# Automatic Groups: A Brief Introduction

Liran Li

May 5, 2025

Submitted to the
Department of Mathematics
of the University of Virginia
in partial fulfillment of the requirements
for the degree of
Bachelor of Arts with Distinction

Faculty Advisor: Professor Thomas Koberda

# Abstract

The goal of this thesis is to provide a gentle introduction to the concept of *automatic groups*, uniting techniques from formal language theory and geometric group theory. We review relevant notions and tools from theoretical computer science and geometric group theory, then survey the basic definitions, properties, and variants of automatic groups. Finally, we discuss examples and non-examples to illustrate when and why a group admits an automatic structure.

# Acknowledgments

This thesis would not have been possible without the support and guidance of many individuals, to whom I owe my deepest gratitude.

First and foremost, I would like to express my heartfelt thanks to my advisor, Professor Thomas Koberda. I first met Professor Koberda in my very first class at UVA, MATH 2315. Enrolling in his Advanced Calculus and Linear Algebra sequence remains one of the best decisions I made during college. It was through his wisdom that I was introduced to the world of higher mathematics, and under his guidance, I developed the determination to pursue pure mathematics. Over the following three years, he mentored me as I explored a variety of mathematical topics, and it was also through his inspiration that I began studying the field I currently focus on: Cryptography. Beyond mathematics, I am deeply grateful for his encouragement during difficult times and for his generous advice on matters both academic and personal. His unwavering support during multiple application seasons has been invaluable. More importantly, I am thankful for the way he shaped my academic character — so much so that, whether as a learner or a teaching assistant, I often find myself asking, "What would Professor Koberda do?"

I would also like to thank my girlfriend, Tianshu Huang, and my parents, Fei Li and Hui Han, for their unconditional love and support, which made these four wonderful years possible and encouraged me to pursue deeper knowledge and higher aspirations. I am equally grateful to all my family members and to Tianshu's parents for their care and encouragement.

At UVA, I have had the privilege of learning from many outstanding professors. I would especially like to thank Professor Ershov, Professor Maloni, Professor Qi, Professor Wang, Professor Abramenko, and Professor Quigley for their guidance and inspiration. I must

also mention three mathematics mentors who played important roles in my early academic journey — Xin Jin, Dong Li, and Zhenyu Li — whose mentorship and care during my elementary, middle, and high school years laid the foundation for my love of mathematics.

Finally, I would like to thank the many friends I met at UVA, who made these years so much more memorable. Due to space limitations, I will mention only a few names: Ziming Chen, Darien Farnham, Samuel Goldberg, Junyang He, Ziang Jiao, Jicheng Li, Yunsheng Lu, Yikai Ma, Lam Nguyen, Yichi Zhang, Hanzhang Zhao, and Yancheng Zhou.

# Contents

# List of Notation

$\boldsymbol{A}$ A finite *alphabet* or *set of generators*. In language theory contexts, $A$ is the set of symbols used to form words. In group-theoretic contexts, $A$ is the finite set generating the group.

$\boldsymbol{a \in A}$ A *letter* in the alphabet $A$.

$\boldsymbol{w}$ Generic *words* (strings) over the alphabet $A$. Often written as $w = a_1 a_2 \dots a_m$.

$\boldsymbol{|w|}$ The *length* of a word $w$. Defined by counting the number of letters in $w$.

$\boldsymbol{\varepsilon}$ The *empty word* (nullstring), which is a string of length 0.

$\boldsymbol{A^*}$ The set of all finite words over the alphabet $A$. Under concatenation, $A^*$ is a *free monoid* on $A$.

$\boldsymbol{L}$ A *language* over the alphabet $A$. Typically a subset of $A^*$.

$\boldsymbol{M}$ A *finite state machine*.

$\boldsymbol{L(M)}$ The *language recognized* by a machine (e.g., a finite state automaton $M$).

$\boldsymbol{L^*}$ **(Kleene Star)** The *star closure* of $L$, i.e. the set of all finite concatenations of strings from $L$.

**\$** A special *padding symbol*.

$\boldsymbol{\neg, \wedge, \vee, \exists, \forall}$ Logical connectives and quantifiers used when extending first-order predicates over languages.

$\boldsymbol{G = \langle A | R \rangle}$ A *group presentation* with finite generating set $A$ and set of defining relators $R \subseteq F(A)$.

$\overline{\boldsymbol{w}}$ The *group element* of $G$ represented by a word $w \in A^*$. Formally, $\pi(w)$ under the projection $\pi : F(A) \to G$.

$\boldsymbol{\ell(g)}$ The *word length* of the group element $g \in G$, i.e. the minimal number of generators (and their inverses) needed to write $g$.

$\boldsymbol{Area(w)}$ Combinatorial area of $w$.

$\boldsymbol{\delta_{\langle A|R\rangle}}$ The *Dehn function* of a given group presentation $G = \langle A|R\rangle$.

$\boldsymbol{\Omega}$ A *Van Kampen diagram.*

$\boldsymbol{\partial\Omega}$ The *boundary* of the Van Kampen diagram $\Omega$.

$\boldsymbol{L(\partial\Omega)}$ The *boundary label* of the Van Kampen diagram $\Omega$.

$\boldsymbol{C}$ A *cell* in the Van Kampen diagram $\Omega$.

$\boldsymbol{L(\partial C)}$ The *boundary label* of a cell $C$ in the Van Kampen diagram.

$\boldsymbol{\Gamma(G, A)}$ The *Cayley graph* of $G$ with respect to the generating set $A$. Vertices are elements of $G$, edges labeled by generators $a \in A$.

$\widehat{\boldsymbol{w}}$ A *path* of the word $w$ in the Cayley graph $\Gamma(G, A)$.

$\boldsymbol{d_A}$ The *word metric* induced by $A$ in $\Gamma(G, A)$.

$\boldsymbol{d_{unif}}$ The *uniform metric* on the set of maps from a set $X$ to a metric space $Y$.

$\boldsymbol{A^{+\epsilon}}$ The *closed $\epsilon$-neighborhood* of a subset $A$ in a metric space $(X, d)$.

$\boldsymbol{dist}$ The *Hausdorff distance* between two subsets of a metric space.

$\boldsymbol{length(p)}$ The *length* of a path $p$ in a metric space $(X, d)$.

$\boldsymbol{[x, y]}$ A chosen *geodesic* from $x$ to $y$ in a geodesic metric space $(X, d)$.

$\boldsymbol{M_x}$ A *multiplier automaton.*

$\boldsymbol{M_\varepsilon}$ The *equality recognizer.*

$\boldsymbol{(A, W, L, M_x)}$ An *automatic structure.*

# Chapter 1

# Introduction and Roadmap

## 1.1 Overview and Motivation

The study of *automatic groups* lies at the intersection of theoretical computer science and geometric group theory. On one hand, core questions in group theory such as the *word problem* and *isoperimetric inequalities* naturally involve algorithms, computability, and language theory. On the other hand, tools from geometry and topology like *Cayley graphs* and *Van Kampen diagrams* provide a powerful framework for examining properties of groups.

Roughly speaking, a group is *automatic* if there is a finite state automaton that recognizes well-behaved normal forms for all elements of the group, with additional automata controlling how those normal forms respond to multiplication by generators. Intuitively, this makes core group-theoretic problems (e.g., testing whether two words represent the same element) solvable via uniform, finite-state computational procedures. As a result, automatic groups:

- **Admit quadratic isoperimetric inequalities.** They satisfy strong bounds on how "difficult" it can be to prove a word is trivial.
- **Have a quadratic-time word problem.** One can algorithmically decide if two words represent the same element within time proportional to the product of their lengths.
- **Exhibit rich geometric structures.** Well-known families of groups such as *hyperbolic groups* and *Euclidean groups* are automatic.

Nonetheless, many interesting groups (e.g., certain nilpotent or Baumslag–Solitar groups) fail to be automatic. Understanding precisely which groups are automatic remains an active area of research, with open questions on algorithmic, geometric, and algebraic fronts.

## 1.2 Outline of the Thesis

The thesis is organized into four main technical chapters, structured to give a comprehensive view of automatic groups from foundational definitions and properties to applications and examples:

- **Chapter 2: Theory of Computation Background.**
  Lays out key concepts in formal language theory—alphabets, words, regular languages, finite state automata, closure properties, and the pumping lemma. These form the backbone for defining when a group's normal forms can be recognized by a finite automaton.

- **Chapter 3: Geometric Group Theory Background.**
  Introduces relative fundamental notions in group theory (group presentations, Dehn functions, word problems) and further relative geometric group theory ideas (Cayley graphs, Van Kampen diagrams, hyperbolicity).

- **Chapter 4: Automatic Group Theory.**
  Defines synchronous automatic groups, details their main properties (quadratic isoperimetric inequalities, Lipschitz conditions, biautomatic variants), and presents key algorithms (Todd–Coxeter, Knuth–Bendix) for constructing or verifying automatic structures. Variants such as **ShortLex**-automatic, geodesic automatic, and prefix-closed structures are highlighted.

- **Chapter 5: Applications of Automatic Groups.**
  Explores examples of classes of groups do (or do not) admit automatic structures. Focuses on hyperbolic groups, Euclidean groups, nilpotent groups, and Baumslag–Solitar groups, illustrating how theoretical definitions from Chapter 4 manifest in practice.

With these components in place, the thesis aims to be a largely self-contained account of automatic group theory, bridging the essential computational and geometric group theoretic viewpoints.

## 1.3 Guidance of Reading

Here we provide a gentle guideline for readers from different areas and with different purposes to read this introduction.

- Readers with a theoretical computer science background may wish to focus on Chapter 2 in detail for the formal definitions of regular languages and automata used in this thesis. Chapter 3's treatment of group theory (particularly Cayley graphs and word problems) will then clarify how computational machinery interfaces with group presentations. Chapters 4 and 5 complete the link between finite automata, geometric group theory, and examples.

- Readers from a geometric group theory background might skim Chapter 2 to ensure familiarity with automata theory and relevant definitions. Chapter 3 will feel more natural, discussing Cayley graphs, Dehn functions, and hyperbolicity. Chapter 4 then formalizes automatic groups, connecting their geometric and algebraic properties to finite state automata recognition. Chapter 5's concrete examples reveal why certain groups are automatic (or not).

# Chapter 2

# Theory of Computation Background

## 2.1 Languages

In mathematics, an *alphabet* is typically treated as a finite set of abstract symbols, and a *word* is a finite sequence of these symbols. While this framework may differ from common usage in natural languages or programming contexts, it provides the precision needed for formal language theory.

One of the central notions in this theory, and a focal point of this thesis, is the *finite state automaton*—a computational model capable of recognizing certain classes of languages. As we will see, such automata also play an important role in group-theoretic settings: by representing words in a group's generators, one can employ automata-based techniques to develop efficient algorithms for a variety of group-computational tasks.

**Definition 2.1.1** (Alphabet, Letter, and Word)**.** An *alphabet* $A$ is a finite set, sometimes we use the terminology *word alphabet*. A *letter* is an element of the set $A$, and a *word* or a *string* over the alphabet $A$ is a finite sequence of letters.

We can think of the word as given an integer $m \geq 0$ and a mapping $\{1, ..., m\} \to A$.

**Definition 2.1.2** (Word Length)**.** With such mapping in mind, if $w : \{1, ..., m\} \to A$ is a word, we call $m$ the *word length* of $w$, denoted by $|w|$.

**Definition 2.1.3** (Empty Word)**.** If $|w| = 0$, that is $m = 0$, the domain will be the nullset and there is a unique such mapping, we call it the *empty word*, or *nullstring*, denoted by $\varepsilon$.

Sometimes in addition we denote it by $\varepsilon_A$ to indicate the alphabet $A$.

Note that besides the obscure mapping, the word of length $m$ can also be identified with the Cartesian product $A^m = \underbrace{A \times A \times \cdots \times A}_{m \text{ copies}}$, but the difference here is that the elements in this Cartesian product are written out as strings rather than usual parenthesis and commas. With such identification, we can define the set $A^*$ and the binary operation called concatenation as follows:

**Definition 2.1.4** (String over $A$, Concatenation)**.** We denote the set of strings over the alphabet $A$ by $A^*$, if $w_1 = a_1 \cdots a_m$ and $w_2 = b_1 \cdots b_n \in A^*$, we define the concatenation $w_1 w_2 = a_1 \cdots a_m b_1 \cdots b_n$, which gives a word of length $|w_1| + |w_2| = m + n$.

**Proposition 2.1.5.** *Suppose we have an alphabet $A$ and the set $A^*$ of strings over $A$. Then the concatenation gives a binary operation on $A^*$ and $A^*$ is a monoid.*

**Definition 2.1.6** (Subword, Prefix, Suffix)**.** If $w$ is a word over an alphabet $A$, a *subword* $u \in A^*$ is a word such that $w = puq$ for some $p, q \in A^*$. We say that $p$ is a *prefix* of $w$, that $q$ is a suffix of $w$. If $t \geq 0$ is an integer, by $w(t)$ we mean the prefix of length $t$ of $w$. If $t \geq |w|$, it means $w$ itself.

**Definition 2.1.7** (Language)**.** A *language $L$ with alphabet $A$* is a subset of $A^*$.

Naturally, we want to extend the operation of concatenation of words to languages as follows:

**Definition 2.1.8** (Concatenation of Languages)**.** Let $L_1$ and $L_2$ be languages over the same alphabet $A$, we define the *concatenation of languages* $L_1 L_2$ to be the set of stings $w$ such that $w = w_1 w_2 \in A^*$, with $w_1 \in L_1$ and $w_2 \in L_2$.

**Definition 2.1.9** (Star Closure)**.** Given a language $L$, we define the *star closure* of $L$ as

$$L^* \coloneqq \bigcup_{n \geq 0} L^n$$

where $L^0 = \{\varepsilon\}$ and $L^n \coloneqq L^{n-1} L$ is defined inductively for $n > 0$. It is also called the *Kleene closure*.

**Remark 2.1.10.** Remark that if $L = \emptyset$, then its star closure is $L^* = \{\varepsilon\}$, a language with only one element: the empty word. However, if $L = A$, the alphabet, we will have two definitions of $A^*$, they are the same object.

In some cases, we may wish to interpret $n$-tuples of strings as strings of $n$-tuples. However, the individual strings may have different lengths. To ensure that the lengths are consistent so that the strings are easier to work with, we introduce the following definition:

**Definition 2.1.11** (Padded Alphabet)**.** Let \$ denote the *end-of-string padding symbol*, let $A$ be an alphabet that does not contain \$. We define the *padded alphabet associated with* $A$ to be the set $B \coloneqq A \cup \{\$\}$. For alphabets $A_1, ..., A_n$ with corresponding padded alphabets $B_1, ..., B_n$ and padding symbols $\$_1, ..., \$_n$, we define the *padded alphabet associated with* $(A_1, ..., A_n)$ to be the set

$$B \coloneqq B_1 \times \cdots \times B_n \backslash \{(\$_1, ..., \$_n)\}$$

Next, we introduce some commonly used orderings on alphabets, which will be referenced in later discussions. Given an alphabet $A$, we consider the orderings between two strings $v, w \in A^*$:

1. *Lexicographic order.* In the *lexicographic order*, with a fixed ordering on the alphabet $A$, for two words $v = a_1 a_2 \cdots a_m$ and $w = b_1 b_2 \cdots b_n$ in $A^*$, $v < w$ if and only if there exists an index $k \leq \min\{m, n\}$ such that $a_i = b_i$ for all $1 \leq i < k$ and $a_k < b_k$, or $v$ is a proper prefix of $w$.

2. *Length order.* In the *length order*, $v < w$ if and only if $v$ is shorter than $w$. This is a partial order.

3. ***ShortLex** order.* In the ***ShortLex** order*, $v < w$ if and only if $v$ is shorter than $w$, or they have the same length and $v$ comes before $w$ in lexicographical order.

4. *Dictionary order.* In the *dictionary order*, $v < w$ if and only if $v$ is a proper prefix of $w$, or $v$ and $w$ have prefixes $v'$ and $w'$ of the same length such that $v'$ comes before $w'$ in lexicographical order.

