UNIVERSITY OF CALIFORNIA
Santa Barbara

Standard Artin Group Generators and Dual Garside Algorithms

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

Doctor of Philosophy

in

Mathematics

by

Ashlee Keolalaulani Kalauli

Committee in Charge:

Professor Jon McCammond, Chair

Professor Daryl Cooper

Professor Darren Long

December 2021

The Dissertation of
Ashlee Keolalaulani Kalauli is approved:

Professor Daryl Cooper

Professor Darren Long

Professor Jon McCammond, Committee Chairperson

December 2021

Standard Artin Group Generators and Dual Garside Algorithms

by

Ashlee Keolalaulani Kalauli

To my littles, Cauchy and Sashimi; thank you both for always giving me a reason to smile.

Acknowledgements

ʻAʻohe hana nui ke alu ʻia (no task is too big, when done together by all).

This process, of completing this written work and my doctoral degree, has been arduous. There were many times where I doubted myself but support and faith of the people mentioned here never wavered. Though the list of people I mention here is but a fraction of those who have helped me along the way, know that I would not be writing these acknowledgements without you. Mahalo nui.

To my sister, Kayla-Ann Kalauli; thank you for the encouragement, the kick in the behind when I needed it, and for looking after my littles when I was away. I could not have done this without someone to talk to about the triumphs and tragedies of this journey. Thank you, truly. To my parents, John and Toni Kalauli, and my grandparents Anthony and Edna Gomes, thank you for rooting me in strength and holding me upright when so many times I was ready to fall. To my grandparents Keola and Chu-chu Kalauli, thank you for providing me a home to foster and build my love for math. To my extended Gomes, Kalauli, and Cypriano ʻohana, your hugs, smiles, and good wishes kept me going. Mahalo.

To my teachers, turned mentors, turned best friends, Laura Tavares and Roberto Pelayo; thank you for guiding me in directions I never knew were open to me. Mrs. Tavares, thank you for seeing all of me – me as a Hawaiian, me as a student turned educator, me as a ranch hand, and me as a friend – and for loving me for all that I am. Bob, thank you for being with me every step of the way; at the start in Hilo, through the highs and the many lows that came after that, and for knowing that I could do this even when I didn't.

I would not have endured if I did not find an ʻohana outside of Hawaiʻi. To my EDGE family, especially my 2019 EDGErs, thank you for showing me how much I grew and how much I can offer; you helped me to feel valuable which is something I lost. To my Indigenous Mathematicians ʻohana, thank you for helping me to see that there is a space for myself and people like me in mathematics.

Many thanks to my UCSB family that helped me to make a home in Santa Barbara; to my IM softball and basketball teams, my board game and virtual game night crew, my DRP mentees, and especially to Michael Dougherty, Joshua Pankau, Garo Sarajian, Marcos Reyes, and Elizabeth Crow. Many thanks for the guiding light shined by the UCSB faculty and staff and a special thanks to Ken Millett, Christopher Ograin, Medina Price and Rachel Zaragoza.

Finally, thank you to my committee members Daryl Cooper and Darren Long for helping me to refine my mathematical interests and providing me continual academic support throughout my years at UCSB. A special, heartfelt mahalo nui to my advisor Jon McCammond who helped me make UCSB my home, who taught math in a way that made everything seem more insightful and meaningful, and who helped me to find myself as a mathematician.

Curriculum Vitæ

Ashlee Keolalaulani Kalauli

Education

Ph.D. Mathematics | University of California, Santa Barbara | 2021 (expected)

M.A. Mathematics | University of California, Santa Barbara | 2018

M.A. Teaching | University of Hawai‘i at Hilo | 2015

B.A. Mathematics | University of Hawai‘i at Hilo | 2013

Publications

"The Impact of a Summer Intensive Mathematics and Chemistry Program at a Minority-Serving Institution in Hawai‘i" (with R. Pelayo, N. Furumo, S. Juvik), *The International Journal of Science, Mathematics and Technology Learning*, Volume 21, Issue 1, 13-24 (2014)

"Factorization Properties of Leamer Monoids" (with J. Haarmann, A. Moran, C. O'Neill, and R. Pelayo) *Semigroup Forum*, Volume 89, Issue 2, 409-421, (2014)

Awards

Department of Mathematics Outstanding Teaching Assistant | University of California, Santa Barbara | 2020

Graduate Student Association Excellence in Teaching | University of California, Santa Barbara | 2019

Academic Senate Outstanding Teaching Assistant | University of California, Santa Barbara | 2019

Eugene Cota-Robles Fellowship | University of California, Santa Barbara | 2015

Ford Foundation Predoctoral Fellowship | 2015

Research Interests

Geometric Group Theory | Coxeter groups, Artin groups, the word problem

Mathematics Education | online instruction, writing in mathematics, secondary and higher education curriculum development

Abstract

Standard Artin Group Generators and Dual Garside Algorithms

Ashlee Keolalaulani Kalauli

In 1925 Emil Artin introduced the braid groups, a widely studied class of groups which have applications in many mathematical and non-mathematical fields. These were later generalized to a larger class of groups, now called Artin groups, which have elegant, finite presentations. Despite these nice presentations, the word problem for most Artin groups has not been solved. The word problem, which seeks to algorithmically determine when two words represent the same group element, has been studied for over a century and a positive solution is essential to any systematic computational study of an infinite discrete group.

In 1972 Brieskorn and Saito showed that the spherical Artin groups have a solvable word problem using, in modern terminology, Garside structures. In 2017 Jon McCammond and Robert Sulway showed that Euclidean Artin groups also have a decidable word problem using the dual presentations of Euclidean Artin groups. These dual Euclidean Artin groups are isomorphic to their corresponding Euclidean Artin groups, but their group presentations include an infinite generating set. This infinite generating set gives some dual Euclidean Artin groups a Garside structure which provides a nice solution to the word problem. In this dissertation we study the algorithm that solves the word problem in the dual Euclidean Artin group of type $\textsc{Art}(\widetilde{A}_2)$ using its infinite generating set. We then rework this algorithm to solve the word problem for $\textsc{Art}(\widetilde{A}_2)$ using its standard, finite generating set.

Contents

List of Figures x

1 Introduction 1

I General Background 7

2 The Word Problem 9
2.1 Dehn's Word Problem 9

3 Coxeter Groups and Artin Groups 14
3.1 Coxeter groups 14
3.2 Artin Groups 26
3.3 Known Results for Coxeter and Artin Groups 31

4 Solutions to the Word Problem for Braid_n 34
4.1 Artin's Solution for Braid_n 34
4.2 Posets, Garside Groups and Garside Structures 40
4.3 Garside Structures and the Word Problem 43
4.4 A Garside Structure for Braid_n 46
4.5 Dual Garside Structure for Braid_n 57

5 Dual Euclidean Artin Groups 64
5.1 Isometries 64
5.2 Intervals 66
5.3 Coxeter Elements 70
5.4 Dual Artin Groups 74

II A Specific Example: $\text{Art}(\widetilde{A}_2)$ 76

6 The Dual Generating Set 78
6.1 The Coxeter complex and the Davis complex of $\text{Art}(\widetilde{A}_2)$ 78
6.2 Dual Generators 81

7 A Dual Solution to the Word Problem 87
7.1 Descendants of Dual Generators 87
7.2 Meets of Dual Generators 91
7.3 Pairs of Dual Generators in Normal Form 96

7.4 The Dual Algorithm 101

8 From Dual Generators to Standard Generators 103
8.1 The Hurwitz Action on the Dual Generating Set 103
8.2 The Hurwitz Action on the Standard Generating Set 107

9 A New, Standard Solution to the Word Problem 116
9.1 A Standard, Positive Normal Form 116
9.2 A Standard Normal Form 120
9.3 Standard Actions on Fundamental Chambers 121
9.4 Dual Actions on Strips 124

10 Future Work 129

A Appendix 131
A.1 An Algorithm Solving the Word Problem for Art(A_n) 131

Bibliography 150

List of Figures

1.1 Relationship between Artin and Coxeter Groups 1
1.2 The generator σ_1 of the 4-stand Braid group B_4 2
1.3 The generating set for the 4-strand braid group as a Garside group 4
1.4 The generating set for the Dual Euclidean Artin Group $\textsc{Art}^*(\tilde{A}_2, w)$ revealing its Garside structure 5

3.1 The product of permutations $\sigma \cdot \tau$ 16
3.2 The Coxeter matrix M, Coxeter diagram Γ and Schläfli matrix C for the group $\textsc{Sym}_4$ 19
3.3 Coxeter diagrams of all spherical Coxeter groups 21
3.4 The Coxeter diagrams of Euclidean Coxeter groups 23
3.5 A portion of the Coxeter complex of $\textsc{Cox}(\widetilde{A}_2)$ 24
3.6 The product of two braids with 4 strands where the braid on the left corresponds to the top braid in the concatenation of two braids 26
3.7 The images of f_i for a standard generator, σ_i, of B_n 27
3.8 The standard generator σ_i of B_n 28
3.9 A summary of the nice actions of $\textsc{Cox}(\Gamma)$ on spaces and the groups that arise 31
3.10 Pure Salvetti complex of type A_2 32

4.1 The combing of a 4-strand braid 37
4.2 The combing of a 4-strand braid, continued 38
4.3 A combed, 4-strand braid 39
4.4 Positive half-twists of adjacent strands $\sigma_1 = (1\ 2), \sigma_2 = (2\ ,3), \sigma_3 = (3\ 4)$. 42
4.5 The full, half-twist $(14)(23)$ in the 4-stand braid group 43
4.6 The set of left divisors of the full, half-twist 44
4.7 The generators of the 4-strand braid group, relabeled as integers 47
4.8 Rewriting $a \cdot b$ as $a' \cdot b'$ where $a' = a \cdot (rc(a) \wedge b)$ and $b' = (rc(a) \wedge b)^{-1} \cdot b$. 49
4.9 A trivial braid 53
4.10 The partition lattice Π_4 58
4.11 The crossing partition $\{13, 24\}$ and the noncrossing partition $\{124, 3\}$ of Π_4 58
4.12 The noncrossing partition lattice NC_4 59
4.13 The rotation braids $\delta_{\{1,2,3\}}, \delta_{\{2,3,4\}}$, and $\delta_{\{1,2,3\}}\delta_{\{2,3,4\}}$ 61

5.1 The coarse structure for a maximal hyperbolic isometry 69
5.2 The axial features of $\textsc{Cox}(\widetilde{G}_2)$ 72
5.3 The coarse structure for the $\widetilde{G}_2$ interval 73

6.1 A portion of the Coxeter complex and the Davis complex of $\textsc{Cox}(\widetilde{A}_2)$ . . 79

6.2 The Coxeter complex of $\textsc{Cox}(\widetilde{A}_2)$, a fundamental chamber, and the Coxeter axis 80
6.3 A strip in the Coxeter complex of type $\widetilde{A}_2$ 82
6.4 The lattice for the dual generating set of $\textsc{Art}(\widetilde{A}_2)$ 84
6.5 The general coarse structure of $\textsc{Art}(\widetilde{A}_2)$ 85

7.1 A factorization of tr_{ev} 89
7.2 Descendants of tr_{ev} and ro_{2k} 95
7.3 Descendants of two rotations from the same infinite family 96
7.4 Descendants of two rotations from different infinite families 96

8.1 Factorizations of w 106
8.2 Graph of the Hurwitz action on factorizations of w 107
8.3 Translating horizontal reflections via paths in the Coxeter complex 110

9.1 Simple generator descriptions and path descriptions 123
9.2 Right to left versus left to right actions of dual generators on strips . . . 125
9.3 The dual algorithm as strips and the standard algorithm as paths 127

1. Introduction

In 1910 Max Dehn introduced three fundamental questions that he sought to answer for all finitely presented groups. The first of these questions, which is known as the word problem, asks whether or not there is an algorithm that can decide if two words w and w', expressed as products of group generators and their inverses, represent the same element of the group. Over the next few decades, mathematicians were able to show that a variety of finitely presented groups have a solvable word problem and also produce examples of groups whose word problem is unsolvable. Over a century later, answering this question still remains a priority for various classes of groups.

Coxeter groups and Artin groups include many familiar groups, including the symmetric group on n elements and the n-strand braid groups. Coxeter groups are defined by their group presentations, and these can be encoded in and recovered from a labeled graph known as the Coxeter diagram or a matrix known as the Coxeter matrix. Coxeter groups are also realized geometrically as groups whose generators are reflections that act nicely on a metric vector space. The relations encode the angles at which the fixed hyperplanes of these reflections meet.

$$\begin{array}{ccccccc} \textsc{Braid}_n & \in & \textsc{Spherical Artin} & \subseteq & \textsc{Artin} \\ \downarrow\!\!\downarrow & & \downarrow\!\!\downarrow & & \downarrow\!\!\downarrow \\ \textsc{Sym}_n & \in & \textsc{Spherical Coxeter} & \subseteq & \textsc{Coxeter} \end{array}$$

Figure 1.1: Relationship between Artin and Coxeter Groups

Artin groups, like Coxeter groups, are defined by their group presentations, and the presentations for Artin groups can also be recovered with the use of the corresponding Coxeter diagram or Coxeter matrix since the two types of presentations are very similar. See Figure 1.1 for a schematic of the relationship between Artin and Coxeter groups. The n-strand braid group, Braid_n, is an Artin group introduced by Emil Artin in 1925 [Art25]. This well-known and well-studied group is now also known as the Artin group of type A_{n-1}. The elements of Artin's braid group can be seen as braided strands that cross over and under one another, anchored that the top and bottom like the one pictured in Figure 1.2. The group operation is concatenation.

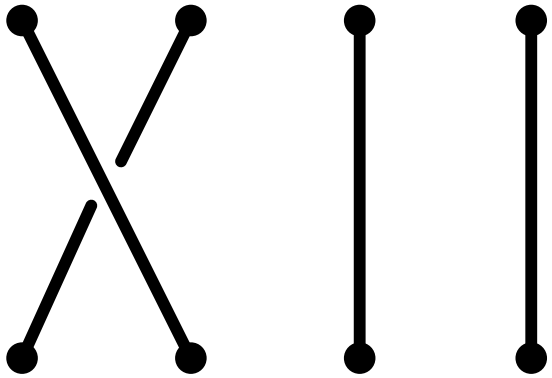


Figure 1.2: The generator σ_1 of the 4-stand Braid group B_4

The traditional presentation of the n-strand braid group has $n-1$ generators consisting of braids with a single crossing of adjacent strands and there two types of relations. The first is a commuting relation for products of generators that involve disjoint strands and the second is a relation for those generators that have a single strand in common. In particular, if we let σ_i represent the braid whose i-th strand crosses over the $(i+1)$-st

strand and all other strands remain unmoved, the braid group can be defined using the following group presentation:

$$\textsc{Braid}_n = \left\langle \sigma_1, \sigma_2, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_t\sigma_s = \sigma_s\sigma_t & \text{if } |t-s| > 1 \\ \sigma_t\sigma_s\sigma_t = \sigma_s\sigma_t\sigma_s & \text{if } |t-s| = 1 \end{array} \right\rangle$$

The braid groups are examples of *spherical Artin groups*, otherwise known as *finite-type Artin groups*.

Spherical Artin groups are derived from the spherical Coxeter groups which act geometrically on spheres. That is, spherical Coxeter groups act cocompactly and properly discontinuously by isometries on a sphere and are generated by reflections. This facilitates a geometric approach to abstract algebraic questions like that of the word problem.

In 1965, Frank Garside, guided by geometric observations, gave a new solution to the conjugacy problem for the braid groups, and, as a consequence, a new solution to the word problem [Gar65]. In 1972, E. Brieskorn and K. Saito extended Garside's results to all spherical Artin groups [BS72]. In 1998, Patrick Dehornoy and Luis Paris axiomatized Garside's work and introduced the concept of a Garside structure. Dehornoy and Paris focused on monoids M with a fixed, finite generating set S and a special element Δ. If M, S, and Δ satisfy certain basic properties, the monoid M embeds in its group of fractions G and there is a nice, uniform solution to the word problem for G [DP99].

The distinguished element in a Garside monoid is its *Garside element*, usually denoted by Δ. It is defined by three properties. First, the left and right divisors of Δ must coincide. Second, this common set of divisors must generate the group. Lastly, this set of divisors must form a lattice. These properties allow one to rewrite a product of divisors

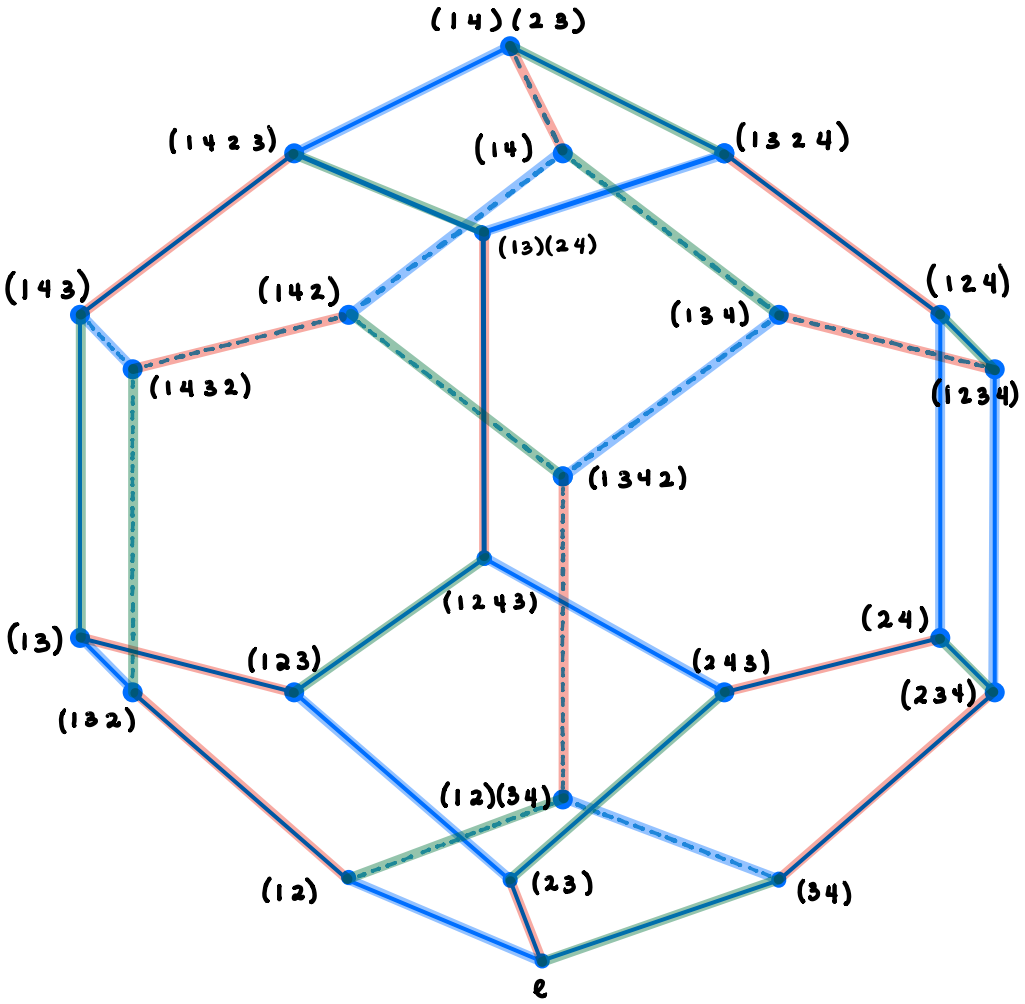


Figure 1.3: The generating set for the 4-strand braid group as a Garside group

of Δ using their meets and joins. If the set of divisors of Δ is finite, the group of fractions is called a Garside group. If the set of divisors of Δ is infinite (but still nice enough for the algorithms to work), the group of fractions is called a *non-standard* Garside group. The standard Garside generating set for the 4-strand braid group can be seen in Figure 1.3, indexed by the 23 nontrivial permutations (see Chapter 4 for details).

In 2017, Jon McCammond and Robert Sulway showed that Euclidean Artin groups are isomorphic to what they call Dual Euclidean Artin groups [MS17]. Furthermore,

they show that some of these Dual Euclidean Artin groups have a non-standard Garside structure. More generally, all dual Euclidean Artin groups embed in a non-standard Garside group. Both possibilities lead to a solution to the word problem. Though the word problem for Artin groups in general remains open, it is conjectured that all Artin groups have a solvable word problem.

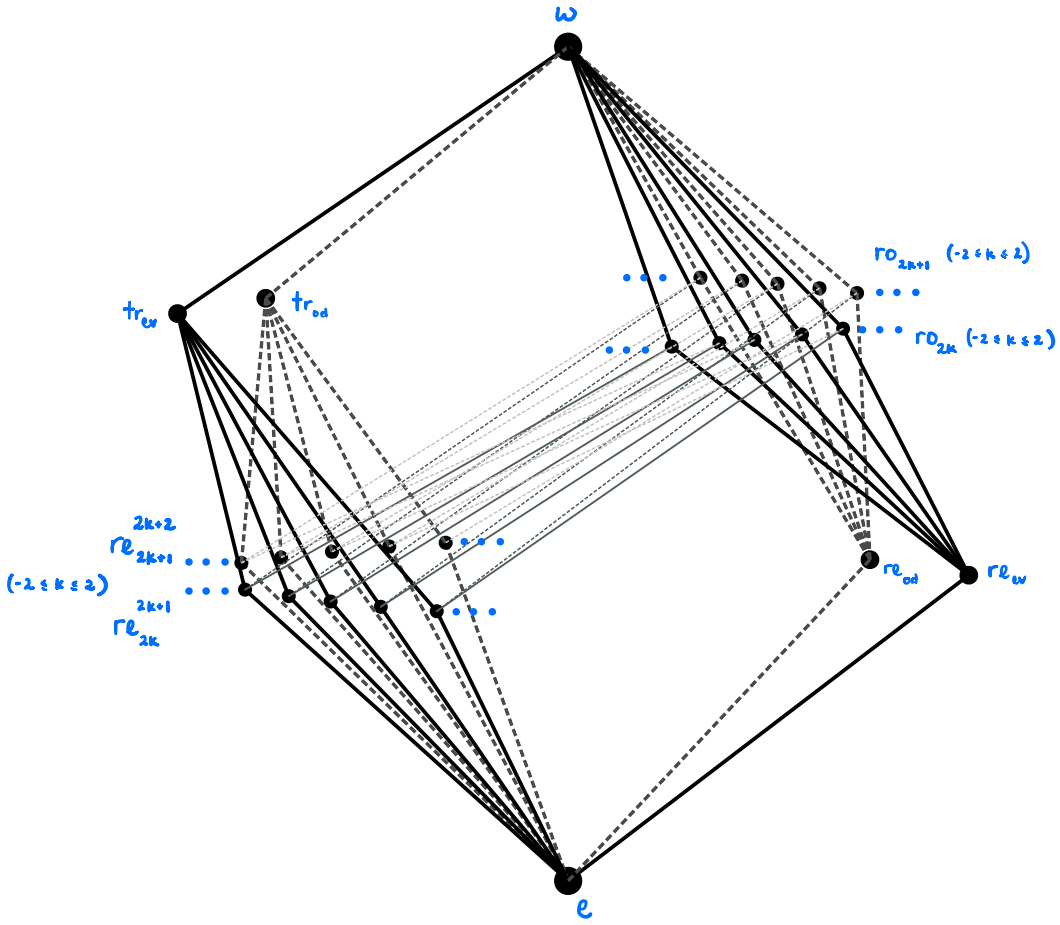


Figure 1.4: The generating set for the Dual Euclidean Artin Group $\text{ART}^*(\tilde{A}_2, w)$ revealing its Garside structure

In this dissertation, we investigate the Garside structure of the dual Euclidean Artin group $\text{ART}^*(\tilde{A}_2, w)$. For a specific choice of Coxeter element, this group has a (non-standard) Garside structure with an infinite generating set. See Figure 1.4. We first

investigate the algorithm that solves the word problem for the Dual Euclidean Artin group $\textsc{Art}^*(\tilde{A}_2, w)$. We then translate the dual generating set to the standard generating set of $\textsc{Art}(\widetilde{A}_2)$ and use this algorithm to solve the word problem for $\textsc{Art}(\widetilde{A}_2)$ with respect to its standard generating set. The algorithm in the standard generating set is new and this is the main result in this dissertation.

This dissertation is organized into two parts. Part I contains a general background discussion of the word problem, Coxeter groups, Artin groups, and dual Euclidean Artin groups. Part II looks closely at the Euclidean Artin group $\textsc{Art}(\widetilde{A}_2)$ and presents two solutions to the word problem for this group using two different generating sets; one infinite, and the other finite.

Part I

General Background

The first part of this dissertation provides the general background needed for the new results in the second part. Chapter 2 introduces Dehn's word problem and it reviews solutions to the word problem for several groups. In Chapter 3, we provide an introduction to Coxeter groups and Artin groups using the symmetric group and the braid groups as guiding examples. Chapter 4 discusses Garside theory as a systematic way to provide solutions to the word problem. We define Garside groups and Garside structures and show how their defining properties yield nice solutions to the word problem. Chapter 5 provides an introduction to Dual Euclidean Artin groups with particular emphasis on the Garside structure associated with these groups.

2. The Word Problem

The word problem, posed by Max Dehn in 1910, is a central question in algorithmic group theory. It asks if there is a well-defined process for determining if a word, written in terms of a fixed generating set of the group, represents the identity element. While this question has generated a tremendous amount of research, the early progress made in solving the word problem for braid groups came by way of a very time consuming algorithm. In this chapter, we discuss the word problem, the way solutions to the word problem arise, and provide examples of groups that have solvable word problems and comment on the existence of groups whose word problem cannot be solved.

2.1 Dehn's Word Problem

Stemming from a specific topological question pertaining to fundamental groups of surfaces, in 1910 Max Dehn posed three fundamental questions to be answered for any finitely presented group. The first question was known as the *word problem* and asked the following: for a group G with a fixed finite presentation $G = \langle S \mid R \rangle$, given two words w and w', is there an algorithm to decide if w is equivalent to w'? This can be translated to the similar yet equivalent question that asks, given a word w in the free monoid generated by the set $S \cup S^{-1}$, is there an algorithm to decide if w represents the identity element of G? In what follows, we discuss the word problem, solutions to the word problem for various groups, and why the defining properties of Garside groups yield nice solutions to the word problem.

Definition 2.1.1. (Words in Groups) Let S be a set. We take S^{-1} be the set in one-to-one correspondence with S whose elements represent the inverses of the elements in S. We refer to the elements of $S \cup S^{-1}$ as *letters* and a finite sequence of elements from $S \cup S^{-1}$ as a *word*. Words made up exclusively from letters in S are called a *positive words* and words consisting of letters from S^{-1} only are called *negative words*. The *length* of a word is the number of letters in the word, counting multiplicity. Equivalence classes of words can be constructed based on the deletion or insertion of subwords of the form ss^{-1} or $s^{-1}s$. A word is said to be *reduced* if it contains no inverse pair subwords of the form ss^{-1} or $s^{-1}s$ for $s \in S$. We use $[w]$ to represent the set of all words equivalent to w by inserting or deleting subwords of this form. Moreover, every word can be reduced by systematically removing such subwords and this reduction process is confluent so that the reduced word that results is unique. Thus, every equivalence class $[w]$ has a unique reduced element of minimum length. The multiplication of two equivalence classes is the equivalence class of the concatenation of the representatives of these equivalence classes, which also has a unique reduced element. The set of all finite length words in $S \cup S^{-1}$ is denoted $(S \cup S^{-1})^*$. If we take the empty word, [] to be the identity, the set of all reduced words in $(S \cup S^{-1})^*$ under the operation of multiplication described above is known as the *free group with generating set* S, denoted $\mathbb{F}_S$. The image of the set S is called a *basis* for $\mathbb{F}_S$.

A well-known result that can be found in any graduate level text is as follows:

Theorem 2.1.2. Every finitely generated group is the quotient group of a free group by a normal subgroup.

This result will help us to define group presentations. Let R be a subset of $\mathbb{F}_S$. We say that N is the *normal closure* of R if N is the intersection of all normal subgroups that contain the elements of R.

Definition 2.1.3. (Group Presentations) A *presentation* is a set S of generators and a set $R \subseteq \mathbb{F}_S$ of *relators*. The group G it presents is the quotient group of $\mathbb{F}_S$ by the normal closure of R. A group defined via a presentation is called a *marked group* since it has a distinguished generating set S. In this case, we write $G = \langle S \mid R \rangle$. We can think of G as a set of equivalence classes where two words w and w' are related if we can write w as w' using either a finite sequence of insertions or deletions of subwords of the form $s_i s_i^{-1}$ or $s_i^{-1} s_i$ with $s_i \in S$ or by inserting or deleting reduced words representing the elements in R. At times, it is more convenient to use relations instead of relators. A *relation* is a statement equating two words rather than a single word implicitly set equal to the identity.

Definition 2.1.4. (Normal Forms) Because each element in a group G can be seen as an entire equivalence class, when discussing the word problem, these words with multiple spellings must be handled carefully. It helps to have a prescribed way of writing a given element $g \in G$ as a product of the generators in S and their inverses. In fact, solutions to the word problem are often some sort of uniform procedure or *algorithm* that puts words into what we call *normal form*, a uniquely determined word that represents each group element.

Specifically, a normal form is a function. Let $\pi : (S \cup S^{-1})^* \to \mathbb{F}_S/N = G$ be the evaluation map such that $w \mapsto [w]$. Since S generates G, we know that π is onto. A

normal form is a function $\eta : G \to (S \cup S^{-1})^*$ such that the composition $\pi \circ \eta : G \to G$ is the identity map. Typically, when one refers to a normal form, they mean to reference not the function η but the image of η in $(S \cup S^{-1})^*$. It is precisely this image, that allows us to determine whether or not a group element is really the identity. If a normal form function exists for a group $G = \langle S \mid R \rangle$, and it can be computed algorithmically, then the word problem for G is *solvable*.

In the decades that followed Dehn's posing of the Word Problem, there was much interest in finding general solutions. Most results came by imposing some restrictions on the groups or solutions that came for only very specific presentations. One of the first major results, by Wilhelm Magnus, was the following [Mag32]:

Theorem 2.1.5. *Every one relator group has a solvable word problem.*

Decades later, through constructive examples of finitely presented groups, it was learned that a question as simple as asking whether or not an element is equivalent to the identity might not have a simple answer [Boo57], [Nov55].

Theorem 2.1.6. *There exist finitely presented groups with unsolvable word problems.*

In what follows, we give some examples of groups with solvable word problems and their normal forms.

Example 2.1.7. (The Free Group) The free group $\mathbb{F}_S$ with basis S, which was introduced in Definition 2.1.1, has a solvable word problem. The free group $\mathbb{F}_S$ has a presentation $\langle S \mid \varnothing \rangle$. Given a word written in $S \cup S^{-1}$, we can repeatedly delete any subwords of the form ss^{-1} or $s^{-1}s$ for any $s \in S$. Continuing this process until there are no subwords of this

form gives us a normal form for $\mathbb{F}_S$. The process of removing these words is confluent, so the order in which they are removed does not change the final result. The original word will be trivial if and only if the reduced word is the empty string.

Example 2.1.8. (Finitely Generated Abelian Groups) Let G be a finitely generated abelian group. Using the fundamental theorem of finitely generated abelian groups, we can write G as a finite direct product of cyclic groups. Next, we can choose a generating set $S = \{s_1, \ldots, s_n\}$ for G that has one standard generator for each of these cyclic summands. Since the group is abelian, any word in this generating set can be grouped to be written as $s_1^{k_1} \cdots s_n^{k_n}$, with restrictions on the integers k_i when the corresponding cyclic summand is finite. By our choice of generating set, this representation is also uniquely determined for each group element. This produces a normal form where the original word represents the identity element if and only if $k_i = 0$ for all i. Should a different generating set T for the group be given, one could construct an algorithm first to change the word in $T \cup T^{-1}$ to a word in $S \cup S^{-1}$, simplify the word to its normal form over S, and finally convert it back into a word in $T \cup T^{-1}$. Conversions of this type show that the word problem is independent of the choice of finite generating set. More generally, the word problem is closely related to the growth of the Dehn function, which is also independent of the generating set chosen.

3. Coxeter Groups and Artin Groups

Coxeter groups and Artin groups are two closely related classes of groups of central importance in geometric group theory. Moreover, the connection between these two classes of groups allow results from one class to motivate results in the other. More specifically, Coxeter groups, which are much better understood than Artin groups, are the source of inspiration for many results on Artin groups. Coxeter groups are defined by simple presentations that can be recovered via matrix constructions, diagramatically, or geometrically as groups generated by reflections acting on metric vector spaces. Since Artin groups are built from Coxeter groups, understanding this relationship can provide direction when studying the more complicated Artin groups. This chapter will provide many of the definitions and key examples that will be utilized in the remainder of this dissertation. We begin with a basic example of a Coxeter group and its corresponding Artin group.

3.1 Coxeter groups

Coxeter groups provide some of the algebraic building blocks related to this dissertation, and they include the discrete groups generated by reflections acting on Euclidean space. All Coxeter groups are well-understood objects. We begin this section with a discussion of the symmetric group as a motivating example of a Coxeter group. We then formally define Coxeter groups and view the symmetric group in this light.

Definition 3.1.1. (Permutations) For each $n \in \mathbb{N}$, let $[n]$ represent the set $\{1, 2, \ldots, n\}$. A *permutation* is a bijection between a left copy of $[n]$ to a right copy of $[n]$. To describe bijections we commonly use *cycles* such as $\sigma = (a_1\ a_2\ \cdots\ a_t)$, which refers to the permutation where a_i on the left corresponds to a_{i+1} on the right for $i < t$, a_t on the left to a_1 on the right, and fixes any element other than the a_i appearing in the cycle. The multiplication of permutations corresponds to the composition of bijections; this operation can be viewed by concatenating the correspondences left-to-right to match the conventions of the programming package Sage. In particular, every bijection from $[n]$ to itself can be written as a product (or composition) of disjoint cycles in some non-unique way, as disjoint cycles pairwise commute. The permutation σ acts on $[n]$ from the right and from the left. If a_i on the left corresponds to a_{i+1} on the right, then $a_i \cdot \sigma = a_{i+1}$ and $\sigma \cdot a_{i+1} = a_i$.

