2004

The isoperimetric problem in finitely presented groups

James E. Kittock
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Kittock, James E., "The isoperimetric problem in finitely presented groups" (2004). *Master's Theses*. 2541.
DOI: <https://doi.org/10.31979/etd.abqg-a7ss>
https://scholarworks.sjsu.edu/etd_theses/2541

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

THE ISOPERIMETRIC PROBLEM IN FINITELY PRESENTED GROUPS

A Thesis

Presented to

The Faculty of the Department of Mathematics

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

James E. Kittock

May 2004

UMI Number: 1420423

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI Microform 1420423

Copyright 2004 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

© 2004

James E. Kittock

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF MATHEMATICS

A handwritten signature in black ink, appearing to read "Timothy Hsu", written over a horizontal line.

Dr. Timothy Hsu

A handwritten signature in black ink, appearing to read "Roger Alperin", written over a horizontal line.

Dr. Roger Alperin

A handwritten signature in black ink, appearing to read "Eloise Hamann", written over a horizontal line.

Dr. Eloise Hamann

APPROVED FOR THE UNIVERSITY

A handwritten signature in black ink, appearing to read "Pamela C. Stach", written over a horizontal line.

ABSTRACT

THE ISOPERIMETRIC PROBLEM IN FINITELY PRESENTED GROUPS

by James E. Kittock

In this thesis, we show that applying the idea of the “isoperimetric problem” leads to a geometric characterization of the difficulty of the Word Problem for a group. We begin with an overview of necessary concepts from combinatorial and geometric group theory, culminating in the definition of Dehn diagrams and proof of their equivalence with identity-equivalent words in a presentation of a group. Using Dehn diagrams, we define the Dehn function of a group, explain what it means for a group to satisfy an isoperimetric inequality, and prove that the groups whose Word Problem is solvable are precisely those whose Dehn function is a recursive function. As an application of these ideas, we define automatic groups and prove that they obey a quadratic isoperimetric inequality. Finally, we introduce an algorithm for estimating the Dehn function of a finitely presented group and show results of computer experiments with this algorithm.

DEDICATION

To my fiancée, Leslie Walker, for her continued love and support.

ACKNOWLEDGEMENTS

I would like to acknowledge my advisor, Dr. Timothy Hsu, for pushing me to accomplish more than I had expected to; his motivation and direction made this thesis possible. I would also like to thank the other members of my committee for their detailed and probing comments on drafts of this thesis.

CONTENTS

CHAPTER

1	COMBINATORIAL GROUP THEORY AND THE WORD PROBLEM	1
1.1	Preliminaries	1
1.1.1	Internal structures of groups	2
1.1.2	Maps between groups	4
1.2	Free Groups	5
1.3	Presentations of Groups	8
1.3.1	Properties of R^*	11
1.3.2	Finitely Presented Groups	15
1.4	The Word Problem	16
2	INTRODUCTION TO GEOMETRIC GROUP THEORY	19
2.1	1-Complexes and 2-Complexes	19
2.2	The Cayley Graph and the Cayley Complex	23
2.2.1	The Cayley Graph	23
2.2.2	Paths in the Cayley Graph	24
2.2.3	The Cayley Complex	27
2.3	Dehn Diagrams	29
2.3.1	Dehn Diagrams and Identity-Equivalent Words	33
2.3.2	Reduced Dehn Diagrams	40

3	THE ISOPERIMETRIC PROBLEM	45
3.1	The Dehn Function of a Group	45
3.2	Examples of Dehn Functions	47
3.3	Dehn Diagrams and the Word Problem	51
4	AUTOMATIC GROUPS	59
4.1	Finite State Automata and Regular Languages	59
4.1.1	Formal Construction of FSAs	62
4.1.2	Regular Languages and FSAs	65
4.2	Automatic Groups	67
4.3	The Quadratic Isoperimetric Inequality for Automatic Groups	71
5	ESTIMATING THE DEHN FUNCTION	79
5.1	Aspherical Presentations	79
5.2	An Estimation Algorithm for Aspherical Presentations	82
5.3	Computer Experiments with the Algorithm	88
5.3.1	Groups with quadratic Dehn functions	89
5.3.2	Groups with Dehn functions of other polynomial orders	94
5.3.3	Groups with non-polynomial Dehn functions	97
5.3.4	Spherical presentations	99
5.4	Conclusions and Future Work	100
	BIBLIOGRAPHY	102

FIGURES

Figure

1.1	Symmetries of the equilateral triangle	9
2.1	Examples of 0-, 1-, and 2-complexes	22
2.2	An edge in a Cayley graph	23
2.3	Examples of Cayley graphs	25
2.4	Examples of Cayley complexes	28
2.5	Example of a Dehn diagram	30
2.6	Adding a face to a Dehn diagram	32
2.7	Constructing the provisional diagram in Theorem 2.3.4	36
2.8	Reducing the boundary in Theorem 2.3.4	38
2.9	Two Dehn diagrams for the same word	41
2.10	Illustration for Lemma 2.3.10	43
3.1	Dehn diagrams in the Baumslag-Solitar group	50
3.2	Illustration for Lemma 3.3.1	53
4.1	The vending machine FSA	60
4.2	Examples of acceptor FSAs	61
4.3	An FSA that accepts words in F_1	65
4.4	Automatic structure for $\mathbb{Z} \times \mathbb{Z}$	70

4.5	Illustration for Lemma 4.3.3	73
4.6	Illustration for Theorem 4.3.6	75
5.1	Algorithm for estimating the Dehn function of an aspherical presentation.	85
5.2	Illustration of the estimation algorithm (1 of 2)	86
5.3	Illustration of the estimation algorithm (2 of 2)	87
5.4	Estimated Dehn function of $\mathbb{Z} \times \mathbb{Z}$	91
5.5	Estimated Dehn function of the Thompson group F	92
5.6	Estimated Dehn functions of genus-2 surface and Γ_3 groups	96
5.7	Estimates of non-polynomial Dehn functions	98

CHAPTER 1

COMBINATORIAL GROUP THEORY AND THE WORD PROBLEM

In this chapter, we introduce ideas from *combinatorial group theory*, which is the study of *presentations* of groups. Presentations are a compact way of representing groups without prior knowledge of their detailed internal structure. Along the way, we will review the basics of group theory and define *free groups*. We will close by describing one of the historically important problems in combinatorial group theory, the *Word Problem* for groups.

1.1 Preliminaries

Although we assume prior knowledge on the part of the reader equivalent to an undergraduate course in group theory, we begin by recalling the basic definitions of group theory and stating some key theorems without proof. The informed reader may wish to skip to Section 1.3.1, where some slightly non-standard material is introduced.

Definition 1.1.1 (group). A *group* $(G, *)$ consists of a set G and a binary relation $* : G \times G \rightarrow G$, where

- (1) $*$ is associative, so that $a * (b * c) = (a * b) * c$ for all $a, b, c \in G$,
- (2) G is closed under $*$, so that for all $a, b \in G$, $a * b \in G$,

(3) G has an identity element, denoted 1 , such that for all $a \in G$, $1 * a = a * 1 = a$,

and

(4) every element of G has an inverse, so that for all $a \in G$ there exists some

$$a^{-1} \in G \text{ such that } a * a^{-1} = a^{-1} * a = 1.$$

Note that conditions (1) and (2) alone give the definition of a *semigroup*; conditions (1) through (3) give the definition of a *monoid*.

Except where additional clarity is required, we will refer to the group $(G, *)$ as simply G . We will generally refer to the binary relation as “multiplication” and write ab in place of $a * b$. Associativity allows us to write repeated applications of the operation unambiguously without parentheses; for example, $(aa)a = a(aa)$, so we can simply write aaa . We can abbreviate such expressions using exponential notation; for example, we can write a^3 in place of aaa or a^{-2} in place of $a^{-1}a^{-1}$. We define $a^0 = 1$, and we can then add exponents as in elementary algebra; for example, $a^{-1}a^{-1}aa = a^{-2}a^2 = a^0 = 1$.

1.1.1 Internal structures of groups

Inside a group G there exist structures which also have the properties of a group, known as *subgroups of G* . A subgroup of G divides G into equal-sized sets called *cosets*. One important class of subgroups are the *normal subgroups*, which are closed under an operation called *conjugation*. Normal subgroups and cosets also allow us to define a sort of group division, which, in turn, allows us to construct *quotient groups*. The following definitions make these notions precise.

Definition 1.1.2 (subgroup). Given a group G , a nonempty subset $H \subseteq G$ is a *subgroup* if it satisfies the conditions for being a group; in particular, it must contain the identity and be closed under multiplication and inverses (the associativity of the

operation is inherited from the containing group). We will use the notation $H \leq G$ to denote this relationship.

If $H \leq G$ and $H \neq G$, then H is a *proper* subgroup of G ; we will write $H \subsetneq G$ when we need to denote that H is a proper subgroup. If $H \leq G$ and $H \neq \{1\}$ (the trivial group), then H is a *non-trivial* subgroup of G .

Definition 1.1.3 (cosets). Given a group G , a subgroup $H \leq G$, and an element $g \in G$, we define $gH = \{gh : h \in H\}$ to be the *left coset of H with respect to g* . The family $\{gH : g \in G\}$ is the family of all left cosets of H . The family of right cosets, $\{Hg : g \in G\}$ is defined similarly, and it can be shown that the number of right and left cosets of H is the same. This number is referred to as the *index* of H in G , denoted $[G : H]$. By a simple counting argument, it can be shown that $|G| = [G : H]|H|$, where $|G|$ and $|H|$ denote the number of elements in the group and subgroup, respectively.

Definition 1.1.4 (conjugate). Let G be a group and let $g, h \in G$. We refer to the element of G given by $g^{-1}hg$ as the *conjugate* of h by g and refer to this operation on h as *conjugation* by g .

Definition 1.1.5 (normal subgroup). Given a group G and a subgroup $N \leq G$, we say N is a *normal* subgroup of G if, for all choices of $g \in G$ and $n \in N$, we have $g^{-1}ng \in N$. In other words, a subgroup is normal if it is closed under conjugation by the elements of the group. This is equivalent to requiring that the left and right cosets of N coincide; that is, $gN = Ng$ for all $g \in G$. We will use the notation $N \trianglelefteq G$ for this relationship.

Definition 1.1.6 (quotient group). Let $N \trianglelefteq G$, and let $G/N = \{gN : g \in G\}$ be the family of all cosets of N (since N is normal in G , we don't need to specify left or

right cosets). This set is a group with the operation defined by $(g_1N)(g_2N) = (g_1g_2)N$. Observing that $1N = N$ plays the role of the identity, it is straightforward to verify the group properties. G/N is the *quotient group* (or *factor group*) of G with respect to N .

1.1.2 Maps between groups

Maps between groups play an important role in group theory; of particular interest are maps which preserve the relationships between elements of the groups. Such maps are called *group homomorphisms*, and they have a strong connection to normal subgroups. A group homomorphism which is bijective is a *group isomorphism*; isomorphisms provide a basis for considering two groups to be structurally equivalent.

Definition 1.1.7 (group homomorphism). Given groups $(G, *)$ and $(H, \otimes)$, a map $\varphi : G \rightarrow H$ is a *group homomorphism* from G into H if $\varphi(g_1 * g_2) = \varphi(g_1) \otimes \varphi(g_2)$ for all $g_1, g_2 \in G$. It can be shown that $\varphi(G)$, the image of G under φ , is a subgroup of H . If $\varphi(G) = H$, that is, if φ is onto, we say H is a *homomorphic image* of G .

Definition 1.1.8 (kernel). The *kernel* of a group homomorphism is the pre-image of the identity; given $\varphi : G \rightarrow H$, $\ker \varphi = \{g \in G : \varphi(g) = 1\}$. It can be shown that $\ker \varphi \trianglelefteq G$ and $G/\ker \varphi \cong \varphi(G)$; thus, if H is a homomorphic image of G , then $G/\ker \varphi \cong H$. Furthermore, for any normal subgroup $N \trianglelefteq G$, there is a homomorphism $\theta : g \mapsto gN$ such that $\ker \theta = N$.

Definition 1.1.9 (group isomorphism). If $\varphi : G \rightarrow G'$ is a group homomorphism that is one-to-one and onto (bijective), then φ is a *group isomorphism* between G and G' , and we say G and G' are *isomorphic* groups, written $G \cong G'$. We will generally treat isomorphic groups as equivalent. An isomorphism $\varphi : G \rightarrow G$ is called an *automorphism* of G .

1.2 Free Groups

In many groups, there are relationships between the identity and other elements beyond the obligatory element-inverse relationship. For example, in the additive group $\mathbb{Z}_4$ (the group of integers with the operation of addition modulo 4), we have $1 + 1 + 1 + 1 = 0$. A *free group* is a group that has a distinguished set of generators that satisfy no such relationships. Although free groups are conceptually simple, the construction of such groups requires some rather tedious machinery, which we summarize in this section.

Definition 1.2.1 (alphabet, word). Let A be an arbitrary set, which we will call the *alphabet*, and the elements of which we will correspondingly call *letters*. A finite sequence $w = a_1 a_2 \cdots a_n$ of elements over A will be called a *word*. The *length* of w is the non-negative integer n . The *empty word* is the sequence of length 0, which we will denote by ε .

Definition 1.2.2 (concatenation, free monoid). We define an operation, *concatenation*, on the words of alphabet A , as follows. If $w = a_1 \cdots a_m$ and $v = b_1 \cdots b_n$ are words over A of length m and n respectively, then $wv = a_1 \cdots a_m b_1 \cdots b_n$, and the length of wv is $m + n$. Observe that concatenation is associative, and that $w\varepsilon = \varepsilon w = w$, so ε is the identity with respect to the operation of concatenation. Thus, the closure of A under concatenation is a monoid, the *free monoid* on A , which we will denote as A^* .

Example 1.2.3 (free monoid on two elements). Let $A = \{a, b\}$, then A^* consists of elements $\{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$. If $w = ab$ and $v = ba$, then $wv = abba$.

A free monoid can be extended to a free group by adjoining a set of inverse elements and modifying the operation of concatenation in a straightforward way.

Definition 1.2.4 (monoid of words). Given an alphabet A , we call a disjoint set A^{-1} an *inverse alphabet* if there is a bijection $^{-1} : A \rightarrow A^{-1}$. Then for each element $a \in A$, there exists a unique element $a^{-1} \in A^{-1}$, and for each element $a^{-1} \in A^{-1}$, there exists a unique element $a \in A$. We define $(a^{-1})^{-1} = a$ for each $a^{-1} \in A^{-1}$. Given an alphabet A , we will use W_A to denote the free monoid on $A \cup A^{-1}$, which we will refer to as the *monoid of words over A* .

The elements of A^{-1} are intended to be the inverses of the elements of A , so that $aa^{-1} = \varepsilon$. This leads to the following definition and key theorem regarding *reduced words*.

Definition 1.2.5 (reduced word). A word $w = w_1 \cdots w_n \in W_A$ is *reduced* if there is no i such that $w_{i+1} = w_i^{-1}$.

Theorem 1.2.6 (unique reduction). *For each word $w \in W_A$, there is a unique reduced word $\hat{w} \in W_A$ corresponding to w .*

Proof. The proof of this theorem requires working through some technical details, which can be found in, e.g., [Gri99, pp. 63-65]. $\square$

Example 1.2.7 (reduced words). As before, let $A = \{a, b\}$. Examples of reduced words in W_A include ε , a , ab , and $aba^{-1}b^{-1}$. Examples of words which are not reduced include aa^{-1} , $bb^{-1}b$, and $aabb^{-1}a^{-1}$; they reduce to ε , b , and a respectively. Note that there are two equivalent ways to reduce $bb^{-1}ba$; also note that reducing $abb^{-1}a^{-1}$ requires multiple steps, because eliminating all adjacent element-inverse pairs in the original word leaves a word that is still not reduced. It is situations such as these that complicate the proof of the unique reduction theorem.

We are now ready to define a free group; we state the definition as a theorem.

Theorem 1.2.8 (free group). *Let A be an alphabet, and let F_A be the set of all reduced words in the free monoid W_A . Define the operation $*$ to be concatenation followed by reduction. Then $(F_A, *)$ is a group, called the free group on A . The identity element in F_A is ε , and the inverse of an element $a_1a_2 \cdots a_{n-1}a_n \in F_A$ is $a_n^{-1}a_{n-1}^{-1} \cdots a_2^{-1}a_1^{-1}$.*

Proof. See, e.g., [MKS66, pp. 33–36] or [Gri99, pp. 65–66]. $\square$

Remark 1.2.9 (the use of “=” with unreduced words). Because our focus will be on words in F_A , rather than in W_A , we will write $w_1 = w_2$ whenever the words $w_1, w_2 \in W_A$ reduce to the same word in F_A . Thus, we might write $aba = abb^{-1}ba$, with the understanding that the latter word must be reduced in order for the equivalence to hold in F_A . In some cases, we may explicitly say “with reduction, as needed” to remind the reader of this convention, but even in the absence of such a qualification, the equivalence is implicitly assumed to hold in F_A after reduction. Should we ever require equivalence in W_A , we will explicitly indicate this in the surrounding text.

We will use the notation F_n for the free group on n elements; this is made possible by the following theorem.

Theorem 1.2.10 (isomorphic free groups). *Let F_A and F_B be free groups. Then $F_A \cong F_B$ if and only if A and B have the same cardinality (finite or infinite).*

Proof. See, e.g., [LS77, p. 1]. $\square$

We end this section with a theorem regarding free groups that states that every group is the homomorphic image of a free group. Free groups are sometimes defined as the (unique) groups having this property.

Theorem 1.2.11 (universal property of free groups). *Let A be an alphabet, F_A be the free group on A , and $\iota : A \rightarrow F_A$ be the inclusion map from A into F_A . Let G be*

an arbitrary group, and let $\varphi : A \rightarrow G$ be any map such that the image $\varphi(A) \subseteq G$ is a generating set of G . Then there is a unique surjective homomorphism $\psi : F_A \rightarrow G$ such that $\psi \circ \iota = \varphi$.

Proof. See, e.g., [LS77, p. 1], [Gri99, p. 66–67]. □

1.3 Presentations of Groups

A *presentation* of a group consists of a set of elements that generate the group and a description of the relationships between those elements. Presentations arise naturally in various branches of mathematics; the following example is from geometry.

Example 1.3.1 (symmetries of an equilateral triangle). An equilateral triangle has two basic types of symmetries: rotations and reflections. Let ρ represent a 120-degree rotation around the center, and let μ represent a reflection through a vertical axis; then, as shown on the left in Figure 1.1, all possible rotations are given by ρ and ρ^2 , and all possible reflections are given by μ , $\rho\mu\rho^2$, and $\rho^2\mu\rho$ (where we compose symmetries from left to right). Thus, μ and ρ together generate all of the symmetries of the equilateral triangle.

There are three fundamental relationships between the two basic symmetries and the identity symmetry, which we denote as 1 (this is the symmetry which “does nothing”, leaving the triangle unchanged). If we reflect twice about the same axis, or if we do three clockwise rotations, we end up with the triangle in the starting orientation; that is, $\mu^2 = 1$ and $\rho^3 = 1$. Further, if we reflect, rotate clockwise, then reflect again, the net effect is to rotate the triangle counter-clockwise; that is, $\mu\rho\mu = \rho^{-1}$, or, equivalently, $\mu\rho\mu\rho = 1$. These identity-equivalent sequences of symmetries, which we refer to as *relators*, are shown on the right in Figure 1.1.

