy problem for the braid group is solvable, and offers an algorithm. Much later, Patrick Dehornoy and Luis Paris [29] generalized braid groups into what they called Garside groups, to honour Garside for his contribution to the field. Patrick Dehornoy has made a massive contribution to the field of braids, with dozens of articles, notes and books on braids, ranging from the word problem to ordering braids to cryptographic applications.

1.2 Outline

The structure of this thesis is as follows. First, in Chapter 2 we will introduce different ways one may think about braids. We will see that there are geometric, topological, and most importantly, algebraic methods for studying braids. In Chapter 3 we will consider an interesting combinatorial problem regarding braids: how does one enumerate them? In this chapter we will uncover seemingly unrelated yet interesting properties of braids which will be used in the subsequent chapters. In Chapter 4 we will cover a few of the best, and most used algorithms for comparing braids: given two braids, it is nontrivial to determine whether or not they are the same braid with respect to isotopy. These algorithms do this. In Chapter 5, we take a look at some applications to cryptography. Braids play a role in cryptography thanks to the conjugacy problem on braids, introduced by Artin in 1925. It turns out that the conjugacy problem is not a hard enough problem to base cryptosystems on.

The Appendix has a list of computer programs that a reader may find useful if they want a fast way to compare braids and compute lcms.

Chapter 2

Representations of braids

2.1 Geometric braids

The most intuitive way to think about a braid mathematically is to do so geometrically. Take two finite parallel lines of the same length in \mathbb{R}^3 , L_1 and L_2 , each with equidistant points P_1, \dots, P_n and Q_1, \dots, Q_n placed on them respectively. For precision, we can define L_1 to be the line that connects $(1, 0, 0)$ to $(1, 0, n)$ and L_2 connects $(0, 0, 0)$ to $(0, 0, n)$, with $P_i = (1, 0, i)$ and $Q_i = (0, 0, i)$. This detail is usually ignored however, as the important part is that L_1 and L_2 are parallel and that the P_i 's and Q_i 's are distinct and in order.

A geometric braid on n strands is a collection of n disjoint continuous curves (called strands) that connect the points P_1, \dots, P_n to Q_1, \dots, Q_n . We will discuss this in more detail later, but a braid which connects P_i to Q_i for $i = 1, \dots, n$ is called a pure braid; in general we do not require this property in regular braids, most braids induce a permutation on the points. Two braids b_1 and b_2 are equivalent when they are isotopic to one another, and when this is the case we write $b_1 \equiv b_2$. We think of equivalent braids as being the same.

For a braid on n strands we can express each strand i by a function $\beta_i : [0, 1] \rightarrow \mathbb{R}^3$ so that $\beta_i(0) = (1, 0, i)$ and $\beta_i(1) = (0, 0, j)$, when the braid connects P_i to Q_j . To ensure that no strand has a knot in it, we require that each strand is strictly decreasing in the

CHAPTER 2. REPRESENTATIONS OF BRAIDS

x -coordinate. That is, when $0 \leq t_1 < t_2 \leq 1$, then $\text{proj}_1(\beta_i(t_1)) > \text{proj}_1(\beta_i(t_2))$.¹

We visualize braids by drawing a diagram. The most obvious way to do this is to draw the braid as a 3-dimensional rendering (see figure 2.1a) using a software package such as KnotPlot [54] but this can be rather cumbersome, and may overlook some of the discrete properties of braids (not to mention the difficulty of typesetting). The open source mathematical programming package Sage [57] can also be used to visualize braids. Instead, we draw braids with a *braid diagram*².

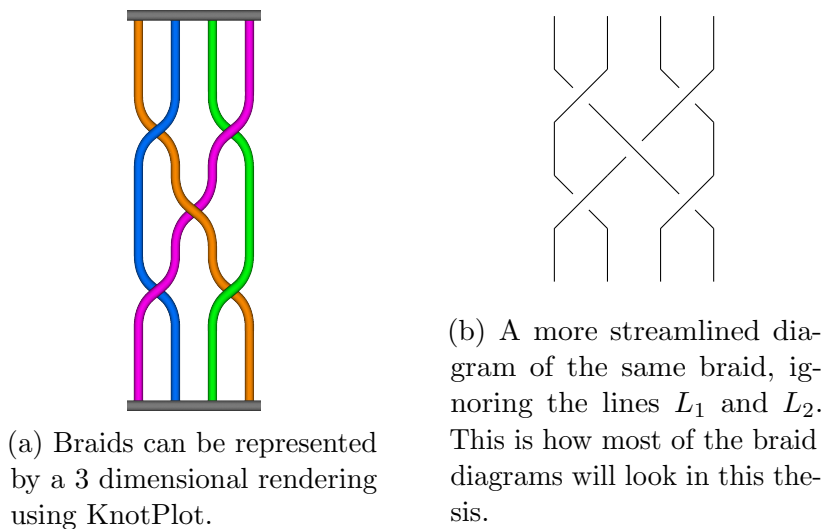


Figure 2.1: Visualizations of braids.

As is the case with technical drawings of knots, the two dimensional projection of a braid suffers from a rather obvious dilemma: how do you draw two strings crossing over one another? The answer is also obvious, but worth noting. Although the strands are each continuous, we denote a strand going underneath another by introducing a break immediately before it passes underneath, and continuing immediately after.

¹Here, proj_1 is the function $(x, y, z) \mapsto x$, and in general $\text{proj}_i : (x_1, \dots, x_k) \mapsto x_i$.

²It is worth noting that in the literature, instead of drawing braids from top to bottom, they are sometimes drawn from left to right.

CHAPTER 2. REPRESENTATIONS OF BRAIDS

The braid in figure 2.1b is composed of straight lines. When a braid has this property we say that it is *polygonal*. For simplicity, we always assume we are working with polygonal braids, or braids which are obviously isotopic to polygonal braids.

Now that we have some elementary tools to talk about braids, we can begin to describe how braids interact with one another, and thus how we define a product. Given two braids b_1 and b_2 , place b_1 over b_2 so that the bottom line of b_1 (L_2) replaces the top line of b_2 (L_1). We then identify the overlapping points and remove the line. We call this product $b_1 b_2$ or $b_1 * b_2$.

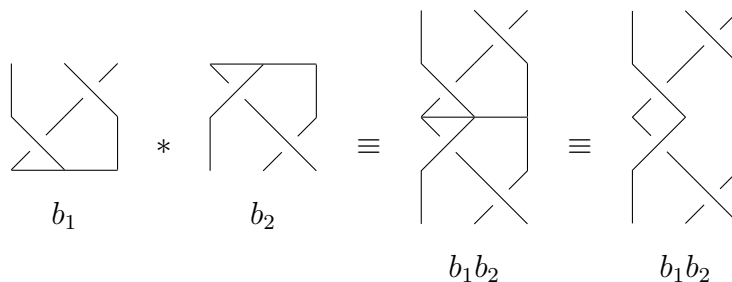


Figure 2.2: The product of braids b_1 and b_2 .

The braid product is called a product for a good reason. Define the pure braid ϵ to be that which connects the points P_i to Q_i without any intertwining of the strands. It is clear that for any braid b , $b * \epsilon \equiv b \equiv \epsilon * b$. That is, ϵ is the identity braid.

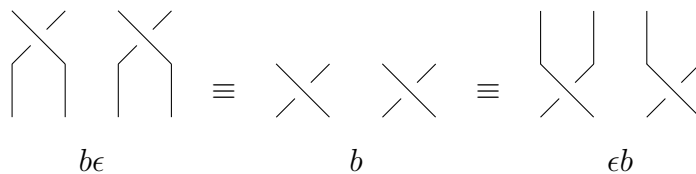


Figure 2.3: The commutative product of b and ϵ is simply b .

Every braid has an inverse: for each braid b , there exists a braid called b^{-1} , which satisfies $b * b^{-1} \equiv b^{-1} b = \epsilon$. Finding the inverse of a braid is simple: if one places a mirror at the bottom of b the braid that appears as the reflection is its inverse.

CHAPTER 2. REPRESENTATIONS OF BRAIDS

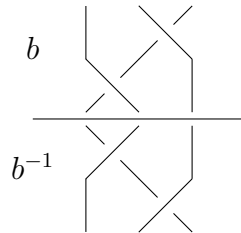


Figure 2.4: The mirror image of a braid is its inverse.

At this point the reader may have already guessed that the braid product is associative. As mentioned earlier, the position of the lines L_1 and L_2 are usually allowed to move throughout space (provided they are parallel and oriented in the same direction) which makes associativity trivial. However, if we require the lines to be fixed, then there still exists an isotopy between $(b_1 * b_2) * b_3$ and $b_1 * (b_2 * b_3)$. It is obtained by compressing b_3 and expanding b_1 , as in figure 2.5.

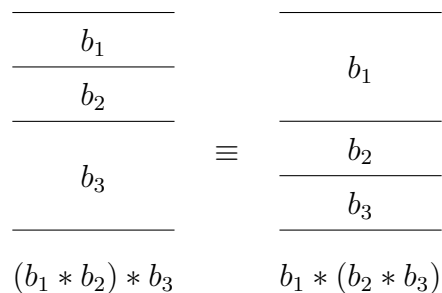


Figure 2.5: Associativity of the braid product.

It follows of course that braids which have the same number of strands form a group under the braid product. Let us refer to this infinite group of geometric braids on n strands as \mathcal{B}_n . This is still loosely defined since we do not have a formal system of describing a braid other than diagrams or explicit formulas for each strand - each element of the braid group \mathcal{B}_n is an equivalence class of isotopic diagrams. Just as Artin was, we are forced to make a choice between describing an ambient isotopy explicitly (which is very difficult), or we opt to rely on intuition and diagrams (which is subject to mistakes). Furthermore, as

Vandermonde wrote, there is a fair amount of extraneous information involved when we write out such explicit formulas. For basic proofs however, one may refer to intuition and a series of diagrams and animations to justify a desired result. For basic results this will suffice, but for more complicated results we will need algebra.

2.2 Artin generators and the classical representation

Notice that in any braid diagram, there are only two types of crossings: The right side passes over the left (call this positive) and where left side passes over the the right: (call this negative)



It is helpful to comb through a braid so that there is only one crossing occurring per horizontal segment at a time. This makes it easier to transform the geometric object into a product of smaller braids. For example, take the braid in figure 2.1b. Working our way down from the top, it is plain to see that it is isotopic to the braid in figure 2.6.

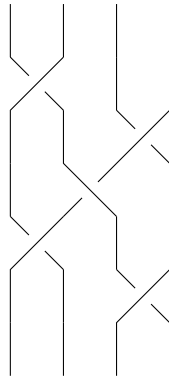


Figure 2.6: A braid.

All we did here was shift the first crossing on the left upwards so that the two crossings at the top of the braid are no longer adjacent to one another. We also performed a similar action on the bottom of the braid.

CHAPTER 2. REPRESENTATIONS OF BRAIDS



Figure 2.7: The important braids σ_i and σ_i^{-1}

If we have n strands in our braid, then there are only $n - 1$ places where there can be a crossing. Thus, if one strand passes over any other, it must cross one at a time, passing over adjacent strands one after another. Denote the crossing of the strand at position i and $i + 1$ by σ_i when the crossing is positive, and σ_i^{-1} otherwise. In the braid in figure 2.6, working our way from the top down, we see the crossings $\sigma_1, \sigma_3, \sigma_2^{-1}, \sigma_1, \sigma_3$ in that order. It makes sense to call this braid $b = \sigma_1 \sigma_3 \sigma_2^{-1} \sigma_1 \sigma_3$, since it is the product of those braids in that order.

When we combed through the braid in figure 2.1b, at the top and bottom we had σ_1 and σ_3 occurring simultaneously, but we opted to place σ_1 before σ_3 to obtain b . There was no reason for this: it is straightforward to see that our braid b is isotopic to $\sigma_3 \sigma_1 \sigma_2^{-1} \sigma_1 \sigma_3$.

Of course, it seems as though this naming scheme for braids is not well defined. This motivates the following definition.¹

Definition 2.8 (the braid relations). Fix n . For $i, j \in \{1, \dots, n - 1\}$,

$$\sigma_i \sigma_j \equiv \sigma_j \sigma_i \text{ if } |i - j| \geq 2 \tag{\mathcal{R}_1}$$

$$\sigma_i \sigma_{i+1} \sigma_i \equiv \sigma_{i+1} \sigma_i \sigma_{i+1} \tag{\mathcal{R}_2}$$

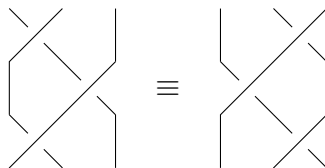
$$\sigma_i \sigma_i^{-1} \equiv \sigma_i^{-1} \sigma_i \equiv \epsilon \tag{\mathcal{R}_3}$$

¹The braid relations are referred to so often that they warrant their own counter, i.e. \mathcal{R}_1 in place of 2.9. The reader will do themselves a favor by memorizing these three relations and their corresponding label to avoid having to refer to this page.

CHAPTER 2. REPRESENTATIONS OF BRAIDS

are called the braid relations.

The first equation is straightforward: if two crossings share no strands, then they commute. The third equation is also trivial: a negative crossing undoes a positive crossing and vice-versa. The second equation needs some explanation.



The diagram above shows the relation $\sigma_1\sigma_2\sigma_1 \equiv \sigma_2\sigma_1\sigma_2$. It can be understood as the braid σ_1 passing underneath the third strand and becoming σ_2 . Of course, these three relations hold, but it is yet to be proven that these are the only braid relations that are required to form the braid group. We would like to show that the group

$$B_n = \left\langle \sigma_1, \dots, \sigma_n \left| \begin{array}{ll} \sigma_i\sigma_j \equiv \sigma_j\sigma_i & |i - j| \geq 2 \\ \sigma_i\sigma_j\sigma_i \equiv \sigma_j\sigma_i\sigma_j & |i - j| = 1 \end{array} \right. \right\rangle$$

is isomorphic to \mathcal{B}_n . The generating set $\Sigma_n = \{\sigma_1, \dots, \sigma_n\}$ is called the set of Artin generators after Artin’s work [7], wherein the author loosely proves by means of braid diagrams (which he calls projections) that each braid can be described by this finite set of generators. He also proves that the braid relations are the only relations that the braid group has. Since the proofs in [7] are intuitive, and in some cases “not even convincing” (as the author put it) Artin used the theory of the punctured disc 22 years later in [8] to formally prove the results in [7].¹ Due to its straightforwardness, the classical representation is often favoured in the literature.

We now prove the result Artin [7] showed in 1925, that the braid relations are the only relations that make up B_n . Instead of relying on diagrams or a punctured disc, we will

¹The classical representation which uses Artin generators is often referred to as the Artin representation. In other literature the Artin representation refers to the n -punctured disc (see Section 2.6), since Artin discovered both of these representations. In this thesis I will refer to the former as the classical representation and the later as the n -punctured disc to avoid confusion.

CHAPTER 2. REPRESENTATIONS OF BRAIDS

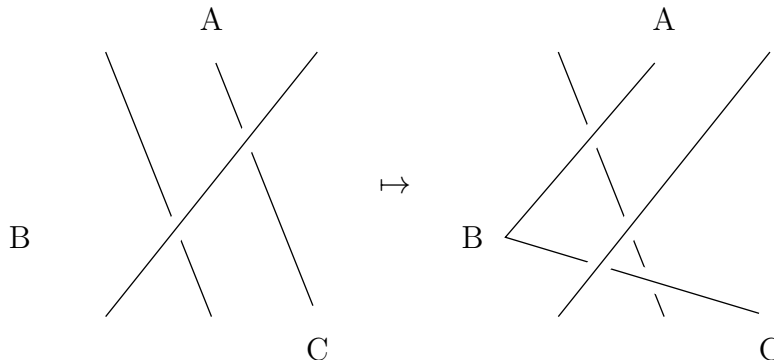


Figure 2.11: A Δ -move on a braid.

prove it in the concrete manner that Kassel et al. do in Section 1.2.2 of [42].

Theorem 2.9. *The geometric braid group \mathcal{B}_n is isomorphic to the classical braid group B_n .*

To prove this, first we must introduce a move which preserves isotopy, called a Δ -move. We then show that two braids are isotopic if and only if one can be transformed into another by a finite series of Δ -moves. It then suffices to show that the braid relations are the outcome of Δ -moves.

Definition 2.10 (Δ -move). Let c be a strand in a braid which connects the points A to C in a straight line with $\text{proj}_1(A) > \text{proj}_1(C)$. Introduce a new point B with $\text{proj}_1(A) > \text{proj}_1(B) > \text{proj}_1(C)$. A move which replaces the segment AC with ABC by dragging the midpoint of AC towards B , all the while keeping the rest of the braid intact is called $\Delta(ABC)$. The pullback of $\Delta(ABC)$, which replaces the segments AB and BC with AC , is denoted $\Delta^{-1}(ABC)$. The moves Δ and Δ^{-1} are called Δ -moves.

Keep in mind that Δ is not a function, but rather a type of isotopy. The move Δ does not keep track of which strands it slides above or below and is thus not well defined. However, the pullback $\Delta^{-1}(ABC)$ happens to be well defined and could be considered a function.

CHAPTER 2. REPRESENTATIONS OF BRAIDS

We first ensure that both braids are polygonal. That is, each is expressed only by straight lines. Any good polygonal approximation algorithm will suffice to find such polygonal braids.

Lemma 2.12. *If two polygonal braids are isotopic, then the isotopy can be described by a series of Δ -moves.*

Proof. Let b_1 and b_2 be isotopic polygonal braids and $f : I \rightarrow B_n$ be an isotopy such that $f(0) = b_1$ and $f(1) = b_2$. Furthermore, by the same reasoning that tells us that any braid is isotopic to a polygonal braid, assume that $f(i)$ is polygonal for all $i \in I$.

A polygonal braid is made up of a multitude of vertices. We also require that the number of vertices on each strands is the same. If one strand has fewer vertices than another, simply add the necessary number of vertices between two adjacent vertices.

We will prove the result for a single strand: call it c when it is on $f(0)$ and d when it is on $f(1)$. Let the vertices along c be denoted c_0, c_1, \dots, c_k and $d_0 = c_0, d_1, \dots, d_k = c_k$ on d for some $k \in \mathbb{Z}$. Each c_i gets moved to d_i in the isotopy. Instead of moving each vertex at the same time, it is possible to describe this same isotopy via Δ -moves.

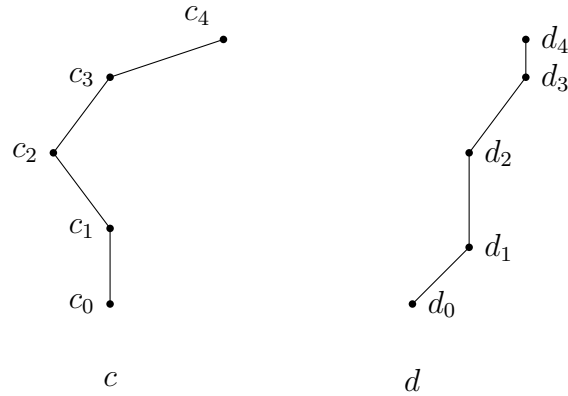
The Δ -moves are as follows: (reading the composition of Δ -moves from right to left)

$$\Delta^{-1}(c_k c_{k-1} d_{k-1}) \circ \dots \circ \Delta(c_2 d_2 d_1) \circ \Delta^{-1}(c_2 d_1 c_1) \circ \Delta(c_1 d_1 c_0)$$

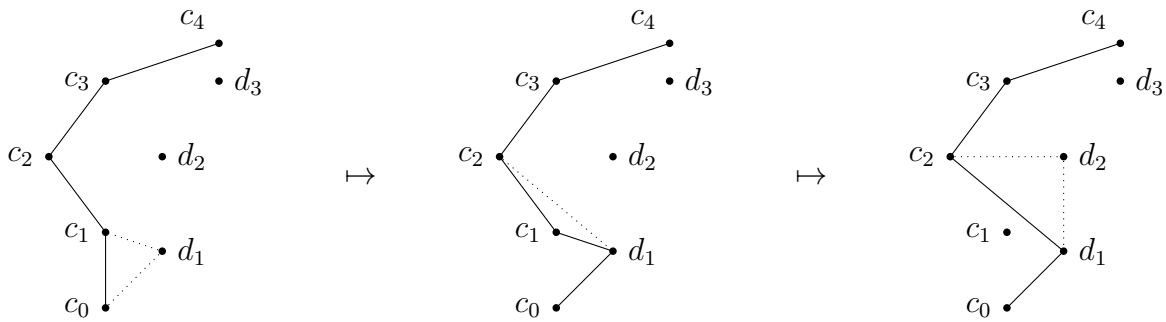
□

Example 2.13. If the list of Δ -moves in the proof of Lemma 2.12 was not illuminating, consider the following polygonal strands which are isotopic.

CHAPTER 2. REPRESENTATIONS OF BRAIDS



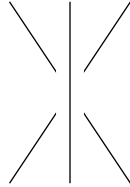
Typically, their isotopy is described by moving each vertex of c to the corresponding vertex d in a continuous fashion. We will do something slightly different. For simplicity, the points d_0, \dots, d_4 have been superimposed onto the line of c so we can see how the isotopy can be described in Δ -moves on those points. The first few Δ moves are $\Delta(c_1, d_1, c_0)$, $\Delta(c_2 d_1 c_1)$ and $\Delta(c_2 d_2 d_1)$.



We continue to make Δ -moves all the way until $\Delta^{-1}(c_4 c_3 d_3)$. The two strands are isotopic via Δ -moves.

Braid diagrams do not always describe the braid precisely. When there are three or more strands that cross over one another at the same point in the projection, there is ambiguity. Take for example the braid diagram below.

CHAPTER 2. REPRESENTATIONS OF BRAIDS



It is impossible to determine from the braid diagram itself whether the first strand passes over the third or vice versa. This is why we have the notion of a *generic braid*. A generic braid has the property that whenever one strand passes over another, there is no other strand underneath. Every geometric braid is isotopic to a generic braid, simply by nudging one of the crossings to a place where there are no other crossings. Furthermore, any Δ -move can be made so that no non-generic braids are introduced. For the remainder of the following lemmas and proofs, we will always assume our braids are generic.