Next, we introduce a fundamental concept in both mathematics and computer science: recursive functions. As the name suggests, they are defined recursively as follows:

**Definition 2.1.12** (Basic Primitive Recursive Function)**.** The *basic primitive recursive functions* are given by the following axioms:

1. *Constant function.* For each natural number $n \in \mathbb{N}$ and every $k$, the *$k$-ary constant function* $C_n^k$ is primitive recursive, which is defined by $C_n^k(x_1, ..., x_k) \coloneqq n$.

2. *Successor function.* The 1*-ary successor function* $S$ which returns the successor of its argument according to the Peano axioms, is primitive recursive and is defined by $S(x) \coloneqq x + 1$.

3. *Projection function.* For all natural numbers $i, k \in \mathbb{N}$ such that $1 \leq i \leq k$, the *$k$-ary projective function* $P_i^k$ is primitive recursive and it is defined by $P_i^k(x_1, ..., x_k) \coloneqq x_i$.

Starting from these three basic primitive recursive functions, we can obtain more complex primitive recursive functions by applying the operations given by the following definition:

**Definition 2.1.13** (Primitive Recursive Function Operations)**.** We define two operations of primitive recursive functions:

1. *Composition operator.* Given any $m$-ary function $h(x_1, ..., x_m)$ and $m$ $k$-ary functions $g_1(x_1, ..., x_k), ..., g_m(x_1, ..., x_k)$, we define the *composition operator* $\circ$ by $h \circ (g_1, ..., g_m) \coloneqq f$, where $f$ is defined by

$$f(x_1, ..., x_k) = h(g_1(x_1, ..., x_k), ..., g_m(x_1, ..., x_k))$$

2. *Primitive recursion operator.* Given any $k$-ary function $g(x_1, ..., x_k)$ and $(k+2)$-ary function $h(y, z, x_1, ..., x_k)$, we define the *primitive recursion operator* $\rho$ by $\rho(g, h) \coloneqq f$, where $f$ is defined recursively by:

$$f(0, x_1, ..., x_k) \coloneqq g(x_1, ..., x_k)$$
$$f(S(y), x_1, ..., x_k) \coloneqq h(y, f(x_1, ..., x_k), x_1, ..., x_k)$$

Finally, we can define the primitive recursive functions:

**Definition 2.1.14** (Primitive Recursive Function)**.** The *primitive recursive functions* consist of the basic primitive recursive functions and all functions obtained from them by applying the two defining operations a finite number of times.

**Definition 2.1.15** (Recursively Enumerable Set)**.** A subset $X \subseteq \mathbb{N}$ is said to be *recursively enumerable* if there exists a recursive function $f : \mathbb{N} \to \mathbb{N}$ such that the range of $f$ is exactly $X$. More generally, a subset $X \subseteq A^*$, where $A$ is a finite alphabet, is *recursively enumerable* if there exists a recursive function $f : \mathbb{N} \to A^*$ such that every element of $X$ appears as some output of $f$.

## 2.2 Machines

In language and automata theory, it is common to classify languages based on the types of machines capable of recognizing them. In this context, we are particularly interested in regular languages, which are precisely the class of languages recognized by finite state automata.

**Definition 2.2.1** (Finite State Automaton)**.** A *finite state automaton* is a 5-tuple $M = (Q, A, \delta, q_0, F)$, where

1. The finite set $Q$ is the *set of states*.
2. The finite set $A$ is the *alphabet*.
3. The function $\delta : Q \times A \to Q$ is the set of *transition functions*.
4. The state $q_0 \in Q$ is the *start state*.
5. The set $F \subseteq Q$ is the *set of accept states*.

Based on such definition of finite state automaton, we define the notion of *computation* of the machine $M$:

**Definition 2.2.2** (Computation of Finite State Automaton)**.** Let $M = (Q, A, \delta, q_0, F)$ be a finite state automaton, then a *computation* of $M$ is a sequence $q_0, a_1, q_1, a_2, q_2, ..., a_n, q_n$

such that $n \geq 0$ and for each $1 \leq i \leq n$, we have a transition function $\delta_i \in \delta$ such that $\delta_i : (q_{i-1}, a_i) \to q_i$. We call the word $a_1 \cdots a_n$ the *label* on such computation, and the computation is called *successful* if $q_n \in F$.

As mentioned earlier, languages are classified according to the types of machines capable of recognizing them. Here we clarify the notion of *recognition*. Since our focus is solely on regular languages and finite state automata, there is no ambiguity regarding the type of machine throughout this article. Accordingly, we now present the formal definition of a regular language.

**Definition 2.2.3** (Regular Language)**.** Let $M = (Q, A, \delta, q_0, F)$ be a finite state automaton, then the language $L(M)$ *recognized* by $M$ is

$$L(M) \coloneqq \{w \in A^* \mid w \text{ is accepted by } M\}$$

and we call such $L(M)$ a *regular language*.

It is worth noting that regular languages can also be defined using *regular expressions*, which are symbolic notations used to describe patterns in strings. A regular expression consists of characters and operators that define sets of strings in a formal language. Furthermore, there are two primary types of finite state automata: deterministic finite state automata (DFSA) and non-deterministic finite state automata (NFSA), both of which play important roles in the theory of computation. However, since discussing these models and their associated results is not the primary focus of this article, we simply state the classical equivalence theorem due to Kleene, Rabin, and Scott:

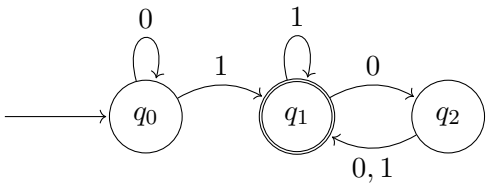
**Theorem 2.2.4.** *Let $A$ be a finite alphabet, then the following three conditions on a language $L$ over $A$ are equivalent:*

1. *The language $L$ is recognized by a deterministic finite state automaton.*
2. *The language $L$ is recognized by a non-deterministic finite state automaton.*
3. *The language $L$ is defined by a regular expression.*

**Example 2.2.5.** Drawing diagram is one important way to study finite state automata, here we see one example of finite state automaton and the associated diagram. Consider the finite state automaton $M = (Q, A, \delta, q_0, F)$ defined by

1. The set of states are: $Q = \{q_0, q_1, q_2\}$.
2. The alphabet is: $A = \{0, 1\}$.
3. The set of transition functions $\delta$ contains the functions:
   (a) For the state $q_0$: $\delta_{0,0} : (q_0, 0) \to q_0$, $\delta_{0,1} : (q_0, 1) \to q_1$.
   (b) For the state $q_1$: $\delta_{1,0} : (q_1, 0) \to q_2$, $\delta_{0,1} : (q_1, 1) \to q_1$.
   (c) For the state $q_2$: $\delta_{2,0} : (q_2, 0) \to q_1$, $\delta_{2,1} : (q_2, 1) \to q_1$.
4. The starting state is $q_0$.
5. The accept state set is: $F = \{q_1\}$.

Then the diagram will be:




**Remark 2.2.6.** Unless explicitly stated otherwise, we assume that all finite automata under consideration have been *normalized*, meaning that all inaccessible states have been removed and all dead states have been combined into a single failure state. A finite state automaton modified in this way is referred to as a *normalized finite state automaton.*

We now state some classical results of regular languages and finite state automata, which will be useful later.

**Theorem 2.2.7** (Reversal is Regular)**.** *Let $A$ be an alphabet, suppose $L$ is a regular language over $A$. Then the language consisting of the strings of $L$ written in the reverse order is also a regular language over $A$.*

*Proof.* Let $M$ be a non-deterministic finite state automaton that accepts the language $L$. We construct a new automaton $M'$ by reversing the direction of all transitions in $M$, designating the original accepting states as a single initial state in $M'$, and taking the original initial state of $M$ as the sole accepting state in $M'$. The resulting automaton $M'$ accepts the reverse of $L$. Hence, the class of regular languages is closed under reversal. □

**Definition 2.2.8** (Prefix Closure)**.** If $L$ is a language over an alphabet $A$, the *prefix closure* of $L$ is the set of all prefixes of strings in $L$. A *prefix-closed* language is one that equals its prefix closure.

**Theorem 2.2.9** (Prefixes)**.** *Let $L$ be a regular language. Then the prefix closure of $L$ and the largest prefix-closed sublanguage of $L$ are regular languages.*

*Proof.* Let $M$ be a deterministic finite state automaton accepting the regular language $L$. If we turn every non-failure state of the machine $M$ into an accept state, the language accepted by the new automaton will be the prefix closure of $L$. If instead of changing all non-failure states, we remove all of the non-accept together with all the arrows pointing to and from such states from the machine $M$, we get an automaton that accepts the largest prefix-closed subset of $L$. □

**Lemma 2.2.10** (Effect of Regularity of Map)**.** *Let $A$ and $B$ be alphabets, $L_A$ be a regular language over $A$, and $L_B$ be a regular language over $B$. Then for any map $f : A \to B$ or $f : A \to B^*$, the image $f(L_A)$ and the inverse image $f^{-1}(L_B)$ are regular languages over $B$ and $A$ respectively.*

More generally, if $f$ is a map from an alphabet $A$ to the set of regular expressions over another alphabet $B$, we say that $f$ is a *substitution*. One can verify that the set of all languages over $A$ forms a semigroup under concatenation, and the set of regular languages over $A$ forms a subsemigroup. In this context, we define the image $f(L_A)$ of a language $L_A \subseteq A^*$ as follows: we first extend $f$ to a semigroup homomorphism from $A^*$ to the semigroup of regular languages over $B$, and then define

$$f(L_A) \coloneqq \bigcup_{w \in L_A} f(w)$$

**Corollary 2.2.11.** *Let $A$ and $B$ be alphabets, for any substitution $f$ between $A$ and $B$, the image $f(L_A)$ is a regular language over $B$.*

We now introduce the operators of first-order predicate calculus on the class of regular languages, with the goal of showing that, when regular languages are represented via finite state automata, these logical operators correspond to concrete operations on the automata.

Firstly we have to define the notion of *predicate*:

**Definition 2.2.12** (Predicate)**.** Let $A$ be an alphabet, a *predicate* $P$ over $A$ is a boolean-valued function on $A^*$.

Note that there is a one-to-one correspondence between the predicates and the languages over a given alphabet $A$, in which a predicate $P : A^* \to \{0, 1\}$ corresponds to the language $P^{-1}(1)$. So naturally we have the definition of *regular predicates*:

**Definition 2.2.13** (Regular Predicate)**.** Let $A$ be an alphabet and let $P$ be a predicate over $A$. We say $P$ is *predicate* if the corresponding language of it is regular.

The operators of first-order predicate calculus are logical $\neg$, $\wedge$, $\vee$, $\exists$, and $\forall$. We can define $\neg$, $\wedge$, $\vee$ to languages via the following way, let $A$ be an alphabet, and let $L_1$, $L_2$ be languages over $A$, then:

1. We realize $\neg L_1$ by $A^* \setminus L_1$.
2. We realize $L_1 \wedge L_2$ by $L_1 \cap L_2$.
3. We realize $L_1 \vee L_2$ by $L_1 \cup L_2$.

Furthermore, the three newly realized languages are actually regular:

**Lemma 2.2.14.** *Let $A$ be an alphabet and let $L_1$, $L_2$ be regular languages over $A$. Then $A^* \setminus L_1$, $L_1 \cap L_2$, and $L_1 \cup L_2$ are regular languages over $A$.*

To realize the two remaining logical operators $\exists$ and $\forall$, we require some additional setup. Firstly we have to define the notion for languages with many variables.

**Definition 2.2.15** (Many-Variable Language)**.** Let $A_1, ..., A_n$ be alphabets, then the *language over* $(A_1, ..., A_n)$ is a set of $n$-tuples of strings $(w_1, ..., w_n)$, where $w_i \in A_i^*$ for each $i$. A language over an $n$-tuple of alphabets is called an *$n$-variable language*.

Based on such many-variable language definition, we can extend our predicate's definition to be:

**Definition 2.2.16** (Predicate of Multi-Variable Language)**.** Let $A_1, ..., A_n$ be alphabets, then a *predicate over* $(A_1, ..., A_n)$ is a boolean valued function on $A_1^* \times \cdots \times A_n^*$.

Now we can realize $\exists$ and $\forall$, we quantify over the last variable of the $n$-tuples in $(A_1, ..., A_n)$:

1. The language $\exists(L)$ is the language over $(A_1, ..., A_{n-1})$ consisting of $n-1$-tuples $(a_1, ..., a_{n-1})$ such that $(a_1, ..., a_{n-1}, a_n) \in L$ for some $a_n \in A_n$.

2. Similarly, the language $\forall(L)$ is the language over $(A_1, ..., A_{n-1})$ consisting of $n-1$-tuples $(a_1, ..., a_{n-1})$ such that $(a_1, ..., a_{n-1}, a_n) \in L$ for all $a_n \in A_n$.

Recall the padded alphabet we defined in Definition 2.1.11, for any language $L$ over $(A_1, ..., A_n)$, the *padded extension* $L^{\$}$ is of $L$ is a padded language over the corresponding padded alphabet $B$. To be more specific, for each $n$-tuple $(w_1, ..., w_n) \in L$, let $m = \max\{|w_1|, ..., |w_n|\}$, and we pad each $w_i$ with $\$_i$'s at the end to make sure that each entry of the padded $n$-tuple $(w_1', ..., w_n') \in L^{\$}$ is of length $m$.

Now, we are ready to perform predicate calculus on multi-variable regular languages:

**Theorem 2.2.17** (Predicate Calculus)**.** *Let $A_1, ..., A_n$ be alphabets, and let $L_1$, $L_2$ be regular languages over $(A_1, ..., A_n)$. Then the following hold:*

1. *The languages $\neg L_1$, $L_1 \wedge L_2$, $L_1 \vee L_2$ are regular languages over $(A_1, ..., A_n)$.*

2. *The languages $\exists(L_1)$ and $\forall(L_1)$ are regular languages over $(A_1, ..., A_{n-1})$.*

3. *For any alphabet $A_{n+1}$, the language*

$$\{(w_1, ..., w_n, w_{n+1}) \mid (w_1, ..., w_n) \in L_1\}$$

*is a regular language over $(A_1, ..., A_{n+1})$.*

4. *For any permutation* $\sigma \in S_n$ *of* $\{1, ..., n\}$*, the language*

$$L_\sigma \coloneqq \{(w_1, ..., w_n) \mid (w_{\sigma(1)}, ..., w_{\sigma(n)}) \in L\}$$

*is a regular language over* $(A_1, ..., A_{n+1})$*.*

**Corollary 2.2.18** (Predicates Closed)**.** *The class of regular predicates is closed under the operators* $\neg$*,* $\wedge$*,* $\vee$*,* $\exists$*,* $\forall$*,* $\Rightarrow$*,* $\Leftrightarrow$*.*

We conclude this introductory chapter with a central theorem in the theory of regular languages—a fundamental tool for determining whether a given language is regular, the proof can be found in [Sip96].

**Theorem 2.2.19** (Pumping Lemma)**.** *Let* $A$ *be an alphabet, suppose* $L$ *is a regular language over* $A$*. There exists a number* $p$ *such that if* $s$ *is any word in* $L$ *of length* $|s| \geq p$*, then it can be divided into three pieces* $s = xyz$ *and the following hold:*

1. *For each number* $i \geq 0$*,* $xy^iz \in L$*.*
2. *The length of the subword* $y$ *should satisfy* $|y| > 0$*.*
3. *The length of the subword* $xy$ *should satisfy* $|xy| \leq p$*.*

# Chapter 3

# Geometric Group Theory Background

## 3.1 Treating Groups as Languages

**Definition 3.1.1** (Generating Set of a Group)**.** $A$ is a *generating set* for a group $G$ if and only if for all $g \in G$, it can be written as a factorization

$$g = a_1^{\pm 1} \cdots a_k^{\pm 1} \tag{3.1}$$

for some $a_1, ..., a_k \in A$. Such $a_1^{\pm 1} \cdots a_k^{\pm 1}$ is called a *word*.

**Definition 3.1.2** (Word Length in Group)**.** The *word length* of $G$ with respect to the generating set $A$, denoted by $\ell(g)$ is the smallest $k$ for which such factorization (3.1) exists.

Since we are using machines, especially finite state automata to study group theoretic problems, it is crucial to be able to speak of groups as languages, that is we have to consider the relationship between groups with generators and languages over alphabets.

