UNIVERSITY OF CALIFORNIA
Santa Barbara

Braid Groups and Euclidean Simplices

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Mathematics

by

Elizabeth Eileen Leyton Chisholm

Committee in Charge:

Professor Jon McCammond, Chair

Professor Stephen Bigelow

Professor Darren Long

June 2015

The Dissertation of
Elizabeth Eileen Leyton Chisholm is approved:

Professor Stephen Bigelow

Professor Darren Long

Professor Jon McCammond, Committee Chairperson

May 2015

Braid Groups and Euclidean Simplices

by

Elizabeth Eileen Leyton Chisholm

For Tata, my kindred spirit.

Acknowledgements

First and foremost, thank you to Jon McCammond without whose support and guidance I would not have succeeded in this endeavor. His insights and belief in me made this dissertation possible. Thank you also to all of the professors at UCSB who helped my growth as a mathematician over the years. Thanks to Medina Price for her invaluable advice and guidance, and for letting me rant when I needed to. I would also like to acknowledge my professors at CSUN who started me on my mathematical career. Thank you to the other graduate students at UCSB that I battled in the trenches with. I would especially like to thank my BMB Arielle Leitner who has always been there to listen and give me advice.

I wouldn't be where I am today without my family. Thank you to my daughter Leila for giving me a reason to build a better life and to my daughter Julia for always giving me a reason to smile. Thank you to my mother Lisa for making so many sacrifices to help me succeed and to my sister Catherine for always being there for me. Thank you to my father Alex, my stepmom Katherine, the rest of my siblings, my grandparents, and all of my extended family and friends, each of whom played a role in helping me achieve my goals. Finally, I would like to thank my husband Danny. Thank you for taking this journey with me and being my biggest source of love and support.

Curriculum Vitæ

Elizabeth Eileen Leyton Chisholm

Education

University of California, Santa Barbara, Santa Barbara, CA	
Ph.D., Mathematics	6/2015
California State University, Northridge, Northridge, CA	
M.S., Mathematics	5/2009
B.S., Applied Mathematics	8/2007

Teaching Activities

UC Santa Barbara (Teaching Associate)	
Math 34B, Calculus for the Life and Social Sciences II	1 quarter
Math 3A, Calculus with Applications I	1 quarter
Math 4B, Differential Equations	1 quarter
UC Santa Barbara (Teaching Assistant)	
Math 34A, Calculus for the Life and Social Sciences I	2 quarters
Math 34B, Calculus for the Life and Social Sciences II	2 quarters
Math 3A, Calculus with Applications I	3 quarters
Math 3C, Linear Algebra and Differential Equations	1 quarter
Math 4B, Differential Equations	2 quarters
Math 6A, Vector Calculus I	2 quarters
Math 108A, Introduction to Linear Algebra	1 quarter
Math 111B, Abstract Algebra	1 quarter
CSU Northridge	
Math 131, Mathematical Ideas	1 semester
Santa Barbara Business College	
Math 110, Elementary Algebra	1 quarter
Math 210, Intermediate Algebra	1 quarter

Fellowships and Awards

UCSB Dept. of Mathematics Graduate Student Teaching Award	2014
UCSB Academic Senate Doctoral Travel Grant	2014
Ralph M. Parsons Fellowship	2009-2014
UCSB Math Dept. Raymond L. Wilder Award	2010
CSUN Math Dept. Outstanding Academic Achievement	2009

NSF GK-12 Fellowship	20007-2008
CSUN Math Dept. Outstanding Academic Achievement	2007
NSF MCTP Grant	2006-2007

Preprints

Braid Groups and Euclidean Simplices, arXiv:1411.1028.
(with Jon McCammond)

Posters and Talks

AWM Poster Session, Joint Mathematics Meetings, San Antonio, TX	1/2015
Configuration Spaces Poster Session, Cortona, Italy	9/2014
5th Annual Women in Mathematics Symposium, USC	10/2012
UCSB Graduate Algebra Seminar, UCSB	11/2011
Nebraska Women in Mathematics Conference, UN - Lincoln	2/2007

Conferences Attended

Opportunities in Mathematical Sciences, UC Irvine	4/2015
Joint Math Meetings, San Antonio, TX	1/2015
Configuration Spaces, Cortona, Italy	9/2014
Joint Math Meetings, Baltimore, CA	1/2014
Joint Math Meetings, San Diego, CA	1/2013
5th Annual Women in Mathematics Symposium, USC	10/2012
Park City Math Institute, Park City, UT	7/2012
4th Annual Women in Mathematics Symposium, LMU	1/2011
3rd Annual Women in Mathematics Symposium, Pomona College	1/2010
8th Annual NSF GK-12 Meeting, Washington, DC	2/2008
Joint Math Meetings, San Diego, CA	1/2008
Women in Mathematics Seminar, Institute of Advanced Study	5/2007
Nebraska Women in Mathematics Conference, UN - Lincoln	2/2007

Abstract

Braid Groups and Euclidean Simplices

Elizabeth Eileen Leyton Chisholm

In the early 2000s, Daan Krammer and Stephen Bigelow independently proved that braid groups are linear. They used the Lawrence-Krammer-Bigelow (LKB) representation for generic values of its variables q and t. The t variable is related to the Garside structure of the braid group used in Krammer's algebraic proof. The q variable, associated with the dual Garside structure of the braid group, has received less attention.

In this dissertation we give a geometric interpretation of the q portion of the LKB representation in terms of an action of the braid group on the space of non-degenerate euclidean simplices. In our interpretation, braid group elements act by systematically reshaping (and relabeling) euclidean simplices. The reshapings associated to the simple elements in the dual Garside structure of the braid group are of an especially elementary type that we call relabeling and rescaling.

Contents

List of Figures **x**

1 Introduction **1**

2 The Braid Groups **6**
2.1 $\textsc{Braid}_n$ as the braid group on n-strings 6
2.2 $\textsc{Braid}_n$ as a mapping class group 9

3 Partitions, Permutations and Braids **13**
3.1 Noncrossing Partitions and Permutations 13
3.2 Dual Simples, Presentations of $\textsc{Braid}_n$ and Complements 19

4 Euclidean Simplices **28**
4.1 Geometry of Euclidean Simplices 28
4.2 Edge Rescaling 32

5 Braid Groups and Euclidean Simplices **46**
5.1 Representations of $\textsc{Braid}_n$ 46
5.2 Braid Groups Act on Euclidean Simplices 50
5.3 Dual Simple Braids Act by Rescaling and Relabeling 53

6 Origins and Future Work **56**
6.1 Origins 56
6.2 Future Work 57

Appendix **60**

Bibliography **75**

List of Figures

2.1 A braid in $\textsc{Braid}_4$ 7
2.2 A product of braids 8
2.3 Representative of $\psi_1 \in \textsc{Braid}_4$ 10
2.4 $\mathbf{D}_n$ 10
2.5 Half-twist ϕ_{12} in D_9 11

3.1 Edges of D_4 15
3.2 Examples of pairs of edges in $\mathbf{D}_9$ 16
3.3 The subsets $\{1,2,4,5\}$ and $\{3,7,8\}$ are crossing. The subsets $\{1,2,4,5\}$ and $\{7,8,9\}$ are noncrossing. 17
3.4 Noncrossing partition lattice NC_4. 18
3.5 The rotation s_{137}. 20
3.6 The dual simple elements in $\textsc{Braid}_4$. 21
3.7 Admissible and nonadmissible partitions in $\mathbf{D}_9$ 23
3.8 $s_{123456789} = s_{123}s_{3456789}$ 24
3.9 Left and right complement example in $\textsc{Sym}_9$. 25
3.10 If $\sigma_1 = (2,3,4,5)$ and $\sigma_2 = (5,6,7)$, then the permutations σ_3, σ_4 and σ_5 defined in Definition 3.2.10 are $\sigma_3 = (1,7,8,9)$, $\sigma_4 = (1,5,8,9)$ and $\sigma_5 = (1,2,8,9)$. 27

4.1 Tetrahedron determined by 4 points, edges labeled by norm. . . . 31
4.2 An edge reshaping $R = R^{12}_{23}$ that rescales e_{12} by a factor of q and fixes e_{23} (i.e. rescales e_{23} by a factor of 1). 35
4.3 The edge rescaling R^{12}_{234} which fixes the triangle Δ_{234} and rescales edge e_{12} by a factor of q. 38
4.4 The edge rescaling $R^{24}_{\mathrm{rc}(24)}$ which rescales the edge e_{24} while fixing the edges e_{23} and e_{14}. 43

Chapter 1

Introduction

The braid groups were formally introduced by Emil Artin in the 1920s [1]. He formalized the notion that braids with n strings form a group called the braid group on n strings, denoted $\textsc{Braid}_n$. In particular, he described this group algebraically as the following:

Definition 1.0.1 (Braid group, standard generators)**.** The *braid group*, $\textsc{Braid}_n$, is a group with presentation

$$\langle s_1, \ldots, s_{n-1} \mid s_i s_{i+1} s_i = s_{i+1} s_i s_{i+1}, i = 1, \ldots, n-2, s_i s_j = s_j s_i, |i-j| > 1 \rangle$$

We will call the generators $s_1, \ldots, s_{n-1}$ the *standard generators* and denote this set by $\textsc{Std}_n$.

Topologists and algebraists have studied these groups extensively. One property of the braid groups that has been investigated is the linearity of $\textsc{Braid}_n$.

Definition 1.0.2 (Linearity of a group)**.** A group G is said to be *linear* if it admits an injective homomorphism $\rho : G \to GL_m(\mathbb{R})$ for some natural number m. Equivalently, a group G is linear if it admits a faithful representation into $GL_m(\mathbb{R})$.

The first candidate for a faithful representation of $\textsc{Braid}_n$ was proposed in the 1930s by Werner Burau and is known as the Burau representation [7]. In 1991, Moody showed that it is not faithful for $n \geq 9$ [19]. This was extended to $n \geq 6$ by Long and Patton and finally to $n \geq 5$ by Bigelow in [17] and [2], respectively. The representation is faithful for $n \geq 3$, but it is still unknown whether the Burau representation is faithful for $n = 4$.

Another candidate for a faithful representation of $\textsc{Braid}_n$ was introduced by Ruth Lawrence in 1999 [16]. This representation was studied extensively by Daan Krammer and Stephen Bigelow and is now commonly called the Lawrence-Krammer-Bigelow (LKB) representation. Three papers regarding the faithfulness of the LKB representation (for suitably generic values of its variables q and t) appeared in the early 2000s. First, Krammer published a proof that the LKB representation is faithful for $n = 4$ [14]. Shortly following this, Bigelow proved this representation is faithful for all n [4]. Bigelow's proof was topological nature,

and Krammer followed this with a more algebraic proof in [15]. The main difference between the two papers by Krammer is that the first uses the dual Garside presentation of the braid groups, also known as the Birman-Ko-Lee representation [5], and the second reverts to the standard presentation of the braid group.

These linearity results of the braid groups were extended to prove linearity of certain Artin groups. In particular, Arjeh Cohen and David Wales proved that Artin groups of finite type are linear [8] and François Digne extended linearity to the Artin groups of crystallographic type [10]. Following this, Luis Paris generalized these results further by proving that the Artin monoid injects in the generalization of the LKB representation [20]. These results, as well as Krammer's algebraic proof, heavily utilize the meaning of the t variable in the representation.

The q variable in the LKB representation has received less attention. Recently, Tetsuya Ito and Bert Wiest made some progress in this when they proved that the highest power of q in the LKB representation of a dual positive braid is its dual Garside length [12]. In this dissertation, we present a new way to interpret the q variable. In order to focus on the q variable, we consider the LKB representation with $t = 1$. We call this the *simplicial representation*. We show that every element of the n-string braid group determines a reshaping of a metric euclidean simplex, and in particular the dual simple elements in the dual Garside structure have an especially nice way of reshaping the simplices.

We have the following two theorems.

Theorem A (Braids act on metric euclidean simplices)**.** *The action of the specialized LKB representation of* $\textsc{Braid}_n$ *with* $t = 1$*, referred to as the simplicial representation, preserves the subset of* $\mathbb{R}^N$*, with* $N = \binom{n}{2}$*, that represents the squared edge lengths of euclidean* $(n-1)$*-simplices.*

More explicitly, let ρ be the simplicial representation of the n-string braid group acting on $\mathbb{R}^N$ with $q \in \mathbb{R}$, $t = 1$, and written with respect to the explicit basis used by Krammer in [14]. We will call this the simplicial representation. Let $\mathbf{v} \in \mathbb{R}^N$ be a tuple representing the squared lengths of the edges of a metric $(n-1)$-simplex, Δ. Let β be a braid and $\mathbf{v}'$ be the image of $\mathbf{v}$ under the action of $\rho(\beta)$. Then $\mathbf{v}'$ also represents the squared lengths of the edges of a euclidean $(n-1)$-simplex, Δ'.

In particular, the standard generators of the braid group, as well as the dual simple elements in the dual garside structure of the braid group, reshape simplices in a very elementary way that we call edge rescaling.

Definition 1.0.3 (Edge Rescaling)**.** Let Δ and Δ' be two euclidean simplices with labeled vertices in a common vector space. We say that an edge e in Δ is merely *rescaled* if it and the corresponding edge e' in Δ' point in the same direction. More generally, we say that Δ' is an *edge rescaling* of Δ if there exist enough pairs of

corresponding edges pointing in the same direction (but with possibly different lengths) to form a vector space basis out of these common direction vectors.

We then have the following theorem.

Theorem B (Dual simple braids relabel and rescale)**.** *Under the simplicial representation of the braid group, each dual simple braid acts by relabeling the vertices and rescaling specific edges.*

The structure of the dissertation is as follows. The first chapter discusses the braid groups in a traditional sense, while the second introduces the dual Garside structure. Next, we discuss the geometry of Euclidean simplices and edge rescalings in particular. The fourth chapter discusses representations of the braid group and focuses on the action of the simplicial representation on euclidean simplices as in Theorem A. In the fifth chapter we focus how dual simple braids act on simplices, in particular addressing Theorem B. The sixth chapter focuses on future plans.

Chapter 2

The Braid Groups

2.1 Braid_n as the braid group on n-strings

Traditionally BRAID_n is thought of as the braid group on n-strings. We now make this idea more precise. For a more in depth discussion see [13].