Lemma 2.14. *The Δ -moves can be described by the braid relations.*

Proof. First we show that one can split up a Δ -move so that it consists only of Δ -moves which interfere with (or, creates crossings with) exactly two or three strands. Consider a move $\Delta(ABC)$ which interferes with more than three strands. Take a point A' on AB , C' on BC and B' on AC so that $\text{proj}_1(A') > \text{proj}_1(B') > \text{proj}_1(C')$. Notice that

$$\Delta(ABC) = \Delta(A'BC') \circ \Delta^{-1}(A'B'C') \circ \Delta(AA'B') \circ \Delta(B'C'C),$$

as in figure 2.15. Since each Δ -move can be split up into smaller moves, we can reduce the proof to dealing with Δ -moves that interfere with either one strand, or two strands that cross inside the triangle ABC . If a Δ -move interferes with three or more strands, break it up into smaller moves and consider them separately.

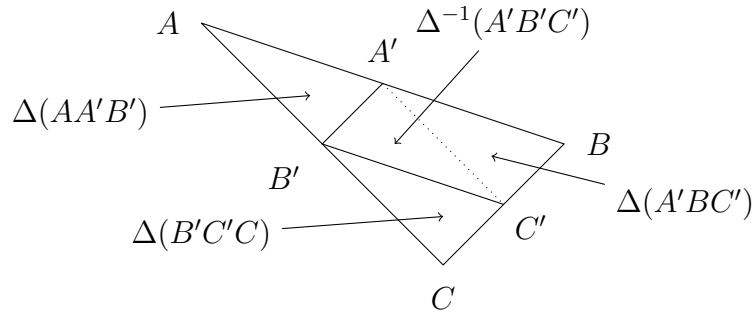
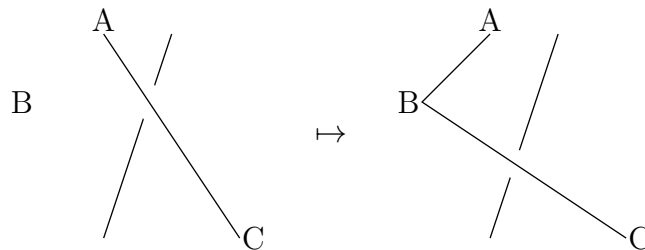


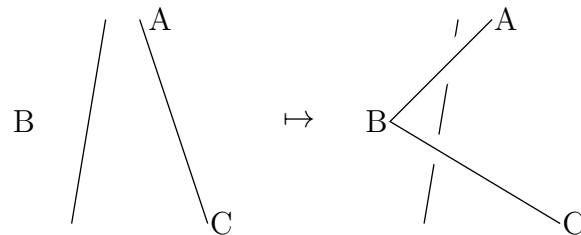
Figure 2.15: A Δ -move can be broken up into smaller moves.

Case 1. The Δ -move interferes with one strand, call it c .

If c passes through AC then $\Delta(ABC)$ does not alter any of the crossings, although it affects the position of the crossing. relation \mathcal{R}_1 may or may not be used here, depending on the position of other crossings.



Otherwise c passes through AB and BC , and relation \mathcal{R}_3 is used.

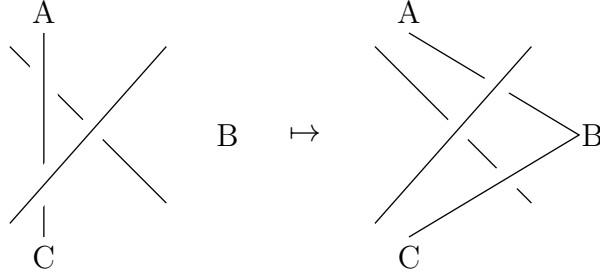


Case 2. The Δ -move interferes with two strands which cross inside ABC .

Let c_1 and c_2 be the two strands. We can further reduce our work by only considering

CHAPTER 2. REPRESENTATIONS OF BRAIDS

triangles ABC such that both c_1 and c_2 cross through AC . If they do not, break ABC into smaller triangles until they do. There are hence six outcomes of this, all variations of the following Δ -move,



which is exactly the case of relation \mathcal{R}_2 . Other cases also involve relation \mathcal{R}_3 as well as relation \mathcal{R}_2 . For simplicity and to conserve space, instead of covering all cases by diagram, we will describe them in terms of their Artin generators. Start with

$$\sigma_1\sigma_2\sigma_1^{-1} \mapsto \sigma_2^{-1}\sigma_1\sigma_2.$$

This follows because $\sigma_1\sigma_2\sigma_1^{-1} \equiv (\sigma_2^{-1}\sigma_2)\sigma_1\sigma_2\sigma_1^{-1} \equiv \sigma_2^{-1}(\sigma_1\sigma_2\sigma_1)\sigma_1^{-1} \equiv \sigma_2^{-1}\sigma_1\sigma_2$. The other cases have similar proofs. We end the proof with a table of all six cases.

$$\begin{array}{ll} \sigma_1\sigma_2\sigma_1 & \equiv \sigma_2\sigma_1\sigma_2 & \sigma_1^{-1}\sigma_2\sigma_1 & \equiv \sigma_2\sigma_1\sigma_2^{-1} \\ \sigma_1\sigma_2\sigma_1^{-1} & \equiv \sigma_2^{-1}\sigma_1\sigma_2 & \sigma_1^{-1}\sigma_2^{-1}\sigma_1 & \equiv \sigma_2\sigma_1^{-1}\sigma_2^{-1} \\ \sigma_1\sigma_2^{-1}\sigma_1^{-1} & \equiv \sigma_2^{-1}\sigma_1^{-1}\sigma_2 & \sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1} & \equiv \sigma_2^{-1}\sigma_1^{-1}\sigma_2^{-1} \end{array} \quad \square$$

Proof of 2.9. The proof follows immediately from Lemmas 2.12 and 2.14: every geometric braid can be parsed into a product of σ_i 's, and the braids in \mathcal{B}_n satisfy the braid relations and only the braid relations. Therefore the two groups B_n and \mathcal{B}_n are isomorphic. \square

Now that 2.9 is established we identify the two groups and simply write B_n when we are talking about the braid group. Typically, we think of braids in terms of their Artin representation, and refer to the geometric version for a consistency check.

Keep in mind that n refers to how many strands are included in the braid, even though there are only $n - 1$ generators in B_n . Typographical errors and inconsistencies between

CHAPTER 2. REPRESENTATIONS OF BRAIDS

literature are easily corrected since there is a natural injection from B_n into B_m for any $1 < n < m$. So any equivalence of braids in B_n holds in B_m as well. Furthermore, some authors introduce the notion of the *stable braid group* B or B_∞ which is the limit of B_n as $n \rightarrow \infty$. This allows us to talk about a property of a particular braid without concern for how many strands or generators it has. We can have n generators or n strands, whichever fits the context best.

A quick note on the symbols $=$ and \equiv .

There is a subtle difference between $=$ and \equiv . When two braids are equal, it means they are equal in every way. No matter what the context — they can be geometric braids, or members of the classical braid group B_n . If $b = \sigma_3^{-1}\sigma_1\sigma_2^{-1}\sigma_3$, and $b' = b$, then $b' = \sigma_3^{-1}\sigma_1\sigma_2^{-1}\sigma_3$. If $b'' = \sigma_1\sigma_2\sigma_3^{-1}\sigma_2^{-1}$, then $b'' \equiv b$ but $b'' \neq b$.

The symbol \equiv means equality in the braid group, and $=$ means equality in terms of the word used to represent them in the braid group.

2.3 The braid monoid

A monoid is a set of elements with an associative binary operation and an identity element. A monoid in which every element has an inverse is a group. Indeed a group is a monoid, but a monoid is not always a group.

Consider the set of braids with crossings oriented in the same direction. Adopting the same operation (concatenation) as the braid group, one can clearly see that the operation is associative, and that the product of two “positive braids” is also positive.

CHAPTER 2. REPRESENTATIONS OF BRAIDS

Definition 2.16 (braid monoid). The monoid on generators $\Sigma_n = \{\sigma_1, \dots, \sigma_{n-1}\}$ with relations \mathcal{R}_1 and \mathcal{R}_2 is called the braid monoid. We write

$$B_n^+ = \left\langle \sigma_1, \dots, \sigma_{n-1} \left| \begin{array}{ll} \sigma_i \sigma_j \equiv^+ \sigma_j \sigma_i & |i - j| \geq 2 \\ \sigma_i \sigma_j \sigma_i \equiv^+ \sigma_j \sigma_i \sigma_j & |i - j| = 1 \end{array} \right. \right\rangle^+$$

to denote the braid monoid. When two positive braids b_1 and b_2 are equivalent in the braid monoid, we write $b_1 \equiv^+ b_2$.

The braid monoid naturally inherits many properties of the braid group, but has some properties that the braid group does not have. One such property is that it is homogeneous, which is to say that if two positive braids are equivalent, then they have the same number of generators: $b_1 \equiv^+ b_2 \Rightarrow |b_1| = |b_2|$. Homogeneity is important since length is the main ingredient for enumeration, as we will see in Chapter 3.

Some of the most important properties are yet to come. We will see in later sections that the braid monoid is cancellative, (that is, if $bb_1 \equiv^+ bb_2$ then $b_1 \equiv^+ b_2$) and that every braid b in B_n has the property that $b \equiv b_1 b_2^{-1}$, where b_1 and b_2 are positive braids. This fact is important for a number of reasons. For one, the algorithm which computes the form of b_1 and b_2 gives us an algorithm for deciding when two braids (positive or not) are equivalent to one another, which we will discuss in Chapter 4. It also gives us an algorithm to compute the lcm of two braids in Chapter 3. Most importantly, however, it tells us that the natural map $B_n^+ \rightarrow B_n$ is injective. In other words the distinction between \equiv and \equiv^+ is immaterial when comparing positive braids. Examples of familiar monoids which have a natural injective homomorphism into a group are $(\mathbb{N}, +) \rightarrow (\mathbb{Z}, +)$, and $(\mathbb{Z}, \cdot) \rightarrow (\mathbb{Q}, \cdot)$. We can state that $3 = 3$ regardless of which group or monoid we are talking about.

We will end this section with the statement of a theorem often referred to as Ore's Condition [18], which is in reference to semigroups: monoids without (necessarily) an identity element. Instead we will state it in terms of a monoid, since that is how it is relevant to us. We do not yet have the machinery to demonstrate that the braid monoid satisfies the condition, but we will develop this in Chapter 3.

CHAPTER 2. REPRESENTATIONS OF BRAIDS

Definition 2.17 (cancellative). A monoid M is said to be left-cancellative (resp. right-cancellative) if for all elements $a, b, c \in M$, $ab \equiv ac$ implies $b \equiv c$ ($ac \equiv bc \Rightarrow a \equiv b$, resp.).

Definition 2.18 (common multiple). Let M be a monoid with elements a, b . A common multiple of a and b exists when there exists a' and b' such that $aa' \equiv bb'$. The braid aa' (or bb') is called a common multiple of a and b .

Theorem 2.19 (Ore's Condition [18]). *Let M be a left-cancellative monoid such that any two elements admit a common right multiple. Then there exists a group G which is unique up to isomorphism such that*

1. *there exists an injective homomorphism $I : M \rightarrow G$,*
2. *for every element $g \in G$ there exists $a, b \in M$ such that $g = I(a)I(b)^{-1}$.*

The proof of Ore's Condition will be omitted but can be found in [18]. The proof follows the same steps a second or third year undergraduate algebra class would follow to give the construction of \mathbb{Q} from \mathbb{N} .

2.4 Birman, Ko and Lee's representation.

Birman, Ko, and Lee [9] introduced a different representation of braids. Just like in the classical representation, each generator represents two strands crossing. Instead of adjacent strands like the Artin generators, these new generators represent arbitrary strands crossing over one another. As such, each generator has two subscripts. The crossing of strands s and t is denoted $\sigma_{s,t}$, see figure 2.20 for a diagram. Since these generators are still braids, they can be expressed as a product of Artin generators. Hence when $1 \leq s < t \leq n$,

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

CHAPTER 2. REPRESENTATIONS OF BRAIDS

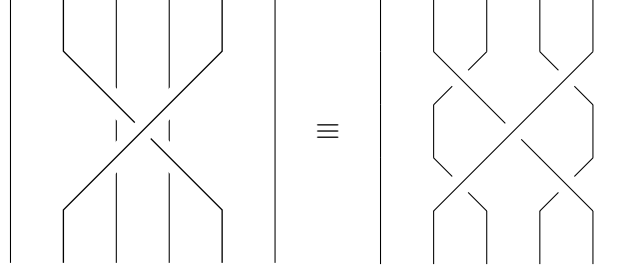


Figure 2.20: A braid diagram of the generator $\sigma_{2,5}$.

When $s > t$, we define $\sigma_{s,t} = \sigma_{t,s}$ and write the smaller subscript first whenever convenient. The new representation has generators $\Sigma'_n = \{\sigma_{s,t} : 1 \leq s < t \leq n\}$ with relations

$$\sigma_{q,r}\sigma_{s,t} \equiv \sigma_{s,t}\sigma_{q,r} \quad \text{if } (t-r)(t-q)(s-r)(s-q) \neq 0 \quad (2.21)$$

$$\sigma_{s,t}\sigma_{r,s} \equiv \sigma_{r,t}\sigma_{s,t} \equiv \sigma_{r,s}\sigma_{r,t} \quad \text{if } 1 \leq r < s < t \leq n. \quad (2.22)$$

relation 2.21 is similar to relation \mathcal{R}_1 — if two crossings share no strands, then they commute. relation 2.22 considers the case when two crossings share a strand. We will cover one of the cases with an informal isotopy.

Theorem 2.23. *The group B_n is isomorphic to the group with generators Σ'_n and relations 2.21 and 2.22.*

Proof. We start with the classical representation of the braid group, and add the new generators, and their relations.

$$B_n = \left\langle \Sigma_n \left| \begin{array}{ll} \sigma_i\sigma_j \equiv \sigma_j\sigma_i & |i-j| \geq 2 \\ \sigma_i\sigma_j\sigma_i \equiv \sigma_j\sigma_i\sigma_j & |i-j| = 1 \end{array} \right. \right\rangle$$

$$= \left\langle \Sigma_n \cup \Sigma'_n \left| \begin{array}{ll} \sigma_{s,t} = (\sigma_{t-1}\sigma_{t-2}\cdots\sigma_{s+1})\sigma_s(\sigma_{s+1}^{-1}\sigma_s^{-1}\cdots\sigma_{t-1}^{-1}) & 1 \leq s < t \leq n \\ \sigma_i\sigma_j \equiv \sigma_j\sigma_i & |i-j| \geq 2 \\ \sigma_i\sigma_j\sigma_i \equiv \sigma_j\sigma_i\sigma_j & |i-j| = 1 \\ \text{relations 2.21 and 2.22} \end{array} \right. \right\rangle.$$

CHAPTER 2. REPRESENTATIONS OF BRAIDS

But since $\sigma_i = \sigma_{i,i+1}$, we can replace all instances of σ_i with $\sigma_{i,i+1}$:

$$B_n = \left\langle \Sigma'_n \left| \begin{array}{ll} \sigma_{s,t} = (\sigma_{t-1,t} \cdots \sigma_{s+1,s+2}) \sigma_{s,s+1} (\sigma_{s+1,s+2}^{-1} \cdots \sigma_{t-1,t}^{-1}) & 1 \leq s < t \leq n \\ \sigma_{i,i+1} \sigma_{j,j+1} \equiv \sigma_{j,j+1} \sigma_{i,i+1} & |i-j| \geq 2 \\ \sigma_{i,i+1} \sigma_{j,j+1} \sigma_{i,i+1} \equiv \sigma_{j,j+1} \sigma_{i,i+1} \sigma_{j,j+1} & |i-j| = 1 \\ \text{relations 2.21 and 2.22} & \end{array} \right. \right\rangle.$$

All that remains is to show that the relations

$$\sigma_{s,t} = (\sigma_{t-1,t} \cdots \sigma_{s+1,s+2}) \sigma_{s,s+1} (\sigma_{s+1,s+2}^{-1} \cdots \sigma_{t-1,t}^{-1}) \quad 1 \leq s < t \leq n \quad (2.24)$$

$$\sigma_{i,i+1} \sigma_{j,j+1} \equiv \sigma_{j,j+1} \sigma_{i,i+1} \quad |i-j| \geq 2 \quad (2.25)$$

$$\sigma_{i,i+1} \sigma_{j,j+1} \sigma_{i,i+1} \equiv \sigma_{j,j+1} \sigma_{i,i+1} \sigma_{j,j+1} \quad |i-j| = 1 \quad (2.26)$$

follow from relations 2.21 and 2.22. relation 2.25 follows from relation 2.21 immediately, since $|i-j| \geq 2$ implies $(j+1-i)(j+1-i+1)(j-i)(j-i+1) > 0$.

For relation 2.26, write $s = i$, $t = j = i + 1$ and $r = j + 1$, and use relation 2.22 twice to obtain

$$\begin{aligned} \sigma_{s,t}(\sigma_{s,r} \sigma_{t,r}) &\equiv \sigma_{s,t} \sigma_{s,r} \sigma_{t,r} \\ \sigma_{s,t}(\sigma_{t,r} \sigma_{s,t}) &\equiv (\sigma_{s,t} \sigma_{s,r}) \sigma_{t,r} \\ \sigma_{s,t} \sigma_{t,r} \sigma_{s,t} &\equiv (\sigma_{t,r} \sigma_{s,t}) \sigma_{t,r} \\ \sigma_{i,i+1} \sigma_{j,j+1} \sigma_{i,i+1} &\equiv \sigma_{j,j+1} \sigma_{i,i+1} \sigma_{j,j+1}, \end{aligned}$$

as desired. Finally we consider relation 2.24. If $t = s + 1$ the result is trivial, so consider the case when $t > s + 1$. Suppose the result holds when $t = s + k$ for $k > 2$. It follows then that $\sigma_{s,t+1} \equiv \sigma_{t,t+1} \sigma_{s,t} \sigma_{t,t+1}^{-1}$. If we let $r = t + 1$ then relation 2.22 gives us

$$\begin{aligned} \sigma_{s,r} &\equiv \sigma_{s,r} (\sigma_{t,r} \sigma_{t,r}^{-1}) \\ &\equiv (\sigma_{t,r} \sigma_{s,t}) \sigma_{t,r}^{-1} \\ &\equiv \sigma_{t,r} (\sigma_{t-1,t} \cdots \sigma_{s+1,s+2} \sigma_{s,s+1} \sigma_{s+1,s+2}^{-1} \cdots \sigma_{t-1,t}^{-1}) \sigma_{t,r}^{-1}, \end{aligned}$$

CHAPTER 2. REPRESENTATIONS OF BRAIDS

as desired. Since relations 2.24 to 2.26 can be obtained from relations 2.21 and 2.22, we get that

$$\begin{aligned}
 B_n &= \left\langle \Sigma_n \left| \begin{array}{ll} \sigma_i \sigma_j \equiv \sigma_j \sigma_i & |i - j| \geq 2 \\ \sigma_i \sigma_j \sigma_i \equiv \sigma_j \sigma_i \sigma_j & |i - j| = 1 \end{array} \right. \right\rangle \\
 &= \left\langle \Sigma'_n \left| \begin{array}{ll} \sigma_{q,r} \sigma_{s,t} \equiv \sigma_{s,t} \sigma_{q,r} & \text{if } (t - r)(t - q)(s - r)(s - q) > 0 \\ \sigma_{s,t} \sigma_{r,s} \equiv \sigma_{r,t} \sigma_{s,t} \equiv \sigma_{r,s} \sigma_{r,t} & \text{if } 1 \leq r < s < t \leq n. \end{array} \right. \right\rangle.
 \end{aligned}$$

□

2.5 Pure braids.

There is a natural homomorphism π from the braid group to the symmetric group found by following the strands of a braid. The image $\pi(b)$ is called the *permutation of b* . For reasons that will become clear later, (and are completely superfluous to this section), the permutation $\pi(\sigma_i)$ is defined in a backwards kind of way: $\pi(\sigma_i) = (i + 1, i)$, and in general $\pi(b)$ can be found by following each strand from the *bottom* of the braid to the top.

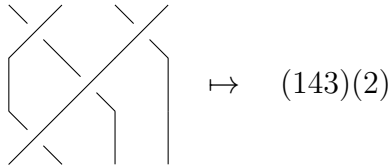


Figure 2.27: The braid $\sigma_1 \sigma_3 \sigma_2 \sigma_1$ and its associated permutation.

An important part of this map is its kernel. This subgroup of B_n has a special name: the *pure braid group on n strands*, denoted P_n . The fact that σ_i is not pure means that P_n does not decompose into the same Artin generators as B_n .

CHAPTER 2. REPRESENTATIONS OF BRAIDS

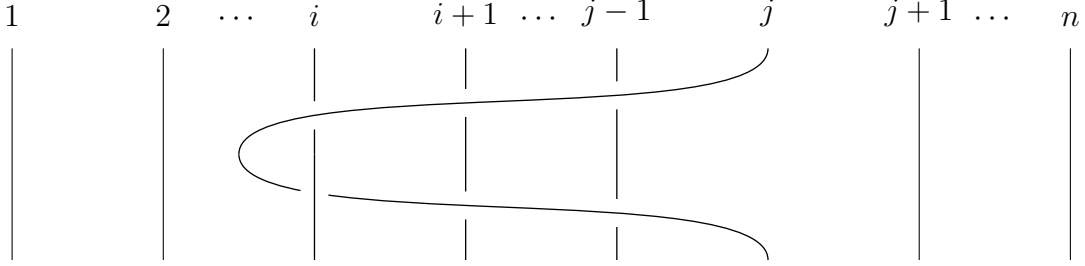


Figure 2.28: The pure braid $A_{i,j}$.