**Definition 3.1.3** (Group Language Representation)**.** Let $G$ be a group and let $A \subset G$ be a finite set of elements of $G$, We interpret concatenation in language as multiplication in the group $G$, we define a semigroup homomorphism $\pi : A^* \to G$. If $w$ is a string over $A$, we say that $\pi(w)$ is the element of $G$ *represented by* $w$, denoted by $\overline{w}$.

**Definition 3.1.4** (Semigroup Generator)**.** If the semigroup homomorphism $\pi$ defined above is surjective, which means every element of $G$ is represented by one or more string over the set $A$, we say that $A$ is a set of *semigroup generators* for $G$, or equivalently we say that $A$

*generates $G$ as a semigroup*. At the same time, we say $A$ *generates $G$ as a group* if $A \cup A^{-1}$ generates $G$ as a semigroup.

**Example 3.1.5.** For example, the set $\{1\} \subset \mathbb{Z}$ generates $\mathbb{Z}$ as a group, not a semigroup. But $\{2, -3\} \subset \mathbb{Z}$ generates $\mathbb{Z}$ as a semigroup, not a group.

Now, consider a group $G$ and an alphabet $A$, we provide a more general definition regarding the generators:

**Definition 3.1.6** (Generators)**.** Let $G$ be a group, $A$ be an alphabet, and $p : A \to G$ be a map that need not be injective. We extend $p$ to a semigroup homomorphism $\pi : A^* \to G$. If this ho morphism is surjective, which is equivalent to saying that $\pi(A)$ *generates $G$ as a semigroup*, we say that $A$ is a *set of semigroup generators for $G$*. If $\pi(A)$ *generates $G$ as a group*, we say that $A$ is a *set of group generators for $G$*.

**Remark 3.1.7.** If the alphabet $A$ has a total order, then we talk about an *ordered set* of semigroup or group generators. Notation wise, if $x \in A$ and $g \in G$, generally we write $gx$ instead of $gp(x)$, $x^{-1}g$ instead of $(p(x))^{-1}g$. Note that if we adjoin the *end-of-string symbol* \$ to the alphabet $A$, then we can also extend the homomorphism $p$ so that $p(\$)$ is the identity element in $G$.

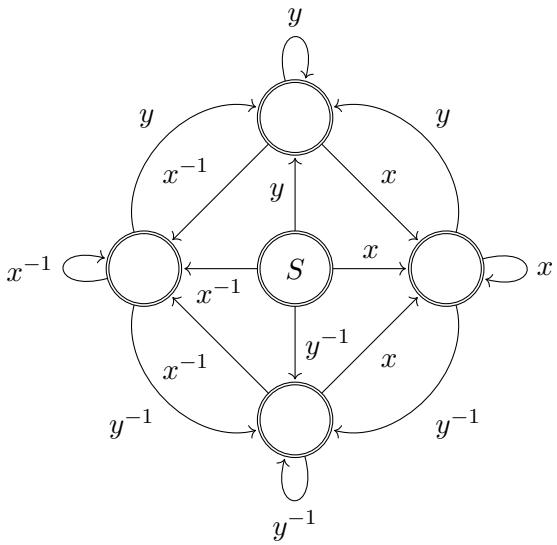
From now on, by the abuse of notation, by a *set of semigroup or group generators*, we really mean a set together with the map $p : A \to G$. Given an alphabet $A$, with the set $A^*$ of all the strings over it, we care about some given language $L \subseteq A^*$, together with the restricted map $\pi|_L : A^* \to G$, again by the abuse of notation, we write it as $\pi : L \to G$. The case that we are most interested in is when $\pi$ is surjective, when every group element $g \in G$ can be represented by at least one string in the language $L$ over $A$. Among these desirable situations, there is one that is the best, that is when $\pi : L \to G$ is one-to-one. In this case we say that the language $L$ has the *uniqueness property*. Even though the situation is so tantalizing, a lot of languages associated to groups that we work on will not have such property.

**Remark 3.1.8.** In order to be easier to be studies with groups, we often want the alphabet $A$ of semigroup generators to be *closed under inversion*, theoretically which means that

there exists an involution $\iota : A \to A$ such that $p(\iota(x)) = p(x)^{-1}$ for all generators $x \in A$. We call $\iota(x)$ the *formal inverse* of generator $x \in A$, denoted $x^{-1}$.

Now we are ready to see our first familiar example, which illustrates the idea of speaking of groups in languages.

**Example 3.1.9** (Free Group on Two Generators)**.** Let $F_2$ be the free group on two generators $x$ and $y$, we choose the alphabet $A = \{x, y, x^{-1}, y^{-1}\}$, which is closed under inversion. We let $L \subseteq A^*$ be the language of all reduced strings. Note that this language is regular, as it can be accepted by the following finite state automaton. Furthermore, the map $\pi : L \to G$ is a bijection.




**Example 3.1.10** (Free Abelian Group on Three Generators)**.** Let $G_3$ be the free abelian group on three group generators $x, y, z$. We have the alphabet $A = \{x, y, z, x^{-1}, y^{-1}, z^{-1}\}$ and consider the regular language $L$ defined by the regular expression:

$$(x^* \vee (x^{-1})^*)(y^* \vee (y^{-1})^*)(z^* \vee (z^{-1})^*)$$

the map $\pi : L \to G$ will be bijective.

## 3.2 Group Presentations, the Dehn Function, and the Word Problem

There has long been a deep connection between formal language theory and group theory. One of the earliest and most significant is a decision problem known as the *word problem*

*for groups*. Before stating this fundamental question, we first introduce one of the most important notational tools in geometric group theory: the notion of a *group presentation*:

**Definition 3.2.1** (Group Presentation)**.** A *group presentation* is a pair $(A, R)$, where $A$ is finite and $R \subseteq F(A)$ which is the free group on $A$. We say $(A, R)$ is *finite* if $R$ is also finite. The group defined by a presentation $(A, R)$ is denoted by $\langle A|R\rangle = F(A)/N$, where $N = \langle\langle R\rangle\rangle$, means the *normal closure* of $R$ in $F(A)$, that is the smallest normal subgroup of $F(A)$ that contains $R$. Elements of $R$ are called the *defining relators*, while the elements of $N$ are called the *relators*.

**Remark 3.2.2.** Note that $R$ is possibly infinite. We are interested in the case that a group $G = \langle A|R\rangle$ is finitely presented.

**Proposition 3.2.3.** *Let $G$ be a finitely presented group with a presentation $\langle A|R\rangle$. Then any word $w \in F(A)$ is equivalent to the identity if and only if it can be written in the form:*

$$w = \prod_{i=1}^{n} v_i r_i^{\pm 1} v_i^{-1} \tag{3.2}$$

*with $r_i \in R$ and $v_i \in F(A)$ for all $i$.*

Then we can state the word problem for groups:

**Definition 3.2.4** (Word Problem for Groups)**.** Let $G = \langle A|R\rangle$ be a finitely presented group. The *word problem* for $G$ is the decision problem of determining, given a word $w \in F(A)$, whether there exists an algorithm that decides whether $w$ represents the identity element in $G$.

To make the connection between group theory and formal language theory more concrete, we introduce the following definition and result:

**Definition 3.2.5** (Regularly Generated Group)**.** Let $G = \langle A|R\rangle$ be a finitely presented group and let $L$ be a regular language over $A$. Then we say $G$ is *regularly generated* if the map $\pi : L \to G$ is surjective and the inverse image $\pi^{-1}(id_G) \subseteq L$ is also regular.

**Theorem 3.2.6** (Regularly Generated)**.** *Let $G$ be a finitely presented group. Suppose $G$ is regularly generated, then $G$ has a solvable word problem.*

*Proof.* Let $G = \langle A|R\rangle$ be a finitely presented and regularly generated group, and let $w$ be a word over the generating set $A$. We want to determine whether $\overline{w}$ represents the identity element in $G$. By the definition of being regularly generated, there exists a regular language $L \subseteq A^*$ such that:

1. Every element of $G$ is represented by at least one string in $L$.

2. The subset $L_0 = \{u \in L \mid \overline{u} = \mathrm{id}_G\}$ is also regular.

Now let $\pi : F(A) \to G$ be the natural projection from the free group $F(A)$ onto $G$, and let $K = Ker(\pi)$ be its kernel. Then $K$ consists of precisely those reduced words in $F(A)$ that represent the identity in $G$. Since $G$ is finitely presented, $K$ is recursively enumerable: we can systematically list all finite products of relators (and their conjugates/inverses) in the normal closure of $R$ to generate every element of $K$.

We must decide whether $w \in K$. Although $K$ might not be a decidable set, it is recursively enumerable. We consider the set

$$A' \coloneqq \pi^{-1}(\overline{w}) = \{u \in F(A) \mid \pi(u) = \overline{w}\}$$

Each word $u \in A'$ differs from $w$ by a finite product of conjugates of the defining relators in $R$ or their inverses; equivalently, $uw^{-1} \in K$. Since $K$ is recursively enumerable and $R$ is finite, we can enumerate all such finite insertions of subwords representing the identity in all possible positions of $w$. This process systematically lists every $u \in A'$. Thus $A'$ is also recursively enumerable.

Now we use the fact that $L$ is regular, so $L$ is at least recursively enumerable. Hence their intersection $A' \cap L$ is recursively enumerable because the intersection of two recursively enumerable sets is recursively enumerable. Moreover, because $G$ is regularly generated, $A' \cap L$ is nonempty: there must be some word $w' \in L$ with $\overline{w'} = \overline{w}$. An algorithm that enumerates elements of $A' \cap L$ will eventually produce such a $w'$. At that point, we can check whether $w' \in L_0$, i.e. if $\overline{w'} = id_G$, by a simple membership test in the regular language $L_0$. This completes the decision procedure, proving that $G$ has a solvable word problem. □

This naturally leads to a more general question: how can we determine whether the

word problem is solvable for a given finitely presented group $G$? To address this question, we require additional setup and background.

**Definition 3.2.7** (Combinatorial Area)**.** Given a group presentation $G = \langle A|R\rangle$, let $N$ be the normal closure of $R$ in $F(A)$. Then the *combinatorial area* for $w \in N$ is the smallest $n$ for which the factorization (3.2) exists.

Now we introduce a geometric notion: the Van Kampen diagrams. What's the point of studying such diagrams? Van Kampen diagrams provide a geometric way to illustrate and construct factorization of the form (3.2)

**Definition 3.2.8** (Van Kampen Diagram)**.** Let $A$ be a finite alphabet and let $\Omega$ be a finite connected plane graph whose edges are oriented and labeled by elements of $A$. We call $\Omega$ a *Van Kampen diagram.*

**Definition 3.2.9** (Cell in Van Kampen Diagram)**.** A bounded component $C$ of $\mathbb{R}^2\backslash\Omega$ is called a *cell.*

**Definition 3.2.10** (Boundary Label)**.** Given a cell $C$, let $e_1, ..., e_k$ be edges listed as we traverse along $\partial C$ in some direction starting from some vertex, then we define the *boundary label* $L(\partial C)$ of the cell $C$ as follows:

$$L(\partial C) = \prod_{i=1}^{k} L(e_i)^{\epsilon_i}$$

where

$$\epsilon_i = \begin{cases} 1 & \text{if } e_i \text{ is traversed in positive direction} \\ -1 & \text{if } e_i \text{ is traversed in negative direction} \end{cases}$$

**Remark 3.2.11.** We should not think of $L(\partial C)$ as an element of $F(A)$, but as an element of $W(A)$, the set of all finite words over the alphabet $A \cup A^{-1}$, not necessarily reduced.

**Definition 3.2.12** (Boundary of Van Kampen Diagram)**.** By abuse of notation, we define the *boundary of the Van Kampen Diagram* by $\partial\Omega = \partial U$, where $U$ is the unique unbounded connected component of $\mathbb{R}^2\backslash\Omega$. We also define the *Van Kampen Diagram boundary label* $L(\partial\Omega)$ in the same way.

With all these settings, we are ready to define the Van Kampen diagram over a group presentation $G = \langle A|R\rangle$:

**Definition 3.2.13** (Van Kampen Diagram over Group Presentation)**.** Given a group presentation $G = \langle A|R\rangle$, we will say that $\Omega$ is a *Van Kampen diagram over the presentation* $\langle A|R\rangle$ if for any cell $C$ of $\Omega$, $L(\partial F) \in R$ up to inverses or cyclic shifts.

**Definition 3.2.14** (Disk Diagram)**.** A Van Kampen Diagram $\Omega$ is called a *disk-diagram* if $\mathbb{R}^2\backslash U$ is homeomorphic to a closed disk.

**Theorem 3.2.15** (Van Kampen Lemma)**.** *Let $G = \langle A|R\rangle$ be a finitely generated group and let $w \in F(A)$. Then $\overline{w} = id_G$ if and only if there exists a Van Kampen diagram $\Omega$ over $\langle A|R\rangle$ such that its boundary label satisfies $L(\partial\Omega) = w$.*

*Proof.* Suppose $w \in F(A)$ and $\overline{w} = id_G$. We want to show that there exists a Van Kampen diagram $\Omega$ such that $L(\partial\Omega) = w$ in $W(A)$ and $Area(\Omega) \leq Area(w)$. Let $n = Area(w)$, so by definition we will have $w = \prod_{i=1}^{n} v_i r^{\pm 1} v_i^{-1}$ (Form (3.2)), note that this is an equality in $F(A)$. For each $i$, let $\Omega_i$ be the lollipop diagram. Now let $\Omega$ be the union of all $\Omega_i$ with based points identified. Then we have $Area(\Omega) = n$ and $L(\partial\Omega) = \prod_{i=1}^{n} v_i r_i^{\pm 1} v_i^{-1}$. Now it remains to do the cancellations in $\partial\Omega$ to make $L(\partial\Omega)$ reduced and thus equal to $w$ in $F(A)$.

Conversely, suppose that $\overline{w} = L(\partial\Omega)$ for some $\Omega$, the Van Kampen diagram over $\langle A, R\rangle$. We want to show $\overline{w} = id_G$ and $Area(w) \leq Area(\Omega)$. We'll do the case when $\Omega$ is a disk diagram. We proceed by induction on $Area(\Omega)$. Firstly we claim that there exists a cell $C$ such that $\partial C \cap \partial\Omega$ is non-empty and connected. We fix such cell $C$ and let $z = L(\partial C \cap \partial\Omega)$. Then if $\partial C \cap \partial\Omega$ is traversed clockwise, we will have $L(\partial C) = zv$ for some $v \in W(A)$ and $L(\partial\Omega) = zu = w$ for some $u \in W(A)$. Let $D$ be the original Van Kampen diagram: $D = \mathbb{R}^2\backslash U$ for $U$ to be the unique unbounded component of $\mathbb{R}^2 \setminus \Omega$. Then $D \setminus C$ is also a disk Van Kampen diagram with $Area(D \setminus C) = Area(D) - 1$ and $\ell(\partial(D\backslash C)) = v^{-1}u$. By the induction hypothesis, we have $\overline{v^{-1}u} = id_G$ and $Area(v^{-1}u) \leq Area(D) - 1$. Now we have $w = zu = zv(zv)^{-1}zu = zv \cdot (v^{-1}u)$. Note that $zv$ is a conjugate of an element of $R \cup R^{-1}$, hence $\overline{zv} = id_G$, therefore $\overline{w} = \overline{(zv) \cdot v^{-1}u} = id_G$. Also we have

$$Area(w) \leq Area(zv) + Area(v^{-1}u) \leq 1 + (Area(D) - 1) = Area(D)$$

Note that same strategy also applies to the general case of $\Omega$. $\square$

**Definition 3.2.16** (Dehn Function)**.** Let $G = \langle A|R\rangle$ be finitely presented. The *Dehn function* of the presentation $\langle A|R\rangle$ is a function $\delta_{\langle A|R\rangle} : \mathbb{N} \to \mathbb{Z}_{\geq 0}$ defined as follows:

$$\delta_{\langle A|R\rangle}(n) \coloneqq \max\{Area(w) \mid \overline{w} = id_G \text{ and } \ell(w) \leq n\}$$

Note that by the Van Kampen Lemma (Theorem 3.2.15), $Area(w)$ is also equal to

$$Area(w) = \min\{Area(\Omega) \mid \Omega \text{ is a Van Kampen Diagram with } L(\partial\Omega) = w\}$$

**Definition 3.2.17** (Type of Van Kampen Diagram)**.** A Van Kampen diagram $\Omega$ has *type* $(k, n)$ if $\Omega$ has $k$ cells and $\ell(L(\partial\Omega)) = n$.