Definition 3.1.2. (Permutations as Functions) We can also view permutations as functions using function notation to describe them. When doing so, the domain is on the left and the range is on the right which assures that function composition agrees with the compositions of bijections defined above.

For a concrete example, suppose $S = [4] = \{1, 2, 3, 4\}$ and $\sigma(1) = 3, \sigma(2) = 1, \sigma(3) = 4, \sigma(4) = 2$. Then the cycle notation for σ is given by $\sigma = (1243)$. If τ is a permutation such that $\tau(1) = 3, \tau(2) = 4, \tau(3) = 1$, and $\tau(4) = 2$ then the cycle notation for τ is given by $\tau = (13)(24)$. Because these permutations are bijective functions, we can compose any two permutations to yield another. These compositions occur left to right so that the product $\sigma \cdot \tau = (1243)(13)(24) = (14)$. See Figure 3.1.

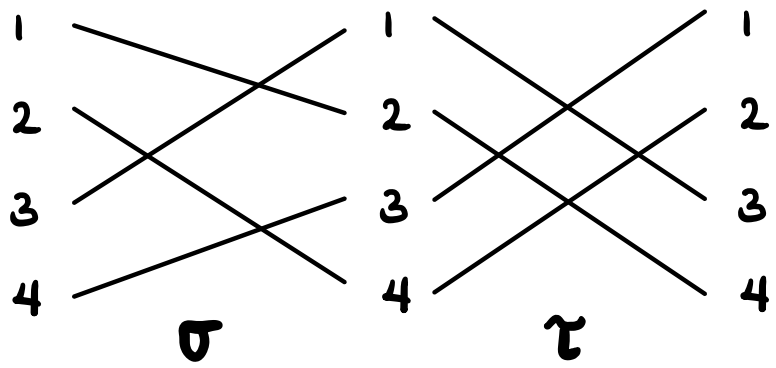


Figure 3.1: The product of permutations $\sigma \cdot \tau$

Definition 3.1.3. (Symmetric Group) The *symmetric group* of rank n is the group of permutations of $[n]$ under composition. It turns out that the symmetric group can be presented as follows:

$$\textsc{Sym}_n = \left\langle \sigma_1, \sigma_2, \ldots, \sigma_{n-1} \;\middle|\; \sigma_i^2 = e \text{ and } \begin{array}{ll} \sigma_t\sigma_s = \sigma_s\sigma_t & \text{if } |t-s| > 1 \\ \sigma_t\sigma_s\sigma_t = \sigma_s\sigma_t\sigma_s & \text{if } |t-s| = 1 \end{array} \right\rangle,$$

where σ_i is the permutation $(i \;\; i+1)$ and e represents the identity element. We refer to cycles of length two as *transpositions* and let e denote the identity permutation. A transposition is said to be *adjacent* if it is the transposition of two consecutive integers. The standard generating set of $\textsc{Sym}_n$ is the set of all adjacent transpositions.

Definition 3.1.4. (Coxeter group) A *Coxeter group* is a marked group (i.e. a group with a fixed generating set S) with a special type of presentation. In particular, there are only two types of relations: $s^2 = e$ for all $s \in S$ and, for all $s, t \in S$, there is at most one relation of the form $(st)^m = e$ for some integer $m = m(s,t)$.

Remark 3.1.5. (The Symmetric Group) Let $S = \{\sigma_1, \sigma_2, \ldots, \sigma_{n-1}\}$ be the standard generators given in the group presentation for $\textsc{Sym}_n$. Notice that using the relation $\sigma_i^2 = e$, the relation $(\sigma_t\sigma_s)^2 = e$ can be rewritten as $\sigma_t\sigma_s = \sigma_s\sigma_t$ if $|t-s| > 1$ (since $\sigma_i^2 = e$), and the relation $(\sigma_t\sigma_s)^3 = e$ can be rewritten as $\sigma_t\sigma_s\sigma_t = \sigma_s\sigma_t\sigma_s$ if $|t-s| = 1$. With the generating set S subject to these rewritten relations, we see that $\textsc{Sym}_n$ is indeed a Coxeter group. For concreteness, consider

$$\begin{aligned}\textsc{Sym}_4 &= \langle \sigma_1, \sigma_2, \sigma_3 \mid \sigma_1^2, \sigma_2^2, \sigma_3^2,\ (\sigma_1\sigma_2)^3, (\sigma_2\sigma_3)^3,\ (\sigma_1\sigma_3)^2 \rangle \\ &= \langle \sigma_1, \sigma_2, \sigma_3 \mid \sigma_1^2, \sigma_2^2, \sigma_3^2,\ \sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2, \sigma_2\sigma_3\sigma_2 = \sigma_3\sigma_2\sigma_3,\ \sigma_1\sigma_3 = \sigma_3\sigma_1 \rangle .\end{aligned}$$

Using the first presentation, we see that $\textsc{Sym}_4$ is a Coxeter group. which we denote $\textsc{Cox}(\Gamma)$. The second presentation will be used to show the close relationship between $\textsc{Sym}_4$ and the 4-strand braid group $\textsc{Braid}_4$.

A group presentation for a Coxeter group can be encoded in a Coxeter matrix or a Coxeter diagram.

Definition 3.1.6. (Coxeter matrix) Given a set S, we say that $M : S \times S \to \{1, 2, \ldots, \infty\}$ is *Coxeter matrix* if the following properties hold:

(1) $M(a, b) = M(b, a)$ for all $a, b \in S$; that is, M is a symmetric matrix, and

(2) $M(a, b) = 1$ if and only if $a = b$.

The Coxeter group W is a group with generators in S subject to the relations derived from the Coxeter matrix M. Given $S = \{s_1, s_2, \ldots, s_n\}$, the Coxeter group W with

generators S would have the following group presentation

$$W = \left\langle s_1, \ldots, s_n \mid (s_i s_j)^{M(s_i,s_j)} = 1 \right\rangle.$$

In the case that $M(s_i, s_j) = \infty$, no relation of the form $(s_i s_j)^{M(s_i,s_j)}$ is imposed. Notice that $M_{ii} = 1$ implies $s_i^2 = 1$ for all i, and therefore, all the generators of W are involutions. The pair (W, S) where W is a Coxeter group with generating set S is called a *Coxeter system*.

Definition 3.1.7. (Coxeter diagram) A *Coxeter diagram* is a simple graph (with no loops or multiple edges) with labels on its edges. In particular, a Coxeter diagram has a vertex for each $s_i \in S$, an unlabeled edge connecting two vertices labeled s_i and s_j if $M(s_i, s_j) = 3$, and an edge labeled k if $M(s_i, s_j) = k > 3$ (including the possibility that $k = \infty$). If $M(s_i, s_j) = 2$, then the vertices s_i and s_j are not connected by an edge. We can use the presentation encoded in Γ to define the Coxeter group W and denote it using $\mathrm{Cox}(\Gamma)$. If Γ is connected, we say $\mathrm{Cox}(\Gamma)$ is *irreducible*.

Definition 3.1.8. (Parabolic subgroup) Given a Coxeter diagram Γ, let $V(\Gamma)$ denote the set of vertices of Γ. We then take Γ_I to be the subgraph of Γ formed by the subset $I \subset V(\Gamma)$ with all of the edges in Γ that connect pairs of vertices in I. The group corresponding to the diagram Γ_I is a subgroup of $\mathrm{Cox}(\Gamma)$. In particular, given a Coxeter system (W, S) with Coxeter diagram Γ and $I \subset S = V(\Gamma)$, we let $W_I = \mathrm{Cox}(\Gamma_I)$ be the subgroup of W generated by I so that (W_I, I) is a Coxeter system. Subgroups of W obtained in this way are called *parabolic subgroups* of W.

We revisit the symmetric group as a Coxeter group with these new defining properties. The following image contains the Coxeter matrix M, Coxeter Diagram Γ, and the Schläfli matrix C for $\textsc{Sym}_4$.

$$M = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 1 & 3 \\ 2 & 3 & 1 \end{bmatrix} \qquad \Gamma = \quad \overset{\sigma_1}{\bullet} \!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\! \overset{\sigma_2}{\bullet} \!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\! \overset{\sigma_3}{\bullet} \qquad C = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Figure 3.2: The Coxeter matrix M, Coxeter diagram Γ and Schläfli matrix C for the group $\textsc{Sym}_4$

Coxeter groups also act nicely on certain geometric objects. In particular, there is a faithful, linear representation due to Jacques Tits that allows us to take a closer look at the geometry of a Coxeter group.

Theorem 3.1.9. *Let (W, S) be a Coxeter system with corresponding Coxeter diagram Γ and Coxeter matrix M, then there is a faithful representation $\rho : W \to GL(n, \mathbb{R})$, such that the image of each element of $S = \{s_1, \ldots, s_n\}$ is a linear involution fixing a hyperplane and for $s_i, s_j \in S$, with $i \neq j$, $\rho(s_i)\rho(s_j)$ has order M_{ij}.*

Definition 3.1.10. (Schläfli matrix) We construct the representation described in Theorem 3.1.9 using a symmetric matrix known as the *Schläfli matrix* denoted C whose entries are given by

$$c_{ij} = \begin{cases} -2\cos(\pi/M_{ij}) & \text{when } M_{ij} \in \mathbb{Z}^+ \\ -2 & \text{when } M_{ij} = \infty \end{cases}.$$

Let (W, S) be an irreducible Coxeter system, and let C denote its corresponding Schläfli matrix. Note that since M is symmetric, C is also symmetric matrix; and as a real symmetric matrix, C has real eigenvalues. If all of the eigenvalues of C are positive, W acts properly, discontinuously, and cocompactly by isometries (i.e. *geometrically*) on

a sphere in Euclidean space. If C has one zero eigenvalue and no negative eigenvalues, then it turns out that W acts geometrically on Euclidean space. In what follows, we'll refer to these Coxeter groups as *spherical* and *Euclidean Coxeter groups*, respectively.

Definition 3.1.11. (Reflections in Euclidean Space) Let $\mathbb{R}^n$ denote n-dimensional Euclidean space with the standard dot product, represented by $\cdot$. Let $\alpha \in \mathbb{R}^n$ be a nonzero vector and take H_α to be the hyperplane of $\mathbb{R}^n$ given by $H_\alpha = \{v \in \mathbb{R}^n \mid v \cdot \alpha = 0\}$. Define R_α to be the linear transformation $R_\alpha : \mathbb{R}^n \to \mathbb{R}^n$ where $R_\alpha(v) = v - 2\dfrac{v \cdot a}{\alpha \cdot \alpha}\alpha$. We call R_α the *reflection through* H_α. Notice that R_α^2 is the identity map for all $v \in \mathbb{R}^n$, $R_\alpha(\alpha) = -\alpha$, and $R_\alpha(\beta) = \beta$ for all $\beta \in H_\alpha$.

Definition 3.1.12. (Root Systems) Let Φ denote a finite set of nonzero vectors in $\mathbb{R}^n$. We refer to the elements of Φ as *roots* and Φ as a *root system* if the following properties hold:

1. $\Phi \cap \mathbb{R}\alpha = \{\alpha, -\alpha\}$ for all $\alpha \in \Phi$ and

2. $R_\alpha \Phi = \Phi$ for all $\alpha \in \Phi$.

We say that Φ is *crystallographic* if for every $\alpha, \beta \in \Phi$, $2\dfrac{\alpha \cdot \beta}{\alpha \cdot \alpha} \in \mathbb{Z}$.

Definition 3.1.13. (Spherical Coxeter Groups) Let (W, S) be an irreducible Coxeter system with corresponding Coxeter matrix M, where W is finite and $S = \{s_1, \ldots, s_n\}$. For the generating set S, there is a set of vectors $\alpha_1, \ldots \alpha_n \in \mathbb{R}^n$ that are pairwise non-acute so that the hyperplanes H_{α_i} and H_{α_j} intersect with a dihedral angle of $\pi/M(s_i, s_j)$. We can then recover the root system Φ by considering the orbit of the vectors α_i. Moreover, the reflection group generated by R_{α_i} is isomorphic to W.

Consider the complement of the union of H_α for all $\alpha \in \Phi$, which consists of contractible connected components. Each component deformation retracts onto its intersection with $\mathbb{S}^{n-1} \subset \mathbb{R}^n$. We refer to the closure of a component in $\mathbb{S}^{n-1}$ as a *chamber*. The *fundamental chamber* is the one where the dot product with each α_i is positive. The set of chambers gives a cell structure for the sphere, called the *Coxeter complex*, which is invariant under the group action by W.

In 1934, Coxeter classified all finite irreducible (spherical) Coxeter groups in terms of their Coxeter diagrams. See Figure 3.3.

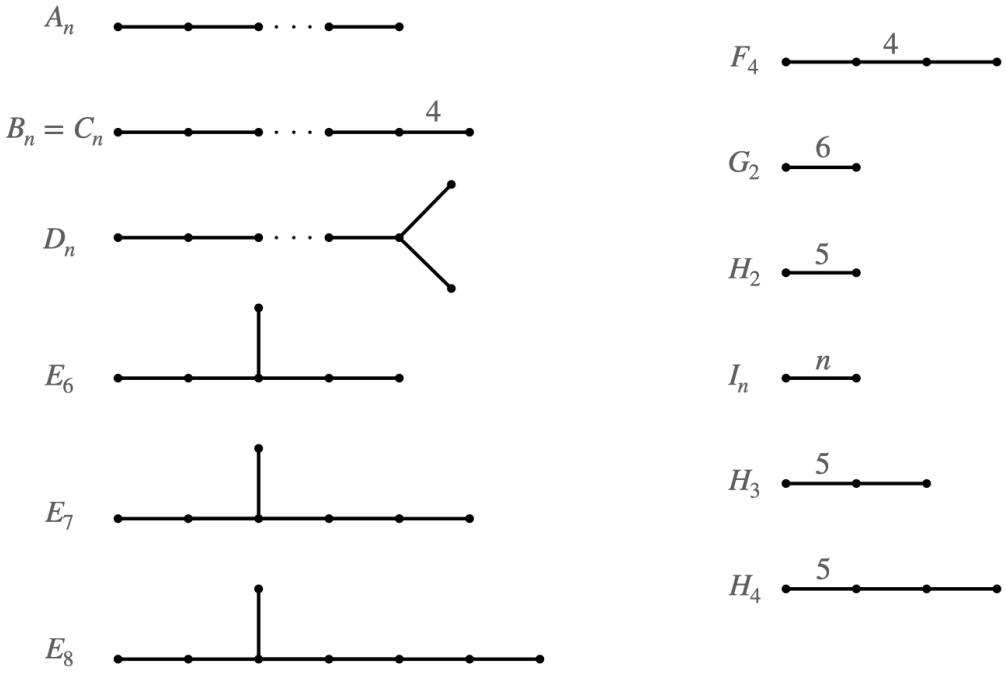


Figure 3.3: Coxeter diagrams of all spherical Coxeter groups

Example 3.1.14. (The Symmetric Group) We revisit Example 3.1.5 to put these new definitions in context. We saw that $\textsc{Sym}_n$ is a Coxeter group via its presentation. We can also write $\textsc{Sym}_n$ as $\textsc{Cox}(A_{n-1})$ whose Coxeter diagram is pictured in Figure 3.3 consisting of $n-1$ vertices. Let $S = \{\sigma_1, \ldots, \sigma_{n-1}\}$ be the generating set of $\textsc{Sym}_n$. Each σ_i corresponds to the vector $\alpha_i = \mathbf{e}_i - \mathbf{e}_{i+1} \in \mathbb{R}^n$ where $\mathbf{e_i}$ represents the ith standard basis vector of $\mathbb{R}^n$. We can represent each hyperplane H_{α_i} via the equation $x_i = x_{i+1}$. The reflection R_{α_i} acts on $\mathbb{R}^n$ by swapping the i-th and $(i+1)$-st coordinates of the vector $(x_1, x_2, \ldots, x_n)$. Notice, each of the α_i are contained in the orthogonal complement of the span of the vector $(1, 1, \ldots, 1)$. The span of the vectors α_i is a subspace isomorphic to $\mathbb{R}^{n-1}$. In particular, $\textsc{Sym}_n$ acts on this subspace as a group generated by the reflections R_{α_i}. The Coxeter complex is then given by the intersection of $\mathbb{S}^{n-2}$ with this this $(n-1)$-dimensional subspace, giving a tessellation on $\mathbb{S}^{n-2}$ of $n!$ spherical simplices of dimension $n-2$.

Definition 3.1.15. (Euclidean Coxeter Groups) Given an irreducible crystallographic spherical Coxeter group W, we can obtain the irreducible Euclidean Coxeter group $\widetilde{W}$ by adding an affine hyperplane to the existing hyperplanes corresponding to W. If $W = \textsc{Cox}(\Gamma)$, we produce the Coxeter diagram for $\widetilde{W}$ by adding a vertex to Γ. In Figure 3.4, we find all possible Coxeter diagrams for Euclidean Coxeter groups where the highlighted vertex and dashed edged indicate what vertex is added to the Coxeter diagram of each corresponding spherical Coxeter group. We note that all of the edge labels found in Euclidean Coxeter diagrams are either $2, 3, 4$ or 6 (or ∞ in $\widetilde{A}_1$). In fact, every possible irreducible Coxeter group arises in this way.

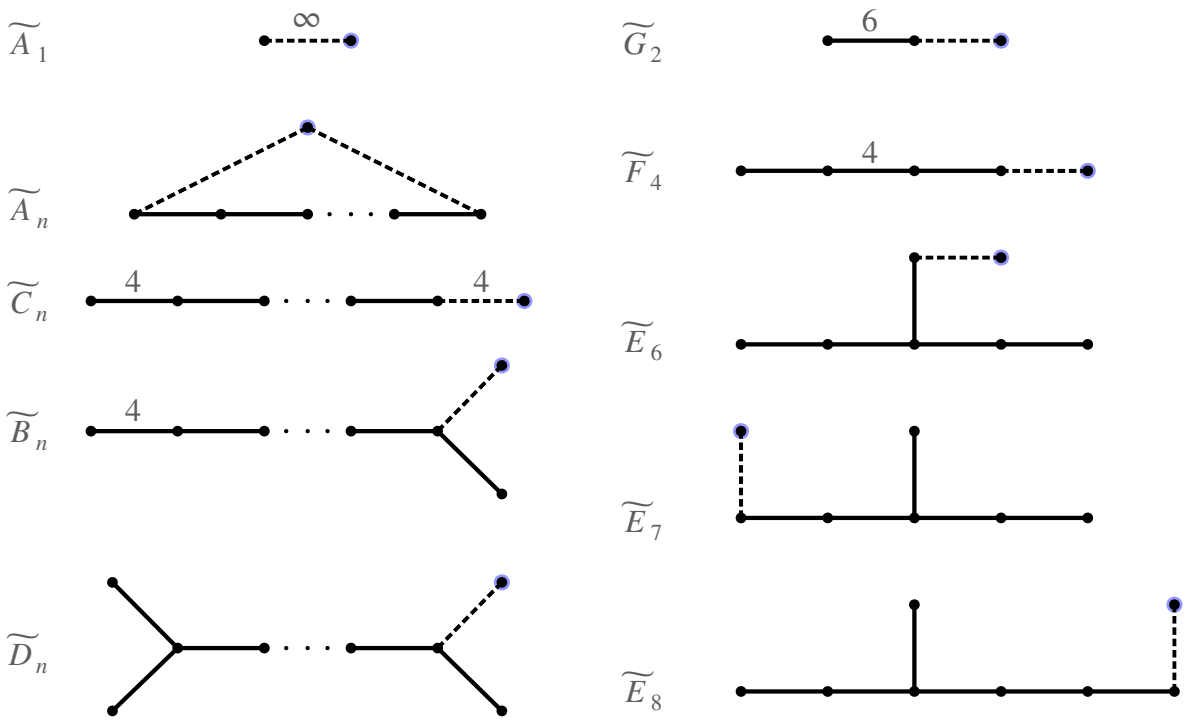


Figure 3.4: The Coxeter diagrams of Euclidean Coxeter groups

Definition 3.1.16. (Euclidean Coxeter Simplices) Let $\widetilde{W}$ be an irreducible Euclidean Coxeter group. The Coxeter diagram corresponding to $\widetilde{W}$ represents a simplex in Euclidean space, which we call *Euclidean Coxeter simplices*, in the following way: the vertices of the Coxeter diagram corresponds to its codimension one faces, and the edges labeled $2, 3, 4$ and 6 correspond to the faces of the simplex that intersect at dihedral angles of $\frac{\pi}{2}, \frac{\pi}{3}, \frac{\pi}{4}$, or $\frac{\pi}{6}$ respectively. This describes the Euclidean Coxeter simplices corresponding to each Coxeter diagram with the exception of $\widetilde{A}_1$ (which corresponds to a 1-simplex in $\mathbb{R}$ whose faces are the endpoints).

Definition 3.1.17. (Coxeter Complex) Let $\widetilde{\Gamma}$ denote the Coxeter diagram corresponding to the Euclidean Coxeter group $\widetilde{W}$. Let τ denote the Euclidean n-simplex described in Definition 3.1.16. Then, $\widetilde{W} = \mathrm{Cox}(\widetilde{\Gamma})$ can be seen as the group generated by the $n+1$

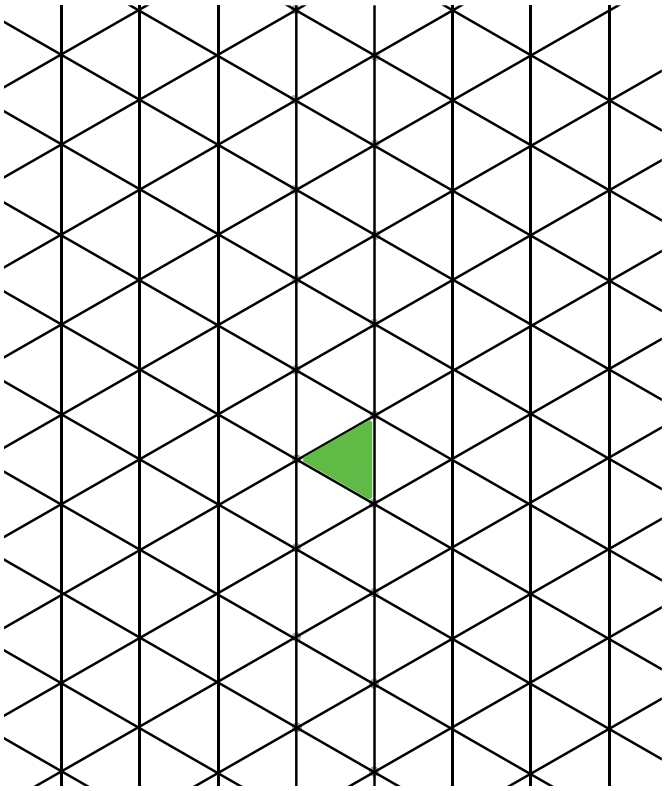


Figure 3.5: A portion of the Coxeter complex of $\text{Cox}(\widetilde{A}_2)$

reflections that fix a facet of τ. Using the orbit of τ under this group action, we give Euclidean space the structure of a metric simplicial complex called the *Coxeter Complex* of W. We refer to the top dimensional simplices as *chambers* and we refer to τ as the fundamental chamber. For concreteness, the Euclidean tiling for the Coxeter group $\text{Cox}(\widetilde{A}_2)$ can be seen in Figure 3.5 with the fundamental chamber highlighted in green.

Using the Cayley graph of a Coxeter group, Michael Davis constructed a cell complex corresponding to a Coxeter group. To describe this complex, we start with a few definitions.

Definition 3.1.18. (Tits Cone) Given a Coxeter group W with corresponding nonsingular Schläfli matrix M, the hyperplanes orthogonal to the standard basis vectors e_i bound

a closed simplicial cone $\mathcal{C}$. The union of the images of $\mathcal{C}$ under the action of W is called the *Tits cone*.

Definition 3.1.19. (W-permutahedron) Given a spherical Coxeter group W, let σ be a top-dimensional simplex in the corresponding tiling of the sphere. Let $\mathcal{C}$ be the simplicial Euclidean cone generated by non-negative scalar multiples of the points in σ. There is a unique point p in the simplicial cone $\mathcal{C}$ such that the distance between p and each of its facets is $\frac{1}{2}$. A *W-permutahedron* is a Euclidean polytope given by the convex hull of the W-orbit of p.

Definition 3.1.20. (Cayley graph) Given a Coxeter system (W, S), the unoriented, right *Cayley graph* of W consists of vertices labeled by the elements of W and edges labeled by elements of (W, S) that starts at the vertex w and ends at the vertex $w \cdot s$.

Remark 3.1.21. Via the Tits construction, the right Cayley graph of W is actually the 1-skeleton of the cell complex dual to the Tits cone. In particular, the Cayley graph contains a vertex w corresponding to the image $w\mathcal{C}$ of the simplicial cone $\mathcal{C}$ for each $w \in W$. Two vertices are connected by an edge when the corresponding simplicial cones share a common codimension 1 face.

We now define the Davis complex which is constructed by attaching W-permutahedra to its unoriented Cayley graph.

Definition 3.1.22. (Davis complex) Let (W, S) be a spherical Coxeter group with corresponding unoriented, Cayley graph Σ. For each subset $S' \subset S$ such that the parabolic subgroup $W_{S'}$ is finite and for each element $w \in W$, we attach a metric $W_{S'}$-permutahedron

to the vertices of Γ labeled by the elements in the coset $wW_{S'}$ in W. When $w'W_{S'}$ and $w''W_{S''}$ are two cosets such that $w'W_{S'} \subset w''W_{S''}$ then we identify the $W_{S'}$-permutahedron attached to the vertices labeled by the elements in $w'W_{S'}$ as the face of the $W_{S''}$-permutahedron attached to the vertices labeled by the elements in $w''W_{S''}$.

The cell structure of the Davis complex given here is slightly different from the original complex given by Davis in [Dav83]. However, this description will help us to describe the *Salvetti complex* in the following section.

3.2 Artin Groups

Artin groups, which are closely related to Coxeter groups, are introduced in this section. The first ever mention of Artin groups in the literature appears in [Del72] and [BS72]. The general theory of Artin groups can be found in [McC17], a survey article by McCammond. We provide all necessary definitions to make this relationship concrete, including the introduction of the group $\textsc{Art}(\widetilde{A}_2)$, the central group of focus in this dissertation.

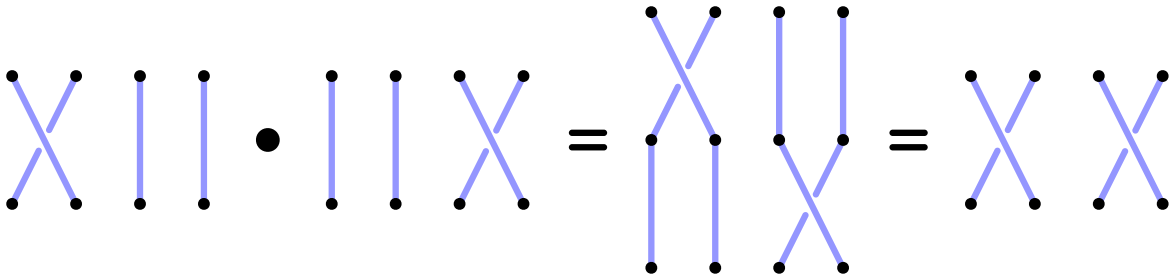


Figure 3.6: The product of two braids with 4 strands where the braid on the left corresponds to the top braid in the concatenation of two braids

The n-strand braid group can be informally described using pictures like that in Figure 3.6, where n braided strings in Euclidean 3-space are anchored at their top and bottom at n-distinguished points. These strings may not intersect, and the natural height function on the strands has no local maxima or minima. We formalize this definition below.

Definition 3.2.1. (The Braid Group) In what follows we give two definitions of Artin's braid groups, beginning with a topological one. Fix an integer n. Let $p_1, p_2, \ldots, p_n$ be n distinguished distinct points in $\mathbb{R}^2$. Let $(f_1, f_2, \ldots, f_n)$ be an n-tuple of functions $f_i : [0,1] \to \mathbb{R}^2$ such that $f_i(0) = p_i$ and $f_i(1) = p_j$ for some $j \in [n]$ and the images of $f_i(t)$ are disjoint for all $t \in [0,1]$.

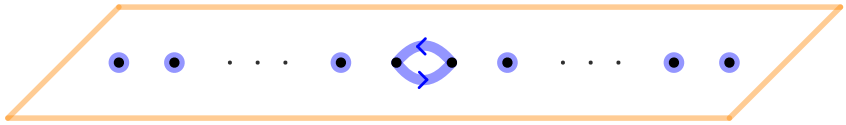


Figure 3.7: The images of f_i for a standard generator, σ_i, of B_n

Define the *strands* to be the n paths from $[0,1] \to \mathbb{R}^2 \times [0,1]$ given by $t \to (f_i(t), t)$. The union of these n strands make up a representative of a braid and a *braid* is an isotopy class of a representative of a braid.

The n-strand *braid group*, denoted $\textsc{Braid}_n$, is the group of braids with n-strands. The identity braid is represented by the n-paths such that $f_i(t) = p_i$ for all $i \in [n]$. The product of two braids represented by $(f_1(t), f_2(t), \ldots, f_n(t))$ and $(g_1(t), g_2(t), \ldots, g_n(t))$

is represented by their *concatenation*. This product is given by

$$(f \cdot g)_i(t) = \begin{cases} f_i(2t) & 0 \le t \le \frac{1}{2} \\ g_j(2t-1) & \frac{1}{2} \le t \le 1 \end{cases} \quad \text{where } f_i(1) = p_j.$$

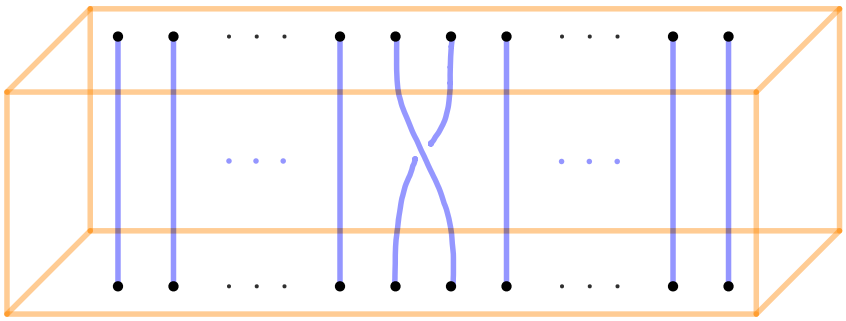


Figure 3.8: The standard generator σ_i of B_n

Notice that each braid determines a permutation on the points $p_1, p_2, \ldots, p_n$, which induces a surjection $\textsc{Braid}_n \to \textsc{Sym}_n$.

The braid group $\textsc{Braid}_n$ can be defined by the following presentation:

$$\textsc{Braid}_n = \left\langle \sigma_1, \sigma_2, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_t\sigma_s = \sigma_s\sigma_t & \text{if } |t-s| > 1 \\ \sigma_t\sigma_s\sigma_t = \sigma_s\sigma_t\sigma_s & \text{if } |t-s| = 1 \end{array} \right\rangle$$

Notice that adding the relations $\sigma_i^2 = 1$ to the presentation of $\textsc{Braid}_n$ yields the standard presentation for the symmetric group $\textsc{Sym}_n$. The symmetric group on n letters and the n-strand braid group are examples of Coxeter groups and Artin groups, respectively. In what follows we discuss the defining properties of these classes of groups.

Remark 3.2.2. (The Braid Monoid) The monoid inside of the braid group generated by $\sigma_1, \sigma_2, \ldots, \sigma_{n-1}$ is called the *braid monoid*. The braid monoid has the same presentation as the braid group given in Definition 3.2.1 but lacks the inverse elements in the group.

Definition 3.2.3. (Artin Group) Given a Coxeter diagram Γ, we can define the Artin group, denoted $\textsc{Art}(\Gamma)$, where each vertex of Γ represents a generator of the Artin group subject to the following relation: if $(ab)^{M(a,b)} = e$ is a relation in $\textsc{Cox}(\Gamma)$, then $\textsc{Art}(\Gamma)$ has a relation that equates the two length $M(a,b)$ words that alternate between a and b. For example, if in $\textsc{Cox}(\Gamma)$, $a, b \in S$ where $M(a,b) = 4$, the relation given by $(ab)^4 = e$ in $\textsc{Cox}(\Gamma)$ becomes the relation $abab = baba$ in $\textsc{Art}(\Gamma)$. The group $\textsc{Art}(\Gamma)$ is said to be *spherical* or *Euclidean* when $\textsc{Cox}(\Gamma)$ is spherical or Euclidean.

Example 3.2.4. (The 4-strand Braid Group) In the way that the symmetric group $\textsc{Sym}_n$ was our example of choice for a Coxeter group, the n-strand Braid group is the favored example of an Artin group. Let Γ be the Coxeter diagram given below.

$\Gamma = A_3 =$ 

We saw that this gives rise to the group presentation

$$\textsc{Sym}_4 = \textsc{Cox}(A_3) = \left\langle \sigma_1, \sigma_2, \sigma_3 \;\middle|\; \sigma_1^2, \sigma_2^2, \sigma_3^2,\ (\sigma_1\sigma_2)^3, (\sigma_2\sigma_3)^3,\ (\sigma_1\sigma_3)^2 \right\rangle.$$

By Definition 3.2.3,

$$\textsc{Braid}_4 = \textsc{Art}(A_3) = \langle \sigma_1, \sigma_2, \sigma_3 \;|\; \sigma_1\sigma_3 = \sigma_3\sigma_1,\ \sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2,\ \sigma_2\sigma_3\sigma_2 = \sigma_3\sigma_2\sigma_3 \rangle$$

Example 3.2.5. ($\textsc{Art}(\widetilde{A}_2)$) The later chapters will focus on a particular Euclidean Artin group of type $\widetilde{A}_2$. Referring to Figure 3.4, we see that the Coxeter diagram of type $\widetilde{A}_2$ is represented by an equilateral triangle with three vertices and three unlabeled edges. If we label the vertices a, b, c, the Coxeter diagram gives the following presentation for the Coxeter group of type $\widetilde{A}_2$

$$\textsc{Cox}(\widetilde{A}_2) = \left\langle a, b, c \;\mid\; a^2, b^2, c^2,\ (ab)^3, (bc)^3, (ac)^3 \right\rangle.$$

Hence, by Definition 3.2.3, the group presentation for the Euclidean Artin group of type $\widetilde{A}_2$ is given by

$$\textsc{Art}(\widetilde{A}_2) = \langle a, b, c \;\mid\; aba = bab,\ bcb = cbc,\ aca = cac \rangle.$$

Though the group presentation of the 4-strand braid group and the Euclidean Artin group of type $\widetilde{A}_2$ don't seem to differ by much, we'll soon see that these slight differences make for very different known results about each type of Artin group.