Theorem 2.29. *The pure braid group P_n is generated by the set $\{A_{i,j} : 1 \leq i < j \leq n\}$, where $A_{i,j}$ is the pure braid in figure 2.28 subject to the following relations*

$$A_{r,s}^{-1}A_{i,j}A_{r,s} \equiv \begin{cases} A_{i,j} & \text{when } 1 \leq r < s < i < j \leq n \text{ or } 1 \leq i < r < s < j \leq n \\ A_{r,j}A_{i,j}A_{r,j}^{-1} & \text{when } 1 \leq r < s = i < j \leq n \\ (A_{i,j}A_{s,j})A_{i,j}(A_{i,j}A_{s,j})^{-1} & \text{when } 1 \leq r = i < s < j \leq n \\ (A_{r,j}A_{s,j}A_{r,j}^{-1}A_{s,j}^{-1})A_{i,j}(A_{r,j}A_{s,j}A_{r,j}^{-1}A_{s,j}^{-1})^{-1} & \text{when } 1 \leq r < i < s < j \leq n. \end{cases}$$

It is worth noting that $A_{i,j} \equiv (\sigma_{j-1}\sigma_{j-2}\dots\sigma_{i+1})\sigma_i^2(\sigma_{j-1}\sigma_{j-2}\dots\sigma_{i+1})^{-1}$.

It is possible to prove this by using the Reidemeister-Schreier rewriting process as in Appendix I of [41], but we will prove this using a short exact sequence as in [10]. That is to say, three groups A, B and C along with homomorphisms f_0, \dots, f_3 such that the image of f_i is equal to the kernel of f_{i+1} and

$$\{1\} \xrightarrow{f_0} A \xrightarrow{f_1} B \xrightarrow{f_2} C \xrightarrow{f_3} \{1\}.$$

Here $\{1\}$ is the trivial group. The role of f_0 and f_3 are to ensure that f_1 is one-to-one and that f_2 is onto. By the First Isomorphism Theorem and the fact that $\ker(f_2) = \text{im}(f_1)$, we have the nice property that $C \cong B/A$. We say that the short exact sequence is split if $B \cong C \oplus A$ as well. We will use the Splitting Lemma which states

CHAPTER 2. REPRESENTATIONS OF BRAIDS

Lemma 2.30. *Given a short exact sequence $\{1\} \xrightarrow{f_0} A \xrightarrow{f_1} B \xrightarrow{f_2} C \xrightarrow{f_3} \{1\}$, the following are equivalent:*

- *there exists an $\overline{f_1} : B \rightarrow A$ such that $f_1 \overline{f_1}$ is the identity map on B*
- *there exists an $\overline{f_2} : C \rightarrow B$ such that $\overline{f_2} f_2$ is the identity map on C*
- *the short exact sequence is split.*

We will use this lemma without proof. A curious reader can find a proof in [15] if they desire.

Proof of 2.29. Let F_n be the free group generated by $\{A_{i,n}\}_{i=1}^{n-1}$. The split short exact sequence we use is

$$\{1\} \rightarrow F_n \xrightarrow{f_1} P_n \xrightarrow{f_2} P_{n-1} \rightarrow \{1\}.$$

The homomorphism f_1 is the inclusion map since $F_n \subset P_n$, and f_2 is a special homomorphism which removes the n -th strand from a pure braid, and leaves the rest of the braid intact. Clearly, a candidate for $\overline{f_2}$ exists: the inclusion map $\iota : P_{n-1} \rightarrow P_n$ has the property that $f_2 \circ \overline{f_2}(p) = p$ for all $p \in P_{n-1}$. From this we deduce that $P_n \cong P_{n-1} \oplus F_n$.

From here a basic induction argument works to make the step from P_{n-1} to $P_n \cong P_{n-1} \oplus F_n$, with all of the relations following. The idea is that the elements from F_n do not introduce any new relations except those that bump the index up to n . \square

2.6 The punctured disc.

We saw before that algebra makes a great substitute for geometry when studying braids. In this chapter, we introduce a topological way to study braids. This theory was introduced by Artin [8] and was studied thoroughly by Birman [14] in 1975. Although we do not use the punctured disc again in this thesis, this is a well studied representation of the braid group.

CHAPTER 2. REPRESENTATIONS OF BRAIDS

Let $n \geq 1$ be fixed, and D_n be the disc centered at $((n+2)/2, 0)$ with diameter $n+2$. Let $D_n^- = D_n \setminus \{(1, 0), (2, 0), \dots, (n, 0)\}$, that is, D_n^- is an n -times punctured disc. Since we may wish to refer to each of these points, let $P_i = (i, 0)$.

It is helpful to refer to D_n^- in a more general way so we can use topology. We will continue to use D_n^- as our model, but the theory we apply will use the fact that D_n^- is a closed, connected, orientable surface with genus $g = 0$, $b = 1$ boundary components, and n puncture points. By a fundamental theorem due to Möbius, D_n^- is homeomorphic to a surface referred to by $S_{0,1,n}$. For simplicity, we refer to $S_{0,1,n}$ as simply S . The boundary of S is denoted δS .

Definition 2.31 (mapping class group). Let $\text{Homeo}^+(S, \delta S)$ be the group of orientation preserving homeomorphisms of S . The mapping class group of S , denoted $\mathcal{M} = \mathcal{M}_{0,1,n}$ is the unique group of isotopy classes of elements of $\text{Homeo}^+(S, \delta S)$, where each isotopy fixes the boundary δS . Also, $\mathcal{M}_{0,1,\hat{n}}$ is the group of isotopy classes which fix the order of the points P_1, \dots, P_n .

Other than the fact that we have a disc with n points missing and that we sometimes consider braids with n strands, it is not immediately obvious how these two are connected. So, we imagine a braid constrained to the inside of a cylinder equal to $D_n \times [0, 1]$. Furthermore, let us require that the strands are connected to the points P_1, \dots, P_n on the top and bottom of the cylinder.

The basic result of this section is that \mathcal{B}_n is isomorphic to $\mathcal{M}_{0,1,n}$. The isomorphism is visualized in figure 2.32. The easy direction of the isomorphism is from \mathcal{M} to \mathcal{B}_n . We start by taking a homeomorphism from \mathcal{M} and sweep it through the cylinder $D_n \times [0, 1]$ starting from the top to the bottom. Each of the punctured points leaves a trace which corresponds to a strand. Since there are n punctured points, then there are n strands which do not intersect and therefore a braid is in the cylinder.

The harder direction of the isomorphism is from $\mathcal{B}_n \rightarrow \mathcal{M}_{0,1,n}$. We can imagine a similar picture — placing a braid $b \in \mathcal{B}_n$ in a cylinder with top D_n^- , and i -th strand beginning at P_i . The idea is that we sweep the top disc and keep track of where the strand moves

CHAPTER 2. REPRESENTATIONS OF BRAIDS

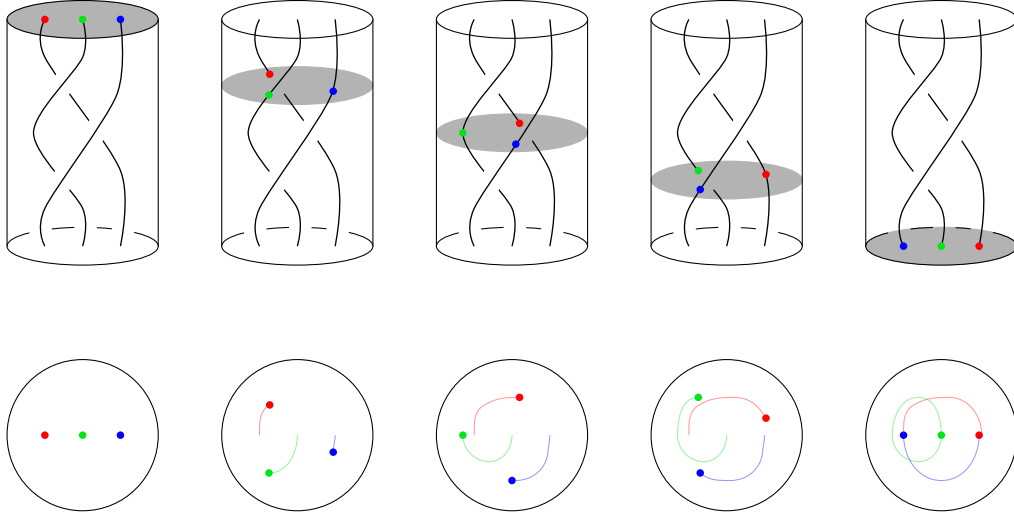


Figure 2.32: Geometric braids and the mapping class group of D_n^- .

throughout the disc. By the bottom of the cylinder the strands are back in the same place where they began. The image under this isomorphism is the element that follows the points P_i along the braid as it was swept through the cylinder.

Of course, complete proofs exist but will be omitted since the required topology is beyond the scope of this thesis. The idea of the proof (as Birman and Brendle give in [10]) is to first show that $P_n \cong \mathcal{M}_{0,1,\hat{n}}$, and then compare the short exact sequences

$$\{1\} \rightarrow P_n \rightarrow B_n \rightarrow \mathcal{S}_n \rightarrow \{1\} \quad \text{and} \quad \{1\} \rightarrow \mathcal{M}_{0,1,\hat{n}} \rightarrow \mathcal{M}_{0,1,n} \rightarrow \mathcal{S}_n \rightarrow \{1\}$$

where \mathcal{S}_n is the symmetric group. Since the first and last two groups are isomorphic, by a well known result known as the Five-Lemma, the middle groups (B_n and $\mathcal{M}_{0,1,n}$) are isomorphic as well.

A much closer inspection of $\mathcal{M}_{0,1,n}$ and its connection to the braid group can be found in [42], where the authors further demonstrate that the braid group can be viewed as a configuration space as well.

Chapter 3

Enumeration

We have seen so far that braids have a home in both algebra and topology. In this chapter we will explore some of the combinatorial aspects of braids. Albenque and Nadeau [2] show that there is a very nice way to count the number of positive braids with respect to the number of Artin generators used to represent them. Their result operates under the assumption that positive braids admit common multiples and that the braid monoid is cancellative. We will therefore need to use Dehornoy's work on subword reversing diagrams [24, 25, 26, 27] to prove the preliminary results. We will see later that subword reversing plays a large role in other aspects of braids, particularly in solving the braid isotopy problem.

3.1 Divisibility in the braid monoid.

In this section we will explore the notion of divisibility and multiplicity in the braid monoid and see that it has the structure of a lattice. First we will cover a few basic theorems so that the definitions can be well defined. These theorems can be found in [26], but were first published by Garside [34]. Notice that if two positive braids are equivalent in B_n^+ then they are also equivalent in B_n . The converse has not been proven yet (this is Ore's Condition), but will be in Section 3.2. This implies that the following theorems could be restated in terms of positive braids in B_n and \equiv as opposed to B_n^+ and \equiv^+ .

CHAPTER 3. ENUMERATION

Proposition 3.1. For $i \leq j < k$ the following equations hold in B_n^+ :

$$\sigma_j(\sigma_k\sigma_{k-1}\cdots\sigma_{i+1}\sigma_i) \equiv^+ (\sigma_k\sigma_{k-1}\cdots\sigma_{i+1}\sigma_i)\sigma_{j+1} \quad (3.2)$$

$$\sigma_{j+1}(\sigma_i\sigma_{i+1}\cdots\sigma_k) \equiv^+ (\sigma_i\sigma_{i+1}\cdots\sigma_k)\sigma_j. \quad (3.3)$$

Proof. By applying the braid relation \mathcal{R}_1 numerous times, it is straightforward to see that

$$\sigma_j\sigma_k\sigma_{k-1}\cdots\sigma_{i+1}\sigma_i \equiv^+ \sigma_k\sigma_{k-1}\cdots\sigma_j\sigma_{j+1}\sigma_j\cdots\sigma_{i+1}\sigma_i$$

and applying relation \mathcal{R}_2 , we get

$$\sigma_k\sigma_{k-1}\cdots\sigma_j\sigma_{j+1}\sigma_j\cdots\sigma_{i+1}\sigma_i \equiv^+ \sigma_k\sigma_{k-1}\cdots\sigma_{j+1}\sigma_j\sigma_{j+1}\cdots\sigma_{i+1}\sigma_i.$$

We can apply relation \mathcal{R}_1 again to obtain

$$\sigma_k\sigma_{k-1}\cdots\sigma_{j+1}\sigma_j\sigma_{j+1}\cdots\sigma_{i+1}\sigma_i \equiv^+ \sigma_k\sigma_{k-1}\cdots\sigma_{i+1}\sigma_i\sigma_{j+1}.$$

A symmetric argument holds for equation 3.3. □

Pictorially, the proof above can be seen by arranging a braid in which the rightmost strand passes above all other strands, with a single crossing before it. The idea of the proof is that we push the crossing over the strand that passes under all the other strands.

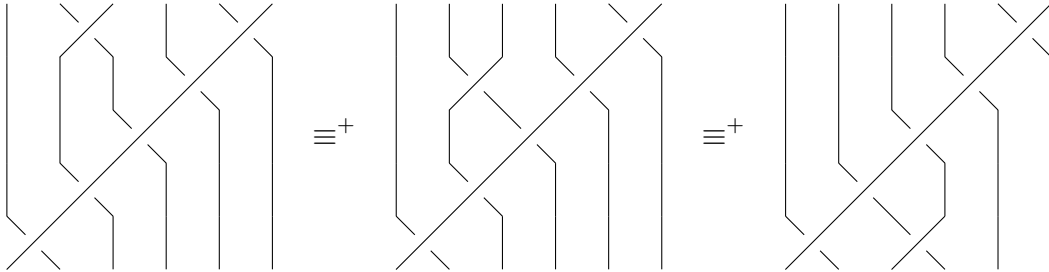


Figure 3.4: A consistency check on 3.1.

CHAPTER 3. ENUMERATION

The braid $\sigma_n \sigma_{n-1} \dots \sigma_1$ takes the right-most strand and passes it over the other n strands. This braid has nice properties, like the result of 3.1. As such, we denote this braid by the symbol $\delta_n = \sigma_n \sigma_{n-1} \dots \sigma_1$. Indeed, 3.1 can be restated as $\sigma_k \delta_n \equiv^+ \delta_n \sigma_{k+1}$. A similar braid is $\delta_{i,j} = \sigma_j \sigma_{j-1} \dots \sigma_i$ for integers $1 \leq i \leq j \leq n$.

The braid on $n + 1$ strands obtained by twisting the first $m + 1 \leq n + 1$ strands by 180 degrees is called Δ_m . This braid is equal to $\Delta_m = \delta_1 \delta_2 \dots \delta_m$. A similar braid is $\Delta_{i,j} = \delta_{i,i} \delta_{i,i+1} \dots \delta_{i,j}$, which is the same braid, except here we only rotate the strands i through $j + 1$. For example, figure 3.5 shows what $\Delta_{2,5} = \sigma_2 \sigma_3 \sigma_2 \sigma_4 \sigma_3 \sigma_2 \sigma_5 \sigma_4 \sigma_3 \sigma_2$ looks like in B_6 .

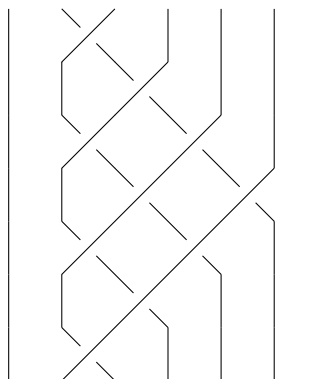


Figure 3.5: A diagram of $\Delta_{2,5}$ in B_6 .

The braid Δ_m is an important braid with many properties, and is often called the fundamental braid. Later, we will see how these properties are important when it comes to divisibility.

Lemma 3.6. For $n \geq 1$,

$$\Delta_n \equiv^+ \sigma_1 \sigma_2 \dots \sigma_n \Delta_{n-1}. \tag{3.7}$$

Proof. We will apply induction on n . When $n = 1$ or 2 the result is trivial, so let $n \geq 3$.

CHAPTER 3. ENUMERATION

By induction hypothesis, and the fact that Δ_{n-2} only has elements σ_1 through σ_{n-2} ,

$$\begin{aligned}
 \Delta_n &\equiv^+ \Delta_{n-1}\delta_n \\
 &\equiv^+ \sigma_1 \cdots \sigma_{n-1}\Delta_{n-2}\delta_n \\
 &\equiv^+ \sigma_1 \cdots \sigma_{n-1}\Delta_{n-2}\sigma_n\delta_{n-1} \\
 &\equiv^+ \sigma_1 \cdots \sigma_n\Delta_{n-2}\delta_{n-1} \\
 &\equiv^+ \sigma_1 \cdots \sigma_n\Delta_{n-1}
 \end{aligned}$$

as desired. □

Proposition 3.8. For $1 \leq i \leq n$,

$$\sigma_i\Delta_n \equiv^+ \Delta_n\sigma_{n-i+1}.$$

Proof. Again, we perform induction on n . When $n = 1$, the result is trivial. Suppose $n \geq 2$. If $i < n$, then by the induction hypothesis and equation 3.2, we get

$$\sigma_i\Delta_n \equiv^+ \sigma_i\Delta_{n-1}\delta_n \equiv^+ \Delta_{n-1}\sigma_{n-i}\delta_n \equiv^+ \Delta_{n-1}\delta_n\sigma_{n-i+1} \equiv^+ \Delta_n\sigma_{n-i+1}.$$

When $i = n$, by 3.1, equation 3.3, and lemma 3.6, we get

$$\sigma_n\Delta_n \equiv^+ \sigma_n\sigma_1 \cdots \sigma_n\Delta_{n-1} \equiv^+ \sigma_1 \cdots \sigma_n\sigma_{n-1}\Delta_{n-1} \equiv^+ \sigma_1 \cdots \sigma_n\Delta_{n-1}\sigma_1 \equiv^+ \Delta_n\sigma_1$$

as desired. □

Lemma 3.9. For $1 \leq i \leq n$, there exists a braid $b_{i,n}$ such that $\sigma_i b_{i,n} \equiv^+ \Delta_n$.

Proof. Again, we perform induction on n . The result is trivial for $n = 1$ and 2, so assume $n > 2$. If $i < n$ then by the induction hypothesis, we get a braid $b_{i,n-1}$ such that $\sigma_i b_{i,n-1} \equiv^+ \Delta_{n-1}$. Let $b_{i,n} = b_{i,n-1}\delta_n$. It follows that

$$\sigma_i b_{i,n} \equiv^+ \sigma_i b_{i,n-1}\delta_n \equiv^+ \Delta_{n-1}\delta_n \equiv^+ \Delta_n$$

CHAPTER 3. ENUMERATION

as desired. Now suppose that $i = n$. Notice that the braid $\Delta_{2,n}$ contains generators σ_i where $2 \leq i \leq n$. As such, by equation 3.2,

$$\Delta_{n-1}\sigma_n\sigma_{n-1}\cdots\sigma_1 \equiv^+ \sigma_n\sigma_{n-1}\cdots\sigma_1\Delta_{2,n}$$

since as each element σ_i of Δ_{n-1} passes through $\sigma_n\sigma_{n-1}\cdots\sigma_1$, it turns into σ_{i+1} . Thus, let $b_{n,n} = \sigma_{n-1}\cdots\sigma_1\Delta_{2,n}$, since

$$\sigma_n b_{n,n} \equiv^+ \sigma_n\sigma_{n-1}\cdots\sigma_1\Delta_{2,n} \equiv^+ \Delta_{n-1}\sigma_n\sigma_{n-1}\cdots\sigma_1 \equiv^+ \Delta_n,$$

as desired. □

Definition 3.10 (Divisibility, multiple). We briefly introduced this notion in Section 2.3 but will remind the reader. In the context of monoids, and therefore positive braids, we say that a word a left-divides (*resp. right-divides*) b if there exists a c such that $b \equiv^+ ac$ (*resp. $b \equiv^+ ca$*). When this is the case, we say that b is a right-multiple of a and write $a \preceq b$.

For example, the braid $b = \sigma_1\sigma_2\sigma_3$ trivially divides $b' = \sigma_1\sigma_2\sigma_3\sigma_4$. However, $b \preceq b'' = \sigma_2\sigma_1\sigma_2\sigma_3$ as well since $b'' \equiv^+ \sigma_1\sigma_2\sigma_3\sigma_1$. Being a prefix is sufficient for divisibility, but certainly not necessary.

Lemma 3.11. *Let b be a braid in B_n^+ of length at most ℓ . Then Δ_n^ℓ is a right-multiple of b .*

Proof. Let b_1 be a braid of length ℓ . We will apply induction on ℓ . Lemma 3.9 covers the case when $\ell = 1$. For $\ell \geq 2$, suppose that $b_1 \neq \epsilon$ so that $b_1 = b'_1\sigma_i$, so that $|b'_1| = \ell - 1$. By the induction hypothesis, there exists a word b'_2 such that $b'_1b'_2 = \Delta_n^{\ell-1}$. Let φ_n be the function that takes σ_i and replaces it with σ_{n-i} . By 3.8, it follows that $b'_2\Delta_n \equiv \Delta_n\varphi(b'_2)$. Set $b_2 = b_{i,n}\varphi_n(b'_2)$, where $b_{i,n}$ is as in Lemma 3.9. It follows that

$$b_1b_2 \equiv^+ b'_1\sigma_1b_{i,n}\varphi_n(b'_2) \equiv^+ b'_1\Delta_n\varphi_n(b'_2) \equiv^+ b'_1b'_2\Delta_n \equiv^+ \Delta_n^{\ell-1}\Delta_n \equiv^+ \Delta_n^\ell.$$

□

We are now equipped to discuss common multiples.

CHAPTER 3. ENUMERATION

Theorem 3.12. *Any two positive braids admit a common right-multiple.*

Proof. Let b_1 and b_2 be braids and $\ell \geq \max\{|b_1|, |b_2|\}$. Then by Lemma 3.11, both b_1 and b_2 left-divide Δ_n^ℓ . \square

It is worth noting that a symmetric argument works for the existence of a common left-multiple.

Definition 3.13 (Least common right-multiple). A least common right-multiple of a finite set of braids A , denoted $\text{lcm}(A)$ is a braid b in which every element $a \in A$ left-divides b , and any other braid b' which also has this property is either equivalent to b or a right-multiple of b .

The notion of an lcm can be generalized to any monoid. There is also a symmetric definition for a left-lcm, but by convention, unless otherwise stated, we always talk about least common right-multiples. Whether or not an lcm for a given set is unique or not is unclear at this point. Answering this question is the goal of Section 3.2, and 3.32 answers it in the positive and gives an algorithm to compute it.