The generating set $\{\mu, \rho\}$ and the set of relators $\{\rho^2, \mu^3, \mu\rho\mu\rho\}$ define the symmetry group of the equilateral triangle, up to isomorphism. We can summarize this information in the following compact way: $\langle \mu, \rho \mid \rho^3, \mu^2, \mu\rho\mu\rho \rangle$; this is a *presentation* of the symmetry group of the equilateral triangle.

We now use free groups to formalize the notion of a presentation.

Definition 1.3.2 (presentation). Let A be an alphabet; we will refer to it as a set of *generators* for the group being presented. Let $R \subseteq F_A$ be a set of reduced words over A ; we will refer to it as a set of *relators*. Let R^* denote the normal closure of R ; that is, R^* is the smallest normal subgroup of F_A containing R . Then we define $\langle A \mid R \rangle = F_A/R^*$. If $G \cong \langle A \mid R \rangle$, we say $\langle A \mid R \rangle$ is a *presentation* of G .

The elements of R define the elements of F_A which are equivalent to the identity in the group being presented. Lemma 1.3.5 will tell us how to construct R^* .

Remark 1.3.3 (conventions regarding presentations). When $\langle A \mid R \rangle$ is a presentation of group G , we will usually write $G = \langle A \mid R \rangle$ unless context requires us to be more precise. When writing relators, we will typically abbreviate the commutator $aba^{-1}b^{-1}$ as $[a, b]$.

We now define a standard map from words in F_A to elements in G .

Definition 1.3.4 (representation map). By the Homomorphism Theorem for groups, the fact that $G = \langle A \mid R \rangle = F_A/R^*$ implies that there is an onto homomorphism $\pi : F_A \rightarrow G$ such that $\ker \pi = R^*$. We call this homomorphism the *representation map* for $\langle A \mid R \rangle$. We will have occasion to extend π to W_A ; the understanding is that $\pi(w)$, for $w \in W_A$, is $\pi(\hat{w})$ where $\hat{w} \in F_A$ is the unique reduced word corresponding to w . Given a word $w \in F_A$ (or W_A), we will abbreviate $\pi(w)$ as $\bar{w}$, and refer to $\bar{w}$ as the element of G *represented by* w .

Note that π is generally not one-to-one: a group element can be represented by multiple reduced words, as shown in Example 1.3.11, below.

1.3.1 Properties of R^*

This section contains some slightly non-standard material. Recall that R^* is the normal closure of the set of relators, R . The following lemma gives us a convenient way to describe the elements of R^* .

Lemma 1.3.5. *Let w be a word in F_A . Then $w \in R^*$ if and only if w can be written as a product of conjugates of relators. That is, $\bar{w} = 1$ if and only if*

$$w = \prod_{i=1}^n c_i^{-1} r_i c_i, \text{ for some } c_i \in F_A, r_i \in R \cup R^{-1}, \quad (1.1)$$

with reduction, as needed.

Proof. Recall that R^* is defined to be the smallest normal subgroup containing R . We proceed by constructing R^* from R ; at each step, we show that the words in R^* can be written in the form (1.1).

- (1) R^* must contain each $r \in R$. A single relator r can be written as a product of the form (1.1) with $n = 1$, $r_1 = r$, $c_1 = \varepsilon$.
- (2) To be a subgroup of F_A , R^* must contain the identity, which is the empty word, ε . The empty word can be written as a trivial product of the form (1.1) with $n = 0$.
- (3) To be a subgroup of F_A , R^* must be closed under inverses, so for any $r \in R$, r^{-1} must be in R^* . The inverse of a relator r can be written in the form (1.1) with $n = 1$, $r_1 = r^{-1}$, $c_1 = \varepsilon$.

- (4) To be a normal subgroup of F_A , R^* must be closed under conjugation by elements in F_A ; that is, for any $c \in F_A$ and $r \in R \cup R^{-1}$, $c^{-1}rc$ must be in R . This is a product of the form (1.1) with $n = 1$.
- (5) To be a subgroup of F_A , R^* must be closed under multiplication, so for any $c_1, \dots, c_n \in F_A$ and $r_1, \dots, r_n \in R \cup R^{-1}$, we must have $c_1^{-1}r_1c_1 \cdots c_n^{-1}r_nc_n$ in R^* . This is clearly a product of the form (1.1).

Note that R^* is closed under inverses: if $c_1^{-1}r_1c_1 \cdots c_n^{-1}r_nc_n \in R^*$, then

$$\begin{aligned} (c_1^{-1}r_1c_1 \cdots c_n^{-1}r_nc_n)^{-1} &= (c_n^{-1}r_nc_n)^{-1} \cdots (c_1^{-1}r_1c_1)^{-1} \\ &= c_n^{-1}r_n^{-1}c_n \cdots c_1^{-1}r_1^{-1}c_1 \end{aligned} \quad (1.2)$$

will be in R^* , by construction.

Similarly, R^* is closed under conjugation: if $c_1^{-1}r_1c_1 \cdots c_n^{-1}r_nc_n \in R^*$, then for any $d \in F_A$,

$$\begin{aligned} d^{-1} (c_1^{-1}r_1c_1 \cdots c_n^{-1}r_nc_n) d &= (d^{-1}c_1^{-1})r_1(c_1d) \cdots (d^{-1}c_n^{-1})r_n(c_nd) \\ &= (c_1d)^{-1}r_1(c_1d) \cdots (c_nd)^{-1}r_n(c_nd) \end{aligned} \quad (1.3)$$

will be in R^* , by construction.

R^* now satisfies all of the requirements for a normal subgroup, and we have added the fewest possible elements, so R^* is the smallest subgroup containing R . Furthermore, all of the elements of R^* can be written in the form (1.1). $\square$

It should be noted that this decomposition into products of conjugates of relators is not unique. For example, in the presentation $\langle a, b \mid [a, b] \rangle$, the word $w = a^2ba^{-1}ba^{-1}b^{-2}$ can be written both as

$$w = [a, b] \cdot (b^{-1})^{-1}[a, b](b^{-1}) \cdot (b^2ab^{-2})^{-1}[a, b](b^2ab^{-2}), \quad (1.4)$$

and as

$$w = (a^{-1})^{-1}[a, b](a^{-1}) \cdot [a, b] \cdot (b^{-1})^{-1}[a, b](b^{-1}). \quad (1.5)$$

(Note that this is the same word that will be used in Example 2.3.1 in Chapter 2.)

In the above example, the two decompositions have the same number of terms, but it is also possible for two decompositions of a given word to have a different number of terms. A trivial way that this can happen is for adjacent terms in a product to be inverses. Consider a product of the form (1.1) for which there is some i such that $c_i = c_{i-1}$ and $r_i = r_{i-1}^{-1}$, so that $(c_{i-1}^{-1}r_{i-1}c_{i-1})(c_i^{-1}r_ic_i) = \varepsilon$ after reduction; the product formed by removing terms i and $i - 1$ will represent the same word, but will have two fewer terms. It can also be shown that there are presentations in which a given word can have decompositions which differ non-trivially.

In some situations, it will be useful for us to consider the following class of “well-behaved” products of conjugates, in which the conjugator and the relator in each term do not cancel with one another, even partially. We will use the following non-standard definition to prove results in Chapter 2.

Definition 1.3.6 (non-cancelling product of conjugates of relators). A product of the form (1.1) is called *non-cancelling* if, for all i , each term $c_i^{-1}r_ic_i$ is reduced internally, without cancellation.

Note that the above definition says nothing about partial cancellation between terms; such cancellation is generally necessary in order to write an arbitrary word as a product of conjugates of relators.

In general, it may not be possible to transform a product that fails to meet the definition of “non-cancelling” into a non-cancelling product. As an example, consider the presentation $\langle A \mid R \rangle = \langle a, b \mid aba \rangle$ and the word baa . This word can be decomposed as $a^{-1}(aba)a$, but some cancellation is required; in fact, there is no way to write baa

as a product of conjugates over this presentation without partial cancellation between some conjugator and the relator aba . Note that this could be remedied simply by adding baa to the set of relators, R . This addition would not change R^* ; baa would have been in R^* to begin with, because it is a conjugate of aba .

The following definition and its accompanying lemma formalize this idea, specifying a particular type of presentation for which every product of the form (1.1) can be transformed into a non-cancelling product. The idea for the definition is from [LS77, p. 239].

Definition 1.3.7 (symmetrized presentation). A presentation $\langle A \mid R \rangle$ is symmetrized if, for each relator r in R , all cyclic permutations of r and r^{-1} are also included in R .

Observe that the cyclic permutations of a relator r are simply cyclically reduced conjugates of r ; all such conjugates (along with their inverses) will be in R^* independent of the symmetrization. Thus, the symmetrization of R does not change R^* ; furthermore, each relator has only finitely many cyclic permutations, so if R is finite, it remains finite after being symmetrized.

As an example, if we start with $R = \{abc\}$, then, after symmetrization, $R = \{abc, bca, cab, (abc)^{-1}, (bca)^{-1}, (cab)^{-1}\}$. Symmetrized presentations have the following useful property.

Lemma 1.3.8. *Let $\langle A \mid R \rangle$ be a symmetrized presentation, and let $w \in F_A$ be such that $\bar{w} = 1$. Then w can be written as a non-cancelling product of conjugates of relators.*

Proof. By Lemma 1.3.5, w can be written as a product of conjugates of relators as shown in (1.1). If the product satisfies the definition of “non-cancelling”, we are done; if not, there are one or more terms such that some cancellation occurs.

Let $c^{-1}rc$ be such a term, so that $r = a_1a_2\cdots a_m$ and $c = b_1b_2\cdots b_n$, where $a_i, b_i \in A \cup A^{-1}$. If r cancels with c , then $a_m = b_1^{-1}$; if c^{-1} cancels with r , then $b_1 = a_1$. Let $c = b_1c'$, so that $c^{-1}rc = (b_1c')^{-1}r(b_1c') = c'^{-1}(b_1^{-1}rb_1)c'$. Let $r' = b_1^{-1}rb_1$, then r' is a cyclic permutation of r , and, since R is symmetrized, $r' \in R$. Since $c^{-1}rc = c'^{-1}r'c'$ (after cancellation) we can replace $c^{-1}rc$ with $c'^{-1}r'c'$ in the product of conjugates. If cancellation occurs in $c'^{-1}r'c'$, we can repeat this process until we have eliminated all of the cancellation in $c^{-1}rc$; the process is guaranteed to terminate because c is finite.

By applying this process to each term that fails to meet the condition in the definition of “non-cancelling”, we can transform any product of conjugates over a symmetrized presentation into a non-cancelling product. $\square$

1.3.2 Finitely Presented Groups

In this thesis, we will restrict our attention to *finitely presented groups*, which are defined as follows.

Definition 1.3.9 (finitely presented group). Let $G = \langle A | R \rangle$. We say G is *finitely generated* when A is a finite set, and G is *finitely related* when R is a finite set. G is *finitely presented* if it is both finitely generated and finitely related.

As illustrated by the following examples, both finite and infinite groups can have finite presentations.

Example 1.3.10. Observe that if $R = \emptyset$ then $R^* = \{\varepsilon\}$. Thus $\langle A | \emptyset \rangle = F_A / \{\varepsilon\} = F_A$. Some authors take $\langle A | \emptyset \rangle$ to be the definition of F_A ; see, e.g., [MKS66, p. 33]. In the special case where $|A| = 1$, $\langle A | \emptyset \rangle \cong F_1 \cong \mathbb{Z}$.

Example 1.3.11. Let $G = \langle a, b | [a, b] \rangle$. The relator $[a, b]$ tells us that $\overline{[a, b]} = 1$, so $\overline{aba^{-1}b^{-1}} = 1$, so $\overline{ab} = \overline{ba}$, so $\overline{ab} = \overline{ba}$ (because the representation map is a

homomorphism). Thus, the elements represented by a and b commute in G , and, by an easy induction, the same is true of all powers of a and b . We have no other relators, so $G \cong \mathbb{Z} \times \mathbb{Z}$. Note that $1 \in G$ has many representatives in F_A , including ε , $[a, b]$, $abab^{-1}a^{-2}$, etc. Thus, the representation map $\pi : F_A \rightarrow G$ is not one-to-one.

Example 1.3.12. Add the relators a^2, b^3 to the previous example, so that we have the presentation $G = \langle a, b \mid [a, b], a^2, b^3 \rangle$. We now know that $\overline{a^2} = 1$ and $\overline{b^3} = 1$, so there are only two distinct powers of a and three distinct powers of b ; as before, the elements represented by a and b and all of their powers commute in G . Thus, $G \cong \mathbb{Z}_2 \times \mathbb{Z}_3$. Note that $\mathbb{Z}_2 \times \mathbb{Z}_3 \cong \mathbb{Z}_6$, suggesting the alternative presentation $G = \langle a \mid a^6 \rangle$. This illustrates the point that presentations are not unique. It is often possible to present a given group in multiple ways, both finitely and infinitely; a presentation is sometimes described as “natural” when it reflects the structure of the group in some particularly good way.

1.4 The Word Problem

In Example 1.3.1, we saw that a presentation for a group can arise from detailed knowledge about a mathematical structure. However, presentations are often constructed without such prior knowledge. It turns out that extracting information about a group from an arbitrary presentation is very difficult in the general case. Three basic problems regarding the discovery of structural information from presentations were given by Max Dehn in 1911:

- *The Word Problem:* Is an arbitrary word equivalent to the identity? That is, given $\langle A \mid R \rangle$ and $w \in F_A$, is $\overline{w} = 1$?
- *The Conjugacy Problem:* Are two arbitrary words conjugates? That is, given $\langle A \mid R \rangle$ and $w_1, w_2 \in F_A$, is $\overline{w_1} = g\overline{w_2}g^{-1}$ for some $g \in \langle A \mid R \rangle$?

- *The Isomorphism Problem:* Do two arbitrary presentations present isomorphic groups? That is, given $\langle A \mid R \rangle$ and $\langle A' \mid R' \rangle$, is it the case that $\langle A \mid R \rangle \cong \langle A' \mid R' \rangle$?

If the Conjugacy Problem is solvable, then the Word Problem must be solvable; in particular, the Word Problem for a word w is equivalent to the Conjugacy Problem $\overline{w} \stackrel{?}{=} g\overline{\varepsilon}g^{-1}$. Similarly, it can be shown that if the Isomorphism Problem is solvable, then the Word Problem and Conjugacy Problem must be solvable. Thus, these problems are increasingly difficult.

Although the Word Problem is evidently solvable for some presentations (for example, $F_A = \langle A \mid \emptyset \rangle$, in which only ε represents the identity), it was independently proved by Novikov and Boone in 1955 that the Word Problem is generally *unsolvable* for finitely presented groups [LS77]. That is, there exist finite presentations for which there is no finite procedure for determining if an arbitrary word is equivalent to the identity. An immediate implication is that the Conjugacy and Isomorphism Problems are also unsolvable in the general case.

There do exist broad classes of presentations for which the Word Problem has been shown to be solvable, such as one-relator presentations and commutative groups. The Conjugacy Problem has also been shown to be solvable for some of these classes of groups, but the Isomorphism Problem has been shown to be unsolvable even for seemingly simple cases, such as determining if a finite presentation is equivalent to the trivial group. Further details on these decision problems and their history can be found in [LS77, pp. 88–89] and [MKS66, pp. 24–26].

In this thesis, we will apply geometric ideas to derive the *Dehn function* of a presentation of a group, and we will show that the Word Problem for a presentation is solvable when the Dehn function is a so-called *recursive function*. As an

application of these ideas, we will define the family of *automatic groups* and prove that their Dehn functions satisfy a quadratic inequality. Finally, we will introduce an algorithm for estimating the Dehn function of a finite presentation and report the results of computer experiments with this algorithm. In the next chapter, we proceed by introducing some necessary concepts from geometric group theory.

CHAPTER 2

INTRODUCTION TO GEOMETRIC GROUP THEORY

In this chapter, we provide an overview of some aspects of *geometric group theory*, which is the application of geometric techniques to the study of groups. We begin by defining constructions known as *1-complexes* and *2-complexes*. Complexes are then used to define the *Cayley graph* and *Cayley complex* of a group, and we give several examples of each. Finally, we define and prove results concerning *Dehn diagrams* over a presentation of a group. In particular, we show that Dehn diagrams represent the identity-equivalent elements in the group.

2.1 1-Complexes and 2-Complexes

A *1-complex* is essentially a directed graph in which the edges come in pairs; if we then “fill in” areas “enclosed” by edges of the graph in a particular way, we get a *2-complex*. In this section, we define these constructions combinatorially, following [LS77, p. 115]. In particular, we omit details of the topological realization of these constructions; interested readers may refer to an introductory topology text such as [Kin93].

Definition 2.1.1 (1-complex). A *1-complex* is a 5-tuple $C^1 = (V, E, \alpha, \omega, {}^{-1})$, constructed as follows:

- V is a set of discrete points (a 0 -complex), corresponding to the vertices of a graph.
- E is a set of line segments, corresponding to the directed edges of a graph.
- $\alpha : E \rightarrow V$ and $\omega : E \rightarrow V$ are maps that define how the edges in E are attached to the vertices in V . For each edge $e \in E$, $\alpha(e)$ and $\omega(e)$ are the *start vertex* and *terminal vertex* of e , respectively. If $\alpha(e) = v_1$ and $\omega(e) = v_2$, then we write $e = (v_1, v_2)$ and say edge e *runs from* v_1 *to* v_2 . Note that it is possible to have self-loops; that is, we allow edges e such that $\alpha(e) = \omega(e)$.
- $^{-1} : E \rightarrow E$ is a map that carries each edge to a distinct *inverse edge* which runs the opposite direction between the same vertices. That is, for each $e \in E$, we require there to be some $e^{-1} \in E$ such that $e^{-1} \neq e$, $\alpha(e^{-1}) = \omega(e)$, and $\omega(e^{-1}) = \alpha(e)$. Furthermore, we require $(e^{-1})^{-1} = e$ (for a self-loop, we define $e^{-1} = e$).

Definition 2.1.2 (undirected edge, orientation). Observe that $^{-1}$ partitions E into two subsets; each pair (e, e^{-1}) corresponds to an *undirected edge* in the graph. An *orientation* of a 1-complex is a choice of one element from each such pair. When drawing pictures of oriented 1-complexes, we will use an arrow to indicate the direction of the “chosen” edge and omit the implicit inverse edge.

If we start at a given vertex in the 1-complex and move from vertex to vertex along connecting edges, we trace a *path* in the 1-complex. Formally:

Definition 2.1.3 (path). A *path* in a 1-complex is a finite sequence $p = e_1 e_2 \cdots e_n$ of edges such that $\alpha(e_i) = \omega(e_{i-1})$ for $i = 2, \dots, n$. We say p has *length* n , denoted by $|p| = n$. We will use the notation $p(i)$ to refer to a particular edge on a path, so that

$p(i) = e_i$ in the sequence above. We extend the maps α and ω to p in the obvious way: $\alpha(p) = \alpha(p(1))$ and $\omega(p) = \omega(p(n))$; these are the *start vertex* and *terminal vertex* of p , respectively. For each $v \in V$, we assume the existence of an *empty path* 1_v such that $\alpha(1_v) = \omega(1_v) = v$ and $|1_v| = 0$.