Definition 2.1.1 (Braid on n-strings)**.** A *braid on n-strings* is a set $b \subset \mathbb{R}^2 \times I$ consisting of n disjoint topological intervals called the strings of b such that the projection from $\mathbb{R}^2 \times I$ to I maps each string homeomorphically onto I and

$$b \cap (\mathbb{R}^2 \times 0) = \{(1,0,0),(2,0,0),\ldots,(n,0,0)\}$$

$$b \cap (\mathbb{R}^2 \times 1) = \{(1,0,1),(2,0,1),\ldots,(n,0,1)$$

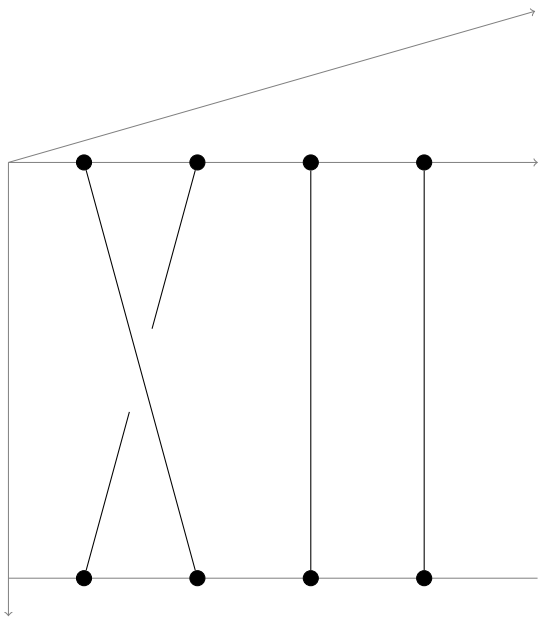

Figure 2.1: A braid in $\textsc{Braid}_4$

Based on this definition, we see that each string of b connects some point $(i, 0, 0)$ to some $(j, 0, 1)$ and intersects each plane $\mathbb{R}^2 \times \{t\}$ for $t \in [0, 1]$ at exactly one point. An example of a braid in $\textsc{Braid}_4$ is shown in Figure 2.1.

Definition 2.1.2 (Products of braids on n-strings)**.** Given two braids on n-strings b_1 and b_2, we define their *product* $b_1 b_2$ as the set of points $(x, y, t) \in \mathbb{R}^2 \times I$ such that $(x, y, 2t) \in b_1$ if $0 \leq t \leq \frac{1}{2}$ and $(x, y, 2t - 1) \in b_2$ for $\frac{1}{2} \leq t \leq 1$.

Figure 2.2 shows the product of the braid in $\textsc{Braid}_4$ shown in Figure 2.1 with itself. This multiplication is associative and has an identity element given by the trivial braid $\{1, 2, \ldots, n\} \times \{0\} \times I \subset \mathbb{R}^2 \times I$.

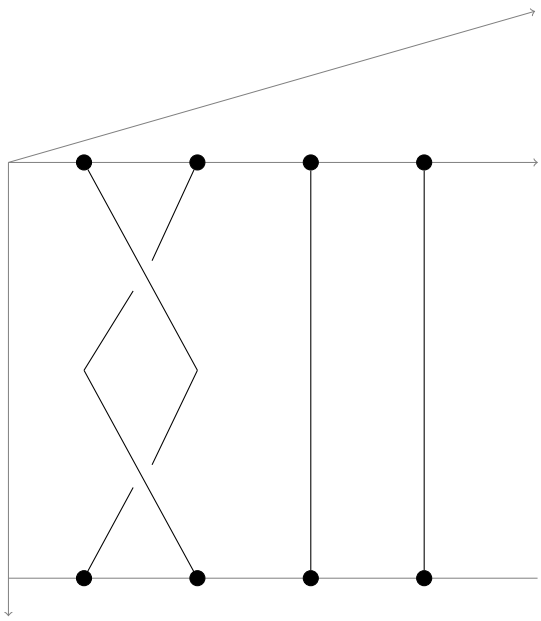

Figure 2.2: A product of braids

Definition 2.1.3 (Isotopic braids)**.** Two braids on n-stings b and b' are *isotopic* if there exists a continuous map $F : b \times I \to \mathbb{R}^2 \times I$ such that for each $s \in I$ we have $F(b \times s)$ is a braid on n-strings, $F(b \times 0) = b$ and $F(b \times 1) = b'$.

It is clear that Definition 2.1.3 gives an equivalence relation on the braids on n-strings. It is also clear that if b_1 is isotopic to b_1' and b_2 is isotopic to b_2' then $b_1 b_2$ is isotopic to $b_1' b_2'$.

Theorem 2.1.4. *Isotopy classes of braids on n-strings with multiplication as in Definition 2.1.2 form a group, denoted $\mathcal{B}_n$. Moreover, this group is isomorphic to* $\textsc{Braid}_n$ *as defined in Definition 1.0.1.*

Proof. Based on Definition 2.1.2 and Definition 2.1.3 it is clear that $\mathcal{B}_n$ is a group. Let ψ_i denote the equivalence class with representative the braid with i^{th} string connecting $(i, 0, 0)$ and $(i+1, 0, 1)$, the $(i+1)^{st}$ string connecting $(i+1, 0, 0)$ and $(i, 0, 1)$ with the i^{th} string behind the $(i+1)^{st}$ string, and j^{th} string connecting $(j, 0, 0)$ and $(j, 0, 1)$ for $j \neq i, i+1$. See Figure 2.3 for an example.

Define a map

$$\textsc{Braid}_n \to \mathcal{B}_n$$

$$s_i \mapsto \psi_i$$

for $i = 1, \ldots, n-1$. It can be shown that $\psi_1, \ldots, \psi_{n-1}$ satisfy the braid relations and that this map is an isomorphism. See [13] for a detailed proof of this theorem.

□

2.2 Braid$_n$ as a mapping class group

The braid group $\textsc{Braid}_n$ is also isomorphic to the mapping class group of an n-punctured disk. We now make this more precise.

Definition 2.2.1 (Half-twist)**.** Let $\mathbf{D}_n$ be a topological disc in $\mathbb{R}^2$ with labeled punctures p_i for $i = 1, \ldots, n$. Given two punctures i and j in $\mathbf{D}_n$, we define the following homeomorphism. Connect i and j by a straight line arc and take a small neighborhood of this path avoiding all other punctures. Define the *half-twist* ϕ_{ij}

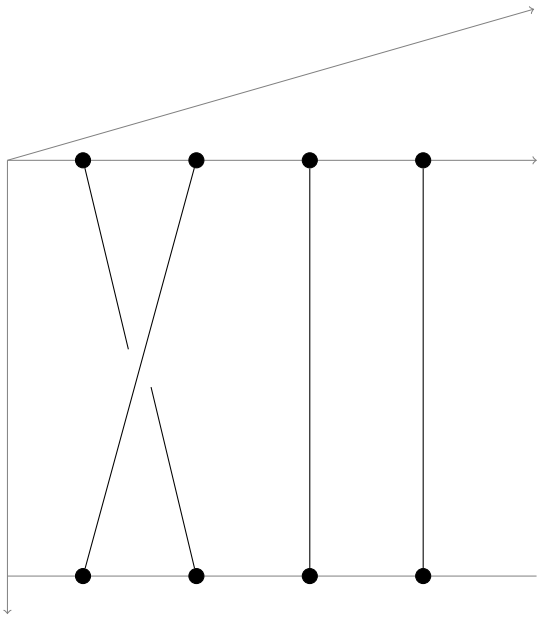

Figure 2.3: Representative of $\psi_1 \in \textsc{Braid}_4$

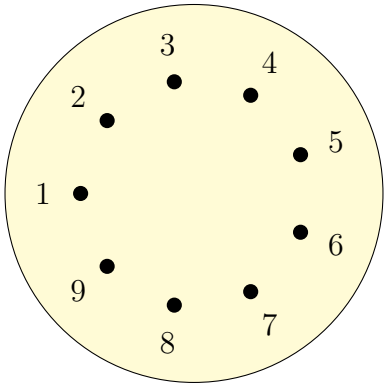

Figure 2.4: $\mathbf{D}_n$

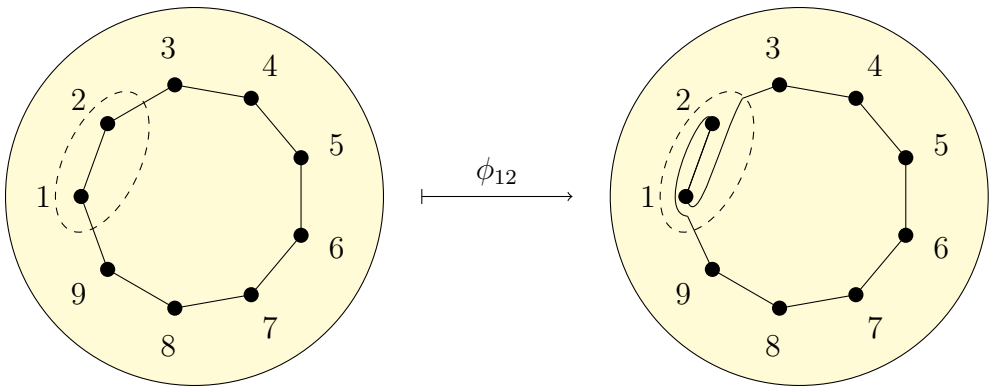

Figure 2.5: Half-twist ϕ_{12} in D_9

to be the homeomorphism switching i and j clockwise around each other in this neighborhood where punctures pass on the left to avoid collision.

The half-twist ϕ_{12} in $\textsc{Braid}_8$ is pictured in Figure 2.5. To witness this homeomorhpism, we have included the n-gon along with its image under the homeomorphism.

Definition 2.2.2 (Mapping class group of $\mathbf{D}_n$)**.** The mapping class group of $\mathbf{D}_n$, $\textsc{Mcg}(\mathbf{D}_n)$ is the set of equivalence classes of homeomorphisms of $\mathbf{D}_n$ that fix the boundary of $\mathbf{D}_n$.

It is a well known fact that $\textsc{Braid}_n$ is isomorphic to the mapping class group $\textsc{Mcg}(\mathbf{D}_n)$. For an in depth discussion of this, see [11].

Theorem 2.2.3. *The mapping class group* $\textsc{Mcg}(\mathbf{D}_n)$ *is generated by the* $n-1$ *equivalence classes of half-twists* $\phi_{i,i+1}$ *for* $i = 1, \ldots, n-1$. *Moreover, this group is isomorphic to the braid group* $\textsc{Braid}_n$ *in Definition 1.0.1.*

Proof. To see that $\textsc{Mcg}(\mathbf{D}_n)$ is isomorphic to $\textsc{Braid}_n$, we define a map

$$\textsc{Mcg}(\mathbf{D}_n) \to \mathcal{B}_n$$

$$\phi_{i,i+1} \mapsto \psi_i$$

For a discussion of this isomorphism see [11]. Since we know that $\mathcal{B} \cong \textsc{Braid}_n$ this completes the proof. In particular, we can identify the equivalence class $\phi_{i,i+1}$ with the standard generator s_i.

□

Remark 2.2.4. Note that by identifying the standard generators of $\textsc{Braid}_n$ with equivalence classes of homeomorphisms, we will compose braids from right to left as with function composition.

Chapter 3

Partitions, Permutations and Braids

In this chapter we first discuss non-crossing partitions and permutations in a convexly punctured disc, and then use these ideas to discuss dual simple elements, presentations of the braid group and some properties.

3.1 Noncrossing Partitions and Permutations

In this section convexly punctured discs are used to define the lattice of non-crossing partitions as well as noncrossing permutations.

Definition 3.1.1 (Convexly punctured disc)**.** Let $\mathbf{D}_n$ be a topological disc in the euclidean plane with a distinguished n-element subset that we call its *punctures* or *vertices*. When the disc $\mathbf{D}_n$ is a convex subset of $\mathbb{R}^2$ and the convex hull of its n punctures is an n-gon (i.e. every puncture occurs as a vertex of the convex hull) then we say that $\mathbf{D}_n$ is a *convexly punctured disc*. See Figure 2.4. There is a natural cyclic ordering of the vertices corresponding to the clockwise orientation of the boundary cycle of the n-gon. A labeling of the vertices is said to be *standard* if it uses the set $[n] := \{1, 2, ..., n\}$ and the vertices are labeled in the natural cyclic order. More generally, when the vertices p_i are bijectively labeled by elements i in a finite set A, we refer to the convexly punctured disc as $\mathbf{D}_A$.

There are important two element subsets of the vertices of $\mathbf{D}_n$ that we call *edges*.

Definition 3.1.2 (Edges)**.** Let $\mathbf{D}_n$ be a convexly punctured disc. For each two element subset $\{i, j\} \subset [n]$, the convex hull of the corresponding points p_i and p_j in $\mathbf{D}_n$ is called an *edge*, denoted $e_{i,j} = e_{j,i}$, or possibly e_{ij} when the comma is not needed for clarity. When a *standard name* is needed we insist $i < j$. The number of edges is $\binom{n}{2}$ and we consistently use N for this number throughout the dissertation. For later use, it is also convenient to impose a *standard order* on the set of all $N = \binom{n}{2}$ edges. We do so by lexicographically ordering them by their standard names.

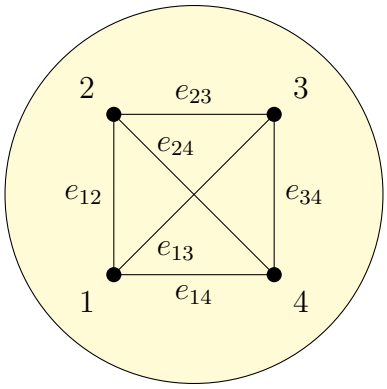

Figure 3.1: Edges of D_4

In $\mathbf{D}_4$, for example, the standard names of its $6 = \binom{4}{2}$ edges in their standard order are e_{12}, e_{13}, e_{14}, e_{23}, e_{24} and e_{34}. See Figure 3.1.

We also need words to describe the position of one edge relative to another.

Definition 3.1.3 (Pairs of edges)**.** Let (e_{ij}, e_{kl}) be an ordered pair of edges in $\mathbf{D}_n$. Consider the subdisc $\mathbf{D}_B$ where $B = \{i, j, k, l\}$. Inside of $\mathbf{D}_B$, there are precisely five possible configurations for (e_{ij}, e_{kl}):

- *noncrossing*: all four endpoints are distinct and the edges do not cross.
- *crossing*: all four endpoints are distinct and the edges cross.
- *identical*: e_{ij} and e_{kl} have both endpoints in common.