3.2 Least common multiples via subword reversing.

The goal of this section is to first prove that the lcm of two braids exists, and then come up with an algorithm that can compute it. In doing so, we will think of braids in a slightly different sense than we have previously. We will be manipulating braids with respect to their classical representation except that this time, instead of thinking of them as equivalence classes of products of Artin generators, we think of them as a *word* over the alphabet $\Sigma_n = \{\sigma_1, \dots, \sigma_{n-1}\}$. Since we will often be referring to this set with n fixed, define $\Sigma = \Sigma_n$ and $\Sigma^\pm = \{\sigma_1, \sigma_1^{-1}, \dots, \sigma_{n-1}, \sigma_{n-1}^{-1}\}$. The manipulations that we introduce respect the braid equivalences, however we distinguish braids which may be equivalent but have different presentations. However, we may wish to talk about braid words as braids, so we may go back and forth from talking about braids as words in Σ^* , to talking about equivalences

CHAPTER 3. ENUMERATION

between the braids that the words represent. This is where the distinction of \equiv and $=$ is of utmost importance.

Definition 3.14 (complement). Define $C : \Sigma \times \Sigma \rightarrow \Sigma^*$ as:

$$C(\sigma_i, \sigma_j) = \begin{cases} \sigma_j & \text{if } |i - j| \geq 2 \\ \sigma_j \sigma_i & \text{if } |i - j| = 1 \\ \epsilon & \text{if } i = j. \end{cases}$$

Let R be the set of all relations in B_n^+ as defined by relation \mathcal{R}_1 and relation \mathcal{R}_2 so

$$R = \{(\sigma_i \sigma_j, \sigma_j \sigma_i), (\sigma_k \sigma_{k+1} \sigma_k, \sigma_{k+1} \sigma_k \sigma_{k+1}) : 1 \leq i < j - 1 < n, 1 \leq k < n - 1\},$$

and $B_n^+ = \langle \Sigma : a \equiv^+ b, (a, b) \in R \rangle^+$. By construction of C , we can write

$$R = \{(\sigma_i C(\sigma_j, \sigma_i), \sigma_j C(\sigma_i, \sigma_j)) : 1 \leq i \leq j \leq n - 1\}. \quad (3.15)$$

It follows that for all σ_i and σ_j that $\sigma_i^{-1} \sigma_j \equiv C(\sigma_i, \sigma_j) C(\sigma_j, \sigma_i)^{-1}$ as braids in B_n . The function C is called a complement on the monoid B_n^+ . Complements are not unique to B_n^+ , and can be defined in any monoid where equation 3.15 holds. The idea is that the complement gives us an easy way to take inverse generators from the left hand side of a product to the right hand side in B_n .

Definition 3.16 (reversing). Let $b = x \sigma_i^{-1} \sigma_j y$, with $x, y \in \Sigma^{\pm*}$. We say that b is reversible in one step to b_1 if

$$b_1 = x C(\sigma_i, \sigma_j) C(\sigma_j, \sigma_i)^{-1} y.$$

We write $b \curvearrowright^1 b_1$. Notice that $b \equiv b_1$ as braids in B_n .

Furthermore, for $p \geq 1$, we say that b is reversible to b_p in p steps if there exist braids b_1, \dots, b_p such that $b \curvearrowright^1 b_1 \curvearrowright^1 b_2 \curvearrowright^1 \dots \curvearrowright^1 b_p$. When this is the case we write $b \curvearrowright^p b_p$, and sometimes omit the reference to p and simply say b is reversible to b_p . We call (b, b_1, \dots, b_p) the reversing sequence.

CHAPTER 3. ENUMERATION

Example 3.17. The word $\sigma_1\sigma_3^{-1}\sigma_2$ reverses to $\sigma_1C(\sigma_3, \sigma_2)C(\sigma_2, \sigma_3)^{-1} = \sigma_1\sigma_2\sigma_3\sigma_2^{-1}\sigma_3^{-1}$. In one move we were able to move all the inverse relations to the end of the word.

The idea of this function is to push the negative elements to the right hand side of a word that represents a braid. Geometrically, one may imagine sorting the crossings so that all of the negatively oriented crossings are at the bottom of the braid. As such, it is possible to iterate the function C until there are no more subwords of the form $\sigma_i^{-1}\sigma_j$. The theme of the next few lemmas is the form of words that are not reversible any further. Is it possible that every braid is reversible to a braid of the form $b_1b_2^{-1}$ with $b_1, b_2 \in \Sigma$? This question is not actually trivial — at this point it is still unclear whether or not the process of repeatedly reversing a braid word terminates.

Definition 3.18 (extended complement). Given a complement C on a monoid \overline{M} with generating set M , define the extended complement to be a function $C^* : M^* \times M^* \rightarrow M^*$ such that $C^*(u, v) = v'$ if and only if there exists a word u' such that $u^{-1}v$ reverses to $v'u'^{-1}$.

Certainly $C^*|_{M \times M} = C$, but the rest of the domain is still unclear. To tackle this problem we can use commutative diagrams.

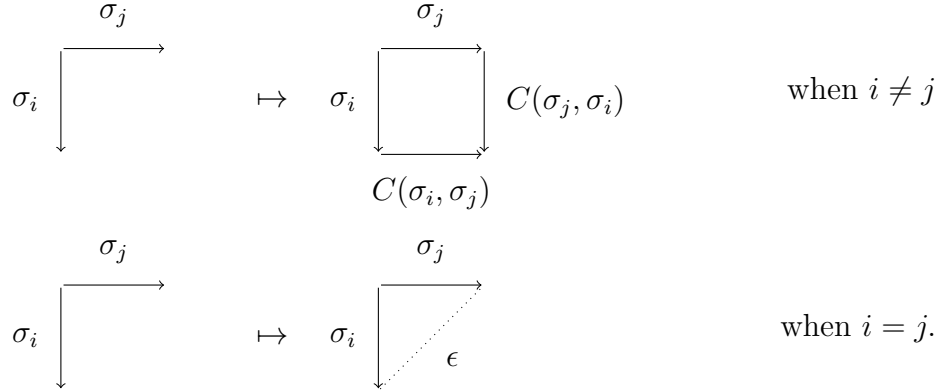
Definition 3.19 (reversing diagram). We can use a diagram as a way of visualizing a reversing sequence of words, and is subject to the rules outlined below. This type of diagram is called a reversing diagram.

Let $w = \sigma_{i_1}^{\varepsilon_1} \cdots \sigma_{i_k}^{\varepsilon_k}$ be a braid word. We begin a reversing diagram by drawing a series of connected arrows indexed by the letters of w . Starting with $\sigma_{i_1}^{\varepsilon_1}$, if $\varepsilon_1 = 1$, then the first arrow is horizontal, pointing to the right, and if $\varepsilon_1 = -1$, then the arrow is vertical, pointing downward. In both cases, the arrow is labelled by σ_{i_1} , omitting ε_1 in the label. We do this for the rest of the $\sigma_{i_j}^{\varepsilon_j}$'s, connecting the arrows, obtaining a staircase shape provided that the ε_j 's vary.

We proceed to fill in the diagram in the following way. We choose any northwest corner where two arrows meet tail to tail - call this an inside corner. We fill in the diagram using

CHAPTER 3. ENUMERATION

the following rules.



Note that the new arrows introduced by $C(\sigma_i, \sigma_j)$ and $C(\sigma_j, \sigma_i)$ may be equal to $\sigma_j\sigma_i$ and $\sigma_i\sigma_j$, respectively i.e. when $|i - j| = 1$. In this case that portion of the diagram gets two arrows, one for each letter.

We continue to fill out the diagram in every corner possible. We treat the dotted lines labeled by ϵ as if they do not exist. The diagram is commutative by definition of C , when following an arrow in the opposite direction, that counts as an inverse braid.

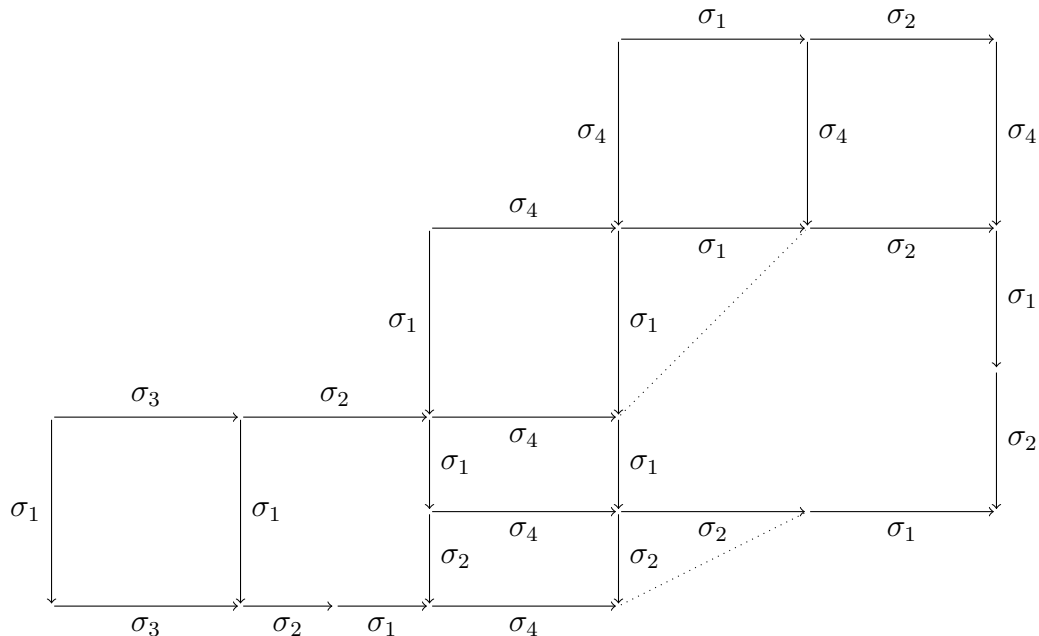
Example 3.20. Using a reversing diagram, we will reverse the braid

$$\sigma_1^{-1}\sigma_3\sigma_2\sigma_1^{-1}\sigma_4\sigma_4^{-1}\sigma_1\sigma_2.$$

This reversing diagram is finite, and we will soon learn that all reversing diagrams are too. Begin with the set of labelled arrows along the north west corners, and fill in each corner one at a time, noticing that some corners introduce multiple arrows. By counting the number of boxes and dotted lines, we can see that

$$\sigma_1^{-1}\sigma_3\sigma_2\sigma_1^{-1}\sigma_4\sigma_4^{-1}\sigma_1\sigma_2 \curvearrowright^{10} \sigma_3\sigma_2\sigma_1\sigma_4\sigma_1(\sigma_4\sigma_1\sigma_2)^{-1}.$$

CHAPTER 3. ENUMERATION



Since each inside corner defines a unique outside corner, the diagram is not determined by the order in which we fill in the corners. Reversing diagrams offer a way of visualizing reversing moves but begs the original question: for any given word, is there a reversing sequence that ends in a word of the form uv^{-1} with $u, v \in \Sigma^*$? That is, is there a maximal finite diagram for any given braid word w ? This requires a little more work to answer.

Lemma 3.21. *Given an extended complement on \overline{M} and two words $u, v \in M^*$, the words $C^*(u, v)$ and $C^*(v, u)$ exist if and only if the reversing diagram of $u^{-1}v$ is finite, in which case $uC^*(u, v) \equiv^+ vC^*(v, u)$.*

When this is the case, $C^(u, v)$ is the positive word obtained by following the arrows along the base of the diagram to the outside corner, and $C^*(v, u)$ is obtained by following the arrows from the end of the diagram (where the last arrow points) down to the outside corner.*

Proof. Suppose the diagram is finite. Then by definition of C^* , we have that $C^*(u, v) \equiv^+ v'$ if and only if $u^{-1}v$ reverses to $v'u^{-1}$. Setting v' to equal the bottom row of arrows and u'

CHAPTER 3. ENUMERATION

be the right-most column of arrows yields the result, since $u^{-1}v$ is reversible to any path in the diagram.

Conversely, if $C^*(u, v)$ and $C^*(v, u)$ exist, then it follows by definition of the extended complement that $u^{-1}v \rightsquigarrow C^*(u, v)C^*(v, u)^{-1}$. \square

Definition 3.22 (complete complement). A complement C on a monoid \overline{M} with generating set M is called *complete* when it has the following property for all u, u', v, v' in M^* in which $C^*(u, v)$ exists:

$$u \equiv^+ u' \text{ and } v \equiv^+ v' \text{ in } \overline{M} \quad \text{imply} \quad C^*(u, v) \equiv^+ C^*(u', v').$$

Lemma 3.23. *If C is a complete complement on the monoid \overline{M} then the following are equivalent for all $u, v \in M^*$:*

1. $u \equiv^+ v$ in \overline{M}
2. $C^*(u, v) = C^*(v, u) = \epsilon$
3. $u^{-1}v \rightsquigarrow \epsilon$

Proof. If $u \equiv^+ v$ then since $\epsilon\epsilon \rightsquigarrow \epsilon$, and by completeness it follows that $u^{-1}u \rightsquigarrow \epsilon$ and hence $C^*(u, u) = \epsilon$. The fact that $u^{-1}v \rightsquigarrow \epsilon$ implies $u \equiv^* v$ completes the equivalence. \square

If we can show that the complement defined for B_n^+ is complete, the above lemma will help us solve the braid isotopy problem.

Lemma 3.24 (completeness conditions). *Let C be a complement on \overline{M} . If*

1. for all $r, s \in M$, $|C(r, s)| = |C(s, r)|$,
2. for all $r, s, t \in M$,

$$C^*(u, sC(s, t)) \equiv^+ C^*(r, tC(t, s)) \text{ and } C^*(sC(s, t), r) \equiv^+ C^*(tC(t, s), r), \quad (3.25)$$

where the left hand side exists if and only if the right hand side exists,

CHAPTER 3. ENUMERATION

then C is complete.

An important detail is that since $r, s, t \in M$, the second condition states that C^* agrees with the \equiv^+ relation on the smallest level: when one argument is a letter and the other argument is one of the monoid relations in R (where R is the set of relations such that $\overline{M} = \langle M : R \rangle$). In equation 3.15 we saw that the braid relations are all instances of $rC(r, s) \equiv sC(s, r)$ where $r, s \in M$.

Proof. The goal is to show that C is complete, which means we need to show that for all $u, v, u', v' \in \overline{M}$ such that $u \equiv^+ u'$ and $v \equiv^+ v'$, we have $C^*(u, v) \equiv^+ C^*(u', v')$ and $C^*(v, u) \equiv^+ C^*(v', u')$.

By the first condition, length is preserved in C , so we can apply induction on the length of $uC^*(u, v)$. Our induction hypothesis is for $|uC^*(u, v)| = |vC^*(v, u)| = k < m$, we have

$$\text{if } u \equiv u' \text{ and } v \equiv v' \text{ then } C^*(u, v) \equiv^+ C^*(u', v') \text{ and } C^*(v, u) \equiv^+ C^*(v', u') \quad (3.26)$$

When $k = 0$ or $k = 1$, (3.26) is trivial. Since u and v play symmetric roles, for the induction step if we can prove for $k = m$

$$\text{if } v \equiv v' \text{ then } C^*(u, v) \equiv^+ C^*(u, v') \text{ and } C^*(v, u) \equiv^+ C^*(v', u) \quad (3.27)$$

then a symmetric argument would complete the induction step. As such we will assume (3.26) for $k < m$, and try to prove (3.27) for $|uC^*(u, v)| = |vC^*(v, u)| = k = m$.

Notice that for any two elements $v \equiv v' \in \overline{M}$, there exists a finite sequence of derivations starting from v and at each step replacing a single instance of $sC(s, t)$ with $tC(t, s)$ and ending in v' . If a result holds for all instances of $v_1sC(s, t)v_2 \equiv v_1tC(t, s)v_2$, where $v_1, v_2 \in \overline{M}$ then it holds for $v \equiv v'$. Therefore we can reduce the induction step to

$$\begin{aligned} &\text{if } v = v_1sC(s, t)v_2, \text{ and } v' = v_1tC(t, s)v_2 \text{ then} \\ &C^*(u, v) \equiv^+ C^*(u, v') \text{ and } C^*(v, u) \equiv^+ C^*(v', u) \end{aligned}$$

CHAPTER 3. ENUMERATION

All that remains is to do is to compare the reversing diagrams of $u^{-1}v$ and $u^{-1}v'$. Our assumption that $|uC^*(u, v)| = m$ means that $C^*(u, v)$ exists and is finite, so it is a good idea to start there. From here we can see that there exists $r \in M$ and $u_1, \dots, u_5, v_3, \dots, v_7 \in \overline{M}$

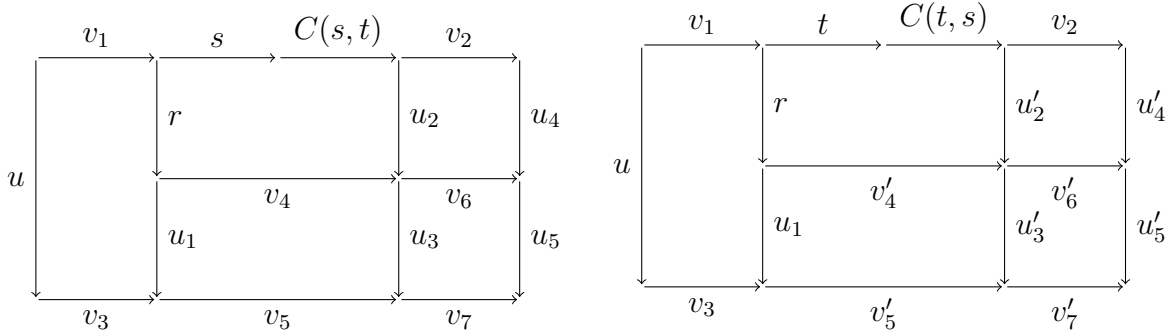


Figure 3.28: The reversing diagram for $u^{-1}v$ and $u^{-1}v'$.

such that $u^{-1}v_1 \simeq v_3u_1^{-1}r^{-1}$, and so on according to the reversing diagram. Note that it is possible that some of these are ϵ . If we compare this to the reversing diagram of uv'^{-1} , the only difference is that instead of $sC(s, t)$ across the top, we get $tC(t, s)$; the left-most corner is exactly the same in both diagrams. Now, since $r^{-1}sC(s, t) \simeq v_4u_2^{-1}$, it follows that $r^{-1}tC(t, s) \simeq v'_4, u_2'^{-1}$ with $u_2 \equiv^+ u_2'$ and $v_4 \equiv^+ v_4'$. By the induction hypothesis, a similar argument holds for every corner since C is compatible with equivalence for all u, v with $|uC^*(u, v)| < m$. So

$$v_3v_5v_7 \equiv v_3v_5'v_7' \quad \text{and} \quad u_4u_5 \equiv^+ u_4'u_5'.$$

All that remains is to notice that $C^*(u, v) = v_3v_5v_7 \equiv^+ v_3v_5'v_7' = C^*(u, v')$ and $C^*(v, u) = u_4u_5 \equiv^+ u_4'u_5' = C^*(v', u)$. \square

Theorem 3.29. *The braid complement C is complete.*

Proof. The proof is mostly clerical, as all of the work was done in Lemma 3.24. First, condition 1 is met by definition of C . Condition 2 is a relatively simple check. The letters

CHAPTER 3. ENUMERATION

u, v, w must be in Σ_n , so let $u = \sigma_i, v = \sigma_j, w = \sigma_k$. We need to verify equation 3.25 for all values $i, j, k \in \{1, \dots, n-1\}$. Of course, if the value holds for a particular value of i, j and k then it also holds for $i+1, j+1$ and $k+1$. So all that matters is the distance between i, j and k .

If $i = j$ or $j = k$ or $i = k$, the relations of 3.25 are satisfied trivially. Furthermore, suppose for example that $i = 1, j = 2, k = 4$ and 3.25 is satisfied. Then the result also holds for i, j and $k+1$. We can further reduce our cases to when the values $|i-j|, |i-k|$ and $|j-k|$ are equal to 1 or 2. It suffices to check that all of the values

$$\{i, j, k\} \in \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{1, 3, 5\}\}$$

satisfy equation 3.25.

We will prove one case, since the other cases are similar. Let $u = \sigma_2, v = \sigma_1$ and $w = \sigma_4$; we will check

$$\begin{aligned} C^*(\sigma_2, \sigma_1 C(\sigma_1, \sigma_4)) &\equiv^+ C^*(\sigma_2, \sigma_4 C(\sigma_4, \sigma_1)) \quad \text{and} \\ C^*(\sigma_1 C(\sigma_1, \sigma_4), \sigma_2) &\equiv^+ C^*(\sigma_4 C(\sigma_4, \sigma_1), \sigma_2). \end{aligned}$$

We have that $C(\sigma_1, \sigma_4) = \sigma_4, C(\sigma_4, \sigma_1) = \sigma_1$ so

$$\begin{aligned} C^*(\sigma_2, \sigma_1 \sigma_4) &= \sigma_1 \sigma_2 \sigma_4 \equiv^+ \sigma_4 \sigma_1 \sigma_2 = C^*(\sigma_2, \sigma_4 \sigma_1) \quad \text{and} \\ C^*(\sigma_1 \sigma_4, \sigma_2) &= \sigma_2 \sigma_1 = C^*(\sigma_4 \sigma_1, 2) \end{aligned}$$

as desired. □

For a complete proof of 3.29 see Appendix A.1.

Corollary 3.30. *The braid monoid is left-cancellative.*

Proof. Let $su \equiv^+ sv$. Since C^* is a complete complement, it follows that $u^{-1}s^{-1}sv \curvearrowright \epsilon$ and so $u^{-1}v \curvearrowright \epsilon$ and hence $u \equiv^+ v$. □

CHAPTER 3. ENUMERATION

The fact that the braid monoid is cancellative is especially important. Not only does it allow us to count positive braids, but we can also now apply 2.19 (Ore's Condition). That is, the natural embedding of the positive braid monoid into the braid group is injective. It follows that the distinction between \equiv and \equiv^+ is immaterial when comparing positive braids. For positive braids b_1, b_2 , we have $b_1 \equiv^+ b_2$ if and only if $b_1 \equiv b_2$. From this point forward we will drop the $^+$ notation with the new ability to speak of equivalence without being tied to the context of the braid monoid. If two positive braids are equivalent in the braid group, they are equivalent in the braid monoid as well.