In what follows, we describe the *Salvetti complex* which is a space similar to the Davis complex but for Artin groups. We start by defining *oriented permutahedra*.

Definition 3.2.6. (Oriented W-permutahedra) Let v be a vertex of a polytope P that is a W-permutahedron. There is a unique vertex v' such that the vector from v to v' passes through the center of P. We then orient each edge in the 1-skeleton of P so the dot product with the vector $\vec{vv'}$ is positive and use this to define a height function.

$$\begin{aligned} \textsc{Cox}(\Gamma) &\curvearrowright \textsc{Davis}(\Gamma) \\ \textsc{Cox}(\Gamma) &\curvearrowright \textsc{Salv}(\Gamma) \\ \textsc{Art}(\Gamma) &= \pi_1(\textsc{PSalv}(\Gamma)) \end{aligned}$$

Figure 3.9: A summary of the nice actions of $\textsc{Cox}(\Gamma)$ on spaces and the groups that arise

Definition 3.2.7. (Oriented Davis complex, Salvetti complex) Given a Coxeter diagram Γ, let $\textsc{Davis}(\Gamma)$ be the Davis complex of $\textsc{Cox}(\Gamma)$. Given the vertex set of the Davis complex, the *oriented Davis complex* replaces each W-permutahedron in the Davis complex with multiple copies, one for each possible orientation. The resulting space is known as the *pure Salvetti complex*. The Coxeter group $\textsc{Cox}(\Gamma)$ acts freely on the pure Salvetti complex. The quotient of the pure Salvetti complex by this action is a 1-vertex complex that has one oriented $W_{S'}$-permutahedron for each subset $S' \subset S$ for which $W_{S'} = \langle S' \rangle$ is finite. We refer to this quotient as the *Salvetti complex* and denote it $\textsc{Salv}(\Gamma)$. The fundamental group of the Salvetti complex is precisely $\textsc{Art}(\Gamma)$. A summary of the spaces and groups discussed in this chapter is given in Figure 3.9.

Example 3.2.8. (Pure Salvetti complex of type A_2) The Davis complex for the Coxeter group of type A_2 is a hexagon consisting of 6 vertices and 6 unoriented edges. The pure Salvetti complex has 6 vertices, 12 oriented edges, and 6 hexagons and can be found in Figure 3.10. The quotient will have a single vertex, 2 edges, and a single hexagon.

3.3 Known Results for Coxeter and Artin Groups

Though Coxeter groups and Artin groups are closely related, many more group-theoretic and algorithmic properties are known about the former. In fact, this dissertation

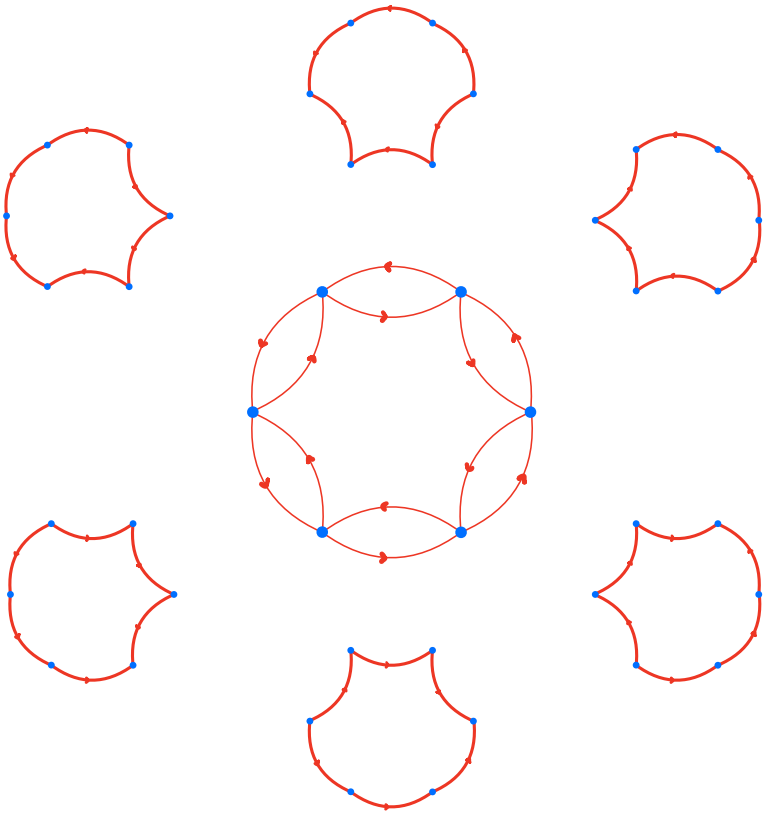


Figure 3.10: Pure Salvetti complex of type A_2

provides some progress in understanding algorithms related the Artin group $\textsc{Art}(\widetilde{A}_2)$. We begin with some results for Coxeter groups.

Coxeter groups are fairly well-understood. As was mentioned previously, all finite Coxeter groups were classified in 1935 by H. S. M. Coxeter. Coxeter groups are also linear. That is, they admit a faithful representation into $\mathrm{GL}(V)$ [Tit13]. Brink and Howlett also showed that Coxeter groups are automatic which implies the word problem for all Coxeter groups is solvable [BH93].

Unlike Coxeter groups, Artin groups are not as well understood. Many conjectures remain open. The following list can be found in [GP12].

Conjecture 3.3.1. *The following conjectures are currently still open.*

1. *Artin groups are torsion free.*

2. *Nonspherical Artin groups have a trivial center.*

3. *Artin groups have a solvable word problem.*

4. *Artin groups satisfy the $K(\pi, 1)$ conjecture.*

Most of the known results related to these conjectures address very specific types of Artin groups and recently, many strides have been made to prove these conjectures for Euclidean Artin groups. In 2017, Jon McCammond and Rob Sulway were able to show that irreducible Euclidean Artin groups are torsion free and centerless [MS17]. In 2019, Paolini and Salvetti proved the $K(\pi, 1)$ conjecture [PS21] for Euclidean Artin groups using the algebraic structures identified by McCammond and Sulway. These results also support that these types of Artin groups are torsion free. However, these conjectures still remain open for Artin groups in general.

All spherical Artin groups are known to have a solvable word problem. McCammond and Sulway also show that Euclidean Artin groups have decidable word problems using their Dual Artin group presentations. Throughout this document, we build on that work and provide a solution to the word problem for the Artin group $\textsc{Art}(\widetilde{A}_2)$ using the standard generating set for the group.

4. Solutions to the Word Problem for Braid_n

In this chapter, we present a series of solutions to the word problem for Braid_n. We proceed by presenting solutions chronologically starting with Artin's solution presented in 1947 [Art47]. Though Frank Garside sought a solution to the conjugacy problem for the braid groups, his work spurred interest in Garside theory which is used to create efficient solutions to the word problem for groups. This chapter also introduces Garside structures as a major tool for solving the word problem in Artin groups. We present a solution that follows Garside's work that was axiomatized, algebraically, by Luis Paris and Patrick Dehornoy. This solution is given by what we call a *fraction normal form* since it relies on a Garside structure on the braid monoid with a generating set that contains the standard generators of the braid group. We also discuss the left-greedy normal form that we use in Part II. We conclude with another solution to the word problem for the braid groups that we'll refer to as a *dual solution*. This solution makes use of a dual presentation for the braid groups found by Birman, Ko, and Lee [Bir98]. The corresponding dual Garside structure was described by David Bessis [Bes03] and by Tom Brady and Colum Watt [BW02].

4.1 Artin's Solution for Braid_n

After introducing the braid groups with what we know as the standard presentation in [Art47], Artin also provides a solution to the word problem for these groups. Algebraically, Artin made use of a map from the n-strand braid group Braid_n to the

symmetric group on n letters SYM_n that sends the generator σ_i to the transposition $(i\ i+1)$. Since braids not in the kernel cannot represent the identity, the word problem can focus on pure braids. In particular, the kernel of this map is called the *pure n-strand braid group*, denoted PBRAID_n, and describes all braids whose strands begin and end in the same position. Using group theoretic properties of the kernel of this map, Artin was able to put braids into a normal form that he called a *combed braid*. Only pure braids are considered in Artin's solution.

Definition 4.1.1. (Combed braid) We illustrate what it means for a braid to be *combed* using the 4-strand braid group. Let $\tau \in \text{PBRAID}_4$. Then, τ can then be represented as word of the form $w_1w_2w_3$ such that w_1 is a word in the standard generating set that always involves the first strand (i.e., the strand starting in the first position at the very beginning of the braid), w_2 is a word that always involves strand 2 (but never strand 1), and w_3 is a word that only involves only strands 3 and 4 (while strands 1 and 2 remain unmoved). In general, the subword w_i will represent a braid such that strand i moves about all strands $j > i$ while all strands $k < i$ remain uninvolved with the remainder of the braid. For a visual example, see Figure 4.3.

According to the above definition, turning an arbitrary pure braid into a combed braid results in each strand, from left to right, moving one at a time, while the other strands remain unmoved. Algebraically, this is accomplished by considering a pure n-strand braid β and letting $f(\beta)$ represent the pure $n-1$-strand braid obtained by forgetting the first string. We still take $f(\beta)$ to be in P_n via inclusion. Then, β and $f(\beta)$ represents the same braid except with the last strand of $f(\beta)$ changed so that it has no interaction with

the other strands. If we let K be the kernel of f, this gives a split, short exact sequence

$$1 \to K \to P_n \to P_{n-1} \to 1.$$

The kernel is free and corresponds to the fundamental group of the $n-1$ times punctured disk. This process can then be iterated on P_{n-1} and so forth to obtain a combed braid.

In what follows, we provide an example of a braid that we put in normal form, keeping track of the braid as a word written with respect to the standard generating set of the 4-strand braid group. The standard presentation of the 4-strand braid group is given again below.

$$\textsc{Braid}_4 = \langle \sigma_1, \sigma_2, \sigma_3 \ \mid \ \sigma_1\sigma_3 = \sigma_3\sigma_1,\ \sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2,\ \sigma_2\sigma_3\sigma_2 = \sigma_3\sigma_2\sigma_3 \rangle$$

Consider the braid given in Figure 4.1 (a). This braid, written as a product of the above generators and their inverses is given by

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

In this example, we use a 4-strand pure braid, that is, a braid whose strands begin and end in the same position. For ease of description, we've colored each strand as well. We see the red strand moves throughout the braid but there are two crossings circled in Figure 4.1 (b), that do not involve the first strand indicating that we must manipulate the braid so that those crossings occur below any other crossings involving the red strand. Notice that the braid can be manipulated in a way that these crossings merely slide past the

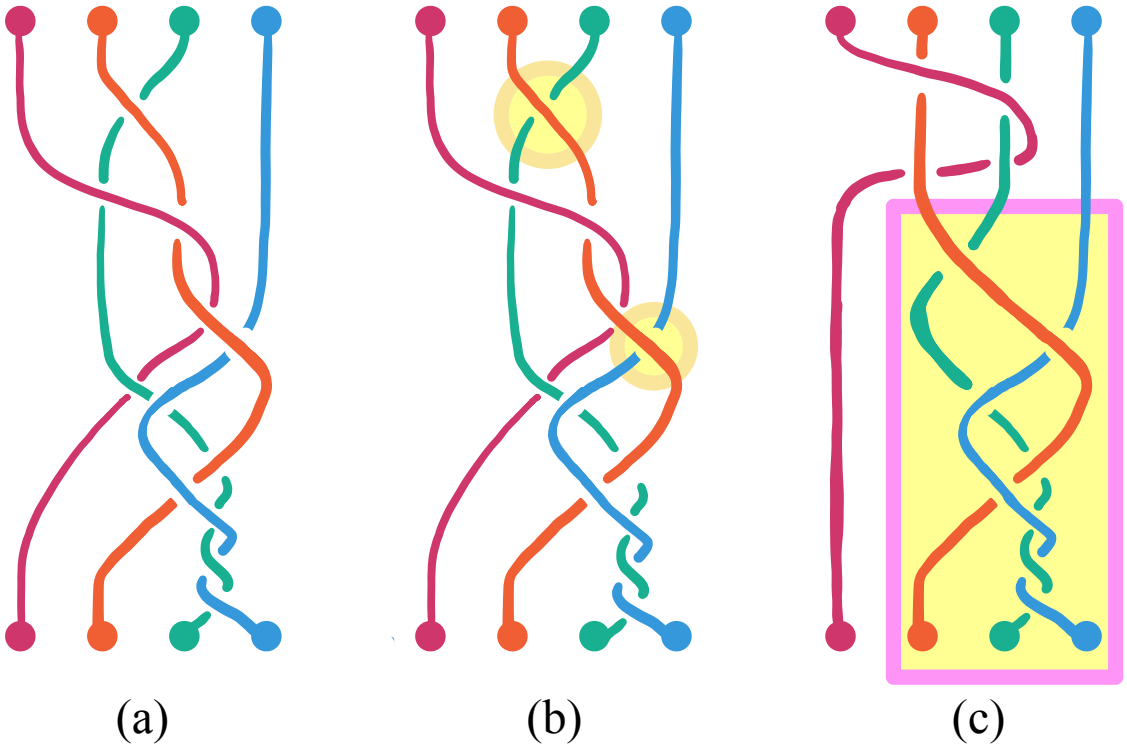


Figure 4.1: The combing of a 4-strand braid

crossings involving the red strand. This visual manipulation is performed algebraically by making use of the relations $\sigma_1\sigma_3 = \sigma_3\sigma_1$ and $\sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2$. This results in the braid pictured in Figure 4.1 (c), which can be represented algebraically by

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

Notice that the first strand moves throughout the braid before all other strands. We can now proceed to focus on just the remaining 3 strands or the portion of the braid boxed in Figure 4.1 (c). Again, we wish to manipulate this braid so that the orange strand, that is, the left-most strand in 4.2 (a), moves throughout the braid before the other strands do. There is but one crossing that doesn't allow the orange strand to do so and it is

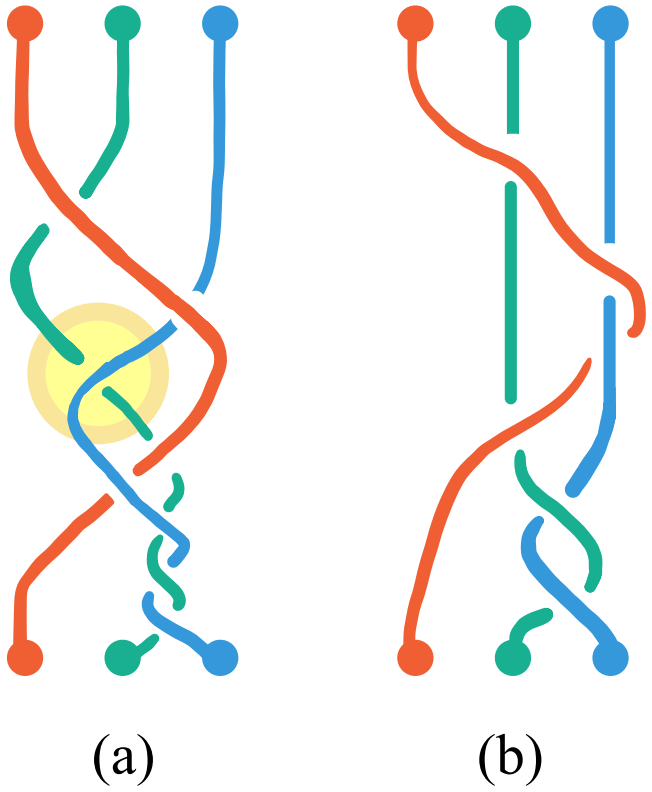


Figure 4.2: The combing of a 4-strand braid, continued

highlighted in Figure 4.2 (a). Unlike the previous means of manipulation, this crossing cannot be slid under the crossings of the orange strand below it. Moving this crossing down requires the use of the braid relations given in the presentation of the group above. In particular, the relation $\sigma_2\sigma_3\sigma_2 = \sigma_3\sigma_2\sigma_3$ allows us to rewrite the subword $\sigma_2^{-1}\sigma_3^{-1}\sigma_2$ as $\sigma_3\sigma_2^{-1}\sigma_3^{-1}$. We see in Figure 4.1 (b), this portion of the braid rewritten algebraically as $\sigma_2\sigma_3\sigma_3\sigma_2^{-1}\sigma_3\sigma_3$.

This allows us to rewrite the original braid from Figure 4.1 (a) as the braid

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

is pictured in Figure 4.3.

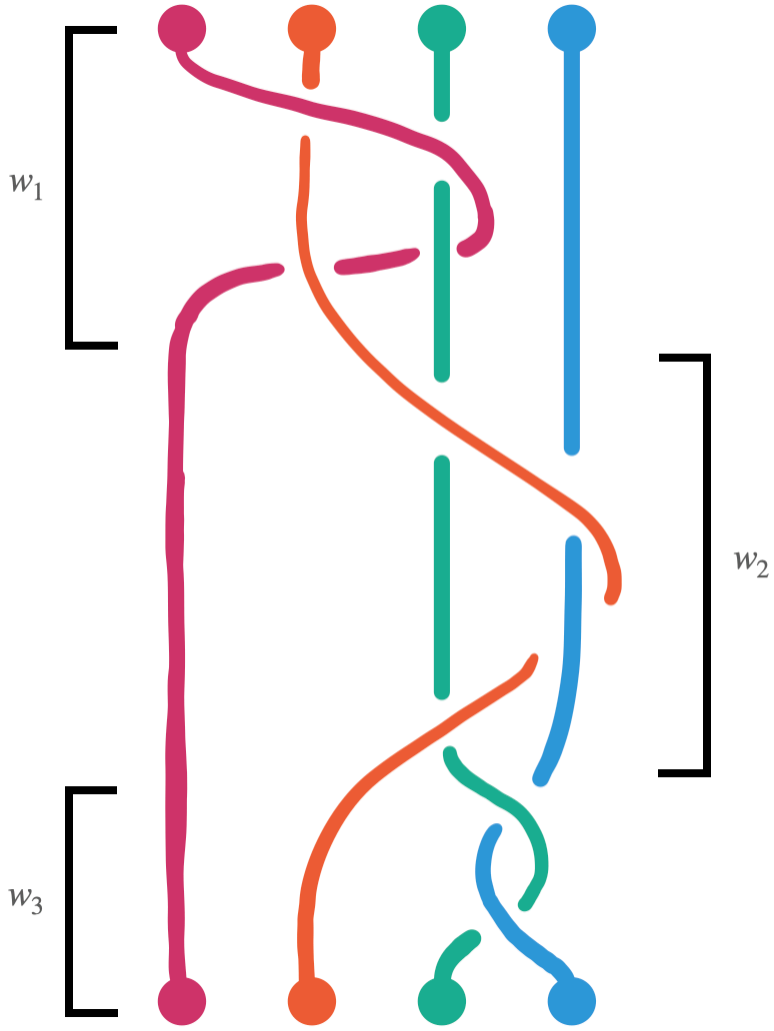


Figure 4.3: A combed, 4-strand braid

The process described above was Artin's algorithm that solves the word problem for the braid groups with respect to the standard generating set. However, Artin provides this solution in [Art47] accompanied by the following warning, "although it has been proved that every braid can be deformed into a similar normal form the writer is convinced that any attempt to carry this out on a living person would only lead to violent protests and discrimination against mathematics. He would therefore discourage such an experiment."

Artin's warning is related to the fact that his algorithm is exponential in the length of the word expressed in terms of the standard generators and their inverses. In the next section, we analyze solutions to the word problem for the braid groups that relied on algorithms that are much more efficient.

4.2 Posets, Garside Groups and Garside Structures

This section provides the necessary definitions for Garside groups and Garside structures, starting with basic terminology related to posets. We then define the Garside monoid and, by passing to its group of fractions, define the Garside group. An in-depth analysis of Garside's work and its many applications can be found in [Deh15].

Definition 4.2.1. (Posets) Let P be a partially ordered set. We say that P is *bounded* if it contains both a minimum and a maximum element. By restricting the partial order of P onto a subset $Q \subset P$ we get a *subposet* structure on Q. If Q is a subposet where any two of its elements are comparable with respect to the partial order, we say that Q is a *chain*. If Q is a finite chain, then its length is given by $|Q| - 1$. If Q is a finite chain and Q is bounded, we refer to the maximum and minimum elements as *endpoints*. If Q is a finite chain that is not a subposet of a strictly larger finite chain with the same endpoints, we say that Q is *saturated*. The poset P is *graded* if for all $x \leq y$, a saturated chain always exists from x to y and the length of all such saturated chains are the same. We refer to intervals of the form $[x, x]$ as *trivial* intervals and intervals $[x, y]$ such that $x < y$ is a saturated chain of length 1 are what we call *covering relations* of P.

Definition 4.2.2. (Lattices) Let Q be a subset of a poset P with partial order $\leq$. We say that $p \in P$ is a lower bound of Q if $p \leq q$ for all $q \in Q$. If the set of all lower bounds for Q has a maximum element, we call that element the greatest lower bound or *meet* of Q, denoted $\bigwedge Q$. We define analogously the least upper bound or *join* of Q, denoted $\bigvee Q$. If Q contains just two elements u, v we use $u \wedge v$ and $u \vee v$ to denote the meet and join, respectively. We say that P is a *lattice* if every pair of elements have a meet and join.

Definition 4.2.3. (Garside Groups and Garside Structures) Let M be a monoid. An element of M is *irreducible* if it is not the product of two non-units. We say that M is *atomic* if every element that is not a unit can be written as a product of irreducible elements. M is *left cancellative* if for any $a \in M$, $ab = ac$ implies $b = c$ for all $b, c \in M$. Similarly, M is *right cancellative* if for any $a \in M$, $ba = ca$ implies $b = c$ for all $b, c \in M$. We say M is cancellative if M is both left and right cancellative. The element a is a *left divisor* of b if there exists some $c \in M$ such that $ac = b$. Given that M is atomic, the left divisibility relation, which we will denote $\leq_L$, is a partial order. Right divisors are defined analogously and the partial order of right division is denoted $\leq_R$. A monoid M is a *Garside monoid* if it has the following defining properties:

- M is atomic, cancellative, and finitely generated;
- There exists an element called the *Garside element*, denoted Δ, such that $\{a \in M : a \leq_L \Delta\} = \{a \in M : a \leq_R \Delta\}$; and
- $(M, \leq_L)$ and $(M, \leq_R)$ are lattices.

Due to a well known result of Ore [Ore31], any Garside monoid embeds into the corresponding *Garside group* by passing to the group of fractions.

Example 4.2.4. (The 4-Strand Braid Group) In what follows, we use permutations to describe certain braids in the 4-strand positive braid monoid. In particular, if we ignore how the strands of a braid twist and cross, a braid on 4-strands will describe a permutation of 4 elements. For example, the generator σ_i can be represented by the

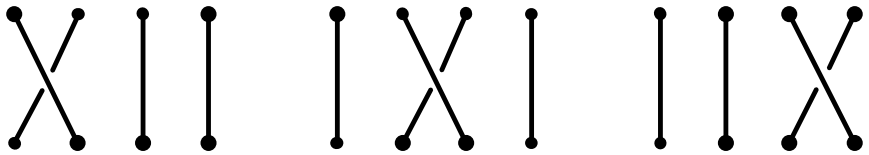


Figure 4.4: Positive half-twists of adjacent strands $\sigma_1 = (1\ 2), \sigma_2 = (2\ ,3), \sigma_3 = (3\ 4)$

transposition $(i, i+1)$. For these generators, if strand i crosses over strand $i+1$, we refer to those braids as a *positive half-twist of adjacent strands*. The three examples of positive half twists of adjacent strands in the 4-strand braid group are can be found in Figure 4.4. We call any braid created by taking products of positive half-twists of adjacent strands a *positive braid*. An example of a positive braid is seen in Figure 4.5. This braid is also known as the *full, half-twist* and can be represented by the permutation $(14)(23)$. The set of all left divisors of the full, half-twist happens to be equal to the set of right divisors of the full, half-twist. Moreover, labelling these divisors with the permutation that describes them reveals the finite set of 24 elements represented by the vertices of the permutahedron pictured in Figure 4.6. The 1-skeleton of the permutahedron can be made into a poset with a height function that is discussed in Section 4.4. An edge in the permutahedron connects two vertices v_g and v_h if there is a positive half-twist of

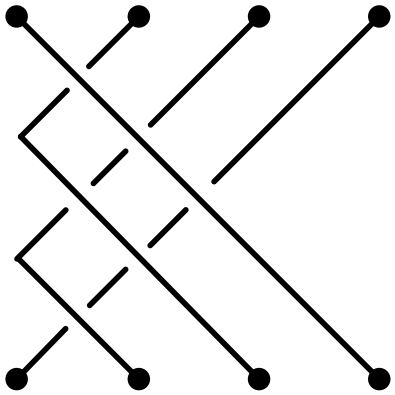


Figure 4.5: The full, half-twist (14)(23) in the 4-stand braid group

adjacent strands $(i, i+1)$ such that $g \cdot (i, i+1) = h$. Since the three positive half-twists of adjacent strands that generate the braid monoid are amongst the 23 nontrivial elements represented by the vertices in the permutahedron, these 23 elements will also generate the braid monoid. Hence, with the Garside element given by the the full half-twist, the braid monoid is a Garside monoid, giving implies the braid group the structure of a Garside group.

4.3 Garside Structures and the Word Problem

Using the defining properties of a Garside group, it is possible to give a rather efficient solution to the word problem. In particular, because the group G is the group of fractions of the Garside monoid M, the word problem for G can be reduced to the word problem in M. We start this section by discussing exactly how a solution to the word problem can be produced relying on the defining properties of a Garside monoid.

Definition 4.3.1. (Hasse Diagram) Suppose G is a Garside group with Garside element Δ. Let S be a generating set for G such that $S = \{s \in G \mid s \leq_L \Delta\}$. That is, all elements

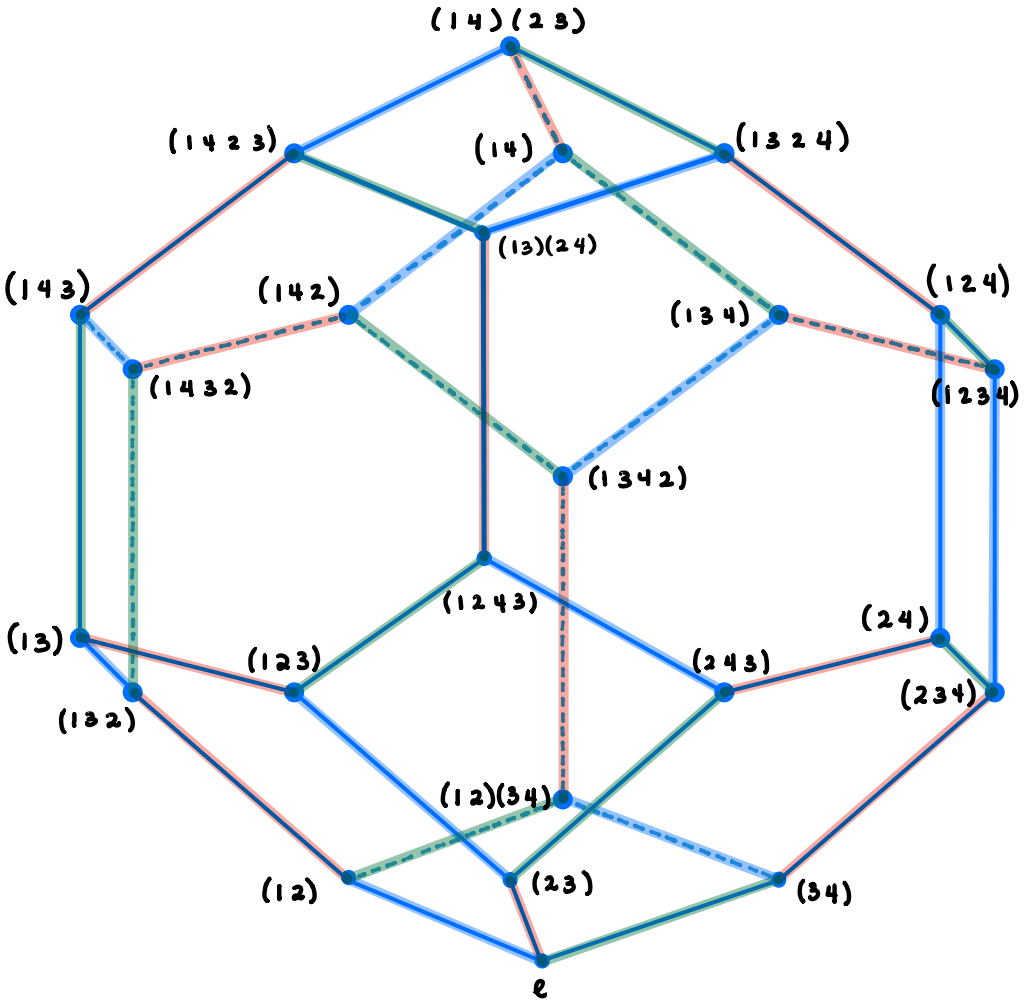


Figure 4.6: The set of left divisors of the full, half-twist

of S are left divisors of the Garside element Δ. Because G is a Garside group, we know that the elements of S generate G and that S is a poset with respect to left division. We use this partial order to define the *Hasse diagram* which has a vertex for every element in the poset and an edge from a vertex s_i to a vertex s_j if and only if $[s_i, s_j]$ is a covering relation (see Definition 4.2.1). Moreover, the diagram is drawn so that the lower end of an edge corresponds to the smaller element in the covering relation.

The permutahedron in Figure 4.6 could also serve as a Hasse diagram for the identity element and the 23 element generating set for the 4-strand braid group with some slight modifications. Let e denote the identity element and Δ denote the full half-twist. Then we could label each edge from s to t with the positive half twist of adjacent strands, $(i, i+1)$ such that $t = s \cdot (i, i+1)$. In Figure 4.6, instead of labeling each edge, we color each edge according to the permutation that you would right multiply by to get from one permutation to the next with a blue edge, red edge, and green edge representing right multiplication by (12), (23), and (34), respectively. In this context, each interval could also be represented by a word. Trivial intervals will be labeled by the empty word whereas the interval $[e, (24)]$ could be represented using the words $\{(23)(34)(23), (34)(23)(34)\}$ which can be seen in Figure 4.6 as increasing paths from the vertex labeled e to the vertex labeled (24).

Definition 4.3.2. (Reversible Words) All Garside groups have reversing processes which contribute to solutions to the word problem. A word w in $(S \cup S^{-1})^*$ is said to be *reversible* to the word w' if we can get from the word w to the word w' by replacing a positive, negative subword such as xy^{-1} with a negative, positive subword such as $(x')^{-1}y'$ with $x, y, x', y' \in S$. This reversing process can continue as long as w contains a subword of the form xy^{-1}. When a subword of this form no longer exists, the reversing process rewrites w as a word of the form $u^{-1}v$ where both u and v are positive words, with $u, v \in S^*$. In other words, all words w can be rewritten as the product of a negative subword and a positive subword. In Garside groups, the reversing process always terminates [Deh15].

We give a presentation and use this fact to describe a solution to the word problem for the 4-strand braid group.

4.4 A Garside Structure for Braid_n

In this section, we give a solution to the word problem using the standard Garside structure built from the standard generating set of the braid group. This solution is due to Frank Garside who produced a solution to the conjugacy problem for braid groups in 1968. Although the solution discussed here can be used for the n-strand braid group, we use the 4-strand braid group to discuss the particulars of the Garside structure and how algorithm relies on that structure.

The standard Garside generators of the 4-strand braid group are indexed by 23 of the vertices of the permutahedron in Figure 1.3, representing nontrivial positive simple braids. The bottom vertex indexes the identity, which we exclude from our generating set. The algorithm is programmed in Python using the computer algorithm Sage. To work with braids in Sage, we think of the vertices of the permutahedron as being numbered from v_0 to v_{23} with v_0 representing the identity and v_{23} representing Δ, the full, half-twist. In the code, we will use the subscript of these vertices to represent the positive simple braid that was previously represented by the permutations that labeled the vertices of the permutahedron. See Figure 4.7 for comparison. Since positive integers represent the positive braids that serve as our generators, a negative integer, $-k$, will represent the inverse of the positive braid k. For example, the braid pictured in Figure 4.9 can be represented by the product of permutations $(142)(1342)(1432)(132)(123)$ or by the string of integers $[18, 12, 14, 8, 7]$.