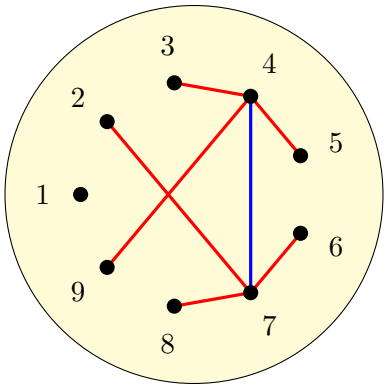

Figure 3.2: Examples of pairs of edges in $\mathbf{D}_9$

- *clockwise*: e_{ij} and e_{kl} have exactly one endpoint in common and the convex hull of the three endpoints is a triangle where e_{ij} is followed by e_{kl} clockwise on the boundary. In this case, we say that e_{kl} is *to the right* of e_{ij}.

- *counterclockwise*: e_{ij} and e_{kl} have exactly one endpoint in common and the convex hull of the three endpoints is a triangle where e_{ij} is followed by e_{kl} counterclockwise on the boundary. In thise case, we say that e_{kl} is *to the left* of e_{ij}.

Consider the example in Figure 3.2. The edges e_{34}, e_{49} and e_{67} are to the left of the edge e_{47} and the edges e_{27}, e_{78} and e_{45} are to the right. This is because ordered pairs such as (e_{34}, e_{47}) and (e_{67}, e_{47}) are clockwise while the ordered pair (e_{27}, e_{47}) is counterclockwise.

We now define noncrossing partitions in a convexly punctured disc.

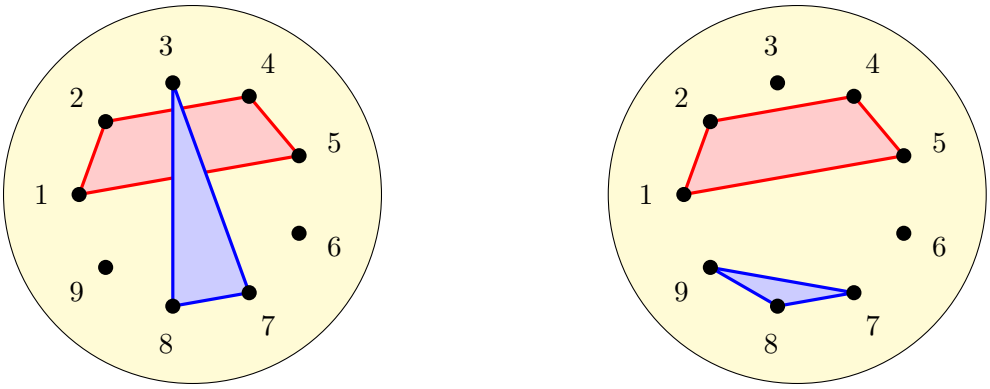

Figure 3.3: The subsets $\{1, 2, 4, 5\}$ and $\{3, 7, 8\}$ are crossing. The subsets $\{1, 2, 4, 5\}$ and $\{7, 8, 9\}$ are noncrossing.

Definition 3.1.4 (Noncrossing partitions)**.** Let $\mathbf{D}_n$ be a convexly puncured disc. Two subsets $B, B' \subset [n]$ are *noncrossing* when the convex hulls of the vertices of B and B', respectively, are completely disjoint. More generally, a partition σ of $[n]$ is *noncrossing* when its blocks are pairwise noncrossing. Noncrossing partitions can be partially ordered by refinement, where $\sigma < \tau$ if and only if each block of σ is contained in some block of τ. With this ordering, the set of all noncrossing partitions of $[n]$ form a bounded graded lattice denoted NC_n. The number of noncrossing partitions in NC_n is $C_n = \frac{1}{n+1}\binom{2n}{n}$, the n-th Catalan number.

See Figure 3.3 for an example of two noncrossing subsets in $\mathbf{D}_9$. The poset NC_4 is shown in Figure 3.4.

To each noncrossing partition of $[n]$ we can assign a permutation in $\textsc{Sym}_n$.

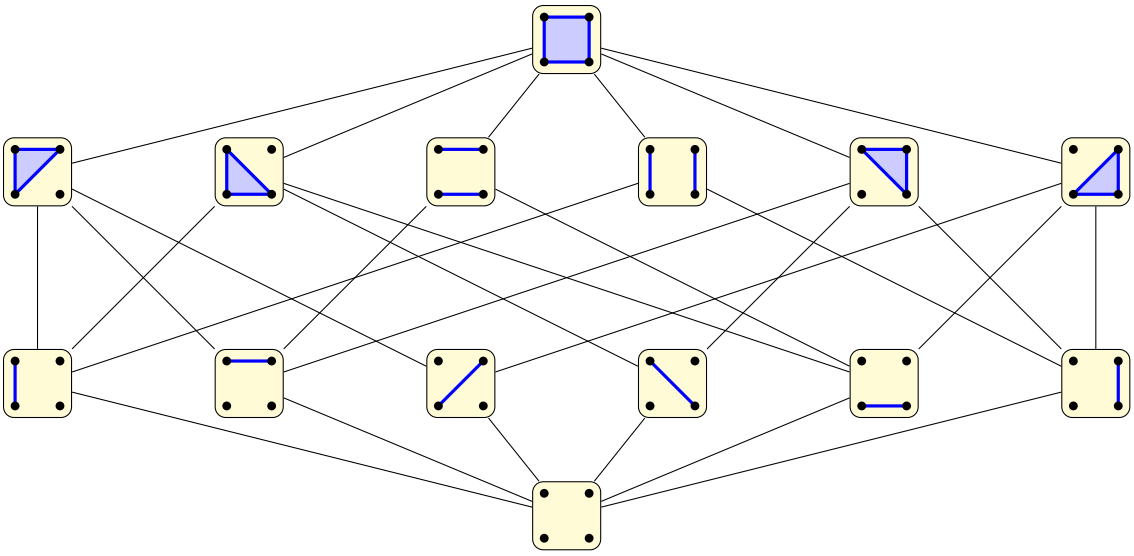

Figure 3.4: Noncrossing partition lattice NC_4.

Definition 3.1.5 (Noncrossing permutations)**.** To every subset $B \subset [n]$ with $|B| \geq 2$ we can associate a permutation in $\textsc{Sym}_n$ by linearly ordering the elements of B. In general, we associate a permutation to each partition by multiplying the permutations that correspond to its blocks. This product is well-defined as the blocks of a noncrossing partitions are disjoint and hence their corresponding permutations commute. We identify each noncrossing partition with its noncrossing permutation using the same symbol for both. The permutation associated to the full set $[n]$ is an important n-cycle that we call δ.

As an example, the subset $B = \{1, 3, 4\}$ becomes the permutation (134) and the noncrossing partition $\sigma = \{\{1, 3, 4\}, \{2\}, \{5, 6, 7, 8, 9\}\}$ becomes the permutation $\sigma = (134)(56789)$.

3.2 Dual Simples, Presentations of Braid$_n$ and Complements

In this section we discuss the dual simple braids, presentations of $\textsc{Braid}_n$ and introduce the definition and properties of the left and right complement.

From Theorem 2.2.2, we know that elements of the braid group can be identified with equivalence classes of motions of the vertices in $\mathbf{D}_n$. The dual simple braids are a finite set of braids indexed by the noncrossing permutations as follows.

Definition 3.2.1 (Rotations)**.** For each set $B \subset [n]$, let P_B be the convex hull of the vertices indexed by B inside of $\mathbf{D}_B$. The braid group element s_B is the motion where each labeled point in $\mathbf{D}_B$ moves clockwise along the boundary of P_B to the next vertex, leaving all other vertices fixed. The braid element s_δ is the motion where each labeled point in $\mathbf{D}_n$ moves clockwise along the boundary of the convex n-gon formed by the vertics of $\mathbf{D}_n$ to the next vertex. When $|B| = 1$, the motion is trivial. When B has two elements, the motion is the same as the half-twist in Definition 2.2.1.

See Figure 3.5 for an example of a rotation in B_9.

The association of rotations to subsets of $[n]$ allows us to associate each noncrossing partition to an element of the braid group.

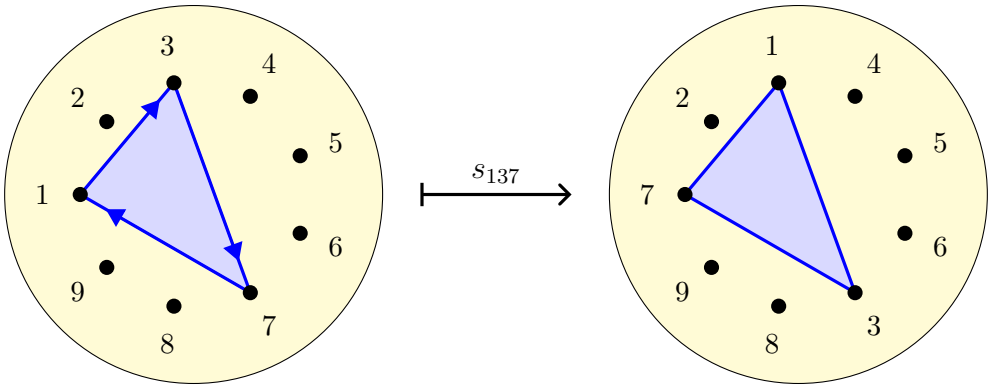

Figure 3.5: The rotation s_{137}.

Definition 3.2.2 (Dual simple braids)**.** The *dual simple braids* are elements of $\textsc{Braid}_n$ in one-to-one correspondence with the noncrossing partitions NC_n. In particular, for a noncrossing partition σ, we can associate the braid given by the product of the rotations corresponding to each of its blocks denoted s_σ. This is well defined as the blocks of a noncrossing partition correspond to rotations that occur in disjoint subdiscs.

The dual simple braids in $\textsc{Braid}_4$ written as products of rotations are shown in Figure 3.6.

There are four particular subsets of the dual simple braids that we will use in this dissertation, and as such we give them the following names. First, we note that the standard generators given in Definition 1.0.1 can be rewritten in our new notation as the rotation $s_{i,i+1}$.

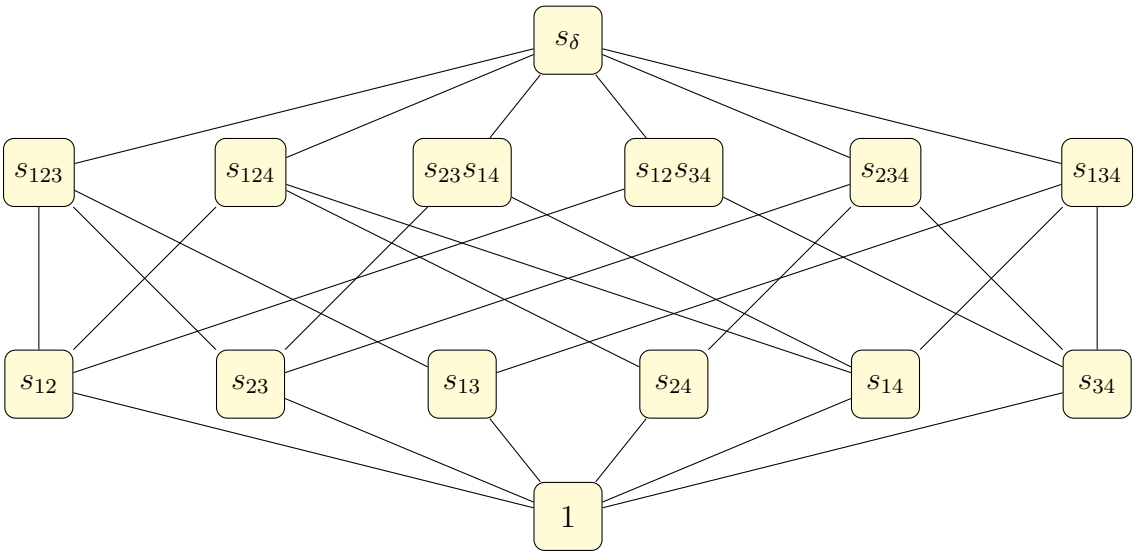

Figure 3.6: The dual simple elements in $\textsc{Braid}_4$.

Definition 3.2.3 (Four sets of simple braids)**.** The *standard generating set* of the braid group $\textsc{Braid}_n$ is the set identified in Definition 1.0.1 consisting of the $n-1$ rotations $s_{i,i+1}$ for $i = \{1, \ldots, n-1\}$. The *dual generators* of $\textsc{Braid}_n$ are the $N = \binom{n}{2}$ set of rotations s_B where $|B| = 2$. The *rotations* are the elements of $\textsc{Braid}_n$ given by s_B where $|B| \neq 1$. The full set of all dual simple braids form a fourth set. We write $\textsc{Std}_n \subset \textsc{Gen}_n \subset \textsc{Rot}_n \subset \textsc{Simp}_n$ for these four nested sets, whose sizes are $n-1$, N, $2^n - n$ and $C_n = \frac{1}{n+2}\binom{2n}{n}$.

When $n = 4$, these sets have 3, 6, 11 and 14 elements, respectively. It is clear that each of the sets in Definition 3.2.3 generate $\textsc{Braid}_n$. We can use these sets to develop alternate presentations of the braid group. In particular, we can restate

the dual presentation of the braid group in [5], as well as the rotation presentation of [18], using our language.

Note again that we are breaking with convention and elements of $\textsc{Sym}_n$, as well as elements of $\textsc{Braid}_n$, are multiplied from *right to left* as in function composition. For example, the product $(1,2,3)\cdot(3,4,5)$ is $(1,2,3,4,5)$ and the product $s_{123}\cdot s_{345}$ is s_{12345}.

In order to define relations between the elements of $\textsc{Gen}_n$ and $\textsc{Rot}_n$, we first discuss special partitions of $[n]$ called *admissible partitions*. Following the convention of [18], we represent subsets of $[n]$ with uppercase letters such as A, B, C and elements of $[n]$ as lower case letters such as i, j, k. Further, we abbreviate unions such as $\{i\} \cup B \cup C$ as iBC, removing braces around single elements and using juxtaposition to indicate union.

Definition 3.2.4 (Admissible partitions)**.** If $A_1, \ldots, A_k$ are pairwise disjoint subsets of $[n]$ such that there is a place to start reading the boundary cycle of the convex hull of the points in $\cup_{i=1}^{k} A_i$ so that, reading clockwise, one encounters all of the elements in A_1 followed by A_2, followed by A_3, etc., then the partition $\{A_1, \ldots, A_k\}$ is called *admissible*.

Examples and non-examples of admissible partitions are shown in Figure 3.7. The partition ({1,2,3},{4,5}, {6,7,8}) is an admissible partition as is $(\{4,5\}, \{6,7,8\}, \{1,2,3\})$, but ({6,7,8}, {4,5}, {1,2,3}) is not.