**Proposition 3.2.18.** *Let $G = \langle A|R\rangle$ be a fixed finitely presented group and let $k, n \in \mathbb{N}$ be two fixed natural numbers. Then exists only finitely many Van Kampen diagrams of type $(k, n)$ over the fixed finite group presentation $\langle A|R\rangle$. Moreover, there exists an algorithm listing all such diagrams for all $k, n$.*

**Definition 3.2.19** (Isoperimetric Inequality)**.** If the group $G$ has a finite presentation which in turn has Dehn function that is bounded above by a function that is linear, quadratic, etc., we say that $G$ satisfies a *linear, quadratic, etc., isoperimetric inequality.*

**Remark 3.2.20** (Group Invariance of Isoperimetric Inequality)**.** Although the Dehn function depends on the group presentation, if $\phi_1$ and $\phi_2$ are Dehn functions for different presentations of the same group $G$, it is not hard to verify that given any $n \in \mathbb{N}$, the following inequality holds:

$$\phi_2(n) \leq N_1\phi(N_2 n)$$

where $N_1$ is the maximum area of an old relator in terms of new ones, while $N_2$ is the maximum length of a new generator in terms of old ones. This inequality tells us that the isoperimetric inequality is group-invariant.

**Theorem 3.2.21** (Solvable Word Problem for Group)**.** *Let $G = \langle A|R\rangle$ be a finitely presented group and let $\delta_{\langle A|R\rangle}$ be the Dehn function of this presentation. Then the followings are equivalent:*

1. *The Dehn function $\delta_{\langle A|R\rangle}$ is a recursive function.*

2. *There exists a recursive function $f$ such that $\delta(n) \leq f(n)$ for all $n \in \mathbb{N}$.*

3. *The group $G$ has a solvable word problem.*

*Proof.* Note that the implication (1) $\Rightarrow$ (2) is clear. For (2) $\Rightarrow$ (3), we let $w \in F(A)$ and we need to determine whether $\overline{w} = id_G$ or not. We let $n = \ell(w)$, recall that $\delta(n) \leq f(n)$ for all $n \in \mathbb{N}$ for some recursive function $f$. Hence there exists $0 \leq k = \delta(n) \leq f(n)$ and a Van Kampen diagram $\Omega$ over $\langle A|R\rangle$ of type $(k, n)$ such that $L(\partial\Omega) = w$. So we have the algorithm:

1. Compute $f(n)$.

2. List all Van Kampen diagrams $\Omega$ of type $(k, n)$ with $0 \leq k = \delta(n) \leq f(n)$. If one of these diagrams has $L(\partial\Omega) = w$, then $\overline{w} = id_G$, otherwise we will have $\overline{w} \neq id_G$.

Finally, for (3) $\Rightarrow$ (1), suppose the group $G$ has a solvable word problem. Then for ann $n \in \mathbb{N}$, we need to compute $\delta(n)$. Firstly we can compute $W_n \coloneqq \{w \in F(A) \mid \ell(w) \leq n \text{ and } \overline{w} = id_G\}$ since $G$ has a solvable word problem. Then we compute $\delta(n)$ as follows: for any $w \in W_n$, there exists some Van Kampen diagram $\Omega$ with $L(\partial\Omega) = w$. Since for any $k \in \mathbb{N}$, there exists finitely many diagrams of type $(k, n)$, such diagrams can be algorithmically enumerated. We can just list all such diagrams for $k = 1$, $k = 2$, and so on. Until we find one with $L(\partial\Omega) = w$, now we set $k(w)$ to be the smallest $k$ that works for $w$. By setting $\delta_{\langle A|R\rangle}(n) = \max\{k(w) \mid w \in W_n\}$, we are done. □

## 3.3 Cayley Graphs and Hyperbolic Groups

Now, given a finitely presented group $G = \langle A|R\rangle$, the choice of a set of generators $A$ for a group $G$ gives us a notion of distance in $G$, two distinct elements are at distance one if they can be obtained from one another by right multiplication of a generator. By connecting such neighboring elements with an edge labeled with the corresponding generator, we obtain the *Cayley graph* of the group $G$ with respect to the generator set $A$:

**Definition 3.3.1** (Cayley Graph)**.** Let $G$ be a group and let $A \subseteq G$ be a generating set for $G$, not necessarily symmetric and not necessarily excluding the identity. The *Cayley graph* $\Gamma(G, A)$ is the directed, edge-labeled graph defined as follows:

1. The vertex set of $\Gamma(G, A)$ is $G$, each group element $g \in G$ is a vertex in the graph.
2. For each $g \in G$ and $a \in A$, there is a directed edge from $g$ to $ga$, labeled by $a \in A$.

If $A$ is symmetric, then $\Gamma(G, A)$ can be viewed as an undirected graph.

**Remark 3.3.2.** Throughout, we assume that the generating set $S$ is symmetric, meaning that if $s \in S$, then $s^{-1} \in S$ as well. This ensures that the corresponding Cayley graph is undirected.

Recall that we mentioned before that two distinct elements are at distance one if they can be obtained from one another by right multiplication of a generator, this gives us the *word metric* on the Cayley graph $\Gamma(G, A)$, denoted by $d_A$. Clearly, the distance between two elements $g_1, g_2 \in G$ in the group metric is $\ell(g_1^{-1} g_2)$, which is the word length of $g_1^{-1} g_2$.

**Definition 3.3.3** (Path in Cayley Graph)**.** Given a word $w$ over a set of semigroup generators $A$, we define a *path* $\widehat{w} : [0, \infty) \to \Gamma(G, A)$ in the Cayley graph as: if $t \in [0, \infty)$ is an integer, the value $\widehat{w}(t) \coloneqq \overline{w(t)}$ is the image of the prefix of $w$ of length $t$ in the group $G$. Then if $t$ is not an integer, we move along the respective edges with unit speed, thus $\widehat{w}$ travels at unit speed for parameter values in the interval $[0, |w|]$, then stops.

We also introduce additional metrics that will be used throughout this thesis:

**Definition 3.3.4** (Uniform Metric)**.** Let $X$ be any set and $Y$ be a metric space with metric $d_Y$, the *uniform metric* on the set of maps from $X$ to $Y$ is the metric defined by

$$d_{unif}(f, g) \coloneqq \sup_{x \in X} d_Y(f(x), g(x))$$

where $f, g : X \to Y$ are maps from $X$ to $Y$.

**Remark 3.3.5.** Note that such uniform metric might be infinite if $Y$ is not a bounded metric space.

**Definition 3.3.6** (Closed $\epsilon$-Neighborhood)**.** Let $(X, d)$ be a metric space and let $A$ be a subset of a metric space $(X, d)$. For $\epsilon \in \mathbb{R}_{\geq 0}$, we define:

$$A^{+\epsilon} \coloneqq \{x \in X \mid d(x, a) \leq \epsilon \text{ for some } a \in A\}$$

to be the *closed $\epsilon$-neighborhood* of $A$.

**Definition 3.3.7** (Hausdorff Distance)**.** Let $(X, d)$ be a metric space and let $A, B$ be subsets of $X$. Then we define the *Hausdorff distance* between $A$ and $B$ to be:

$$dist(A, B) \coloneqq \inf\{\epsilon \geq 0 \mid B \subseteq A^{+\epsilon} \text{ and } A \subseteq B^{+\epsilon}\}$$

We conclude this brief review of the fundamentals of geometric group theory by introducing the notion of *word hyperbolic groups*, which serves as one of the main motivations for studying the central concept of this thesis: automatic groups. As a key class of examples of automatic groups, hyperbolic groups play a crucial role in modern research in geometric group theory.

To state the precise definition of a word hyperbolic group, we must first introduce the notions of a *geodesic space* and a *hyperbolic geodesic space*.

Let $(X, d)$ be a metric space, and consider a continuous path $p : [a, b] \to X$. The *length* of the path $p$, denoted $length(p)$, is defined by

$$length(p) \coloneqq \sup_{a \leq t_0 \leq t_1 \leq \cdots \leq t_n = b} \sum_{i=0}^{n-1} d(p(t_i), p(t_{i+1}))$$

Based on such definition, we can define the notion of a *geodesic* in a metric space:

**Definition 3.3.8** (Geodesic)**.** Let $(X, d)$ be a metric space, and let $p : [a, b] \to X$ be a path. Then the path $p$ is called a *geodesic* if its length $length(p)$ is equal to the distance between the two endpoints:

$$length(p) = d(p(a), p(b))$$

Thus, we can upgrade the normal metric space to a *geodesic metric space*:

**Definition 3.3.9** (Geodesic Metric Space)**.** A metric space $(X, d)$ is a *geodesic metric space* if for all $x, y \in X$, there exists a geodesic path $p$ from $x$ to $y$. If $(X, d)$ is geodesic and $x, y \in X$, by $[x, y]$ we will denote a chosen geodesic from $x$ to $y$.

**Remark 3.3.10.** Note that a geodesic path is not necessarily unique.

We are now only a few steps away from defining the notion of a word hyperbolic group, but a few more preparatory definitions are still required.

**Definition 3.3.11** (Geodesic Triangle)**.** Let $(X, d)$ be a geodesic metric space, and let $x, y, z \in X$. A *geodesic triangle with vertices* $x, y, z$ is the union of any geodesics $[x, y]$, $[x, z]$, and $[y, z]$. This triangle will be denoted by $T = [x, y, z]$.

Based on such notion of geodesic triangle, we can define one condition for a geodesic metric space to have hyperbolicity:

**Definition 3.3.12** ($\delta$-Slim Condition)**.** Let $(X, d)$ be a geodesic metric space, and let $T = [x, y, z]$ be a geodesic triangle in $X$. We say that $T$ is *$\delta$-slim* if for every point $p$ on one side of the triangle, say $e_1$, there exists a point $q$ on the union of the other two sides $e_2 \cup e_3$ such that

$$d(p, q) \leq \delta.$$

We say that the geodesic space $X$ satisfies the $\mathrm{Hyp}_S(\delta)$ *condition* if every geodesic triangle in $X$ is $\delta$-slim.

There exist several equivalent formulations of hyperbolicity in geodesic spaces, including the $\mathrm{Hyp}_T(\delta)$ condition for $\delta$-thin triangles, the $\mathrm{Hyp}_I(\delta)$ condition for $\delta$-insize (or $\delta$-small), and the $\mathrm{Hyp}_G(\delta)$ condition based on Gromov's four-point criterion. These definitions are equivalent up to scaling of the constant $\delta$. However, introducing and verifying these equivalences requires additional technical development that lies beyond the central scope of this thesis. For this reason, we adopt the $\delta$-slim triangle condition as our working definition of hyperbolicity:

**Definition 3.3.13** (Hyperbolic Geodesic Metric Space)**.** Let $(X, d)$ be a geodesic metric space. We say that $X$ is *hyperbolic* if it satisfies the $\mathrm{Hyp}_S(\delta)$ condition for some $\delta \geq 0$.

Finally, we can state the precise definition of *word hyperbolic group*:

**Definition 3.3.14** (Word Hyperbolic Group)**.** Let $G$ be a finitely generated group and let $A$ be a finite generating set of $G$. Then $G$ is *word hyperbolic* if its Cayley graph $\Gamma(G, A)$ is a hyperbolic geodesic metric space.

The definition of a word hyperbolic group is independent of the choice of finite generating set $A$, though this is not immediately obvious. This invariance can be established using the notion of *quasi-isometries*. While a full treatment of hyperbolic metric spaces and hyperbolic groups lies beyond the scope of this thesis, we introduce several key notions below that will be used in the subsequent exposition.

Firstly we see the notion of *quasi-geodesics*, which will be revisited in Section 5.1.

**Definition 3.3.15** (Pseudomap)**.** Let $(X, d_X)$ and $(Y, d_Y)$ be metric spaces, a *pseudomap* is a relation $f : X \to Y$ such that for every $x \in X$, there exists $y \in Y$ that is $f$-related to $x$, namely $y = f(x)$.

A pseudomap can be thought of as a generalized function that may assign multiple values to a single input. Although this notion is somewhat imprecise, it provides a useful foundation for defining more refined concepts. We now introduce a sequence of definitions, each building on the last, which will ultimately lead to the notion of *quasi-geodesics*.

**Definition 3.3.16** ($(k, \epsilon)$-Pseudomap)**.** Let $k \geq 1$ and $\epsilon \geq 0$ be real numbers, then a *$(k, \epsilon)$-pseudomap* $f : X \to Y$ is a pseudomap such that

$$d_Y(f(x_1), f(x_2)) \leq k \cdot d_X(x_1, x_2) + \epsilon$$

for all $x_1, x_2 \in X$ and their corresponding choices $f(x_1), f(x_2) \in Y$.

If the $(k, \epsilon)$-pseudomap satisfies one further condition, it will leads us to the $(k, \epsilon)$-*pseudoisometric embedding*:

**Definition 3.3.17** ($(k, \epsilon)$-Pseudoisometric Embedding)**.** Let $k \geq 1$ and $\epsilon \geq 0$ be real numbers, then a *$(k, \epsilon)$-pseudoisometric embedding* $f$ is a $(k, \epsilon)$-pseudomap $f$ such that

$$d_X(x_1, x_2) \leq k \cdot d_Y(f(x_1), f(x_2)) + \epsilon$$

for all $x_1, x_2 \in X$.

Finally, we can define the *$(k, \epsilon)$-quasi-geodesic*:

**Definition 3.3.18** ($(k, \epsilon)$-Quasi-Geodesic)**.** Let $k \geq 1$ and $\epsilon \geq 0$ be real numbers, then a $(k, \epsilon)$-*quasi-geodesic* in the metric $Y$ is a $(k, \epsilon)$-pseudoisometric embedding of an interval in $Y$.

**Remark 3.3.19.** While some mathematicians define a $(k, \epsilon)$-quasi-geodesic as a function from an interval into a metric space satisfying coarse distance bounds, our definition via pseudoisometric embeddings allows greater generality, such as set-valued maps or partial correspondences. When the pseudomap is single-valued and defined on the entire interval, the two notions are equivalent up to reparameterization.

**Definition 3.3.20** (Geodesic String)**.** Let $G$ be a group with a set of semigroup generators $A$. A string $w \in A^*$ is called a *geodesic* if it has minimal length among all strings representing the element $\overline{w} \in G$.

**Definition 3.3.21** (**ShortLex**-Geodesic String)**.** Let $G$ be a group with a set of semigroup generators $A$. A string $w \in A^*$ is called a ***ShortLex**-geodesic* if it is the minimum with respect to the **ShortLex** order among all strings representing the element $\overline{w} \in G$.

Given the definition of geodesic string, we have the following result involving the uniform metric and Hausdorff distance we defined earlier:

**Lemma 3.3.22** (Hausdorff Implies Uniform)**.** *Let $G$ be a group with a finite set $A$ of semigroup generators that is closed under inversion, and let $L$ be a prefix-closed regular language over $A$. Suppose that for any two geodesic strings $u, v \in L$ whose images under $\pi : A^* \to G$ have distance at most 1, there exists a constant $k > 1$ such that the Hausdorff distance between the corresponding paths $\widehat{u}$ and $\widehat{v}$ in the Cayley graph $\Gamma(G, A)$ is bounded by $k$. Then, for any two elements $\alpha, \beta \in L$ giving rise to geodesics $\widehat{\alpha}$ and $\widehat{\beta} \in \Gamma(G, A)$, both starting at the basepoint and ending at a distance of at most one apart, the Hausdorff distance between $\widehat{\alpha}$ and $\widehat{\beta}$ is uniformly bounded by $2k$.*

*Proof.* Let $|\alpha| \geq t \geq 0$, by hypothesis, there exists an $s$ with $|\beta| \geq s \geq 0$ such that $d(\widehat{\alpha}(t), \widehat{\beta}(s)) \leq k$. Now we claim $|s - t| \leq k$, otherwise either $\widehat{\alpha}$ or $\widehat{\beta}$ will fail to be geodesic. Hence we conclude that for any $t$, we will have $d(\widehat{\alpha}(t), \widehat{\beta}(t)) \leq 2k$. □

# Chapter 4

# Automatic Group Theory

Throughout this thesis, we restrict our attention to *synchronous* automatic groups. Although *asynchronous* automatic groups are also of significant interest and value, they fall outside the scope of our present discussion. In synchronous automatic groups, the two strings are read at the same speed by the multiplier automata (which will be defined later). In contrast, for asynchronous ones, the two strings may be read at very different speeds.