If $p_1, p_2, \dots, p_m$ is a sequence of paths such that $\alpha(p_{i+1}) = \omega(p_i)$ (so that they connect end-to-end), then $p = p_1 p_2 \cdots p_m$ is understood to be the path that travels from $\alpha(p_1)$ to $\omega(p_m)$ along each of the p_i 's in turn. We define certain paths relative to a given path, as follows.

Definition 2.1.4 (inverse path, sub-path). Given a path p , we define the *inverse path* in the obvious way: $p^{-1} = p(n)^{-1} p(n-1)^{-1} \cdots p(1)^{-1}$. A *sub-path* of p is a sequence $q = p(j) p(j+1) \cdots p(k)$ for some $1 \leq j \leq k \leq n$. If $j \neq 1$ or $k \neq n$, then q is a *proper sub-path* of p .

We will sometimes wish to restrict our attention to particularly “nice” paths, hence the following definition.

Definition 2.1.5 (reduced path, simple path). A path is *reduced* if it contains no “spurs” or “backtracks”; that is, $e_i \neq e_{i-1}^{-1}$ for all $i = 2, \dots, n$. A path is *simple* if it contains no self crossings; that is, $\alpha(e_i) \neq \alpha(e_j)$ and $\omega(e_i) \neq \omega(e_j)$ whenever $i \neq j$.

Loops—paths that are closed—will play an important role in the construction of a 2-complex.

Definition 2.1.6 (loop). A *loop* is a path ℓ whose start vertex and terminal vertex are the same; that is, $\alpha(\ell) = \omega(\ell)$. If a loop is reduced and, furthermore, $\ell(n) \neq \ell(1)^{-1}$, we say it is *cyclically reduced*.

We can now define a 2-complex as an extension of a 1-complex.

Definition 2.1.7 (2-complex). A 2-complex is a 7-tuple $C^2 = (V, E, \alpha, \omega, {}^{-1}, F, \partial)$, constructed as follows:

- $C^1 = (V, E, \alpha, \omega, {}^{-1})$ a 1-complex, called the *1-skeleton* of C .
- F is a set of *faces* or *2-cells*, which are homeomorphic copies of the unit disc.
- ∂ is a map that defines, for each face f in F , a loop in C^1 called the *boundary of f* . Unless otherwise specified, we assume boundaries are travelled in counter-clockwise order starting from an arbitrary basepoint.
- ${}^{-1}$ is extended to map $F \rightarrow F$ in the expected way; it carries each face f to a distinct *inverse face*, whose boundary is the inverse of the boundary of f . In particular, we require that for each $f \in F$, there is $f^{-1} \in F$ such that $f^{-1} \neq f$ and $\partial(f^{-1}) = (\partial f)^{-1}$. We further require that $(f^{-1})^{-1} = f$.

By convention, faces in 2-complexes will be represented as shaded areas. Figure 2.1 provides examples of 0-, 1-, and 2-complexes.

Figure 2.1: *Examples of 0-, 1-, and 2-complexes.* Left: A 0-complex, which is simply a collection of vertices. Center: A 1-complex, formed by adding edges connecting some of the vertices in the 0-complex. Right: A 2-complex, formed by adding faces to some of the regions enclosed by edges in the 1-complex.

Definition 2.1.8 (oriented 2-complex). Observe that ${}^{-1}$ partitions F into two subsets, an *orientation* of C^2 is a choice of one element from each (f, f^{-1}) pair, along with a choice of orientation for the skeleton C^1 (see 2.1.2).

In the following section, we use complexes to define geometric structures which can be derived from the presentation of a group.

2.2 The Cayley Graph and the Cayley Complex

Before proceeding, let us recall some basic facts and notation regarding a group G with finite presentation $\langle A | R \rangle$. A is a generating set or “alphabet” for the group. W_A is the monoid of words over $A \cup A^{-1}$, and F_A is the free group of reduced words over $A \cup A^{-1}$. R is a set of relators, and R^* is the normal closure of R . $G \cong F/R^*$, and the element in G represented by a word w in W_A (or F_A) is represented by $\overline{w}$.

2.2.1 The Cayley Graph

A *Cayley graph* for a group (sometimes called a “Cayley diagram” or “Cayley digraph”) shows how elements are related by the group operation.

Figure 2.2: *An edge in a Cayley graph.* Edges in the Cayley graph for a group G with generating alphabet A are of the form (g, ga) , where $g \in G$ and $a \in A$.

Definition 2.2.1 (Cayley graph). Given a group G , the *Cayley graph* with respect to generating alphabet A is an oriented 1-complex $\mathcal{C}^1(G, A) = (V, E, \alpha, \omega, \cdot^{-1})$ along with an associated *labelling map* $\lambda : E \rightarrow A \cup A^{-1}$, constructed as follows:

- The set of vertices V is a copy of the set of group elements in G ; that is, for each element in $g \in G$, there is one element in V . In a mild abuse of notation,

we will denote the corresponding vertex as simply g ; it should be clear from context whether we are talking about the vertex in the Cayley graph or the group element itself.

- The set of edges E is a copy of $G \times A \cup A^{-1}$. For each $g \in G$ and each $a \in A$, there is an edge $(g, ga) \in E$; this edge represents element g being carried to element ga under right multiplication by a in the group G (see Figure 2.2). The set of inverse edges correspond to right multiplication by elements of A^{-1} ; in particular, $(g, ga)^{-1} = (ga, gaa^{-1}) = (ga, g)$.
- The map λ is defined by $\lambda((g, ga)) = a$. Thus λ assigns a label to each edge according to the generator associated with the edge. Clearly, $\lambda((g, ga)^{-1}) = [\lambda(g, ga)]^{-1} = a^{-1}$.

Figure 2.3 gives several examples of Cayley graphs; as mentioned in 2.1.2, inverse edges have been omitted (traversing the edges “against” the arrowheads can be thought of as traversing the inverse edges).

2.2.2 Paths in the Cayley Graph

We now consider some important properties of paths in the Cayley graph. As in any 1-complex, a path in a Cayley graph is a sequence of connecting edges, but due to the particular relationship between the vertices of each edge in the Cayley graph, a path in the Cayley graph is of the following form:

$$p = (g_0, g_1)(g_1, g_2) \cdots (g_{n-1}, g_n), \text{ where } g_i = g_{i-1}a_i \text{ for some } a_i \in A \cup A^{-1} \quad (2.1)$$

$$= (g_0, g_0a_1)(g_0a_1, g_0a_1a_2) \cdots (g_0a_1 \cdots a_{n-1}, g_0a_1 \cdots a_n), \text{ or, equivalently,} \quad (2.2)$$

$$= (g_0, g_0a_1)(g_0a_1, g_0a_1a_2) \cdots (g_0a_1 \cdots a_{n-1}, g_0a_1 \cdots a_n) \quad (2.3)$$

Figure 2.3: *Examples of Cayley graphs.* The Cayley graphs for two finite groups and two infinite groups are shown. The graph for F_2 is modeled after [EPC⁺92, p. 36]; note that the graph continues to branch indefinitely.

We abbreviate (2.3) as $(g; w)$ where $w = a_1 a_2 \cdots a_n \in F_A$, and observe that each path in the Cayley graph is of this form. Conversely, given $g \in G$ and $w \in F_A$ there is some path $p = (g; w)$. For such a path, $|p| = |w|$. We can extend the labelling map λ to paths in the Cayley graph in the obvious way; for a path $p = (g; w)$, we have $\lambda(p) = w$.

Paths in the Cayley graph can also be thought of as solutions to equations in the group. Given some $g, h \in G$, a path p for which $\alpha(p) = g$ and $\omega(p) = h$ represents a solution w to the equation $gw = h$; in particular, $w = \lambda(p)$. Since every such equation in a group has a solution and A generates G , this implies that the Cayley graph is connected. Conversely, given some $g \in G$ and $w \in F_A$, if $h = gw$, then there is a path $p = (g; w)$ such that $\omega(p) = h$.

This equivalence can be restricted to reduced paths and reduced words in F_A ; this leads us to the following observation.

Lemma 2.2.2. *Let G be a group with generating alphabet A . Then a reduced path p in $\mathcal{C}^1(G, A)$ is a loop if and only if $\overline{\lambda(p)} = 1$ in G .*

Proof. Let p be a reduced loop in the Cayley graph of G with respect to A . Since p is a loop, $\alpha(p) = \omega(p) = g$, for some $g \in G$. Thus, $g\overline{\lambda(p)} = g$, that is, $\overline{\lambda(p)} = 1$.

Conversely, let p be a reduced path in $\mathcal{C}^1(G, A)$ such that $\overline{\lambda(p)} = 1$. Then $\alpha(p) = g$ for some $g \in G$, and $\omega(p) = g\overline{\lambda(p)} = g$, so $\alpha(p) = \omega(p)$ and thus p is a loop. $\square$

The following definition uses paths to provide a meaningful concept of distance between vertices in the Cayley graph.

Definition 2.2.3 (word metric). Let G be a group with generating alphabet A . Given $g, h \in G$, the distance between g and h in $\mathcal{C}^1(G, A)$ is defined to be the length

of the shortest path connecting them. If the length of this path is n , we say that the distance between g and h is n in the *word metric*.

We now extend the Cayley graph to a 2-complex.

2.2.3 The Cayley Complex

Just as we used a 1-complex as the skeleton of a 2-complex in the general case, we can use a Cayley graph as the skeleton of a particular 2-complex, called the *Cayley complex*. Intuitively, the faces of a Cayley complex are formed by “pasting in” copies of the relators from a presentation of the group under consideration.

Definition 2.2.4 (Cayley complex). Given a group G with finite presentation $\langle A \mid R \rangle$, the corresponding *Cayley complex* is a 2-complex $\mathcal{C}^2(A, R) = (V, E, \alpha, \omega, {}^{-1}, F, \partial)$, constructed as follows:

- $\mathcal{C}^1(G, A) = (V, E, \alpha, \omega, {}^{-1})$ is the Cayley graph of G with respect to generating alphabet A ; as before, λ is labelling map associated with $\mathcal{C}^1(G, A)$.
- F is a copy of $G \times R \cup R^{-1}$; in particular, for each $g \in G$ and each $r \in R$, there is a face $f(g, r)$ such that $\partial f = (g; r)$, so $\lambda(\partial f) = r$. The inverse faces arise in the natural way: $f(g, r)^{-1} = f(g, r^{-1})$. As noted in Lemma 2.2.2, we can be assured this path is a loop because elements of R are equivalent to the identity in G .

Figure 2.4 gives several examples of Cayley complexes.

In principle, we could solve the Word Problem for a given presentation by forming the Cayley complex and checking if a particular word represents a loop in the skeleton. However, in general, it is impossible to build the Cayley complex for

Figure 2.4: *Examples of Cayley complexes.* The Cayley complexes for the presentations in Figure 2.3 are shown. Each face (shaded) has been labelled with its boundary word; the distinguished vertex represents the identity. Note that the Cayley complex for F_2 is just the Cayley graph for F_2 ; there are no relators in the presentation, so there are no faces to add to the Cayley graph.

an arbitrary presentation. As discussed at the end of the previous chapter, it is an unsolvable problem to determine if an arbitrary presentation is isomorphic to the trivial group, so it is impossible, in general, to determine even something as simple as whether or not the Cayley complex has more than one vertex. In the next section, we consider another type of 2-complex—the Dehn diagram—which gives useful information about a presentation, but which is actually constructible.

2.3 Dehn Diagrams

In this section, we explore *Dehn diagrams*, which some authors (e.g., [Bri02, Ger99]) refer to as “van Kampen diagrams.” Dehn diagrams provide a geometric representation of the identity-equivalent words in a presentation. Intuitively, a Dehn diagram is something like an enclosed region of the Cayley complex of a group; since the diagram is enclosed, its boundary is a loop, and so its boundary word is equivalent to the identity in the group.

Our primary goal in this section is to prove the equivalence of Dehn diagrams and identity-equivalent words. Our treatment is informed by [LS77, pp. 233–240] and [EPC⁺92, p. 40]; however, our approach is strictly combinatorial. We begin with an example before proceeding to a formal definition.

Example 2.3.1. Let $\langle A \mid R \rangle = \langle a, b \mid [a, b] \rangle$, and let $w = a^2ba^{-1}ba^{-1}b^{-2}$ be a word in F_A . The 2-complex shown in Figure 2.5 is a Dehn diagram for w . Starting from the designated basepoint, the diagram’s boundary word (read counter-clockwise) is $a^2ba^{-1}ba^{-1}b^{-2}$. The boundary of the diagram is a loop, so we know that w is equivalent to the identity. We can also see this algebraically; the group is isomorphic to $\mathbb{Z} \times \mathbb{Z}$, so $w = a^2a^{-1}a^{-1}bbb^{-2} = a^0b^0 = 1$. The boundaries of the faces of the Dehn diagram are labelled with copies of the relator $r = [a, b] = aba^{-1}b^{-1}$, and the diagram can be

thought of as a region within the Cayley complex. However, the diagram's basepoint is arbitrary and need not coincide with the vertex in the Cayley complex corresponding to the identity element in the group.

Figure 2.5: *Example of a Dehn diagram.* This 2-complex is a Dehn diagram for the word $w = a^2ba^{-1}ba^{-1}b^{-2}$ over the presentation $\langle a, b \mid [a, b] \rangle$.

To simplify the following definitions and proofs, we will assume, without loss of generality, that presentations have been symmetrized (see Definition 1.3.7). The benefit of this assumption is that we will not need to specify the basepoints of face boundaries within a Dehn diagram. To avoid trivialities, we will further assume, again without loss of generality, that presentations contain only cyclically reduced relator words of length at least two (the presence of a relator that does not meet these conditions would imply that one of the generators is equivalent to the identity and could be discarded). We now provide a formal definition of Dehn diagrams.

Definition 2.3.2 (Dehn diagram). A *Dehn diagram* of area n (where $n \geq 0$ is finite) over a symmetrized presentation $\langle A \mid R \rangle$ is an oriented 2-complex D with distinguished vertex v_0 , boundary loop ∂D starting at v_0 (and traversed in standard counter-clockwise order), and labelling map $\lambda : E \rightarrow A \cup A^{-1}$, constructed by the following process.

Start with a single vertex, v_0 . If $n = 0$, let $\partial D = 1_{v_0}$, let $\lambda(\partial D) = \varepsilon$, and stop the construction. This results in the *trivial Dehn diagram*. For $n \geq 0$, start with the trivial Dehn diagram and iteratively add n faces, as follows.

To an existing diagram, D , add a face, f , along with its boundary loop, ∂f , by choosing one of the following five options, as illustrated in Figure 2.6.

- (1) a vertex on ∂f is connected to a vertex on ∂D by a simple reduced path p ,
- (2) a vertex on ∂f is identified with a vertex on ∂D ,
- (3) a proper sub-path of ∂f is identified with a sub-path of ∂D ,
- (4) all of ∂f is identified with a sub-path q of ∂D , or
- (5) all of ∂f is identified with a sub-path q of ∂D and sub-paths p and p' of ∂D are identified with each other, where $\alpha(p) = \omega(q)$ and $\alpha(p') = \alpha(q)$, $|p| = |p'|$, and p traverses ∂D backwards (that is, clockwise).

In each case, f must be added such that v_0 remains on ∂D .

As each face is added, the labelling map λ is extended to ∂f and, in option 1, p , such that the following conditions hold:

- $\lambda(\partial f) = r$, for some $r \in R$,
- $\lambda(p) \in F_A$, and
- $\lambda(\partial D) \in F_A$.

Note that the above definition requires the boundary word of a Dehn diagram to be reduced, but we do *not* require the boundary word of a Dehn diagram to be cyclically reduced—it is possible that $\lambda(\partial D(1)) = \lambda(\partial D(|\partial D|))^{-1}$. In contrast, the boundary word of each face is a relator, and relators, by assumption, are cyclically reduced.

Figure 2.6: *Adding a face to a Dehn diagram.* The five options for adding a face, f , to an existing diagram, D , as described in Definition 2.3.2.

2.3.1 Dehn Diagrams and Identity-Equivalent Words

Whereas the the Cayley complex is a single, unique construction for the given presentation (up to isomorphism), there are many Dehn diagrams over a given presentation. Each Dehn diagram represents an identity-equivalent word; conversely, there is at least one Dehn diagram for each identity-equivalent word. We now formally state and prove these results.

Theorem 2.3.3 (Dehn diagrams represent identity-equivalent words). *Let D be a Dehn diagram of area n with distinguished vertex v_0 and labelling map λ over a symmetrized presentation $\langle A \mid R \rangle$. Then $\lambda(\partial D) \in R^*$, where ∂D is the boundary loop of D beginning at v_0 .*

Proof. We proceed by induction on n , the area of the Dehn diagram. If $n = 0$, then $\partial D = \varepsilon$, and ε is in R^* for any presentation, thus proving the base case.

Now assume that the claim holds for Dehn diagrams with $n - 1$ faces, and let D be a Dehn diagram with n faces. D was constructed from some diagram D' by adding a face f_n . By the induction hypothesis, $\lambda(\partial D') \in R^*$, so by Lemma 1.3.5, we can write

$$\lambda(\partial D') = \prod_{i=1}^{n-1} c_i^{-1} r_i c_i, \text{ for some } c_i \in F_A \text{ and } r_i \in R. \quad (2.4)$$

Face f_n was added to D' using one of the five options in Definition 2.3.2, as follows:

In the case of options 1, 2, 4, or 5, let v_1 be the vertex on $\partial D'$ at which ∂f_n is attached (either directly, or by the path p), and let s and t be the (possibly empty) sub-paths of $\partial D'$ such that $\alpha(s) = v_0$, $\omega(s) = v_1$, $\alpha(t) = v_1$, and $\omega(t) = v_0$. We now consider the boundary word $\lambda(\partial D)$ in each of these options.

$$\begin{aligned}
\text{Option 1: } \lambda(\partial D) &= \lambda(sp^{-1}(\partial f_n)pt) \\
&= \lambda(s(tt^{-1})p^{-1}(\partial f_n)pt) \text{ (with reduction, as needed)} \\
&= \lambda(st)\lambda((pt)^{-1}\partial f_n(pt)) \\
&= \lambda(\partial D')c_n^{-1}r_nc_n, \text{ where } c_n = \lambda(pt) \text{ and } \partial f_n = r_n \in R.
\end{aligned} \tag{2.5}$$

$$\begin{aligned}
\text{Option 2: } \lambda(\partial D) &= \lambda(s(\partial f_n)t) \\
&= \lambda(stt^{-1}(\partial f_n)t) \text{ (with reduction, as needed)} \\
&= \lambda(st)\lambda(t^{-1})\lambda(\partial f_n)\lambda(t) \\
&= \lambda(\partial D')c_n^{-1}r_nc_n, \text{ where } c_n = \lambda(t) \text{ and } \partial f_n = r_n \in R.
\end{aligned} \tag{2.6}$$

$$\begin{aligned}
\text{Option 4: } \lambda(\partial D) &= \lambda(st) \\
&= \lambda(sq(\partial f_n)t) \text{ (with reduction, as needed)} \\
&= \lambda(sqtt^{-1}(\partial f_n)t) \text{ (with reduction, as needed)} \\
&= \lambda(sqt)\lambda(t^{-1}(\partial f_n)t) \\
&= \lambda(\partial D')c_n^{-1}r_nc_n, \text{ where } c_n = \lambda(t) \text{ and } \partial f_n = r_n \in R.
\end{aligned} \tag{2.7}$$

Option 5: $\lambda(\partial D) = \lambda(st)$

$$\begin{aligned}
&= \lambda(s(p^{-1}p)t) \text{ (with reduction, as needed)} \\
&= \lambda(sp^{-1}(q(\partial f_n))pt) \text{ (with reduction, as needed)} \\
&= \lambda(sp^{-1}q(ptt^{-1}p^{-1})(\partial f_n)pt) \text{ (with reduction, as needed)} \\
&= \lambda(sp^{-1}qpt)\lambda((pt)^{-1}(\partial f_n)(pt)) \\
&= \lambda(\partial D')c_n^{-1}r_nc_n, \text{ where } c_n = \lambda(pt) \text{ and } \partial f_n = r_n \in R.
\end{aligned} \tag{2.8}$$

In the case of option 3, let v_1 and v_2 be the endpoints of the sub-path of $\partial D'$ that has been identified with a proper sub-path of ∂f_n . Let q refer to this sub-path (which is part of both $\partial D'$ and ∂f_n), and let p be the complement of q in ∂f_n , so that $\partial f_n = q^{-1}p$. Let s and t be the sub-paths of $\partial D'$ such that $\alpha(s) = v_0$, $\omega(s) = v_2$, $\alpha(t) = v_1$, and $\omega(t) = v_0$. Now consider the boundary word:

$$\begin{aligned}
\lambda(\partial D) &= \lambda(spt) \\
&= \lambda(s(qtt^{-1}q^{-1})pt) \text{ (with reduction, as needed)} \\
&= \lambda(sqt)\lambda(t^{-1}\partial f_nt) \text{ (because } \partial f_n = q^{-1}p) \\
&= \lambda(\partial D')c_n^{-1}r_nc_n, \text{ where } c_n = \lambda(t) \text{ and } \partial f_n = r_n \in R.
\end{aligned} \tag{2.9}$$

By substituting 2.4 into each of 2.5 through 2.9, we obtain

$$\lambda(\partial D) = \prod_{i=1}^n c_i^{-1}r_ic_i, \tag{2.10}$$

where $c_i \in F_A$ and $r_i \in R$. Thus, by Lemma 1.3.5, $\partial D \in R^*$, as required. $\square$

The next theorem provides a converse of Theorem 2.3.3.