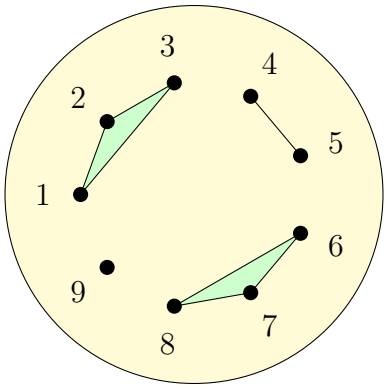

Figure 3.7: Admissible and nonadmissible partitions in $\mathbf{D}_9$

Based on Definition 3.2.2 and the multiplication of elements in $\textsc{Sym}_n$, we have the following relations for elements of $\textsc{Rot}_n$.

Definition 3.2.5 (Rotation relations)**.** The elements in $\textsc{Rot}_n$ as defined in Definition 3.2.3 satisfy the following two types of relations:

$$s_B s_C = s_C s_B \text{ when } B \text{ and } C \text{ are non-crossing (commutation)}$$

$$s_{BiC} = s_{Bi} s_{iC} \text{ when } (B, \{i\}, C) \text{ is admissible (factorization)}$$

We note that the factorization relation is slightly different from the original literature as we are composing braids from right to left.

See Figure 3.8 for an example of the factorization relation. We then have the following presentaiton of the braid group.

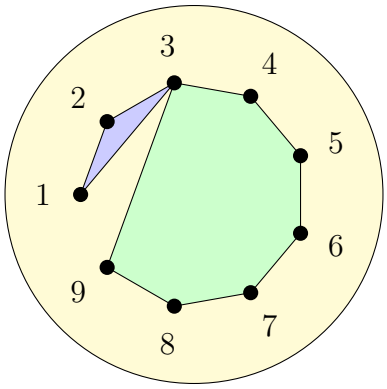

Figure 3.8: $s_{123456789} = s_{123}s_{3456789}$

We can now write the Birman-Ko-Lee presentation of the braid groups in [5], the *dual presentation*, using our notation.

Definition 3.2.6 (Dual presentation)**.** Using the generators of $\textsc{Braid}_n$ given by the set $\textsc{Gen}_n = \{s_B \mid |B| = 2\}$, we have the following presentation:

$$\left\langle \{s_{ij}\}_{1\le i<j\le n} \;\middle|\; \begin{array}{ll} s_{ij}s_{kl} = s_{ij}s_{kl} & \text{when } \{i,j\} \text{ and } \{k,l\} \text{ are non-crossing} \\ s_{ijk} = s_{ij}s_{jk} & \text{when } (\{i\},\{j\},\{k\}) \text{ is admissible} \end{array} \right\rangle$$

Using our new language, we can also define a presentation of the braid group with rotations as generators.

Definition 3.2.7 (Rotation presentation)**.** The rotations of $\textsc{Braid}_n$, $\textsc{Rot}_n$, as defined in Definition 3.2.3 generate the braid group $\textsc{Braid}_n$ with the following relations:

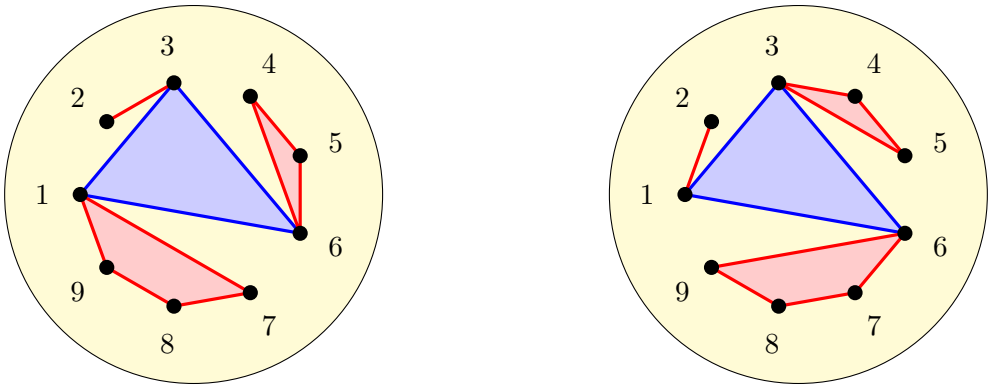

Figure 3.9: Left and right complement example in SYM_9.

$$s_B s_C = s_C s_B \text{ when } B \text{ and } C \text{ are non-crossing (commutation)}$$

$$s_{Bi} s_{iC} = s_{BiC} \text{ when } (B, \{i\}, C) \text{ is admissible (factorization)}$$

Another construction that will be crucial in our later results is the left and right complement of a noncrossing permutation.

Definition 3.2.8 (Complements)**.** Let σ be a noncrossing permutation. Recall that δ is the n-cycle of $(1, 2, \ldots, n)$. The *left complement* of σ is the unique element σ' where $\sigma'\sigma = \delta$. Similarly, the *right complement* of σ is the unique element σ'' where $\sigma\sigma'' = \delta$. These permutations are denoted $\sigma' = \text{lc}(\sigma)$ and $\sigma'' = \text{rc}(\sigma)$. The permutations $\text{lc}(\sigma)$ and $\text{rc}(\sigma)$ are always also noncrossing permutations, and in particular the edges in the blocks of $\text{lc}(\sigma)$, respectively $\text{rc}(\sigma)$, consist of the edges

that are to the left, respectively to the right, or noncrossing with respect to each of the edges in the blocks of σ.

In Figure 3.9, $(23)(456)(789)$ is the left complement of (136) in SYM_9 and $(12)(345)(6789)$ is its right complement.

The following observation is not crucial to our results, but we sometimes use this language.

Remark 3.2.9 (Hypertrees)**.** A *hypergraph* is a generalization of a graph where its *hyperedges* are allowed to span more than two vertices, and a *hypertree* is the natural generalization of a tree. As can be seen in Figure 3.9, the blocks of the noncrossing partition associated to a dual simple element and the blocks of one of its complements together form the hyperedges of a planar hypertree.

Using the notion of left and right complements, we now introduce a definition that will be useful in proving later results.

Definition 3.2.10 (Five permutations)**.** If σ_1 and σ_2 are permutations in SYM_n such that σ_1, σ_2 and their product $\sigma_1\sigma_2$ are all three noncrossing, then there exist noncrossing permutations σ_3, σ_4 and σ_5 such that $\delta = \sigma_1\sigma_2\sigma_3 = \sigma_1\sigma_4\sigma_2 = \sigma_5\sigma_1\sigma_2$.

The permutations σ_5 and σ_3 in Definition 3.2.10 are simply the left and right complements of the product $\sigma_1\sigma_2$, while σ_4 is obtained by conjugation. An example is shown in Figure 3.10.

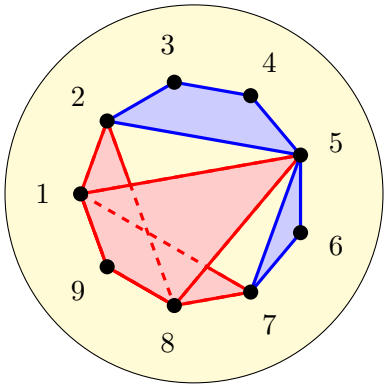

Figure 3.10: If $\sigma_1 = (2, 3, 4, 5)$ and $\sigma_2 = (5, 6, 7)$, then the permutations σ_3, σ_4 and σ_5 defined in Definition 3.2.10 are $\sigma_3 = (1, 7, 8, 9)$, $\sigma_4 = (1, 5, 8, 9)$ and $\sigma_5 = (1, 2, 8, 9)$.

Chapter 4

Euclidean Simplices

In this chapter we discuss the geometry of Euclidean simplices, and then a particular reshaping that we call edge rescaling.

4.1 Geometry of Euclidean Simplices

In this section we discuss geometry of euclidean simplices with n labeled vertices. We start by distinguishing between points and vectors as in [6].

Definition 4.1.1 (Points and Vectors)**.** Let V be an $(n-1)$-dimensional real vector space with a positive definite inner product. Let E be an $(n-1)$-dimensional euclidean space, which may be defined as a set with a fixed simply-transitive action

of the additive group of V. We call the elements of V *vectors* and the elements of E *points*. We write $\langle u, v\rangle$ for the inner product of two vectors u and v.

Vectors and points in V and E, respectively, can be used to define other objects that we calle edges and lax vectors.

Definition 4.1.2 (Edges and Lax Vectors)**.** Given two points p, p', the line segment connecting them is called an *edge* and p, p' are its *endpoints*. We also define two vectors from these points by using the simply-transitive action of V on E: the unique vector v that sends p to p' and the unique vector $-v$ that sends p' to p. The pair of vectors $\pm v$ is a *lax vector*. When p and p' are labeled, for example as p_i and p_j, we call the edge they span $e_{ij} = e_{ji}$ and the vectors they determine $v = v_{ij}$ and $-v = v_{ji}$.

Since V is equipped with an inner product, its vectors have norms.

Definition 4.1.3 (Norm)**.** The positive definite inner product defined on V gives rise to a norm on V given by the map $\textsc{Norm} : V \to \mathbb{R}$ where $\textsc{Norm}(v) = \langle v, v\rangle$. Note that the norm of a lax vector is well defined as $\textsc{Norm}(v) = \textsc{Norm}(-v)$. We also define the norm of an edge to be the norm of the lax vector determined by its endpoints. For readability, when points are labeled we write $a_{ij} = \textsc{Norm}(v_{ij})$.

Using our definition of points in E, we can define $(n-1)$-dimensional euclidean simplices.

Definition 4.1.4 (Simplices)**.** Let $\{p_i\}$ be a set of n labeled points in E. These points are said to be in *general position* if they are not contained in a proper affine subspace E. If the n points $\{p_i\}$ are in general position, then their convex hull determines a *labeled euclidean simplex* Δ of dimension $(n-1)$ and the points p_i are called the vertices of Δ.

Throughout the dissertation we use the following convention. Given a labeled euclidean simplex Δ with labeled vertices $p_1, \ldots, p_n$, we can identify simplicial faces of Δ with subsets of the vertices in the convexly punctured disc $\mathbf{D}_n$. For example, the three blocks of the left complement of s_{136} in Figure 3.9 correspond to an edge, a triangle and a tetrahedron in any 8-dimensional simplex Δ with 9 labeled vertices.

We are primarily interested in the isometry class of a labeled euclidean simplex Δ and this is completely determined by the ordered list of the norms of its edges.

Definition 4.1.5 (Edge norm vectors)**.** Let Δ be a labeled eulidean simplex with n vertices. The *edge norm vector* of Δ is a column vector $\mathbf{v} \in \mathbb{R}^N$ with $N = \binom{n}{2}$ consisting of the positive real numbers a_{ij} that are the norms of the edges e_{ij} of Δ, listed in the standard lexicographic order of edges discussed in Definition 3.1.2.

Edge norm vectors characterize classes of labeled euclidean simplices. When we reshape simplices, the modifications to the edges lengths manifest in its edge norm vector. Given two vectors u and v, we know that their inner product can

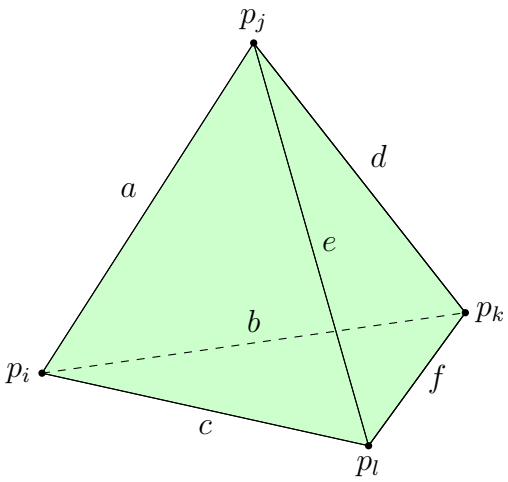

Figure 4.1: Tetrahedron determined by 4 points, edges labeled by norm.

be determined completely using norms by the formula $2\langle u, v\rangle = \textsc{Norm}(u+v) - \textsc{Norm}(u) - \textsc{Norm}(v)$. In general, however, we need a formula to determine the inner product of two vectors formed by four not necessarily distint points in a euclidean space E.

Proposition 4.1.6 (Inner products and norms)**.** *Let $p_i, p_j, p_k, p_l \in E$ be four not necessarily distint points. Then $2\langle v_{ij}, v_{kl}\rangle = a_{il} + a_{jk} - a_{ik} - a_{jl}$.*

Proof. To improve readability, we write a, b, c, d, e and f for the norms a_{ij}, a_{ik}, a_{il}, a_{jk}, a_{jl} and a_{kl}, respectively. See Figure 4.1. Expanding the norms of $v_{ik} = v_{ij} + v_{jk}$ and $v_{jl} = v_{jk} + v_{kl}$ produces the identities

$$b = a + d + 2\langle v_{ij}, v_{jk}\rangle$$

$$e = d + f + 2\langle v_{jk}, v_{kl}\rangle$$

Expanding the norm of $v_{il} = v_{ij} + v_{jk} + v_{kl}$ produces

$$c = a + d + f + 2\langle v_{ij}, v_{jk}\rangle + 2\langle v_{jk}, v_{kl}\rangle + 2\langle v_{ij}, v_{kl}\rangle$$

Thus

$$2\langle v_{ij}, v_{jk}\rangle = b - a - d,$$

$$2\langle v_{jk}, v_{kl}\rangle = e - d - f$$

and

$$2\langle v_{ij}, v_{kl}\rangle = c - a - d - f - (b - a - d) - (e - d - f) = c + d - b - e$$

□

4.2 Edge Rescaling

We now define a class of geometric shapings of labeled euclidean simplices that we call edge rescalings and investigate their properties.

Definition 4.2.1 (Edge Rescaling)**.** Let Δ and Δ' be two labeled euclidean simplices with n vertices in the same euclidean space E. An edge e_{ij} of Δ is *rescaled* if it and the corresponding edge e'_{ij} in Δ' point in the same direction. More generally, we say Δ' is an *edge rescaling* of Δ if there exist enough pairs of corresponding edges pointing in the same direction (with possibly different lengths) to form a basis for the vector space out of these common direction vectors.