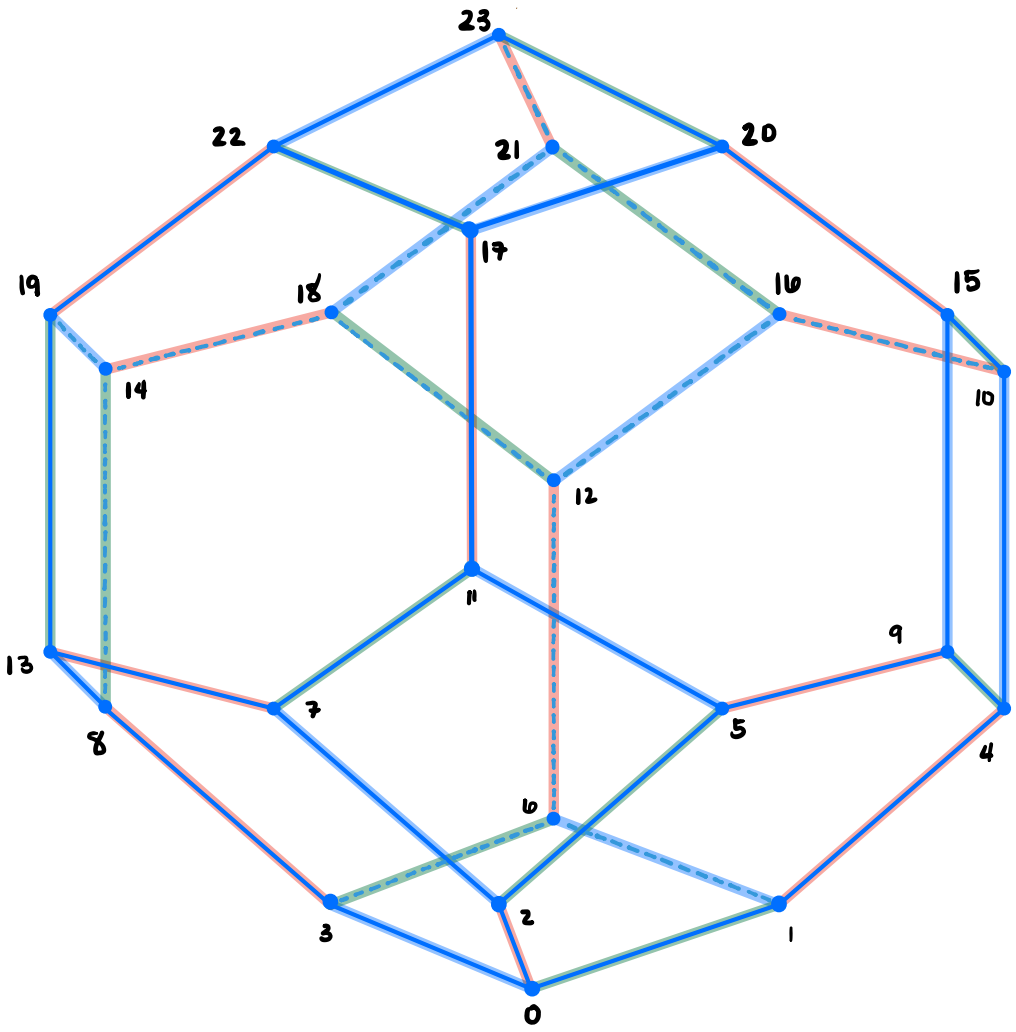


Figure 4.7: The generators of the 4-strand braid group, relabeled as integers

The algorithm is made up of four major functions. We first describe how to put an arbitrary pair of generators in normal form. We mentioned in Section 4.2 that the 1-skeleton of the permutahedron can be made into a poset. To do so, the braid τ can be written as a product of positive half-twists of adjacent strands. The height of τ is equal to the number of positive half-twists of adjacent strands in the product. Equivalently, τ can be represented by the vertex v_k in the permutahedron whose height is given by the length of any shortest path from v_0 to v_k. If each edge in the 1-skeleton of the permutahedron

is oriented by the heights of its endpoints, then this becomes a Hasse diagram that turns into a lattice.

Given $G = \langle S \mid R \rangle$, let $a, b \in S$. We define the *right complement* of a to be the unique braid $rc(a)$ such that $a \cdot rc(a) = \Delta$. Similarly, the *left complement* of a is the braid denoted $lc(a)$ such that $lc(a) \cdot a = \Delta$. The product $a \cdot b$ is put into normal form using the following rewriting process:

$$a \cdot b = a' \cdot b'$$

where $a' = a \cdot (rc(a) \wedge b)$ and $b' = (rc(a) \wedge b)^{-1} \cdot b$. This rewriting process is also illustrated in Figure 4.8. Notice that

$$\text{height}(a) \leq \text{height}(a \cdot (rc(a) \wedge b)) \text{ and } \text{height}(b) \geq \text{height}(b')$$

and the inequality is strict whenever $a \neq a'$. This means that the number assigned to a' is larger than the number assigned to a and similarly, the number assigned to b' is less than the number assigned to b. In particular, the pair (a', b') is lower in reverse dictionary order. After storing all of the necessary terms involved in this rewriting process, the function A.16 performs the above rewriting process.

Example 4.4.1. (Algorithm 1: Product of two generators in normal form) For example, consider the product of braids represented by the integers $[11] \cdot [15]$. We have

$$rc([11]) = [11] \text{ and } [11] \wedge [15] = [5].$$

Hence,

$$[11] \cdot [15] = ([11] \cdot [5]) \cdot [7] = [22] \cdot [7].$$

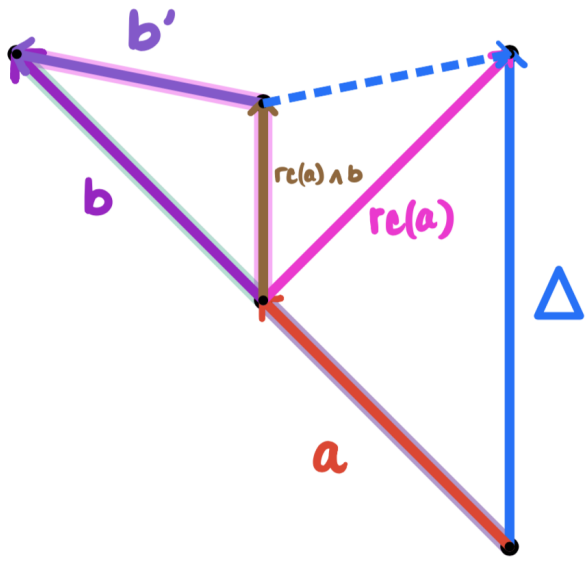


Figure 4.8: Rewriting $a \cdot b$ as $a' \cdot b'$ where $a' = a \cdot (rc(a) \wedge b)$ and $b' = (rc(a) \wedge b)^{-1} \cdot b$

Using the previous rewriting process, we now look to put an arbitrary positive word in normal form. Given an arbitrary product of generators, we'd like to iteratively apply the previous rewriting process. We must keep in mind that upon rewriting a pair of generators within a word, you must ensure that the previously rewritten pair is still in normal form. We rely on the function A.17 which performs the above rewriting process at a desired location in a word. Then, we can accurately rewrite pairs of generators in a positive word, from left to right, and with every updated pair we can backtrack to ensure the word remains in normal form before proceeding. The function that puts positive words in normal form is found in A.18.

Example 4.4.2. (Algorithm 2: A positive word in normal form) Using this function on the word represented by the string of integers $[14, 8, 7]$, we yield the following output which details the step-by-step rewriting process, indicating that $[22, 7, 0]$ is an equivalent word in normal form.

```
input: RedPosWord([14,8,7])
output: ([22,0,7], -1)
([22,7,0], 0)
([22,7,0], 1)
([22,7,0], 2)
[22,7,0]
```

Putting an arbitrary negative word in normal form can be done using the previously defined functions. Since the inverse of a negative word is a positive word written in reverse, the function putting a positive word in normal form can be applied to the inverse. Once the inverse is in normal form, we simply take the inverse again to produce a negative word in normal form. This is done using the function A.23.

Example 4.4.3. (Algorithm 3: A negative word in normal form) For example, consider the negative word given by $[-15, -11]$. This function yields the following output which first takes the inverse of the word, yielding a positive word, puts that word in normal form (see Example 4.4.1), and concludes by taking the inverse of this new word in normal form.

```
input: RedNegWord([-15,-11])
```

```
output: ([22,7], -1)

[22,7,0]
```

The above sample output should not come as a surprise since the inverse of the word $[-15, -11]$ is given by $[11, 15]$ which was put into normal form in Example 4.4.1.

We conclude with a function that performs the word reversing process (see Definition 4.3.2) so that the algorithm works in the braid group and not just the braid monoid. We start by rewriting the product of a generator and the inverse of a generator represented by $i \cdot -j$ as an equivalent product $-i' \cdot j'$ for every pair of generators i and j. We store this information in a matrix to be called upon by the function A.20. This function uses a similar "spot" function to the one mentioned previously to keep track of where the negative letters appear in the rewriting process. The rewrites conclude when an arbitrary word is rewritten as the product xy where x is a negative subword and y is a positive subword.

Example 4.4.4. (Algorithm 4: Word reversing algorithm) In the sample output given below, we start by inputting an arbitrary word $[5, -16, 12, -5, 10, -18, -4, 8, 7]$. At each step, this algorithm finds the aforementioned positive, negative pair and rewrites it as a new negative, positive pair. This rewriting process continues until it produces an equivalent word represented by a string of nonpositive integers followed by a string of nonnegative integers. In this case, we get an equivalent word given by $[-15, -11, 0, 0, 14, 0, 0, 8, 7]$.

```
input: negposword([5,-16,12,-5,10,-18,-4,8,7])

output: [-15, 8, 12, -5, 10, -18, -4, 8, 7]

[-15, 8, -8, 11, 10, -18, -4, 8, 7]
```

```
[-15, 0, 0, 11, 10, -18, -4, 8, 7]

[-15, 0, 0, 11, -13, 4, -4, 8, 7]

[-15, 0, 0, -11, 14, 4, -4, 8, 7]

[-15, 0, -11, 0, 14, 4, -4, 8, 7]

[-15, -11, 0, 0, 14, 4, -4, 8, 7]

[-15, -11, 0, 0, 14, 0, 0, 8, 7]
```

Algorithms 1 – 4 make up the algorithm that puts arbitrary words into normal form. We use algorithms 2 and 3 on the negative and positive subwords that appear as outputs of algorithm 4. Then, the only pair that needs to be checked is the middle negative, positive pair of generators. If that pair needs no update, we have a word in normal form. Should the pair need an update, the negative and positive words may need to be put into normal form yet again but this process will terminate and can be done with the function A.26. We mention that although this example uses the 4-strand braid group, the algorithm is programmed to provide a solution to the word problem for the n-strand braid group. The algorithm and all of its culminating functions can be found in Appendix A.1.

Example 4.4.5. (An arbitrary word in normal form) In this example, we provide several examples of braids put in normal form. Take first, the braid given by Figure 4.9. This braid is an example of a trivial braid that is drawn as a product of 5 nontrivial braids. This braid can also be represented by the word

$$[5, -16, 12, -5, 10, -18, -4, 8, 7].$$

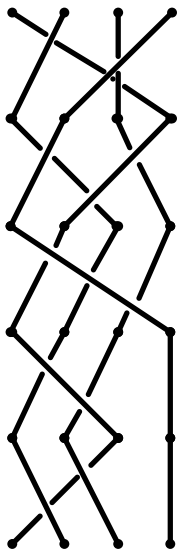


Figure 4.9: A trivial braid

According to the algorithms described above, we confirm that this braid is indeed the trivial braid with the following:

```
input: RNormForm([5,-16,12,-5,10,-18,-4,8,7])

output: [-15, 8, 12, -5, 10, -18, -4, 8, 7]

[-15, 8, -8, 11, 10, -18, -4, 8, 7]

[-15, 0, 0, 11, 10, -18, -4, 8, 7]

[-15, 0, 0, 11, -13, 4, -4, 8, 7]

[-15, 0, 0, -11, 14, 4, -4, 8, 7]

[-15, 0, -11, 0, 14, 4, -4, 8, 7]

[-15, -11, 0, 0, 14, 4, -4, 8, 7]

[-15, -11, 0, 0, 14, 0, 0, 8, 7]

negative subword

[-15,-11]
```

```
([22, 7], -1)

positive subword

[14,8,7]

([22, 0, 7], -1)

([22, 7, 0], 0)

([22, 7, 0], 1)

([22, 7, 0], 2)

newNPword

[-7, -22, 22, 7, 0]

Update (neg,pos) Pair

[-7, 0, 0, 7, 0]

remove 0s

[-7, 7]

Update (neg,pos) Pair

[0, 0]

remove 0s

[]
```

We now consider the braid that is similar to the braid given above with the exception of a single sign change described by the following string of integers:

$$[5, -16, 12, 5, 10, -18, -4, 8, 7]$$

We see that this braid, in normal form is not the trivial braid but it can be represented as a word of length 4.

```
input: RNormForm([5,-16,12,5,10,-18,-4,8,7])

output: [-15, 8, 12, 5, 10, -18, -4, 8, 7]

[-15, 8, 12, 5, -13, 4, -4, 8, 7]

[-15, 8, 12, -15, 14, 4, -4, 8, 7]

[-15, 8, -12, 7, 14, 4, -4, 8, 7]

[-15, -1, 0, 7, 14, 4, -4, 8, 7]

[-15, -1, 0, 7, 14, 0, 0, 8, 7]

negative subword

[-15,-1]

([9, 7], -1)

positive subword

[7,14,8,7]

([7, 14, 8, 7], 1)

([7, 22, 0, 7], 0)

([19, 14, 0, 7], -1)

([19, 14, 0, 7], 2)

([19, 14, 7, 0], 1)

([19, 21, 0, 0], 0)

([19, 21, 0, 0], 1)

([19, 21, 0, 0], 2)
```

```
([19, 21, 0, 0], 3)

newNPword

[-7, -9, 19, 21, 0, 0]

Update (neg,pos) Pair

[-7, -4, 14, 21, 0, 0]

remove 0s

[-7, -4, 14, 21]

reduce new list

negative subword

([4, 7], 1)

positive subword

([22, 10], -1)

newNPword

[-7, -4, 22, 10]
```

Remark 4.4.6. (Standard normal form) The algorithm described in Section 4.4 relies on a normal form for the monoid and then extends that to the group of fractions. Alternatively, and more conventionally, given an arbitrary word in the group of the form $s_1 s_2^{-1} s_3 s_4^{-1} \cdots s_n^{-1}$, one could simply replace each s_i^{-1} with $\Delta^{-1} \cdot lc(s_i)$ and then move the Δ^{-1} to the front using conjugation (i.e. replace $t\Delta^{-1}$ with $\Delta^{-1}t'$ where $t' = \Delta t \Delta^{-1}$).

Once it is in the form $\Delta^k u_1 u_2 \cdots u_n$ with $k \in \mathbb{Z}$ and $u_1 u_2 \cdots u_n$ some positive word, we simply put the positive word in left greedy normal form and absorb any Δ's produced into the Δ^k at the front followed by proper, nontrivial factors of Δ, $u_1, \ldots, u_n$. This is

the *left-greedy normal form* for any word in the Garside group and this is the normal form that we will use in Part II.

4.5 Dual Garside Structure for Braid_n

After Garside's solution in 1968, Brieskorn and Saito investigated all spherical Artin groups through a combinatorial lens in 1972. They provided solutions to both the word problem and the conjugation problem and determined the center of these spherical Artin groups as well [BS72]. In 1998, Dehornoy and Paris showed that braid groups and more generally, all spherical Artin groups are Garside groups and that all Garside groups are biautomatic which therefore, have simple algorithms that solve the word problem [DP99]. In the same year, Birman, Ko, and Lee introduced a new presentation for Braid_n and with it, a new solution to the word problem. In this section we introduce the Birman-Ko-Lee presentation and their solution to the word problem and conclude with a discussion of how this new presentation sparked an interest in dual presentations for Artin groups in general. We start by introducing noncrossing partitions.

Definition 4.5.1. (Partitions) Given a finite set S, a *partition* of S is a collection of pairwise disjoint subsets, called *blocks*, whose union is all of S. Let $[n] = \{1, 2, \ldots, n\}$. The partitions of $[n]$ form a poset with respect to refinement; that is, given partition π and τ, we have $\pi \leq \tau$ if each block of π is contained in a block of τ. With respect to this partial order, this poset, denoted Π_n forms a lattice called the *partition lattice* of rank n. Figure 4.10 contains the partition lattice Π_4.

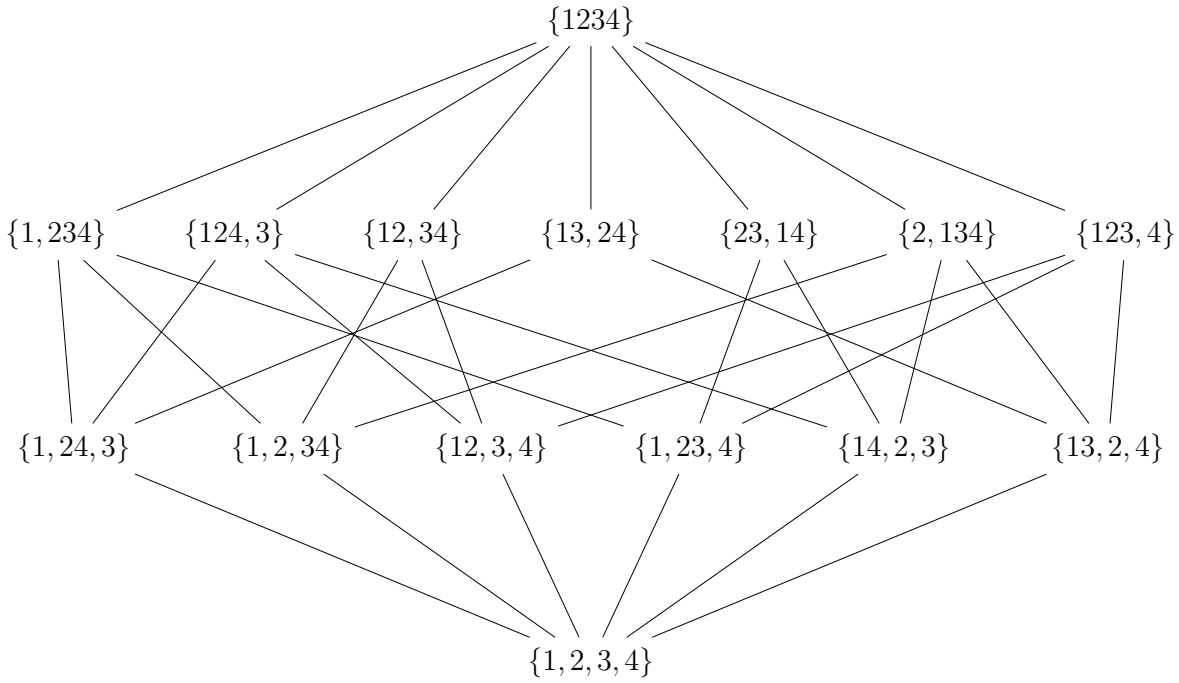


Figure 4.10: The partition lattice Π_4

Before defining a noncrossing partition, we introduce a lens by which we may view these objects. We can identify $[n]$ with a convex regular n-gon D_n whose vertices are labeled counterclockwise by p_j for $j \in [n]$. Then a subset $A \subseteq [n]$ determines a set of vertices $P_A = \{p_j \mid j \in A\}$. The convex polygon associated to A is the convex hull of P_A denoted $\textsc{Conv}(P_A)$.

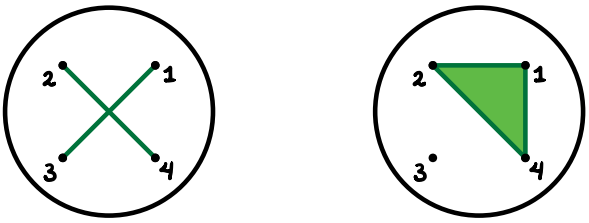


Figure 4.11: The crossing partition $\{13, 24\}$ and the noncrossing partition $\{124, 3\}$ of Π_4

Definition 4.5.2. (Noncrossing Partitions) Using the polygon associated to a subset A of $[n]$, we see that each partition of $[n]$ can be identified with a collection of the convex hulls of its blocks. A partition is *noncrossing* if the convex hulls are pairwise disjoint. See Figure 4.11 for an example of a crossing partition and noncrossing partition of Π_4. We denote the set of all noncrossing partitions as NC_n and we see that NC_n is a subposet of Π_n. The noncrossing partition lattice NC_4 can be found in Figure 4.12.

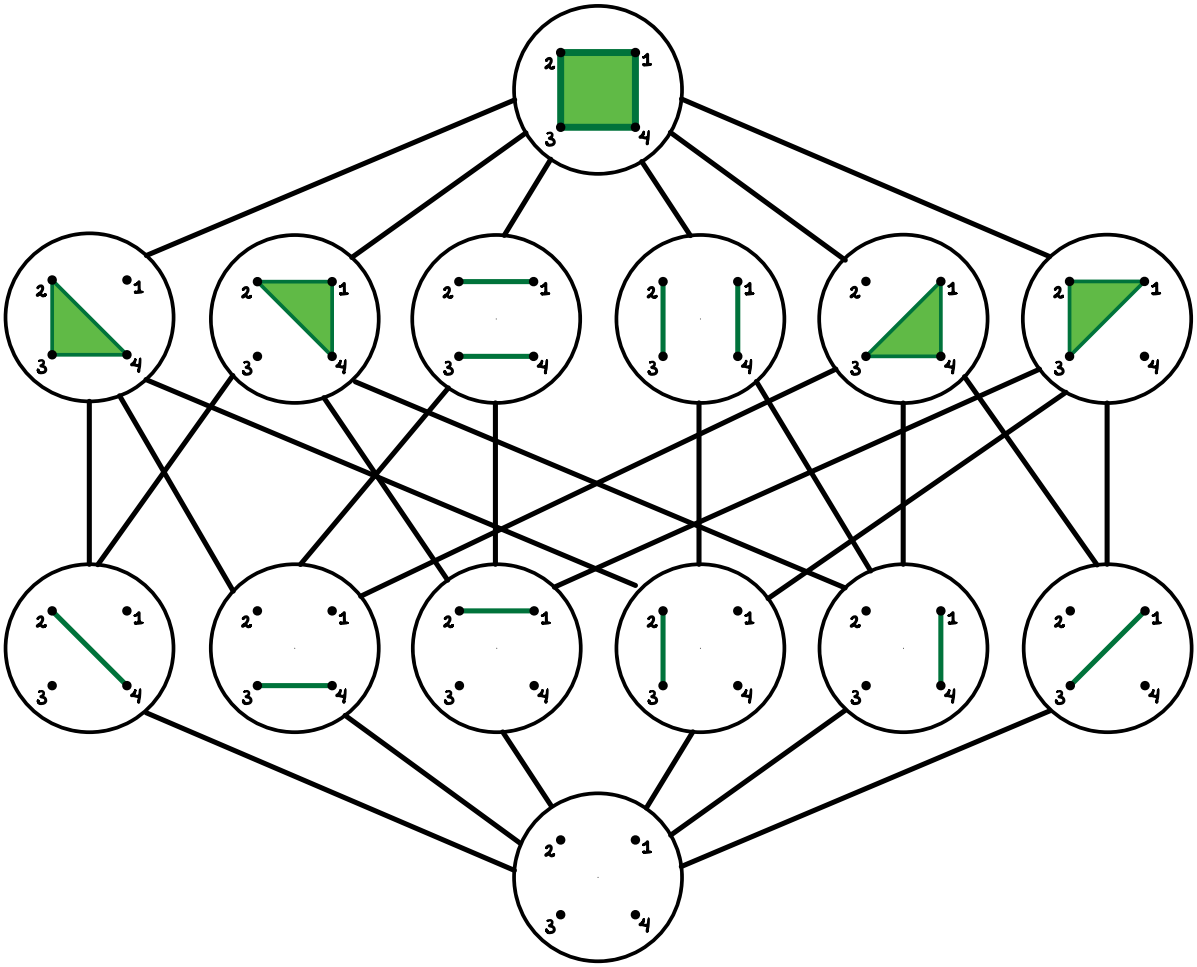


Figure 4.12: The noncrossing partition lattice NC_4

Both NC_n and Π_n are bounded and graded posets. And because both are lattices, unique meets and joins exist for both partitions. In particular, for $\pi, \tau \in \mathrm{NC}_4$, the meet

$\pi \wedge \tau$ is the coarsest common refinement of π and τ whereas the join $\pi \vee \tau$ is the finest *noncrossing* partition that has both π and τ as a refinement.

A noncrossing partition can be interpreted as a permutation and thus, as a braid. To see this, recall that we let D_n represent the regular n-gon with vertices p_j. Given $A \subseteq [n]$ such that $|A| = k > 2$, we define the *standard subdisk* for A to be the convex hull of the vertices in P_A, denoted D_A. Notice that D_A is a k-gon homeomorphic to the 2-dimensional disk. With this convention, D_n is equivalent to $D_{[n]}$. When $|A| = k = 2$, we define D_A to be a slight thickening of the edge so that it is also homeomorphic to the disk. Suppose $A = \{i, j\}$ and let e_{ij} denote the straight line segment connecting p_i and p_j. Taking two copies of the path along e_{ij}, and bend them so that they become injective paths from p_i to p_j with disjoint interiors that bound a bigon in D_n. Should an edge, e_{ij} lie in the boundary of D_n, we keep one of the copies unmoved in the boundary of the bigon. If $A = \{i\}$, then $|A| = k = 1$, and D_A is just the point p_i and we note that in this case, D_A is not homeomorphic to the disk. We note here that the bending of edges described above are chosen so that for all subsets $A, B \subseteq [n]$, the standard subdisks D_A and D_B intersect if and only if the convex hulls of P_A and P_B intersect.

We can now discuss how moving the points in these subdisks allow us to recover braids in $\textsc{Braid}_n$.

Definition 4.5.3. (Rotation braid) Given $A \subset [n]$, we get the *rotation braid* δ_A by taking the vertices in P_A and rotating them, counterclockwise, around the boundary of the subdisk D_A so that each vertex travels along a single edge until it reaches the next vertex. If $|A| = 1$, δ_A represents the identity braid. If $|A| = 2$, then D_A is the bigon

described above. In this case, the braid δ_A is also called a *positive half-twist*. Should $A = [n]$, the rotation braid δ_A is represented by δ_n instead. We note that if $|A| = k$, then $\textsc{Perm}(A)$ is the k-cycle permutation obtained by ordering the elements of A in increasing order.

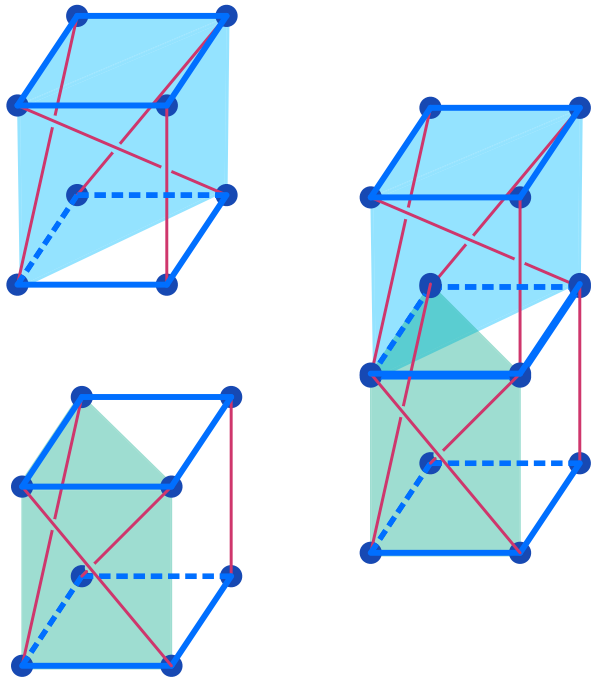


Figure 4.13: The rotation braids $\delta_{\{1,2,3\}}, \delta_{\{2,3,4\}}$, and $\delta_{\{1,2,3\}}\delta_{\{2,3,4\}}$

Notice that because the positive half-twists include the positive half twists of adjacent strands, the standard generators for the braid group, we see that the set of all rotation braids $\{\delta_A \mid A \subseteq [n]\}$ form a new, larger generating set for $\textsc{Braid}_n$. We use the rotation braids to give a different presentation for $\textsc{Braid}_n$.

Definition 4.5.4. (Dual simple braids) Let $\pi = \{A_1, A_2, \ldots, A_k\}$ be a noncrossing partition in NC_n. We define the *dual simple braid* δ_π to be the product of rotation braids $\delta_{A_1}\delta_{A_2}\cdots\delta_{A_k}$. If π has a unique, non-singleton block A, we use δ_A to represent δ_π. In the

case that the singleton block $A = [n]$, we let δ_n represent $\delta_{[n]}$. We refer to the set of dual simple braids as DS_n.

With this, we get an injection $\mathrm{NC}_n \to \textsc{Braid}_n$ and with the map $\textsc{Braid}_n \to \textsc{Sym}_n$ we also get an injection $\mathrm{NC}_n \to \textsc{Sym}_n$ sending $\pi \mapsto \sigma_\pi$. So the noncrossing partition π is associated to the dual simple braid δ_π and a noncrossing permutation σ_π. Moreover, the map $\mathrm{NC}_n \to \textsc{Braid}_n$ allows us to embed the noncrossing partition lattice into a Cayley graph for the braid group. Therefore, many properties of the poset structure for NC_n can be used to analyze the group structure of $\textsc{Braid}_n$. We start with a definition that allows use to order dual simple braids.

Definition 4.5.5. (Properly ordered) Let A and B be subsets of $[n]$. We say the ordered pair (A, B) is properly ordered if the corresponding convex hulls $\textsc{Conv}(P_A)$ and $\textsc{Conv}(P_B)$ are disjoint or they intersect in the single vertex p_i with the property that the sequence $p_{i+1}, p_{i+2}, \ldots$ encounters all elements of $P_A \setminus \{p_i\}$ before any element of P_B. Similarly, if π and τ are noncrossing partitions of $[n]$, then (π, τ) is properly ordered if for each pair of blocks $A \in \pi$ and $B \in \tau$, we have (A, B) is properly ordered.

McCammond in [McC] uses the following proposition to prove a number of properties regarding factorizations of dual simple braids.

Proposition 4.5.6. *If $\sigma, \tau \in \mathrm{NC}_n$, then (σ, τ) are properly ordered if and only if $\delta_\sigma \delta_\tau = \delta_{\sigma \vee \tau}$.*

We are interested in the set of all nontrivial dual simple braids as our generating set for $\textsc{Braid}_n$. We then include every relation of the form $\alpha_1 \alpha_2 = \alpha$ where α_1, α_2 and α are

nontrivial dual simple braids. But this condition is easily rephrased using the notion of proper ordering allowing us to present the braid group as follows [Bir98], [Bes03].

Theorem 4.5.7. *(Dual Presentation) Let* DS_n^* *denote the set of nontrivial dual simple braids. Then,* DS_n^* *generates the n-strand braid group with the following presentation:*

$$\textsc{Braid}_n = \langle \mathrm{DS}_n^* \mid \delta_\pi \delta_\tau = \delta_{\pi \vee \tau} \text{ if } (\pi, \tau) \text{ properly ordered} \rangle$$

We call this the dual presentation *for the braid group.*

Generalizing the constructions made by Birman, Ko, and Lee, David Bessis gives a dual presentation for all spherical Artin groups and shows that these groups are Garside groups with respect to these presentations [Bes03]. In the final chapter of Part I of this dissertation, we take a look at dual Euclidean Artin groups whose structure was recently made clear by McCammond, Sulway, and Brady.

Remark 4.5.8. The image of NC_n gives $\textsc{Braid}_n$ a new Garside structure with a new Garside element given by $(12\cdots n)$, a new set of simple elements, and a new, bigger, positive dual braid monoid. In particular, the standard Garside algorithms can be run using these new structures to give a different solution to the word problem for $\textsc{Braid}_n$.

5. Dual Euclidean Artin Groups

Spherical Artin groups have been pretty well understood ever since their introduction in 1972 [Del72]. A natural next course of study would be the Euclidean Artin groups but unlike their spherical counterparts, questions regarding these groups remained unanswered for decades. Recently, Jon McCammond along with Noel Brady and Robert Sulway, published a trilogy of articles that shed light on the structure of Euclidean Artin groups [BM15, McC15, MS17]. In particular, they show that every Euclidean Artin groups is isomorphic to a dual Euclidean Artin group which is the same group but with a different presentation. Moreover, they show that each dual Euclidean Artin group is either a nonstandard Garside group or the subgroup of a nonstandard Garside group. Importantly, this shows that Euclidean Artin groups have a solvable word problem. In this chapter, we introduce the dual Euclidean Artin groups and describe their structure.

5.1 Isometries

To establish some notation, we start with a discussion of Euclidean isometries and distinguish between points and vectors as in [BM15].

Definition 5.1.1. (Points and Vectors) In what follows, let V denote a n-dimensional Euclidean vector space with the standard inner product. We take E to be the *affine vector space* which is a set with a simply transitive $v \in V$ action such that $x+v$ represents

the image of the point $x \in E$ under the action by the vector $v \in V$. We refer to the elements of E as *points* and the elements of V as *vectors*.

We now identify nice subspaces of V and E that will allow us to discuss invariants of Euclidean isometries.

Definition 5.1.2. (Linear and Affine Subspaces) If a subset of V is closed under linear combination, we say it is *linear*. We say subset of either V or E is *affine* if for any pair of elements in the subset, the line passing through these elements is also contained in the subset. It follows that V contains linear subspaces that pass through the origin and affine subspaces that may or may not pass through the origin. The vector space E contains only affine subspaces as the origin is not identified in that space, which means that linear combinations are not defined.

Definition 5.1.3. (Invariants) Let w be an isometry of E. Define the *move-set* of w to be the subset $\textsc{Mov}(w) \subset V$ of all motions of the points in E. That is

$$\textsc{Mov}(w) = \{\lambda \in V \mid x + \lambda = w(x) \text{ for some } x \in E\}.$$

$\textsc{Mov}(w)$ is an affine subspace of V and is therefore a translation of a linear subspace. If we let U denote the linear subspace of V that differs from $\textsc{Mov}(w)$ by a translation, with μ the unique vector in $\textsc{Mov}(w)$ closest to the origin, then we call $U + \mu = \{\lambda + \mu \mid \lambda \in U\}$ the *standard form* of $\textsc{Mov}(w)$. The points in E that undergo the motion μ form an affine subspace of E which we refer to as the *min-set* of w, denoted $\textsc{Min}(w) \subset E$. The

proof showing $\textsc{Mov}(w)$ and $\textsc{Min}(w)$ are affine subspaces of V and E respectively can be found in [BM15].