Theorem 2.3.4 (There is a Dehn diagram for each identity-equivalent word).

Let $\langle A | R \rangle$ be a symmetrized presentation, and let $w \in R^*$. Then there exists a Dehn diagram, D , over $\langle A | R \rangle$ such that $\lambda(\partial D) = w$.

Proof. Given w , write it as a non-cancelling product of conjugates of relators,

$$w = \prod_{i=1}^n c_i^{-1} r_i c_i, \quad c_i \in F_A \text{ and } r_i \in R. \quad (2.11)$$

This product is guaranteed to exist by Lemma 1.3.8.

We proceed by induction on n , the number of terms in the non-cancelling product. If $n = 0$, then $w = \varepsilon$. Choose D to be the trivial Dehn diagram; then $\lambda(\partial D) = \varepsilon$, thus proving the base case. Now assume the claim holds for a non-cancelling product with $n - 1$ terms. Let

$$w' = \prod_{i=1}^{n-1} c_i^{-1} r_i c_i, \text{ where the } c_i\text{'s and } r_i\text{'s are the same as in (2.11),} \quad (2.12)$$

so that $w = w' c_n^{-1} r_n c_n$. Clearly, this must also be a non-cancelling product. By the induction hypothesis, there must exist a Dehn diagram D' of area $n - 1$ with distinguished vertex v_0 such that $\lambda(\partial D') = w'$. Recall that $|c_n|$ and $|r_n|$ represent the lengths of the words c_n and r_n . We now show how to construct D from D' ; see Figure 2.7 for an illustration of the first stage of this process.

Figure 2.7: *Constructing the provisional diagram in Theorem 2.3.4.* The “lollipop”, at left, is attached to the diagram D' , resulting in the provisional diagram D , at right.

Create a labelled 2-complex L (for “lollipop”) as follows. Start with a single vertex, v_0 . Attach a simple connected path p , consisting of $|c_n|$ edges, so that $\omega(p) = v_0$, and label p so that $\lambda(p) = c_n$. Attach a loop, consisting of $|r_n|$ edges, to the vertex $\alpha(p)$, attach a face f to the loop so that the loop is ∂f , and label ∂f so that $\lambda(\partial f) = r_n$. The boundary of L , starting from v_0 , is thus given by $\partial L = p^{-1}(\partial f)p$, so $\lambda(\partial L) = c_n^{-1}r_nc_n$.

Construct a provisional diagram D by identifying v_0 in L with v_0 in D' so that $\partial D = \partial D'\partial L$. In particular, L is attached so that its boundary is traversed last when tracing the boundary of D starting at v_0 (in standard counter-clockwise order). Thus, the boundary word of D is given by $\lambda(\partial D) = \lambda(\partial D')\lambda(\partial L) = w'c_n^{-1}r_nc_n$.

Now, it is possible that the boundary word of D is not a reduced word; however, by the induction hypothesis, the boundary word of D' is the reduced word w' , and $\lambda(\partial L)$ is the non-cancelling conjugator $c_n^{-1}r_nc_n$. Thus, if $\lambda(\partial D)$ is not reduced, then the cancellation must occur around the vertex v_0 , where L is attached to D' . In particular, the labels of one or more edges at the *beginning* of ∂L must correspond to the inverses of the labels of one or more edges at the *end* of $\partial D'$.

To make this notion precise, let $\partial D'(1), \partial D'(2), \dots, \partial D'(|w'|)$ and $\partial L(1), \partial L(2), \dots, \partial L(|r_n| + 2|c_n|)$ be the edges comprising the boundaries of D' and L , respectively; recall that, for each i , $\lambda(\partial D'(i))$ and $\lambda(\partial L(i))$ are elements of $A \cup A^{-1}$. Now, if $\lambda(\partial D)$ is not reduced, then there is some largest integer $m \geq 0$ such that for all $1 \leq i \leq m$, $\lambda(\partial L(i)) = \lambda(\partial D'(|w'| - i + 1))^{-1}$. We refer to m as the *length of the match* between w' and $\lambda(\partial L)$. For example, if $w' = \dots ab^{-1}a$ and $\lambda(\partial L) = a^{-1}ba \dots$, then $m = 2$.

If $m = 0$, then $\lambda(\partial D)$ is reduced, and we are done. If $m > 0$, then we take our provisional diagram D and identify pairs of edges from ∂L and $\partial D'$ to eliminate the unreduced part of the boundary word. We identify $\partial L(1)$ with $\partial D'(|w'|)$, $\partial L(2)$ with $\partial D'(|w'| - 1)$, and so on, until we have identified $\partial L(m)$ with $\partial D'(|w'| - m + 1)$.

Figure 2.8: *Reducing the boundary in Theorem 2.3.4.* The possible cases in which provisional diagram D has an unreduced boundary word, along with the results of reducing the boundary in each case, are shown.

We now examine the results of this identification process by considering the possible cases for the length of the match, m ; see Figure 2.8 for an illustration of each case.

If $0 \leq m < |c_n|$, then c_n^{-1} cancels either partially or not at all with the end of w' . The identification of edges on ∂L and $\partial D'$ is equivalent to removing the first m edges of the path p and then attaching L to D' by the remainder of p . This corresponds to option 1 in Definition 2.3.2.

If $m = |c_n|$, then c_n^{-1} cancels completely with the end of w' , but r_n does not cancel with the adjacent sub-word of w' . The identification of edges on ∂L and $\partial D'$ is equivalent to removing path p from L (so that only face f and its boundary ∂f remain) and then attaching L to D' by a vertex. This corresponds to option 2 in Definition 2.3.2.

If $|c_n| < m < |c_n| + |r_n|$, then c_n^{-1} cancels completely with the end of w' , and r_n cancels partially with the adjacent sub-word of w' . The identification of edges on ∂L and $\partial D'$ is equivalent to removing path p from L and then attaching L to D' by identifying successive edges on ∂f with successive edges on $\partial D'$. This corresponds to option 3 in Definition 2.3.2.

If $m = |c_n| + |r_n|$, then c_n^{-1} and r_n both cancel completely with the end of w' . The identification of edges on ∂L and $\partial D'$ is equivalent to removing path p from L and then attaching L to D' by identifying all of ∂f with a sub-path of $\partial D'$. This corresponds to option 4 in Definition 2.3.2; however, it is possible that $\lambda(\partial D)$ is still unreduced, as follows.

The identification of edges on ∂L and $\partial D'$ leaves ∂f attached to $\partial D'$ by a single vertex, call it v_1 . Define sub-paths s and t of $\partial D'$ such that $\alpha(s) = \alpha(t) = v_1$, $\omega(s) = \omega(t) = v_0$ and $\partial D' = s^{-1}t$. Now, it is possible that $\lambda(s)$ and $\lambda(t)$ cancel, as they were not adjacent on $\partial D'$. Let m' be the largest integer such that $\lambda(s(i)) = \lambda(t(i))^{-1}$ for

$1 \leq i \leq m'$. If $m' = 0$, then $\lambda(s)$ and $\lambda(t)$ do not cancel, and we are done. Otherwise, further identify edges $s(1), \dots, s(m')$ with edges $t(1), \dots, t(m')$, respectively. This corresponds to option 5 in Definition 2.3.2.

In each case, the face f and its boundary has been added to D' using one of the options allowed in the definition of a Dehn diagram (2.3.2), and $\lambda(\partial D)$ is a reduced word, as required. Thus, D is now a valid Dehn diagram. $\square$

The following corollary follows immediately from Theorem 2.3.4.

Corollary 2.3.5. *Each Dehn diagram with n faces corresponds to a product of the form (2.11) with n terms; conversely, a product with n terms has a Dehn diagram with n faces.* $\square$

2.3.2 Reduced Dehn Diagrams

In this section, we examine some of the peculiar constructions which are allowed by the definition of a Dehn diagram. We begin with two examples.

Example 2.3.6 (a trivial spherical diagram). Given some $r \in R$, the product $w = \varepsilon^{-1}r\varepsilon \cdot \varepsilon^{-1}r^{-1}\varepsilon$ (which is non-cancelling in the sense of Definition 1.3.6) results in a Dehn diagram that consists of two opposing faces sandwiched together, as follows:

The edges along the boundaries of faces f_1 and f_2 are successively identified until the boundaries are completely identified. Topologically, this construction is homeomorphic to a sphere. The boundary word of this diagram is ε , which is unsurprising,

since w reduces to ε ; indeed, w could be expressed more simply as the trivial product of zero terms.

Example 2.3.7 (superfluous faces). Consider the presentation $\langle a, b \mid [a, b] \rangle$, and let $r = [a, b]$. The word $w = aba^{-1}baba^{-1}b^{-3}$ can be decomposed into the following product of conjugates of relators:

$$w = r \cdot (b^{-1})^{-1}r(b^{-1}) \cdot (b^{-2})^{-1}r(b^{-2}) \cdot (ab^2a^{-1}b^{-3})^{-1}r^{-1}(ab^2a^{-1}b^{-3}) \quad (2.13)$$

However, w can also be decomposed into a simpler product of conjugates of relators:

$$w = r \cdot (b^{-2})^{-1}r(b^{-2}) \quad (2.14)$$

As shown in Figure 2.9, constructing the Dehn diagrams for these two products reveals that the diagram for the first product contains adjoining faces that have inverse boundary words that are in some sense superfluous.

Figure 2.9: *Two Dehn diagrams for the same word.* Both diagrams shown have boundary word $aba^{-1}baba^{-1}b^{-3}$ over $\langle a, b \mid [a, b] \rangle$; the diagram at left has “unnecessary” faces.

Following [LS77, p. 241], we now formally characterize the Dehn diagrams which do and do not contain such “extra” faces and explore their relationship to products of conjugates of relators which do and do not contain “extra” terms.

Definition 2.3.8 (unreduced and reduced Dehn diagrams). A Dehn diagram is said to be *unreduced* if there exist faces f_1 and f_2 such that ∂f_1 and ∂f_2 share an edge and the complementary boundary words are inverses. That is, if $\partial f_1 = ep$ and $\partial f_2 = e^{-1}q$, then the diagram is unreduced if $\lambda(q) = \lambda(p)^{-1}$. A diagram which is not unreduced is said to be *reduced*.

In terms of this definition, the diagram in Example 2.3.6 and the diagram on the left in Figure 2.9 are both unreduced; the diagram on the right in Figure 2.9 is reduced. In both cases, the unreduced diagram arises because the product of conjugates of relators from which the diagram was built had “unnecessary” terms.

Definition 2.3.9 (minimal product of conjugates of relators). Given a word w over some presentation $\langle A | R \rangle$, a product of conjugates of relators of the form (2.11) is *minimal* if there does not exist another product for w of the same form with fewer terms.

Clearly, there must be some minimal product for a given word, because the number of terms in a product is at fewest zero; however, a minimal product will generally not be unique. As indicated by the examples above, non-minimal products give rise to unreduced diagrams. The following lemma formalizes this observation.

Lemma 2.3.10. *Let w be a word over $\langle A | R \rangle$ such that $w \in R^*$, and let D be the Dehn diagram corresponding to a given decomposition of w into a product of conjugates of relators. If D is unreduced, then the product is not minimal.*

Proof. Let $\langle A | R \rangle$ and w be given such that $w \in R^*$. Let a decomposition of w into a product of the form (2.11) be given such that the Dehn diagram, D , constructed by the process described in the proof of Theorem 2.3.4 is unreduced. Let n be the number of terms in the product, and assume the product is, in fact, minimal.

Figure 2.10: *Illustration for Lemma 2.3.10.* Starting with the unreduced Dehn diagram at left, we consolidate the faces whose boundaries cancel, leading to a simpler product of conjugates. The leftmost diagram corresponds to the product 2.13; the rightmost diagram corresponds to the product 2.14.

Since D is unreduced, there must be faces f_1 and f_2 with common edge e such that $\partial f_1 = ep$ and $\partial f_2 = e^{-1}q$ for some paths p and q such that $\lambda(q) = \lambda(p)^{-1}$. Form a 2-complex D' by deleting the edge e and merging faces f_1 and f_2 to a single face f' . Since e was shared between two faces, it cannot have been on the boundary of D , and thus deleting it does not change the boundary word, so $\lambda(\partial D') = \lambda(\partial D)$.

Now, the boundary word of f' is $\lambda(\partial f') = pq$, which reduces to 1, but there is no guarantee that $\lambda(\partial f') \in R$, so D' may not be a valid Dehn diagram. However, we could nevertheless use a version of the process described in Theorem 2.3.3 to deconstruct D' and obtain a product of $n - 1$ conjugates. In this product, $n - 2$ of the terms would be the conjugates of relators (the boundary words of the faces from D that were not modified). The exceptional term would be a conjugate of the boundary word of $\lambda(\partial f')$; since this term reduces to 1, it can be eliminated from the product. This leaves a product of $n - 2$ conjugates of relators which reduces to w , contradicting the assumed minimality of the product of n terms. $\square$

This lemma has the following immediate corollary.

Corollary 2.3.11. *Given any unreduced Dehn diagram, there is some reduced Dehn diagram with the same boundary word.*

In particular, when using Dehn diagrams to study identity-equivalent words, we can restrict our attention to reduced Dehn diagrams without any loss of generality.

Note that the converse of Lemma 2.3.10 does not hold for arbitrary presentations. In particular, there exist presentations that admit reduced diagrams for the empty word which have non-zero area. Since in any presentation the empty word can be represented by the trivial Dehn diagram, these diagrams are not minimal, despite being reduced. We will revisit this issue in Chapter 5.

CHAPTER 3

THE ISOPERIMETRIC PROBLEM

In this chapter, we apply geometric ideas to the study of the Word Problem. In particular, we will show how Dehn diagrams can be used to characterize the “difficulty” of the Word Problem for a given presentation. We will define a function, called the *Dehn function*, that measures how fast the Dehn diagrams over a presentation grow with respect to the length of identity-equivalent words. We show that this function is invariant with respect to the choice of presentation for a given group, and then prove that the Word Problem is solvable for a group if and only if the group’s Dehn function is computable in finite time. We largely follow the development of these ideas as presented in [EPC⁺92, Ch. 2], with the addition of some ideas from [Ger93] and [Ger99].

3.1 The Dehn Function of a Group

The classical “isoperimetric problem” was to find a plane curve of a fixed perimeter that encloses the greatest possible area (the solution is a circle). The analogous problem for a presentation of a group is to find the reduced Dehn diagram of a given boundary length with the greatest possible area.

Recall that the area of a Dehn diagram is the number of faces, which corresponds to the number of terms in the decomposition of the boundary word into a product of

conjugates of relators. As noted in Section 1.3.1, there are potentially many decompositions of a given identity-equivalent word; there are some presentations for which it is possible to have reduced Dehn diagrams of arbitrarily large area for a given word (more on this in Section 5.1). This leads us to the following definition.

Definition 3.1.1 (combinatorial area, minimal Dehn diagram). Let $\langle A | R \rangle$ be a presentation of a group, and let $w \in F_A$ be given such that $w \in R^*$ (that is, $\bar{w} = 1$ in the group being presented). Let $\mathcal{D}_w$ represent the set of all Dehn diagrams for w , then the *combinatorial area of w* , written $\text{area}(w)$, is then given by:

$$\text{area}(w) = \min_{D \in \mathcal{D}_w} \text{area}(D) \quad (3.1)$$

That is, $\text{area}(w)$ is the area of the smallest Dehn diagram representing w ; such a diagram is said to be a *minimal* Dehn diagram for w . Note that $\mathcal{D}_w$ is guaranteed to be non-empty by Theorem 2.3.4, so this minimum is well-defined; however, the minimal diagram for a given word may not, in general, be unique.

The isoperimetric problem for a presentation can now be stated as follows: given a perimeter length l , find a Dehn diagram of maximal area relative to all other Dehn diagrams of boundary length (at most) l . The *Dehn function* of a presentation measures how this maximal area grows with word length.

Definition 3.1.2 (Dehn function). The Dehn function of a presentation $\langle A | R \rangle$ is defined as

$$\phi(l) = \max \{ \text{area}(w) : w \in F_A, w \in R^*, |w| \leq l \} \quad (3.2)$$

That is, $\phi(l)$ is the largest area of a minimal diagram corresponding to an identity-equivalent word of length at most l .

Note that the definition of the Dehn function is non-constructive, as it presupposes that we know all of the identity-equivalent words over the given presentation.

We have defined the Dehn function relative to a particular presentation, and it is not immediately obvious that the Dehn functions of different presentations of the same group will be at all similar. Fortunately, the following theorem, which we state without proof, shows that the Dehn functions of isomorphic presentations obey an equivalence relation that allows us to meaningfully speak of the Dehn function of a group.

Theorem 3.1.3 (invariance of the Dehn function). *For two functions $f : \mathbb{N} \rightarrow \mathbb{N}$ and $g : \mathbb{N} \rightarrow \mathbb{N}$, define $f \preceq g$ if there exist constants $A, B, C, D, E > 0$ so that $f(n) \leq Ag(Bn + C) + Dn + E$ for all $n \in \mathbb{N}$. Define an equivalence relation $f \sim g$ if both $f \preceq g$ and $g \preceq f$. Let ϕ be the Dehn function of a finite presentation $\langle A | R \rangle$ and ϕ' be the Dehn function of a finite presentation $\langle A' | R' \rangle$ such that $\langle A | R \rangle \cong \langle A' | R' \rangle$. Then $\phi \sim \phi'$.*

Proof. See [Ger92]. □

In particular, this theorem allows us to describe the Dehn function of a group as being linear, quadratic, cubic, etc. Groups are also sometimes described as satisfying an *isoperimetric inequality*. For example, if a group satisfies a quadratic isoperimetric inequality, then its Dehn function is bounded above by some quadratic function. Note that groups with Dehn functions of both linear and quadratic orders satisfy a quadratic isoperimetric inequality. In the next section, we give examples of groups with Dehn functions of known order.