Remark 4.2.2 (Spanning Trees)**.** Let Δ' be an edge rescaling of a euclidean simplex Δ. By definition, there are enough pairs of edges that are rescaled to form a basis for E. Moreover, a minimal set of these edges would form a spanning tree in the 1-skeleton of Δ consisting of these rescaled edges. However, there may exist more than one of these spanning trees. In particular, given any two edges that share an endpoint and are rescaled by the same factor, the third edge of the triangle they span would also be rescaled by that factor. Thus any two of the vectors that form this triangle could be used in the spanning tree of Δ. Because of this, it makes more sense to identify the maximal simplicial faces of Δ that are rescaled by the same factor. In particular, in light of Remark 3.2.9, we can construct a partition of the vertices where each block consists of the vertices of the maximal simplicial subsimplex for each scale factor, called a *canonical spanning hypertree*. The aforementioned spanning trees could then be identified by selecting edges inside the blocks of the canonical spanning hypertree.

Definition 4.2.3 (Edge Rescaling Maps)**.** It is clear from Definition 4.2.1 that an edge rescaling of a simplex Δ is completely described by the edges that are rescaled and their scale factors. In particular, we can define an *edge rescaling map* R from the space of euclidean simplices to itself that rescales each simplex Δ to a new simplex Δ'. Notice that each edge rescaling map R is invertible by simply

rescaling the edges rescaled by R by the multiplicative inverse of the scale factors of R.

We are particularly interested in the edge rescalings where the only scale factors are 1 and q, and we call these *q-rescalings*. We introduce a special notation for these edge rescalings as follows. Let R be a q-rescaling. Let σ be a partition where the edges in the blocks index the subsimplices which are rescaled by q, and let τ be a partition where the edges in the blocks index the subsimplices rescaled by 1. Then we write $R = R^{\sigma}_{\tau}$.

Every edge rescaling map R can be described as a matrix. The following proposition makes this more precise.

Proposition 4.2.4 (Edge Rescaling Matrices)**.** *Let R be an edge rescaling map and $\mathbf{v} \in \mathbb{R}^N$ an edge norm vector for a labeled euclidean simplex Δ. Then there exists an N by N matrix M that describes the map R whose entries only depend on R and not on $\mathbf{v}$ or Δ. In particular, $R(\mathbf{v}) = M \cdot \mathbf{v}$ for all $\mathbf{v}$ and Δ.*

Proof. Let T be a spanning tree in the 1-skeleton of Δ consisting of edges that are rescaled by R. Let e_{ij} be an edge in Δ and v_{ij} its corresponding vector. We can write the vector v_{ij} as a sum of edges in T, and hence we can write the corresponding vector v'_{ij} of Δ' as a sum where the vectors in the sum are rescaled accordingly. The norm of v'_{ij}, denoted a'_{ij}, can then be expressed as a linear combination of vectors in T. Then a'_{ij} can be written as a linear combation of

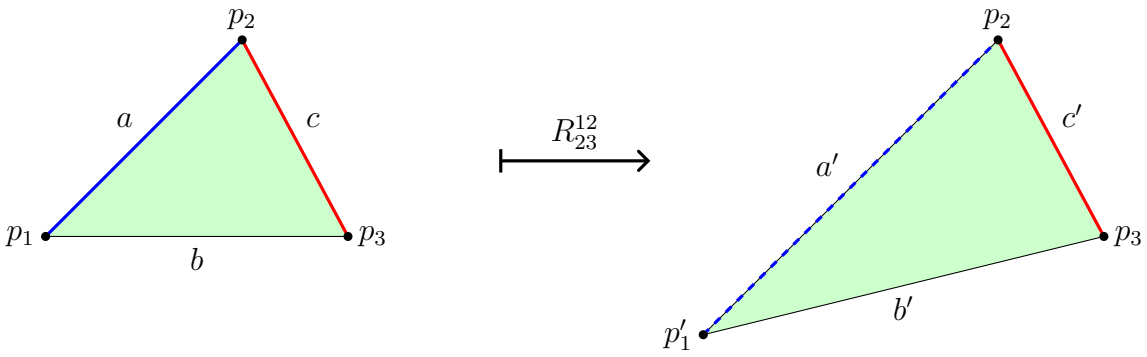

Figure 4.2: An edge reshaping $R = R_{23}^{12}$ that rescales e_{12} by a factor of q and fixes e_{23} (i.e. rescales e_{23} by a factor of 1).

inner products of edges in T where the cofficients do not dependent on the original edge norms. Using Proposition 4.1.6 we can then write these inner products in terms of the original edge norms. Hence, we obtain a'_{ij} as a linear combination of the original edge norms in $\mathbf{v}$ whose coefficients do not depend on $\mathbf{v}$. We construct M using these coefficients.

□

In abuse of notation, we use R to represent both the edge rescaling map and the edge rescaling matrix associated to R. We now give a specific example of an edge rescaling map by demonstrating the rescaling of a triangle.

Proposition 4.2.5 (Edge Rescaling a Triangle)**.** *Let Δ be a labeled euclidean triangle and Δ' the labeled euclidean triangle obtained by the edge rescaling R_{23}^{12}. Then the edge norms of Δ' can be computed from the edge norms of Δ as follows:*

$$a'_{12} = q^2 a_{12}$$

$$a'_{13} = (q^2 - q)a_{12} + qa_{13} + (1 - q)a_{23}$$

$$a'_{23} = a_{23}$$

In particular, we can describe the action of R on the edge norm vector of Δ as a matrix with entries in $\mathbb{Z}[q]$:

$$\mathbf{v}' = \begin{bmatrix} a'_{12} \\ a'_{13} \\ a'_{23} \end{bmatrix} = \begin{bmatrix} q^2 & 0 & 0 \\ q^2 - q & q & 1 - q \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{12} \\ a_{13} \\ a_{23} \end{bmatrix} = R \cdot \begin{bmatrix} a_{12} \\ a_{13} \\ a_{23} \end{bmatrix} = R \cdot \mathbf{v} \qquad (4.1)$$

Proof. To minimize notation, we denote a_{12}, a_{13} and a_{23} by a, b and c, respectively, and add primes for the corresponding norms in Δ'. In the original triangle Δ, we have the following:

$$a = \langle v_{12}, v_{12} \rangle$$

$$b = \langle v_{13}, v_{13} \rangle = \langle v_{12} + v_{23}, v_{12} + v_{23} = a + 2\langle v_{12}, v_{23} \rangle + c$$

$$c = \langle v_{23}, v_{23} \rangle$$

Note in particular that the above equation for b shows that

$$2\langle v_{12}, v_{23}\rangle = b - a - c$$

Since R rescales e_{12} by q and e_{23} by 1, for the edge norms of Δ' we have:

$$a' = \langle qv_{12}, qv_{12}\rangle = q^2\langle v_{12}, v_{12}\rangle = q^2 a$$

$$b' = \langle qv_{12} + v_{23}, qv_{12} + v_{23}\rangle = q^2\langle v_{12}, v_{12}\rangle + 2q\langle v_{12}, v_{23}\rangle + \langle v_{23}, v_{23}\rangle$$

$$= q^2 a + q(b - a - c) + c = (q^2 - q)a + qb + (1 - q)c$$

$$c' = \langle v_{23}, v_{23}\rangle = c$$

□

Notice in Proposition 4.2.5 that the matrix for R not only consists of entries from $\mathbb{Z}[q]$, but in particular these entries are polynomials of degree at most 2. This is not special to R^{12}_{23} and we have the following result.

Proposition 4.2.6 (Quadratic Matrices)**.** *Let $R = R^{\sigma}_{\tau}$ be a q-rescaling of a labeled $(n-1)$-dimensional euclidean simplex Δ. The effect of R on the edge norm vector $\mathbf{v}$ of Δ is realized by multiplying on the left by an N by N matrix with entries in $\mathbb{Z}[q]$ of degree at most 2.*

Proof. Notice from Proposition 4.2.5 that whenever an edge is rescaled by a factor of q, the edge norm of the corresponding edge in Δ' is multiplied by q^2. In

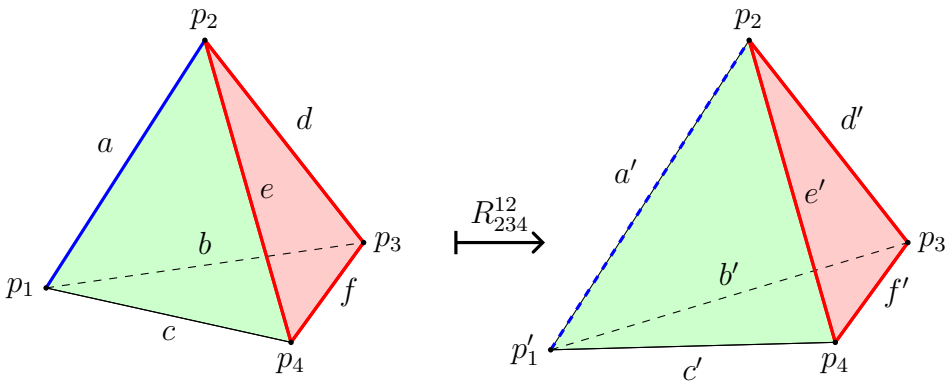

Figure 4.3: The edge rescaling R^{12}_{234} which fixes the triangle Δ_{234} and rescales edge e_{12} by a factor of q.

particular, there is at most one q on each side of any given inner product, and hence any given coefficient is at most quadratic. □

A natural next example to consider is rescaling a tetrahedron, which we demonstrate below.

Example 4.2.7 (Rescaling a Tetrahedron)**.** Consider the edge rescaling R^{12}_{234} shown in Figure 4.3. We can compute all of the new edge norms using Proposition 4.2.5. To minimize notation, we label the edge norms a through f for a_{12} through a_{34} in lexicographic order. The edge rescaling $R = R^{12}_{234}$ can be described as follows:

$$R \cdot \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} q^2 a \\ (q^2-q)a + qb + (1-q)d \\ (q^2-q)a + qc + (1-q)e \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} a' \\ b' \\ c' \\ d' \\ e' \\ f' \end{bmatrix} \tag{4.2}$$

Thus the matrix that encodes the rescaling R is

$$R = \begin{bmatrix} q^2 & 0 & 0 & 0 & 0 & 0 \\ q^2-q & q & 0 & 1-q & 0 & 0 \\ q^2-q & 0 & q & 0 & 1-q & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

We now describe a way of defining the matrices for an edge rescaling when the simplices are higher dimensional.

Definition 4.2.8 (Row Descriptions)**.** As seen above, the new edge norms under an edge rescaling are determined by the rows of the matrix $R = R^{\sigma}_{\tau}$. We now introduce a way to describt these rows. Recall that the e_{ij} represents both an edge in a labeled euclidean simplex Δ and an edge in $\mathbf{D}_n$. We can also think

of the edge e_{ij} as an element of a basis of the vector space $\mathbb{R}^N$ containing the edge norm vectors. For example, the second row of the matrix for $R = R^{12}_{23}$ can be described in the basis e_{12}, e_{13}, e_{23} in Proposition 4.2.5 as the row vector $(q^2 - q, q, 1 - q)$, or equivalently, as the linear combination $(q^2 - q)e_{12} + qe_{13} + (1 - q)e_{23}$. Moreover, to select the second row of the matrix R, we act on the vector $(0, 1, 0)$, which corresponds to e_{13}, from the *right* by R. In other words, $(e_{13})R = q^2 - q)e_{12} + qe_{13} + (1 - q)e_{23}$.

Remark 4.2.9 (Left and Right)**.** Notice that the linear combination that describes the image of a basis vector acted on by an edge rescaling matrix R from the right looks very similar to the corresponding entry in the edge normal vector when acted on by R from the left. This is because both are encoding the entries of one row of R. It is important to keep in mind that we act on edge norm vectors from the left, but when describing the matrices for edge rescalings we define them by acting on basis vectors from the right.

At this point, it should not be surprising that the e_{kl} row of the matrix $R^{ij}_{\mathrm{rc}(ij)}$ only depends on the geometric relationship between the edges e_{ij} and e_{kl} (as described in Definition 3.1.3) in the punctured disc $\mathbf{D}_n$. In particular, we can describe the rows of matrix $R^{ij}_{\mathrm{rc}(ij)}$ based on the positions of the basis vectors e_{kl} relative to e_{ij}. We first do an example.

Example 4.2.10 (Rescaling a Boundary Edge)**.** In this example we give explicit row descriptions for the q-rescalings which stretch a single boundary edge while fixing either its left or its right complement as in Definition 3.2.8. We first compute the row description of the matrix $R = R^{12}_{\mathrm{rc}(12)}$.

$$(e_{kl})R = \begin{cases} q^2 e_{kl} & \text{identical } (k=1, l=2) \\ e_{kl} & \text{noncrossing } (k,l>2) \\ e_{kl} & \text{to the right } (k=2) \\ (q^2-q)e_{12} + qe_{kl} + (1-q)e_{2l} & \text{to the left } (k=1) \end{cases} \tag{4.4}$$

This is a natural generalization of the trianglar and tetrahedral examples in the new notation. The row description of $R = R^{ij}_{\mathrm{rc}(ij)}$ with $j = i+1 \mod n$ is similar, with the added notation of using e_{new} for the third edge of the triangle formed when e_{ij} and e_{kl} have exactly one end point in common. Using the language of Definition 3.1.3, this means e_{kl} is either to the left or to the right of e_{ij}.

$$(e_{kl})R = \begin{cases} q^2 e_{kl} & \text{identical} \\ e_{kl} & \text{noncrossing} \\ e_{kl} & \text{to the right} \\ (q^2-q)e_{ij} + qe_{kl} + (1-q)e_{\mathrm{new}} & \text{to the left} \end{cases} \tag{4.5}$$

Switching from the right complement to the left complement causes only very minor changes. The row description of the matrix $R = R^{12}_{\mathrm{lc}(12)}$ is as follows.

$$(e_{kl})R = \begin{cases} q^2 e_{kl} & \text{identical } (k=1, l=2) \\ e_{kl} & \text{noncrossing } (k,l>2) \\ e_{kl} & \text{to the left } (k=1) \\ (q^2-q)e_{12} + qe_{kl} + (1-q)e_{1l} & \text{to the right } (k=2) \end{cases} \tag{4.6}$$

And finally, we gie the row description of $R^{ij}_{\mathrm{lc}(ij)}$ with $j = i+1 \mod n$, with the same convention that e_{new} denotes the third edge of the triangle when e_{ij} and e_{kl} have exactly one endpoint in common.

$$(e_{kl})R = \begin{cases} q^2 e_{kl} & \text{identical} \\ e_{kl} & \text{noncrossing} \\ e_{kl} & \text{to the left} \\ (q^2-q)e_{ij} + qe_{kl} + (1-q)e_{\mathrm{new}} & \text{to the right} \end{cases} \tag{4.7}$$

Notice that edge rescaling a boundary edge is particularly nice because in this case the basis vector e_{kl} never crosses e_{ij}. In order to extend Example 4.2.10, we need a computation for edge rescalings corresponding to diagonal edges.