Definition 5.1.4. (Reflections and Translations) Given a Euclidean isometry w, should $\textsc{Mov}(w)$ contain the origin, then μ is the zero vector and $\textsc{Min}(w)$ is equivalent to the set of points fixed by w. In this case, we say that w is an *elliptic* isometry and $\textsc{Min}(w) = \textsc{Fix}(w)$, the points fixed by w. Alternatively, if w fixes no points, then μ is nontrivial and $\textsc{Mov}(w)$ forms a nonlinear affine subspace. In this case, we say w is a *hyperbolic* isometry. Elementary examples of elliptic and hyperbolic isometries are reflections and translations, respectively. Let H represent a hyperplane in E. There is a unique, nontrivial isometry re_H fixing H called a *reflection* where $\textsc{Fix}(re_H) = H$ and $\textsc{Mov}(re_H)$ is a line through the origin in V. Next, let $\lambda \in V$ be a nontrivial vector. We define a *translation* isometry, denoted tr_λ, to be the isometry where $tr_\lambda(x) = x + \lambda$. Notice, $\textsc{Min}(tr_\lambda) = E$ and $\textsc{Mov}(tr_\lambda) = \{\lambda\}$.

5.2 Intervals

In what follows, we build up the conventions needed to construct intervals that are also posets.

Definition 5.2.1. (Intervals in Metric Spaces) Given the metric space (X, d), let $x, y, z \in X$. We say that z is *between* x and y whenever the triangle inequality becomes an equality. Specifically, z is between x and y when $d(x, z) + d(z, y) = d(x, y)$. The *interval* $[x, y]$ is the set of all points between x and y. We can put a partial order on the interval $[x, y]$ by defining $z \leq w$ whenever $d(x, z) + d(z, w) + d(w, y) = d(x, y)$.

We use this definition to analogously define intervals in a group using its Cayley graph as a metric space.

Definition 5.2.2. (Intervals in Groups) Let G be a marked group with a fixed generating set S that is symmetric and injects into G. The *right Cayley graph of G with respect to S* is a labeled, directed graph denoted $\textsc{Cay}(G, S)$ with vertices indexed by G and edges indexed by $G \times S$. In particular, the edge $e_{(g,s)}$ starts at the vertex v_g and ends at the vertex v_h if $h = g \cdot s$. The natural left action of G on its right Cayley graph is faithful, vertex transitive, and preserves graph labels and orientation. The *distance* $d(g, h)$ is the combinatorial length of the shortest path in the Cayley graph from v_g to v_h. This allows us to define a metric on G, which depends on its generating set S. That is, for $g, h \in G$ the *interval* $[g, h]$ is the poset of the group elements between g and h. We can view $[g, h]$ as the portion of the Cayley graph between g and h that consists of the union of all minimal length directed paths from the vertex v_g to v_h that encodes the poset structure described previously.

Because Cayley graphs are homogeneous, we know $d(g, h) = d(1, g^{-1}h)$. Moreover, the interval $[g, h]$ is isomorphic to $[1, g^{-1}h]$ as edge-labeled, directed graphs. McCammond and Brady thoroughly investigate intervals of the form $[1, w]$ where w is a Euclidean isometry and the Cayley graph of the group of isometries of E is generated by **all** reflections to characterize the poset structure of all Euclidean intervals. They start by describing a nice relationship between the invariants of a Euclidean isometry w and the invariants of a reflection re. The *reflection length* of w, or how far w is from the identity in its Cayley graph, satisfies the following, a result known as Scherk's theorem [Sch50].

Theorem 5.2.3. *Let w be a Euclidean isometry with a k-dimensional move-set. If w is elliptic, its reflection length is k. If w is hyperbolic, its reflection length is $k+2$.*

When a Euclidean isometry w is multiplied by a reflection re, the reflection length is completely classified using $\text{MOV}(w) = U + \mu$ and the reflection re (see [BM15]). The basic invariants of the product $re \cdot w$ results in the following theorem.

Theorem 5.2.4 (Brady, McCammond)**.** *Let w be an elliptic isometry with* $\text{MOV}(w) = U \subset V$*. Then, $[1, w]$ is a lattice.*

Using reflection length, Brady and McCammond completely characterize when the interval $[1, w]$ is a lattice. Furthermore, they define a coarse structure of the elements in the interval.

Definition 5.2.5. (Coarse Structure) Let w be a hyperbolic Euclidean isometry of maximal reflection length. Geometrically, this hyperbolic isometry of maximal reflection length yields an axis that we can orient vertically. Then any motion that moves points perpendicular to this axis we will call horizontal and any motion with any vertical motion we will call vertical. For any $u \in [1, w]$, there exists a unique $v \in [1, w]$ such that $uv = w$. Using this fact, we can partition the elements of the of the interval $[1, w]$ to reveal its structure using a diagram like that in Figure 5.1. Each row in the diagram indicates the type of isometry of u and v respectively and each column is distinguished by the dimensions of their basic invariants. Elements that appear in the top row have hyperbolic-elliptic factorizations, elements in the middle row have elliptic-elliptic factorizations, and elements in the bottom row have elliptic-hyperbolic factorizations (see row labels in Figure 5.1). When either u or v is hyperbolic, the other must be elliptic and

every point is moved strictly horizontally under the elliptic motion. In other words, the fixed set of the elliptic isometry is invariant under vertical translation. When both u and v are elliptic, some points are moved vertically under each isometry. Across the bottom row, the dimension of $\textsc{Fix}(u)$ decreases and the dimension of $\textsc{Mov}(v)$ increases as we move left to right. In the middle row, the dimension of $\textsc{Fix}(u)$ decreases and the dimension of $\textsc{Fix}(v)$ increases as we move left to right. Finally, in the top row, the dimension of $\textsc{Mov}(u)$ increases and the dimension of $\textsc{Fix}(v)$ increases as we move left to right.

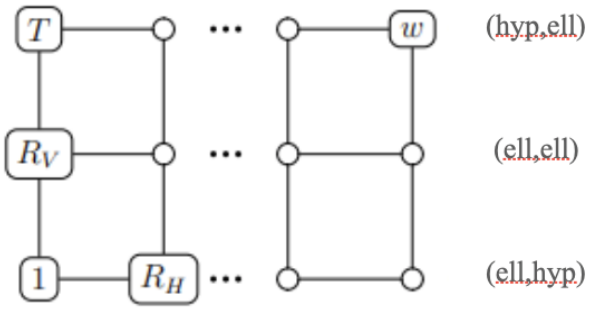


Figure 5.1: The coarse structure for a maximal hyperbolic isometry

We can identify several special elements within the diagram. In particular, the box in the lower left is the identity element representing the factorization $1 \cdot w = w$. Similarly, the box in the upper right-hand corner is the element w representing the factorization $w \cdot 1 = w$. Reflection length can also be recovered from the diagram as an element's length corresponds to the number of steps its box is from the lower left-hand corner.

Beyond these particular elements, we can also identify several families of elements within the diagram as well. The elements located in the upper left-hand corner are hyperbolic isometries in the interval $[1, w]$ with reflection length 2. We can recover these elements by multiplying two reflections that fix parallel hyperplanes. Such a product

produces a pure translation t_λ where the translation vector λ is an element of $\textsc{Mov}(w)$. Note that only some of the translations tr_λ are contained in the interval $[1, w]$. The other two isometries of interest have reflection length 1. We start with the elements in the box to the right of the identity. Since these elements have reflection length 1, we know they are reflections. Moreover, being in the bottom row implies that these elements only move points horizontally. We call these elements *horizontal reflections* and represent the set of elements using R_H. The box just above the identity contains reflections whose fixed hyperplane is not invariant under vertical translation and therefore, move points some vertically. We call these elements *vertical reflections* and represent the set of elements with R_V. Though the diagram in Figure 5.1 is visually appealing, ideally, the diagram could be tipped slightly so that the the box containing 1 appears at the bottom of the diagram, the box containing w is at the top, the boxes containing R_v and R_h appear at a height of 1 and the box containing T will be at height 2.

5.3 Coxeter Elements

In this section, we use the structure of intervals in the Euclidean isometry group to gain an understanding of intervals within Euclidean Coxeter groups. These intervals will allow us to view Euclidean Artin groups as a nonstandard Garside group, or a subgroup of one.

In what follows, we let $W = \textsc{Cox}(\widetilde{X}_n)$ be an irreducible Coxeter group that acts geometrically on an n-dimensional Euclidean space E with generating set S.

Definition 5.3.1. (Coxeter Elements) A Coxeter element $w \in W$ is a product of elements in S. There are many Coxeter elements of a group W depending on the order of the product. However, in the case where $\widetilde{X}_n \neq \widetilde{A}_n$, all Coxeter elements belong to the same conjugacy class and act on their corresponding tiling of the Euclidean plane, their Coxeter complex, in the same way [McC15]. In the case of the Euclidean Coxeter groups of type $\widetilde{A}_n$ an ordering on the Coxeter diagram, which is a cycle, helps to identify the Coxeter element of interest. For a given Coxeter element with a fixed factorization, you can orient the edges of the Coxeter diagram based on which vertex of the edge appears first in the factorization. Using this oriented Coxeter diagram, the conjugacy class of the Coxeter element for the group of type $\widetilde{A}_n$ is determined by how many edges point in the clockwise direction and how many edges are pointed counterclockwise. The only Coxeter element that has a lattice interval is the one whose oriented diagram has all edges but one pointing in the same direction. Note that $\textsc{Cox}(\widetilde{A}_2)$ has only one Coxeter element up to conjugacy.

Definition 5.3.2. (The Coxeter Axis) Let w be a Coxeter element for the irreducible Coxeter group $W = \textsc{Cox}(\widetilde{X}_n)$. We know w is a hyperbolic isometry with reflection length $n+1$ measured in W. The min-set of w, $\textsc{Min}(w)$ is a line in E called the *Coxeter axis*. The top dimensional simplices whose interior nontrivially intersects the Coxeter axis are called *axial simplices* and the vertices of these simplices are called *axial vertices*. In Figure 5.2, we see that the Coxeter element of $\textsc{Cox}(\widetilde{G}_2)$ is a glide reflection whose glide axis is given by the dashed line. The axial vertices are the thickened, colored dots and the axial simplices are heavily shaded.

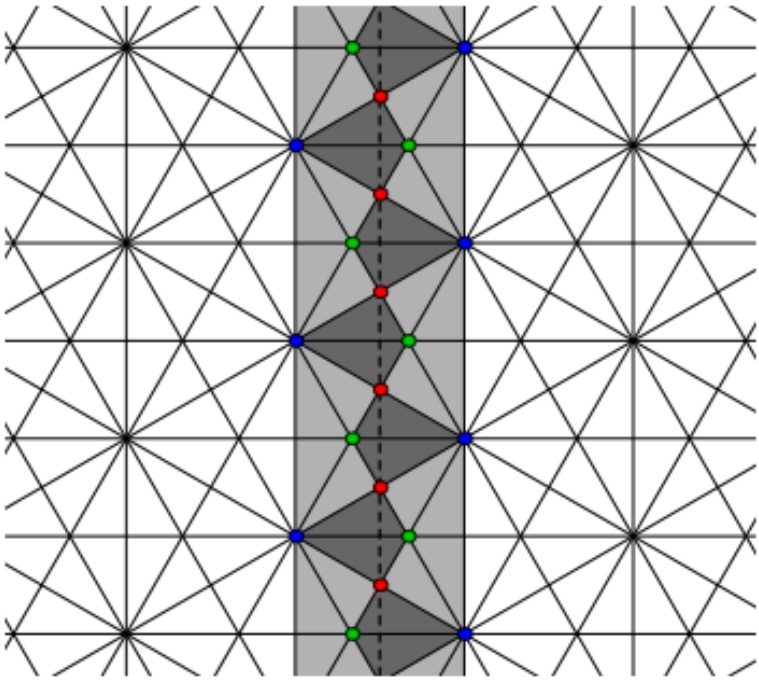


Figure 5.2: The axial features of $\mathrm{Cox}(\widetilde{G}_2)$

Definition 5.3.3. (Strips) We define a *strip* in the Coxeter complex of a Euclidean Artin group W as the convex hull of the axial vertices. We refer to the union of all axial simplices as the *core of the strip*. In Figure 5.2, the core of the strip is the heavily shaded region whereas the strip is given by the union of the lightly shaded and the heavily shaded regions.

In the interval $[1, w]$, every edge is labeled by a reflection in $W = \mathrm{Cox}(\widetilde{X}_n)$ and the product of elements in S that produces the Coxeter element w also produces a minimal length path from v_1 to v_w in $[1, w]$. Therefore, the elements in this Coxeter interval $[1, w]$ will have a coarse structure like that defined in Definition 5.2.5. We should note that only a proper subset of the reflections in W actually label edges in the interval $[1, w]$.

The reflections that do appear as edge labels in the interval can be characterized by the following theorem.

Theorem 5.3.4 (McCammond)**.** *Let w be a Coxeter element of an irreducible Euclidean Coxeter group $W = \mathrm{Cox}(\widetilde{X}_n)$. A reflection labels an edge in the interval $[1, w]$ if and only if its fixed hyperplane contains an axial vertex.*

From this characterization, we recover the coarse structure of the group by separating the vertical and horizontal reflections. In doing so, we actually see that there are infinitely many vertical reflections and a finite number of horizontal reflections. The course structure of the interval in $\mathrm{Cox}(\widetilde{G}_2)$ is given in Figure 5.3. In the top and bottom rows of Figure 5.3, the numbers in the boxes correspond to the number of elements that uphold the coarse structure of the group. In the middle row, the boxes containing 6 corresponds to the number of infinite families within the coarse structure of the group. The next chapter provides a detailed analysis of the coarse structure of the interval in $\widetilde{A}_2$ which is detailed in Figure 6.4.

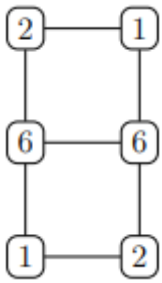


Figure 5.3: The coarse structure for the $\widetilde{G}_2$ interval

5.4 Dual Artin Groups

The notion of dual Artin groups first appear for the braid groups thanks to Birman, Ko, and Lee [Bir98]. Dual spherical Artin groups were studied by David Besis [Bes03] and by Tom Brady and Colum Watt [BW02]. In what follows, we present dual Euclidean Artin groups.

Definition 5.4.1. (Interval groups) Let G be a group with a fixed, symmetric generating set and let $[1, g]$ be an interval in G. The *interval group* G_g is generated by the labels of edges in the interval $[1, g]$ subject to the set of all relations that are visible in the interval. The function from the generators of G_g to G extends to a group homomorphism from G_g to G as well since the relations used to define G_g is a subset of the relations of G. If the labels on the edges of the interval $[1, g]$ include a generating set for G then this map is onto.

Definition 5.4.2. (Dual Artin Groups) Let $W = \textsc{Cox}(\Gamma)$ be a Coxeter group with a Coxeter element w. The interval group W_w defined by the interval $[1, w]$ is called the *dual Artin group* and is denoted $\textsc{Art}^*(\Gamma, w)$. Because these group presentations will depend on a choice of Coxeter element, we include w in the notation representing the dual Artin group.

Theorem 5.4.3 (McCammond-Sulway[MS17])**.** *If $W = \textsc{Cox}(\widetilde{X}_n)$ is an irreducible Euclidean Coxeter group generated by its reflections, and w is a Coxeter element, then the dual Artin group $W_w = \textsc{Art}^*(\widetilde{X}_n, w)$ is isomorphic to $\textsc{Art}(\widetilde{X}_n)$.*

We mentioned previously that Digne provided a presentation for the dual Artin groups of type $\widetilde{A}_n$ and $\widetilde{C}_n$ that was Garside so McCammond and Sulway's theorem generalizes

this result. In particular, Digne uses an embedding of $\textsc{Art}(\widetilde{A}_n) \hookrightarrow \textsc{Art}(B_{n+1})$ to show that Euclidean Artin groups of type $\widetilde{A}_n$ have an infinite-type Garside structure. When taking a closer looks at the Coxeter intervals, McCammond and John Crisp discovered that groups of type $\widetilde{A}_n$, $\widetilde{C}_n$ and $\widetilde{G}_2$ are actually the only groups whose Coxeter intervals are lattice and therefore the only dual Artin groups whose Garside structures were already, fairly well understood. Turning our attention to the groups of type $\widetilde{A}_n$ specifically, we find that there is only one choice of Coxeter element that yields an interval that is a lattice. In the following chapter, we take a closer look at $\textsc{Art}(\widetilde{A}_2)$ and this interval with this specific choice of Coxeter element.

Part II

A Specific Example: $\mathbf{Art}(\widetilde{A}_2)$

In Part II of this text, we focus on the word problem for the Artin group of type $\widetilde{A}_2$. We provide two solutions to the word problem for $\textsc{Art}(\widetilde{A}_2)$. Starting with Chapter 6, we take a closer look the generating set of the dual Artin group of type $\textsc{Art}^*(\widetilde{A}_2, w)$. In Chapter 7, we discuss the Garside structure of the dual Euclidean Artin group $\textsc{Art}^*(\widetilde{A}_2, w)$ and use that structure to provide an algorithm that solves the word problem for the monoid of type $\textsc{Art}(\widetilde{A}_2)$ with respect to the dual generating set. Guided by the dual solution, we then look to solve the word problem for $\textsc{Art}(\widetilde{A}_2)$ with respect to the standard generating set. In Chapter 8, we use the Hurwitz action translate the dual generating set to the standard generating set of $\textsc{Art}(\widetilde{A}_2)$. After translating these dual generators, we use strips in the Coxeter complex of $\widetilde{A}_2$ and the dual algorithm to produce a solution to the word problem with respect to the standard generating set.

6. The Dual Generating Set

For the remainder of this document, we focus on the Euclidean Artin group of type $\widetilde{A}_2$. Recall that in general, the Artin groups of type $\widetilde{A}_n$ have distinct dual presentations. With a careful choice of Coxeter element, it is a Garside presentation and has a Garside structure. For $n = 2$, there is only one conjugacy class of Coxeter element and therefore only one dual presentation. It was first identified by Francois Digne. In what follows, we provide an analysis of the dual generating set and solve the word problem for the dual monoid of type $\widetilde{A}_2$.

6.1 The Coxeter complex and the Davis complex of $\mathrm{Art}(\widetilde{A}_2)$

The standard presentation of $\mathrm{Art}(\widetilde{A}_2)$ is given by

$$\langle a, b, c \mid aba = bab, bcb = cbc, aca = cac \rangle.$$

We saw in Chapter 3 that the Coxeter complex of type $\mathrm{Art}(\widetilde{A}_2)$ is given by a tiling of the Euclidean plane by equilateral triangles. The Davis complex for $\mathrm{Cox}(\widetilde{A}_2)$ is the dual, hexagonal tiling of the Euclidean plane. Figure 6.1 shows both the triangular Coxeter complex and the hexagonal Davis complex.

The elements of $\mathrm{Cox}(\widetilde{A}_2)$ are reflections about hyperplanes (lines) that can be seen in the Coxeter complex. Though the generating elements of $\mathrm{Art}(\widetilde{A}_2)$ are not reflections, the dual generating set is nicely visualized using the hyperplanes in the Coxeter complex

in the same way that the standard and dual generators of the braid group can be indexed by permutations. In what follows, when we refer to reflections, we do so with respect to Euclidean isometries that give rise to the dual generating set. These reflections are discussed as elements of $\textsc{Art}(\widetilde{A}_2)$ but the local picture is visualized as reflections about hyperplanes in the Coxeter complex of $\textsc{Cox}(\widetilde{A}_2)$.

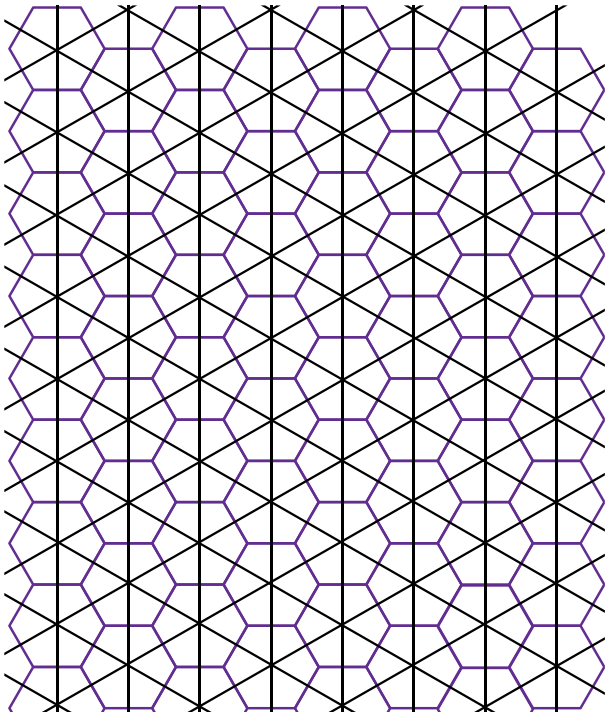


Figure 6.1: A portion of the Coxeter complex and the Davis complex of $\textsc{Cox}(\widetilde{A}_2)$

Example 6.1.1. (The Coxeter Axis of $\textsc{Art}(\widetilde{A}_2)$) A portion of the Coxeter complex for $\textsc{Cox}(\widetilde{A}_2)$ is provided in Figure 6.2. A fundamental chamber of the Coxeter complex, which we will denote τ, is shaded in green. The reflections fixing the sides of τ are given by the standard generators a, b, and c and the image of τ after multiplication by the Coxeter element $w = abc$, is the glide reflection up, along the Coxeter axis shaded in pink. The fundamental chamber τ and our choice of Coxeter element w determines what

we call a *Coxeter axis*, which is represented by the vertical, dashed line in Figure 6.2. The Coxeter axis intersects infinitely many, top-dimensional simplices. The simplices through which the Coxeter axis passes are called *axial simplices* and the vertices of these simplices are called *axial vertices*. In the case of an arbitrary Euclidean Artin group W, there is a discrete set of equally spaced points x_i for $i \in \mathbb{Z}$ along the Coxeter axis that are not contained in the interior of some top-dimensional simplex. In Figure 6.2, these points are where the dotted gray Coxeter axis crosses the edge of the Coxeter complex. We will focus instead on the linear ordering of the axial vertices which is described in further detail in Section 6.2.

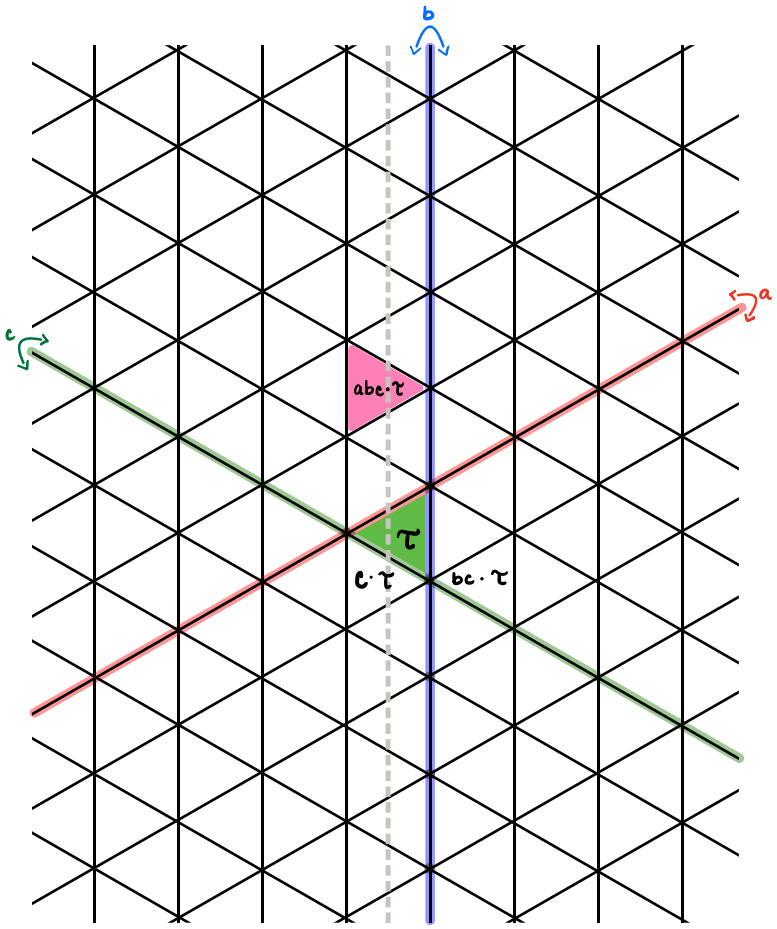


Figure 6.2: The Coxeter complex of $\mathrm{Cox}(\widetilde{A}_2)$, a fundamental chamber, and the Coxeter axis

Definition 6.1.2. (Strips of $\textsc{Art}(\widetilde{A}_2)$) Recall that we defined both a strip and the core of a strip in Definition 5.3.3. In the case of $\textsc{Art}(\widetilde{A}_2)$, the strip and the core of the strip coincide. In the case of a Euclidean Artin group W that is not of type A, the core of the strip is a subcomplex of the strip like we saw in Figure 5.2. In general, we can orient the strip so that traveling *up* the strip corresponds to traveling from the fundamental chamber to the image of τ under w.

6.2 Dual Generators

We saw in Chapter 5 that the coarse structure for $\textsc{Art}(\widetilde{A}_2)$ is generated by two types of reflections. We also saw that the dual presentation is constructed is by looking at the action of a Coxeter element w on the Coxeter complex. By using the factorizations and labeling all of the reflections, we get the presentation for the dual Artin group $\textsc{Art}^*(\widetilde{A}_2, w)$.

We can describe all of the reflections in the dual Euclidean Artin group of type $\widetilde{A_2}$ that appear as left divisors of $w = abc$ using the strip. The equilateral triangles along the Coxeter axis are linearly ordered by where they hit the Coxeter axis. As such, there is a linear ordering on the vertices with respect to the Coxeter axis as well. We start by labeling the axial vertices by integer values see Figure 6.3. There is a canonical linear ordering of the vertices of the fundamental chamber by height, and we take the 0−th vertex to be the middle axial vertex of the fundamental chamber. Traveling up the strip, each consecutive axial vertex will be labeled with a positive integer value with all even vertices on one side of the strip, and all odd vertices on the other. We can label the

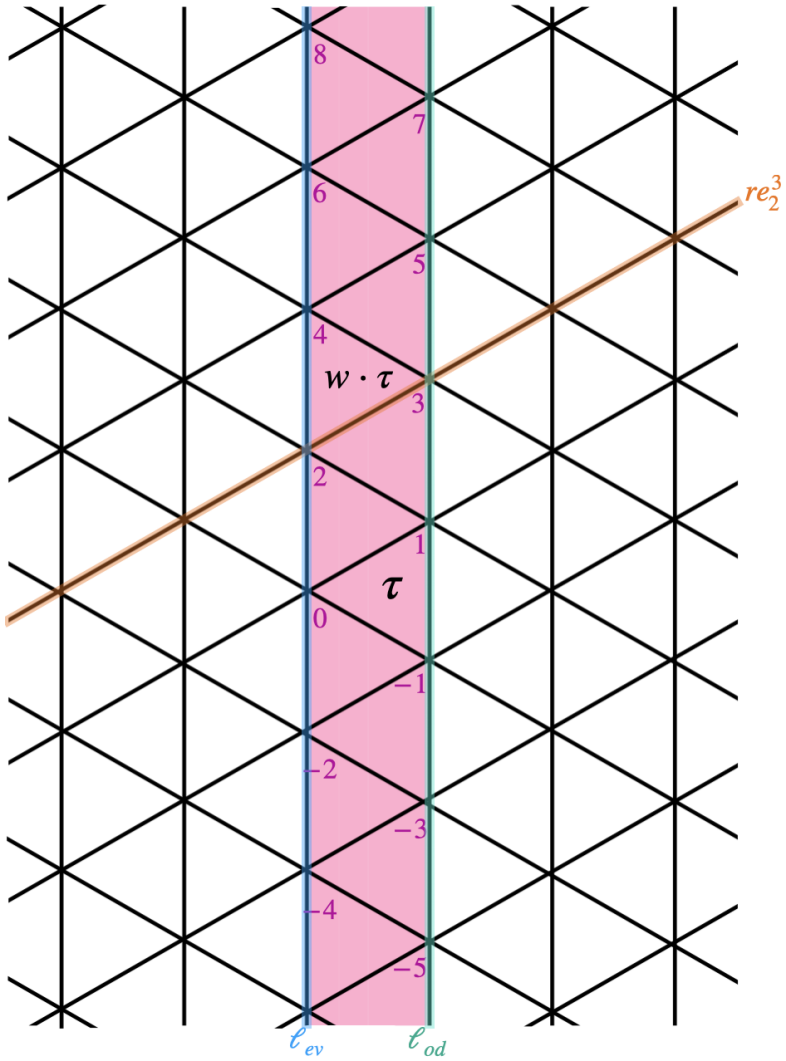


Figure 6.3: A strip in the Coxeter complex of type $\widetilde{A}_2$

axial vertices below the 0-th vertex analogously. In particular, a reflection is below (a left divisor of) w if and only if it crosses through an axial vertex. With this, we begin our analysis of the reflections in $\textsc{Art}(\widetilde{A}_2)$.

Example 6.2.1. (Horizontal and Vertical Reflections) We can describe all of the left divisors of w using the strip. In what follows, we use actions on the strip to distinguish between our horizontal and vertical reflections.

Let ℓ_{ev} be the vertical line containing all even axial vertices, that is, the line on the left-hand side of the strip. We take ℓ_{od} be the vertical line containing all odd axial vertices, or equivalently, the line on the right-hand side of the strip. These lines are highlighted in blue and green respectively in Figure 6.3. The image of the strip after multiplying by a horizontal reflection intersects the strip in a line that contains all even vertices or all odd vertices. The *horizontal reflections* of $\textsc{Art}(\widetilde{A}_2)$ will be denoted re_{ev} and re_{od}. The horizontal reflection re_{ev} fixes the left side of the strip, ℓ_{ev}. The horizontal reflection re_{od} fixes the right side of the strip, ℓ_{od}.

Each of the *vertical reflections* occur in a hyperplane that intersects the i-th and $(i+1)$-st vertices along the strip. These vertical reflections come in two infinite families. we distinguish between the two using the notation re^{2j}_{2j-1} and re^{2k+1}_{2k} where $j, k \in \mathbb{Z}$. Using the labeled vertices along the strip we see that the reflections re^{2j}_{2j-1} occur in negatively sloped lines and the reflections re^{2k+1}_{2k} occur in the positively sloped lines. For example, the vertical reflection re^3_2 fixes the line through the vertices labeled 2 and 3 and is highlighted in orange in Figure 6.3.

After establishing our horizontal and vertical reflections, we describe the height 2 generators seen in the coarse structure of $\textsc{Art}(\widetilde{A}_2)$.

Example 6.2.2. (Translations and Rotations) The coarse structure of $\textsc{Art}^*(\widetilde{A}_2, w)$ contains two translations, which we will denote tr_{ev} and tr_{od}. The image of the strip under the action of tr_{ev} and tr_{od} will cross over the ℓ_{ev} and ℓ_{od}, respectively. For concreteness, the product of reflections $re^3_2 \cdot re^1_0$ is one factorization of tr_{ev}. Using this factorization, we see that tr_{ev} translates vertex 1 to vertex 4 and will move the entire strip in similar

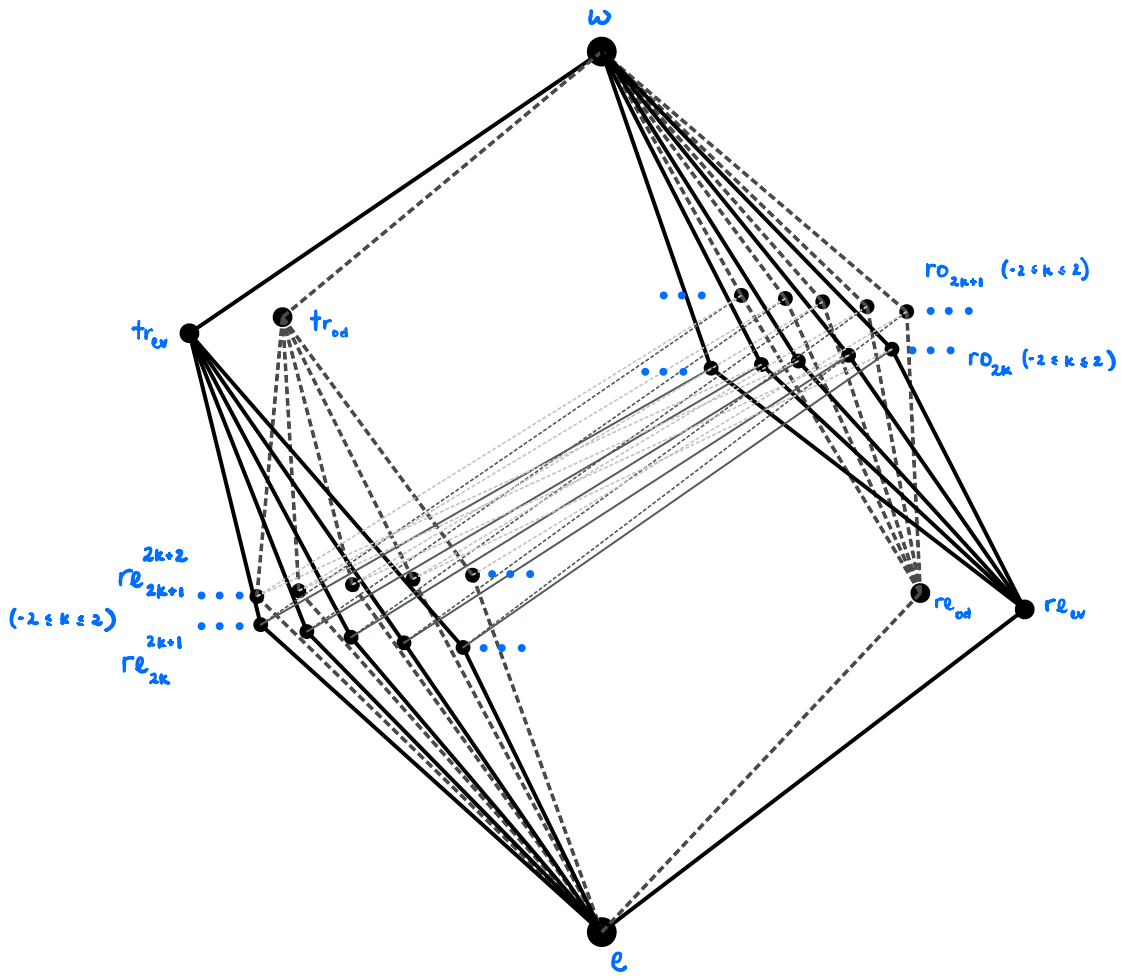


Figure 6.4: The lattice for the dual generating set of $\textsc{Art}(\widetilde{A}_2)$

fashion. Similarly, $re_1^2 \cdot re_{-1}^0$ is one possible factorization of tr_{od} and this translation moves vertex 0 to vertex 3.