Theorem 3.31. *The image $C^*(u, v)$ is always defined for any braids $u, v \in S^*$.*

Proof. By 3.12, there exist words u' and v' such that $uu' \equiv^+ vv'$, and by Lemma 3.23 $u'^{-1}u^{-1}vv' \curvearrowright \epsilon$ and hence is finite. The reversing diagram of $u^{-1}v$ is inherently inside the reversing diagram of $u'^{-1}u^{-1}vv'$ so it is also finite. \square

Theorem 3.32. *Any two braids admit a unique least common multiple.*

Proof. The function $C^* : B_n^+ \times B_n^+ \rightarrow B_n^+$ has the following properties for all u, v, u' and v' in B_n^+ :

$$uC^*(u, v) \equiv vC^*(v, u) \tag{3.33}$$

and if $uv' \equiv vu'$ then there exists a w such that

$$u' \equiv C^*(v, u)w \text{ and } v' \equiv C^*(u, v)w, \tag{3.34}$$

and so we conclude that w is a multiple of u' and v' .

Let b_1 and b_2 be in B_n^+ , and let $b_3 = b_1C^*(b_1, b_2) \equiv b_2C^*(b_2, b_1)$, by equation 3.33. That is, b_3 is a common multiple of b_1 and b_2 . We will show that b_3 is the unique (up to equivalence) least common multiple.

Let b'_3 be any common multiple of b_1 and b_2 . Then there exist b'_1 and b'_2 such that $b'_3 \equiv b_1b'_2 \equiv b_2b'_1$. By equation 3.34, there exists a braid w such that

$$b'_1 \equiv C^*(b_2, b_1)w \text{ and } b'_2 \equiv C^*(b_1b_2)w.$$

CHAPTER 3. ENUMERATION

It follows that $b'_3 \equiv b_1 b'_2 \equiv b_1 C^*(b_1 b_2) w = b_3 w$, so b'_3 is a multiple of w . Thus $\text{lcm}(b_1, b_2) \equiv b_1 C^*(b_1, b_2) \equiv b_2 C^*(b_2, b_1)$.

For uniqueness, note that divisibility is antisymmetric in the braid monoid — if $x \equiv yz$ and $y \equiv xz'$ then simply measuring the length of each side implies that $x \equiv y$. Hence if there exists another lcm b'_3 , then $b_3 \preceq b'_3$, $b'_3 \preceq b_3$ and hence $b_3 \equiv b'_3$. \square

3.3 How to count positive braids.

We need only one more definition to start counting braids.

Definition 3.35 (clique). Given a monoid with generator set $M = \{s_1, \dots, s_n\}$, a clique is a subset of M that admits a common right-multiple.

Since any set of braids admits a common right-multiple, any subset of $\{\sigma_1, \dots, \sigma_n\}$ is a clique.

The following lemma, which is the main result for this section, makes use of the set $\mathbb{Z}\langle \overline{M} \rangle$, where \overline{M} is a monoid. Here $\mathbb{Z}\langle \overline{M} \rangle$ is the set of (possibly) infinite formal linear combinations of elements of \overline{M} , with coefficients in \mathbb{Z} . The product of two such linear combinations is calculated as follows:

$$\sum_{m \in \overline{M}} a_m m \cdot \sum_{m \in \overline{M}} b_m m = \sum_{m \in \overline{M}} c_m m,$$

where $c_m = \sum_{xy=m} a_x b_y$. Here xy is the monoid product of x and y .

Lemma 3.36 (Albenque & Nadeau [2]). *Let M be a left-cancellative monoid with generator set S and identity element ϵ , and has the property that any set that admits a common right-multiple also admits a least common right-multiple (lcm). Let \mathcal{Q} be the set of all cliques of M . Then the following equation holds in $\mathbb{Z}\langle M \rangle$:*

$$\left(\sum_{Q \in \mathcal{Q}} (-1)^{|Q|} \text{lcm}(Q) \right) \cdot \left(\sum_{m \in M} m \right) = \epsilon. \quad (3.37)$$

CHAPTER 3. ENUMERATION

Proof. In the interest of simpler notation, for any set $A \subset M$, write $m_A = \text{lcm}(A)$. For $m \in M$, let $\mathcal{Q}(m) \subseteq \mathcal{Q}$ be the subsets Q of S in which every $s \in Q$ left-divides m .

Since every subset of \mathcal{Q} has a least common multiple, it follows that $\text{lcm}(A) \preceq \text{lcm}(B)$ if and only if $A \subseteq B$ for $A, B \in \mathcal{Q}$. Therefore for any $m \in M$ there exists a unique set $Q_m \subseteq S$ such that $\mathcal{Q}(m) = \mathcal{P}(Q_m)$ (the power-set of Q_m).

Define an arbitrary order $<$ on S . Define a function $s : M \setminus \epsilon \rightarrow S$ by $s(m) = \max_{<} \{Q_m\}$.

Fix $m \in M$, and define a sign-reversing involution on $\mathcal{Q}(m)$: $\Phi_m(Q) = Q \Delta s(m)$, the symmetric difference of Q and $s(m)$. This is a sign-reversing involution since Φ_m affects the parity of $|Q|$, and $\Phi_m^2(Q) = Q$; therefore for $m \neq \epsilon$

$$\sum_{Q \in \mathcal{Q}(m)} (-1)^{|Q|} = 0. \quad (3.38)$$

Notice that by construction of $\mathcal{Q}(m)$, if $Q \in \mathcal{Q}(m)$ then $m_Q \preceq m$, which happens exactly when there exists an m' such that $m_Q m' = m$. Thus, when we fix m , equation 3.38 is equivalent to

$$\sum_{\substack{(Q, m') \in \mathcal{Q} \times M \\ m_Q m' = m}} (-1)^{|Q|} = \begin{cases} 0 & \text{if } m \neq \epsilon \\ 1 & \text{if } m = \epsilon. \end{cases}$$

Returning to the left hand side of equation 3.37, we can write

$$\left(\sum_{Q \in \mathcal{Q}} (-1)^{|Q|} m_Q \right) \cdot \left(\sum_{m' \in M} m' \right) = \sum_{m' \in M} \left(\sum_{\substack{(Q, m') \in \mathcal{Q} \times M \\ m_Q m' = m}} (-1)^{|Q|} \right) m' = \epsilon$$

as desired. □

CHAPTER 3. ENUMERATION

Corollary 3.39 (Albenque [1]). *The generating function for the positive braid monoid B_n^+ is*

$$B_n(x) = \sum_{b \in B_n^+} x^{|b|} = \frac{1}{\sum_{Q \in \mathcal{Q}} (-1)^{|Q|} x^{|m_Q|}},$$

where \mathcal{Q} is the clique set for B_n^+ .

Proof. We will prove the result for homogeneous monoids first, and the result will apply to the braid monoid. Assume that M is a homogeneous monoid, that is, if $m_1 \equiv m_2$ then $|m_1| = |m_2|$. Then the following is a ring homomorphism from $\mathbb{Z}\langle M \rangle$ to $\mathbb{Z}[x]$:

$$\sum_{m \in M} c_m m \mapsto \sum_{m \in M} c_m x^{|m|}.$$

Applying this to both sides of Lemma 3.36 gives us

$$\left(\sum_{Q \in \mathcal{Q}} (-1)^{|Q|} x^{|m_Q|} \right) \cdot \left(\sum_{m \in M} x^{|m|} \right) = 1.$$

This applies to the braid monoid because B_n^+ is homogeneous, every set of braids admits an lcm, and B_n^+ is cancellative. \square

The generating function is certainly not in a closed form, but is still easy to calculate. Define the set $Q_{i,j} = \{\sigma_i, \dots, \sigma_{j-1}\}$ for $i < j$. Given any $Q \in \mathcal{Q}$, we can write Q as a disjoint union of $Q_{i,j}$'s. One can use an inductive argument using the subword reversing algorithm to determine that $\text{lcm}(Q_{i,j}) = \Delta_{i,j}$. From there, it is easy to see that for $Q = \cup_{(i,j) \in \Gamma} Q_{i,j}$,

$$m_Q = \text{lcm}(Q) = \prod_{(i,j) \in \Gamma} \Delta_{i,j},$$

which is the braid product of $\Delta_{i,j}$ over the counting set Γ .


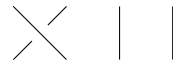


CHAPTER 3. ENUMERATION

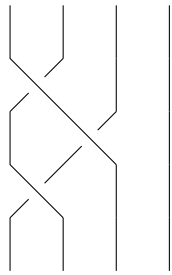
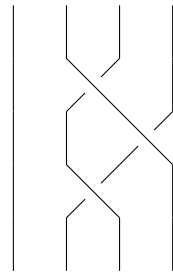
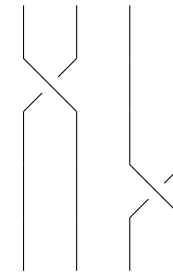
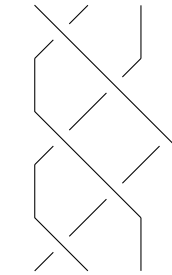
Example 3.40. We apply 3.39 to determine the generating function for B_4^+ .

Recall that the generators for B_4^+ are $\{\sigma_1, \sigma_2, \sigma_3\}$. It follows that the clique set for B_4^+ is equal to

$$\mathcal{Q} = \{Q_i\}_{i=0}^7 = \{\emptyset, \{\sigma_1\}, \{\sigma_2\}, \{\sigma_3\}, \{\sigma_1, \sigma_2\}, \{\sigma_2, \sigma_3\}, \{\sigma_1, \sigma_3\}, \{\sigma_1, \sigma_2, \sigma_3\}\}.$$

The lcm, and all of the relevant information for each of these cliques in order is:

			
$\text{lcm}(Q_0) = \epsilon$	$\text{lcm}(Q_1) = \sigma_1$	$\text{lcm}(Q_2) = \sigma_2$	$\text{lcm}(Q_3) = \sigma_3$
$ Q_0 = 0, \epsilon = 0$	$ Q_1 = 1, \sigma_1 = 1$	$ Q_2 = 1, \sigma_2 = 1$	$ Q_3 = 1, \sigma_3 = 1$

			
$\text{lcm}(Q_4) = \Delta_2$	$\text{lcm}(Q_5) = \Delta_{2,3}$	$\text{lcm}(Q_6) = \sigma_1\sigma_3$	$\text{lcm}(Q_7) = \Delta_3$
$ Q_4 = 2, \epsilon = 3$	$ Q_5 = 2, \sigma_1 = 3$	$ Q_6 = 2, \sigma_2 = 2$	$ Q_7 = 3, \sigma_3 = 6$

It follows that the generating function for B_4^+ is

$$B_4(x) = \frac{1}{1 - 3x + 2x^3 + x^2 - x^6}.$$

Chapter 4

Isotopy Problem

In Chapter 2 we covered some of the many different ways one can represent a braid. In each representation it was clear that there are many different ways to represent the same braid. Given two equivalent braids, it is often easy to prove that they are equivalent (a list of relations that transforms one to the other will suffice), but proving that two braids are not equivalent can take some more work. This is called the isotopy problem.

4.1 Invariants and algorithms.

There are a number of quick tests which tell you that two braids are not the same. For example, braids have a natural homomorphism π onto the symmetric group, with kernel equal to the pure braid group P_n , as discussed in Section 2.5. The image of b under π is called the permutation of b . This homomorphism can be an easy way to tell if two braids are not equal and is easy to compute. The problem with an invariant is that just because the image of two braids might be equal does not imply that the braids are equivalent.

Definition 4.1 ((complete) isotopy invariant). A map φ from the braid group B_n to some set X such that if $\varphi(b) \neq \varphi(b')$ then $b \not\equiv b'$ is called an isotopy invariant. A complete isotopy invariant is an invariant with the extra property that if $\varphi(b) = \varphi(b')$ then $b \equiv b'$.

CHAPTER 4. ISOTOPY PROBLEM

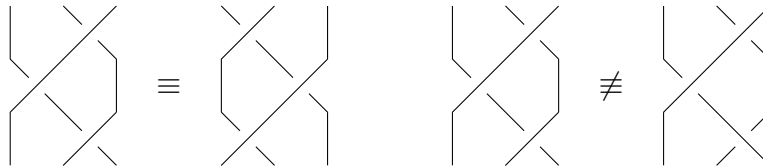


Figure 4.2: Equivalent braids imply equal permutations, but the converse is false.

Finding a complete isotopy invariant is a solution to the braid isotopy problem. When braids are expressed as a word, a complete isotopy invariant is called a solution to the braid word problem. As we will see more than once in this chapter, one does not require the set X in a complete isotopy invariant to be entirely different from the preimage B_n . In fact, we may find a subset of B_n which suffices to serve as X , i.e., if we can find a one-to-one function f from $B_n \rightarrow B_n$ such that $b \equiv f(b)$, we call $f(b)$ a *normal form* of b with respect to f and is indeed a solution to the isotopy problem.

To solve the braid isotopy problem, we do not require a complete isotopy invariant or a normal form. Sometimes all we seek is an algorithm. It should be fairly obvious that the braid isotopy problem for positive braids is decidable, since the equivalence class of each positive braid is finite (as opposed to the always infinite equivalence class for each braid $b \equiv b\sigma_1\sigma_1^{-1}$). Consider Algorithm 4.3, which takes as input two braids written in terms of their Artin generators and returns as output whether or not they are equivalent. For brevity, let R be a function from $B_n^+ \rightarrow \mathcal{P}(B_n^+)$ (the power-set of B_n^+) which takes as input a braid and returns the finite set of all braids which are obtained by applying exactly one braid relation to it wherever possible.

The algorithm will end since the *for* loop will end when either $b_2 \in X$, or $R(x) = X$ for each $x \in X$, in which case X is the entire equivalence class of $b_1b_2^{-1}$. If $\epsilon \in X$ then the algorithm ends with a positive result, since $b_1b_2^{-1} \equiv \epsilon$ implies $b_1 \equiv b_2$. Otherwise, there is no finite sequence of relations that take $b_1b_2^{-1}$ to ϵ so $b_1 \not\equiv b_2$.

The ability to recognize braid equivalence is important to us because of cryptographic applications of braid theory. The involvement in a cryptosystem requires us to be able to efficiently recognize whether or not braids are equivalent; Algorithm 4.3 is not efficient.


```

input :  $b_1, b_2$ 
 $X = \{b_1\}$ 
for  $x \in X$  do
  |  $X := X \cup R(x)$ 
  | if  $b_2 \in X$  then
  | | return " $b_1 \equiv b_2$ "
  | end
end
return " $b_1 \not\equiv b_2$ "

```

Algorithm 4.3: Naïve algorithm.

4.2 Artin's algorithm.

Artin offers a solution to the braid isotopy problem, albeit a little clumsy and inefficient, and computer implementation is unclear. That said, it is a good example of writing braids in a normal form, and an appropriate one to start with as it was the first algorithm in history to solve the braid isotopy problem.

Consider a pure braid $b \in P_n$, which connects the points P_i to Q_i by curves C_i for $i = 1, \dots, n + 1$. Let b_1 be the braid obtained by removing C_1 from b and replacing it with the curve D_1 , which connects P_1 to Q_1 by a straight line that does not interfere with any other strands. Define $c_1 = bb_1^{-1}$. Then c_1 has the special property that if you remove the curve C_1 from c_1 and replace it with D_1 then you are left with $b_1b_1^{-1} = \epsilon$. This type of braid is called a 1-pure braid. A similar definition exists for an i -pure braid.

Disregarding the fact that C_1 is still in the braid c_1 , performing the same isotopy which takes $b_1b_1^{-1}$ to ϵ (which can be easily found by undoing opposite crossings from the centre outwards), and stretching c_1 as much as necessary - we will find that C_1 is usually tangled up in the otherwise identical (that is to say equal to ϵ) braid. This braid c_1 is called a 1-pure braid, since by removing the first strand C_1 and replacing it with D_1 , we obtain the identity. Now, notice that $bb_1^{-1} \equiv c_1$ so $b \equiv c_1b_1$. That is to say, any braid is isotopic to the product of a 1-pure braid c_1 , and a braid b_1 with its first strand not interfering with the rest of the braid. In other words, we may think of b_1 as a braid in B_{n-1} but a new strand

CHAPTER 4. ISOTOPY PROBLEM

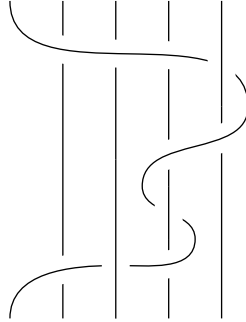


Figure 4.4: An example of a 1-pure braid.

is introduced on the left so that it is in fact in B_n . It follows that any pure braid on $n + 1$ strands has unique decomposition $b = c_1 c_2 \dots c_{n+1}$, where each c_i is i -pure. It should be clear that in the decomposition of ϵ , each $c_i = \epsilon$.

Let φ be the function that takes as input a braid $b \in B_n$ and gives as output the 1-pure braid c_1 and the braid $b_1 \in B_{n-1}$. Consider Algorithm 4.5 more of a proof of concept, or

```

input :  $b_1, b_2$ 
 $b = b_1 b_2^{-1}$ 
if  $b$  is not pure then
  | return " $b_1 \not\equiv b_2$ "
end
for  $i = 1, \dots, n$  do
  |  $\varphi(b) = (b_i, c_i)$ 
  | if  $c_i \neq \epsilon$  then
  | | return " $b_1 \not\equiv b_2$ "
  | end
  |  $b := b_i$ 
end
return " $b_1 \equiv b_2$ "

```

Algorithm 4.5: Artin's algorithm

an exercise than one to be implemented in a computer program. Instead of improving this algorithm or studying its complexity, it is wise to move on to more efficient algorithms that

may actually have the prospect of being used in a cryptographic application.

4.3 Subword reversing.

In Section 3.2 we defined the extended complement $C^* : \Sigma_n^* \times \Sigma_n^* \rightarrow \Sigma_n^*$ which calculates the least common multiple of two braids. In this section we will show that C^* can be used to solve the word problem on braids.

Recall that the function $C^*(u, v)$ is always defined for positive words u, v (see 3.31), and that $u \equiv v$ if and only if $C^*(u, v) = C^*(v, u) = \epsilon$, if and only if $u^{-1}v \curvearrowright \epsilon$ (see Lemma 3.23). As such, the braid isotopy problem can be directly solved for positive braids.

```

input : positive braid words  $b_1, b_2$ 
if  $C^*(b_1, b_2) = \epsilon$  and  $C^*(b_2, b_1) = \epsilon$  then
  | return " $b_1 \equiv b_2$ "
else
  | return " $b_1 \not\equiv b_2$ "
end
    
```

Algorithm 4.6: Subword reversing for positive braids.

Subword reversing can be used to solve the isotopy problem for general braids too. We just have to reverse the word twice.

Lemma 4.7. *Any braid word b is reversible in a finite number of steps to uv^{-1} for positive braids u, v .*

Proof. We can decompose b into a composition of positive braids $u_1, v_1, \dots, u_k, v_k$ such that $b = u_1^{-1}v_1u_2^{-1}v_2 \cdots u_k^{-1}v_k$. If $k = 1$ then the result follows from 3.31. Suppose that the result holds for $k - 1$. Then $b \curvearrowright u'v'^{-1}u_k^{-1}v_k$. But $(u_kv')^{-1}v_k \curvearrowright u''v^{-1}$ for positive words u'' and v . Let $u = u'u''$, so that $b \curvearrowright u'u''v^{-1} = uv^{-1}$, as desired. \square

CHAPTER 4. ISOTOPY PROBLEM

The idea of Algorithm 4.8 is that $b_1 \equiv b_2$ if and only if $b = b_1 b_2^{-1} \equiv \epsilon$. We reverse b to uv^{-1} and note that $b \equiv \epsilon$ if and only if $u \equiv v$ if and only if $u^{-1}v \curvearrowright \epsilon$. Subword reversing is the first efficient algorithm we have seen so far.

```
input : braid words  $b_1, b_2$ 
reverse  $b_1 b_2^{-1} \curvearrowright uv^{-1}$  with  $u, v$  positive
if  $C^*(u, v) = \epsilon$  and  $C^*(v, u) = \epsilon$  then
|   return " $b_1 \equiv b_2$ "
else
|   return " $b_1 \not\equiv b_2$ "
end
```

Algorithm 4.8: Subword reversing for general braids.

4.4 The greedy normal form.

We have seen that the equivalence classes for braids are infinite, so the motivation to have a unique representative for that equivalence class is strong. In this section we will use the theory of Garside [34] to determine a canonical form for a braid, and refer to [31] to analyze the algorithms. Although the foundation of this work is due to Garside, we follow the notation and proofs of Dehornoy [26] for consistency.

We know how to compute the lcm of a set of braids, but we have not yet defined its dual - the greatest common divisor for B_n^+ .

Definition 4.9 (greatest common divisor). Let b_1 and b_2 be positive braids. Then $c \equiv \gcd(b_1, b_2)$ if and only if c left divides¹ b_1 and b_2 , and any other c' with this property also has the property that $c' \preceq c$.

Notice that we do not have an algorithm to compute the gcd like we do for the lcm. The following proof will give us insight on how to compute it.

¹Recall that we omit the "left" part of left divide and stick to left division. If we wanted to we could define the above as "left gcd" and symmetrically define a "right gcd."

CHAPTER 4. ISOTOPY PROBLEM

Lemma 4.10. *The gcd of any two positive braids exists and is unique.*

Proof. Set X to be the finite set of positive braids that divide b_1 and b_2 , and set $c = \text{lcm } X$. The set X exists and is finite since $\epsilon \in X$ and if $a \preceq b$ then $|a| \leq |b|$. By definition of lcm, $c \in X$, and all $x \in X$ divide c . \square

The idea is that we take all divisors of b and c and compute their lcm. For large braids (in terms of length) this can be difficult to compute.

Definition 4.11 (greedy normal form). Let b be any positive braid and define the *head* of b to be $H(b) = \text{gcd}(\Delta_n, b)$. We say that a sequence of positive braids (b_1, b_2, \dots, b_k) is a *normal sequence* if $b_i = H(b_i b_{i+1} \cdots b_k)$ for $i = 1, \dots, k-1$. We say that a positive braid $b \in B_n^+$ is in *greedy normal form* if $b = b_1 b_2 \cdots b_k$ and (b_1, b_2, \dots, b_k) is normal.