Proposition 4.2.11 (Diagonal Edges)**.** *Let Δ be a labeled euclidean tetrahedron and Δ' the labeled euclidean tetrahedron obtained by the edge rescaling $R = R^{24}_{rc(24)}$. The new edge norm a'_{13} can be computed from the edge norms of Δ as follows:*

$$a'_{13} = a_{13} + (q-1)^2 a_{24} + (q-1)(a_{14} + a_{23} - a_{12} - a_{34})$$

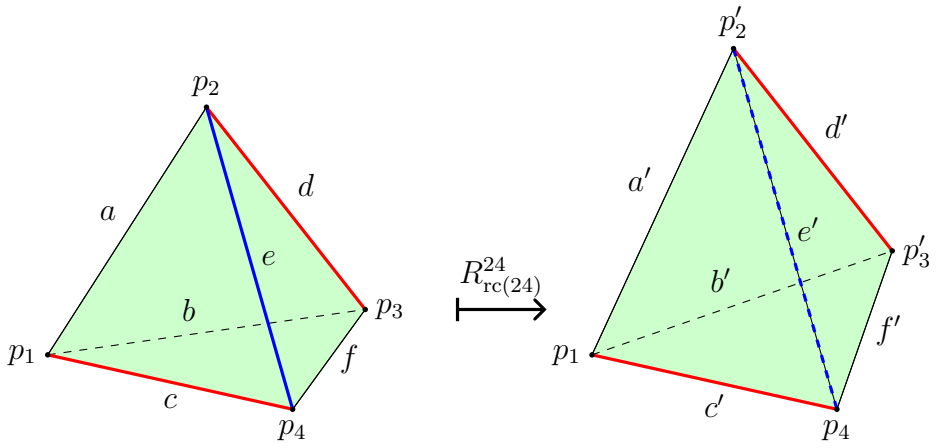

Figure 4.4: The edge rescaling $R^{24}_{\mathrm{rc}(24)}$ which rescales the edge e_{24} while fixing the edges e_{23} and e_{14}.

Proof. Since

$$v_{13} = v_{12} + v_{24} + v_{43},$$

we have

$$v'_{13} = v_{12} + qv_{24} + v_{43}.$$

Expanding, we find

$$a'_{13} = \langle v'_{13}, v'_{13} \rangle = a_{12} + q^2 a_{24} + a_{34} + 2q\langle v_{12}, v_{24} \rangle + 2q\langle v_{24}, v_{43} \rangle + 2\langle v_{12}, v_{34} \rangle.$$

Using Proposition 4.1.6 we have

$$\langle v_{12}, v_{24} \rangle = a_{14} - a_{12} - a_{24},$$

$$\langle v_{24}, v_{43} \rangle = a_{23} - a_{24} - a_{34}$$

and

$$\langle v_{12}, v_{43} \rangle = a_{13} + a_{24} - a_{14} - a_{23}.$$

Substituting and simplifying yields the result. □

Notice that the edge norms involved in the new edge norm a'_{13} obtained by the rescaling $R^{24}_{\mathrm{rc}(24)}$ are precisely those that correspond to the edges in the convex hull of the points p_2, p_4, p_1, p_3. This is true in general, and we describe the general situation below.

Example 4.2.12 (Rescaling a Diagonal Edge)**.** The row description for the general rescaling matrix $R^{ij}_{\mathrm{rc}(ij)}$ is essentially identical to the one in Example 4.2.10 with the added case where e_{ij} and e_{kl} cross. We can then use Proposition 4.2.11 to compute $(e_{kl})R^{ij}_{\mathrm{rc}(ij)}$. In particular, if the clockwise ordering of i, j, k, l is (k, i, l, j) then we have

$$(e_{kl})R^{ij}_{\mathrm{rc}(ij)} = e_{kl} + (q-1)^2 e_{ij} + (q-1)e_{kj} + (q-1)e_{il} + (1-q)e_{ki} + (1-q)e_{lj}.$$

Notice that this agrees with the calculation of a'_{13} in Proposition 4.2.11.

More generally, we can describe $(e_{kl})R^{ij}_{\mathrm{rc}(ij)}$ using the convex hull of the points p_i, p_j, p_k, p_l in $\mathbf{D}_n$. The answer is e_{kl} plus $(q-1)^2 e_{ij}$ plus $(q-1)$ times the two boundary edges of the convex hull which are simultaneously to the right of e_{ij} and to the left of e_{kl} plus $(1-q)$ times the two boundary edges which are simultaneously to the right of e_{ij} and to the left of e_{kl}. Notice again that this description agrees

with the calculation of a'_{13} in Proposition 4.2.11. The row description for $R^{ij}_{\mathrm{lc}(ij)}$ can be found similarly.

Chapter 5

Braid Groups and Euclidean Simplices

In this chapter, we define three explicit representations of the braid group and prove the first of our two main theorems, Theorem A.

5.1 Representations of Braid$_n$

We now discuss three explicit representations of the braid group: the Lawrence-Krammer-Bigelow (LKB) representation, the simplicial representation and the permutation representation. The first representation, and the most complicated, is the LKB representation.

Definition 5.1.1 (LKB representation)**.** Let q and t be nonzero positive real numbers, $\mathcal{E}$ be the set $\{e_{ij}\}$ with $1 \leq i < j \leq n$ of size $N = \binom{n}{2}$ and $\mathbb{R}^N$ be the $N = \binom{n}{2}$-dimensional real vector space with $\mathcal{E}$ as its ordered basis. The *LKB representation* of the braid group is the map $\rho : \textsc{Braid}_n \to GL_N(\mathbb{R})$ defined by the following action (from the right) of the standard braid group generators s_{ij} (with $j = i + 1$ and $1 \leq i < n$) on elements of $\mathcal{E}$.

$$(e_{kl})\rho(s_{ij}) = \begin{cases} tq^2 e_{kl} & i = k, j = l \\ e_{kl} & i, j \notin \{k, l\} \\ e_{jl} & i = k, j < l \\ e_{kj} & i = l \\ t(q^2 - q)e_{ij} + qe_{ki} + (1 - q)e_{kl} & k < i, j = l \\ (q^2 - q)e_{ij} + qe_{il} + (1 - q)e_{kl} & j = k \end{cases} \tag{5.1}$$

We make two remarks about this definition.

Remark 5.1.2 (Left/right Actions)**.** In the literature, this action is written as an action from the left. Our alteration only has the effect of transposing the matrices for the standard generators. We make this change in order to match up the the matrices of the representation with the obviously very similar edge rescaling matrices discussed in the previous section.

Remark 5.1.3 (The Sign of the t-Variable)**.** The t variable depends on the linear ordering of the vertices and is associated to the standard presentation of the

braid group. The presence of t obscures the fundamentally cyclically symmetric nature of the dependence on q. To highlight this cyclic symmetry, we consider the specialization of the LKB representation with $t = 1$. We also note that there are inconsistencies in the literature regarding the sign of t. The variable t in [14] corresponds to $-t$ in [4] (with an additional sign correction in [3]) and [12]. We have written the LKB representation using Krammer's sign convention. If we had followed Bigelow's we would be setting t equal to -1.

In light of our first main theorem, we call the simplified LKB representation the *simplicial representation* of the braid group.

Definition 5.1.4 (Simplicial Representation)**.** The *simplicial representation* of the braid group is a specialization of the LKB representation with t set equal to 1. Concretely, let q be a nonzero positive real number, $\mathcal{E}$ be the set $\{e_{ij}\}$ with $1 \leq i < j \leq n$ in lexicographic order and $\mathbb{R}^N$ be the $N = \binom{n}{2}$-dimensional real vector space with $\mathcal{E}$ as its ordered basis. The *simplicial representation* of the braid group is defined by the following action (from the right) of the standard braid group generators s_{ij} (with $j = i + 1$ and $1 \leq i < n$) on elements of $\mathcal{E}$. We write S_σ for the matrix that represents s_σ with respect to the ordered basis $\mathcal{E}$ and we have introduced the notation e_{new} to denote the third side of the triangle when

e_{ij} and e_{kl} have exactly one endpoint in common as in the previous section.

$$(e_{kl})S_{ij} = \begin{cases} q^2 e_{kl} & i = k, j = l \\ e_{kl} & i, j \notin \{k, l\} \\ e_{\text{new}} & i = k, j < l \\ e_{\text{new}} & i = l \\ (q^2 - q)e_{ij} + qe_{\text{new}} + (1 - q)e_{kl} & k < i, j = l \\ (q^2 - q)e_{ij} + qe_{\text{new}} + (1 - q)e_{kl} & j = k \end{cases} \tag{5.2}$$

Now that the t variable has been eliminated, some of the rows are identical and they can be rewritten using the language of Definition 3.1.3.

$$(e_{kl})S_{ij} = \begin{cases} q^2 e_{kl} & \text{identical} \\ e_{kl} & \text{noncrossing} \\ e_{\text{new}} & \text{to the left} \\ (q^2 - q)e_{ij} + qe_{\text{new}} + (1 - q)e_{kl} & \text{to the right} \end{cases} \tag{5.3}$$

The third representation is obtained by also eliminating the q variable.

Definition 5.1.5 (Permutation representation)**.** The *permutation representation* of the braid group that we are interested in is the one obtained from the simplicial representation by setting $q = 1$ (or both $t = q = 1$ in the LKB representation). This describes the permutation of the edges induced by the corresponding permutation of the vertices. We write P_σ for the matrix corresponding to s_σ. Its row

description is as follows.

$$(e_{kl})P_{ij} = \begin{cases} e_{kl} & \text{identical, crossing or noncrossing} \\ e_{\text{new}} & \text{to the left or right} \end{cases} \tag{5.4}$$

At this point it should be clear that the simplicial representation matrix S_{ij} is nearly identical to the edge rescaling matrix $R^{ij}_{\text{rc}(ij)}$ given in Example 4.2.10. The difference is the permutation matrix P_{ij}.

5.2 Braid Groups Act on Euclidean Simplices

In this section, we connect the simplicial representation of the previous section with the edge rescaling matrices the previous chapter. We begin with an example.

Example 5.2.1 (Geometry of S_{12})**.** The matrices corresponding to the first standard generator s_{12} of the four string braid group in the simplicial representation and the permutation representation are as follows:

$$S_{12} = \begin{bmatrix} q^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ q^2-q & q & 0 & 1-q & 0 & 0 \\ q^2-q & 0 & q & 0 & 1-q & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad P_{12} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.5}$$

It is straightforward to check that $S_{12} = P_{12}R^{12}_{234} = R^{12}_{134}P_{12}$. The matrix $R^{12}_{234} = R^{12}_{\mathrm{rc}(12)}$ is listed explicitly in Example 4.2.7 and the row description of $R^{12}_{134} = R^{12}_{\mathrm{lc}(12)}$ is given in Example 4.2.10.

For ease of proving our main results, we state one final definition.

Definition 5.2.2 (Relabeling and Rescaling)**.** Let σ be a noncrossing permutation in SYM_n and S_σ the explicit matrix representing s_σ in the simplicial representation. We say that S_σ *relabels and rescales* if

$$S_\sigma = P_\sigma R^\sigma_{\mathrm{rc}(\sigma)} = R^\sigma_{\mathrm{lc}(\sigma)} P_\sigma$$

where $\mathrm{rc}(\sigma)$ and $\mathrm{lc}(\sigma)$ are the left and right complements of σ as in Definition 3.2.8.

In light of Definition 4.2.1 and Definition 5.1.5, we can describe Definition 5.2.2 geometrically. The matrix S_σ relabels and rescales if the effect it has on labeled euclidean simplices is to first rescale the edges by fixing those in the right complement of σ and multiplying the edges in σ by a factor of q, then relabel the edges according to the edge relabeling induced by P_σ. Alternatively, the edge relabeling P_σ can be performed first and the edges that are fixed are those in the left complement as opposed to the right complement. In this language, Example 5.2.1 shows that the matrix S_{12} in the simplicial representation of BRAID_4 relabels and rescales. In fact, the matrix representation under the simplicial representation of each stanrdard generator of the braid group relabels and rescales.

Proposition 5.2.3 (Standard generators)**.** *For every standard generator s_{ij} of the braid group* $\textsc{Braid}_n$*, the corresponding matrix S_{ij} in the simplicial representation relabels and rescales.*

Proof. This is essentially immediate once we compare the row descriptions for the matrix S_{ij} in Definition 5.1.4 with the row descriptions for the edge rescalings $R^{ij}_{\mathrm{rc}(ij)}$ and $R^{ij}_{\mathrm{lc}(ij)}$ in Example 4.2.10. In particular, we note that multiplying on the left by P_{ij} switches the rows to the left of e_{ij} with the rows to the right of e_{ij} while multiplying by P_{ij} on the right permtues the columns and thus the subscripts on the edges that occur in the various terms of the row descriptions.

□

We can now use Proposition 5.2.3 to deduce our first result.

Theorem A (Braids reshape simplices)**.** *The simplicial representation of the n-string braid group preserves the set of $\binom{n}{2}$-tuples of positive reals that represent the squared edge lengths of a nondegenerate euclidean simplex with n labeled vertices.*

Proof. First note that it suffices to prove that this holds for some generating set of $\textsc{Braid}_n$. Next, both vertex relabelings and edge rescalings clearly preserve the set of $\binom{n}{2}$-tuples that describe the squared edge lengths of a nondegenerate euclidean simplex with n labeled vertices, so Proposition 5.2.3 completes the proof. □

5.3 Dual Simple Braids Act by Rescaling and Relabeling

By Theorem A, we know that the dual simple braids as in Definition 3.2.2 corresponding to the standard generators relabel and rescale. In this section, prove our second main result that *all* dual simple braids relabel and rescale. First, we have the following result.