The action of ro_k on the strip results in a rotation of the strip centered at the axial vertex k. Hence, our rotations come in two infinite families, depending on what side of the strip serves as a center of rotation. We denote these rotations with ro_{2n} and ro_{2n+1} for some $n \in \mathbb{Z}$. The rotations are obtained as products of two height 1 reflections and can have all types of height 1 reflections as left divisors. In particular, the rotations ro_{2k} are $\dfrac{2\pi}{3}$ counterclockwise rotations centered at the even vertex k. Rotations ro_{2k+} are

clockwise rotations centered at the odd vertex k. We elaborate more on this in Chapter 7.

Now that we've established all of the dual generators for $\text{ART}^*(\widetilde{A}_2, w)$, we can see how they all relate to one another in Figure 6.4. Notice the infinite families of vertical reflections and rotations that appear on the lower left and upper right of Figure 6.4 and in the "middle row" of the coarse structure in Figure 6.5. In addition, we see just two horizontal reflections and two translations that appear in the lower right and upper left. This dual generating set forms a lattice, revealing the Garside structure of the group that will help in creating the dual solution to the word problem.

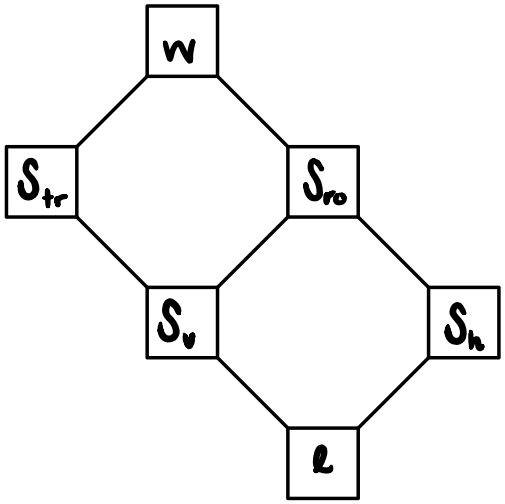


Figure 6.5: The general coarse structure of $\text{ART}(\widetilde{A}_2)$

Example 6.2.3. (Coarse Structure of $\text{ART}^*(\widetilde{A}_2, w)$) Given the descriptions of the dual generators, we can now provide a more specific coarse structure for $\text{ART}(\widetilde{A}_2)$ seen in Figure 6.5. The infinite set of all vertical reflections re_i^{i+1} will be denoted S_v and the finite set containing the two horizontal reflections re_{ev} and re_{od} will be denoted S_h. We

let $S_1 = S_v \cup S_h$ represent all height 1 dual generators. For the height 2 dual generators, we let S_{tr} be the set containing the two translations tr_{ev} and tr_{od} and S_{ro} will denote the infinite family of rotations ro_i. We let $S_2 = S_{tr} \cup S_{ro}$ denote all height 2 dual generators. The general coarse structure using this notation can be found in Figure 6.5.

7. A Dual Solution to the Word Problem

The dual Euclidean Artin group $\textsc{Art}^*(\widetilde{A}_2, w)$ with the Coxeter element $w = abc$ has a Garside structure and we rely on that structure to produce a solution to the word problem. We refer to this solution as a *dual solution*.

7.1 Descendants of Dual Generators

Recall that a key part of the algorithm for Garside groups relies on unique meets and joins for every pair of generators. In spite of having infinitely many dual generators, finding the meets and joins of every pair of generators does not become much more difficult. We start by discussing the left divisors of each element within the coarse structure of $\textsc{Art}^*(\widetilde{A}_2, w)$.

We start by identifying all of the immediate descendants (left divisors) of the dual generators, beginning with the height 1 elements which include all of our reflections. Every reflection has only the identity as a descendant. On the other hand, the Coxeter element w has every dual generator as a descendant.

Definition 7.1.1. (Parity) Given $k \in \mathbb{Z}$, let $\bar{k}$ denote the parity of k. Notationally, if k is even, let $\bar{k} = ev$ and if k is odd, let $\bar{k} = od$.

Lemma 7.1.2. *(Descendants of Translations) The nontrivial descendants of tr_{ev} include tr_{ev} and all vertical reflections of the form re_{2i}^{2i+1} for all $i \in \mathbb{Z}$. The nontrivial descendants of tr_{od} include tr_{od} and vertical reflections of the form re_{2k-1}^{2k} for all $k \in \mathbb{Z}$.*

Proof. Geometrically, to produce a translation in a particular direction as a product of two reflections, we must use reflections that move points in that direction. In this case, each translation is achieved as the product of two, consecutive vertical reflections of the same family. Using Definition 7.1.1, each translation can written in the following way:

$$tr_{\bar{k}} = re_{i+2}^{i+3} \cdot re_{i}^{i+1}, \text{ for all } i \in \mathbb{Z} \text{ such that } \bar{i} = \bar{k}. \tag{7.1}$$

That is,

$$tr_{ev} = re_{2i}^{2i+1} \cdot re_{2i-2}^{2i-1} \text{ and } tr_{od} = re_{2i+1}^{2i+2} \cdot re_{2i-1}^{2i}$$

for any integer i. This gives all possible factorizations of tr_{ev} and tr_{od}. Furthermore, Equation 7.1 includes all of the nontrivial factorizations of length 2 of tr_{ev} and tr_{od} given the geometric constraints. □

Hence, the nontrivial descendants of tr_{ev} include reflections of the form re_{2i}^{2i+1} for all $i \in \mathbb{Z}$. Similarly, the nontrivial descendants of tr_{od} are all reflections of the form re_{2i-1}^{2i}, $i \in \mathbb{Z}$. We let R_{tr} denote the set of all factorization relations given by Equation 7.1. Figure 7.1 shows just one such factorization of tr_{ev} given by $re_0^1 \cdot re_{-2}^{-1}$.

Remark 7.1.3. The factorizations given in this section are according to the left action of the dual generators on the strip. For example, if we let τ represent the fundamental chamber in a strip, then one possible factorization of tr_{ev} is given by $re_0^1 \cdot re_{-2}^{-1}$. We then see that $tr_{ev} \rightharpoonup \tau = (re_0^1 \cdot re_{-2}^{-1}) \rightharpoonup \tau$ according to Figure 7.1.

Lemma 7.1.4. *(Descendants of Rotations) The nontrivial descendants of the rotation* ro_k *are* $re_{\bar{k}}$, re_k^{k+1} *and* re_{k-1}^k.

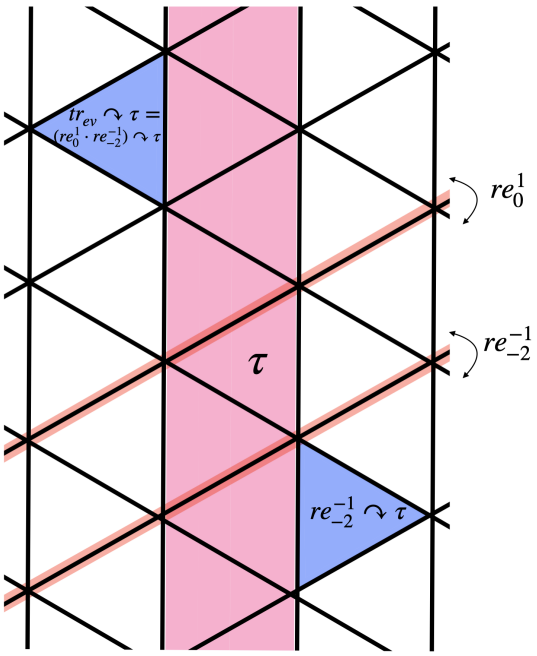


Figure 7.1: A factorization of tr_{ev}

Proof. Recall that the rotations ro_k with $\bar{k} = ev$ are $\dfrac{2\pi}{3}$ counterclockwise rotations centered at the even vertex k. Rotations ro_k such that $\bar{k} = od$ are $\dfrac{2\pi}{3}$ clockwise rotations centered at the odd vertex k. Geometrically, when writing ro_k as a product of two reflections, both reflections need to fix the unique point fixed by ro_k, the vertex k. Moreover, each rotation is comprised of two reflections that differ by a $\dfrac{\pi}{3}$ angle in the appropriate order. This leads us to the following three, nontrivial factorizations for the rotation ro_k:

$$ro_k = re_{k-1}^k \cdot re_{\bar{k}} = re_k^{k+1} \cdot re_{k-1}^k = re_{\bar{k}} \cdot re_k^{k+1}. \tag{7.2}$$

Using Equation 7.2, we can see that each infinite family of rotations are factored as follows.

$$ro_{2n} = re_{2n-1}^{2n} \cdot re_{ev} = re_{2n}^{2n+1} \cdot re_{2n-1}^{2n} = re_{ev} \cdot re_{2n}^{2n+1}$$

$$ro_{2n+1} = re_{2n}^{2n+1} \cdot re_{od} = re_{2n+1}^{2n+2} \cdot re_{2n}^{2n+1} = re_{od} \cdot re_{2n+1}^{2n+2}.$$

Because of the geometric constraints, we know that these are the only possible factorizations which in turn, provide the aforementioned descendants of the rotations ro_k. □

From Lemma 7.1.4, we know the rotation ro_k will have as descendants the horizontal reflection $re_{\bar{k}}$ along with any vertical reflection that contains the subscript or superscript k (re_k^{k+1} and re_{k-1}^k). We let R_{ro} denote the set of all factorization relations given by Equation 7.2.

With the above factorizations, we are now able to provide the dual presentation for $\textsc{Art}^*(\widetilde{A}_2, w)$. Recall the sets used to describe the coarse structure of the group $\textsc{Art}^*(\widetilde{A}_2, w)$ in Example 6.2.3 where $S_1 = S_v \cup S_h$ represents all height 1 dual generators and $S_2 = S_{tr} \cup S_{ro}$ denotes all height 2 dual generators. We then have the following group presentation for $\textsc{Art}^*(\widetilde{A}_2, w)$ with infinitely many generators and infinitely many relations.

Proposition 7.1.5. *(Dual Presentation for $\textsc{Art}^*(\widetilde{A}_2, w)$) The dual presentation for the dual Euclidean Artin group of type $\widetilde{A}_2$ is given by*

$$\textsc{Art}^*(\widetilde{A}_2, w) = \langle S_v, S_h, S_{tr}, S_{ro} \mid R_{tr}, R_{ro} \rangle$$

Proof. By [Bes03] and [MS17] the relations for group presentation come from the factorizations of the height 2 dual generators. In this case, that includes the two translations and the two infinite families of rotations. By Lemma 7.1.2 and Lemma 7.1.4, the sets R_{tr} and R_{ro} include all of the desired factorizations. □

7.2 Meets of Dual Generators

Because there are only six types of nontrivial elements that are left divisors of w, the descendants given above will help determine the meets and joins of every pair of dual generators. The computations of the meets of pairs of dual generators will generally fall into four cases: (1) at least one generator is the identity or w, (2) both generators are in S_1, (3) one generator is in S_1 and the other is in S_2, and (4) both generators are in S_2. We consider each of these cases separately.

Lemma 7.2.1. *(Meets of Elements with the Identity or w) Given an arbitrary dual generator, $x \wedge w = x$ and $x \wedge e = e$.*

Proof. Because the Coxeter element w has every element as a descendant, we know that $x \wedge w = x$. On the other hand, the identity element e has only itself as a descendant. Therefore, $x \wedge e = e$. □

Lemma 7.2.2. *(Meets of Elements in S_1) Given distinct height 1 dual generators h_1 and h_1', $h_1 \wedge h_1' = e$.*

Proof. The elements $h_1, h_1' \in S_1$ have only themselves and the identity element as descendants. Because these elements are distinct, what results is a trivial meet $h_1 \wedge h_1' = e$. □

Results of lemmas 7.2.1 and 7.2.2 are summarized in what follows.

$$
\begin{array}{lrcl}
\text{for } x \in S, & e \wedge x & = & e,\ x \in S \\
\text{for } x \in S, & x \wedge w & = & x \\
\text{for } re_{\bar{i}}, re_{\bar{j}} \in S_h, & re_{\bar{i}} \wedge re_{\bar{j}} & = & \begin{cases} re_{\bar{i}},\ \bar{i} = \bar{j} \\ e,\ \bar{i} \neq \bar{j} \end{cases} \\
\text{for } re_i^{i+1}, re_j^{j+1} \in S_v, & re_i^{i+1} \wedge re_j^{j+1} & = & \begin{cases} re_i^{i+1},\ i = j \\ e,\ i \neq j \end{cases} \\
\text{for } re_{\bar{i}} \in S_h, re_j^{j+1} \in S_v, & re_{\bar{i}} \wedge re_j^{j+1} & = & e
\end{array}
$$

Lemma 7.2.3. *(Meets of generators in S_1 and S_2) The meet of a generator in S_1 and a generator in S_2 is given by the following:*

$$
\begin{array}{lrcl}
for\ re_{\bar{i}} \in S_h, tr_{\bar{k}} \in S_{tr}, & re_{\bar{i}} \wedge tr_{\bar{k}} & = & e \\
for\ re_i^{i+1} \in S_v, tr_{\bar{k}} \in S_{tr}, & re_i^{i+1} \wedge tr_{\bar{k}} & = & \begin{cases} re_i^{i+1},\ \bar{i} = \bar{k} \\ e,\ \bar{i} \neq \bar{k} \end{cases} \\
for\ re_{\bar{i}} \in S_h, ro_\ell \in S_{ro}, & re_{\bar{i}} \wedge ro_\ell & = & \begin{cases} re_{\bar{i}},\ \bar{i} = \bar{\ell} \\ e,\ \bar{i} \neq \bar{j} \end{cases} \\
for\ re_i^{i+1} \in S_v, ro_\ell \in S_{ro}, & re_i^{i+1} \wedge ro_\ell & = & \begin{cases} re_i^{i+1},\ \ell = i\ or\ \ell = i+1 \\ e,\ otherwise \end{cases}
\end{array}
$$

Proof. Given an element in S_2, its descendants include itself, some reflections, and the identity. Therefore, the meet of a generator in S_2 and a generator in S_1 is either going to be trivial or the generator in S_1 if and only if the chosen generator in S_1 is a descendant of the chosen generator in S_2. □

Lemma 7.2.4. *(Meets of Two Generators in S_2) The meet of two generators in S_2 is given by the following:*

$$
\begin{aligned}
\text{for } tr_{\bar{i}}, tr_{\bar{j}} \in S_{tr}, \qquad tr_{\bar{i}} \wedge tr_{\bar{j}} &= \begin{cases} tr_{\bar{i}}, \ \bar{i} = \bar{j} \\ e, \ \bar{i} \neq \bar{j} \end{cases} \\
\text{for } ro_i, ro_j \in S_{ro} \qquad ro_i \wedge ro_j &= \begin{cases} re_{ev}, \ \bar{i} = \bar{j} = ev \\ re_{od}, \ \bar{i} = \bar{j} = od \\ re_i^j, \ j = i+1 \\ re_j^i, \ i = j+1 \\ ro_i, \ i = j \\ e, \quad otherwise \end{cases} \\
\text{for } tr_{\bar{i}} \in S_{tr}, ro_j \in S_{ro}, \quad tr_{\bar{i}} \wedge ro_j &= \begin{cases} re_j^{j+1}, \ \bar{i} = \bar{j} \\ re_{j-1}^j, \ \bar{i} \neq \bar{j} \end{cases}
\end{aligned}
$$

Proof. The meet of two dual generators in S_2 is the most complicated case, and the meets given in the statement can be seen using the geometry of the dual generators with respect to the the strip in the Coxeter complex of type $\widetilde{A}_2$. We know that nontrivial meets occur when the elements of S_2 share a common descendant. We know that a vertical reflection is a descendant of tr_{ev} if and only if the slope of the line of reflection is positive. Analogously, a vertical reflection is a descendant of tr_{od} if and only if the slope of the line of reflection is negative. Lastly, a reflection is a descendant of a rotation if and only if it fixes the point that is the center of rotation.

With this, we are able to conclude that tr_{ev} and tr_{od} have only the identity as a common descendant since the reflections that appear as descendants for the translations will have different slopes respectively. Therefore, the meet of tr_{ev} and tr_{od} is trivial.

In general, two distinct rotations will share a common descendant if and only if the centers of the two rotations lie on a line in the 1-skeleton of the Coxeter complex. This implies that two rotations from the same infinite family will have centers of rotation that lie on the line of reflection for re_{ev} or re_{od} and thus, for $\bar{i} = \bar{j}$, $ro_i \wedge ro_j = re_{\bar{i}}$. If the two rotations are not from the same infinite family, the centers of rotation must be close enough to have a nontrivial meet. That is, if $\bar{i} \neq \bar{j}$, it must be that $|i - j| = 1$ for a nontrivial meet of re_i^j or re_j^i to exist.

Given a translation $tr_{\bar{i}}$ and a rotation ro_j, there will always exist a nontrivial meet. The translation has a family of vertical reflections, with a particular slope, as descendants and the rotation indicates a particular point that must be fixed by one of those descendants. Hence, the meet of a translation $tr_{\bar{i}}$ and a rotation ro_j is given by re_j^{j+1} if $\bar{i} = \bar{j}$ and re_{j-1}^j if $\bar{i} \neq \bar{j}$. □

In what follows, we elaborate on the geometry discussed in the previous proof. In Figure 7.2, the lines highlighted in orange represent the reflections that are descendants of tr_{ev}. Each red vertex in Figure 7.2 represents the center of the rotation ro_{2k}. For each of these rotations, there are three highlighted red lines representing the descendants of ro_{2k} which include two vertical reflections and one horizontal reflection. From this, we see that a meet between tr_{ev} and ro_{2k} is nontrivial and equal to the vertical reflection re_{2k-1}^{2k} and the red and orange lines will coincide on the line connecting axial vertices

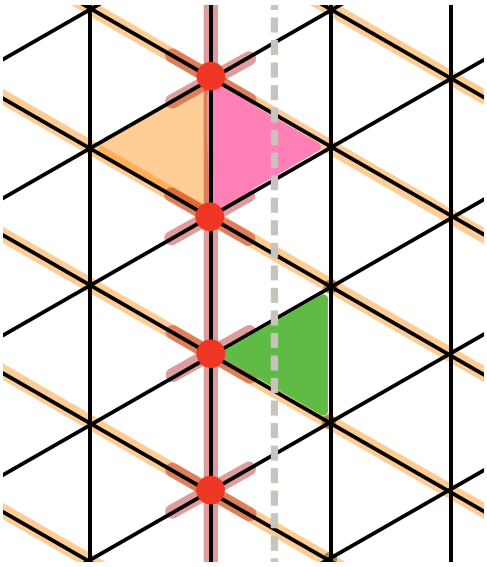


Figure 7.2: Descendants of tr_{ev} and ro_{2k}

$2k-1$ and $2k$. The meets of tr_{od} and the rotations centered on the odd axial vertices behave similarly.

In Figure 7.3, each red vertex represents the center of the rotation ro_{2k} for some $k \in \mathbb{Z}$. For each red vertex, we see three lines highlighted in red representing the descendants of that rotation. The only common highlighted line shared by two distinct rotations is the line of reflection representing re_{ev}. So we see that for $k \neq n \in \mathbb{Z}$, $ro_{2k} \wedge ro_{2n} = re_{ev}$. Analogously, $ro_{2k-1} \wedge ro_{2\ell-1} = re_{od}$. In Figure 7.4 we compare the descendants of two rotations from distinct infinite families, ro_{2k} and $ro_{2\ell-1}$. Again for each rotation, we have 3 descendants highlighted at each vertex in red and blue respectively. From the image, we see a nontrivial meet will occur at a vertical reflection about a line where the red and blue lines coincide. This occurs only when the centers of these rotations share a line. That is, when $|2k-(2\ell-1)| = 1$.

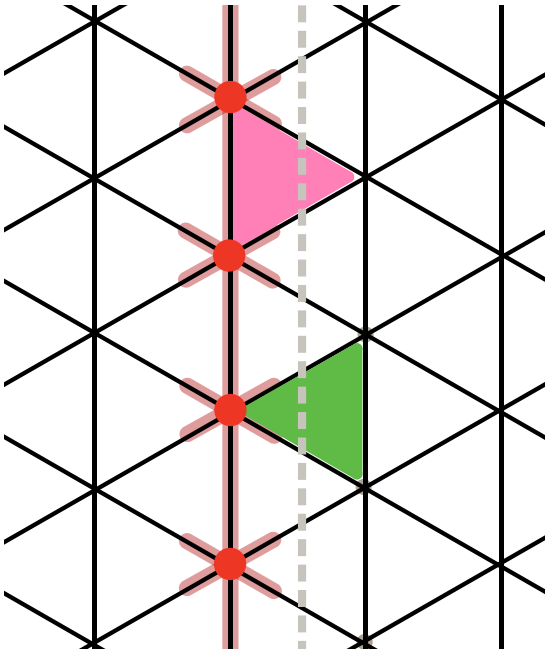


Figure 7.3: Descendants of two rotations from the same infinite family

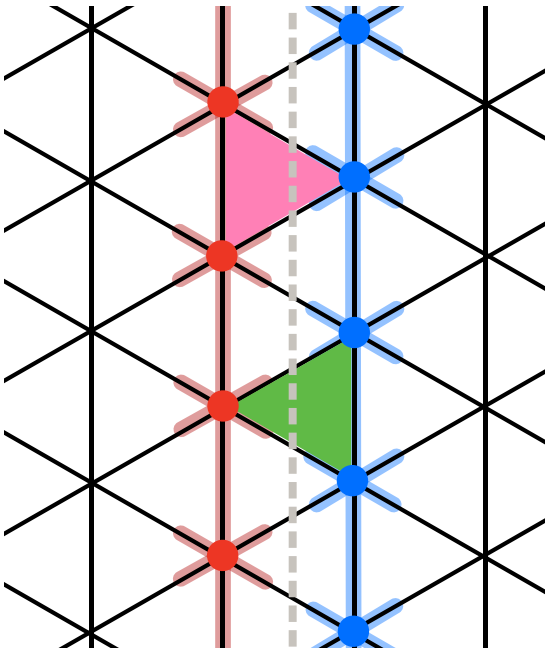


Figure 7.4: Descendants of two rotations from different infinite families

7.3 Pairs of Dual Generators in Normal Form

Taking advantage of the Garside structure of $\textsc{Art}^*(\widetilde{A}_2, w)$, once the meets of the dual generators are identified, we can proceed to put products of two generators into normal form. We use the same rewriting process seen in Chapter 4 where the product $a \cdot b$ is rewritten as $a' \cdot b'$ where $a' = [a \cdot (rc(a) \wedge b)]$ and $b' = (rc(a) \wedge b)^{-1} \cdot b$. Though we have infinitely many dual generators, computing the right and left complements is easily done given the coarse structure of $\textsc{Art}^*(\widetilde{A}_2, w)$. In particular, the right and left complements for specific dual generators are given in Table 7.1.

The left and right complements of arbitrary vertical reflections and rotations will change with their subscripts. Because it is a straight forward computation, we provide the following lemma without proof.

Table 7.1: Right ($x \cdot rc(x) = w$) and left ($lc(x) \cdot x = w$) complements

x	e	re_{ev}	re_{od}	$\cdots$	re_3^4	re_4^5	$\cdots$	tr_{ev}	tr_{od}	$\cdots$	ro_4	ro_5	$\cdots$	w
$lc(x)$	w	tr_{od}	tr_{ev}	$\cdots$	ro_5	ro_6	$\cdots$	re_{ev}	re_{od}	$\cdots$	re_5^6	re_6^7	$\cdots$	e
$rc(x)$	w	tr_{ev}	tr_{od}	$\cdots$	ro_2	ro_3	$\cdots$	re_{od}	re_{ev}	$\cdots$	re_2^3	re_3^4	$\cdots$	e

Lemma 7.3.1. *The left complements of vertical reflections and rotations are given by* $lc(re_i^{i+1}) = ro_{i+2}$ *and* $lc(ro_k) = re_{k+1}^{k+2}$. *The right complements of vertical reflections and rotations are given by* $rc(re_i^{i+1}) = ro_{i-1}$ *and* $rc(ro_k) = re_{k-2}^{k-1}$.

Using the complements and the meets from Section 7.2, we provide the pairs of dual generators whose product is rewritten when put in normal form. Because we have an infinite generating set, a table containing all pairs in normal form could be organized in the following way. Recall that S_1 is the set containing all height 1 generators including two horizontal reflections and two infinite families of vertical reflections. S_2 is the set containing all height 2 generators including two translations and two infinite families of rotations.

Table 7.2: Organizing Dual Generators in Normal Form

$S_1 \times S_2$	$S_1 \times S_1$
$S_2 \times S_2$	$S_2 \times S_1$

Each of the four components of Table 7.2 can be broken up into four smaller parts: a 2×2 table, a $2 \times \infty$ table, an $\infty \times 2$ table and an $\infty \times \infty$ table. We provide each of these in what follows. For all of the tables that follow, a blank entry indicates a product of dual generators that are already in normal form.

We start with Table 7.3 which puts generators from $S_2 \times S_1$ in normal form. The product of a height 2 element and a height 1 element will either update to w or it is already in normal form. In particular, the height 1 element is must be the right complement of the height 2 for an update to occur. We arrange this table so that the column labels contain the complements of the elements listed in the row labels. Hence, all updates occur on the diagonal and the final table is as follows.

Table 7.3: $S_2 \times S_1$ Generators in Normal Form

$x \setminus y$	re_{ev}	re_{od}	$\cdots$	re_2^3	re_1^2	re_0^1	re_{-1}^0	$\cdots$
tr_{ev}	w							
tr_{od}		w						
$\vdots$			$\ddots$					
ro_1				w				
ro_2					w			
ro_3						w		
ro_4							w	
$\vdots$								$\ddots$

Next we provide Table 7.4 which puts the product $x \mid y$ into normal form where $x, y \in S_1$. Because both generators are of height 1, the only updates that will occur are those whose product represents a generator of height 2. We notice that the 2×2 part of the table is already in normal form. In the $2 \times \infty$ and $\infty \times 2$ parts of the table, we see a zig-zag pattern of rotations ro_k. In the $\infty \times \infty$ part of Table 7.4, we have a diagonal of rotations ro_k and a superdiagonal alternating translations tr_{ev} and tr_{od}. The updates in this table are as expected since certain products of height 1 dual generators are the precise factorizations of the height 2 dual generators.

Table 7.4: $S_1 \times S_1$ Generators in Normal Form

$x \setminus y$	re_{ev}	re_{od}	$\cdots$	re_1^2	re_0^1	re_{-1}^0	re_{-2}^{-1}	$\cdots$
re_{ev}			$\cdots$		ro_0		ro_{-2}	$\cdots$
re_{od}			$\cdots$	ro_1		ro_{-1}		$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$				
re_2^3		ro_3		ro_2	tr_{ev}			
re_1^2	ro_2				ro_1	tr_{od}		
re_0^1		ro_1				ro_0	tr_{ev}	
re_{-1}^0	ro_0						ro_{-1}	$\ddots$
$\vdots$	$\vdots$	$\vdots$						$\ddots$

In what follows we have Table 7.5 which puts the product $x \mid y$ into normal form where $x, y \in S_2$. Given the product of two height 2 generators, if there is an update, it will be a new product where the first factor is w. Otherwise, the product will be in normal form. In particular, we see that the 2×2 part of the table is already in normal form. In the $2 \times \infty$ and $\infty \times 2$ parts of the table, we see a zig-zag pattern of a product of the form $w \mid re_i^{i+1}$. In the $\infty \times \infty$ part of Table 7.4, we have an alternating diagonal of the product $w \mid re_{ev}$ and $w \mid re_{od}$ and a superdiagonal of products of the form $w \mid re_i^{i+1}$.

The only table that remains is Table 7.6 which puts the product $x \mid y$ into normal form where $x \in S_1$ and $y \in S_2$. This is the most complicated of tables because of the variety of ways the products can update but the generalizations are still rather nice. Because tr_{od} and tr_{ev} are the right complements of re_{od} and re_{ev} respectively, the 2×2 part of this table contains w along the diagonal. In the $2 \times \infty$ part of the table, we see a zig-zag pattern of the form $ro_k \mid re_{\bar{\ell}}$ where $\bar{k} \neq \bar{\ell}$. In the $\infty \times 2$ part of the table, we see a similar pattern of the form $tr_{\bar{\ell}} \mid re_i^{i+1}$ where $\bar{\ell} = \bar{i}$. In the $\infty \times \infty$ part of Table 7.6, we

Table 7.5: $S_2 \times S_2$ Generators in Normal Form

$x \setminus y$	tr_{od}	tr_{ev}	$\cdots$	ro_2	ro_1	ro_0	ro_{-1}	$\cdots$
tr_{od}			$\cdots$	$w \mid re_2^3$		$w \mid re_0^1$		$\cdots$
tr_{ev}			$\cdots$		$w \mid re_1^2$		$w \mid re_{-1}^0$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$				
ro_3	$w \mid re_{-1}^0$			$w \mid re_{ev}$	$w \mid re_0^1$			
ro_2		$w \mid re_{-2}^{-1}$			$w \mid re_{od}$	$w \mid re_{-1}^0$		
ro_1	$w \mid re_{-3}^{-2}$					$w \mid re_{ev}$	$w \mid re_{-2}^{-1}$	
ro_0		$w \mid re_{-4}^{-3}$					$w \mid re_{od}$	$\ddots$
$\vdots$	$\vdots$	$\vdots$						$\ddots$

have a diagonal of w, a subdiagonal of $ro_k \mid re_{\bar{\ell}}$ such that $\bar{k} = \bar{\ell}$, and a super diagonal of $tr_{\bar{\ell}} \mid re_i^{i+1}$ such that $\bar{\ell} \neq \bar{i}$.

Table 7.6: $S_1 \times S_2$ Generators in Normal Form

$x \backslash rc(x)$	tr_{od}	tr_{ev}	$\cdots$	ro_6	ro_5	ro_4	ro_3	$\cdots$
re_{od}	w		$\cdots$	$ro_5 \mid re_{ev}$		$ro_3 \mid re_{ev}$		$\cdots$
re_{ev}		w	$\cdots$		$ro_4 \mid re_{od}$		$ro_2 \mid re_{od}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$				
re_7^8	$tr_{od} \mid re_3^4$		$\ddots$	w	$tr_{od} \mid re_4^5$			
re_6^7		$tr_{ev} \mid re_2^3$		$ro_6 \mid re_{ev}$	w	$tr_{ev} \mid re_3^4$		
re_5^6	$tr_{od} \mid re_1^2$				$ro_5 \mid re_{od}$	w	$tr_{od} \mid re_2^3$	
re_4^5		$tr_{ev} \mid re_0^1$				$ro_4 \mid re_{ev}$	w	$\ddots$
$\vdots$	$\vdots$	$\vdots$					$\ddots$	$\ddots$

With Tables 7.6, 7.4, 7.5, and 7.3, we can put an arbitrary positive word in the dual generating set into normal form.

7.4 The Dual Algorithm

To put an arbitrary word, with respect to the dual generating set into normal form, we use the following three steps. Consider the arbitrary word given by, $u_1 \mid u_2 \mid \cdots \mid u_n$. Step one requires us to first identify which of the u_i represent inverses of dual generators. Suppose, for example, that every other letter represents an inverse of a dual generator. Then, the word could be rewritten as

$$u_1 \mid u_2^{-1} \mid u_3 \mid u_4^{-1} \mid \cdots \mid u_n.$$

Using Remark 4.4.6, we rewrite each of the inverses, u^{-1} as $w^{-1} \mid lc(u)$. After step one, the word is then given by

$$u_1 \mid w^{-1} \mid lc(u_2) \mid u_3 \mid w^{-1} \mid lc(u_4) \mid \cdots \mid u_n.$$

The next step of the algorithm is to use conjugation by w to pull the w^{-1}'s to the front of the word. In particular, a subword of the form $u \mid w^{-1}$ is rewritten as $w^{-1} \mid v$ where $v = wuw^{-1}$ and v is positive. After step two, the word is now of the form

$$w^k \mid v_1 \mid v_1 \mid \cdots \mid v_n$$

where each v_i is a proper, nontrivial factor of w and k is a nonpositive integer. For simplicity, we have removed the bars between the w^{-1}'s at the beginning. In step three, we put the positive word given by $v_1 \mid v_2 \mid \cdots v_n$ into normal form using Tables 7.3 through

7.6. This will pull positive powers of w to the left. The result is a word of the form

$$w^{k+\ell} \mid v_1' \mid v_2' \mid \cdots \mid v_n'$$

where ℓ is a nonnegative integer. We determine that the word is trivial if and only if $k + \ell = 0$ and each v_i' is the identity.

8. From Dual Generators to Standard Generators

The purpose of this Chapter is to convert the normal form for $\textsc{Art}^*(\widetilde{A}_2, w)$ into a normal form for $\textsc{Art}(\widetilde{A}_2)$. The normal form provided for $\textsc{Art}^*(\widetilde{A}_2, w)$ in Chapter 7 makes use of the dual presentation which has an infinite generating set. To proceed with finding an algorithm for the standard generating set of $\textsc{Art}(\widetilde{A}_2)$, using the dual algorithm, we translate the dual generators to read as words in the classical generating set.

8.1 The Hurwitz Action on the Dual Generating Set

The standard generating set appears in the dual generating set as $a = re_0^1$, $b = re_{od}$, and $c = re_{-1}^0$. We also know that the Coxeter element w can be factored as abc in the standard generating set. As in Chapter 7, we use vertical bars to distinguish various factorizations. For example, w is given by $a \mid b \mid c$ and therefore $re_0^1 \mid re_{od} \mid re_{-1}^0$ with respect to the dual generating set. We now seek to find a canonical form for the rest of the dual generators in terms of the standard generators. We can apply the Hurwitz action to the initial factorization of w in terms of the dual generators to begin translating the rest of the dual generating set.