3.2 Examples of Dehn Functions

Finite groups, free groups, and the so-called “word hyperbolic” groups have been shown to satisfy a linear isoperimetric inequality. Hyperbolic groups (also known as “negatively curved” groups) have presentations whose Cayley complexes

induce hyperbolic metric spaces. We provide one example of a hyperbolic group; an introduction to the topic with references can be found in [Ger99].

Example 3.2.1 (Fundamental group of a genus-2 surface). The fundamental group of a genus-2 surface (such as a double torus) is presented by $\langle a, b, c, d \mid [a, b][c, d] \rangle$. This presentation's Cayley complex is a tiling in which eight octagons meet at a vertex; such a tiling is possible only in a hyperbolic plane.

In some cases, such as the following example, it is possible to develop an explicit formula for the Dehn function.

Example 3.2.2 (A presentation of $\mathbb{Z} \times \mathbb{Z}$). The Cayley complex of the presentation $\langle a, b \mid [a, b] \rangle \cong \mathbb{Z} \times \mathbb{Z}$ is a square tiling of the plane, as shown at top right in Figure 2.4 (p. 28). It can be shown that in such a tiling, the largest possible area for a given boundary length is given by a shape that is as close as possible to a square. Thus, the presentation has a quadratic Dehn function [Bri02]. This Dehn function for this presentation is given by:

$$\phi(l) = \begin{cases} l^2/16 & \text{if } l \bmod 4 \equiv 0; \\ (l^2 - 4)/16 & \text{if } l \bmod 4 \equiv 2; \\ \phi(l - 1) & \text{if } l \text{ is odd.} \end{cases} \quad (3.3)$$

The “odd” condition is due to the fact that all identity-equivalent words in this presentation have even length.

In general, determining the Dehn function of a given group can be a difficult task, requiring the application of extensive algebraic, geometric, and topological machinery. The process often starts with establishing upper and lower bounds on the isoperimetric inequalities that the group must satisfy. As the results are refined, it may be possible to uniquely pinpoint the actual order of the Dehn function.

Example 3.2.3 (The Thompson group F). The Thompson group F provides an interesting example of the process of narrowing down the order of a group's Dehn function. F has a “natural” infinite presentation which can be viewed as representing a certain group of functions from the unit interval onto itself; it also has a finite presentation, which we will see in Section 5.3.1.

Historically, the first result regarding the Dehn function of F was a proof of an exponential upper bound. It was subsequently proved that the Dehn function of F is strictly subexponential. Later, it was proved that the Dehn function of F has a polynomial upper bound of n^5 , and it was recently proved that the Dehn function of F is, in fact, quadratic [Gub02].

Unlike groups that satisfy a linear isoperimetric inequality, groups such as $\mathbb{Z} \times \mathbb{Z}$ and the Thompson group F that satisfy a quadratic isoperimetric inequality do not seem to have any other collective property; Gersten described the situation as a “zoo” of unrelated types of groups [Ger99]. In the next chapter, we will examine one “species” in this zoo.

Interestingly, it has been proven that there do not exist groups whose Dehn functions are of order intermediate between linear and quadratic, but for any integer power c there exist groups whose Dehn functions are of order n^c . [Bri02]

Looking beyond polynomials, there are also groups with exponential Dehn functions, such as the following example.

Example 3.2.4 (The Baumslag-Solitar group $B_{1,2}$). The groups $B_{p,q}$ presented by the parameterized presentation $\langle a, b \mid b^{-1}a^pba^{-q} \rangle$ (for positive integers p and q) are known as the Baumslag-Solitar groups. The Baumslag-Solitar group $B_{1,2}$ is presented by $\langle a, b \mid b^{-1}aba^{-2} \rangle$. This group has an exponential Dehn function; specifically, it can be shown that for any integer $n \geq 1$, there is a Dehn diagram with boundary length

$4n + 4$ and area $2^{n+1} - 2$ [EPC⁺92]. This exponentially large area is made possible by a construction such as that shown in Figure 3.1. These diagrams are not necessarily the diagrams of greatest area given their boundary length; rather, they provide an exponential lower bound on $\phi(l)$. That is, if $l = 4n + 4$, we have $\phi(l) \geq 2^{n+1} - 2$.

Figure 3.1: *Dehn diagrams in the Baumslag-Solitar group.* These diagrams show how Dehn diagrams of exponentially large area are constructed in the Baumslag-Solitar group. Both diagrams have boundary length $4n + 4$ and area $2^{n+1} - 2$. At left, $n = 3$. The boundary word (read from the bottom left corner) is $a^{-1}b^3ab^{-3}ab^3a^{-1}b^{-3}$, which has length 16; the diagram has area 14. At right, $n = 7$; the boundary word has length 32 and the diagram has area 254. Diagrams modeled after [EPC⁺92, pp. 158–160].

There are many more groups with known Dehn functions; see [Bri02] for a recent survey. We now turn our attention to the relationship between the Dehn function and the Word Problem for a finitely presented group.

3.3 Dehn Diagrams and the Word Problem

Dehn diagrams potentially give us a way to solve the Word Problem. In particular, if we can generate all Dehn diagrams with boundary length l , then we can determine whether or not an arbitrary word of length l is equivalent to the identity simply by checking whether or not it is the boundary of one of the generated Dehn diagrams.

It is fairly clear that there is a systematic process by which we can construct Dehn diagrams for a finitely-presented group: we simply construct all diagrams of area one, followed by all diagrams of area two, etc. This corresponds to constructing all products of conjugators with one term, followed by all products of conjugators with two terms, etc. However, while there are finitely many choices for the relators in the product, there *appears* to be no bound on the number of possible choices for the conjugators; even though the presentation is finitely generated, the conjugators could be arbitrarily long. Obviously, if there are an infinite number of possibilities for the conjugator in products with one term, we are never going to be able to enumerate all such products (and hence all diagrams of area one) in finite time, much less move on to products with two terms. Fortunately, the following lemma tells us that we can determine, in advance, the maximum length of the conjugators for Dehn diagrams of a given boundary length and area. This lemma is analogous to both Lemma 2.2.4 in [EPC⁺92, p. 42] and the remark after Lemma 1.2 in [LS77, p. 239].

Lemma 3.3.1 (conjugator lengths are bounded). *Let $\langle A \mid R \rangle$ be a symmetrized presentation of a group, and let m be the length of the longest relator in R . Let w be an identity equivalent word in F_A . Then there exists a decomposition of w of the form (1.1) with n terms such that $|c_i| \leq |w| + (n - 1)m$ for all $i = 1, \dots, n$.*

Proof. Let the presentation $\langle A \mid R \rangle$, the word w , and a decomposition of w into a

product of n conjugates of relators be given. Let D be the Dehn diagram with distinguished vertex v_0 constructed for this decomposition as in the proof of Theorem 2.3.4; then $\lambda(\partial D) = w$ and the area of D is n . We will form a new product

$$w = \prod_{i=1}^{\hat{n}} \hat{c}_i^{-1} \hat{r}_i \hat{c}_i, \text{ where } \hat{c}_i \in F_A, \hat{r}_i \in R, \text{ and } \hat{n} \leq n. \quad (3.4)$$

The basic idea is that we will form a succession of new diagrams by removing faces from D one after the other (note that the resulting diagrams may not have reduced boundary words, and thus may not be valid Dehn diagrams).

Start by choosing a face $\hat{f}$ to remove from D as follows. Trace the boundary ∂D counter-clockwise from v_0 and keep track of the faces whose boundaries share at least one edge with ∂D . Order the faces in this list by the order in which they are *first* encountered, and let f_n be the last face on the list. Note that f_n may not be the *last* face encountered, but it will be the the last “new” face encountered as ∂D is traced; furthermore, f_n must only intersect ∂D along a single connected path.

Next, decompose ∂D into sub-paths, as follows. Trace ∂D *clockwise* (that is, backwards) from v_0 to the first vertex that is also on ∂f_n ; let v_1 be this vertex. Let p be the path that runs from v_1 to v_0 counter-clockwise along ∂D . Let q be the longest possible connected sub-path of the intersection of ∂f_n and ∂D such that the terminal vertex of q is v_1 . As shown in Figure 3.2, there are three cases for the length of q .

Figure 3.2: *Illustration for Lemma 3.3.1.* The three cases for the length of the path q are shown; from left to right: $0 < |q| < |\partial f_n|$, $|q| = 0$, and $|q| = |\partial f_n|$.

We now consider each of the cases shown in Figure 3.2.

- Case 1: $0 < |q| < |\partial f_n|$; that is, q is neither empty nor a loop, as shown at left in Figure 3.2. Let v_2 be the start vertex of q , and let u be the path such that $\partial f = uq$. Let s be the path that runs from v_0 to v_2 , so that $\partial D = sqp$. Choose $\hat{r}_n = \partial f_n$ and $\hat{c}_n = \lambda(p)$, and note that $|\hat{c}_n| \leq |w|$ because $|p| \leq |\partial D|$. Form diagram D' by deleting f_n and the edges and interior vertices of q . Observe that $\partial D' = su^{-1}p$, and let $w' = \lambda(\partial D')$. Then $w = w'\hat{c}_n^{-1}\hat{r}_n\hat{c}_n$, as follows:

$$\begin{aligned}
 w'\hat{c}_n^{-1}\hat{r}_n\hat{c}_n &= \lambda(\partial D')\lambda(p)^{-1}\lambda(\partial f_n)\lambda(p) \\
 &= \lambda(su^{-1}p)\lambda(p^{-1})\lambda(uq)\lambda(p) \\
 &= \lambda(su^{-1}pp^{-1}uqp) \\
 &= \lambda(sq p) = w, \text{ as required.}
 \end{aligned} \tag{3.5}$$

Furthermore, $|\partial D'| \leq |\partial D| + |\partial f_n|$, so $|\partial D'| \leq |\partial D| + m$ (because $\partial f_n \in R$, and m is the length of the longest relator in $\langle A | R \rangle$).

- Case 2: $|q| = 0$; that is, q is empty, as shown at center in Figure 3.2. Observe that deleting f_n leaves a “hole” in the diagram, so that the boundary of $\partial D'$ is not a simple path. To remedy this, separate v_1 into two vertices, v_1 and v_2 . The argument now proceeds as in the first case, with $q = \varepsilon$, $u = \partial f_n$, and $\partial D = sp$.
- Case 3: $|q| = |\partial f_n|$; that is, q is the entire boundary loop of f_n , as shown at right in Figure 3.2. After deleting f_n and q , if $\partial D'$ is a reduced path, the argument proceeds as in the first case (with $u = \varepsilon$); however, it may happen that a “spur” is left, in which case it must also be deleted. To be precise, let v_3 be the vertex where the spur joins the rest of $\partial D'$, and let t be the path that runs from v_1 to v_3 . Let p' and s' be the paths such that $p = tp'$ and

$s = s't^{-1}$. Delete t (except for v_3), so that $\partial D' = s'p'$. Let $w' = \lambda(\partial D')$; then $w = w'\hat{c}_n^{-1}\hat{r}_n\hat{c}_n$, as follows:

$$\begin{aligned}
w'\hat{c}_n^{-1}\hat{r}_n\hat{c}_n &= \lambda(\partial D')\lambda(p)^{-1}\lambda(\partial f_n)\lambda(p) \\
&= \lambda(s'p')\lambda(p^{-1})\lambda(q)\lambda(p) \\
&= \lambda(s'(t^{-1}t)p'p^{-1}qp) \text{ (with reduction, as needed)} \\
&= \lambda(spp^{-1}qp) \\
&= \lambda(sq p) = w, \text{ as required.}
\end{aligned} \tag{3.6}$$

Furthermore, $|\partial D'| < |\partial D|$, so again $w' \leq |w| + m$.

Repeat the process starting with D' ; form D'' with boundary word w'' and choose $\hat{r}_{n-1}$ and $\hat{c}_{n-1}$. Then $w = w''\hat{c}_{n-1}^{-1}\hat{r}_{n-1}\hat{c}_{n-1}\hat{c}_n^{-1}\hat{r}_n\hat{c}_n$. Observe that $|\hat{c}_{n-1}| \leq |w'|$; since $w' \leq |w| + m$, it must be the case that $|\hat{c}_{n-1}| \leq |w| + m$.

Note that this process can be repeated at most n times; after removing all faces that share at least one edge with ∂D , we will eventually be left with something that has no boundary. This something may either be a single point or a collection of spheres, which can safely be ignored, since the spheres do not contribute to the boundary word. Let $\hat{n}$ be the number of faces that share at least one edge with the boundary; after m applications of the process, we obtain the product (3.4). For $i = 1, \dots, \hat{n} - 1$, observe that $|\hat{c}_{\hat{n}-i}| \leq |w^{(i)}|$; since $|w^{(i)}| \leq |w^{(i-1)}| + m$, it must be the case that $|\hat{c}_{\hat{n}-i}| \leq |w| + im$. In particular, $|\hat{c}_i| \leq |w| + (n-1)m$ for all i , as required. $\square$

With this lemma in hand, we can systematically generate all diagrams of a given boundary length and a given area in finite time, at least in principle.

Remark 3.3.2 (theoretical vs. practical “finite time”). Recall that our goal here is to show that we can solve the Word Problem *in theory*, not necessarily in

practice. If we bound our generation of Dehn diagrams using only the preceding lemma, the time required might be enormous; consider Example 2.3.1 in the previous chapter. In that example, we have a Dehn diagram of area three for the word $w = a^2ba^{-1}ba^{-1}b^{-2}$ over the presentation $\langle A | R \rangle = \langle a, b | [a, b] \rangle$. Let's say we were given w and were instructed to find it by searching systematically through the Dehn diagrams of area three. In terms of the above lemma, $m = 4$, $|w| = 8$, and $n = 3$, so we would have to consider conjugators of length up to $8 + (3 - 1)4 = 16$. Since $A \cup A^{-1}$ has 4 symbols, there are $4 \cdot 3^{15} \approx 5.7 \times 10^7$ reduced words of length 16 (we have 4 choices for the first symbol, then each subsequent symbol can be anything other than the inverse of the preceding one). Once the presentation is symmetrized, we have 8 choices for the relator, yielding a total of 4.6×10^8 possibilities for each term in the product. Since there are three terms, we have a total of about $(4.6 \times 10^8)^3 \approx 9.7 \times 10^{25}$ possible diagrams to consider. If we can generate one million diagrams per second on a computer, it would still take on the order of 10^{12} years to generate all the necessary diagrams. The alert reader may have noticed that some of these diagrams will be unreduced; however, even if we could somehow restrict ourselves to considering reduced diagrams, the number would still be enormous.

At this point, we still have an unresolved issue with our process for solving the Word Problem using Dehn diagrams—given a word w , we generate finitely many diagrams of area one, finitely many of area two, finitely many of area three, etc., but how do we know when to stop? The answer is provided by the Dehn function. In particular, $\phi(|w|)$ tells us the largest area we must consider when searching for a diagram that has w as its boundary word.

This, in turn, leads to the question of whether or not we can compute ϕ . If so, then we have a solution to the Word Problem; if not, we seem to be in trouble.

The following theorem formalizes this observation in terms of *recursive functions*, following [EPC⁺92, Theorem 2.2.5]. The theory of recursive functions is beyond the scope of this thesis, but essentially a recursive function is one that can be computed in some finite amount of time.¹

Theorem 3.3.3. *The Word Problem for a finitely presented group is solvable if and only if the group’s Dehn function is a recursive function.*

Proof. Assume the group has a recursive Dehn function; then we apply the process described above. In particular, given an arbitrary word $w \in F_A$ over $\langle A \mid R \rangle$, we can compute $\phi(|w|)$ in finite time. We then generate all Dehn diagrams of areas $n = 1, \dots, \phi(|w|)$; $\bar{w} = 1$ if and only if w corresponds to the boundary of any of these Dehn diagrams. Lemma 3.3.1, along with the fact that the group is finitely related, guarantees that we need only check a finite number of diagrams. Thus, we can determine whether or not $\bar{w} = 1$ in finite time, so the Word Problem is solvable.

Conversely, assume the Word Problem for the group is solvable. Let $G = \langle A \mid R \rangle$ be a specific presentation for the group in question. We show that ϕ can be computed in finite time, and hence is recursive. We begin by observing that the group is finitely generated, so there are finitely many words of length no greater than each natural number l . Since the Word Problem is solvable, we can determine in finite time whether or not each of these words is equivalent to the identity. Thus, for each value of l , we can compute in finite time the set $\{w : w \in F_A, |w| \leq l, \bar{w} = 1\}$. For each member w of this set, we need only check a finite number of diagrams for equivalence to w ; as before, this is guaranteed by Lemma 3.3.1, along with the fact that the group is finitely related. To get $\text{area}(w)$, we simply take the minimum of the areas of these

¹ The idea known variously as “Church’s Thesis” or the “Church-Turing Hypothesis” states that recursive functions are precisely those functions which are “effectively computable”. See [End00] or [HU79] for more on this matter from the viewpoint of mathematical logic or computer science, respectively.

(finitely many) diagrams, and we then take the maximum of these (finitely many) areas to get the value of $\phi(l)$. $\square$

Groups with solvable Word Problems have a broad spectrum of Dehn functions. At one extreme, consider a group with a Dehn function $\phi(l) \equiv 0$. In such a group, the minimal representation of the identity is always a Dehn diagram consisting of zero relators; in particular, this is a free group. The Word Problem in a free group is correspondingly easy to solve, as the only reduced word which is equivalent to the identity is the empty word. At the other end of the spectrum, whose Dehn functions are recursive but grow faster than any exponential (see [Ger93] for an example); in such groups, there can exist Dehn diagrams of enormous area with very short boundaries.

As an application of the ideas in this chapter, we now turn our attention to one of the species represented in the “quadratic zoo”, the so-called *automatic groups*.

CHAPTER 4

AUTOMATIC GROUPS

Informally, automatic groups are groups whose Word Problem can be solved using a type of abstract computing device known as a *finite state automaton*. We begin this chapter with an introduction to finite state automata and regular languages; informed readers may wish to skip to Section 4.2, where we introduce automatic groups. After formally defining automatic groups, we prove that they satisfy a quadratic isoperimetric inequality. This shows that the Word Problem for automatic groups is solvable in principle; we close the chapter by showing that the Word Problem for automatic groups is, in fact, solvable in quadratic time.

4.1 Finite State Automata and Regular Languages

In this section, we give an overview of *finite state automata* (henceforth abbreviated “FSA” in the singular and “FSAs” in the plural), define them formally, and discuss their relationship to a class of formal languages called *regular languages*.

An FSA is an abstract model of a simple machine which, at any given moment, can be in one of a finite number of states. External inputs cause the FSA to move between states according to a *transition rule*, which is essentially the FSA’s “computer program”. FSAs are usually defined without explicit outputs; rather, the implicit output from an FSA is its current state.

Figure 4.1: *The vending machine FSA*. As described in Example 4.1.1, this diagram represents the control logic for a simple vending machine. The circles represent states and the arrows represent the possible transitions between states.