Theorem 4.12. *The greedy normal form of a positive braid is unique: $b_1 \equiv b_2$ if and only if they are both equivalent to a unique braid in greedy normal form.*

Partial proof. We will show that any positive braid has a normal form. Simply compute braids b_1, b_2 such that $b \equiv b_1 b_2$ where $b_1 = \text{gcd}(\Delta_n, b)$, and repeat on b_2 . These braids can be computed using the extended complement function from Section 4.3.

Keep doing this until we are left with a divisor of Δ_n . This terminates by the cancellative nature of B_n^+ , and since $\text{gcd}(b, \Delta_n)$ always exists.

What remains to be shown is uniqueness. Notice that uniqueness up to equivalence will not suffice. We need a unique braid word, which would come from a unique head of a positive braid. For this, we need to understand the theory of *permutation braids* and *simple braids*. \square

A greedy normal form gives us a distinguished representative for the equivalence classes of positive braids. We will see that this is not only a solution to the braid isotopy problem, but gives us a new way to store a braid in a computer. Having an efficient way of computing a normal form will change the way we work with braids.

CHAPTER 4. ISOTOPY PROBLEM

Our work here is far from done. In addition to finishing the above proof, if we want to effectively use the greedy normal form as a solution to the braid isotopy problem or use it in any computer environment, an efficient algorithm for computing the greedy normal form is of utmost importance. Since we currently have no efficient way of computing the gcd of a set of braids, our current algorithm is not efficient.

The existence of this normal form is a classical result appearing first in Garside's 1969 paper, [34] although his method is unconcerned with efficiency. Different algorithms for finding this canonical form can also be found in [30, 31], but we will stick with the methods used by Dehornoy [26].

In Section 3.1 we defined a braid $\delta_{i,j} = \sigma_{j-1}\sigma_{j-2}\cdots\sigma_i$ and proved a number of propositions involving it. Here we will introduce $\beta_{i,j} = \sigma_i\sigma_{i+1}\cdots\sigma_{j-1}$ and assume a few of the symmetrical results of $\delta_{i,j}$. In particular, notice that $\beta_{i,j}\sigma_k \equiv \sigma_{k+1}\beta_{i,j}$ whenever $i \leq k \leq j$.

Braids have a close relationship with the symmetric group. We have seen mappings from the braid group to \mathcal{S}_n but not the other way around. Consider this one.

Definition 4.13 (permutation braid). Let f be a permutation in \mathcal{S}_n . We can recursively define a positive braid from f with the following function. Let $\text{br}(1) = \epsilon$ and

$$\text{br}(f) = \beta_{f(k),k} \text{br}(g),$$

where $k = \max\{s \in \{1, \dots, n\} : s \neq f(s)\}$ (the largest number moved by f) and $g \in \mathcal{S}_n$ is defined as

$$g(i) = \begin{cases} f(i) & \text{when } i < k \text{ and } f(i) < f(k) \\ f(i) - 1 & \text{when } i < k \text{ and } f(i) > f(k) \\ i & \text{when } i \geq k. \end{cases}$$

The idea is that we connect the $f(k)$ -th strand to the k -th strand starting with $k = n$ and working down. If a positive braid b is in the set $\{\text{br}(f) : f \in \mathcal{S}_n\}$ then we call b a *permutation braid*. Since we are concerned with uniqueness, we need equality: $b \equiv \text{br}(f)$, does not imply b is a permutation braid.

Example 4.14. Take $f = (1743)(26)(5) \in \mathcal{S}_7$. The corresponding permutation braid in

CHAPTER 4. ISOTOPY PROBLEM

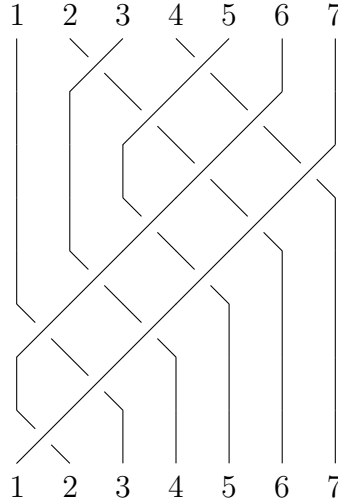


Figure 4.15: The permutation braid $\text{br}((1743)(26)(5))$.

B_7^+ is $\beta_{47}\beta_{26}\beta_{35}\beta_{24}\beta_{13}\beta_{12}$ with intermediate permutations $(1625431)(7)$, $(153)(24)(6)(7)$, $(1426)(5)(6)(7)$, $(13)(2)(4)(5)(6)(7)$ and $(12)(3)(4)(5)(6)(7)$.

It should be fairly clear that the permutation that corresponds to a braid as in Section 2.5 is closely related to permutation braids. Indeed $\pi(\text{br}(s)) = s$ for all $s \in \mathcal{S}_n$. Recall the important braid Δ_n . The permutation $\pi(\Delta_n) = \omega_n \in \mathcal{S}_n$ is the permutation such that for all $1 \leq i \leq n-1$, $\omega_n(i) = n-i$. It also happens that $\text{br}(\omega_n) = \Delta_n$. In general $\text{br}(\pi(b)) \neq b$.

Permutation braids have a number of very nice properties. The first thing to notice is that every two strands cross exactly once. Many more properties are to come.

Definition 4.16 (simple braid). Let f be a permutation on n and define the inversion number of f to be $\text{Inv}(f) = |\{(i, j) : 1 \leq i < j \leq n, f(i) > f(j)\}|$. We say that a positive braid is simple if its length is equal to the inversion number of its permutation. That is b is simple if and only if $|b| = \text{Inv}(\pi(b))$.

Although defined differently, simple braids are actually exactly the permutation braids. It turns out that it is extremely useful to be able to use the defining property of simple

CHAPTER 4. ISOTOPY PROBLEM

braids when talking about permutation braids. To prove this, we need more of the theory Dehornoy provides in [26].

Lemma 4.17. *Let b_1 and b_2 be positive braids such that b_1b_2 is simple. Then b_1 and b_2 are simple too.*

Proof. We start by making the claim that for all positive braids b_1 and b_2 , the inequality

$$\text{Inv}(\pi(b_1b_2)) \leq |b_1| + \text{Inv}(\pi(b_2)). \quad (4.18)$$

holds. This follows inductively from the fact that

$$\text{Inv}(s_i f) = \begin{cases} \text{Inv}(f) + 1 & \text{if } f^{-1}(i) < f^{-1}(i+1) \\ \text{Inv}(f) - 1 & \text{if } f^{-1}(i) > f^{-1}(i+1) \end{cases} \quad (4.19)$$

for all $s_i \in \mathcal{S}_n$. To see this, notice that s_i will change the inversion number by ± 1 . If $f^{-1}(i) < f^{-1}(i+1)$ then $(s_i f)^{-1}(i) > (s_i f)^{-1}(i+1)$ so the inversion number is increased by one. A similar argument holds for when the inversion number is decreased by one.

Finally, to see that equation 4.18 is all we need, notice that if b_2 is not simple then $\text{Inv}(\pi(b_2)) < |b_2|$ and then $|\text{Inv}(\pi(b_1b_2))| < |b_1| + |b_2|$, contradicting that b_1b_2 is simple. So b_2 is simple, and a symmetric argument holds for b_1 . \square

The above lemma has a clear catch-line: simple braids decompose into smaller simple braids. The next lemma's catch-line is less clear but it is a step in the opposite direction: how can we construct a simple braid from a permutation braid? It will be used as a technical tool for the proof that simple braids are exactly the permutation braids.

Lemma 4.20. *Let $f \in \mathcal{S}_n$ such that $\text{br}(f)$ is simple. Then for all $1 \leq i \leq n-1$ we have either*

1. $f^{-1}(i) < f^{-1}(i+1)$ and $\sigma_i \text{br}(f) = \text{br}(s_i f)$ is simple or
2. $f^{-1}(i) > f^{-1}(i+1)$ and $\sigma_i \text{br}(f)$ is not simple.

CHAPTER 4. ISOTOPY PROBLEM

Proof. Most of the theorem is worked out from equation 4.19. In case 1, since $\text{br}(f)$ is simple, then $|\sigma_i \text{br}(f)| = \text{Inv}(s_i f)$. In case 2, $|\sigma_i \text{br}(f)| = \text{Inv}(f) + 1$, and $\text{Inv}(s_i f) = \text{Inv}(f) - 1$. The only thing that remains to show is that in the first case, $\sigma_i \text{br}(f) = \text{br}(s_i f)$. For this we do induction on k , the largest number moved by f .

If $k = 1$, then f is the identity and the result is trivial. So assume $k > 1$. By definition of k , $f(k) < k$, and $\text{br}(f) = \sigma_{f(k),k} \text{br}(g)$ where g is defined in 4.13. Now, set $f' = s_i f$, and g' the associated braid as per 4.13.

The rest of the proof can be split up into five cases, based i .

1. When $i < f(k) - 1$. Then the largest number moved by f' is k as well. So $g' = s_i g$, and $\text{br}(s_i g) = \sigma_i \text{br}(g)$ by the induction hypothesis. Now, since $i < f(k) - 1$ we get that $\sigma_i \sigma_{f(k),k} \equiv \sigma_{f(k),k} \sigma_i$, hence

$$\text{br}(f') = \sigma_{f(k),k} \text{br}(g') = \sigma_{f(k),k} \sigma_i \text{br}(g) = \sigma_i \sigma_{f(k),k} \text{br}(g) = \sigma_i \text{br}(f).$$

2. When $i = f(k) - 1$. Now, the largest number moved by f' is still k but

$$s_i f(k) = (f(k) - 1, f(k)) f(k) = f(k) - 1$$

and $g' = g$. Hence

$$\text{br}(f') = \sigma_{f(k)-1,k} \text{br}(g') = \sigma_i \sigma_{f(k),k} \text{br}(g) = \sigma_i \text{br}(f).$$

3. When $i = f(k)$ we violate the hypothesis that $f^{-1}(i) < f^{-1}(i + 1)$.
4. When $f(k) \leq i \leq k - 1$, the largest number moved by f' is k , and $f(k) = f'(k)$, and $g' = s_{i-1} g$. Hence

$$\text{br}(f') = \sigma_{f(k),k} \text{br}(g') = \sigma_{f(k),k} \sigma_{i-1} \text{br}(g) = \sigma_i \sigma_{f(k),k} \text{br}(g) = \sigma_i \text{br}(f).$$

5. When $i \geq k$, then the largest number moved by f' is $i + 1$, and $f'(i + 1) = i$, and $g' = f$. It follows immediately that $\text{br}(f') = s_i \text{br}(f)$.

□

We are finally equipped to show that simple braids are exactly the permutation braids.

Theorem 4.21. *A braid is simple if and only if it is a permutation braid.*

Proof. Assume that b is simple. We will show by induction on the length of b that $b = \text{br}(\pi(b))$. If $|b| = 0$ or 1 , the result is trivial. So suppose that all simple braids b' with $2 \leq |b'| < k$ are permutation braids. Let b be any simple braid of length k so that $b = \sigma_i b'$ and $\pi(b) = s_i \pi(b')$. Since simple braids decompose into simple braids, b' must be simple and therefore a permutation braid by the induction hypothesis. By Lemma 4.20, $b = \sigma_i \text{br}(\pi(b')) = \text{br}(s_i \pi(b')) \text{br}(\pi(b))$ so b is a permutation braid.

Now assume that b is a permutation braid, and that $\pi(b) = f$. We will do induction on $\text{Inv}(f)$. When $\text{Inv}(f) = 0$, $f = \text{id}$ which is simple. Suppose $\text{Inv}(f) \geq 1$, then there exists an i such that $f^{-1}(i) > f^{-1}(i+1)$. Let $g = s_i f$ so that $\text{Inv}(g) < \text{Inv}(f)$. Also notice that $s_i g = s_i s_i f = f$. By the induction hypothesis, g is simple, and by Lemma 4.20, $\text{br}(f) = \sigma_i \text{br}(g)$ is simple. □

Now that we know permutation braids enjoy all the nice properties of simple braids, we get the following theorem.

Theorem 4.22. *The permutation braids of \mathcal{S}_n are exactly the left and right divisors of Δ_n .*

Proof. If $f = \omega_n = (n, n-1, \dots, 1)$, then $\text{br}(\omega_n) = \Delta_n$ which of course is a divisor of itself. Assume that $f \neq \omega_n$. Then there must exist an i such that $f^{-1}(i) < f^{-1}(i+1)$ so that $\text{Inv}(s_i f) > \text{Inv}(f)$, hence since f is simple $\sigma_{i_0} \text{br}(f) = \text{br}(s_{i_0} f)$ is simple also. So long as $s_{i_0} f \neq \omega_n$, we can find another s_{i_1} such that $\text{Inv}(s_{i_1} s_{i_0} f) > \text{Inv}(s_{i_0} f)$ and $\sigma_{i_1} \sigma_{i_0} \text{br}(f) = \text{br}(s_{i_1} s_{i_0} f)$, until we can do so no more, in which case

$$\sigma_{i_t} \cdots \sigma_{i_0} \text{br}(f) = \text{br}(s_{i_t} \cdots s_{i_0} f) = \text{br}(\omega_n) = \Delta_n.$$

Hence $\text{br}(f)$ right divides Δ_n , and a symmetric argument shows that $\text{br}(f)$ left divides Δ_n .

CHAPTER 4. ISOTOPY PROBLEM

Conversely, since Δ_n is a permutation braid, if $b_1 b_2 \equiv \Delta_n$ then both b_1 and b_2 are simple by Lemma 4.17. \square

We are finally at the point where we can show that the head of a braid is unique - and hence prove that the greedy normal form is unique.

Lemma 4.23. *The head of any positive braid b is the unique maximal simple braid which divides b . That is, if b' is simple and $b' \preceq b$, then $b' \preceq H(b)$.*

Proof. Since $H(b) = \gcd(\Delta_n, b)$, $H(b) \preceq \Delta_n$ and hence $H(b)$ is simple. Suppose that there were another braid b' such that b' was a divisor of Δ_n and $H(b) \preceq b'$. Then $H(b) \equiv b'$ by definition of gcd. Furthermore, $H(b) \equiv b'$ means that $H(b) = b'$ since they are both simple and simple braids are defined by their permutation, which satisfies the uniqueness property. \square

The greedy normal form is called such because we continue to “divide off” the largest divisor of Δ_n we can. We are finally able to return to our unfinished proof.

Proof of 4.12, continued. We have already shown that given a positive braid b , one can repeatedly find a sequence of positive braids (b_1, \dots, b_k) such that $b_i = \gcd(b_i \cdots b_k)$ and $b \equiv b_1 \cdots b_k$. Uniqueness follows from Lemma 4.23. \square

An important fact to remember is that the equivalence class for a permutation braid can be large. A permutation braid has a distinguished element and is uniquely defined by its mapping under π .

Computing the greedy normal form.

Just because the definition is greedy does not mean that our algorithm for computing it has to be greedy in the same sense. We will not simply take a braid as input, and then compute the head of the braid and so on. We will take the more streamlined approach that Dehornoy presents in [24, 26], and one which is more flexible when computing products. It

CHAPTER 4. ISOTOPY PROBLEM

turns out that normal sequences enjoy a number of useful properties which make it easier to compute the greedy normal form.

Recall the extended complement function C^* . We will use C^* in the following lemmas, and will simplify the notation by writing it in the style of a binary operation. So instead of writing $C^*(a, b) = c$ we write $a \setminus b = c$.

Recall that C^* already solved the braid isotopy problem, so one might ask why we are using it to solve the braid isotopy problem again? There are two reasons. First, we will only be using C^* on simple braids, and the final algorithm does not actually need to use C^* (although we will opt to do so for efficiency sake). Second, the ability to have a normal form of a braid is different than a truth value regarding equivalence. We will use a normal form in Chapter 5 to send braids over a network to scramble their factors. Having a distinguished member for the equivalence class of braids has a multitude of virtues.

Recall that the complement function is easy to compute and has the property that if $a \setminus b = c$ then $ac \equiv \text{lcm}(a, b)$. This is why it is called the complement function - it is the complement of the lcm. The following facts will prove to be very useful in the upcoming lemmas.

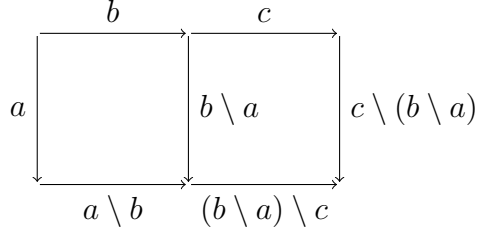
Lemma 4.24. *The following facts hold for all positive braids a, b and c :*

1. $a \preceq b$ if and only if $b \setminus a = \epsilon$,
2. $a \setminus (bc) = (a \setminus b)((b \setminus a) \setminus c)$ and $(bc) \setminus a = c \setminus (b \setminus a)$,
3. and $a \preceq bc$ if and only if $b \setminus a \preceq c$.

Proof. We will prove each of the facts separately.

1. Notice that if $a \preceq b$ then $\text{lcm}(a, b) \equiv b$. It follows that $b \setminus a = \epsilon$. A reverse argument holds for the converse.
2. The first statement follows immediately from the following reversing diagram.

CHAPTER 4. ISOTOPY PROBLEM



The bottom row of the reversing diagram is of course equal to $a \setminus (bc)$. A similar argument holds for $(bc) \setminus a = c \setminus (b \setminus a)$.

3. This proof is straightforward after one notices that $a \preceq bc$ if and only if $\text{lcm}(a, b) \preceq bc$. Then,

$$\begin{aligned}
 a \preceq bc &\Leftrightarrow \text{lcm}(a, b) \preceq bc \\
 &\Leftrightarrow b(b \setminus a) \preceq bc \\
 &\Leftrightarrow b \setminus a \preceq c.
 \end{aligned}$$

The last line is valid because of the cancellative property of B_n^+ . □

Lemma 4.25 (Local characterization). *Let (b_1, \dots, b_k) be a sequence of simple braids. The sequence is normal if and only if for each $1 \leq i \leq k - 1$, the sequence (x_i, x_{i+1}) is normal.*

Proof. First notice that (x, y) is a normal sequence if and only if for all simple braids $z \in B_n^+$, $z \preceq xy$ implies that $z \preceq x$. Now, by definition of the head of a braid, we have that for any sequence of simple braids (b_1, \dots, b_k) , $b_i \preceq H(b_i b_{i+1}) \preceq H(b_i b_{i+1} \cdots b_k)$ for $1 \leq i \leq k - 1$.

Assume that (b_1, \dots, b_k) is a normal sequence of simple braids. It follows that since $b_i = H(b_i \cdots b_k)$, we get that $b_i = H(b_i b_{i+1})$.

Conversely, suppose that for all $1 \leq i \leq k - 1$, the pair (b_i, b_{i+1}) is a normal sequence. We will show by induction on k that $b_1 = H(b_1 \cdots b_k)$. Clearly if (b_1, b_2) is normal then $b_1 = H(b_1 b_2)$ so the base case is taken care of already.

CHAPTER 4. ISOTOPY PROBLEM

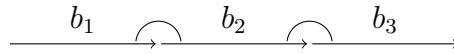
Now suppose for all $1 < i < k$ that $b_i = H(b_i \cdots b_k)$. We will show that $b_1 = H(b_1 \cdots b_k)$. Clearly $b_1 \preceq b_1 \cdots b_k$. Suppose that there exists a simple braid a such that $a \preceq b_1 \cdots b_k$, we want to show that $a \preceq b_1$ so that $b_1 = \gcd(\Delta_n, b_1 \cdots b_k)$.

From Lemma 4.24, we get $a \preceq b_1 \cdots b_k$ if and only if $b_1 \cdots b_k \setminus a = \epsilon$. Now we can use Lemma 4.24 again to obtain the expression

$$\begin{aligned} b_2 \cdots b_k \setminus (b_1 \setminus a) &= (b_1 \setminus b_1 b_2 \cdots b_k) \setminus (b_1 \setminus a) \\ &= b_1 b_2 \cdots b_k \setminus a \\ &= \epsilon, \end{aligned}$$

which is the case if and only if $b_1 \setminus a \preceq b_2 \cdots b_k$. Since (b_2, \dots, b_k) is normal by the induction hypothesis, and $b_1 \setminus a$ is a simple braid, $b_1 \setminus a \preceq b_2$. This happens if and only if $a \preceq b_1 b_2$, and since (b_1, b_2) is normal, $a \preceq b_1$. \square

We can now think of a normal sequence of braids as a chain of normal pairs. We can depict this in a diagram by representing each braid b_i by an arrow, and drawing a link between the arrows of b_i and b_{i+1} if and only if (b_i, b_{i+1}) is normal. A normal sequence can therefore be thought of as a diagram with a link between each arrow. This will provide a useful visualization for upcoming proofs.



This lemma is the first of a few to touch on the fact that the greedy normal form can be decomposed into smaller steps. When it comes to computing it, we will find that the correct decomposition will make computation much easier.

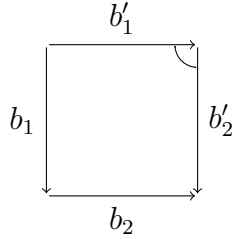
Lemma 4.26. *Let b_1 and b_2 be simple braids. Then there exist simple braids b'_1 and b'_2 such that $b'_1 b'_2 \equiv b_1 b_2$ and (b'_1, b'_2) is normal.*

Proof. The existence of b'_1 is clear, just let $b'_1 = H(b_1 b_2)$. All that remains to show is that $b'_2 = b'_1 \setminus b_1 b_2$ is simple as well. Since $b_1 \preceq b'_1$, write $b_1 a \equiv b'_1$. Now we have that $b_1 a b'_2 \equiv b_1 b_2$.

CHAPTER 4. ISOTOPY PROBLEM

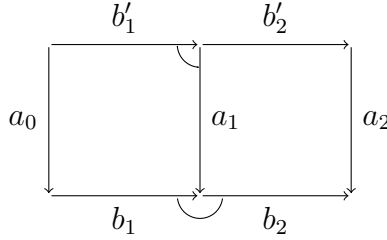
Since B_n^+ is cancellative, we have $ab'_2 \equiv b_2$, and since b_2 is simple, b'_2 must be simple as well by Lemma 4.17. \square

By drawing arrows vertically, we can depict the above lemma as the following commutative diagram.



Dehornoy describes the following move as a domino move. We will see that if we have a normal sequence, and add another simple braid to the left, the above move will make a new normal sequence.

Lemma 4.27. *Suppose that the following diagram of simple braids commutes and that (b_1, b_2) and (b'_1, a_1) are normal.*



Then (b'_1, b'_2) is normal as well.