Proposition 5.3.1 (Products)**.** *Let σ_1 and σ_2 be noncrossing permutations in* $\textsc{Sym}_n$ *such that s_{σ_1}, s_{σ_2} and $s_{\sigma_1\sigma_2}$ are dual simple braids. If both S_{σ_1} and S_{σ_2} relabel and rescale, then $S_{\sigma_1\sigma_2}$ relabels and rescales.*

Proof. By Definition 3.2.10, we know that there exist noncrossing permutations σ_3, σ_4 and σ_5 such that

$$\delta = \sigma_1\sigma_2\sigma_3 = \sigma_1\sigma_4\sigma_2 = \sigma_5\sigma_1\sigma_2.$$

We then have the following equalities for $S_{\sigma_1\sigma_2} = S_{\sigma_1}S_{\sigma_2}$:

$$\begin{aligned} S_{\sigma_1}S_{\sigma_2} &= (P_{\sigma_1}R^{\sigma_1}_{\sigma_4\sigma_2})(R^{\sigma_2}_{\sigma_1\sigma_4}P_{\sigma_2}) \\ &= P_{\sigma_1}R^{\sigma_1\sigma_2}_{\sigma_4}P_{\sigma_2} \\ &= P_{\sigma_1}P_{\sigma_2}R^{\sigma_1\sigma_2}_{\sigma_3} \\ &= R^{\sigma_1\sigma_2}_{\sigma_5}P_{\sigma_1}P_{\sigma_2} \end{aligned}$$

The first equality uses the hypotheses that S_{σ_1} and S_{σ_2} relabel and rescale. The second rewrites the product $R^{\sigma_1}_{\sigma_4\sigma_2}R^{\sigma_2}_{\sigma_1\sigma_4}$ as a single edge rescaling. This edge rescaling can be desribed as follows. The edge rescaling $R^{\sigma_2}_{\sigma_1\sigma_4}$ rescales the edges in σ_2 by a factor of q and holds the edges in the product $\sigma_1\sigma_4$ fixed. In particular, the edges in σ_4 and σ_2 are fixed. The edge rescaling $R^{\sigma_1}_{\sigma_4\sigma_2}$ rescales the edges in σ_1 by a factor of q and holds the edges in the product $\sigma_1\sigma_4$ fixed. In particular, the edges in σ_1 and σ_4 are fixed. The result of the product is then to rescale the edges in the product $\sigma_1\sigma_2$ by a factor of q, leaving the edges in σ_4 fixed. Hence the second equality. The third and fourth equalities are a result of conjugating the points involved by appropriate permutation matrices.

Since $P_{\sigma_1}P_{\sigma_2} = P_{\sigma_1\sigma_2}$ and σ_3 and σ_5 are the left and right complements of $\sigma_1\sigma_2$, respectively, this completes the proof.

□

Note that the above equalities could be arranged to show that if any two of S_{σ_1}, S_{σ_2} and $S_{\sigma_1\sigma_2}$ relabel and rescale then so does the third. Propositions 5.2.3 and 5.3.1 do not immediately prove our main result because not all dual simple braids are a product of two standard generators. We need to extend Proposition 5.2.3 to the full set of dual generators.

Proposition 5.3.2 (Dual generators)**.** *For every dual generator $s_{ij} \in \text{GEN}_n$ of the braid group BRAID_n, the corresponding matrix S_{ij} in the simplicial representation relabels and rescales.*

Proof. An explicit description of the matrix for s_{ij} under the LKB representation is given in Krammer's earlier paper [14]. If we set $t = 1$ in that description, we find that the matrix S_{ij} has the exact same description as it does when $j = i + 1$ as given in Definition 5.1.4 except that a new case must be added that gives the result of $(e_{kl})S_{ij}$ when e_{ij} and e_{kl} cross. This simplification of the answer listed in [14] agrees with the corresponding row of $P_{ij}R^{ij}_{\text{rc}(ij)}$, which is the same as the corresponding row of $R^{ij}_{\text{lc}(ij)}P_{ij}$ obtained by combining Example 4.2.12 and Definition 5.1.5. □

The second main result is an immediate corollary of Proposition 5.3.2.

Theorem B (Dual simple braids relabel and rescale)**.** *Under the simplicial representation of the braid group, each dual simple braid relabels and rescales. Concretely, for each $\sigma \in NC_n$, we have $S_\sigma = P_\sigma R^\sigma_{rc(\sigma)} = R^\sigma_{lc(\sigma)}P_\sigma$.*

Proof. Proposition 5.3.2 shows this is true for the dual generators of BRAID_n. Proposition 5.3.1 and induction extend this fact to all dual simple braids. □

Chapter 6

Origins and Future Work

In this final chapter we make a few remarks about the origins of these results as well as some directions for future work.

6.1 Origins

As stated in the Introduction, three papers on linearity of the braid groups appeared in the early 2000s. The results of these papers were extended to other types of Artin groups by Cohen-Wales [9], Digne [10] and Paris [21]. Each of these papers proves linearity of the positive monoid. When the Artin group is spherical, the group is the group of fractions of the positive monoid and hence linearity of the monoid implies linearity of the group.

For general Artin groups, it is known that the monoid generated by the standard minimal generating set is not large enough for the Artin group to be the group of fractions. However, the positive monoid generated by the dual generating set may be large enough. Thus it would be advantageous to be able to prove linearity of the braid groups using the dual generators, as in Krammer's proof for $\textsc{Braid}_4$ [14]. A proof based on the dual generating set, focusing on the q-variable, may lead to generalizations similar to those of Cohen and Wales [9], Digne [10] and Paris [21].

We first wrote code in `sage` to investigate the properties of the LKB matrices and found, experimentally, nice ways to decompose them and we began to isolate the changes that each factor was making. The interpretation of the simplified version as modifications of edge norms of simplices was one of the final steps in our evolving understanding of these representations. We have included the code used to aid future invesetigations in the appendix.

6.2 Future Work

We conclude this dissertation with three directions for additional research.

Remark 6.2.1 (Linearity)**.** The simplicial representation is not faithful for $n \geq 5$ because it is the symmetric tensor square of the Burau representation [14]. In order

to establish a new proof of linearity utilizing the q variable, the t variable needs to remain guitably generic. One project is to extend the geometric interpretation given in this dissertation so that the t variable can remain generic, and to extend the interpretation give here as part of a new proof of braid group linearity focused on the q variable. In particular, one should try to prove that the dual positive monoid generated by the dual generators acts faithfully by using a ping-pong argument similar to the one in Krammer's first paper [14] proving linearity of $\textsc{Braid}_4$.

Remark 6.2.2 (Dual Garside Length)**.** The dual Garside structure of the braid group discussed in [5] equips the braid group with a way to define a normal form for dual braids. One of the facts conjectured by Karmmer in [14] is that the highest power of q in the LKB representation of a dual positive braid is twice its dual Garside length. Recently, Tetsuya Ito and Bert Wiest posted an article proving this fact [12]. An idea is to use the geometric understanding of the q variable presented in this dissertation to reprove this fact in a more elementary way.

Remark 6.2.3 (Spherical Artin groups)**.** The set of labeled euclidean simplices, with dilated simplices identified, is one of the standard parameterizations of the higher rank symmetric space $SL(V)/SO(V)$ and the simplicial representation appears to act on this space by isometries. Once this action is made explicit, it

should be possible to define a similar construction and to give a similar interpretation for all of the spherical Artin groups once the focus on labeled euclidean simplices is replaced by linear transformations of root systems.

Appendix

Here we will present the sage code used to investigate the simplicial representation, as well as display some examples of what it does. The entirety of the code is as follows.

```
#Simplicial Representation
#Elizabeth Leyton Chisholm and Jon McCammond

### helper functions ###
import os    # for system('clear')
import numpy # for cumsum
import sys   # for flush and write
import itertools # for creating ncp_list quickly

flush = sys.stdout.flush
write = sys.stdout.write

def initialize():
    os.system('clear');
    print "Simplicial representation "
    print "Start with create_atoms(n)"
    print

def nrange(n):
    "returns the first n counting numbers"
    return range(1,n+1)
```

```
def excess(li):
    "used for absolute reflection length"
    return sum([len(i)-1 for i in li])

def distinct(li):
    "returns True if the set of permutations is distinct"
    return all([li.count(i)==1 for i in set(li)])

def cycle(li):
    "turns a list into a simple cycle"
    return PermutationGroupElement([tuple(li)])

def create_dictionaries():
    """ Creates two global dictionaries:
    pa_d turns a pair into an atom (2-tuples to transpositions)
    ta_d turns a tuple into a set of atoms """
    global pa_d,ta_d
    pa_d=dict([[tuple(i),cycle(i)] for i in Combinations(strings,2)])
    def tup_atoms(tu): return frozenset([pa_d[tuple(i)] for i in
    Combinations(tu,2)])
    ta_d=dict([[tuple(i),tup_atoms(i)] for i in subsets(strings)
    if len(i)>1])

def ar_len(g):
    "absolute reflection length of the permutation g"
    return excess(g.cycle_tuples())

def ar_atoms(g):
    "set of reflections below g"
    return set().union(*[ta_d[i] for i in g.cycle_tuples()])

def ar_interval(g):
    "ar_interval(g) returns the permutations between id and g
    sorted by reflection length as a set of frozen sets"
    ans = [frozenset([g])]
    for i in range(ar_len(g)):
        ans.append(frozenset([g*i for g in ans[-1]
        for i in ar_atoms(g)]))
    ans.reverse()
    return ans
```

```
def create_rc_perm(f):
    "returns the right complement of the permutation f"
    return f^-1*delta

def create_lc_perm(f):
    "returns the left complement of the permutation f"
    return delta*f^-1

def create_compl_d():
    "creates a dictionary for the right complements and
    left complements for all noncrossing permutations"
    global rcp_d, lcp_d
    rcp_d = dict([[f,create_rc_perm(f)] for f in ncp])
    lcp_d = dict([[f,create_lc_perm(f)] for f in ncp])

def lgnf_perm_pair(i,j):
    "returns True if the (ncp[i], ncp[j]) forms a normal form pair"
    a=ar_atoms(ncp[j])
    b=ar_atoms(rcp_d[ncp[i]])
    return a.intersection(b)==set([])

def create_perm_nfl():
    "creates pairs (i,j) where ncp[i]ncp[j] is in normal form"
    global perm_nfl,nfl
    perm_nfl= [(i,j) for i in range(1,len(ncp)-1)
    for j in range(1,len(ncp)-1) if lgnf_perm_pair(i,j)]
    nfl = sorted([(j-1,i-1) for (i,j) in perm_nfl])

def set_strings(sn):
    "set the number of strings and do various precomputations"
    global strings, n_strings, delta, nc_part, nc_part_l,
    ncp, n_simples
    strings = nrange(sn)
    n_strings = len(strings)
    create_dictionaries()
    #delta is the noncrossing permutation (1, 2, ... , n):
    delta = cycle(strings)
    #the noncrossing permutationss between id and delta
    ordered by reflection length as type frozenset:
```

```
    nc_part = ar_interval(delta)
    #the list form of nc_part:
    nc_part_l = [list(i) for i in nc_part]
    #list of noncrossing permutations:
    ncp=list(itertools.chain.from_iterable(nc_part_l))
    #creates the left and right complements for the ncp's:
    create_compl_d()
    #creates the normal form pairs of length 2 for the ncp's:
    create_perm_nfl()
    n_simples = len(ncp)
    print " number of strings = {}".format(n_strings)

def poly_coeff(poly,k):
    "returns the coefficient of q^k in the polynomial poly"
    if poly==0: return 0
    c=poly.coeffs()
    if k<len(c): return c[k]
    return 0

def matrix_coeffs(mat,k):
    "returns the matrix coefficient for q^k"
    return matrix([[poly_coeff(i,k) for i in j] for j in mat])

def gen_split(mat):
    "returns the constant matrix, the coefficient matrix of q,
    and the coefficient matrix of q^2 of mat as a list"
    return [matrix_coeffs(mat,i) for i in range(3)]

def gen_perm(mat):
    "returns the sum of the matrices in gen_split(mat)"
    return sum(gen_split(mat))

def perm_l(mat):
    "strips the permutation off from the left of mat"
    return [gen_perm(mat)^-1*i for i in gen_split(mat)]
def perm_r(mat):
    "strips the permutation off from the right of mat"
    return [i*gen_perm(mat)^-1 for i in gen_split(mat)]

#defines Rq as the univariate polynomial Ring in q
```

```
over the rationals
Rq.<q>=QQ[]

def next_string(i,sn):
    "returns the next puncture clockwise from the puncture i"
    j=mod(i+1,sn).lift()
    if j==0: j=j+sn
    return j

def admissible(i,j,k):
    "returns True if an admissible partition"
    if not distinct([i,j,k]): return False
    return cycle(sorted([i,j,k]))==cycle([j,k])*cycle([i,j])

def std_gen(i,sn):
    "output is the simplicial representation matrix r_i,i+1"
    #creates the edges
    edgs=[cycle([x,y]) for x in nrange(sn) for y in nrange(sn)
    if x!=y and x < y]
    #number of edges
    eN=binomial(sn,2)
    #ans is a matrix over Rq of dimension eN by eN:
    ans=matrix(Rq,eN)
    j=next_string(i,sn)
    f=cycle([i,j])
    a=edgs.index(f)
    #creates the simplicial representation of r_i,i+1:
    for k in range(1,sn+1):
        if admissible(i,j,k):
            b=edgs.index(cycle([j,k]))
            c=edgs.index(cycle([k,i]))
            ans[a,a]=q^2
            ans[b,a]=q^2-q
            ans[b,b]=1-q
            ans[b,c]=q
            ans[c,b]=1
    for d in [edgs.index(g) for g in edgs if (f*g==g*f) and (f<>g)]:
        ans[d,d]=1
    # this line only works between forks cannot cross boundary forks
    return ans
```

```
def std_gens(n):
    "creates the simplicial representations for the standard
    generators"
    return [std_gen(i,n) for i in range(1,n)]

def create_std_gens(sn):
    "creates the simplicial representation matrices for the
    standard generators"
    global Id,Z,ao
    global StdGen,Delta,D
    global edges,eNum,enames,valpha
    #creates the edges as tuples:
    edges = [(x,y) for x in range(1,sn+1) for y in range(1,sn+1)
    if x!=y and x < y]
    #creates the edges as cycles - and thus unordered:
    edges_c = [cycle([i,j]) for (i,j) in edges]
    eNum = len(edges)
    Ra.<a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z>=SR[]
    alphabet=[a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z]
    valpha=vector(alphabet[:eNum])
    enames=[10*i+j for (i,j) in edges]
    ao=vector([1 for i in range(eNum)])
    Id = matrix.identity(eNum)
    Z = matrix.zero(eNum,eNum)
    #the set of simplicial representations for the standard
    generators:
    StdGen = std_gens(sn)
    #simplicial representation of delta:
    Delta = prod(StdGen)
    D = gen_perm(Delta)

def lc(mat):
    "Left complement of mat wrt Delta"
    [a,b,c]=gen_split(mat)
    pt=transpose(sum([a,b,c]))
    return D*pt*(q^2*a+q*b+c)*pt

def rc(mat):
    "Right complement of mat wrt Delta"
```

```
    [a,b,c]=gen_split(mat)
    pt=transpose(sum([a,b,c]))
    return pt*(q^2*a+q*b+c)*pt*D

def create_boundary_cycles():
    "creates a boundary cycle dictionary"
    global bcd
    bcd=dict([[(i,j),create_bc(i,j)] for (i,j) in edges])

def create_bc(i,j):
    "makes the matrix for the boundary cycle r_i..j"
    return prod([StdGen[i-1+k] for k in range(j-i)])

def bc(i,j):
    "returns the boundary cycle matrix"
    if j>i: return bcd[(i,j)]
    return Id

def create_atom(i,j):
    "returns the atom r_ij by computing left and right
    complements of boundary cycles"
    return lc(bc(i,j-1)*rc(bc(i,j)))

def create_atom_d():
    "creates a dictionary for the cycles of the form (ij)"
    global atd
    atd=dict([[(i,j),create_atom(i,j)] for (i,j) in edges])

def atom(i,j):
    "returns the precomputed atom r_ij"
    if j>i: return atd[(i,j)]
    return Id

def split_tup(l):
    "splits a list into its adjacent pairs"
    return [l[i:i+2] for i in range(len(l)-1)]

def tup2mat(l):
    "turns a tuple (with ordered entries) into a matrix "
    return prod([atom(i,j) for [i,j] in split_tup(l)])
```

```
def perm2mat(p):
    "turns a permutation into a matrix"
    return prod([tup2mat(c) for c in p.cycle_tuples()])

def tuples_inside(g):
    "returns a LIST of the edges inside the simple as a tuple"
    return [p.cycle_tuples()[0] for p in ar_atoms(g)]

def ncp_graph_rc(g):
    "makes a graph out of the simple g with edges the tuples
    inside g and in the right complement"
    return Graph(tuples_inside(g)
    + tuples_inside(create_rc_perm(g)))

def ncp_graph_lc(g):
    "makes a graph out of the simple g with edges the tuples
    inside g and in the left complement"
    return Graph(tuples_inside(g)
    + tuples_inside(create_lc_perm(g)))

def check_below(g,(i,j),(k,l)):
    "checks if one edge is below another in the hypertree"
    if i in g.shortest_path(k,l) and j in g.shortest_path(k,l):
        return 1
    else:
        return 0

def zeta(g):
    "makes the zeta matrix for a given hypertree"
    ans = matrix(Rq,eNum)
    for i in range(0,eNum):
        for j in range(0,eNum):
            ans[i, j] = check_below(g, edges[j], edges[i])
    return ans

def create_zetas():
    "makes the zeta functions for the ncps using both right
    (zeta_rcs) and left (zeta_lcs) complement"
    global zeta_rcs, zeta_lcs
```

```
    zeta_rcs = []
    for i in range(1, len(nc_part)-1):
        zeta_rcs = zeta_rcs + [zeta(ncp_graph_rc(g))
       for g in nc_part[i]]
    zeta_lcs = []
    for i in range(1, len(nc_part)-1):
        zeta_lcs = zeta_lcs + [zeta(ncp_graph_lc(g))
        for g in nc_part[i]]

def create_atoms(sn):
    """creates the simplicial representation matrices for
    the noncrossing partitions, as well as
    other matrices that may be useful"""
    global rr,ss,cc,pp,cc_pl,cc_pr,vv
    set_strings(sn)
    create_std_gens(sn)
    create_boundary_cycles()
    create_atom_d()
    print "creating zetas"
    create_zetas()
    print "creating rr"
    #rr is the simplicial representation matrices for the
    noncrossing permutations (excluding identity):
    rr = []
    for i in range(1,len(nc_part)-1):
        print " level {} created".format(i)
        rr = rr+[perm2mat(r) for r in nc_part[i]]
    print "creating ss"
    #ss records gen_split(r) for each matrix in rr:
    ss = [gen_split(r) for r in rr]
    print "creating cc"
    cc = [c for [a,b,c] in ss]
    print "creating pp"
    #pp is the permutation representation matrices for each
    noncrossing permutation:
    pp = [sum([a,b,c]) for [a,b,c] in ss]
    print "creating cc_pr"
    #cc_pr is the set of c's with the permutation part stripped
    off from the right:
    cc_pr = [cc[i]*transpose(pp[i]) for i in range(len(cc))]
```

```
    print "creating cc_pl"
    #cc_pl is the set of c's with the permutation part stripped
    off from the left:
    cc_pl = [transpose(pp[i])*cc[i] for i in range(len(cc))]

initialize()
```