One common use of FSAs is to model control units, such as those of elevators or vending machines, as in the following example.

Example 4.1.1 (vending machine FSA). Consider a simple electro-mechanical vending machine that accepts only quarters and dispenses several kinds of candy, each of which costs fifty cents. As its input, this machine’s control unit receives three types of notifications: a quarter has been inserted, the coin return lever has been pressed, or one of the dispensing buttons has been pressed. The machine starts in what we might call the “idle” state. It remains in the idle state until a quarter is inserted; then it moves to the “active” state. In the active state, the machine will either move to the “ready” state if a quarter is inserted, or it will return to the idle state if the coin return lever is pressed (see Figure 4.1). Finally, from the ready state, the machine will return to the idle state if any of the dispensing buttons are pushed or if the coin return lever is pressed.

Note that the vending machine FSA has no means of recording the history of events other than by moving between states. This is the fundamental limitation of

FSAs—they cannot store information beyond that encoded by their current state.

FSAs can also be used to search or validate text. In such a usage, the FSA acts as an “acceptor”, rather than as a control unit, meaning that some of its states are designated as “accept” states. A sequence of inputs is given to the FSA; when the sequence is complete, if the FSA is in an accept state, the input is accepted, otherwise the input is rejected.

Figure 4.2: *Examples of acceptor FSAs.* As explained in Example 4.1.2, the FSA at top is a simple phone number validator, and the FSA at bottom finds words starting with “kit”.

Example 4.1.2 (acceptor FSAs). Two examples of acceptor FSAs are shown in Figure 4.2. The top FSA accepts strings of the form “ $d_1d_2d_3 - d_4d_5d_6d_7$ ”, where $d_i \in \{0, 1, \dots, 9\}$; this FSA could be used to (superficially) validate phone numbers. Such an FSA is somewhat unwieldy: it consists of a long chain of states, one for each number or punctuation character, with the final state being the accept state. In general, an FSA that accepts finite-length strings must have at least as many states as the finite length to be accepted. This is because an FSA has no “counter” or other means of keeping track of the number of times it has visited a given state.

The second FSA shown accepts any string of lowercase letters starting with the

letters “kit” and delimited from surrounding text by a space. This FSA could be used to find words starting with “kit” in a document. Note that state s_3 in this FSA has a self-loop; such a state allows an FSA to accept arbitrarily long strings.

It is natural to ask what happens on an “invalid” input string, such as “12-...” for the phone number validator, as no transition is shown for the input “-” at state s_2 . By convention, there is a single “fail” state which is not shown in the diagram for simplicity; all “illegal” transitions go to this state, from which there is no exit.

With these examples in mind, we shall henceforth regard all FSAs as “acceptors”. We now proceed to formally define FSAs and describe precisely the inputs they accept.

4.1.1 Formal Construction of FSAs

Our development of FSAs follows [HU79]; however, the notation has been chosen for compatibility with the rest of the material in this thesis.

Definition 4.1.3 (finite state automaton). A finite state automaton is a 5-tuple $M = (A, S, Y, s_0, \delta)$, defined as follows:

- A is the *alphabet*, a finite set of arbitrary symbols,
- S is the *state set*, a finite set (disjoint from A), of arbitrary symbols,
- $Y \subseteq S$ is the set of *accept states*,
- $s_0 \in S$ is the *start state*, and
- $\delta : S \times A \rightarrow S$ is the *transition map*.

The elements of the alphabet are called *letters*, and strings of letters are called *words*. We require that the map δ be total; that is, we require it to be defined for every input pair (s, a) , where $s \in S$ and $a \in A$.

An FSA starts in state s_0 and takes as input a string composed of letters in A . The FSA moves from state to state according to the map δ as it processes each letter in the input string. The input is accepted if the FSA finishes in one of the states in Y (for “yes”), otherwise it is rejected. To formalize the behavior of an FSA, we require some additional definitions.

Definition 4.1.4 (Kleene closure). Given an alphabet A , the *Kleene closure*, denoted A^* , is the free monoid on A , as defined in Section 1.2.¹ The operation in the monoid is concatenation and the identity is the empty string, ε .

An FSA’s transition map, δ , takes letters as inputs, but it can be recursively extended to take words as inputs.

Definition 4.1.5 (acceptance map). Given an FSA with transition map $\delta : S \times A \rightarrow S$, the *acceptance map*, $\delta^* : S \times A^* \rightarrow S$, is defined as follows:

$$\begin{aligned}\delta^*(s, \varepsilon) &= s \\ \delta^*(s, wa) &= \delta(\delta^*(s, w), a)\end{aligned}$$

where ε is the empty string, representing no input. For a word $w \in A^*$, we say the FSA *accepts* w if $\delta^*(w) \in Y$ and *rejects* w if $\delta^*(w) \notin Y$.

We will often have cause to refer to the set of all strings accepted by an FSA, which is defined as follows.

¹ In contrast, W_A is the free monoid on $A \cup A^{-1}$. Computer scientists generally define A^* in an equivalent way without appealing to group theory; see, for example, [HU79, p. 28].

Definition 4.1.6 (language accepted by an FSA). Let $M = (A, S, Y, s_0, \delta)$ be an FSA. The *language accepted by M* is the set $\mathcal{L}(M) = \{w \in A^* : \delta^*(s_0, w) \in Y\}$; that is, $\mathcal{L}(M)$ is the set words over A which, when processed from the start state, leave the FSA in an accept state.

To simplify proofs, we require that FSAs be simplified in two ways, as described in the following definition.

Definition 4.1.7 (normalized FSA). An FSA is *normalized* if:

- there is at most one state s such that $\delta(s, a) = s$ for all $a \in A$ (such a state has no “exit” and is called the *fail* state), and
- there is no state s such that $\{w \in A^* : \delta^*(s_0, w) = s\} = \emptyset$ (such a state is said to be “inaccessible”, because there is no sequence of transitions to it from the start state).

It can be shown that normalized FSAs have the same computational power as general FSAs, so we can restrict our attention to normalized FSAs without loss of generality.

To provide some insight into the possible connections between FSAs and groups, we conclude our introduction with an example of an FSA that accepts exactly the words in a free group.

Example 4.1.8 (FSA that accepts reduced words). Let F_1 be a free group on one generator. A normalized FSA that accepts words in F_1 can be constructed by choosing $S = \{s_0, s_1, s_2, s_3\}$, $Y = \{s_0, s_1, s_2\}$, $A = \{a, A\}$ (where A represents a^{-1}) and defining $\delta : S \times A \rightarrow S$ as shown in the table in Figure 4.3. Note that state s_3 is the fail state, from which there is no exit; any word which contains adjacent

cancelling letters (such as “ aaA ”) sends the FSA to this state; by convention, this state and the transitions to it are suppressed in the diagram.

Figure 4.3: An FSA that accepts words in F_1 . The state transition table and diagram are shown for the FSA described in Example 4.1.8.

We now explore connection between FSAs and a class of formal languages known as *regular languages*.

4.1.2 Regular Languages and FSAs

Informally, a regular language over an alphabet A is the subset of words in A^* that match a particular type of “wildcard” expression known as a *regular expression*.

Definition 4.1.9 (regular expression). Let A be an alphabet (any finite set of symbols). A *regular expression* over A is defined recursively as follows:

- (1) ε (the empty string) is a regular expression,
- (2) for each $a \in A$, $\mathbf{a}$ is a regular expression, and
- (3) if $\mathbf{s}$ and $\mathbf{t}$ are regular expressions, then so are $(\mathbf{st})$, $(\mathbf{s} + \mathbf{t})$, and $(\mathbf{s})^*$.

The third part of the definition allows regular expressions to be combined in three ways: $(\mathbf{st})$ represents the concatenation of two regular expressions, $(\mathbf{s} + \mathbf{t})$ indicates a choice between two regular expressions, and $(\mathbf{s})^*$ allows a regular expression to be repeated an arbitrary number of times.² By convention, we will use boldface type to denote regular expressions and to distinguish them from the strings they represent.

Definition 4.1.10 (regular language). Let $\mathbf{r}$ be a regular expression over an alphabet A . Then $\mathbf{r}$ denotes a subset of A^* , $\mathcal{L}(\mathbf{r})$, which is defined recursively as follows:

$\mathbf{r}$	$\mathcal{L}(\mathbf{r})$
ε	$\{\varepsilon\}$
$\mathbf{a}$	$\{a\}$
$(\mathbf{st})$	$\mathcal{L}(\mathbf{s})\mathcal{L}(\mathbf{t}) = \{st : s \in \mathcal{L}(\mathbf{s}), t \in \mathcal{L}(\mathbf{t})\}$
$(\mathbf{s} + \mathbf{t})$	$\mathcal{L}(\mathbf{s}) \cup \mathcal{L}(\mathbf{t})$
$(\mathbf{s})^*$	$\mathcal{L}(\mathbf{s})^*$, the Kleene closure of $\mathcal{L}(\mathbf{s})$.

A set $\mathcal{L} \subseteq A^*$ is a *regular language* over A if there is some regular expression $\mathbf{r}$ over A such that $\mathcal{L} = \mathcal{L}(\mathbf{r})$.

Example 4.1.11. Let $A = \{a, b\}$, and $\mathbf{r} = ((\mathbf{a}(\mathbf{a})^*) + (\mathbf{b})^*)$. Then $\mathcal{L}(\mathbf{r})$ is the set of all strings in A^* which consist either of either one or more a 's or zero or more b 's.

The following theorem states that the regular languages are precisely the languages accepted by FSAs. We will use the notation $\mathcal{L}(M)$ to denote the language accepted by the FSA M .

Theorem 4.1.12 (equivalence of regular languages and FSAs). *Let $\mathcal{L}$ be a*

² Users of some computer operating systems may be familiar with regular expressions from programs such as “grep” and “sed”; in those programs, the vertical bar character, “|”, is used in place of “+” for specifying a choice between two regular expressions.

language over some finite alphabet A . Then $\mathcal{L}$ is a regular language if and only if $\mathcal{L}$ is accepted by a finite state automaton with inputs over A .

Proof. See [EPC⁺92, pp. 12–14] for the standard proof of this theorem; it is given in more detail in [HU79, Ch. 2]. $\square$

One reason that regular languages are important is that they are *decidable*; that is, membership or non-membership in a regular language can be determined in finite time. Assume we are given a regular language $\mathcal{L}$ over an alphabet A and some word $w \in A^*$; by the above theorem, $\mathcal{L}$ is equivalent to some FSA, M . We can check if w is in $\mathcal{L}$ by giving w to M as input. In time proportional to the length of w , M will either accept or reject w , corresponding to the cases $w \in \mathcal{L}$ or $w \notin \mathcal{L}$, respectively.

We now show that FSAs can be used to define a class of groups for which the Word Problem is solvable.

4.2 Automatic Groups

In this section, we define automatic groups and prove important properties related to the Word Problem; we follow the exposition in [EPC⁺92, Ch. 2] with some machinery borrowed from [Far92].

Informally, an *automatic group* is one for which there exists a collection of FSAs that can recognize and compare elements of the group. One reason that automatic groups are of interest is that we can solve the Word Problem for an automatic group by using these FSAs to compare an arbitrary word with the empty word; in a finite amount of time, the FSAs will tell us whether or not the arbitrary word represents the same group element as the empty word, which is the identity.

Remark 4.2.1 (an assumption regarding presentations). Without loss of generality, we will henceforth assume that if $G = \langle A \mid R \rangle$ is a finite presentation for a

group, then the set A includes both generators and their inverses. In other words, $A = X \cup X^{-1}$, where X is a set of generators for G , and X^{-1} is a set, distinct from X and of the same cardinality as X , the elements of which represent the inverses of the elements of X . We say “ X generates G as a group” and “ A generates G as a monoid”. Under the assumption that A generates G as a monoid, the Kleene closure of A is the free monoid over $X \cup X^{-1}$; that is, $A^* = W_X$. Thus, as noted in Section 1.3, the representation map $\pi : F_X \rightarrow G$ can be extended to A^* by implicitly including the reduction step. As before, for $w \in A^*$, we will use the notation $\bar{w}$ as shorthand for $\pi(w)$.

In order for an FSA to compare words over some alphabet A , it must simultaneously take two words as input. This means that the input alphabet for the FSA should not be A , but rather $A \times A$. However, the input words can be of different lengths, which is problematic; if, for example, $|w_1| < |w_2|$, then there is no element in $(A \times A)^*$ corresponding to (w_1, w_2) . The solution is to pad the shorter word with some meaningless symbol. The necessary technical machinery is contained in the following definition, which follows the treatment in [Far92].

Definition 4.2.2 (two-variable padded language). Let A be an alphabet, let $A^{2\$} = (A \cup \{\$\}) \times (A \cup \{\$\}) \setminus (\$, \$)$. Let M be an FSA whose transition function is $\delta : S \times A^{2\$} \rightarrow S$. Let u and v be arbitrary words in A^* , assume $|u| > |v|$, and let $n = |u| - |v|$. Define $v' = v\n ; that is, v is v padded with n instances of the padding symbol $\$$. Say that (u, v) is accepted by M if $\delta((u, v')) \in Y$; the cases $|u| < |v|$ and $|u| = |v|$ are handled in the obvious ways. We say that $\mathcal{L}(M) = \{(u, v) : u, v \in A^* \text{ and } (u, v) \text{ is accepted by } M\}$ is a *regular two-variable language over the padded alphabet A* .

We are now ready to formally define automatic groups, following [EPC⁺92].

Definition 4.2.3 (automatic structure, automatic group). An *automatic structure* for a group G with finite presentation $G = \langle A \mid R \rangle$ consists of the following:

- a regular language of *canonical words*, $\mathcal{L}_W$, such that $\mathcal{L}_W \subseteq A^*$ and $\{\bar{w} : w \in \mathcal{L}_W\} = G$,
- an *equality recognizer* FSA, $M_=$, over $A^{2\mathbb{S}}$, such that $\mathcal{L}(M_+) = \{(u, v) : u, v \in \mathcal{L}_W \text{ and } \bar{u} = \bar{v}\}$
- for each $a \in A$, a *word multiplier* FSA, M_a , over $A^{2\mathbb{S}}$, such that $\mathcal{L}(M_a) = \{(u, v) : u, v \in \mathcal{L}_W \text{ and } \bar{v} = \overline{ua}\}$

An *automatic group* is a group for which some automatic structure exists.

Note that the regular language of canonical words must contain at least one representative—but may contain multiple representatives—of each element in G . The equality and multiplier automata are required to handle only these canonical words, as opposed to arbitrary words in A^* .

Example 4.2.4 (automatic structure for $\mathbb{Z} \times \mathbb{Z}$). Consider a presentation of the group $\mathbb{Z} \times \mathbb{Z}$ with generating alphabet $\{a, A, b, B\}$ (where $A = a^{-1}$ and $B = b^{-1}$). We begin by observing that all of the generators commute with one another, which allows us to develop a regular language of canonical words as follows. First, we can choose to ignore words that contain both a 's and A 's or both b 's and B 's. Second, we can require that all of the a 's or A 's precede all of the b 's or B 's. Thus, we choose regular expression $\mathbf{r} = (((\mathbf{a})^* + (\mathbf{A})^*)((\mathbf{b})^* + (\mathbf{B})^*))$ to represent our canonical words, so that $\mathcal{L}_W = \mathcal{L}(\mathbf{r})$.

We construct the equality recognizer, M_+ , as shown on the left in Figure 4.4; note that it handles only words in $\mathcal{L}_W$. For example, M_+ cannot recognize that aab

and baa are equivalent in this presentation, as it does not know how to deal with words where b 's come before a 's. Finally, we construct the word multiplier automata, M_a , M_A , M_b , and M_B . M_a is shown at right in Figure 4.4; it accepts pairs of words of the form $(a^m b^n, a^{m+1} b^n)$ and of the form $(A^m b^n, A^{m-1} b^n)$. Again, it does not understand pairs of words that are not in $\mathcal{L}_W$, such as (ab, aba) , even if the second word in the pair is, in fact, the product of the first word with the generator a . The other multiplier automata (not shown) have the same basic structure.

Figure 4.4: Automatic structure for $\mathbb{Z} \times \mathbb{Z}$. See Example 4.2.4 for details.

Although a particular group may have many automatic structures, the above definition is non-constructive and gives us no insight as to how we might find even one automatic structure for a given group. Algorithms for finding automatic structures do exist, but it turns out that it can take an unbounded amount of time to find the automatic structure for a group, even if it is known *a priori* to be automatic. The

issues involved in finding automatic structures are outside the scope of this thesis; see [EPC⁺92, Chs. 5 & 6] for an extensive exploration of both theoretical and practical issues.

Not only is it difficult to find automatic structures, the question of whether or not a group is automatic is inherently undecidable; that is, there is no algorithm which will take a presentation and answer “automatic” or “not automatic” in a finite amount of time. It is easy to show that the trivial group is automatic; given an automatic structure, it is easy to check if it represents the trivial group. Thus, if we could find an automatic structure for an arbitrary presentation in finite time, we could readily check if the presentation was isomorphic to the trivial group. However, this would contradict the unsolvability of the Isomorphism Problem for the trivial group (see Section 1.4).

Despite these difficulties, we are not completely helpless with regards to characterizing automatic groups; in the following section, we use Dehn diagrams and the idea of an isoperimetric inequality to get a partial test for automaticity.

4.3 The Quadratic Isoperimetric Inequality for Automatic Groups

In this section, we show that automatic groups satisfy a quadratic isoperimetric inequality; we follow [EPC⁺92, Sec. 2.3], with some ideas from [Ger99]. The basic idea behind the proof is straightforward. Recall that an identity-equivalent word corresponds to a loop in the Cayley complex. For an automatic group, we will show that an identity-equivalent word of length n has a Dehn diagram with area of order n^2 . In particular, we will show that there is a presentation for the group in which the Dehn diagram for a word of length n can be divided into order n strips, each containing order n faces. The proof uses the following definitions.

Definition 4.3.1 (word prefix). For a word $w = a_1a_2 \cdots a_n$, the *prefix of w of length t* is the word $w(t) = a_1a_2 \cdots a_t$ if $t < n$; if $t \geq n$, then $w(t) = w$.

Definition 4.3.2 (k -fellow travellers). If $u, v \in A^*$ are words such that the distance between $\overline{u(t)}$ and $\overline{v(t)}$ in the word metric is less than k for all t , then we say u and v are *k -fellow travellers*.

As shown by the following lemma, the paths to words that correspond to neighboring vertices in the Cayley graph of an automatic group never travel more than some pre-determined distance apart. Some authors (e.g., [EPC⁺92]) refer to this as the “Lipschitz property”, a name that comes from analysis. Also, some authors (e.g., [Ger99]) use the k -fellow traveller property as part of the definition of automatic groups, in place of the existence of the multiplier automata.

Lemma 4.3.3 (k -fellow traveller property). *Let $G = \langle A \mid R \rangle$ be an automatic group. Then there is a constant k (dependent on the choice of automatic structure) such that any two words which differ by multiplication by a generator are k -fellow travellers.*

Proof. We begin by observing that there are finitely many multiplier automata (since A is finite), and that each has a finite number of states. Let N be the maximum number of states in any of the multiplier automata, and choose $k = 2N + 1$.