In this domino move a \frown can be drawn between the arrows of b'_1 and b'_2 .

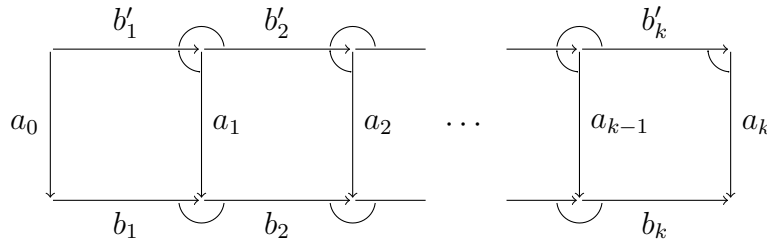
Proof. All we need to show is that $b'_1 = H(b'_1 b'_2)$. So suppose that a simple braid c divides $b'_1 b'_2$, we will see that $c \preccurlyeq b'_1$. Trivially, $c \preccurlyeq b'_1 b'_2 a_2$, so by the commutativity of the diagram, $c \preccurlyeq a_0 b_1 b_2$. Using the tools of Lemma 4.24, we get $a_0 \setminus c \preccurlyeq b_1 b_2$. Since a_0 and c are both simple, $a_0 \setminus c$ is simple as well. So since $a_0 \setminus c \preccurlyeq b_1 b_2$ and (b_1, b_2) is normal, $a_0 \setminus c \preccurlyeq b_1$.

CHAPTER 4. ISOTOPY PROBLEM

Using the same tools again we get that $c \preceq a_0 b_1$ and therefore also $b'_1 a_1$. Since by the hypothesis (b'_1, a_1) is normal, $c \preceq b'_1$. \square

The domino move is the key to computing the greedy normal form. Because normal sequences have this property, we can compute the greedy normal form of the product of simple braids very efficiently. Braids already decompose into simple braids: σ_i is simple.

Theorem 4.28. *Let $b = b_1 \cdots b_k \in B_n^+$ with (b_1, \dots, b_k) normal and, $a \in B_n^+$ simple. Then there exists a normal sequence $(b'_1, b'_2, \dots, b'_k, a_k)$ such that $ab \equiv b'_1 \cdots b'_k a_k$ where $a = a_0, a_1, \dots, a_k$ are the simple braids as per the commutative diagram below.*



The idea here is to start with an L-shaped diagram of $a_0 b_1 \cdots b_k$. Then fill in the corner to obtain a_1 and b'_1 , and so on. We repeat until we are left with $b'_1 b'_2 \cdots b'_k a_k$, with the possibility that $a_k = \epsilon$. There is nothing to prove - (b'_1, \dots, b'_k, a_k) is normal by the previous lemmas. We finally have the machinery to construct an algorithm. One may object that we

```

input : positive braid word  $b$  of length  $\ell$ 
set  $S = (b[\ell])$ 
for  $i = \ell - 1, \ell - 2, \dots, 1$  do
|   perform domino moves on  $b[i] \cup S$  to obtain  $S'$ 
|   set  $S = S'$ 
end
return  $S$ 

```

Algorithm 4.29: Computing the greedy normal form.

do not have an explicit algorithm for computing individual domino moves, this is because

CHAPTER 4. ISOTOPY PROBLEM

we do not have algorithms for computing gcds. When the index of the braid group n is low there are a small enough number of simple braids ($(n!)^2$ of them) that computing the head of each product of two simple braids can be stored in a small amount of memory.

Epstein et al. [31] provide an algorithm for computing gcds of permutation braids by using the properties of a lattice. They describe a finite state automaton which uses a sort and merge algorithm to find the right candidate. They are able to do this in $\mathcal{O}(n \log n)$ time.

Theorem 4.30. *For a fixed n , Algorithm 4.29 computes in $\mathcal{O}(\ell^2)$ time. For variable n , the time is $\mathcal{O}(\ell^2 n \log n)$.*

Proof. Since there are a total of $(n!)^2$ pairs of simple words in B_n^+ , when n is fixed computing each individual domino move takes constant time. A full sequence of domino moves at each step can be no longer than ℓ . We do ℓ sequences of domino moves, so we get a total of $\mathcal{O}(\ell^2)$. When n is variable, we use the merge and sort algorithm of [31] to compute the domino moves. See [57] for an implementation. \square

4.5 The left-normal form for the braid group.

In Chapter 3 we used Lemma 3.9 to show that for every positive braid $b \in B_n^+$, there exists another braid $b' \in B_n^+$ such that $bb' \equiv \Delta_n^\ell$. We can use the same lemma to obtain a similar result.

Proposition 4.31. *For every braid $b \in B_n$, there exists an integer $p \geq 0$ and positive braid b' such that $b \equiv \Delta_n^{-p} b'$.*

Proof. For each i , let u_i be the positive braid from Lemma 3.9 such that $\sigma_i u_i \equiv \Delta_n$. Now, let $w_i = u_i \Delta_n$. Even though w_i is positive, $w_i \equiv \sigma_i^{-1} \Delta_n^2$.

Let k be the number of inverse Artin generators σ_i^{-1} in b . Let b' be the positive braid obtained by replacing each σ_i^{-1} with w_i . Then since Δ_n^2 commutes with all σ_i , it follows that $\Delta_n^{2k} b \equiv b'$. Hence $b \equiv \Delta_n^{-2k} b'$ as desired. \square

CHAPTER 4. ISOTOPY PROBLEM

Now we can obtain the greedy normal form for b' so that $b \equiv \Delta_n^{2k} b_1, \dots, b_t$ with (b_1, \dots, b_t) a normal sequence. The left-normal form for the braid group is almost this.

Definition 4.32 (left-normal form). Let $b \in B_n$. We say that the braid $b' = \Delta_n^{-k} b_1 \cdots b_t$ is the left-normal form of b if $b \equiv b'$, k is minimal and (b_1, \dots, b_t) is normal. The integer t is called the complexity of b .

The minimality of k is the only extra step we might have to take, since 4.31 showed us how to put the braid in an “almost” left-normal form. The idea is that if $b_1 = \Delta_n$, then remove b_1 from the normal sequence, and reduce k by one, and try again. This ensures that the left-normal form is compatible with the greedy normal form.

Chapter 5

Braid group cryptography

Cryptography is the mathematics behind sending secret messages. Given recent events, the importance of cryptography in the average citizen's life hardly needs an explanation. The role of cryptography is becoming more and more clear to the public as questions regarding privacy continue to break news stories.

There are many types of cryptosystems. A commonly used one is public key encryption. In this scenario, there is a sender, Alice, and a receiver, Bob, who would like to communicate privately through a public channel. Bob has a pair of keys, one of which is private (K_{priv}) and he doesn't share with anyone, and the other is public (K_{pub}), which anyone who would like to send him a message (or any adversaries for that matter) can see. When Alice sends Bob a message, she does so by encrypting it with the public key, which can only be decrypted with Bob's private key. She sends the encrypted message through the public channel for Bob to decrypt.

Another method of exchanging secret messages is called private key encryption, in which both Alice and Bob share the same private key that nobody else is aware of. There is no public key. In this encryption scheme, Alice and Bob agree on a shared private key which is able to both encrypt and decrypt a message. When they have done this they are free to send private messages to one another through a public forum, e.g. The Internet. The trouble with private key encryption schemes is the method in which the two parties agree

CHAPTER 5. BRAID GROUP CRYPTOGRAPHY

on a private key - the *key exchange*. The canonical protocol is called *Diffie-Hellman* key exchange.

Example 5.1. The Diffie-Hellman key exchange protocol is a widely used protocol with many variations. It is based on the hardness of the discrete logarithm problem: given $g^x \pmod{p}$ for a prime p and $x, g \in \mathbb{Z}_p$, compute x . Knowing how to solve the discrete logarithm problem certainly breaks the cryptosystem.

Public Key : Alice and Bob agree on a prime p , and $g \in \mathbb{Z}_p$
Alice Sends : chooses $x \in \mathbb{Z}_p$, sends g^x
Bob Sends : chooses $y \in \mathbb{Z}_p$, sends g^y
Shared Key : $(g^y)^x = (g^x)^y$

Cryptosystem 5.2: Diffie-Hellman key exchange protocol.

5.1 Security definitions.

The basis to any cryptosystem is a computationally difficult mathematical problem. Examples of these are factoring numbers of the form $n = pq$, where p and q are large prime numbers, or the discrete logarithm problem: given an equation $g^x = h$ in a group where g is a generator, determine x .

The notion of a secure cryptosystem has many different forms. For example, a public key cryptosystem may be able to prevent an adversary from recovering K_{priv} , but at the same time the same adversary could somehow decrypt messages without the key. This is why we have distinct definitions of security.

Definitions of security have two components - the goal of an eavesdropper (who we appropriately name Eve), and her capabilities. We always assume that Eve has access to a universal Turing machine, and that she is pressed for time. That is, the only type of algorithms the adversary can use are those which terminate in polynomial time with respect

CHAPTER 5. BRAID GROUP CRYPTOGRAPHY

to the length of the message or key. We go about proving these security definitions by showing that if Eve can reach her goals with the given capabilities of the security definition, then the method in which she does this (i.e. a polynomial time algorithm) can be used as a black box to solve a mathematical problem for which there is no known polynomial time solution. We thus show that the cryptosystem is at least as hard to break as a problem which is known to be difficult.

Note that this does not prove that the cryptosystem will never be able to be broken. Cryptosystems are often so closely based on well known problems that a solution to the underlying problem immediately breaks the cryptosystem. Sometimes there are flaws in the protocol or implementation which can also allow an adversary to break the cryptosystem without even coming close to solving the underlying problem.

Definition 5.3 (KP). A cryptosystem is *key-private* when the adversary is unable to recover the private keys used.

We similarly define the capabilities an adversary might have. These vary by the type of cryptosystem in question, but a typical one is a chosen plaintext attack.

Definition 5.4 (CPA). We say that an adversary performs *chosen plaintext attack* on a public-key cryptosystem when they have the ability to encrypt any message of their choosing.

A cryptosystem meets a security definition when we outline both the goal and the capability of the adversary, and almost always involves a computationally difficult problem. Sometimes these problems can be contrived. The Diffie-Hellman protocol for example is certainly broken by the discrete logarithm problem, but instead we usually associate the hardness of the Diffie-Hellman protocol with a simpler problem: *The Diffie-Hellman problem*: given g , g^x , and g^y , compute g^{xy} . This problem is in fact equivalent to Diffie-Hellman being key-private.

5.2 Conjugacy problems.

Braids offer a new type of problem that can be used in a cryptosystem: given braids b_1 and b_2 , is there a third braid c such that $b_1 \equiv cb_2c^{-1}$? When the question is answered positively, b_1 and b_2 are said to be conjugated by c , their conjugator. We can state this in a more formal way.

Problem 5.5 (conjugacy decision problem). Given two braids b_1 and b_2 , determine whether or not there exists a braid c such that $b_1 \equiv cb_2c^{-1}$: are b_1 and b_2 conjugate?

The first appearance of the conjugacy decision problem was in Artin's paper [8]. Artin gives an algorithm for solving the braid isotopy problem, as outlined in algorithm 4.5. He introduces a new type of problem that, at the time, had not been solved. He asks the reader to imagine a braid being wound around an axis so the ends meet to form a "closed braid". Given two braids b_1 and b_2 , their closures are isotopic if and only if $b_1 \equiv cb_2c^{-1}$, for some other braid c . He also notices that a solution to this problem could be applied to the problem of identifying knots and links.

Since then, there have been a few variations of the conjugacy decision problem, which are used as underlying problems for different cryptosystems.

Problem 5.6 (conjugacy search problem). Given two conjugate braids b_1 and b_2 , find a conjugating braid c , i.e. find a braid c such that $b_1 \equiv c^{-1}b_2c$.

Problem 5.7 (multiple simultaneous conjugacy search problem). Take m pairs of elements $(b_1, k_1), \dots, (b_m, k_m)$ in B_n^2 in which each pair is conjugated by the same braid. Find the conjugator $c \in B_n$ such that $b_i \equiv c^{-1}k_i c$ for all $i = 1, \dots, m$.

5.3 Cryptosystems based on braid groups.

Braids can be represented in a computer in many different ways. The most straightforward way is to define a finite list of nonzero integers. Each integer i represents σ_i when $i > 0$

CHAPTER 5. BRAID GROUP CRYPTOGRAPHY

and σ_{-i}^{-1} when $i < 0$. The existence of the left-normal form tells us that we can also store braids as a finite list starting with an integer, followed by elements of the symmetric group.

We must be careful when we are relying on the conjugacy problem for braids. For example, if the public braid is $x = \sigma_2\sigma_4\sigma_3$, and we are afraid that an eavesdropper obtains the conjugator $c = \sigma_1\sigma_2\sigma_3$, then we should not send over a public channel $cxc^{-1} = \sigma_1\sigma_2\sigma_3\sigma_2\sigma_4\sigma_3\sigma_3^{-1}\sigma_2^{-1}\sigma_1^{-1}$. It takes little effort to see that the conjugator is $\sigma_1\sigma_2\sigma_3$. Instead we should send the braid in some kind of normal form. For instance, if we sent the left-normal form of cxc^{-1} :

$$y = \sigma_1^{-1}\sigma_2^{-1}\sigma_3^{-1}\sigma_4^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_3^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1}\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_4\sigma_3\sigma_2\sigma_1\sigma_1\sigma_2\sigma_3\sigma_4$$

along with x , it is much more difficult to recover c . Of course, $y \equiv cxc^{-1}$. From this point forward, unless otherwise stated, every braid sent over a public channel is done so in its normal form.

Anshel-Anshel-Fisher-Goldfeld key-exchange.

The Anshel-Anshel-Goldfeld cryptosystem is a theoretical cryptosystem devised in 1999 in [3] by the cryptologists the cryptosystem is named after. In 2001, Fisher [4] joined the original authors to implement the theoretical cryptosystem into one based on the braid group.

<p>Public Key : set of braids $\{k_1, k_2, \dots, k_m\} \subset B_n$ Private Keys: Alice: $a \in \langle k_1, \dots, k_m \rangle$, Bob: $b \in \langle k_1, \dots, k_m \rangle$</p> <p>Bob Sends : $(bk_1b^{-1}, \dots, bk_mb^{-1})$ Alice Sends : $(ak_1a^{-1}, \dots, ak_ma^{-1})$</p> <p>Shared Key : $aba^{-1}b^{-1}$</p>

Cryptosystem 5.8: Anshel-Anshel-Fisher-Goldfeld key-exchange

CHAPTER 5. BRAID GROUP CRYPTOGRAPHY

At this point, one should notice that using the normal form of a braid is important. If an adversary knows k_i , and Bob sends bk_ib^{-1} , then it is not at all difficult to discover b .

Proposition 5.9. *Alice and Bob from the Anshel-Anshel-Fisher-Goldfeld key-exchange can both obtain $aba^{-1}b^{-1}$ efficiently.*

Proof. Alice knows a and receives $(bk_1b^{-1}, \dots, bk_mb^{-1})$ from Bob. Since $a \in \langle k_1, \dots, k_m \rangle$, she knows that $a = x_1 \dots x_t$ where $x_i \in \{k_1, \dots, k_m\}$. Therefore she knows that

$$ba^{-1}b \equiv bx_1b^{-1} \dots bx_tb^{-1}$$

and that

$$a(bx_1b^{-1} \dots bx_tb^{-1}) = aba^{-1}b^{-1}.$$

Since computing the left-normal form is easy, calculating $aba^{-1}b^{-1}$ is as well. A symmetric argument holds showing that Bob can compute the shared key efficiently as well. \square

Proposition 5.10. *Obtaining the private keys a and b in the Anshel-Anshel-Fisher-Goldfeld key-exchange relies on the hardness of multiple simultaneous conjugacy search problem.*

Proof. The proof of this is trivial - the cryptosystem is designed to imitate the multiple simultaneous conjugacy search problem. That is, a is the conjugator of $ak_1a^{-1}, \dots, ak_ma^{-1}$ and b is the conjugator of $bk_1b^{-1}, \dots, bk_mb^{-1}$. There is nothing else to show. \square

Ko et al.'s key-exchange.

The braid group is noncommutative in general, but some elements do commute with one another. Furthermore, there are large subgroups of B_n which commute with other subgroups - precisely subgroups that do not share any strands. Using this fact, Ko et al. [44] construct a new key-exchange protocol based on the celebrated Diffie-Helman key-exchange protocol. The scheme uses a one-way function f that has the property that images are easily computable but preimages are not.

CHAPTER 5. BRAID GROUP CRYPTOGRAPHY

Let n be a positive integer greater than 4, and consider the two subgroups of B_n : LB_n , the subgroup generated by $\{\sigma_1, \dots, \sigma_{\lfloor (n-1)/2 \rfloor}\}$ and RB_n , the subgroup generated by $\{\sigma_{\lceil (n-1)/2 \rceil}, \dots, \sigma_{n-1}\}$. That is to say that LB_n is the set of braids on n strands which only weaves the left half of the strands, and RB_n only weaves the right half strands. This has the nice property that the elements in LB_n commute with the elements in RB_n . The one-way function that Ko et al. propose is

$$f : LB_n \times B_n \rightarrow B_n \times B_n$$

where

$$f(b, c) = (bcb^{-1}, c).$$

A similar function exists with domain $RB_n \times B_n$. This is a one-way function because computing bcb^{-1} is straightforward, but given bcb^{-1} and c , computing b involves solving the conjugacy search problem.

Public Key : $x \in B_n$
Private Keys: Alice: $a \in LB_n$, Bob: $b \in RB_n$

Alice Sends : axa^{-1}
Bob Sends : $bx b^{-1}$

Shared Key : $abxb^{-1}a^{-1} \equiv baxa^{-1}b^{-1}$

Cryptosystem 5.11: Ko et al.'s key-exchange protocol.

Just like Diffie-Hellman, this cryptosystem introduces a new problem which is hard, so that a proof of security can be given.

Problem 5.12 (Diffie-Hellman like conjugacy search problem). Given a braid $x \in B_n$ and axa^{-1} and $bx b^{-1}$ with $a \in LB_n$ and $b \in RB_n$ (but any other information about a and b is unknown), find the braid $abx(ab)^{-1}$ or $bax(ba)^{-1}$.

Ko et al.'s key-exchange protocol being KP is immediately equivalent to the Diffie-Hellman like conjugacy search problem. This means that an efficient solution to the

conjugacy problem would break the cryptosystem.

Ko et al.'s public key encryption scheme.

Ko et al. [44] introduce an encryption scheme that uses a hash function H . This is a type of function which is one-way, and collision resistant: when $a \neq b$, the probability of $H(a) = H(b)$ is negligible. Furthermore, it is computationally infeasible to find elements a and b with $H(a) = H(b)$. Although they don't offer any hash functions to use, Dehornoy [22] recommends using the Dynnikov formulas which are based on the n -punctured disc (see Chapter 5 of [26]). Collision resistance follows from the fact that it is an isotopy invariant (the probability of a collision is null, not just negligible). There are no known algorithms for computing the preimage of the Dynnikov formula of a braid.

The Dynnikov coordinates of a braid is an ordered $2n$ -tuple of integers. Since messages are typically encoded as binary strings, it will be useful if our hash function maps to $\{0, 1\}^*$ as well. This way, we can perform the binary XOR (exclusive-or) involution $\oplus : \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}^k$ defined by

$$(a_i)_{i=1}^k \oplus (b_i)_{i=1}^k = (a_i + b_i \pmod{2})_{i=1}^k.$$

Key Generation	: Bob chooses random braids $x \in B_{\ell+r}$, and $a \in LB_n$
Public Key	: (x, y) , where $y = axa^{-1}$
Private Keys	: a
Encryption	: Alice chooses $b \in RB_n$, sends $(c, d) = (bxb^{-1}, H(byb^{-1}) \oplus m)$
Decryption	: Bob computes $m = H(aca^{-1}) \oplus d$

Cryptosystem 5.13: Ko et al.'s encryption protocol.

CHAPTER 5. BRAID GROUP CRYPTOGRAPHY

When Bob computes $H(aca^{-1}) \oplus d$, he is indeed calculating m since

$$\begin{aligned} H(aca^{-1}) \oplus d &= H(abxb^{-1}a^{-1}) \oplus H(byb^{-1}) \oplus m \\ &= H(abxb^{-1}a^{-1}) \oplus H(baxa^{-1}b^{-1}) \oplus m \\ &= m \end{aligned}$$

since $a \in LB_n$ and $b \in RB_n$ implies $ab = ba$ and hence $H(abxb^{-1}a^{-1}) = H(baxa^{-1}b^{-1})$.

Just like their authentication scheme, Ko et al.'s encryption scheme relies on the Diffie-Hellman like conjugacy search problem for its security.

Sibert, Dehornoy, and Girault's authentication scheme.

A different kind of cryptosystem is an authentication scheme, where somebody can ask somebody else to prove their identity. This is usually in the form of a challenge and response - where the party requesting proof of identity poses a challenge in which only the person in question can answer, e.g. requesting a password.

The following authentication scheme, introduced by Sibert, Dehornoy, and Girault [56] exploits a computationally difficult problem which seemingly has nothing to do with the conjugacy problem.

Problem 5.14 (root extraction). Given an exponent $e \geq 2$ and a braid b^e (in normal form), compute b .

Key Generation: Alice chooses $a \in B_n$ and computes $b = s^2$
Public Key : b
Private Keys : a

Authentication:
for $i = 1$ **to** k **do**
 Alice chooses $r \in B_n$, sends $x = rbr^{-1}$
 Bob sends $\varepsilon \in \{0, 1\}$
 if $\varepsilon = 0$ **then**
 Alice sends $y = r$, Bob checks $x = yby^{-1}$
 else if $\varepsilon = 1$ **then**
 Alice sends $y = rsr^{-1}$, Bob checks $x = y^2$
 end
end

Cryptosystem 5.15: Sibert, Dehornoy, and Girault’s authentication scheme.

The idea of the authentication scheme is for Bob to verify Alice’s identity by randomly choosing 0 or 1 and requesting the answer to a problem only effectively solvable when the secret key is known.