## 4.1 Automatic Groups: Basic Definitions and Properties

String length is a fundamental concept in computation theory. It allows us to estimate the computational cost of running a program on a given input. Sometimes, we want to estimate how long it takes to rewrite a string and how the length of the resulting string compares to the original.

In this section, we are working with group presentations and finite state automata to define the notion of *automatic groups*, the main theme of this thesis. Recall from Proposition 3.2.3 that for a finitely presented group $G = \langle A|R\rangle$, any word $w \in F(A)$ represents the identity element in $G$ if and only if it can be written in the conjugate product form (3.2). Based on this decomposition, we introduced the notions of *combinatorial area* (Definition 3.2.7) and the *Van Kampen diagram* (Definition 3.2.8).

It turns out that when a reduced word representing the identity is written in the conjugate product form (3.2), one can often rewrite this product into a longer one while maintaining control over its structure. Estimating the cost of such rewriting processes requires additional tools. To that end, we briefly introduce the notion of a *dual Van Kampen dia-*

*gram*, which plays a central role in the analysis that follows.

Briefly speaking, given a finitely presented group $G = \langle A|R\rangle$ and a reduced word $w \in F(A)$ that represents the identity in $G$ and is expressed in the conjugate product form (3.2), we can begin constructing a lollipop diagram. This diagram is drawn by placing the conjugates of the relators $v_i r_i^{\pm 1} v_i^{-1}$ around a central basepoint (representing the identity element), arranged clockwise.

Each conjugate $v_i r_i^{\pm 1} v_i^{-1}$ can be visualized as a path: starting at the basepoint, one follows a directed path labeled by $v_i$, which ends at a vertex where a loop labeled by the relator $r_i^{\pm 1}$ is attached. After traversing the loop, one returns to the basepoint by retracing the inverse path $v_i^{-1}$. In this way, the entire term $v_i r_i^{\pm 1} v_i^{-1}$ is represented geometrically as a "lollipop" consisting of a stick (the path) and a head (the loop).

Starting from the lollipop diagram, we construct the *dual Van Kampen diagram* by folding together adjacent edges that have the same label but opposite orientations. In the setting of a free group, this corresponds to pairing off adjacent edges labeled by inverse generators. In the diagram, each such pairing is indicated by an arc connecting the midpoints of the corresponding edges, visually representing their cancellation.

After performing a finite number of such pairings and replacements, we enclose the entire diagram within a large outer circle. The remaining unmatched edges of the lollipop diagram are then connected to the boundary circle, preserving both label and orientation. The resulting structure is the *dual Van Kampen diagram.*

This diagram consists of several regions or "holes", whose number is equal to the combinatorial area of the word $w$. The diagram also contains internal edges connecting the holes to one another, as well as edges connecting the holes to the boundary circle. These structural features will play a key role in the estimates and arguments that follow. The detailed construction and its viewpoint from differential topology can be found on [ECH+92, Page 42].

**Lemma 4.1.1** (Bounding Lengths)**.** *Let $G = \langle A|R\rangle$ be a finitely presented group and let $w \in F(A)$ be a reduced word that is equivalent to the identity element of the group. Suppose $w$ is expressed as a product of $n$ conjugates of relators of the form (3.2). Let $k$ be the*

*maximal length of a relator $r_i \in R$. Then we can rewrite the product so that each $v_i$ has length at most $(|w| + 2k)2^n$.*

*Proof.* Given a word $w$ which is equivalent to the identity element of the group $G$ and is written in the conjugate form (3.2), we can construct a dual Van Kampen diagram as described above. Since the decomposition of the conjugate form is assumed to have $n$ terms, the dual Van Kampen diagram will have $n$ holes. We let $p$ be the basepoint of the word $w$ in the diagram. We start by cutting the diagram along all arcs that begin or end at a hole. After such cutting, we will have multiple connected components, and we pay special attention to the connected component $P$ that contains $p$. Now, since we have cut out all the arcs originated from or ended at the holes, the only remaining arcs are those running from the boundary of the diagram to itself note that there are at most $\frac{1}{2}|w|$ of them. Now, we want to connect the basepoint $p$ to the basepoint of one of the loops in the diagram by a path $u$, in order to construct such path we have to cross at most $\frac{1}{2}|w|$ of such arcs, and it is also possible that once we touch the hole we want to connect, we have to travel additional $\frac{1}{2}|k|$ distance to get to the basepoint of the hole, thus we have to travel the length at most $\frac{1}{2}(|w| + k)$ along the constructed path $u$.

Now, we cut the diagram open along the constructed path $u$, which will give us a diagram with one fewer holes. Note that the new boundary of the diagram will be $w' = wuru^{-1}$, assuming $u$ is connecting the basepoint of the original $w$ to the basepoint of the relator $r$. By assumption $r$ has length at most $k$, and from the computation above we know that $|w'|$ is bounded by $|w| + \frac{1}{2}(|w| + k) + k + \frac{1}{2}(|w| + k) = 2(|w| + k)$. Observe that $w'$ now can be written as the product of $n - 1$ conjugates of relators and we have $w = w'ur^{-1}u^{-1}$, by continuing such fashion, we can show by induction on $n$ such that the upper bound $(|w| + 2k)2^n$ follows. □

**Remark 4.1.2.** The bound from lemma 4.1.1 can be used to show the bounds of the recursive function $f(n)$ used in Theorem 3.2.21.

**Definition 4.1.3** (Automatic Group)**.** Let $G$ be a group and $A$ be a semigroup generator of $G$. We define the *word acceptor* $W$, which is a finite state automaton over $A$, that satisfies: any group element $g \in G$ can be represented as a string in the language $L(W)$. For any

$x \in A \cup \{\varepsilon\}$, we define the *multiplier automaton* $M_x$ such that $(w_1, w_2) \in L(M_x)$ if and only if $w_1, w_2 \in L(W)$ and $\overline{w_1 x} = \overline{w_2}$. Then an *automatic structure on* $G$ consists of the data $(A, W, L, M_x)$, and the group $G$ is an *automatic group* if it admits an automatic structure. For $\varepsilon$, we call $M_\varepsilon$ the *equality recognizer*.

Note that an automatic structure is sometimes called an *automation*.

**Remark 4.1.4.** Although the full notation of an automatic structure of a finitely presented group $G = \langle A|R\rangle$ should be $(A, W, L, M_x)$, sometimes by the abuse of notation we simply write $(A, W)$ or $(A, M_x)$ to show the information for certain usage. We will also use the notation $(A, L)$, where $L$ is a language from the finite state automaton $W$, to represent such automatic structure.

In some cases, the language $L$ associated with an automatic structure of a group $G$ can be large and structurally intricate. The following corollary establishes that it is sufficient to analyze certain restricted languages instead.

**Proposition 4.1.5** (Restriction is Automatic)**.** *Let $(A, L)$ be an automatic structure for a group $G$. Suppose $L' \subset L$ is a regular language over the alphabet $A$ that maps onto $G$. Then $(A, L')$ also gives us an automatic structure for the group $G$.*

In an automatic structure, the accepted pairs $(w_1, w_2)$ in the automaton $M_x$ encode controlled relationships between elements of the group. This not only provides a computational framework for understanding word representations but also imposes geometric constraints on the word metric. In particular, words related by the same automaton exhibit a form of metric control, meaning their corresponding elements in the group remain within a uniformly bounded distance. Intuitively, this ensures that words $w_1$ and $w_2$ recognized by $M_x$ cannot drift too far apart in the Cayley graph. The following lemma formalizes this property by establishing an explicit bound.

**Lemma 4.1.6** (Lipschitz Property)**.** *Let $G$ be an automatic group with automatic structure $(A, M_x)$. Then there exists a constant $k$ such that if $(w_1, w_2)$ is accepted by $M_x$ for one $x \in A \cup \{\varepsilon\}$, the distance between $\overline{w_1(t)}$ and $\overline{w_2(t)}$ in the word metric will be bounded by such $k$ for any integer $t \geq 0$.*

*Proof.* Without loss of generality we assume all automata are normalized, and let the constant $c$ be the maximum number of states in the automata $M_x$ for any $x \in A \cup \{\varepsilon\}$. Now given any integer $t \geq 0$, suppose the multiplier automaton $M_x$ has read the prefixes $w_1(t)$ of $w_1$ and $w_2(t)$ of $w_2$, and the automaton is now in the state $S$. We consider the shortest path representation $(u_1 \times u_2) \in A^* \times A^*$ of arrows to an accept state of $M_x$, then we know that both $u_1$ and $u_2$ should have length less than $c$, and the pair $(w_1(t)u_1, w_2(t)u_2)$ should be accepted by the automaton $M_x$. By acceptance we get $\overline{w_1(t)u_1x} = \overline{w_2(t)u_2}$, and the distance between $\overline{w_1(t)}$ and $\overline{w_2(t)}$ in the Cayley graph of the group $G$ is at most $2c - 1$, hence such Lipschitz constant $k$ exists. □

**Remark 4.1.7.** The constant $k$ depends on the automatic structure, thus it is called a *Lipschitz constant for the automatic structure*, the name of such property comes from analysis. By the definition of uniform metric (Definition 3.3.4), the Lipschitz property tells us that if $\overline{w_1}$ and $\overline{w_2}$ are at a distance at most one from each other, the paths $\widehat{w_1}$ and $\widehat{w_2}$ are at distance at most $k$ from each other in the uniform metric on paths in the Cayley graph.

**Lemma 4.1.8** (Bounded Length Difference)**.** *Let $G$ be an automatic group with an automatic structure $(A, L)$. Then there exists a constant $N$ with the property that for any accepted word $w \in L$ and for any $g \in G$ of distance at most 1 from $\overline{w}$, the following hold:*

1. *The vertex $g$ has some representative of length at most $|w| + N$ in $L$.*
2. *If some representative of $g$ in $L$ has length greater than $|w| + N$, the number of representatives of $g$ in $L$ will be infinite.*

*Proof.* Firstly let the constant $N$ be greater than the number of states in any of the automatic structure's automata, let $w \in L$ be accepted and $g \in G$ be a vertex in the Cayley graph with distance at most one from $\overline{w}$ as stated in the assumption, we let $w'$ be any representative of $g$ in the language $L$, by assumption, one of $(w, w')$ or $(w', w)$ should be accepted by some multiplier automaton $M_x$ for some $x \in A \cup \{\varepsilon\}$. Now note that if $|w'| > |w| + N$, then after reading all of $w$, $M_x$ will undergo more than $N$ transitions, recall that $N$ is greater than the number of states in any of the automata, which indicates $M_x$ should visit the same state twice. Now we consider two operations: firstly, if we eliminate the loop

between two visits to the repeated state, we can shorten $w'$ but still get an accepted word that represents the same group element $g$, thus 1 follows. Then, if we take arbitrarily more traverse around the loop, we will lengthen $w'$ and still get an accepted word that represents $g$, hence 2 follows. □

**Theorem 4.1.9** (Quadratic Isoperimetric Inequality)**.** *Let $G$ be an automatic group. Then the followings hold:*

1. *The group $G = \langle A|R\rangle$ is finitely presented.*
2. *The group $G$ satisfies a quadratic isoperimetric inequality.*

*Proof.* We let $L$ be the language of accepted words of an automatic structure of the group $G$ over $A$. For any word $w$, the prefixes $w(t)$ have representatives $u_t \in L$ of length at most $N|w| + n_0$ with constants $N$ and $n_0$ given by Lemma 4.1.8. Now for each fixed $t$, the representatives $\overline{u_t}$ and $\overline{u_{t+1}}$ are one generator apart, denoted by $x_t$. We know that the uniform distance $\widehat{u_t}$ and $\widehat{u_{t+1}}$ is bounded by the Lipschitz constant $k$ of $L$ by Lipschitz property (Lemma 4.1.6), so the loop $u_t x_t u_{t+1}^{-1}$ can be decomposed into at most $N|w| + n_0$ loops of length at most $2k + 2$. If $w$ is a loop, then it can be decomposed into at most $N|w|^2 + n_0|w|$ loops of length at most $2k + 2$ as well. Now we can get a finite set of relators for $G$ by taking all loops up to the length of $2k + 2$, and we see that in terms of such group presentation $\langle A|R\rangle$, $G$ satisfies a quadratic isoperimetric inequality. □

In other words, any word $w$ representing the identity can be written in the form (3.2) with $n = O(|w|^2)$ time complexity. By Remark 3.2.20, the following result holds:

**Corollary 4.1.10.** *Let $G$ be an automatic group, then it satisfies a quadratic isoperimetric inequality with respect to any group presentation.*

**Theorem 4.1.11** (Quadratic Algorithm)**.** *Let $G$ be an automatic group and $(A, L)$ an automatic structure for $G$. Then for any word $w$ over $A$, we can find a string in $L$ representing the same element of $G$ as $w$ in time proportional to $|w|^2$.*

*Proof.* Without loss of generality we assume the multiplier automata $M_x$ for any $x \in A$ are normalized. Suppose we have an accepted string $u$ and a generator $x$, the problem is we

want to find a representative $v$ of $\overline{ux}$ that can be accepted. We let $S_0$ be the set containing only the initial state $s_0$ of $M_x$. Then for $i > 0$, we inductively define $T_i$ as the set of arrows with source in $S_{i-1}$ together with label $(x_i, y_i)$. For $x_i$, if $i \leq |u|$, it equals the $i$th character in $u$, and of $i > |u|$, it simply equals \$, and $y_i \in A \cup \{\$\}$ is arbitrary. We define $S_i$ as the set of targets of arrows in $T_i$. We define the number $n$ to be the smallest number such that $n \geq |u|$ and $S_n$ includes an accept state of $M_x$. Now we have the labels $y_1, ..., y_n$, by reading off these labels we will get a path of arrows joining the initial state to an accept state, forming the string $v' = y_1 \cdots y_n$ representing $\overline{ux}$. Finally we get $v$ by simply discarding the trailing \$'s.

Observe that the time taken by each step in such inductive construction is guaranteed to be bounded by a constant, because there are only finitely many arrows and states. Thus the overall time is proportional to the number $n$ defined above, from Lemma 4.1.8 we know that such $n$ can only exceed $|u|$ by a bounded amount $N$, otherwise it will no longer be minimal. This implies that given an accepted string $u$, we can find a representative for $\overline{ux}$ in $O(|u|)$ time, and the found representative's length is bounded from above by $|u| + N$. Now if we replace the pair $(x_i, y_i)$ by the reserved one in the previous construction, we will have similar estimates when we multiply $u$ by $x^{-1}$.

By repeating this process, we can find a representative of $\overline{w}$ of length at most $N|w|+n_{id}$, note that $n_{id}$ is the length of a representative of the identity. The time takes is calculated by:

$$O(\sum_{i=1}^{|w|}(N \cdot i + n_{id})) = O(|w|^2)$$

Finally, we note that the entire process is constructive, we can find an accepted representative of the identity element from the automatic structure. We start with arbitrary accepted string $x_1 \cdots x_n$ and we use the procedure above to multiply successively by $x_n^{-1}, ..., x_1^{-1}$. □

**Corollary 4.1.12.** *Let $G$ be an automatic group and let $(A, L)$ be an automatic structure for $G$. Suppose we know some word $e$ over $A$ representing the identity element of the group $G$. Then we can solve the word problem for the group $G$ in quadratic time.*

*Proof.* For a desired word $w$, we can solve the word problem by finding a representative in $L$ for this desired word, then feeding it and $e$ together to the equality recognizer $M_\varepsilon$. □

Sometimes instead of studying the identity directly, we can study the neighborhood around the identity, and such intuition gives us the definition of so-called *standard automata*, as defined below.