Now let u and v be such that $\bar{v} = \bar{u}a$; then $(u, v) \in \mathcal{L}(M_a)$. Let t be given, and choose the shortest possible words $u', v' \in A^*$ such that $(u(t)u', v(t)v') \in \mathcal{L}(M_a)$, as shown in Figure 4.5. Both $|u'|$ and $|v'|$ must be less than N ; otherwise, when processing $(u(t)u', v(t)v')$, the automaton M_a must visit the same state more than once, contradicting the choice of shortest possible words. Because $(u(t)u', v(t)v') \in \mathcal{L}(M_a)$, $\overline{v(t)v'} = \overline{u(t)u'a}$; this implies that the distance between $\overline{v(t)}$ and $\overline{u(t)}$ is at most $|u'| + |v'| + 1 = 2N + 1 = k$. $\square$

Figure 4.5: *Illustration for Lemma 4.3.3.* This construction takes place in the Cayley graph of an automatic group.

The following lemma tells us that if we consider two canonical words that correspond to neighboring vertices in the Cayley graph of an automatic group, the difference in their lengths is bounded above by a constant.

Lemma 4.3.4 (bounded length difference). *Let $G = \langle A \mid R \rangle$ be an automatic group, and let N be the maximum number of states in any of the multiplier automata in a given automatic structure for G . Then for any choice of $w \in \mathcal{L}_W$ and $a \in A$, there is some $w' \in \mathcal{L}_W$ such that $\overline{w'} = \overline{w}a$ and $|w'| \leq |w| + N$.*

Proof. Since $\overline{w}a$ is in G , $\mathcal{L}_W$ is required, by definition, to contain some representative of $\overline{w}a$; let w' be the shortest such representative. Assume $|w'| > |w| + N$. Let $w' = a_1 a_2 \cdots a_n$. We know that $(w, w') \in \mathcal{L}(M_a)$, because $\overline{w'} = \overline{w}a$. Since $|w'| > |w| + N$, we must pad w with more than N copies of the padding symbol, $\$$, when feeding (w, w') to the automaton M_a as input. This means that M_a must visit some state more than once while processing the part of (w, w') that consists of pairs of the form $(\$, a_i)$. In particular, there is some j, k such that $|w| < j < k < |w'|$ and M_a is in the same state while processing both $(\$, a_j)$ and $(\$, a_k)$. This implies that we could

create w'' such that $(w, w'') \in \mathcal{L}(M_a)$ by deleting $a_j a_{j+1} \cdots a_{k-1}$ from w' . That is, $\overline{w''} = \overline{w'a}$, but $|w''| < |w'|$, contradicting the choice of w' as the shortest representative of $\overline{w'a}$. $\square$

The final lemma uses the previous result to show that any word over an automatic group's alphabet has a canonical form whose length is bounded above by a linear function of the word's length.

Lemma 4.3.5 (bounded lengths of canonical forms). *Let $G = \langle A \mid R \rangle$ be an automatic group, and let N be the maximum number of states in any of the multiplier automata in a given automatic structure for G . Let w_ε be some representative of the identity in $\mathcal{L}_W$. Then for any choice of $w \in A^*$, there is some $w' \in \mathcal{L}_W$ such that $\overline{w'} = \overline{w}$ and $|w'| \leq N|w| + |w_\varepsilon|$.*

Proof. Let $w(t)$ be the prefix of length t of $w = a_1 a_2 \cdots a_n$, as per Definition 4.3.1. First, consider $w(1)$; we have $\overline{w(1)} = \overline{a_1} = \overline{w_\varepsilon a_1}$, so by Lemma 4.3.4, there is some $w'_1 \in \mathcal{L}_W$ such that $\overline{w'_1} = \overline{w(1)}$ and $|w'_1| \leq |w_\varepsilon| + N$. Now consider $w(2)$. In this case, $\overline{w(2)} = \overline{a_1 a_2} = \overline{w(1) a_2}$, so by applying Lemma 4.3.4 again, we find that there is some $w'_2 \in \mathcal{L}_W$ such that $\overline{w'_2} = \overline{w(2)}$ and $|w'_2| \leq |w'_1| + N = 2N + |w_\varepsilon|$. By repeating this process a total of $|w|$ times, we find $w' \in \mathcal{L}_W$ such that $\overline{w'} = \overline{w}$ and $|w'| \leq N|w| + |w_\varepsilon|$, as required. $\square$

We are now ready to tackle the main theorem.

Theorem 4.3.6. *Automatic groups satisfy a quadratic isoperimetric inequality.*

Proof. Let $G = \langle A \mid R \rangle$ be an automatic group, with some choice of automatic structure. Without loss of generality, assume R is symmetrized. Let $w \in A^*$ be such that $\overline{w} = 1 \in G$. By Lemma 2.2.2, w corresponds to a loop ℓ_w in $\mathcal{C}^1(G, A)$, the Cayley graph of G with respect to A .

Figure 4.6: *Illustration for Theorem 4.3.6.* Left: One of the $|w|$ sub-loops of the loop ℓ_w is shown. Center: The sub-loop ℓ_t is further divided into $O(|w|)$ sub-loops by connecting corresponding vertices of p_t and p_{t-1} . Right: Each sub-loop of ℓ_t has length at most $2k + 2$; in a presentation that uses these loops as relators, w has combinatorial area $O(|w|^2)$.

Our basic strategy, illustrated in Figure 4.6, is to divide ℓ_w into $O(|w|^2)$ sub-loops; we use these sub-loops to define a finite set of relators R' such that $G \cong \langle A \mid R' \rangle$. We then show that ℓ_w encloses an area of $O(|w|^2)$ in the Cayley complex of $\langle A \mid R' \rangle$.

To proceed, let $w'_t \in \mathcal{L}_W$ be the canonical form, as described in Lemma 4.3.5, of the prefix $w(t)$ of w . Let p_t be the path in $\mathcal{C}^1(G, A)$ that runs from the identity element to the element represented by $w(t)$ such that $\lambda(p_t) = w'_t$, and let e_t be the edge connecting the group elements represented by the prefixes $w(t-1)$ and $w(t)$ (as shown at left in Figure 4.6). Let $\ell_t = p_{t-1}e_t p_t^{-1}$, then $\ell_w = \ell_1 \ell_2 \cdots \ell_{|w|}$, after reduction.

Consider an arbitrary sub-loop ℓ_t . We can decompose ℓ_t into smaller sub-loops by adding in “crossbars” that connect the second vertex on path p_{t-1} with the second vertex on path p_t , the third vertex on path p_{t-1} with the third vertex on path p_t , and so forth (as shown at center in Figure 4.6). This cuts ℓ_t into a series of “rectangular” or “triangular” sub-loops. (We get triangles when one of the paths is shorter than the other, because we must connect several vertices from the longer path to the shorter path’s terminal vertex.)

By Lemma 4.3.3, the paths p_{t-1} and p_t have the k -fellow traveller property, which guarantees that the “crossbars” connecting the paths have length at most k , meaning that the length of each small sub-loop in each sub-loop ℓ_t has length at most $2k + 2$ (as shown at right in Figure 4.6).

At this point, we have shown that the path in $\mathcal{C}^1(G, A)$ corresponding to an arbitrary identity-equivalent word can be assembled from loops of length at most $2k + 2$. Since A is finite, there are finitely many such loops, so if we take the set of all such loops, it must comprise a finite set of relators for G with respect to A . In particular, let $R' = \{w \in F_A \mid \bar{w} = 1, |w| \leq 2k + 2\}$; then $(R')^* = R^*$, so that $\langle A \mid R' \rangle \cong \langle A \mid R \rangle$ and $\mathcal{C}^1(G, A)$ is the skeleton of $\mathcal{C}^2(A, R')$, the Cayley

complex of $\langle A \mid R' \rangle$. The collection of small sub-loops comprising ℓ_w can thus be embedded in $\mathcal{C}^2(A, R')$, and each small sub-loop can then be filled in with a single face corresponding to a relator in R' . This yields a Dehn diagram for w over $\langle A \mid R' \rangle$.

By Lemma 4.3.5, each path p_t has length at most $N|w(t)| + |w_\varepsilon|$ (where N is the maximum number of states in an FSA in the automatic structure for G); clearly $|w(t)| < |w|$, so ℓ_t encloses at most $N|w| + |w_\varepsilon|$ faces. There are $|w|$ such sub-loops, for a total area of at most $N|w|^2 + |w_\varepsilon||w|$. In other words, the combinatorial area of w in the Cayley complex of $\langle A \mid R' \rangle$ is $O(|w|^2)$. Thus, $\langle A \mid R' \rangle$ satisfies a quadratic isoperimetric inequality, and so, by Theorem 3.1.3, G satisfies a quadratic isoperimetric inequality. $\square$

Because automatic groups satisfy a quadratic isoperimetric inequality, it is possible to prove that some groups are not automatic by showing that their Dehn functions are of greater than quadratic order. For example, the Baumslag-Solitar group $B_{1,2}$ has an exponential Dehn function, as discussed in Example 3.2.4 (p. 49); thus, $B_{1,2}$ cannot be automatic.

However, there exist groups with quadratic Dehn functions which have also been proven to *not* be automatic. For example, the 5-dimensional Heisenberg group $\mathcal{H}^5$ is shown in [All98] to satisfy a quadratic isoperimetric inequality, but in [EPC⁺92, Ch. 8], $\mathcal{H}^5$ is shown to not be automatic. We are forced to conclude that automatic groups are a single species genus within Gersten's "quadratic zoo".³

We conclude with an interesting fact about automatic groups. Not only is the Word Problem for an automatic group *theoretically* solvable, but, given an automatic structure, the Word Problem may be *practically solvable*, as stated in the following theorem.

³ At the time of this writing, it was unknown whether or not the Thompson group F (which has a quadratic Dehn function) is a member of the automatic species.

Theorem 4.3.7. *Given an arbitrary word w in an automatic group, it is possible to check if $\overline{w} = 1$ in $O(|w|^2)$ time.*

Sketch of proof. See [EPC⁺92, pp. 50–51] for a detailed proof; the basic idea is as follows. Let $G = \langle A \mid R \rangle$ be an automatic group with a given automatic structure. Given some word $w \in A^*$, we use the multiplier automata to construct a canonical word w' such that $\overline{w'} = \overline{w}$. Using Lemmas 4.3.4 and 4.3.5, it can be shown that this process requires $O(|w|^2)$ time. Then, using the equality recognizer, we compare w' to some known representative of the identity, which requires $O(|w|)$ time, for a total time of $O(|w|^2)$. $\square$

CHAPTER 5

ESTIMATING THE DEHN FUNCTION

In this chapter, we describe an algorithm for estimating the Dehn function of a presentation of a group and report the results of computer experiments with this algorithm. We begin with a theoretical result that will allow us simplify our algorithm by restricting its applicability to a particular class of presentations.

5.1 Aspherical Presentations

Recall that the value of the Dehn function $\phi(l)$ for a given word length l over a presentation $\langle A | R \rangle$ is defined as:

$$\phi(l) = \max \{ \text{area}(w) : w \in F_A, w \in R^*, |w| \leq l \}, \quad (5.1)$$

where

$$\text{area}(w) = \min_{D \in \mathcal{D}_w} \text{area}(D), \quad (5.2)$$

and $\mathcal{D}_w$ is the set of all possible reduced Dehn diagrams for the word w over $\langle A | R \rangle$.

The “min” in (5.2) is important because some presentations admit reduced Dehn diagrams of differing areas for a given identity-equivalent word. These presentations are precisely those that admit multiple reduced diagrams for the empty word. We now formalize this observation, following [LS77].

Definition 5.1.1 (spherical diagram). Let $\langle A \mid R \rangle$ be a finite presentation of a group and let D be a reduced Dehn diagram over $\langle A \mid R \rangle$ such that the boundary of D is the empty word. If D has at least two faces, then D is called a *spherical diagram*, because it is homeomorphic to a sphere. If D consists of exactly two faces, is said to be a *trivial spherical diagram*; otherwise it is said to be a *non-trivial spherical diagram*.

A trivial spherical diagram is shown in Example 2.3.6 (p. 40); a non-trivial spherical diagram is shown in the following example.

Example 5.1.2 (non-trivial spherical diagram). Consider the presentation $\langle a, b, c \mid [a, b], [b, c], [a, c] \rangle$. This presentation of $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$ admits non-trivial spherical diagrams such as the one shown here:

This diagram is reduced, and its boundary word is the empty word, ε . The diagram's combinatorial area is six; however, ε can be represented more simply by the trivial Dehn diagram of area zero, so the product corresponding to this diagram is not minimal.

We use the existence or non-existence of non-trivial spherical diagrams to classify presentations, as follows.

Definition 5.1.3 (aspherical presentation). Let $\langle A | R \rangle$ be a finite presentation of a group. If there exists a non-trivial spherical diagram over $\langle A | R \rangle$, then $\langle A | R \rangle$ is a *spherical* presentation; otherwise, $\langle A | R \rangle$ is an *aspherical* presentation.¹

Before proving the main result of this section, we provide examples of these two types of presentations.

Example 5.1.4 (spherical and aspherical presentations). The presentation $\langle a, b, c \mid [a, b], [b, c], [a, c] \rangle \cong \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$ is spherical because the non-trivial spherical diagram in Example 5.1.2 exists. On the other hand, it can be shown that the presentation $\langle a, b \mid [a, b] \rangle \cong \mathbb{Z} \times \mathbb{Z}$ admits only trivial spherical diagrams, so this presentation is aspherical.

We now prove the main result of this section, which is analogous to [Bri02, Lemma 7.2.9].

Lemma 5.1.5. *Let $\langle A | R \rangle$ be the presentation of a group. Let D be a reduced Dehn diagram over $\langle A | R \rangle$, and let $w = \partial D$ be the boundary word of D . If D is not a minimal diagram for w , then $\langle A | R \rangle$ is a spherical presentation.*

Proof. Let $\langle A | R \rangle$, D , and w be as in the statement of the lemma. Since D is not minimal, there is some other reduced diagram, D' , such that $\lambda(D') = w$ and $\text{area}(D') < \text{area}(D)$. Let D^{-1} represent the diagram formed by taking the “mirror image” of D , so that $\lambda(D^{-1}) = w^{-1}$. Form a new diagram S by joining D^{-1} and D' along their boundaries so that w and w^{-1} are aligned; the boundary word of S is ε , so it is a spherical diagram.

¹ An alternative definition of “aspherical” involving mappings of spheres into the Cayley complex is given in [Bri02]; see Lemma 7.2.9 and the surrounding material. Equivalence of the definitions can be shown using algebraic topology, but that is beyond the scope of this thesis.

Now, although both D and D' are reduced, it is possible that S will not be reduced; this happens if adjoining faces from D and D' have inverse boundary words, as described in Definition 2.3.8. If this happens, we can form a reduced diagram S' by successively eliminating such pairs, as described in the proof of Theorem 2.3.10.

When this process terminates, a non-zero number of faces will remain, because $\text{area}(D') < \text{area}(D)$ and we are removing one face from each diagram at each step. Furthermore, the process cannot result in a trivial spherical diagram, because such the diagram would either have one face from each of D^{-1} and D' or two faces from one of the diagrams. The former is impossible because the diagrams have different areas and the latter is impossible because it would imply that either D or D' was unreduced. Thus, S' is a non-trivial spherical diagram, so $\langle A \mid R \rangle$ is a spherical presentation. $\square$

The following corollary follows immediately from the above lemma.

Corollary 5.1.6. *In an aspherical presentation, reduced Dehn diagrams are minimal.*

Proof. The statement of Lemma 5.1.5 can be summarized as “(reduced and not minimal) implies spherical”. The contrapositive of this is “aspherical implies (not reduced or minimal)”, which is logically equivalent to “aspherical implies (reduced implies minimal)”. $\square$

In an aspherical presentation, every reduced diagram for word w will have the same area; this allows us to ignore the “min” in (5.2) and simplifies the design of our algorithm, which is described in the next section.

5.2 An Estimation Algorithm for Aspherical Presentations

In principle, if a finitely presented group has a solvable Word Problem, we can compute the value of the Dehn function for a given word length by enumerating all

identity-equivalent words of that length and finding a minimal Dehn diagram for each word. In practice, as noted in Remark 3.3.2, this exhaustive approach will take a huge amount of time; moreover, if we don't have a ready means of identifying identity-equivalent words, this method simply won't work.

Since the exhaustive approach is generally unworkable, we will instead estimate the Dehn function by randomly generating Dehn diagrams over a given presentation. By Theorem 2.3.3 each Dehn diagram represents some identity-equivalent word, so this will allow us to randomly sample the identity-equivalent words over the presentation.

We will further restrict our algorithm for use only with aspherical presentations. As described in the previous section, this allows us to assume that each Dehn diagram generated is of minimal area with respect to its boundary word.

Since we are generating Dehn diagrams rather than boundary words, the algorithm can be thought of as looking for the shortest boundary word for a diagram of a given area. In the context of the classical isoperimetric problem, this would amount to generating closed plane curves of a given area and looking for curves with the smallest perimeter, rather than generating closed plane curves of a fixed perimeter and looking for curves that enclose the largest area. From this perspective, the algorithm can be seen to estimate the Dehn function from below.

In particular, our algorithm constructs Dehn diagrams by starting with a single face and adding successive faces semi-randomly using two heuristics that encourage (but do not guarantee) the shortest possible boundary word for a given area.

The first heuristic that we use is to adjoin faces to the diagram only along edges such that the boundary of a newly-added face never completely cancels; this amounts to restricting the construction to option 3 in the definition of a Dehn diagram (2.3.2). We do not use options 1 or 2 because they tend to *increase* the boundary word length.

We do not use options 4 or 5 because they allow the formation of non-trivial spherical diagrams (which in aspherical presentations must necessarily be unreduced).

The second heuristic that we use is to adjoin faces that have the longest possible (non-total) match with the current boundary word. That is, we minimize the length of the “exposed” boundary of a newly-added face, thus increasing the length of the boundary of the diagram by as little as possible. We break ties between matches of equivalent length randomly.²

An outline of the algorithm is provided in Figure 5.1, and Figures 5.2 and 5.3 illustrate the process by which the algorithm constructs Dehn diagrams. In the next section, we discuss computer experiments with an implementation of this algorithm.

² Note that we are *not* using the heuristic of explicitly minimizing the length of the boundary word as each face is added, although the second heuristic will have this effect if the relators in the given presentation are all of the same length. Initial experiments with this alternate heuristic found it to slow down the algorithm without yielding significantly better results.

Inputs

- $\langle A | R \rangle$: a presentation
- m : number of iterations
- n : maximum diagram area

Algorithm

- (1) Symmetrize the set of relators R .
- (2) Repeat m times:
 - (a) Create a new diagram by randomly choosing one relator from R to be the first face.
 - (b) For $i = 2$ to n :
 - (i) For each location along the boundary of the diagram, compute the length of the match of each relator in R at that location.
 - (ii) Choose randomly among the relator/location pairs that have the longest match length.
 - (iii) Modify the diagram by adding the chosen relator to the boundary of the diagram at the chosen location.
 - (iv) Compute the boundary word of the resulting diagram.
 - (v) Let l be the length of the boundary word; if i is greater than the current estimate $\hat{\phi}(l)$ (or if $\hat{\phi}(l)$ is currently undefined), update the estimate $\hat{\phi}(l)$ to be i .

Outputs

- The estimate $\hat{\phi}(l)$ for each length l encountered during the generation of diagrams.

Figure 5.1: Algorithm for estimating the Dehn function of an aspherical presentation.