Definition 8.1.1. (Hurwitz Action) Let G be a group and S a subset of G that is closed under conjugation. Let S^n denote all words of length n made up of elements of S. There is a natural action of $\textsc{Braid}_n$ on S^n where the braid generator σ_i replaces the two-letter

subword xy in the i-th and $(i+1)$-st positions of the word with the subword zx where $z = xyx^{-1} \in S$ and leaves all other positions unchanged. In other words, applying σ_i to the factorization

$$\cdots \mid x \mid y \mid \cdots$$

where x is the i-th factor and y is the $(i+1)$-st factor results in the new factorization where the i-th and $(i+1)$-st factors are now given by

$$\cdots \mid xyx^{-1} \mid x \mid \cdots.$$

It is straight forward to check that the standard relations for the braid group are satisfied. We call this the *Hurwitz action* of the k-strand braid group. Notice that every word in the same orbit under this action evaluates to the same element of the group G. This gives us a well-defined Hurwitz action of the k-strand braid group on the minimal positive factorizations on an element w where $k = d(1, w)$.

Remark 8.1.2. (The Hurwitz Action is Transitive) The Hurwitz action on factorizations of a particular element g in a group G lead to relations that can be seen in the interval $[1, g]$. When the Hurwitz action is transitive on factorizations, these relations are sufficient to define the interval group G_g (see Definition 5.4.1). In 2014, Baumeister, Dyer, Stump and Wegener proved transitivity of the Hurwitz action for all Coxeter groups in complete generality [BDSW14]. McCammond and Sulway use this to show that dual Euclidean Artin groups are in fact isomorphic to Euclidean Artin groups in [MS17].

By applying the Hurwitz action of the 3-strand braid group to the factorization $w = re_0^1 \mid re_{od} \mid re_{-1}^0$, we can determine all possible factorizations of w with respect to the dual generating set.

Proposition 8.1.3. *(Dual Factorization of w) Every factorization of w over the dual generating set is given by one of the following for some $n \in \mathbb{Z}$:*

$$\begin{aligned}
w &= re_{2n}^{2n+1} \mid re_{2n-2}^{2n-1} \mid re_{od} \\
&= re_{2n+1}^{2n+2} \mid re_{2n-1}^{2n} \mid re_{ev} \\
&= re_{2n-1}^{2n} \mid re_{2n-2}^{2n-1} \mid re_{2n-3}^{2n-2} \\
&= re_{2n-1}^{2n} \mid re_{ev} \mid re_{2n-2}^{2n-1} \\
&= re_{2n}^{2n+1} \mid re_{2n-1}^{2n} \mid re_{2n-2}^{2n-1} \\
&= re_{2n}^{2n+1} \mid re_{od} \mid re_{2n-1}^{2n} \\
&= re_{ev} \mid re_{2n}^{2n+1} \mid re_{2n-2}^{2n-1} \\
&= re_{od} \mid re_{2n+1}^{2n+2} \mid re_{2n-1}^{2n}
\end{aligned}$$

Proof. Straightforward computations reveal that the above products are indeed factorizations of w. We recall from Remark 8.1.2 that the Hurwitz action is transitive. Furthermore, the list of factorizations given above is closed under the Hurwitz action. That is, applying the Hurwitz action to any one of the factorizations given above will result in one of the other given factorizations. □

Each of the factorizations of w can be found in Figure 8.1. Starting at a particular factorization, we can arrive at another factorization by applying either σ_1 or σ_2 which is denoted by a single arrow or double arrow respectively. Crossing over the red dashed

lines will change the particular factorization by either adding 1 to or subtracting 1 from the subscript n.

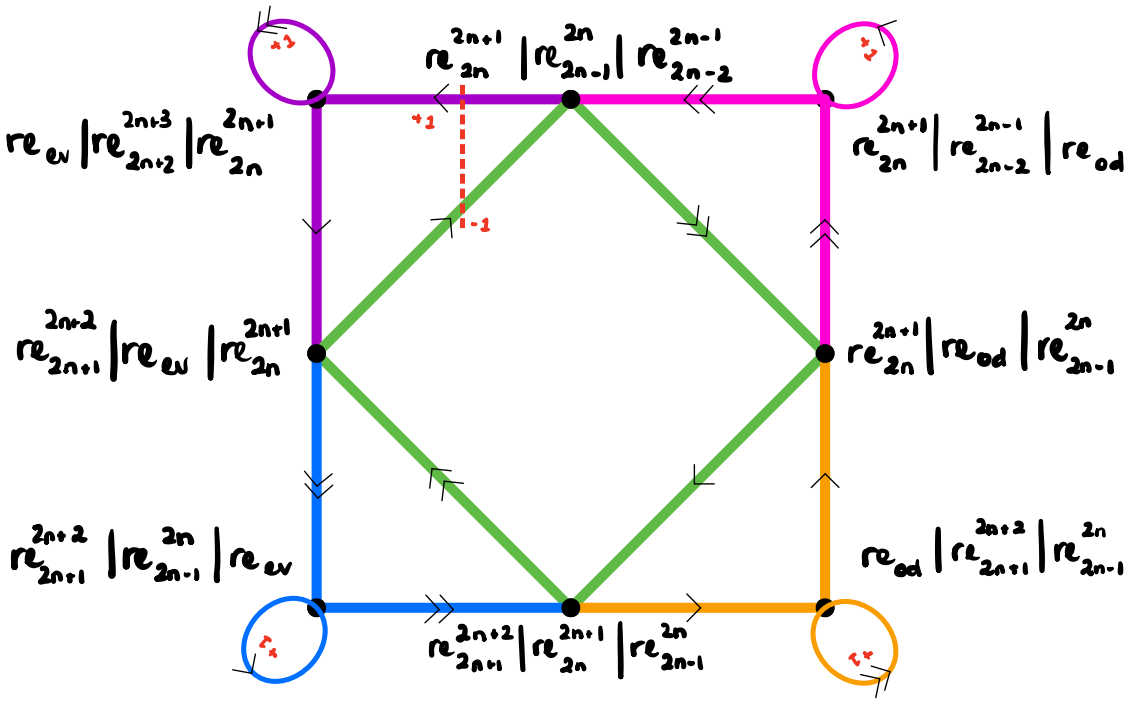


Figure 8.1: Factorizations of w

The Hurwitz action on $w = re_0^1 \mid re_{od} \mid re_{-1}^0$ over the dual generating set gives infinitely many factorizations of w that can be represented by the vertices of the image on the left of Figure 8.2. Because this image is periodic with respect to the four, colored triangles, so too are the general factorizations of w in the dual generating set. The particular Hurwitz action and some of the factorizations are given in the image on the right of Figure 8.2.

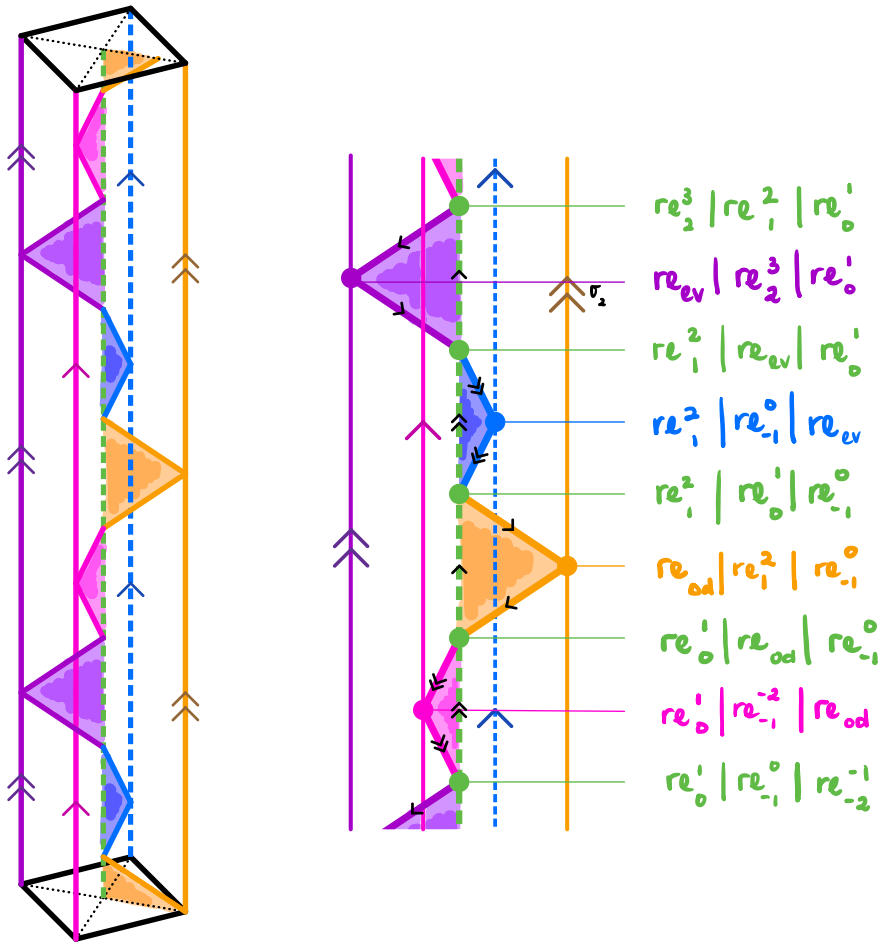


Figure 8.2: Graph of the Hurwitz action on factorizations of w

8.2 The Hurwitz Action on the Standard Generating Set

We now wish to apply the Hurwitz action on $w = a \mid b \mid c$. In this case, an application of the Hurwitz action will result in new factors that are words in the standard generating set.

Example 8.2.1. For a concrete example, we apply $\sigma_1 \in \text{Braid}_3$ to the factorization $w = a \mid b \mid c$. This results in the new factorization $w = aba^{-1} \mid a \mid c$. The elements a and c can be directly translated to the dual generators re_0^1 and re_{-1}^0 respectively. This implies

that the element aba^{-1} represents a dual generator whose right complement is equal to the product $re_0^1 \cdot re_{-1}^0 = ro_0$. Using Table 7.1, we conclude that $aba^{-1} = re_{-2}^{-1}$.

Remark 8.2.2. (Notation for Inverses) In Section 8.2, we write each dual generator in terms of the standard generating set. To distinguish between inverses of the standard generators and the superscripts and subscripts used in the dual generators, we let A denote the inverse of the the standard generator a. Similarly, B and C will denote the inverses of the standard generators b and c. Hence, the dual generator re_{-2}^{-1} from Example 8.2.1 can be written as abA in the standard generating set.

By repetitively applying the Hurwitz action to the factorization $w = a \mid b \mid c$, we can now translate each dual generator into words with respect to the standard generating set. The function given in Listing 8.1 performs the Hurwitz action on a particular factorization. We also refer to this action as a *twist*.

Listing 8.1: The Hurwitz Action function

```
def twist(tuple,n):
    if n == 1:
    newtuple1 = (tuple[0]*tuple[1]*tuple[0]^-1,tuple[0],tuple[2])
    return(newtuple1)
    elif n == 2:
    newtuple2 = (tuple[0],tuple[1]*tuple[2]*tuple[1]^-1,tuple[1])
    return(newtuple2)
    elif n == -1:
```

```
    newtuplem1 = (tuple[1],tuple[1]^-1*tuple[0]*tuple[1],tuple
       [2])
    return(newtuplem1)
    elif n == -2:
    newtuplem2 = (tuple[0],tuple[2],tuple[2]^-1*tuple[1]*tuple
       [2])
    return(newtuplem2)
```

To apply a sequence of twists to a particular factorization of w, we can use the function in Listing 8.2 where the sequence of twists should be given as a list of integers where the integer i represents the action of σ_i on the factorization.

Listing 8.2: Sequence of twists function

```
def twistsequence(tuple, listofns):
    newtuple = tuple
    print newtuple
    for i in range(0,len(listofns)):
        newtuple = twist(newtuple, listofns[i])
        print(newtuple, i+1)
    return(newtuple)
```

Though the functions given will help to quickly write dual generators in terms of standard generators, Figure 8.2 can help us to quickly transcribe the dual generators as well. Let x represent an arbitrary dual generator. We can locate a particular factorization $x \mid y \mid z$ in Figure 8.2 that includes x as a factor. There is a path in the Hurwitz graph

beginning at the factorization $w = re_0^1 \mid re_{od} \mid re_{-1}^0$ and ending at the factorization $x \mid y \mid z$. The edges of that path represents a sequence of twists to apply to the factorization $w = a \mid b \mid c$ and what results are three new factors that are words in the standard generating set that represent the dual generators x, y, and z respectively. Though different paths produce different words, the words represent equivalent elements. Once we have a word representing a particular element, we choose representatives that fit into nice patterns.

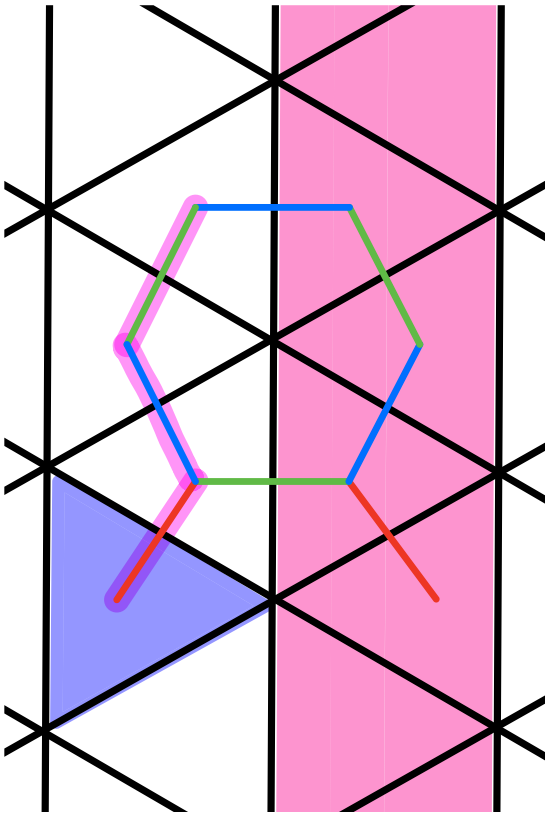


Figure 8.3: Translating horizontal reflections via paths in the Coxeter complex

In what follows, we transcribe every dual generator into words with respect to the standard generating set. We mention that there are a number of equivalent ways to transcribe these elements and the particular transcription that you see below was chosen for several reasons. We note that the chosen word does not always minimize the length of

the word. However, the transcriptions of the dual generators do have a special relationship upon conjugation by w.

We start by transcribing the horizontal reflections. We know that $re_{od} = b$ so what remains is a translation of re_{ev}. The word of shortest length representing re_{ev} is given by $re_{ev} = acA = Cac$. These transcriptions can also be done about the strip in the Coxeter complex of type $\widetilde{A}_2$, see Figure 8.3. The colors in the paths represent each of the standard generators and a highlighted path represents the inverse of that particular generator. By traveling up the strip, crossing the even axis and then traveling back down to the image of $re_{ev} \curvearrowright \tau$, we yield infinitely many transcriptions for both of our horizontal reflections. For example, according to Figure 8.3, the shortest path leads to the word acA representing re_{ev} whereas the longer path indicates the word $abcbCBA$ could also represent re_{ev}.

Remark 8.2.3. (Horizontal reflections and conjugation by W) Suppose $re_{od} = b$ and $re_{ev} = acA$. Upon conjugation by W, we can use the standard relation $cbC = Bcb$ of $\textsc{Art}(\widetilde{A}_2)$ to see that

$$b^W = wbW = (abc)b(CBA) = ab(Bcb)BA = a(bB)c(bB)A = acA = re_{ev}.$$

Hence, conjugation of a representative of a horizontal reflection by W results in a word representing the other horizontal reflection. Furthermore, conjugating $re_{ev} = Cac$ by W^2 results in the following

$$wabc(Cac)CBAW = wa(baB)AW = waAbaAW = ab(cbC)BA = abBcbBA = acA = Cac.$$

This indicates the the horizontal reflections each commute with W^2.

To transcribe the vertical reflections, we apply a series of twists with the function in Listing 8.2 to the factorization $w = a \mid b \mid c$ and using Table 7.1, we get the following:

Table 8.1: Transcribing vertical reflections

$\vdots$	$\vdots$	$\vdots$	$\vdots$
re_8^9	$(abc)^3c(CBA)^3$	re_7^8	$(abc)^2abA(CBA)^2$
re_6^7	$(abc)^2a(CBA)^2$	re_5^6	$(abc)^2c(CBA)^2$
re_4^5	$(abc)abA(CBA)$	re_3^4	$(abc)a(CBA)$
re_2^3	$(abc)c(CBA)$	re_1^2	abA
re_0^1	a	re_{-1}^0	c
re_{-2}^{-1}	$(CBA)abA(abc)$	re_{-3}^{-2}	$(CBA)a(abc)$
re_{-4}^{-3}	$(CBA)c(abc)$	re_{-5}^{-4}	$(CBA)^2abA(abc)^2$
re_{-6}^{-5}	$(CBA)^2a(abc)^2$	re_{-7}^{-6}	$(CBA)^2c(abc)^2$
re_{-8}^{-7}	$(CBA)^3abA(abc)^3$	re_{-9}^{-8}	$(CBA)^3a(abc)^3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

We mentioned previously that the chosen representatives for each dual generator did not minimize the length of the word. For example, the dual generator re_2^3 is given by $(abc)c(CBA)$ which can be simplified to $abcBA$. However, the general patterns with respect to conjugation by abc would be harder to observe should we choose the reduced word to represent the dual generators. We remark more on this following Proposition 8.2.5. Notice that conjugation by w increases or decreases the subscripts of vertical reflections by 3. Hence, every vertical reflection will be a conjugate of the standard representatives that we chose for re_{-1}^0, re_0^1 and re_1^2.

Remark 8.2.4. (Conjugating vertical reflections by w) Within each infinite family, every third vertical reflection is achieved by conjugation by w^2. For concreteness, notice the following closed forms for every third element that includes the element re_0^1:

$$\begin{array}{ccccccccc} \cdots & \to & re_{-6}^{-5} & \to & re_0^1 & \to & re_6^7 & \to & \cdots \\ \cdots & \to & (CBA)^2a(abc)^2 & \to & a & \to & (abc)^2a(CBA)^2 & \to & \cdots \end{array}$$

We also observe that we can move between the two infinite families, by conjugating by w. For example, conjugating $re_0^1 = a$ by w yields the generator $re_{-3}^{-2} = (CBA)a(abc)$. These relationships are observed within both infinite families of the vertical reflections and rotations which we will see below.

Table 8.2: Transcribing rotations

$\vdots$	$\vdots$
ro_6	$(abc)^2ac(CBA)^2$
ro_4	$(abc)ab(CBA)$
ro_2	$(abc)bc(CBA)$
ro_0	ac
ro_{-2}	$(CBA)ab(abc)$
ro_{-4}	$(CBA)bc(abc)$
ro_{-6}	$(CBA)^2ac(abc)^2$
$\vdots$	$\vdots$

$\vdots$	$\vdots$
ro_7	$(abc)^2ab(CBA)^2$
ro_5	$(abc)^2bc(CBA)^2$
ro_3	$(abc)ac(CBA)$
ro_1	ab
ro_{-1}	bc
ro_{-3}	$(CBA)ac(abc)$
ro_{-5}	$(CBA)^2ab(abc)^2$
$\vdots$	$\vdots$

Since all of the height 2 elements of $\textsc{Art}(\widetilde{A}_2)$ can be recovered by taking products of the height 1 elements, we use the tables above to generate the closed forms for the translations as well. Recall that the product of any two consecutive vertical reflections from an infinite family results in a translation. Using Table 8.1 we see that tr_{ev} and tr_{od}

are given by

$$tr_{ev} = re_2^3 \cdot re_0^1 = (abc)c(CBA) \cdot a = abcB = wB,$$

$$tr_{od} = re_1^2 \cdot re_{-1}^0 = abA \cdot c = Babc = Bw.$$

We note that these are not the only words in the standard generators that represent the translations. In particular, any other factorization would yield words that are w^2 conjugates of the ones given above. Similar to the remark made earlier regarding the horizontal reflections, we note here that conjugating tr_{ev} by w results in tr_{od} since

$$tr_{ev}^w = W(wB)w = Bw = tr_{od}.$$

After transcribing each dual generator, we conclude with a proposition that describes all possible representatives of dual generators in the standard generating set. Because all dual generators can be written with respect to the words in Proposition 8.2.5 in the standard generating set, so too can their inverses.

Proposition 8.2.5. *(Dual generators as words in the standard generating set) Let $w = abc$ and $W = CBA$. All of the dual generators can be represented by one of the following words in the standard generating set. Variations of representatives for horizontal reflections and translations may occur as W^2 and w^2 conjugates respectively.*

$$
\begin{aligned}
re_{ev} &= acA \\
re_{od} &= b \\
re_{-3k-1}^{-3k} &= c^{w^k} \\
re_{-3k}^{-3k+1} &= a^{w^k} \\
re_{-3k+1}^{-3k+2} &= (abA)^{w^k} \\
tr_{ev} &= wB \\
tr_{od} &= Bw \\
ro_{-3k-1} &= (bc)^{w^k} \\
ro_{-3k} &= (ac)^{w^k} \\
ro_{-3k+1} &= (ab)^{w^k}
\end{aligned}
$$

Again we note that these representatives could be simplified. For example, consider the generators of the form $re_{-3k}^{-3k+1} = a^{w^k}$. For positive values of k, we see that

$$a^{w^k} = (CBA)^k a(abc)^k = (CBA)^{k-1} CBabc(abc)^{k-1}.$$

We see a very minimal amount of cancellation of just a single pair of letters and only when k is positive. For this reason, we reinforce the choice of using unreduced words to represent the dual generators.

9. A New, Standard Solution to the Word Problem

In Chapter 7, we saw that we can discuss the dual algorithm algebraically in terms of the infinite generating set. In Chapter 8, we saw that each of these dual generators can be written as one of a finite set of representatives conjugated by w to some power. With this, we describe all of the update rules in Table 7.2 in terms of the standard generators in Section 9.1. We conclude this chapter using strips in the Coxeter complex to see the updates described in standard version of the dual algorithm.

9.1 A Standard, Positive Normal Form

Chapter 8 allows us to translate the dual algorithm in the dual generators to an algorithm using the standard generators. As soon as we translate the update rules, we can put every positive word with respect to the standard generating set into normal form. In what follows, we do just that.

We start by providing four tables similar to those given in Chapter 7 which gives the same update rules as the dual algorithm but now in terms of the standard generating set. For clarity, we let $w = abc$ and $W = w^{-1} = CBA$. We note that a direct transcription may need to be simplified using the standard relations but the results do indeed match those found in Table 7.2.

We see similar patterns emerge as we put words in the standard generating set representing dual generators in normal form. We see the patterns emerge in the tables in terms of conjugation by powers of w.

The table putting transcribed generators from $S_2 \times S_1$ in normal form is identical to that of Table 7.3 with the exception that the row and column labels contain the transcribed dual generators, so we omit that transcription here. We begin with Table 9.1 which puts transcribed generators from $S_1 \times S_1$ in normal form. We see that the $2 \times \infty$ and the $\infty \times 2$ parts of the table contain conjugates of ab, ac and bc. In the $\infty \times \infty$ part of the table, the diagonal consists of conjugates of ab, bc, and ac and the superdiagonal alternates between wB and Bw.

Table 9.1: $S_1 \times S_1$ Generators in Normal Form

$x \setminus y$	acA	b	$\cdots$	abA	a	c	$WabAw$	$\cdots$
acA			$\cdots$		ac		$Wabw$	$\cdots$
b			$\cdots$	ab		bc		$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$				
wcW		$wacW$		$wbcW$	wB			
abA	$wbcW$				ab	Bw		
a		ab				ac	wB	
c	ac						bc	$\ddots$
$\vdots$	$\vdots$	$\vdots$						$\ddots$

Table 9.2 contains transcribed products of dual generators of S_2 into normal form. The $2 \times \infty$ and the $\infty \times 2$ parts of this table contains a product of w and conjugates of a, c and abA by powers of w. In the $\infty \times \infty$ part of the table, the diagonal consists of alternating products of $w \mid b$ and $w \mid acA$. The super diagonal consists of products of w's and conjugates of a, c, and abA.

Last, we provide a table of transcribed generators from $S_1 \times S_2$ in normal form. In Table 9.3 we still see a diagonal of w since we've arranged for the right complements of

Table 9.2: $S_2 \times S_2$ Generators in Normal Form

$x \setminus y$	Bw	wB	$\cdots$	$wbcW$	ab	ac	bc	$\cdots$
Bw			$\cdots$	$w \mid wcW$		$w \mid a$		$\cdots$
wB			$\cdots$		$w \mid abA$		$w \mid c$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$				
$wacW$	$w \mid c$			$w \mid acA$	$w \mid a$			
$wbcW$		$w \mid WabAw$			$w \mid b$	$w \mid c$		
ab	$w \mid Waw$					$w \mid acA$	$w \mid WabAw$	
ac		$w \mid Wcw$					$w \mid b$	$\ddots$
$\vdots$	$\vdots$	$\vdots$						$\ddots$

the rows to appear as columns. In the $2 \times \infty$ portion of the table, we see products of the conjugates of ab, ac and bc with either acA or b. In the $\infty \times 2$ portion of the table, we see products of Bw or wB with conjugates of a, abA and c. In the $\infty \times \infty$ part of the table, the superdiagonal consists of products of either Bw or wB with conjugates of a, c and abA by powers of w. The subdiagonal consists of products of conjugates of ac, ab, and ac with either acA or b.

Table 9.3: $S_1 \times S_2$ Generators in Normal Form

$x \backslash rc(x)$	Bw	wB	$\cdots$	w^2acW^2	w^2bcW^2	$wabW$	$wacW$	$\cdots$
b	w		$\cdots$	$w^2bcW^2 \mid acA$		$wacW \mid acA$		$\cdots$
acA		w	$\cdots$		$wabW \mid b$		$wbcW \mid b$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$				
w^2abAW^2	$Bw \mid waW$		$\ddots$	w	$Bw \mid wabAW$			
w^2aW^2		$wB \mid wcW$		$w^2acW^2 \mid acA$	w	$wB \mid waW$		
w^2cW^2	$Bw \mid abA$				$w^2bcW^2 \mid b$	w	$Bw \mid wcW$	
$wabAW$		$wB \mid a$				$wabW \mid acA$	w	$\ddots$
$\vdots$	$\vdots$	$\vdots$					$\ddots$	$\ddots$

We recognize that Tables 9.1 through 9.3 contain entries that could be simplified by removing the bars that distinguish the factorizations. However, doing so would make it harder to recognize what dual normal form it came from. Distinct normal forms have distinct reductions so there is no harm in leaving words in unreduced normal form.

9.2 A Standard Normal Form

We proceed with a discussion of how to put an arbitrary word in the standard generating set into normal form. As in Chapter 7, we put words into normal form using three steps but given a word in the standard generating set, we require an additional, initial step.

Suppose we are given an arbitrary word u written in the standard generating set of $\textsc{Art}(\widetilde{A}_2)$. Our goal is to determine whether or not this word represents the identity. To do so, we proceed as follows. The first step is to breaking the word u into subwords $u_1 \mid u_2 \mid \cdots \mid u_n$ where each u_i represents a dual generator or the inverse of a dual generator. Concretely, we break the word into letters since each letter in an arbitrary word in the standard generating set represents the dual generators re_0^1, re_{od}, re_{-1}^0 or their inverses. Once we have subwords that represent dual generators we can follow the steps outlined in Section 7.4. Step two would be to replace each of the letters that represent inverses of dual generators using the fact that $u^{-1} = w^{-1} \mid lc(u)$ being sure to use Proposition 8.2.5 to describe the left complements. In step three, we use conjugation by w to rewrite the word in the form

$$w^k \mid v_1 \mid v_1 \mid \cdots \mid v_n$$

where k is a nonpositive integer and each v_i is a subword in the standard generating set representing a dual generator. In step four, we proceed to put the subword given by $v_1 \mid v_1 \mid \cdots \mid v_n$ into normal form using Section 9.1. Upon concatenating this subword with the nonpositive powers of w, cancellation will again occur about the powers of w and what results is a word in the standard generating set that is in normal form.

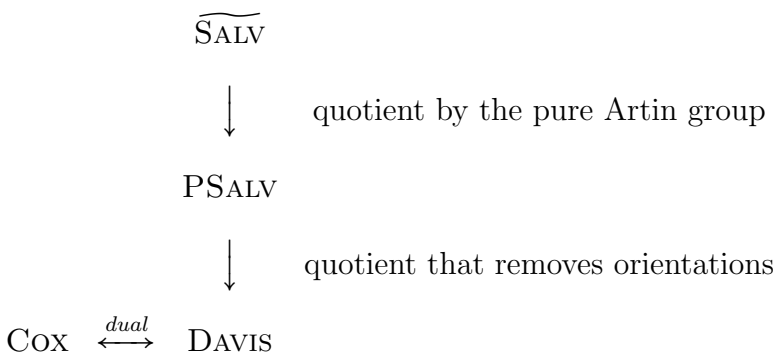
9.3 Standard Actions on Fundamental Chambers

The dual Garside structure for the Artin group of type $\widetilde{A}_2$ was constructed from the action of the Coxeter group $\textsc{Cox}(\widetilde{A}_2)$ on its Coxeter complex. This gives us the Garside structure for $\textsc{Art}^*(\widetilde{A}_2, w)$ that we used to describe the dual algorithm in Chapter 7. Geometrically, we can see how the algorithm works using actions of dual generators on strips or standard generator paths in the Coxeter complex.

The normal forms described in Chapter 7 and Section 9.2 can be represented by paths in the universal cover of the Salvetti complex $\widetilde{\textsc{Salv}}$. We can recognize whether something is in normal form in $\widetilde{\textsc{Salv}}$ by looking at the image of the paths in the pure Salvetti complex PSalv. We can also do this schematically by observing how strips move around in the Coxeter complex Cox.

With the standard generators, paths can be realized using the Davis complex, which is dual to the Coxeter complex, given by the hexagonal tiling of the Euclidean plane. The reader should keep in mind that because these paths are really in the pure Salvetti complex, each of the edges in the hexagonal tiling (see Figure 6.1) is actually a pair of oriented edges, and each hexagon is really 6 oriented hexagons. We take care to

differentiate between an edge representing a standard generator or its inverse as we proceed.



Example 9.3.1. We start by considering an arbitrary product of dual generators abc and its action on the fundamental chamber τ. The action $(abc).\tau$ can be rewritten as $(abcBA).(abA).(a.\tau)$. The action of $(abc).\tau$ is what we call a simple generator description of the action. Rewriting this action as $(abcBA).(abA).(a.\tau)$, that is, the action of a followed by a conjugate of b, followed by a conjugate of c is what we call a path description. In particular, the action of abA on $a.\tau$ is really what the b reflection looks like after being acted on by a and this corresponds to flipping across the blue edge in Figure 9.1. Similarly, $abcBA$ is what c looks like after being acted on by ab which corresponds to flipping across the green edge in Figure 9.1.

We see that the action of abc on τ from right to left results in reflecting about the sides of the fundamental chamber given by the images $c.\tau$, $(bc).\tau$, and $(abc).\tau$. After rewriting the action using conjugation, acting from left to right allows us to view the action on each updated fundamental chamber given by the images $a.\tau$, $(abA).(a.\tau)$, and ending with $(abcBA).(abA).(a.\tau)$ which is equivalent to $(abc).\tau$. This gives us a local notion of a fundamental chamber. In general, multiplying right to left can be viewed as compositions

of the functions that are reflections in the sides of the fundamental chamber. On the other hand, acting from left to right with implicit conjugations allows us to reinterpret the action as a path through each new fundamental chamber.

In fact, every word in the standard generating set can be rewritten so that the action of the word on the fundamental chamber reveals a path in the Coxeter complex using the above rewriting process via conjugation.

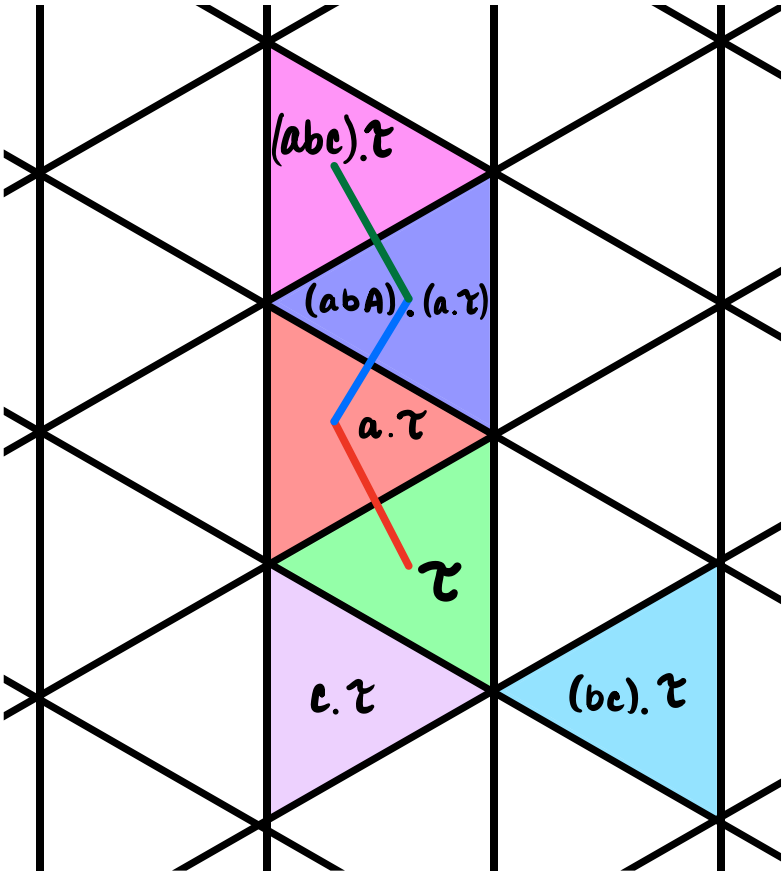


Figure 9.1: Simple generator descriptions and path descriptions

Remark 9.3.2. The reason we distinguish between the generator description and the path description is as follows. In ordinary Coxeter groups, a path has a reduction if there is a pair of simples that can be removed from the word. Algebraically, this may look

strange but geometrically, the reduction corresponds to a path that crosses a hyperplane twice. Moreover, with the generator description, a more complicated action could move the fundamental chamber in abrupt ways throughout the Coxeter complex where the sequence of images of the chambers from the action may not be anywhere near each other. However, with the path description, the action is a local one where each subsequent image in the action will result in a new fundamental chamber that shares a facet with the previous fundamental chamber. With this, we can use this local picture to determine when words are in normal form.