**Definition 4.1.13** (Standard Automata)**.** Let $G$ be a group and let $A$ be a set of semigroup generators of $G$. Let $W$ be a deterministic finite state automaton over $A$ and let $\mathcal{N}$ be a fixed finite neighborhood of the identity in $G$. Suppose $\mathcal{N}$ contains $A$, then for each $x \in A \cup \{\varepsilon\}$, we define a finite state automaton $M_x$ over $(A, A)$, called the *standard automaton $M_x$ based on* $(W, \mathcal{N})$, constructed precisely as the following:

Let $W'$ be a finite state automaton over $A \cup \{\$\}$ with the accepted language $L(W)\$^*$, and let $S$ be the state set of $W'$. The automaton $M_x$ has $S \times S \times \mathcal{N}$ as its set of states. The initial state is $(s_0, s_0, e)$, where $s_0$ is the initial state of $W'$ and $e$ is the identity of $G$, We treat all states of the form $(s_1, s_2, g)$ as a single failure state, where $s_1$ or $s_2$ is a failure state of $W'$. For the transition function, if $M_x$ is in the state $(s_1, s_2, g)$ and it reads a letter $(y_1, y_2) \in B$ for the padded alphabet $B$ associated to $(A, A)$, it goes to the state $(s_1 y_1, s_2 y_2, h)$, where $h = y_1^{-1} g y_2$. But if $h \notin \mathcal{N}$, $M_x$ will directly go to the failure state. The accept states of $M_x$ are the states of the form $(s_1, s_2, x)$, where $s_1$ and $s_2$ are accept states of $W'$.

**Remark 4.1.14.** The basic idea for such standard automata is that $M_x$ will keep track on the action of the deterministic finite state automaton $W$ on the input strings and the word difference between the prefixes so far.

**Example 4.1.15.** For example, we can take the set of elements of $G$ with word length at most $k$ as a fixed finite neighborhood $\mathcal{N}$ of the identity in $G$.

Now, by Theorem 4.1.11, if a group $G$ has an automatic structure, we can solve the word problem and thus know the neighborhood $\mathcal{N}$. However, we often want to build the standard automata without knowing any prior knowledge of an automatic structure of $G$, instead we want to find one. In this situation, sometimes we can obtain information from other sources about what a neighborhood $\mathcal{N}$ of the identity looks like, then this gives us the possibility to bootstrap the procedure from such restricted but useful knowledge, as indicated by the following *standard automata theorem*:

**Theorem 4.1.16** (Standard Automata Theorem)**.** *Let $G$ be an automatic group with automatic structure $(A, W)$. Suppose $\mathcal{N}$ is a sufficiently large but finite neighborhood of the identity. Then the standard automata based on $(W, \mathcal{N})$, together with the alphabet $A$ will also give an automatic structure of the group $G$.*

*Proof.* By the Lipschitz property (Lemma 4.1.6), there is a constant $k > 0$ such that if $(w_1, w_2)$ is accepted by any of the original automata, the uniform distance between the paths $\widehat{w_1}$ and $\widehat{w_2} : [0, \infty) \to \Gamma$ should be less than $k$. We let the neighborhood $\mathcal{N}$ contain the ball of radius $k$ about the identity element in the Cayley graph, and we let $M_x$ be the corresponding standard automata for $x \in A \cup \{\varepsilon\}$. Now it suffices to check the definition of being an automatic structure (Definition 4.1.3). Note that the first condition is trivially satisfied, while the second condition is satisfied by the following: the forward direction is trivial to check. Conversely, we let $(w_1, w_2)$ be a string over $(A, A)$, if $w_1$ and $w_2$ are accepted by $W$ and $\overline{w_1 x} = \overline{w_2}$ for some $x \in A$, the pair $(w_1, w_2)$ will be accepted by the original multiplier automaton, and therefore the distance between $w_1(t)$ and $w_2(t)$ in the Cayley graph should never exceed the chosen fixed value $k$. By definition of the neighborhood $\mathcal{N}$, this implies that $\overline{w_1(t)}^{-1}\overline{w_2(t)} \in \mathcal{N}$ for all $t \geq 0$, thus $(w_1, w_2)$ is accepted by $M_x$, meaning $(w_1, w_2) \in L(M_x)$. □

Now, with the existence of standard automata, we have the following characterization of an automatic structure:

**Theorem 4.1.17** (Characterizing Synchronous)**.** *Let $G$ be a group with a finite semigroup generator $A$ and let $W$ be a finite state automaton over $A$. Suppose the map $\pi : L(W) \to G$ is surjective. Then $A$ and $W$ are part of an automatic structure on $G$ if and only if there exists a number $k$ with the following property: for any two strings $w_1$ and $w_2$ accepted by $W$ which are one generator apart, the corresponding paths $\widehat{w_1}$ and $\widehat{w_2}$ are less than $k$ apart with respect to the uniform distance.*

With this theorem in mind, we can see some examples of automatic groups. Recall that the groups in Example 3.1.9 and 3.1.10 we have seen before are clearly automatic.

**Example 4.1.18** (Finite Group)**.** Any finite group $G$ is automatic. We take the alphabet $A$ equal to $G$ and for the word acceptor $W$, we simply take an automaton which accepts any string over $A$.

**Remark 4.1.19.** The example of finite group tells us that given an automatic structure, it is possible that the map $\pi : L(W) \to G$ is far from being bijective.

**Open Question 4.1.20** (Isomorphism Problem for Automatic Group)**.** Is the isomorphism problem solvable for automatic groups? In other words, is there an algorithm that takes two automatic groups as inputs and decides whether they are isomorphic or not? The current conjecture is that there is no such algorithm, how to prove the non-existence?

Next, we aim to show that the property of being automatic is independent of the choice of generators. This independence is a fundamental feature that distinguishes automaticity from many earlier approaches to combinatorial group theory, where the specific presentation of a group plays a crucial role.

**Theorem 4.1.21** (Automaticity is Generator Invariant)**.** *Let $G$ be a finitely presented group with an automatic structure $(A_1, W_1)$, and let $A_2$ be another finite set of semigroup generators of $G$. Then there exists another finite state automaton $W_2$ over $A_2$ such that $(A_2, W_2)$ also gives an automatic structure on $G$.*

*Proof.* We first claim that adding or removing a generator equal to the identity element does not affect the automatic structure of the group. Let $A_2 = A_1 \cup \{e\}$, by abuse of notation we mean $e$ maps to the identity of the group $G$. We can simply take $W_2 = W_1$, except we change the underlying alphabet from $A_1$ to $A_2$. Since the word metric does not change, by Theorem 4.1.17 we know that $(A_2, W_2)$ is also an automatic structure on $G$. Now we have proved the case of adding, we also want to show the case of deleting: we let $A_1 = A_2 \cup \{e\}$. If $A_2$ is empty, we know that $A_1 = \{e\}$, indicating that $G$ is trivial. By Theorem 4.1.17, the language $\{\varepsilon\}$ over the empty alphabet gives an automatic structure on $G$ trivially.

Next, we consider the situation when $A_2$ is nonempty. Note that when $e$ appears in a string in the language $L(W_1)$, we cannot simply delete it to obtain a corresponding string in $L(W_2)$, this is because Theorem 4.1.17 requires that a pair $\widehat{w_1}, \widehat{w_2}$ of accepted strings should

be uniformly near to each other when their endpoints are within distance 1 in the Cayley graph. Deletion of $e$'s in the string might cause these paths to stop keeping pace with each other. Instead of simply deleting all $e$, we find a string $z$ over $A_2$ with the properties that it has nonzero length $|z| = m$ and it also maps to the identity. We then define the new language $L_2$ of accepted strings by the following way: for each accepted string of $L(W_1)$, we replace every $m$-th occurrence of $e$ by $z$, then we delete all the remaining $e$ in the string.

Now we want to show the newly constructed $L_2$ is regular, we prove by constructing a generalized non-deterministic finite state automaton over $A_2$ that accepts $L_2$, recall that non-deterministic finite state automaton is equivalent to deterministic finite state automaton by Theorem 2.2.4. A state of this automaton is of the form $(s, i)$, where $s$ is a state of the automaton $W_1$ and $i$ is an integer such that $0 \leq i < m$, which is an indicator of how much the modified string is out of the phase with respect to the original one. The start state will be $(s_0, 0)$, where $s_0$ is the start state of the automaton $W_1$, and the accept states are $(s, i)$, where $s$ is an accept of $W_1$. For the transition function, each arrow $(s, x, t)$ of $W_1$ gives rise to $m$ arrows in the new automaton by the following way: if $x \neq e$, the new arrows are $((s, i), x, (t, i))$ for $0 \leq i < m$. If $x = e$, the new arrows will be $((s, 0), z, (t, m-1))$ and $((s, i), \varepsilon, (t, i-1))$ for $1 \leq i < m$. We see that a string in $L_2$ that leaves the new automaton in the accept state $(s, i)$ is $i$ characters longer than some corresponding string accepted by $W_1$, and they represent the same element in $G$. Note that $i$ is bounded by $m$, thus such automaton is valid by Theorem 4.1.17.

Now we proved simple cases, we want to tackle the case when $A_2$ is arbitrary. Without loss of generality we assume $A_1$ and $A_2$ contain an identity element. We choose a number $c > 0$ such that any element of $A_1$ can be written as a string of length of at most $c$ over $A_2$, and a number $c'$ such that any element of $A_2$ can be written as a string of length at most $c'$ over $A_1$. Such $c$ and $c'$ exist because both $A_1$ and $A_2$ are sets of generators for $G$. For each $x \in A_1$, we take a string $u_x \in A_2$ of length $c$ representing $x$, we use the trivial generator to pad the length if necessary. From this we define the new language $L(W_2)$ by replacing $x$ by $u_x$ in the strings of $L(W_1)$. Firstly note that by Lemma 2.2.10 this language is regular. It is clear that the map from $L(W_2)$ to $G$ is surjective, we let the number $k$ given by applying Theorem 4.1.17 to the automatic structure $(A_1, L_1)$. Finally, we claim that Theorem 4.1.17

is satisfied for the automatic structure $(A_2, L_2)$ with the bound $c + kcc'$.

To see why this claim holds, recall that we have two sets of generators: $A_1$ and $A_2$. They give rise to corresponding Cayley graphs $\Gamma_1$ and $\Gamma_2$. Consider two accepted strings in $\Gamma_2$ representing elements $u$ and $v$ in $G$ such that $u$ and $v$ differ by right multiplication by a generator $x \in A_2$. By substitution of strings over $A_1$, we get these two strings. We let $x$ and $y$ be paths in $\Gamma_2$ corresponding to prefixes of equal length, we want to show that their distance in $\Gamma_2$ should be bounded. We start by finding $c$ and $d$ in $\Gamma_1$ also corresponding to prefixes of equal length, and the distances $d(x, c), d(y, d)$ are at most $c/2$. We also express the generator $x \in A_2$ as a product of at most $c'$ generators in $A_1$ by assumption, here suppose we have $x = yz$ for generators $y, z \in A_1$.

Next we bridge the gap between $u$ and $v$ in $\Gamma_1$ by a sequence of at most $c' - 1$ adjacent points, using the expression of $x$ in terms of generators in $A_1$. We draw auxiliary paths from the basepoint of these adjacent points, and mark off points corresponding to the prefixes of the same length as $c$ and $d$. By assumption the distance in $\Gamma_1$ between each of these auxiliary points and the next is bounded by $k$, so we know that the distance in $\Gamma_1$ between $c$ and $d$ is at most $kc'$. Thus the distance between $c$ and $d$ in $\Gamma_2$ is bounded by $kcc'$. Hence, by adding all the bounds, we get the bound $c + kcc'$, which concludes the proof.

□

By Theorem 4.1.21, we observe that while a finitely presented group $G$ may admit an automatic structure that is well-behaved with respect to one generating set, it may also possess a more complex and less practical automatic structure under a different choice of generators. The theorem assures us that we can work with the more convenient structure without loss of generality, as the property of being automatic is invariant under a change of generators.

## 4.2 Automatic Groups: Further Properties and Variants

Recalling Example 4.1.18 along with Remark 4.1.19, we observe that the language of accepted strings in the automaton for a finite group may provide little to no meaningful information about the group itself. In the following, we examine several interesting proper-

ties that such languages may exhibit. The first generally desirable property is the following:

**Definition 4.2.1** (Uniqueness Property of Group Language)**.** The language of accepted strings of an automatic group is said to satisfy the *uniqueness property* if each group element has a unique representative in the language.

Next we want to combine the notion of geodesic string (Definition 3.3.20) and automatic group. We see an example here first, which is a generalization of Example 3.1.9.

**Example 4.2.2.** Example 3.1.9 can be generalized to any group $G$ with any set $A$ of semigroup generators. We set $L$ equal to the set of strings over $A$ which are shortest representatives for elements of $G$. Then the map $\pi : L \to G$ is usually not bijective, but it is certainly surjective and finite-to-one because for any $g \in G$, all elements of $\pi^{-1}(g) \cap L$ must have the same length, and therefore they are finite in number.

From this example we see that the languages of all geodesic strings maps finite-to-one onto $G$, but in general such language does not have to be regular, even not to be recursively enumerable.

**Definition 4.2.3** (Geodesically Automatic Group)**.** Let $G$ be a group with a set $A$ of semigroup generators. Then $G$ is *strongly geodesically automatic* if the language of all geodesics over $A$ forms the whole language part of an automatic structure for $G$. On the other hand, if there exists some language consisting only of geodesics that is part of the language of an automatic structure for $G$, we say that $G$ is *weakly geodesically automatic*.

In other words, a group is said to be *weakly geodesically automatic* if it admits an automatic structure in which every accepted string is geodesic. In contrast, a group is *strongly geodesically automatic* if the language of all geodesics forms the language of an automatic structure for the group.

It should be noted that, in general, the language of all geodesics is not even a regular language. If it were, the group would have a solvable word problem by Theorem 3.2.6.

In fact, the concept of automatic groups originated from Cannon's study of the languages of geodesics in discrete groups of hyperbolic transformations [Can84]. In his paper,

Cannon examined a particularly well-behaved class of geodesic languages: they remain regular regardless of the choice of generators. He implicitly provided a criterion for a group to satisfy this property. Here, we present a more general criterion due to Epstein.

**Theorem 4.2.4** (Geodesic Automaton)**.** *Let $G$ be a group with a finite set $A$ of semigroup generators that is closed under inversion. Let $L$ be a prefix-closed regular language over $A$ and let $L_0$ be the language of geodesic strings in $L$, suppose $L_0$ maps onto $G$ under $\pi : A^* \to G$. Then for any two geodesic strings $u, v \in L$ whose images under $\pi$ are of distance at most one from each other, if there exists a number $k > 1$ such that the Hausdorff distance between the paths $\widehat{u}$ and $\widehat{v}$ in the Cayley graph $\Gamma(G, A)$ is bounded by such $k$, the pair $(A, L_0)$ will give an automatic structure for $G$.*

*Proof.* Note that it suffices to show $L_0$ is a regular language by Theorem 4.1.17, we have to ensure that each prefix is a geodesic. Let $V$ be a finite state automaton accepting $L$, and let $N$ be a ball of radius $2k$ in $G$ around the identity element. We want to construct the standard automata $M_x$ for all $x \in A \cup \{\varepsilon\}$ based on $(V, N)$. Firstly we claim the following language $L_x$ is regular by Corollary 2.2.18:

$$L_x \coloneqq \{w \in L \mid (w, w') \in L(M_x) \Rightarrow |w| < |w'|\}$$

Likewise, the language $L' \coloneqq \left(\bigcup_{x \in A} L_x x \cup \{\varepsilon\}\right) \cap L$ is also regular. We let $L''$ be the largest prefix-closed subset of $L'$, note that $L''$ is regular by Theorem 2.2.9. Now, we claim that $L'' = L_0$.

Since $L$ is prefix-closed, the nullstring $\varepsilon$ is geodesic and is indeed contained in $L$, therefore $\varepsilon \in L_0$ and $\varepsilon \in L''$. Now we take any nontrivial string $w \in L''$ and suppose $w = ux$ for some $x \in A$. We want to show $w$ is a geodesic. Let $v \in L$ be a geodesic string with $\overline{v} = \overline{ux}$, by Lemma 3.3.22, $v$ and $u$ are uniformly bounded by $2k$ apart. Since $v$ and $u$ are both in $L$, $(u, v)$ will be accepted by $M_x$. By the construction of $L''$, we have $ux \in L'$ and $u \in L_x$, which implies that $|v| > |u|$ strictly. Thus $w = ux$ is also a geodesic, hence $w \in L_0$, this concludes $L'' \subseteq L_0$.