Figure 5.2: Illustration of the early stages of building a Dehn diagram for the presentation $\langle a, b \mid [a, b] \rangle$. At top, the symmetrized set of relators R is shown. The horizontal edges represent the generator a and the vertical edges represent b ; each relator has a “base” edge, which is shown with a thickened line. The diagram D_1 is constructed by randomly choosing from among the relators in R ; the base edge of diagram D_1 is the base edge of the chosen relator. In the next step, each edge of D_1 , other than the base edge, is considered in turn. For each edge, the relators from R that have the matching *inverse* label on their base edge are checked to see how much of their boundary matches the boundary of D_1 (if attaching a relator at a given edge would result in an unreduced diagram, the length of the match is treated as being zero). One relator is chosen randomly from among those with the longest (non-total) match and is adjoined to D_1 at the appropriate location to form diagram D_2 . In this example, the resulting estimate would be $\hat{\phi}(6) = 2$.

Figure 5.3: The example from Figure 5.2 is continued. Diagram D_3 has been constructed from D_2 (the details are similar to those of the construction of D_2 from D_1); the resulting estimate is $\hat{\phi}(8) = 3$. In the next step, the edges of D_3 are checked for matches against the relators in R . Two of the relators have a longer match with the boundary of D_3 than the others; using its “best fit” heuristic, the algorithm chooses randomly between these relators to form diagram D_4 . The resulting diagram has the same boundary length as D_3 but has greater area; thus, the estimate of $\hat{\phi}(8)$ is updated to $\hat{\phi}(8) = 4$.

5.3 Computer Experiments with the Algorithm

The algorithm described in the previous section was implemented in C++; the resulting program is about 1500 lines long and uses a moderately complicated object structure for storing and manipulating presentations, words, and Dehn diagrams. During the development, output from the program was hand-checked to ensure that the algorithm was producing correct Dehn diagrams. Once the program was determined to produce valid outputs, it was tested on a wide range of presentations, most with known Dehn functions. In this section, we report the results of those experiments.

A typical experiment began by choosing and encoding a presentation and running the program for a small number of iterations (typically 100) with a relatively small maximum area (typically 200 to 1000). The order of the Dehn function was then estimated based by using linear regression to a table of $(\ln l, \ln \hat{\phi}(l))$ pairs. In such “log-log” data, the slope corresponds to the estimated order of the Dehn function: a slope of 1 corresponds to a Dehn function of estimated order n , a slope of 2 corresponds to an Dehn function of estimated order n^2 , and so forth. The program was then run for more iterations and/or with a larger maximum area, and a new order was estimated. This process was continued until either the estimated order stabilized, or, in some cases, until the running time required became excessive. The longest experiments, with hundreds of thousands of iterations, took over 12 hours to run on a 1.53GHz AMD Athlon XP 1800+.

The broad finding from these experiments was that the performance of the algorithm depends heavily on the presentation that is given to it. For some presentations, a very accurate estimate was obtained after a relatively small number of iterations; other presentations resisted accurate estimates even after tens or hundreds

of thousands of iterations. Two major factors seem to affect the performance of the algorithm.

First, presentations with higher-order Dehn functions generally require both more iterations and a larger maximum area in order to achieve good estimates. This is because these presentations allow for relatively large diagrams with relatively small boundary length, and it takes a lot of searching to find these diagrams.

Second, presentations with many relators or long relators tend to be more difficult for the algorithm. This is because these presentations offer many choices for matching new faces at each step, meaning that the random sample of diagrams covers a relatively smaller portion of the space of all possible diagrams than it does for “simpler” presentations. In particular, this means that the algorithm is relatively less likely to find diagrams of smallest boundary length for a given area.

We now explore the results for several presentations, illustrating these findings along the way. In the following discussion, we will use lower-case letters to represent generators and upper-case letters to represent the corresponding inverses; for example, the commutator $[a, b]$ will be written as $abAB$. This non-standard notation corresponds to the format in which presentations were given to the program.

5.3.1 Groups with quadratic Dehn functions

Groups with quadratic Dehn functions are of particular interest because they seem to have a wide range of characteristics; for example, as was discussed in the previous chapter, some such groups are automatic, while others are not. One such group, $\mathbb{Z} \times \mathbb{Z}$, was chosen for initial testing because it is simple and well-understood. The program was also tested on a more complicated group of known quadratic order, Thompson’s group F . In this section, we discuss the results of experiments with these two presentations.

The program was first tested on the presentation $\langle a, b \mid abAB \rangle \cong \mathbb{Z} \times \mathbb{Z}$. As discussed in Example 3.2.2 (p. 48), there is an explicit formula for this presentation's Dehn function; this gives us something to which we can compare the estimates found by the program.

As shown at top in Figure 5.4, it turns out that the algorithm performs extremely well on this presentation. After just 100 iterations with a maximum area of 100, the exact value of the Dehn function was found for $l \leq 22$; after 1000 iterations, the exact value was found for $l \leq 40$. The log-log technique, illustrated at bottom in Figure 5.4 yields an estimated Dehn function of order $n^{1.998}$, which is essentially indistinguishable from the actual order of n^2 .

The main reason that the algorithm works so well for this group is that at any given time there will only be matches of length 1 or length 2 possible. If any matches of length 2 are available, the program will choose one of them, and this will always lead the program towards a diagram of optimally small boundary length.

The algorithm does not find it this easy to estimate the Dehn function for all groups with quadratic Dehn functions. An example is Thompson's group F , which was discussed in Example 3.2.3 (p. 49). This group has been proven (by rather complex means) to have a Dehn function of quadratic order, but no explicit formula for the Dehn function (or even for a lower bound) is currently available [Gub02].

The results of running the program on the finite presentation of F for varying numbers of iterations with a maximum area of 200 are illustrated in Figure 5.5. It is clear from the plot that the estimates are approaching the actual Dehn function from below, but increasing the number of iterations from 100 to 1000 and again from 1000 to 10000 seems to yield only marginal improvements in the estimate of the Dehn function. This stands in sharp contrast to the performance of the algorithm on the presentation of $\mathbb{Z} \times \mathbb{Z}$.

Figure 5.4: *Estimated Dehn function of $\mathbb{Z} \times \mathbb{Z}$.* Top: the estimated and actual Dehn functions of $\langle a, b \mid abAB \rangle \cong \mathbb{Z} \times \mathbb{Z}$ are shown. Bottom: a log-log plot of the data for 100 iterations, with the linear regression shown. The slope of the line is approximately 2, indicating that the estimated Dehn function is of order n^2 .

Figure 5.5: *Estimated Dehn function of the Thompson group F .* Top: The estimated Dehn function of the Thompson group F is shown for various numbers of iterations. In each case, diagrams were generated with a maximum area of 200. Bottom: The estimated exponents computed using the log-log technique are shown. The center column shows the estimated exponents if all diagrams are taken into consideration. The right column shows the estimated exponents if only diagrams of boundary length $l \geq 50$ are considered; these numbers may better reflect the actual Dehn function of F because there seem to be some peculiarities regarding the smaller Dehn diagrams over the presentation of F (see text).

The relatively poor performance of the algorithm on the presentation of F can be explained in part by the fact that, once symmetrized, the presentation of F has a relatively large number of relators (48), each with relatively many potential matching locations (10 or 14). In contrast, the presentation given for $\mathbb{Z} \times \mathbb{Z}$, once symmetrized, has fewer relators (8), each with fewer potential matching locations (4). This means that at each step, the program must consider many more options for building Dehn diagrams over the presentation of F than it must for $\mathbb{Z} \times \mathbb{Z}$. It may also be the case that the diagrams of largest area in the presentation of F are significantly different than the diagrams of near-largest area; if this is the case, the algorithm's "best match" heuristic is unlikely to work well, as it will tend to be "led astray" as it is building the diagrams.

There also appears to be a peculiarity regarding the Dehn diagrams being found for words of length less than about 50: after an initial jump, these words seem to grow in area very slowly. Examination of Dehn diagrams being found by the program indicates that when the boundary word length is in the 30 to 50 range, the diagrams tend to be tree-like; this causes the Dehn function to be approximately linear. After the word length gets above 50, the diagrams seem to grow in area significantly faster. As shown in Figure 5.5, if only diagrams with boundary word length greater than 50 are considered, the estimated exponent grows much faster as the number of iterations is increased. The reason for this behavior is unclear, but it may be due to the geometry of fitting together the 10-gons and 14-gons that represent the relators in the Thompson group; after an initial tight clustering of faces, the Dehn diagrams may need to "branch out" to a certain size before faces can again be tightly clustered.

5.3.2 Groups with Dehn functions of other polynomial orders

As was remarked in Section 3.2, there exist groups with Dehn functions of all polynomial orders.³ It is relatively easy to find aspherical groups with linear Dehn functions: all 1-relator groups are aspherical, and many 1-relator groups are hyperbolic groups (which have linear Dehn functions) [LS77]. One such group on which the program was tested is the fundamental group of a genus-2 surface, as described in Example 3.2.1 (p. 48).

Groups of higher than quadratic order are also common; for example, the group $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$ has a cubic Dehn function (its Cayley complex is a tiling of three-dimensional space with cubes). However, as was shown in Example 5.1.2, $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$ is spherical, and it seems that many well-known groups of higher order are also spherical. In a paper by Bridson [Bri02], a family of aspherical presentations is described in which each group has a Dehn function of polynomial degree m . Our program was tested on the presentation Γ_3 , which has a Dehn function of order n^2 . In this section, we discuss the results of experiments with these two presentations.

The results of running the program on a presentation of the genus-2 surface group are shown at top in Figure 5.6. The log-log technique yields an estimated Dehn function of order $n^{1.03}$, which is quite close to the theoretical order of n^1 . A linear regression to the raw data yields an estimated slope of 0.169 while the actual slope can be shown to be at least 0.176. Thus, while the correct approximate order has been found, the Dehn function is being underestimated.

Examination of the diagrams being generated by the program reveals the reason for the underestimate: As discussed in Example 3.2.1, the Cayley complex for this presentation is a tiling of the hyperbolic plane, but the diagrams generated by the

³ There also exist groups with non-polynomial orders higher than 2, but there are no groups with orders intermediate between n^1 and n^2 [Bri02]

program are generally tree-like with complete sections of the tiling appearing relatively infrequently. It appears that this happens because completing a vertex in the tiling requires a match of length 2 between the new face and the existing diagram, but many other matches of length 2 are usually possible. Thus, the “best fit” heuristic does not lead inevitably towards the optimal diagram, as it did in the case of $\mathbb{Z} \times \mathbb{Z}$; nevertheless, the algorithm is able to give a “pretty good” estimate of the Dehn function of the genus-2 surface group.

Bridson’s Γ_3 group is an aspherical group that has been shown to have a Dehn function of order n^3 . It has the following finite presentation:

$$\Gamma_3 = \langle a, b, c, d, e \mid [c, d], [c, e], b[a, d], b[a, e], c[b, d], c[b, e] \rangle \quad (5.3)$$

Like Thompson’s group F , Γ_3 has a relatively complex presentation; once symmetrized, the presentation has 56 relators. It thus came as no surprise that the program had difficulty finding the diagrams of small boundary length needed to accurately estimate the Dehn function. What was surprising is the magnitude of this difficulty. As shown at bottom in Figure 5.6, the program did find increasingly better diagrams as it was run for more iterations. However, the estimated exponent found from the log-log regression hovered in the vicinity of 1.2, when it should have been approaching 3.

Part of the reason that Γ_3 is difficult may be that it has two copies of $\mathbb{Z} \times \mathbb{Z}$ in it, represented by the generators $[c, d]$ and $[c, e]$. The “best fit” heuristic may lead the program to create planar diagrams using these generators, keeping the estimated exponent lower than it should be. However, if the program used *only* these generators, it would still find an estimated exponent of 2, which is closer to the theoretical value than it is actually finding by using all of the generators. Thus, further investigation is needed into what makes this group difficult for the program.

Figure 5.6: *Estimated Dehn functions of genus-2 surface and Γ_3 groups.*

5.3.3 Groups with non-polynomial Dehn functions

Dehn functions are not restricted to polynomials; as discussed in Example 3.2.4 (p. 49), the Baumslag-Solitar group $B_{1,2}$ has a Dehn function of order 2^n . A more extreme example is the group presented by $\langle a, b \mid BAbaBabAA \rangle$, which has been shown by Gersten to have a Dehn function that grows faster than any exponential [Ger92]. In this section, we discuss the results of experiments with these two presentations.

As shown at top in Figure 5.7, finding a good estimate of the Dehn function of $B_{1,2}$ for even moderately large values of l is quite challenging for the program. Like the presentation of $\mathbb{Z} \times \mathbb{Z}$, the presentation of $B_{1,2}$ is relatively simple (10 relators after symmetrization), so the difficulty in estimating the Dehn function must come from another source.

The reason that $B_{1,2}$ is difficult for the program appears to be that the minimal Dehn diagrams of a given boundary length get large very quickly, and, since there are exponentially many diagrams of a given area, it becomes increasingly hard for the program to find the diagrams of small boundary length as area increases. Unless it finds these diagrams, the program will underestimate the Dehn function (often severely). However, if given enough time, the program does seem to be able to estimate the Dehn function of $B_{1,2}$; it even finds diagrams that are larger than the lower bound computed in Example 3.2.4.

The results of running the program on the Gersten group are shown at bottom in Figure 5.7. It appears from the plot that the program may be finding optimal diagrams of boundary length up to about 32. It seems unlikely that the program would ever give a good estimate of the estimate of the Dehn function of this group, simply because the diagrams get huge very quickly.

Figure 5.7: *Estimates of non-polynomial Dehn functions.* Top: The estimated Dehn function of the Baumslag-Solitar group $B_{1,2}$; the diagrams discussed in Example 3.2.4 are shown as lower bounds. Bottom: The estimated Dehn function of the Gersten group, which is known to grow faster than any exponential. As the number of iterations is increased, estimated polynomial exponents for the group grow, albeit slowly.

5.3.4 Spherical presentations

Consider a spherical group such as $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$; starting from any Dehn diagram, we can construct larger Dehn diagrams with the same boundary word by attaching spheres such as the one illustrated in Example 5.1.2. Each such sphere adds nothing to the boundary length while increasing the diagram’s combinatorial area by six. An algorithm that allows such diagrams to be created will thus report arbitrarily high values of the Dehn function for each boundary word length.

The algorithm described in this thesis does not allow spherical diagrams to be created, but it nevertheless fails to correctly estimate the Dehn function for spherical groups. This behavior was discovered accidentally when the program reported absurdly high values for the Dehn function of the group $\mathcal{H}^3$, the 3-dimensional Heisenberg group.

Investigation of the diagrams being produced yielded an interesting observation: the program found a diagram of area five that had the same boundary word as a particular relator. The diagram could thus be “substituted” for that relator in any other diagram, thus increasing the second diagram’s area without altering the boundary word. Moreover, one of the faces in the diagram represented that particular relator, so the diagram could be “nested” inside itself. Thus, the program could generate arbitrarily large diagrams of a fixed boundary length, leading to the huge values of the Dehn function.

Further investigation uncovered the known result that $\mathcal{H}^3$ is spherical; furthermore, in any spherical group, some or all of the relators can be written as a product of conjugates of the other relators [LS77].

This incident led to the idea for an alternative use for the program: checking presentations for sphericity. If a presentation is spherical, the program could prove

this by simply finding a spherical diagram. Otherwise, the program could give evidence for, but never prove, asphericity, because it only samples the space of Dehn diagrams.

5.4 Conclusions and Future Work

Based on the experiments described in the preceding section, we conclude that the program can accurately estimate the Dehn function for some, but by no means all, aspherical presentations. In particular, if a presentation has a Dehn function of high order or has complex relators (in the sense of having many relators or long relators), the program may find it difficult or impossible to find the diagrams of relatively small boundary length that are needed to accurately estimate the Dehn function.

There are several ways in which the algorithm could be improved. One obvious improvement would be to extend it to work with spherical presentations. This would require storing the area of the minimal diagram found so far for each boundary word and comparing each new diagram generated to this list. This improvement would not come for free: it would increase both the time and memory requirements of the program.

Another way to possibly improve the program's overall performance would be to apply more sophisticated search techniques. The search algorithm currently implemented is what artificial intelligence researchers refer to as "best-first search". There are many variations and improvements on best-first search in the AI literature. For example, best-first search with backtracking would keep track of the best diagram found of each area, and would then use that diagram as the starting point for all diagrams of greater area, rather than starting from scratch with each iteration. It is also possible that shifting to an entirely different search paradigm (e.g., simulated

annealing, genetic algorithms, etc.) may yield better results.

Regardless of the search technique employed, another possible improvement to the algorithm would be to apply a divide-and-conquer technique by building diagrams using subsets of the relators. This might improve performance in the case of presentations with many relators, such as the presentation of Γ_3 .

Finally, the program has little support for visualizing the Dehn diagrams it generates; during the debugging process, diagrams were painstakingly drawn by hand based on a printout of the relators and their adjacencies. It would make the program easier and more interesting to use if the diagrams could be displayed in a user-friendly fashion.

BIBLIOGRAPHY

- [All98] Daniel Allcock, An isoperimetric inequality for the Heisenberg groups, Geometric and Functional Analysis **8** (1998), 219–233.
- [Bri02] Martin R. Bridson, The geometry of the word problem, Invitations to geometry and topology, Oxf. Grad. Texts Math., vol. 7, Oxford Univ. Press, Oxford, 2002, pp. 29–91.
- [End00] Herbert B. Enderton, A mathematical introduction to logic, 2nd ed., Harcourt Academic Press, 2000.
- [EPC⁺92] David B. A. Epstein, M. S. Paterson, J. W. Cannon, D. F. Holt, S. V. F. Levy, and W. P. Thurston, Word processing in groups, A. K. Peters, Ltd., 1992.
- [Far92] Benson Farb, Automatic groups: A guided tour, L'Enseignement Math. **38** (1992), 291–313.
- [Ger92] S. M. Gersten, Dehn functions and ℓ_1 -norms of finite presentations, Algorithms and Classification in Combinatorial Group Theory (G. Baumslag and C. F. Miller III, eds.), MSRI series, vol. 23, Springer-Verlag, 1992, pp. 195–224.
- [Ger93] ———, Isoperimetric and isodiametric functions of finite presentations, Geometric Group Theory, Volume 1 (G. Niblo and M. Roller, eds.), London Math. Society Lecture Notes Series, vol. 181, Cambridge Univ. Press, 1993, pp. 79–96.
- [Ger99] ———, Introduction to hyperbolic and automatic groups, Proceedings of the CRM Summer School in Combinatorial Group Theory, Banff 1996 (O. Kharlampovich, ed.), CRM Proceedings and Lecture Notes Series, vol. 17, AMS, 1999.
- [Gri99] Pierre A. Grillet, Algebra, John Wiley & Sons, Inc., 1999.

- [Gub02] V.S. Guba, The Dehn function of Richard Thompson's group f is quadratic, <http://arxiv.org/abs/math.GR/0211395>, November 2002.
- [HU79] John E. Hopcroft and Jeffrey D. Ullman, Introduction to automata theory, languages, and computation, Addison-Wesley, 1979.
- [Kin93] L. Christine Kinsey, Topology of surfaces, Springer, 1993.
- [LS77] Roger C. Lyndon and Paul E. Schupp, Combinatorial group theory, Springer-Verlag, 1977.
- [MKS66] Wilhelm Magnus, Abraham Karass, and Donald Solitar, Combinatorial group theory: Presentation of groups in terms of generators and relations, Interscience Publishers, 1966.