9.4 Dual Actions on Strips

There is a similar way of looking at dual generators acting on strips. We start with the simple example with the action of the product $re_4^5 re_0^1$ on the strip. The pink strip in Figure 9.2 contains the fundamental chamber τ and is the strip we'll be acting on. Starting with the right to left action, the pink strip gets moved to the orange strip and the fundamental chamber τ gets moved to its image $re_0^1.\tau$. We then act on the orange strip by the reflection re_4^5, which is a reflection about the line containing the vertices 4 and 5 in the pink strip. This takes the orange strip to the blue strip in Figure 9.2 with the image of τ under this action indicated by the label $re_4^5.(re_0^1.\tau)$.

On the other hand, acting from left to right using conjugation presents the equivalent action of $(re_4^5 re_0^1 (re_4^5)^{-1}).(re_4^5.\tau)$. In this case, this pink strip will move to the yellow strip with τ moving to the image $re_4^5.\tau$. Moving the yellow strip to the blue strip can be done by performing the reflection re_0^1 within the yellow strip (i.e. the reflection about the

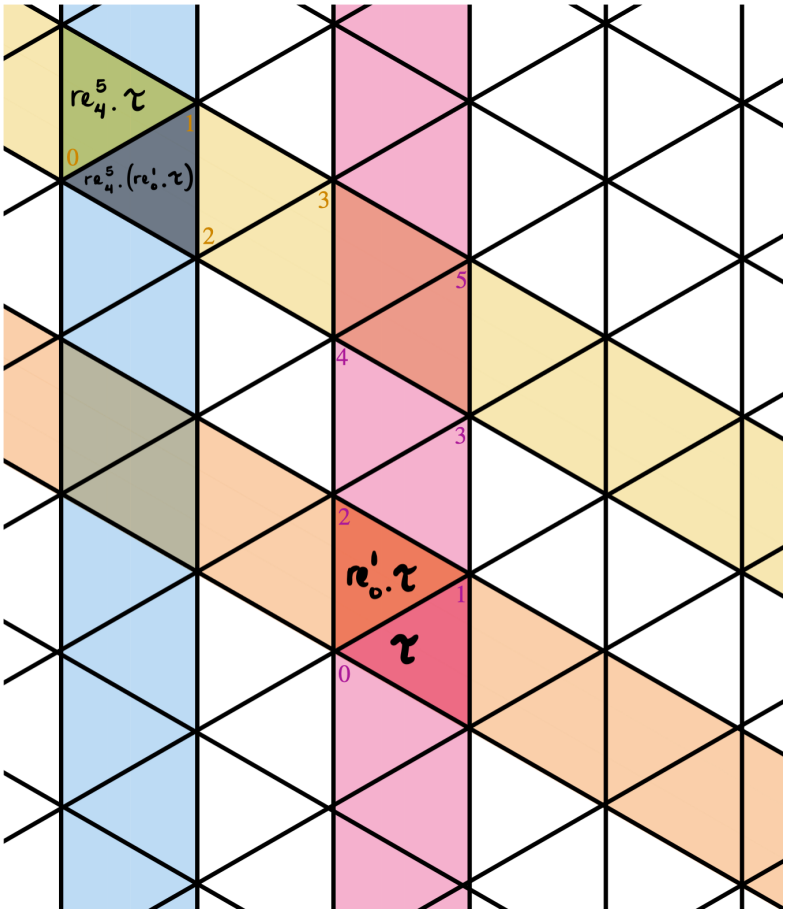


Figure 9.2: Right to left versus left to right actions of dual generators on strips

line containing the vertices $re_4^5.v_0$ and $re_4^5.v_1$ in the yellow strip where v_0 and v_1 represent the vertices 0 and 1 within the pink strip, respectively). Again, the rewritten left to right action allows us to perform each subsequent action **locally**, using only the yellow strip to proceed rather than using the reflections with respect to the pink strip.

With these geometric conventions, we can see the dual algorithm in dual generators and the dual algorithm in the standard algorithms using actions on based strips and their fundamental chambers using the right to left action. According to the dual algorithm, putting the product $re_2^3 \mid ro_0$ into normal form yields the new product $tr_{ev} \mid re_{-1}^0$. Geometrically, the product being updated so that it is in normal form can be viewed

as updating a sequence of three strips in the Coxeter complex; the first strip being the initial pink strip we start with, the second strip being the image of the pink strip under re_2^3 which is then updated to the image of the pink strip under tr_{ev}, and the third strip given by the blue strip achieved by an action of a conjugate of either ro_0 or re_{-1}^0. We can see this explicitly in Figure 9.3. The right to left action of the product $re_2^3 \mid ro_0$ takes the pink strip, to the orange strip (by re_2^3), to the blue strip (by a conjugate of ro_0). The action given by the updated product $tr_{ev} \mid re_{-1}^0$ takes the pink strip to the yellow strip (by tr_{ev}), followed by taking the yellow strip to the blue strip (by a conjugate of re_{-1}^0). We see that the reflection given by the orange strip is updated to a translation given by the yellow strip which corresponds to what we'd expect algebraically.

With respect to the standard algorithm, $wcW \mid ac$ gets updated to $wB \mid c$. We can view this update using paths within these strips. Consider first, $wcW \mid ac$. In Figure 9.3, the first path, given by wcW (which is equivalent to $abcBA$) would take the black vertex to the brown vertex. Then (the local version of) ac would take the brown vertex to the black vertex. Updating the product to $wB \mid c$ first takes the black vertex to the purple vertex via the path given by wB. The purple vertex then moves to the black vertex via the path indicated by c. We see the initial path cross over a hyperplane twice whereas the updated path, given by the standard normal form of the original product, does not. We note that there is an oriented blue edge and green edge along the path that crosses the same hyperplane (that of the line containing the pink vertices 1 and 2) at different points in the path but this is not one of interest in the path simplification.

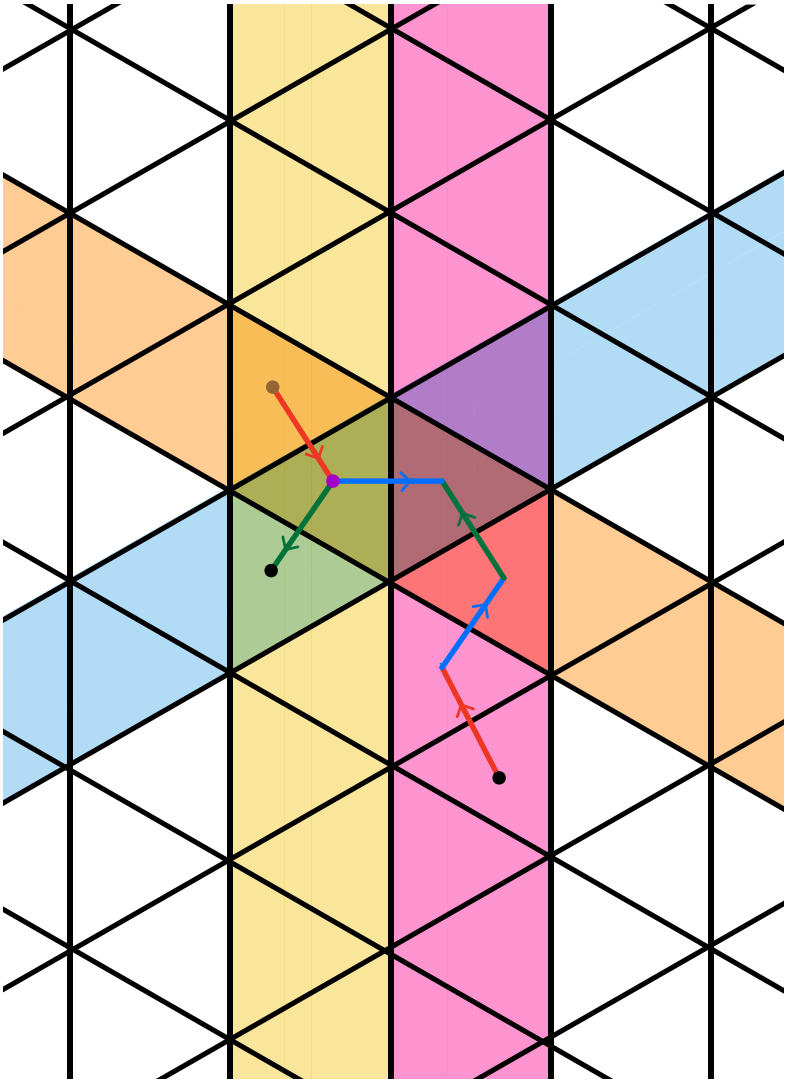


Figure 9.3: The dual algorithm as strips and the standard algorithm as paths

Remark 9.4.1. (A geometric algorithm) Algebraically, our algorithms determine how to put all positive products into normal form. By updating triples of strips, the same can be done geometrically so that all of the algebraic updates can be visualized. In particular, because each strip is invariant under conjugation by w, there are finitely many ways that our strips can move (i.e. a reflection about the sides of the strip, a reflection about a diagonal crossing through the strip, a rotation about a vertex on the strip, or a translation). In the same way that each of the quadrants in Tables 7.3 through

7.6 had general updates, so will the triple of strips that describe the products in each of the quadrants. In cases where two elements of height one combine to form an element of height two, the middle strip may vanish. In the case of the example given by Figure 9.3, we see the orange strip updated to the yellow strip. In this sense, one could transcribe the tables of updates given in Section 7.3 so that they read as updates of triples of strips instead of updates of a product of two generators.

10. Future Work

This final chapter discusses some of the natural directions for future research regarding the word problem for Euclidean Artin groups. In particular, the work discussed in this chapter will make use of the known Garside structures for these groups.

We start by noting that in Chapter 9, the standard solution relies on recognizing whether or not a subword in the standard generating set represents a dual generator. We recall that some our dual generators have multiple representations and the ones that we chose depend on conjugation by w^k. Because recognizing a word in the standard generating set also depends on being able to recognize conjugation by w^k Turing machines are not able to the algorithm described in the previous chapter. One would need a machine that has a counter. In the future, it would be interesting to determine exactly the type of language being used to perform these algorithms and exactly what type of machine is able to do this.

A natural next step would be to apply the approach described throughout Part II to produce a standard solution to the word problem for all Euclidean Artin groups of type $\widetilde{A}_n$. Because the dual Euclidean Artin group of type $\widetilde{A}_n$, with a careful choice of Coxeter element, is a Garside group, finding a dual solution to the word problem should be straightforward. To do this, the algorithms in the dual Euclidean Artin group must be made more explicit. Beyond this, we would need to determine a nice way to transcribe all dual generators into words in the standard generating set so that a standard solution can be written. Though the techniques described in Part II should generalize, we expect

slight differences to occur, especially geometrically, depending on the rank of the Artin group.

In Chapter 5, we mentioned that dual Euclidean Artin groups of type $\widetilde{A}_n$, $\widetilde{C}_n$ and $\widetilde{G}_2$ are the only ones that had Garside structures that were fairly well-understood. Because $\textsc{Art}^*(\widetilde{C}_n, w)$ and $\textsc{Art}^*(\widetilde{G}_2, w)$ are nonstandard Garside groups, we should be able to produce a dual solution for for these groups in ways similar to the one found for $\textsc{Art}^*(\widetilde{A}_2, w)$. Then similarly, we hope to use that dual solution to produce a standard solution to the word problem for $\textsc{Art}(\widetilde{C}_n)$ and $\textsc{Art}(\widetilde{G}_2)$ as well.

A. Appendix

A.1 An Algorithm Solving the Word Problem for Art(A_n)

This section contains the various dictionaries and functions used to create the algorithm that solves the word problem for Art(A_n). Each function contributes to the overall algorithm. In what follows, each function will be preceded by a short description of the function. With each function defined, the overall algorithm will run by initiating the code in Listing A.1 which computes and stores all of the necessary components of the algorithm that is found throughout this section of the appendix.

Listing A.1: Initialization Code

```
def init(n):
global D, genperms, RD, RCD, LCD, DescD, AscD, MM, JM, pn2npM
D = PermutationDictionary(n)
genperms = genPerms(D)
RD = ReversePermutationDictionary(D)
RCD = RComplementDictionary(n,D)
LCD = LComplementDictionary(n,D)
DescD = ImmediateDescendantsDictionary(D,RD,genperms)
AscD = ImmediateAscendantsDictionary(D,RD,genperms)
MM = MeetsMatrix(DescD)
JM = JoinsMatrix(MM)
```

```
pn2npM = posneg2negposM(n)
```

To start, one should declare the number of strands of the braid group of interest with the input `init(n)`.

We begin with some of the smaller functions that allow us to do describe and sort our elements. At times, we need to sort lists or tuples according to a specific entry. In particular, the following function in Listing A.2 is used to sort a tuple of tuples by the first entry.

Listing A.2: Sort Tuples by Entry

```
def getKey(item):
    return item[0]
```

Recall that braids can be represented by permutations and products of braids can be found by composing permutations. Since the computer algebra system interacts nicely with lists, we represent the permutations using a list of integers. The function in Listing A.3 composes two lists as though they were permutations.

Listing A.3: Multiplying Permutations as Lists

```
def ListMult(n,tupleA,tupleB):
    return(tuple(tupleB[tupleA[i]-1] for i in range(n)))
```

The need to compute the inverse of a permutation also arises. The function in Listing A.4 computes the inverse of a list of integers as though it were a permutation.

Listing A.4: Inverses of Permutations as Lists

```
def ListInverse(n,PermAsList):
    tuplepairs = tuple((i+1, PermAsList[i]) for i in range(n))
    swaptuplepairs = tuple((j,i) for (i,j) in tuplepairs)
    sortswaptuplepairs = sorted(swaptuplepairs, key=getKey)
    return(tuple(j for (i,j) in sortswaptuplepairs))
```

Performing operations with these lists may be difficult to keep track of. Hence, we represent each braid or permutation with an integer values. The integer values are assigned to the 23 generators according to height. To keep track of the integer assignments, we create a dictionary. The dictionary has integer keys whose corresponding values are permutations as lists sorted by length or height. The function in Listing A.5 defines said dictionary.

Listing A.5: Permutation Dictionary

```
def PermutationDictionary(n):
    ll = [[Permutation(i).length(),list(i)] for i in Permutations
       (range(1,n+1))]
    ll.sort()
    D = dict((i, tuple(ll[i][1])) for i in range(0,len(ll)))
    return(D)
```

Because of the one-to-one correspondence between the 23 braid elements that we will be using as generators and the 23 nontrivial elements of the symmetric group on 4 letters, we take the generators to be these 23 nontrivial permutations. The function below in

Listing A.6 takes these permutations and writes them as lists. These lists will represent our generators.

Listing A.6: Permutations as Lists

```
def genPerms(D):
    genperms = []
    for i in range(1, n):
        genperms.append(D[i])
    return(genperms)
```

At times, it is helpful to be able to call on the value of a dictionary rather than first identifying the key to get the assigned value. Thus, we create a reverse dictionary using Listing A.7 to be used when doing value to key conversions. Recall that our keys for this dictionary will be lists or tuples rather than integers.

Listing A.7: The Reverse Dictionary

```
def ReversePermutationDictionary(D):
    RD = dict((v, k) for k, v in D.iteritems())
    return(RD)
```

The function in Listing A.8 is yet another dictionary that has as a key, the same integer values that represent our braid generators and a value that represents the right complement (see Section 4.4) of the braid generator.

Listing A.8: Right Complement Dictionary

```
def RComplementDictionary(n,D):
    max = D[len(D)-1]
    RCompD = dict((i, ListMult(n,ListInverse(n,D[i]),max)) for i
       in range(0,len(D)))
    return(RCompD)
```

Analogously, we compute the left complement dictionary using the function in Listing A.9:

Listing A.9: Left Complement Dictionary

```
def LComplementDictionary(n,D):
    max = D[len(D)-1]
    LCompD = dict((i, ListMult(n,max,ListInverse(n,D[i]))) for i
       in range(0,len(D)))
    return(LCompD)
```

Because the braid group is a Garside group, we want to use the fact that our generating set forms a lattice to our advantage. Hence, we need to be able to compute all of the left divisors of a given generator. The following function is used to create a dictionary that consists of just the immediate descendants of a generator. That is, given a generator x an immediate descendant of x is an element k such that $k \cdot h = x$ where h is a positive half-twist of adjacent strands. These descendants correspond to left divisors whose height is one less than the given braid. The function in Listing A.10 returns a list of all such immediate descendants for a given generator.

Listing A.10: Immediate Descendants

```
def ImDe(D,RD,genperms,index):
    list=[]
    for j in range(n-1):
        if RD[ListMult(n,D[index],genperms[j])] < index:
            list.append(RD[ListMult(n,D[index],genperms[j])])
    return(list)
```

The immediate descendants of a braid x is used to find the meet of any two generators. We will also need to compute the join of any two generators and thus, there is a need for computing immediate ascendants as well. That is, elements y such that $y = x \cdot h$ for some positive half-twist of adjacent strands h. The function in Listing A.11 generates a list of all such immediate ascendants for each generator.

Listing A.11: Immediate Ascendants

```
def ImAs(D,RD,genperms,index):
    list=[]
    for j in range(n-1):
        if RD[ListMult(n,D[index],genperms[j])] > index:
            list.append(RD[ListMult(n,D[index],genperms[j])])
    return(list)
```

A feature of our lattice of generators is that every pair of generators have unique meets and joins. We use dictionaries to keep track of immediate descendants and immediate ascendants of each generator. The function in Listing A.12 is a dictionary whose keys are the same integers assigned each braid in the original dictionary and the values of the

dictionary are lists consisting of integer values that correspond to each braid's immediate descendants.

Listing A.12: Dictionary of Immediate Descendants

```
def ImmediateDescendantsDictionary(D,RD,genperms):
    DescD = dict((i,ImDe(D,RD,genperms,i)) for i in range(0,len(D
        )))
    return(DescD)
```

The function in Listing A.13 is used to create the immediate ascendants dictionary whose keys are integers and values are lists of integers representing the immediate ascendants of the corresponding integer.

Listing A.13: Dictionary of Immediate Ascendants

```
def ImmediateAscendantsDictionary(D,RD,genperms):
    AscD = dict((i,ImAs(D,RD,genperms,i)) for i in range(0,len(D)
        ))
    return(AscD)
```

Using the above dictionary, we can now define what we call a *Meets Matrix*, created with the code in Listing A.14, which is an $n! \times n!$ matrix whose ij-th entry is the meet of the i-th and j-th generator.

Listing A.14: The Meets Matrix

```
def MeetsMatrix(DescD):
    List = [[] for i in range(0,len(DescD))]
```

```
    for i in range(0,len(DescD)):
        temp = DescD[i]
        for j in range(0,len(DescD[i])):
            temp = temp + List[DescD[i][j]]
        List[i] = list(set(temp))
    MeetsList = [[0 for i in range(0,len(DescD))] for j in range
       (0,len(DescD))]
    for i in range(1,len(DescD)):
        for j in range(1,len(DescD)):
            Intersection = set(List[i]+[i]+[0]).intersection(set(
               List[j]+[j]+[0]))
            MeetsList[i][j] = max(Intersection)
    return(MeetsList)
```

Similarly, we use the ascendants dictionary to create a Joins Matrix as well. The function Listing A.15 gives the matrix whose ij-th entry is the join of the i-th and j-th generator.

Listing A.15: The Joins Matrix

```
def JoinsMatrix(MM):
    List = [[] for i in range(0,len(AscD))]
    for i in reversed(range(0,len(AscD))):
        temp = AscD[i]
        for j in reversed(range(0,len(AscD[i]))):
            temp = temp + List[AscD[i][j]]
```

```
        List[i] = list(set(temp))
    JoinsList = [[23 for i in range(0,len(AscD))] for j in range
       (0,len(AscD))]
    for i in range(0,len(AscD)-1):
        for j in range(0,len(AscD)-1):
            Intersection = set(List[i]+[i]).intersection(set(List
               [j]+[j]))
            JoinsList[i][j] = min(Intersection)
    return(JoinsList)
```

This is the first function that begins to put words in normal form. We start with the following function which puts the product of a pair of generators into normal form. The pair of generators a and b are given as a two-tuple as an input to the function. The product $a \cdot b$ is put into normal form as the new product $\left[a \cdot \left(a^c \wedge b\right)\right] \cdot b'$ where a^c is the right complement of a, and b' is the element necessary new product is equal to the original product. The function returns a two-tuple that is now in normal form.

Listing A.16: Pairs of Generators in Normal Form

```
def Reduction(a,b):
    c = MM[RD[RCD[a]]][b]
    bprime = ListMult(n,ListInverse(n,D[c]),D[b])
    return(RD[ListMult(n,D[a],D[c])],RD[bprime])
```

To begin putting a product of length greater than two into normal form, we need to be able to apply the previous function multiple times, throughout a list. The following

function performs the previous function's rewrite of a pair of integers at a desired location in a list.

Listing A.17: Rewriting Pairs of Generators at a Specific Spot in a Word

```
def ReduceSpot(listofintegers, spot):
    if Reduction(listofintegers[spot],listofintegers[spot+1]) ==
       (listofintegers[spot],listofintegers[spot+1]):
        return(listofintegers, spot+1)
    if Reduction(listofintegers[spot],listofintegers[spot+1]) !=
       (listofintegers[spot],listofintegers[spot+1]):
        target = list(Reduction(listofintegers[spot],
           listofintegers[spot+1]))
        pos = [spot,spot+1]
        for x,y in zip(pos,target):
            listofintegers[x] = y
        return(listofintegers, spot-1)
```

Using the last two functions, the following finally allows us to put any positive word into normal form. The positive word is input as a list of positive integers and the function returns a new list of nonnegative integers consiting of heights that appear in decreasing order.

Listing A.18: Rewriting a Positive Word

```
def RedPosWord(poslist):
    spot = 0
```

```
    while (spot < len(poslist)-1):
        update = ReduceSpot(poslist,spot)
        print(update)
        poslist = update[0]
        spot = abs(update[1])
    return(poslist)
```

To begin putting an arbitrary word, written in terms of the generators and their inverses, into normal form we need to be able to rewrite that word as the product of a negative subword and a positive subword. The function in Listing A.19 gives us an $n!-1 \times n!-1$ matrix whose ij-th entry rewrites the product $i \cdot -j$ as an equivalent product $-i' \cdot j'$ where the first term is now negative and the second term is now positive.

Listing A.19: The Word Reversing Matrix

```
def posneg2negposM(n):
    List = [[x for i in range(0,factorial(n))] for j in range(0,
       factorial(n))]
    for i in range(0,factorial(n)):
        for j in range(0,factorial(n)):
            List[i][j] = (-RD[ListMult(n,ListInverse(n,D[MM[RD[
               LCD[i]]][RD[LCD[j]]]]),LCD[i])], RD[ListMult(n,
               ListInverse(n,D[MM[RD[LCD[i]]][RD[LCD[j]]]]),LCD[j
               ])])
    return(List)
```

This function in Listing A.20 merely calls on the entries of the above matrix to rewrite the product $i \cdot -j$ as $-i' \cdot j'$.

Listing A.20: Pulling Entries of the Word Reversing Matrix

```
def pn2npT(posint,negint):
    return(pn2npM[posint][abs(negint)])
```

To actually rewrite a word as the product of a negative subword and a positive subword, we need to be able to scan a word and determine where the negative integers that represent inverses of our generators are. The function in Listing A.21 detects exactly where the inverse elements in a word are.

Listing A.21: Finding the Last Negative Entry

```
def findlastneg(tuple):
    k = len(tuple)-1
    while tuple[k] >= 0 and k != -1:
        k = k-1
    return(k)
```

The following function uses the previously defined functions to actually rewrite an arbitrary word as the product of a negative subword and a positive subword.

Listing A.22: The Word Reversing Process

```
def negposword(tuple):
    ncount = 0
    for i in range(len(tuple)):
```

```
    if tuple[i] < 0:
        ncount = ncount+1

lastnegspot = findlastneg(tuple)

while lastnegspot > ncount-1:
    j=0
    while not((tuple[j] >= 0) and (tuple[j+1] < 0)):
        j = j+1
    swap = j
    target = [pn2npM[tuple[swap]][abs(tuple[swap+1])][0],
       pn2npM[tuple[swap]][abs(tuple[swap+1])][1]]
    pos = [swap,swap+1]
    for x,y in zip(pos,target):
        tuple[x] = y
    print tuple
    lastnegspot = findlastneg(tuple)
return(tuple)
```

Putting a purely negative word into normal form is relies largely on the function that puts a positive word in normal form. In particular, the following function will take the inverse of a negative word by reversing the order of the word, and then take the inverse of each letter which results in a positive word. We then put that positive word in normal

form using our previously defined function, and then takes the inverse of each letter, and reverse the order of the word once again.

Listing A.23: Negative Words in Normal Form

```
def RedNegWord(neglist):
    revneglist = list(reversed(neglist))
    makepos = [abs(i) for i in revneglist]
    redpos = RedPosWord(makepos)
    backneg = [-i for i in redpos]
    return(list(reversed(backneg)))
```

The function in Listing A.24 uses all of the previously defined functions to rewrite a word that is one step away from being in normal form. In particular, the following function will take an arbitrary word and rewrite that word as the product of a negative subword and a positive subword. It will then put both the negative subword and positive subword into normal form individually and then reform the product.

Listing A.24: Puting Negative and Positive Subwords in Normal Form

```
def RNPW(ListString):
    npword = negposword(ListString)
    word2reduce = [i for i in npword if i != 0]
    nword = [i for i in word2reduce if i < 0]
    pword = [i for i in word2reduce if i>0]
    print "negative subword"
    rnword = RedNegWord(nword)
```

```
    print"positive subword"
    rpword = RedPosWord(pword)
    print"newNPword"
    redstring = rnword + rpword
    print(redstring)
    return(redstring)
```

In the above rewriting process, it is possible the that middle pair of letters consisting of a negative letter followed by positive letter, is not yet in normal form. The function Listing A.25 puts that pair in normal form.

Listing A.25: Putting the Negtive-Positive Pair in Normal Form

```
def UpdateNegPosPair(stringofints):
    spot = findlastneg(stringofints)
    c = MM[abs(stringofints[spot])][stringofints[spot+1]]
    newn = -RD[ListMult(n,ListInverse(n,D[c]),D[abs(stringofints[
       spot])])]
    newp = RD[ListMult(n,ListInverse(n,D[c]),D[stringofints[spot
       +1]])]
    newstring = [x for x in stringofints]
    newstring[spot] = newn
    newstring[spot+1] = newp
    return(newstring)
```

If the above function is actually used and the product of the negative letter and positive letter in the word is rewritten, it is possible that the negative and positive

subwords need to be rewritten as well. We account for that possibility With the following final function, which takes an arbitrary word and returns an equivalent word in normal form.

Listing A.26: Arbitrary Words in Normal Form

```
def RNormForm(stringofints):
    NPW = RNPW(stringofints)
    while NPW != UpdateNegPosPair(NPW):
        print "Update (neg,pos) Pair"
        print(UpdateNegPosPair(NPW))
        print "remove 0s"
        nNPW = [x for x in UpdateNegPosPair(NPW) if x != 0]
        print(nNPW)
        print "reduce new list"
        NPW = RNPW(nNPW)
        print(NPW)
    return(NPW)
```

For example, take the word in the 4-strand braid group represented by the product of integers in the string $[5, -16, 12, -5, 10, -18, -4, 8, 7]$. The above function yields the following:

```
input: RNormForm([5,-16,12,-5,10,-18,-4,8,7])
```

```
output: [-15, 8, 12, -5, 10, -18, -4, 8, 7]
```

```
[-15, 8, -8, 11, 10, -18, -4, 8, 7]

[-15, 0, 0, 11, 10, -18, -4, 8, 7]

[-15, 0, 0, 11, -13, 4, -4, 8, 7]

[-15, 0, 0, -11, 14, 4, -4, 8, 7]

[-15, 0, -11, 0, 14, 4, -4, 8, 7]

[-15, -11, 0, 0, 14, 4, -4, 8, 7]

[-15, -11, 0, 0, 14, 0, 0, 8, 7]

negative subword

([22, 7], -1)

positive subword

([22, 0, 7], -1)

([22, 7, 0], 0)

([22, 7, 0], 1)

([22, 7, 0], 2)

newNPword

[-7, -22, 22, 7, 0]

Update (neg,pos) Pair

[-7, 0, 0, 7, 0]

remove 0s

[-7, 7]

reduce new list

[0, 0]
```

The following set of functions were used to write each dual generator in terms of the standard generating set by applying the Hurwitz action of the 3-strand braid group to the factorization $w = abc$. We start by defining the free group on three generators a, b, c using `G.<a,b,c> = FreeGroup()`. We know that our Garside element in the dual Artin group is given by the product $\Delta = abc = re\_1^0 \cdot re_{od} \cdot re_0^1$. We apply the Hurwitz action, given by the function A.27 to this particular factorization of Δ and what results is a new factorization with updated entries that represent other dual generators.

Listing A.27: The Hurwitz Action

```
def twist(tuple,n):
    if n == 1:
    newtuple1 = (tuple[0]*tuple[1]*tuple[0]^-1,tuple[0],tuple[2])
    return(newtuple1)
    elif n == 2:
    newtuple2 = (tuple[0],tuple[1]*tuple[2]*tuple[1]^-1,tuple[1])
    return(newtuple2)
    elif n == -1:
    newtuplem1 = (tuple[1],tuple[1]^-1*tuple[0]*tuple[1],tuple
       [2])
    return(newtuplem1)
    elif n == -2:
    newtuplem2 = (tuple[0],tuple[2],tuple[2]^-1*tuple[1]*tuple
       [2])
    return(newtuplem2)
```

To apply a sequence of twists to a particular factorization of Δ, the following function can be used. The sequence of twists should be given as a list of integers where the integer i represents the action of σ_i on the factorization.

```
def twistsequence(tuple, listofns):
    newtuple = tuple
    print newtuple
    for i in range(0,len(listofns)):
        newtuple = twist(newtuple, listofns[i])
        print(newtuple, i+1)
    return(newtuple)
```

Bibliography

[Art25] Emil Artin, *Theorie der Zöpfe*, Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg **4** (1925), 47–72.

[Art47] E. Artin, *Theory of braids*, Annals of Mathematics **48** (1947), 101–126.

[BDSW14] Barbara Baumeiseter, Matthew Dyer, Christian Sump, and Patrick Wegener, *A note on the transitive hurwitz action on decompositions of parabolic coxeter elements*, Proceedings of the AMS, Series B **1(13)** (2014).

[Bes03] David Bessis, *The dual braid monoid*, Ann. Sci. Ecole Norm. **36** (2003), no. 5, 647–683.

[BH93] Brigitte Brink and Robert Howlett, *A finiteness property an an automatic structure for Coxeter groups*, Math. Ann. **296** (1993), 179–190.

[Bir98] Joan Birman, *A new approach to the word and conjugacy problems in the braid groups*, Adv. Math. **139** (1998), 322–353.

[BM15] Noel Brady and Jon McCammond, *Factoring euclidean isometries*, Intl. J of Alg and Comp **25** (2015), no. 1, 325–347.

[Boo57] W.W. Boone, *On certain simple undecidable word problems in group theory, v*, Indag. Math **19** (1957), 22–27.

[BS72] Egbert Brieskorn and Kyoji Saito, *Artin-gruppen und Coxeter-gruppen*, Invent Math **17** (1972), 245–271.

[BW02] Thomas Brady and Colum Watt, *$k(\pi, 1)$'s for Artin groups of finite type*, Proc. of the Conf on Geometric and Combinatorial Group Theory **94** (2002), 225–250.

[Dav83] Michael W. Davis, *Groups generated by reflections and aspherical manifolds not covered by euclidean space*, Annals of Mathematics **117** (1983), 293–324.

[Deh15] Patrick Dehornoy, *Foundations of Garside theory*, EMS Tracts in Mathematics, London, 2015.

[Del72] Pierre Deligne, *Les immeubles des groupes de tresses généralisés*, Invent Math **17** (1972), 273–302.

[DP99] Patrick Dehornoy and Luis Paris, *Gaussion groups and Garside groups*, Proc. of the London Math Soc **79(3)** (1999), 569–604.

[Gar65] Frank Garside, *The theory of knots and associated problems*, Ph.D. thesis, Oxford University, London, 1965.

[GP12] Eddy Godelle and Luis Paris, *Basic questions on Artin-Tits groups*, Configuration Spaces (A. Bjorner, F. Cohen, C. De Concini, C. Procesi, and M. Salvetti, eds.), CRM Series, Pisa, 2012.

[Mag32] W. Magnus, *Das identitätsproblem für gruppen mit einer definierenden relation*, Mathematische Annalen **106** (1932), 295–307.

[McC] Jon McCammond, *Noncross hypertrees*, Preprint.

[McC15] ———, *Dual Euclidean Artin groups and the failure of the lattice property*, J. of Algebra **437** (2015), 308–343.

[McC17] ———, *The mysterious geometry of Artin groups*, Winter Braids lecture Notes **4** (2017), no. 1, 1–30.

[MS17] Jon McCammond and Robert Sulway, *Artin groups of Euclidean type*, Invent. math. **210** (2017), 231–282.

[Nov55] P.S. Novikov, *On the algorithmic unsolvability of the word problem in group theory*, Trudy Mat. Inst. Steklov **44** (1955), 143pp.

[Ore31] Oystein Ore, *Linear equations in non-commutative fields*, Annals of Mathematics **32** (1931), 463–477.

[PS21] Giovanni Paolini and Mario Salvetti, *Proof of the $k(\pi, 1)$ conjecture for affine Artin groups*, Invent. math **224** (2021), 487–572.

[Sch50] Peter Scherk, *On the decomposition of orthogonalities into symmetries*, Proc. Amer. Math. Soc **1** (1950), 481–491.

[Tit13] Jacques Tits, *Collected works, Volumes I-IV*, EMS Heritage of European Mathematics, Brussels, 2013.