Conversely we take any $w \in L_0$, we want to show $w \in L''$. We claim that any prefix $u$ of $w$ is in $L'$, and we show this by induction on the length of $|w|$. The base case of

the null prefix is trivial, we assume $u = vx$ for some $x \in A$. Since $L$ is prefix-closed by assumption, we have $v \in L$. We have to show $v \in L_x$, which is equivalent to show for any $(v, v') \in L(M_x)$, $|v'| > |v|$. This follows by the fact that $u$ is a geodesic string and $\overline{u} = \overline{v'} \in G$, hence $|v'| \geq |u| > |v|$. Hence $L_0 \subseteq L''$ and we conclude the proof. □

The requirement that an automatic structure $(A, L)$ be strongly geodesic is indeed quite restrictive. If the generating set $A$ is closed under inversion, then the automatic structure is said to be *symmetric*, meaning that $L = L^{-1}$. A weaker notion of symmetry is introduced in the following definition:

**Definition 4.2.5** (Biautomatic Group)**.** Let $G$ be an automatic group with automatic structure $(A, L)$ and assume that $A$ is closed under inversion. Then the automatic structure $(A, L)$ is *biautomatic* if $(A, L^{-1})$ also gives an automatic structure for $G$, and the group $G$ is called a *biautomatic group* if it admits a biautomatic structure.

Similar to Theorem 4.1.17, which is used to characterize the automatic structure, we also want to have a way to characterize biautomatic structure:

**Theorem 4.2.6** (Characterizing Biautomatic)**.** *Let $G$ be a group and let $A$ be a finite set of generators closed under inversion. Let $(A, L)$ be an automatic structure such that $L$ maps onto $G$. Then $(A, L)$ is biautomatic if and only if for each $w \in L$ and each pair of generators $x, y \in A$, the uniform distance between the path $\widehat{xwy}$ and the path corresponding to any element of $L$ representing $\overline{xwy}$ is bounded by a fixed constant $k > 1$.*

*Proof.* The language $L^{-1}$ is regular by Theorem 2.2.7, thus the result follows simply from Theorem 4.1.17. □

As one may have noticed, this result is indeed very similar to Theorem 4.1.17, with the key difference that we additionally allow multiplication by generators on both the left and the right. This result also implies that any automatic structure on an abelian group is biautomatic. However, the reverse direction is still an open question:

**Open Question 4.2.7.** Does there exist an automatic group that does not admit any biautomatic structure?

As one might expect, biautomaticity is also independent of the choice of generators. We state this result here without proof, as its justification closely parallels that of Theorem 4.1.21.

**Proposition 4.2.8.** *The property of a group being biautomatic is independent of the choice of generators.*

As a group theoretic problem, the *conjugacy problem* is also heavily studied as the isomorphism problem. It is still unknown that whether the conjugacy problem is solvable for automatic groups. But a certain conclusion can be provided for biautomatic groups.

**Definition 4.2.9** (Conjugacy Problem)**.** Suppose we are given a group $G$ with a finite generating set and two elements $g, h \in G$. The *conjugacy problem* asks whether there exists an element $x \in G$ such that $xgx^{-1} = h$. A group $G$ is said to have a *solvable conjugacy problem* if there exists an algorithm that, given any two elements $g, h \in G$, decides whether they are conjugate.

**Theorem 4.2.10** (Biautomatic Implies Solvable Conjugacy Problem)**.** *If a group $G$ has a biautomatic structure, then it has a solvable conjugacy problem.*

*Proof.* Let $A$ be the set of generators and let $L$ be the language of accepted words over $A$. Given a word $p$ over $A$, we claim that we can construct a finite state automaton $M_p$ such that a pair of strings $(u, v)$ over $A$ is accepted if and only if $\overline{up} = \overline{v}$ and $u, v \in L$. We proceed by induction, if $p = p'x$ with $x \in A$, then the following two languages are equal to each other:

$$\{(u, v) \mid u \in L \wedge v \in L \wedge \overline{up} = \overline{v}\}$$

and

$$\{(u, v) \mid (\exists w)((u, w) \in L(M_{p'}) \wedge (w, v) \in L(M_x))\}$$

where $M_x$ is a multiplier automaton and we assume that $M_{p'}$ has already been constructed. By predicate calculus (Theorem 2.2.17), the latter language is regular, thus the induction holds. Note that $G$ is biautomatic, we can also construct in a similar way a finite state automaton $M^q$ that accepts a pair $(u, v)$ if and only if $u, v \in L$ and $\overline{u} = \overline{qv}$. Combining the

two finite state automata $M_p$ and $M_q$, we get a finite state automaton $M_p^q$ that accepts a pair $(u, v)$ if and only if $u, v \in L$ and $\overline{up} = \overline{qv}$. Hence we see that the conjugacy problem is solved by simply asking whether the finite state automaton $M_p^q$ accepts some string of the form $(u, u)$. □

**Open Question 4.2.11.** Is the conjugacy problem solvable for automatic groups?

Besides the ordering by word length, another ordering that proves to be both useful and crucial in this context is the **ShortLex** ordering.

**Definition 4.2.12** (**ShortLex**-Automatic Group)**.** Let $G$ be a group with a set $A$ of semigroup generators. Then $G$ is ***ShortLex**-automatic* if the language of all **ShortLex**-geodesics over $A$ forms part of an automatic structure for $G$.

Why is **ShortLex**-automaticity interesting? Recall that we previously defined the uniqueness property of the language of an automatic group, which is a highly desirable feature. It turns out that being **ShortLex**-automatic serves as an indicator of the uniqueness property, as stated in the following theorem.

**Theorem 4.2.13** (Uniqueness Property)**.** *Let $G$ be an automatic group with an automatic structure $(A, L)$. Let $L' \subset L$ be the set of strings $w \in L$ such that $w$ is **ShortLex**-minimal in $L \cap \pi^{-1}(\overline{w})$. Then $(A, L')$ is an automatic structure for $G$. In particular, $G$ admits an automatic structure over $A$ with the uniqueness property.*

*Proof.* Let $M_\varepsilon$ be the equality recognizer for the given automatic structure $(A, L)$. Consider the following language:

$$L' = \{w \in L \mid (\forall v)((w, v) \in L(M_\varepsilon) \Rightarrow w \leq v \text{ with respect to } \textbf{ShortLex})\}$$

Since the predicate $w \leq v$ with respect to **ShortLex** can be checked by a finite state automaton, by Corollary 2.2.18 we know that $L'$ is also regular. Recall that restriction is also automatic by Proposition 4.1.5, thus $(A, L')$ is also part of an automatic structure for $G$. In particular, the uniqueness property follows from the **ShortLex** minimality of any $w \in L'$. □

We see an corollary of such uniqueness property, which has been noted independently by a number of researchers:

**Corollary 4.2.14** (Infinite Torsion)**.** *Let $G$ be an infinite group. Suppose every element $g \in G$ has finite order. Then $G$ cannot be automatic.*

*Proof.* We proceed by contradiction. Suppose such infinite group $G$ is automatic, then it admits an automatic structure with the uniqueness property by Theorem 4.2.13. Let $(A, L)$ be such automatic structure with the uniqueness property, then by the pumping lemma (Theorem 2.2.19), there exists some integer $p$ such that any string in $L$ having length greater than $p$ is of the form $uvw$, with $|v| > 0$ and $uv^iw \in L$ for all $i$. However, this is impossible, because $v$ has finite order and $L$ has the uniqueness property. Hence no strings in $L$ should have length that exceeds $p$, and $L$ is thus finite. □

Note that the existence of a finitely generated infinite torsion group is highly nontrivial. The famous group-theoretic study of torsion groups and the conditions under which they can be infinite is known as the *Burnside problem*. This problem is fundamental and has been extensively researched in group theory; however, it has little connection to automatic groups and thus falls outside the scope of this discussion.

What is the relationship between being strongly geodesically automatic, weakly geodesically automatic, and the newly introduced **ShortLex**-automaticity? The following corollary summarizes this result:

**Corollary 4.2.15** (Geodesic Hierarchy)**.** *Let $G_1$ be a strongly geodesically automatic group. Then it is **ShortLex**-automatic for any ordering of the generators. Let $G_2$ be a **ShortLex**-automatic group. Then for some ordering of generators it is weakly geodesically automatic.*

Clearly, ordering plays a crucial role when discussing the relationship between **ShortLex**-automaticity and weakly geodesically automaticity. This observation leads to the following open question:

**Open Question 4.2.16.** Does there exist a group that is weakly geodesically automatic but not **ShortLex**-automatic for any choice of ordering on the generators?

We now examine the final useful property of languages and explore its relationship with the properties discussed thus far.

**Theorem 4.2.17** (Prefix Closure of Automatic Groups)**.** *Let $G$ be an automatic group with an automatic structure $(A, L)$, and let $L'$ be the prefix closure of $L$. Then $(A, L')$ also gives an automatic structure for $G$. Furthermore, let $c$ be the number of states of a finite state automaton accepting $L$ and suppose the number of representatives of any element of $G$ in $L$ is bounded by a number $n > 0$. Then the number of representatives of any element of $G$ in $L$ will be bounded by $nc$.*

*Proof.* Firstly by Theorem 2.2.9 we know that $L'$ is regular. Given the finite state automaton $W$ for the language $L$, we can obtain the automaton $W'$ that accepts the language $L'$ by simply making every live state of $W$ into an accept state. We let the $c$ be the number of states in $W$.

Then we want to show $(A, L')$ actually forms an automatic structure on $G$ by Theorem 4.1.17. Let $w_1$ and $w_2$ be prefixes of strings in $L$ representing elements of $G$ with a distance at most one apart in the Cayley graph. Let $u_1$ and $u_2$ be strings whose lengths are less than $c$ and $w_1u_1$, $w_2u_2$ are in $L$. When we apply Theorem 4.1.17 to $L$, we get a constant $k$, since $\overline{w_1u_1}$ and $\overline{w_2u_2}$ are at most $2c+1$ apart, their corresponding paths $\widehat{w_1u_1}$ and $\widehat{w_2u_2}$ are at a uniform distance at most $k(2c+1)$ apart. Thus it follows that $\widehat{w_1}$ and $\widehat{w_2}$ are a uniform distance at most $k(2c+1)+2c$ apart.

Now it remains to show the last statement, we let $C_g = L' \cap \pi^{-1}(g)$ for $g \in G$. We partition $C_g$ into sets $C_{g,s}$, where $w \in C_{g,s}$ of the automaton $W$ is in the state $s$ after reading $w$. For each state $s$, we take a string $u_S$ leading from $s$ to an accept state. Then we observe that the strings $wu_s$ for $w \in C_{g,s}$ lie in the language $L$ and represent the same element of $G$. By assumption the number of such strings is bounded by $n$, hence $C_g$ has at most $cn$ elements, which concludes the proof. □

Having introduced the uniqueness and prefix-closed properties, we now present the following proposition, which shows that a group containing a **ShortLex**-automatic structure necessarily satisfies both properties simultaneously.

**Proposition 4.2.18** (Geodesic Automaton **ShortLex**)**.** *Let $G$ be an automatic group with an automatic structure $(A, L)$, and suppose $L$ contains the **ShortLex** language $L'$ as being given in Theorem 4.2.13. Then $G$ admits an automatic structure that is prefix-closed and has the uniqueness property simultaneously.*

*Proof.* By Theorem 4.2.13 we know that $(A, L')$ gives an automatic structure for $G$ having the uniqueness property. It suffices to show $L'$ is prefix-closed, this follows easily from the fact that **ShortLex** order is preserved under left and right multiplication by any element of $A^*$. □

We have shown that every automatic structure can be made to have either the uniqueness property or to be prefix-closed. Naturally, we seek to generalize this simultaneity result to all automatic groups, leading to the following open question:

**Open Question 4.2.19.** Given an automatic group $G$ with a generating set $A$, does there exist an automatic structure of $G$ over $A$ whose language is both prefix-closed and satisfies the uniqueness property? If we relax this requirement slightly, what if we allow a change of the generating set?

It is a general fact that the automaticity of a group $G$ is invariant under a change of generators, as stated in Theorem 4.1.21. However, this invariance does not necessarily extend to **ShortLex**-automatic structures. It is possible for a group to be **ShortLex**-automatic with respect to one generating set but not with respect to another.

We present an example of this phenomenon here, without proof. Before introducing the example, we first state an algebraic fact that will serve as a key ingredient in our discussion.

**Theorem 4.2.20** (Direct Product of Automatic Groups is Automatic)**.** *The direct product of two automatic groups is automatic, and the direct product of two biautomatic groups is biautomatic.*

*Proof.* Let $G_1$ and $G_2$ be automatic groups with corresponding automatic structures $(A_1, L_1)$ and $(A_2, L_2)$. We can assume $L_1$ consists of unique representatives for the elements of $G_1$ by Theorem 4.2.13, and we claim $(A_1 \cup A_2, L_1 L_2)$ is an automatic structure for $G = G_1 \times G_2$.

It is clear that $L_1L_2$ maps onto the group $G$, we let $u_1, v_1 \in L_1$, $u_2, v_2 \in L_2$, and we let $u_1u_2$ and $v_1v_2$ represent elements of the group $G$ within one distance from each other in the Cayley graph $\Gamma(G)$. Then we know that $u_1$ and $v_1$ are within one distance apart in $\Gamma(G_1)$, likewise $u_2$ and $v_2$ are within one distance apart in $\Gamma(G_2)$. By Theorem 4.1.17, there exists a constant $c_1$ that bounds the uniform distance between the paths $\widehat{u_1}$ and $\widehat{v_1}$ in $\Gamma(G_1)$, and a constant $c_2$ that bounds the uniform distance between the paths $\widehat{u_2}$ and $\widehat{v_2}$ in $\Gamma(G_2)$, we let the constant $c \coloneqq \max\{c_1, c_2\}$. By the uniqueness property of $L_1$ and Lemma 4.1.8, there is a constant $k$ that bounds the difference in lengths between $u_1$ and $v_1$. Thus the uniform distance between $\widehat{u_1u_2}$ and $\widehat{v_1v_2}$ is bounded by the constant $c + k$.

The proof of the statement about biautomaticity follows in the same way. □

**Example 4.2.21** (Surfeit)**.** We consider the group $P = \langle a, b\rangle \times \langle c, d\rangle$, the direct product of two free groups. The automaticity of free groups is guaranteed by Theorem 4.2.4, and by Theorem 4.2.20, we know that $P$, as a direct product of automatic groups, is itself automatic. Since free groups admit unique geodesic representatives for their elements, it follows that $P$ is **ShortLex**-automatic with respect to the generating set $\{a, b, c, d\}$. However, we claim that $P$ is not **ShortLex**-automatic with respect to the following generating set:

$$\{a, b, c, d, add, bbc, a^{-1}, b^{-1}, c^{-1}, d^{-1}, (add)^{-1}, (bbc)^{-1}\}$$

The fundamental reason why the group $P$ fails to be **ShortLex**-automatic with respect to the second generating set is that strings are efficiently abbreviated by grouping $b$ and $c$ together, as well as by grouping $a$ and $d$ together. However, these two types of groupings typically interfere with one another.

We focus on group elements where both types of abbreviation can be applied multiple times, but not simultaneously. If we choose a representative containing many occurrences of *add* but no occurrences of *bbc*, then, due to the constraints of a finite-state machine, we can identify a substring where relatively few occurrences of *add* appear. In such cases, we can demonstrate that the string could be rewritten more economically using *bbc*, thereby contradicting the assumption of **ShortLex**-automaticity.

In fact, our understanding of automatic groups remains limited. Given that the direct product of automatic groups is itself automatic, one can naturally pose the following

question:

**Open Question 4.2.22.** Are factors of automatic groups automatic? More generally, is a retract of an automatic group automatic?

Although questions about factors and retractions of automatic groups remain unresolved, it is fortunate that the answer concerning finite index subgroup relationships is known:

**Theorem 4.2.23** (Finite Index Subgroup of Automatic Group is Automatic)**.** *Let $G$ be a group and let $H$ be a subgroup of finite index. Then $G$ is automatic if and only if $H$ is, and if $G$ is biautomatic, so is $H$.*

Note that automaticity is preserved by an if and only if statement. However, in the case of biautomaticity, only one direction is known, while the other remains an open problem:

**Open Question 4.2.24.** Let $G$ be a group and let $H$ be a subgroup of finite index. If $H$ is biautomatic, does it follow that $G$ is biautomatic?