We will now demonstrate a few of examples of how this code can be used. The examples will be computed using $\textsc{Braid}_4$ as this is the first interesting case and the matrices are a manageable size.

The first use of this code is to generate the matrices for the simplicial representation. The function `create_atoms(4)` will generate the simplicial representation matrices for $\textsc{Simp}_4$, as well as other useful matrices.

```
sage: create_atoms(4)
 number of strings = 4
creating zetas
creating rr
 level 1 created
 level 2 created
creating ss
creating cc
creating pp
creating cc_pr
creating cc_pl
```

As an example, we will display all of the matrices corresponding to s_{12}. First we have the simplicial representation S_{12}:

```
sage: rr[0]
[    q^2       0       0       0       0       0]
[      0       0       0       1       0       0]
[      0       0       0       0       1       0]
[q^2 - q       q       0  -q + 1       0       0]
[q^2 - q       0       q       0  -q + 1       0]
[      0       0       0       0       0       1]
```

The code also creates a list of matrices for each simple element corresponding to the permutation representation, the matrix of coefficients of q and the matrix of coefficients of q^2, in that order. For S_{12} we have

```
sage: ss[0]
[
[0 0 0 0 0 0]  [ 0  0  0  0  0  0]  [1 0 0 0 0 0]
[0 0 0 1 0 0]  [ 0  0  0  0  0  0]  [0 0 0 0 0 0]
[0 0 0 0 1 0]  [ 0  0  0  0  0  0]  [0 0 0 0 0 0]
[0 0 0 1 0 0]  [-1  1  0 -1  0  0]  [1 0 0 0 0 0]
[0 0 0 0 1 0]  [-1  0  1  0 -1  0]  [1 0 0 0 0 0]
[0 0 0 0 0 1], [ 0  0  0  0  0  0], [0 0 0 0 0 0]
]
```

The lists `cc` and `pp` collect the coefficient matrices of q^2 and the permutation parts of the matrices, respectively. The lists `cc_pr` and `cc_pl` remove the permutation part of the `cc` matrices from the left and right, respectively. So for S_{12} we have

```
sage: cc[0]
[1 0 0 0 0 0]
[0 0 0 0 0 0]
[0 0 0 0 0 0]
```

```
[1 0 0 0 0 0]
[1 0 0 0 0 0]
[0 0 0 0 0 0]
sage: pp[0]
[1 0 0 0 0 0]
[0 0 0 1 0 0]
[0 0 0 0 1 0]
[0 1 0 0 0 0]
[0 0 1 0 0 0]
[0 0 0 0 0 1]
sage: cc_pl[0]
[1 0 0 0 0 0]
[1 0 0 0 0 0]
[1 0 0 0 0 0]
[0 0 0 0 0 0]
[0 0 0 0 0 0]
[0 0 0 0 0 0]
sage: cc_pr[0]
[1 0 0 0 0 0]
[0 0 0 0 0 0]
[0 0 0 0 0 0]
[1 0 0 0 0 0]
[1 0 0 0 0 0]
[0 0 0 0 0 0]
```

The code also creates a vector to represent the squared edge lengths of a simplex:

```
sage: valpha.column()
[a]
[b]
[c]
[d]
[e]
[f]
```

This allows us to see the effect of S_{12} on a tetrahedron with squared edge lengths a, b, c, d, e, f.

```
sage: rr[0]*valpha.column()
[                             q^2*a]
[                                 d]
[                                 e]
[((q - 1)*q)*a + q*b + (-q + 1)*d]
[((q - 1)*q)*a + q*c + (-q + 1)*e]
[                                 f]
```

Of course, it was not clear initially that the vector `valpha` should represent the squared edge lenths of a tetrahedron. However, the investigation of the action of simplicial representation matrices on `valpha` led to this discovery.

One property of the simplicial representation that was not addressed in the dissertation but is interesting and possibly useful concerns the diagonalization of the simplicial representation matrices. To discuss this, we first need to explain the construction of the lists `zeta_rcs` and `zeta_lcs`. Given a dual simple s_σ, we can construct a graph of the vertices of a euclidean simpex Δ by considering the edges in σ and $\mathrm{rc}(\sigma)$ or $\mathrm{lc}(\sigma)$. The code constructs these graphs using the function `ncp_graph_lc` and `ncp_graph_lc`. For s_{12} we would get

```
sage: g = ncp[1]
sage: l = ncp_graph_lc(g)
sage: l
Graph on 4 vertices
sage: l.edges()
[(1, 2, None), (2, 3, None), (2, 4, None), (3, 4, None)]
sage: r = ncp_graph_rc(g)
sage: r
```

```
Graph on 4 vertices
sage: r.edges()
[(1, 2, None), (1, 3, None), (1, 4, None), (3, 4, None)]
```

where `l` and `r` are the graphs using the left and right complement, respectively.

The code then constructs the zeta functions for each of these graphs and collects them in the lists `zeta_lcs` (for graphs using left complements) and `zeta_rcs` (for graphs using right complements). Given a graph corresponding to a simple element and its left or right complement, we create a partial order on the edges in the graph by defining the edge $e_{ij} \leq e_{kl}$ if the unique shortest path connecting the vertices k and l contains the shortest path connecting i and j. The zeta functions are then constructing by recording these relationships in a matrix. For s_{12} we would have

```
sage: zeta_lcs[0]
[1 0 0 0 0 0]
[1 1 0 1 0 0]
[1 0 1 0 1 0]
[0 0 0 1 0 0]
[0 0 0 0 1 0]
[0 0 0 0 0 1]
sage: zeta_rcs[0]
[1 0 0 0 0 0]
[0 1 0 0 0 0]
[0 0 1 0 0 0]
[1 1 0 1 0 0]
[1 0 1 0 1 0]
[0 0 0 0 0 1]
```

The interesting fact about these matrices is that the zeta functions actually diagonalize the edge rescaling maps corresponding to each simple element. For

example, if we strip off the permutation from the left of S_{12} to obtain R^{12}_{134} and conjugate by the zeta function corresponding to the left complement graph of s_{12}, `zeta_lcs[0]` we have the following:

```
sage: zeta_lcs[0].inverse()*pp[0].inverse()*rr[0]*zeta_lcs[0]
[q^2   0   0   0   0   0]
[  0   q   0   0   0   0]
[  0   0   q   0   0   0]
[  0   0   0   1   0   0]
[  0   0   0   0   1   0]
[  0   0   0   0   0   1]
```

Similarly, if we strip off the permutation from the right of S_{12} to obtain R^{12}_{234} and conjugate by the zeta function corresponding to the right complement graph of s_{12}, `zeta_rcs[0]`, we have:

```
sage: zeta_rcs[0].inverse()*rr[0]*pp[0].inverse()*zeta_rcs[0]
[q^2   0   0   0   0   0]
[  0   1   0   0   0   0]
[  0   0   1   0   0   0]
[  0   0   0   q   0   0]
[  0   0   0   0   q   0]
[  0   0   0   0   0   1]
```

This property holds for all of the simple elements. Moreover, even in the original LKB representation with the t variable included and these matrices still diagonalize the LKB representations of the simple elements. This may help in extending the geometric interpretation of the simplicial representation to the LKB representation.

Bibliography

[1] Emil Artin. Theorie der Zöpfe. *Abh. Math. Sem. Univ. Hamburg*, 4(1):47–72, 1925.

[2] Stephen Bigelow. The Burau representation is not faithful for $n = 5$. *Geom. Topol.*, 3:397–404, 1999.

[3] Stephen Bigelow. The Lawrence-Krammer representation. In *Topology and geometry of manifolds (Athens, GA, 2001)*, volume 71 of *Proc. Sympos. Pure Math.*, pages 51–68. Amer. Math. Soc., Providence, RI, 2003.

[4] Stephen J. Bigelow. Braid groups are linear. *J. Amer. Math. Soc.*, 14(2):471–486 (electronic), 2001.

[5] Joan Birman, Ki Hyoung Ko, and Sang Jin Lee. A new approach to the word and conjugacy problems in the braid groups. *Adv. Math.*, 139(2):322–353, 1998.

[6] Noel Brady and Jon McCammond. Factoring euclidean isometries. Available at `arXiv:1312.7780 [math.GR]`.

[7] Werner Burau. Über Zopfinvarianten. *Abh. Math. Sem. Univ. Hamburg*, 9(1):117–124, 1933.

[8] Arjeh M. Cohen and David B. Wales. Linearity of Artin groups of finite type. *Israel J. Math.*, 131:101–123, 2002.

[9] Arjeh M. Cohen and David B. Wales. Linearity of Artin groups of finite type. *Israel J. Math.*, 131:101–123, 2002.

[10] François Digne. On the linearity of Artin braid groups. *J. Algebra*, 268(1):39–57, 2003.

[11] Benson Farb and Dan Margalit. *A primer on mapping class groups*, volume 49 of *Princeton Mathematical Series*. Princeton University Press, Princeton, NJ, 2012.

[12] Tetsuya Ito and Bert Wiest. Lawrence-Krammer-Bigelow representations and dual Garside length of braids. Available at `arXiv:1201.0957 [math.GR]`.

[13] Christian Kassel and Vladimir Turaev. *Braid groups*, volume 247 of *Graduate Texts in Mathematics*. Springer, New York, 2008. With the graphical assistance of Olivier Dodane.

[14] Daan Krammer. The braid group B_4 is linear. *Invent. Math.*, 142(3):451–486, 2000.

[15] Daan Krammer. Braid groups are linear. *Ann. of Math. (2)*, 155(1):131–156, 2002.

[16] R. J. Lawrence. Homological representations of the Hecke algebra. *Comm. Math. Phys.*, 135(1):141–191, 1990.

[17] D. D. Long and M. Paton. The Burau representation is not faithful for $n \geq 6$. *Topology*, 32(2):439–447, 1993.

[18] Dan Margalit and Jon McCammond. Geometric presentations for the pure braid group. *J. Knot Theory Ramifications*, 18(1):1–20, 2009.

[19] John Atwell Moody. The Burau representation of the braid group B_n is unfaithful for large n. *Bull. Amer. Math. Soc. (N.S.)*, 25(2):379–384, 1991.

[20] Luis Paris. Artin monoids inject in their groups. *Comment. Math. Helv.*, 77(3):609–637, 2002.

[21] Luis Paris. Artin monoids inject in their groups. *Comment. Math. Helv.*, 77(3):609–637, 2002.