It should be clear that an impostor (Eve) could reproduce the case when $\varepsilon = 0$ and $\varepsilon = 1$ if she knew the outcome of ε beforehand. If Eve knew ahead of time that $\varepsilon = 0$ then she can simply choose any $r \in B_n$ and send $x = rbr^{-1}$ as the scheme would. On the other hand, if Eve knew that $\varepsilon = 1$ ahead of time, she could prepare by first choosing $y \in B_n$, and then sending $x = y^2$. The problem is that if she guesses incorrectly the outcome of ε then she cannot correctly verify Bob’s question. This is why the scheme waits for Alice to send x before Bob chooses ε , and that the process is repeated k times.

5.4 Attacks on braid group cryptography.

Garside [34] showed that the conjugacy problem is solvable by introducing a finite subset of the infinite conjugacy class of braids, called *summit sets*. The basic idea is as follows.

CHAPTER 5. BRAID GROUP CRYPTOGRAPHY

For a braid $b \in B_n$, let $C(b)$ be the finite conjugacy class of b , that is, $C(b) = \{cbc^{-1} : c \in B_n\}$. Garside showed that there exists a set called $SS(b) \subset C(b)$ which he calls the summit set of b , which has the following properties:

1. The summit set of b relies only on $C(b)$: b is conjugate to c if and only if $SS(b) = SS(c)$. There is no guarantee that $b \in SS(b)$.
2. For each $b \in B_n$, there is an efficient algorithm which computes a unique representative $\tilde{b} \in SS(b)$. (This is related but not equal to the left-normal form of b)
3. There is an algorithm (not necessarily efficient) which can construct $SS(b)$ from the representative \tilde{b} .

```
Input : braids  $b$  and  $c$ 
Compute  $\tilde{b}$  and  $\tilde{c}$ 
while constructing  $SS(b)$  from  $\tilde{b}$  do
  | if  $\tilde{c} \in SS(b)$  then
  | | return  $b$  and  $c$  are conjugate
  | end
end
return  $b$  and  $c$  are not conjugate
```

Algorithm 5.16: Garside's solution to the conjugacy problem.

Garside's algorithm is inefficient, which means that it poses no threat to the cryptosystems which rely on the hardness of the conjugacy decision problem. Since Garside however, there have been a number of improvements which do pose a threat to those cryptosystems. El-Rifai and Morton [30] construct a subset of $SS(b)$, which they call the *super summit set* of b , or $SSS(b)$. The super summit set is calculated by a series of special conjugations called cyclings and decyclings which are easy to compute.

The super summit sets are still exponential in size with respect to n [33] and computing them requires a factor of $n!$, but in doing so gives an extra solution. The method they use to construct the super summit set actually allows one to find the conjugator when the two

CHAPTER 5. BRAID GROUP CRYPTOGRAPHY

braids are conjugate. Hence the conjugacy search problem and multiple conjugacy search problem are solvable.

In 2003, Franco and González-Meneses [32] improved on super summit sets by constructing a subset $USS(b) \subseteq SSS(b)$. The complexity of their algorithm (the size of $USS(b)$) is unknown, but by using probabilistic methods, Birman, Gebhart and González-Meneses were able to show that the algorithm is efficient in practice. See [11, 12, 13] for details on this.

Also in 2003, Lee and Park [47] were able to efficiently solve an instance of the Diffie-Hellman like conjugacy search problem under certain parameters. Then they showed that those parameters are likely to be met while using Ko et al.'s encryption protocol. Still in 2003, Cheon and Jun [17] showed how to solve the Diffie-Hellman like conjugacy search problem in polynomial time with respect to the length ℓ of the key and the braid index n .

There still has been no efficient solution to the root extraction problem on braids, although Styšnev in [58] proves that the problem is decidable in B_n , and Sibert [55] does so for the general case of Garside groups. Furthermore, Groch, Hofheinz and Steinwandt [40] provide a heuristic algorithm which does not solve the root problem, but attacks the Sibert, Dehornoy, and Girault authentication scheme directly.

The hope for braid group cryptography is not lost. In 1991 Paterson [51] showed that there is an NP-complete problem involving braids, the *Minimal Length Problem*: given a braid $b \in B_n$, find a braid $b' \equiv b$ such that $|b'|$ is minimal. No cryptosystems have come of this problem yet. Beyond this, Dehornoy [28] suggests considering operations in the braid group other than the group product. Garber [33] suggests a number of other difficult problems for the braid group as well.

APPENDICES

Appendix A

Computer programs for using braids.

The computer program which was used to compute some of the examples in this thesis can be found at <https://www.dropbox.com/s/xtbifemvixkg976/braid-progs.py> or goo.gl/k0ejeU for short. Sage has some built-in ability to perform actions on braids as well - the documentation found here <http://doc.sagemath.org/html/en/reference/groups/sage/groups/braid.html> is particularly useful.

For this thesis, braids were typeset using the `tangles.sty` package, but in more complicated diagrams (figure 2.28 for example) `TikZ` was used. For three dimensional renderings of braids, `KnotPlot` is very robust. Visit <http://www.knotplot.com> to download a free trial and consult the manual.

A.1 Manual for `braid-progs.py`.

This program can be run in either Python or Sage. To open in a system with only Python installed, one types into a console `python -i /path/to/braid-progs.py`. If Sage is installed on a machine, one can copy and paste the plain text into a compute cell in Sage Notebook and press evaluate. In a Sage terminal session, enter `&attach /path/to/braid-progs.py`. Braids are entered as a list of positive or negative integers.

APPENDIX A. COMPUTER PROGRAMS FOR USING BRAIDS.

Each nonzero integer i represents σ_i or σ_{-i}^{-1} when $i < 0$. For example, to encode $b = \sigma_1\sigma_2^{-1}\sigma_3$, one writes

$$\mathbf{b} = [1, -2, 3]$$

and we are free to perform the following actions on \mathbf{b} , including the braid product, which is simply concatenation since these are strings. In Python, to concatenate strings, the addition symbol $+$ works.

inverse(b)

Computes the inverse of a braid b .

Dynnikov(b,n)

Computes the Dynnikov coordinates of b in B_n as per Chapter 5 of [26]. Note that n is required.

braidRecDynnikov(b1,b2,n)

An efficient algorithm which uses the Dynnikov formulas to solve the word problem. Returns 1 if $b_1 \equiv b_2$ in B_n and 0 otherwise. Again n is required.

complement(s1,s2)

Computes the complement of σ_1 and σ_2 .

subwordReversing(b)

Given a braid b , it will compute the south-east path of the reversing diagram of b as per Section 4.3.

extendedComplement(u,v)

Calculates $C^*(u,v)$ as per 3.18.

braidLCM(b1,b2)

Uses subword reversing to compute $\text{lcm}(b_1, b_2)$.

proveTheorem()

Proves 3.29.

References

- [1] M. Albenque. “Bijective combinatorics of positive braids”. In: *Electronic Notes in Discrete Mathematics* 29.0 (2007), pp. 225–229.
- [2] M. Albenque and P. Nadeau. “Growth function for a class of monoids”. In: *21st International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2009)*. Discrete Math. Theor. Comput. Sci. Proc., AK. Assoc. Discrete Math. Theor. Comput. Sci., Nancy, 2009, pp. 25–38.
- [3] I. Anshel, M. Anshel, and D. Goldfeld. “An algebraic method for public-key cryptography”. In: *Mathematical Research Letters* 6 (1999), pp. 287–292.
- [4] I. Anshel et al. “New key agreement protocols in braid group cryptography”. In: *Topics in cryptology—CT-RSA 2001 (San Francisco, CA)*. Vol. 2020. Lecture Notes in Comput. Sci. Springer, Berlin, 2001, pp. 13–27. DOI: 10.1007/3-540-45353-9_2. URL: http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1007/3-540-45353-9_2.
- [5] E. Artin. “Braids and permutations”. In: *Ann. of Math. (2)* 48 (1947), pp. 643–649. ISSN: 0003-486X.
- [6] E. Artin. “The theory of braids”. In: *American Scientist* 38 (1950), pp. 112–119.
- [7] E. Artin. “Theorie der Zöpfe”. In: *Abh. Math. Sem. Univ. Hamburg* 4.1 (1925), pp. 47–72. ISSN: 0025-5858. DOI: 10.1007/BF02950718. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1007/BF02950718>.
- [8] E. Artin. “Theory of braids”. In: *Ann. of Math. (2)* 48 (1947), pp. 101–126. ISSN: 0003-486X.

REFERENCES

- [9] J. Birman, K. H. Ko, and S. J. Lee. “A new approach to the word and conjugacy problems in the braid groups”. In: *Adv. Math.* 139.2 (1998), pp. 322–353. ISSN: 0001-8708. DOI: 10.1006/aima.1998.1761. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1006/aima.1998.1761>.
- [10] J. S. Birman and T. E. Brendle. “Chapter 2 - Braids: A Survey”. In: ed. by W. M. Thistlethwaite. *Handbook of Knot Theory*. Amsterdam: Elsevier Science, 2005, pp. 19–103. ISBN: 9780444514523.
- [11] J. S. Birman, V. Gebhardt, and J. González-Meneses. “Conjugacy in Garside groups. I. Cyclings, powers and rigidity”. In: *Groups Geom. Dyn.* 1.3 (2007), pp. 221–279. ISSN: 1661-7207. DOI: 10.4171/GGD/12. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.4171/GGD/12>.
- [12] J. S. Birman, V. Gebhardt, and J. González-Meneses. “Conjugacy in Garside groups. II. Structure of the ultra summit set”. In: *Groups Geom. Dyn.* 2.1 (2008), pp. 13–61. ISSN: 1661-7207. DOI: 10.4171/GGD/30. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.4171/GGD/30>.
- [13] J. S. Birman, V. Gebhardt, and J. González-Meneses. “Conjugacy in Garside groups. III. Periodic braids”. In: *J. Algebra* 316.2 (2007), pp. 746–776. ISSN: 0021-8693. DOI: 10.1016/j.jalgebra.2007.02.002. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1016/j.jalgebra.2007.02.002>.
- [14] J. Birman. *Braids, Links, and Mapping Class Groups*. Annals of mathematics studies. University of Tokyo Press, 1975. ISBN: 9780691081496. URL: <https://books.google.com/books?id=thv7L4AQ3J4C>.
- [15] F. Borceux. *Handbook of Categorical Algebra: Volume 2, Categories and Structures*. Cambridge Studies in Philosophy. Cambridge University Press, 1994. ISBN: 9780521441797. URL: <https://books.google.com/books?id=5i2v9q0m5XAC>.
- [16] G. Bredon. *Topology and Geometry*. Graduate Texts in Mathematics. Springer, 1993. ISBN: 9780387979267. URL: https://books.google.com/books?id=G74V6UzL_PUC.

REFERENCES

- [17] J. H. Cheon and B. Jun. “A polynomial time algorithm for the braid Diffie-Hellman conjugacy problem”. In: *Advances in cryptology—CRYPTO 2003*. Vol. 2729. Lecture Notes in Comput. Sci. Springer, Berlin, 2003, pp. 212–225. DOI: 10.1007/978-3-540-45146-4_13. URL: http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1007/978-3-540-45146-4_13.
- [18] P. M. Cohn. *Algebra. Vol. 2*. Second. John Wiley & Sons, Ltd., Chichester, 1989, pp. xvi+428. ISBN: 0-471-92234-X.
- [19] A. Datta et al. *Key Exchange Protocols: Security Definition, Proof Method and Applications*. Cryptology ePrint Archive, Report 2006/056. <http://eprint.iacr.org/>. 2006.
- [20] P. Dehornoy et al. *Ordering Braids*. Mathematical surveys and monographs. American Mathematical Society, 2008. ISBN: 9780821844311. URL: <https://books.google.com/books?id=53krngEACAAJ>.
- [21] P. Dehornoy. “A Fast Method for Comparing Braids”. In: *Advances in Mathematics* 125.2 (1997), pp. 200–235.
- [22] P. Dehornoy. “Braid-based cryptography”. In: *Contemp. Math* 360 (2004), pp. 5–33.
- [23] P. Dehornoy. “Combinatorics of normal sequences of braids”. In: *Journal of Combinatorial Theory, Series A* 114.3 (2007), pp. 389–409.
- [24] P. Dehornoy. “Efficient solutions to the braid isotopy problem”. In: *Discrete Applied Mathematics* 156.16 (2008), pp. 3091–3112.
- [25] P. Dehornoy. “Monoids of O -type, subword reversing, and ordered groups”. In: *J. Group Theory* 17.3 (2014), pp. 465–524. ISSN: 1433-5883. DOI: 10.1515/jgt-2013-0049. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1515/jgt-2013-0049>.
- [26] P. Dehornoy. *Notes on the Braid Isotopy Problem [Lecture Notes]*. Available at: <http://www.math.unicaen.fr/~dehornoy/Surveys/Dhu.pdf>. 2010.

REFERENCES

- [27] P. Dehornoy. “The subword reversing method”. In: *Internat. J. Algebra Comput.* 21.1-2 (2011), pp. 71–118. ISSN: 0218-1967. DOI: 10.1142/S0218196711006091. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1142/S0218196711006091>.
- [28] P. Dehornoy. “Using shifted conjugacy in braid-based cryptography”. In: *Algebraic methods in cryptography*. Vol. 418. Contemp. Math. Amer. Math. Soc., Providence, RI, 2006, pp. 65–73. DOI: 10.1090/conm/418/07946. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1090/conm/418/07946>.
- [29] P. Dehornoy and L. Paris. “Gaussian groups and Garside groups, two generalisations of Artin groups”. In: *Proc. London Math. Soc. (3)* 79.3 (1999), pp. 569–604. ISSN: 0024-6115. DOI: 10.1112/S0024611599012071. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1112/S0024611599012071>.
- [30] E. A. El-Rifai and H. R. Morton. “Algorithms for positive braids”. In: *Quart. J. Math. Oxford Ser. (2)* 45.180 (1994), pp. 479–497. ISSN: 0033-5606. DOI: 10.1093/qmath/45.4.479. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1093/qmath/45.4.479>.
- [31] D. Epstein. *Word Processing in Groups*. Ak Peters Series. Taylor & Francis, 1992. ISBN: 9780867202441. URL: <https://books.google.com/books?id=DQ84Q1Tr-EgC>.
- [32] N. Franco and J. González-Meneses. “Conjugacy problem for braid groups and Garside groups”. In: *Journal of Algebra* 266.1 (2003), pp. 112–132.
- [33] D. Garber. “Braid group cryptography”. In: *Braids*. Vol. 19. Lect. Notes Ser. Inst. Math. Sci. Natl. Univ. Singap. World Sci. Publ., Hackensack, NJ, 2010, pp. 329–403. DOI: 10.1142/9789814291415_0006. URL: http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1142/9789814291415_0006.
- [34] F. A. Garside. “The braid group and other groups”. In: *Quart. J. Math. Oxford Ser. (2)* 20 (1969), pp. 235–254. ISSN: 0033-5606.
- [35] V. Gebhardt. “A new approach to the conjugacy problem in Garside groups”. In: *Journal of Algebra* 292.1 (2005), pp. 282–302.

REFERENCES

- [36] V. Gebhardt and J. González-Meneses. “Generating random braids”. In: *Journal of Combinatorial Theory, Series A* 120.1 (2013), pp. 111–128.
- [37] V. Gebhardt and S. Tawn. “Normal forms of random braids”. In: *Journal of Algebra* 408.0 (2014), pp. 115–137.
- [38] J. González-Meneses. “The n th root of a braid is unique up to conjugacy”. In: *Algebr. Geom. Topol.* 3 (2003), 1103–1118 (electronic). ISSN: 1472-2747. DOI: 10.2140/agt.2003.3.1103. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.2140/agt.2003.3.1103>.
- [39] J. González-Meneses and E. Ventura. “Twisted conjugacy in braid groups”. In: *Israel J. Math.* 201.1 (2014), pp. 455–476. ISSN: 0021-2172. DOI: 10.1007/s11856-014-0032-4. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1007/s11856-014-0032-4>.
- [40] A. Groch, D. Hofheinz, and R. Steinwandt. “A practical attack on the root problem in braid groups”. In: *Algebraic methods in cryptography*. Vol. 418. Contemp. Math. Amer. Math. Soc., Providence, RI, 2006, pp. 121–131. DOI: 10.1090/conm/418/07950. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1090/conm/418/07950>.
- [41] V. Hansen. *Braids and Coverings: Selected Topics*. London Mathematical Society Student Texts. Cambridge University Press, 1989. ISBN: 9780521387576. URL: <https://books.google.com/books?id=t5b059aVJVcC>.
- [42] C. Kassel, O. Dodane, and V. Turaev. *Braid Groups*. Graduate Texts in Mathematics. Springer, 2008. ISBN: 9780387685489. URL: <https://books.google.ca/books?id=y6Cox3XjdroC>.
- [43] K. H. Ko, J. W. Lee, and T. Thomas. “Towards generating secure keys for braid cryptography”. In: *Des. Codes Cryptogr.* 45.3 (2007), pp. 317–333. ISSN: 0925-1022. DOI: 10.1007/s10623-007-9123-0. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1007/s10623-007-9123-0>.

REFERENCES

- [44] K. H. Ko et al. “New public-key cryptosystem using braid groups”. In: *Advances in cryptology—CRYPTO 2000 (Santa Barbara, CA)*. Vol. 1880. Lecture Notes in Comput. Sci. Springer, Berlin, 2000, pp. 166–183. DOI: 10.1007/3-540-44598-6_10. URL: http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1007/3-540-44598-6_10.
- [45] K. H. Ko et al. *New Signature Scheme Using Conjugacy Problem*. Cryptology ePrint Archive, Report 2002/168. <http://eprint.iacr.org/>. 2002.
- [46] C. Krattenthaler. “THE THEORY OF HEAPS AND THE CARTIER–FOATA MONOID”. In: *Appendix of the electronic edition of “Problèmes combinatoires de commutation et réarrangements (2006)*.
- [47] E. Lee and J. H. Park. “Cryptanalysis of the public-key encryption based on braid groups”. In: *Advances in cryptology—EUROCRYPT 2003*. Vol. 2656. Lecture Notes in Comput. Sci. Springer, Berlin, 2003, pp. 477–490. DOI: 10.1007/3-540-39200-9_30. URL: http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1007/3-540-39200-9_30.
- [48] A. Malyutin. “Fast Algorithms for Identification and Comparison of Braids”. English. In: *Journal of Mathematical Sciences* 119.1 (2004), pp. 101–111. ISSN: 1072-3374. DOI: 10.1023/B:JOTH.0000008747.72654.14. URL: <http://dx.doi.org/10.1023/B%3AJOTH.0000008747.72654.14>.
- [49] V. Manturov. *Knot Theory*. CRC Press, 2004. ISBN: 9780203402849. URL: <https://books.google.com/books?id=1RLwUP8LLLcC>.
- [50] N. Mosina and A. Ushakov. “Mean-set attack: cryptanalysis of Sibert et al. authentication protocol”. In: *J. Math. Cryptol.* 4.2 (2010), pp. 149–174. ISSN: 1862-2976. DOI: 10.1515/JMC.2010.006. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1515/JMC.2010.006>.
- [51] M. S. Paterson and A. A. Razborov. “The set of minimal braids is co-NP-complete”. In: *J. Algorithms* 12.3 (1991), pp. 393–408. ISSN: 0196-6774. DOI: 10.1016/0196-6774(91)90011-M. URL: [http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1016/0196-6774\(91\)90011-M](http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1016/0196-6774(91)90011-M).

REFERENCES

- [52] V. Prasolov and A. Sossinsky. *Knots, Links, Braids, and 3-manifolds: An Introduction to the New Invariants in Low-dimensional Topology*. Translations of mathematical monographs. American Mathematical Society, 1997. ISBN: 9780821808986. URL: <https://books.google.com/books?id=znCLtJKnZXQC>.
- [53] J. H. Przytycki. “Classical roots of knot theory”. In: *Chaos Solitons Fractals* 9.4-5 (1998). Knot theory and its applications, pp. 531–545. ISSN: 0960-0779. DOI: 10.1016/S0960-0779(97)00107-0. URL: [http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1016/S0960-0779\(97\)00107-0](http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1016/S0960-0779(97)00107-0).
- [54] R. G. Scharein. “Interactive Topological Drawing”. PhD thesis. Department of Computer Science, The University of British Columbia, 1998.
- [55] H. Sibert. “Extraction of roots in Garside groups”. In: *Comm. Algebra* 30.6 (2002), pp. 2915–2927. ISSN: 0092-7872. DOI: 10.1081/AGB-120003997. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1081/AGB-120003997>.
- [56] H. Sibert, P. Dehornoy, and M. Girault. “Entity authentication schemes using braid word reduction”. In: *Discrete Appl. Math.* 154.2 (2006), pp. 420–436. ISSN: 0166-218X. DOI: 10.1016/j.dam.2005.03.015. URL: <http://dx.doi.org/10.1016/j.dam.2005.03.015>.
- [57] W. Stein et al. *Sage Mathematics Software (Version 6.7)*. The Sage Development Team. 2015. URL: <http://www.sagemath.org>.
- [58] V. B. Styšnev. “Taking the root in the braid group”. In: *Izv. Akad. Nauk SSSR Ser. Mat.* 42.5 (1978), pp. 1120–1131, 1183. ISSN: 0373-2436.
- [59] J.-L. Thiffeault. “Braidlab: A Software Package for Braids and Loops”. In: *arXiv preprint arXiv:1410.0849* (2014).
- [60] A. T. Vandermonde. “Remarques sur les probl’emes de situation”. In: *Memoires de l’Académie Royale des Sciences* (1771), pp. 566–574.
- [61] V. V. Vershinin. “Braids, their Properties and Generalizations”. In: vol. Volume 4. Handbook of Algebra. North-Holland, 2009, pp. 427–465. ISBN: 1570-7954.

REFERENCES

- [62] G. X. Viennot. “Heaps of pieces. I. Basic definitions and combinatorial lemmas”. In: *Combinatoire énumérative (Montreal, Que., 1985/Quebec, Que., 1985)*. Vol. 1234. Lecture Notes in Math. Springer, Berlin, 1986, pp. 321–350. DOI: 10.1007/BFb0072524. URL: <http://dx.doi.org.proxy.lib.uwaterloo.ca/10.1007/BFb0072524>.
- [63] P. Xu. “Growth of the positive braid semigroups”. In: *Journal of Pure and Applied Algebra* 80.2 (1992), pp. 197–215.