## 4.3 Finding Automatic Structures

Having introduced various definitions and theories related to automatic groups, we now turn to the following fundamental question: given a finite state automaton, does it define an automatic structure for some automatic group? Moreover, can we establish axioms or algorithms to determine this? We will address these questions both theoretically and practically in the following sections.

The starting point is to list all axioms for an automatic group that are checkable, as they are expressed in regular predicates. We then present an algorithm by Epstein, Holt, and Thurston, which, while theoretically feasible, is impractical due to its excessive computational time in practice.

### 4.3.1 Axioms

Suppose we have an alphabet $A$ and a finite state automaton $W$ over $A$ that accepts the language $L$. For each $x \in A \cup \{\varepsilon\}$, we also have finite-state automata $M_x$ over $(A, A)$

with languages $L_x$. Recalling that any automatic group must admit an automatic structure $(A, W, L, M_x)$ by Definition 4.1.3, it is natural to ask the converse: suppose we are given these data first, do they define an automatic structure for some group?

We now present a set of axioms due to Thurston, Epstein, Holt, Uri Zwick, and Chuya Hayashi, which provide necessary and sufficient conditions for these data to form an automatic structure for some group. With these axioms in mind, given finite state automata with associated languages and alphabets, we can theoretically verify whether they define an automatic structure through an algorithmic process. Here we simply list all the axioms and results. Readers interested in further details are encouraged to consult the original articles, there are 13 axioms in total.

**Axiom 1.** $(\exists w)(w \in L)$

**Axiom 2.** For each $x \in A \cup \{\varepsilon\}$,

$$(\forall w_1, w_2)((w_1, w_2) \in L(M_x) \Rightarrow (w_1 \in L \wedge w_2 \in L))$$

**Axiom 3.** $(\forall w)(w \in L \Rightarrow (w, w) \in L(M_\varepsilon))$

**Axiom 4.** $(\forall w_1, w_2)((w_1, w_2) \in L(M_\varepsilon) \Rightarrow (w_2, w_1) \in L(M_\varepsilon))$

**Axiom 5.** $(\forall w_1, w_2, w_3)(((w_1, w_2) \in L(M_\varepsilon) \wedge (w_2, w_3) \in L(M_\varepsilon)) \Rightarrow (w_1, w_3) \in L(M_\varepsilon))$

Axioms 3–5 indicate that we can construct $X$, the set of equivalence classes under the equivalence relation defined by $M_\varepsilon$. We denote the equivalence class of a word $w$ by $[w]$.

**Axiom 6.** For each $x \in A$,

$$(\forall w_1)(w_1 \in L \Rightarrow (\exists w_2)((w_1, w_2) \in L(M_x)))$$

**Axiom 7.** For each $x \in A$,

$$(\forall w_1, w_2, w_3)(((w_1, w_2) \in L(M_x) \wedge (w_1, w_3) \in L(M_x)) \Rightarrow (w_2, w_3) \in L(M_x))$$

**Axiom 8.** For each $x \in A$,

$$(\forall w_1, w_2, w_3)(((w_1, w_2) \in L(M_\varepsilon) \wedge (w_1, w_3) \in L(M_x)) \Rightarrow (w_2, w_3) \in L(M_x))$$

Axiom 8 implies that the map $L \to X$ defined in this way factors through $X$, thereby inducing a map $\sigma_x : X \to X$.

**Axiom 9.** For each $x \in A$,

$$(\forall w_2)(w_2 \in L \Rightarrow (\exists w_1)((w_1, w_2) \in L(M_x)))$$

**Axiom 10.** For each $x \in A$,

$$(\forall w_1, w_2, w_3)(((w_1, w_2) \in L(M_x) \wedge (w_3, w_2) \in L(M_x)) \Rightarrow (w_1, w_3) \in L(M_\epsilon))$$

**Axiom 11.** For each $x \in A$,

$$(\forall w_1, w_2, w_3)(((w_1, w_2) \in L(M_\varepsilon) \wedge (w_3, w_1) \in L(M_x)) \Rightarrow (w_3, w_2) \in L(M_x))$$

**Axiom 12.** Let $W$ be the given finite state automaton and let $k$ be the number of states in $W$. Then for each $w, w' \in A^*$ such that $|w|, |w'| \leq k$,

$$(\forall u)((uw \in L \wedge uw' \in L) \Rightarrow ([uw]\sigma_w^{-1} = [uw']\sigma_{w'}^{-1}))$$

**Axiom 13.** Let $W$ be the given finite state automaton and let $k$ be the number of states in $W$. Let $k' > k$ and be greater than the length of some accepted string representing the identity. Then for each string $w$ over $A$ with length $|w| \leq 4k'$:

$$(\exists u \in L)(([u]\sigma_w = [u]) \Rightarrow ((\forall u \in L)([u]\sigma_w = [u])))$$

**Remark 4.3.1.** Note that the above axioms should be considered as conjunctions of several statements, one of each $x \in A \cup \{\varepsilon\}$, because we are not allowed to quantify over the set of generators.

Given all these 13 axioms, we state the following main result:

**Theorem 4.3.2.** *Let $A$ be a finite alphabet, let $W$ be a finite state automaton over $A$ with corresponding accepted language $L$, and for $x \in A \cup \{\varepsilon\}$, let $M_x$ be finite state automata over $(A, A)$. Then $(A, W, L, M_x)$ forms an automatic structure for a group $G$ if and only if Axioms $1 - 13$ are satisfied.*

### 4.3.2 A Naive Algorithm

We present a procedure for constructing an automatic structure from a finite group presentation. This procedure will terminate if the group admits an automatic structure; otherwise, it will run indefinitely.

Note that it is impossible to develop an algorithm that definitively answers **Yes** or **No** to the question of whether a given group is automatic, as the problem of determining whether a group presentation defines the trivial group is undecidable.

Before presenting the procedure, we first introduce some key ingredients.

**Definition 4.3.3** (Partial Cayley Graph)**.** Let $G = \langle A|R\rangle$ be a finitely presented group. A *partial Cayley graph* over $A \cup A^{-1}$ is a finite, connected, directed graph $T$ whose edges are labeled by elements of $A \cup A^{-1}$. If there is an edge labeled $x$ from a vertex $v_1$ to a vertex $v_2$, then there must also be an edge labeled $x^{-1}$ from $v_2$ to $v_1$.

Furthermore, a partial Cayley graph is said to be *unambiguous* if no two directed edges originating from the same vertex have the same label. Let $v$ be a vertex of an unambiguous partial Cayley graph $T$, and let $w$ be a string over $A \cup A^{-1}$. The string $w$ may define a path $\alpha(w, v)$ in $T$ starting at $v$, but it is possible that at some point the next letter in $w$ is not defined for the current vertex. Under this condition, if the entire path $\alpha(w, v)$ is defined in $T$, we set $p(w, v)$ to be the vertex at which the path terminates. Otherwise, $\alpha(w, v)$ and $p(w, v)$ remain undefined.

A path $\alpha(w, v)$ is called a *loop* if $p(w, v) = v$, and it is further called a *simple loop* if no other vertex in $\alpha(w, v)$ is repeated. An unambiguous partial Cayley graph $T$ is said to be *partially homogeneous* if for any simple loop $\alpha(w, v)$ and any vertex $v'$ in $T$, the path $\alpha(w, v')$ is either undefined or also a loop.

**Proposition 4.3.4.** *Any partial Cayley graph $T$ can be made unambiguous and partially homogeneous in a finite number of steps of two types: scissor moves and loop closings.*

As their names suggest, we briefly describe the two types of moves:

1. A *scissor move* consists of identifying two distinct edges that share the same label and the same source vertex, while also identifying their respective target vertices.

2. A *loop closing* occurs when there exist vertices $v$ and $v'$ and a word $w$ such that $\alpha(w, v)$ forms a simple loop, but $\alpha(w, v')$ does not. In this case, we identify $p(w, v')$ with $v'$.

Since there are only finitely many vertices, $T$ becomes unambiguous after sufficiently many scissor moves. Then, by performing a finite number of loop-closing steps, we make $T$ partially homogeneous. However, this process may introduce ambiguity in $T$ once again, which we resolve by applying additional scissor moves. Thus, after a finite sequence of scissor moves and loop closings, $T$ can be made both partially homogeneous and unambiguous.

**Algorithm 4.3.5** (Todd-Coxeter Algorithm)**.** Let $G = \langle A|R\rangle$ be a finitely presented group. The Todd-Coxeter algorithm constructs a sequence of unambiguous, partially homogeneous partial Cayley graphs $T_i$ for each $i \geq 0$ inductively as follows:

1. Initialize with a basepoint $*$ and attach a loop for each relator in $R$. Each loop starts and ends at $*$, with successive edges encoding the corresponding word in $R$. Using Proposition 4.3.4, make the resulting graph unambiguous and partially homogeneous. Denote the resulting graph as $T_0$.

2. Assume $T_i$ has been constructed. If all arrows are defined, then $G$ is finite, and $T_i$ is the complete Cayley graph of $G$. Otherwise, define all missing arrows on vertices of $T_i$, directing them toward newly created vertices. Then, make the resulting graph unambiguous and partially homogeneous, and denote it as $T_{i+1}$.

**Theorem 4.3.6.** *Let $G = \langle A|R\rangle$ be a finitely presented group, and let $W$ be a finite-state automaton over $A$. Then the following procedure terminates if and only if $G$ is an automatic group; otherwise, it runs indefinitely:*

1. *Use the Todd-Coxeter algorithm (Algorithm 4.3.5) to construct the partial Cayley graph $T_i$ for each $i \geq 0$.*

2. *Construct the associated standard automata $M_x(W, T_i)$ for each $x \in A \cup \{\varepsilon\}$ using the partial Cayley graphs $T_i$.*

3. *Verify whether* $W$ *and* $M_x(W, T_i)$ *define an automatic structure for* $G$ *by checking Axioms* 1–13 *and testing the equality* $[\varepsilon]\sigma_w = [\varepsilon]$ *for each* $w \in R$. *Terminate the procedure once a positive result is obtained.*

### 4.3.3 The Knuth-Bendix Procedure

We previously introduced the theoretical Todd-Coxeter Algorithm (Algorithm 4.3.5); however, this approach is highly impractical.

Recall that we discussed the desirable **ShortLex**-automatic structure, which possesses the properties of prefix-closedness and uniqueness. In this section, we trade generality for efficiency by introducing a practical algorithm for finding **ShortLex**-automatic structures: the *Knuth-Bendix completion procedure*.

The input to the Knuth-Bendix procedure is a finite group presentation, and the output is a **ShortLex**-automatic structure that provides a well-defined normal form for words in the group, ensuring both uniqueness and efficient computation. Rather than demonstrating the full procedure, we summarize its key concepts and steps. In addition to the finite group presentation, the Knuth-Bendix procedure relies on several fundamental principles:

1. A *rewriting system* consists of a set of directed rules that allow words in the group to be rewritten into simpler forms. Given a relation $s = t$, we orient it as a rewrite rule $s \to t$, ensuring a consistent reduction strategy. The goal of rewriting is to transform words into a unique normal form by repeatedly applying rewrite rules until no further reductions are possible.

2. The choice of an appropriate ordering is crucial for ensuring termination. Here, we use the **ShortLex** ordering.

3. A rewriting system is said to be *confluent* if, whenever a word can be rewritten in multiple ways, all reduction paths eventually lead to the same unique normal form. This property ensures that every element of the group has a well-defined representative.

4. If two rewrite rules apply to overlapping parts of a word, they may produce different results, forming a *critical pair*. The procedure detects and resolves such conflicts by introducing new rules and iterating until confluence is achieved.

5. The rewriting process must terminate, meaning no infinite sequences of reductions occur. This is guaranteed by choosing a well-founded ordering that prevents infinite descent.

Next, we briefly discuss the algorithmic process, which proceeds as follows:

1. **Orientation of Relations:** Relations are transformed into rewrite rules. Given a relation $s = t$, we determine a direction $s \to t$ based on the **ShortLex** ordering.

2. **Normalization:** Words are systematically replaced using rewrite rules to obtain simpler forms.

3. **Critical Pair Resolution:** Overlapping rewrite rules are examined, and new rules are introduced to ensure that different rewrite sequences lead to a common normal form.

4. **Iteration Until Confluence:** The procedure iterates by adding rules and resolving conflicts until the system becomes both confluent and terminating.

The resulting **ShortLex**-automatic structure provides an efficient mechanism for group computations, enabling fast word reduction and membership checking. Unlike the Todd-Coxeter algorithm, which relies on explicit coset enumeration, the Knuth-Bendix procedure constructs a structured and minimal representation of group elements. Several optimizations that enhance the efficiency of the process can be found in [EHR91, Section 3-6], while more general discussions on the topic are available in [ES00].

## 4.4 The Growth Function

In addition to studying automatic groups through the lens of the Dehn function, we introduce another tool for analyzing automatic groups and broader group-theoretic problems: the growth function.

**Definition 4.4.1** (Growth Function of Languages)**.** The *growth function* of a language $L$ over $A$ is a function $g_L : \mathbb{N} \to \mathbb{N}$ which counts the elements of $L$ of different lengths: $g_L(n) \coloneqq \#(L \cap A^n)$.

Sometimes it is most convenient to encode this information as a formal power series $f_L$, which is called the *generating function* for $g_L$:

$$f_L(t) = \sum_{n=0}^{\infty} g_L(n)t^n$$

Note that if $L = A^*$, then $g_L(n) = |A|^n$, indicating that $|A|^n$ is the upper bound for every language.

**Definition 4.4.2** (Growth Function of Groups)**.** Let $G$ be a finitely generated group and let $A$ be a finite generating set that is closed under inversion. Then the *growth function* of the group $G$ over $A$ is a function $g_{G,A}(n) : \mathbb{N} \to \mathbb{N}$ that counts the number of elements of $G$ that can be expressed in terms of words of length at most $n$ over $A$.

By using the growth function, we can form power series $B_{G,A}(z) = \sum_{n=0}^{\infty} g_{G,A}(n)z^n$, which is called the *growth series* of $G$ over $A$. Let $f_{G,A}(n)$ be the number of elements of $G$ whose shortest representative in $A^*$ has exactly length $n$, we can form another power series $C_{G,A}(z) = \sum_{n=0}^{\infty} f_{G,A}(n)z^n$. As stated in [EIFZ96, Section 1], by the definitions and relation between $g_{G,A}$ and $f_{G,A}$, we have

$$C_{G,A}(z) = B_{G,A}(z)(1 - z)$$

thus studying one of these functions is equivalent to studying the other.

We speak of one growth function or growth series as being *bounded* by other if the bound holds for every $n \in \mathbb{N}$.

**Definition 4.4.3** (Growth Class)**.** We say a group $G$ has *polynomial growth* if its growth function is bounded above by some polynomial function of $n$, and *exponential growth* if its growth function is bounded below by an exponential function of the form $\lambda^n$ with $\lambda > 1$. We also say $G$ has *rational growth* if its growth series is the power series for a rational function of $z$.

As stated in [ECH+92, Page 19 and Proposition 1.3.8], the following fundamental property of regular languages holds, which leads to an important result about automatic groups.

**Proposition 4.4.4.** *Every regular language exhibits either polynomial or exponential growth, and every regular language has rational growth.*

However, while the regular language $L(W)$ in an automatic structure of an automatic group has rational growth, this does not necessarily imply that every automatic group exhibits rational growth. Although most automatic groups do have rational growth, the existence of a unique representative for each element of $G$ in $A^*$, accepted by the associated word acceptor $W$, does not guarantee this property.

Recall that the growth function for $G$ with respect to $A$ is defined using the shortest representatives of elements in $G$, whereas representatives accepted by the word acceptor need not be the shortest. To establish a uniform conclusion relating these two types of growth functions, we refer to [EIFZ96, Proposition 7.5] and conclude the following result for automatic groups:

**Corollary 4.4.5.** *If $G$ admits a geodesic automatic structure, then the growth function of $G$ is rational.*

# Chapter 5

# Applications of Automatic Groups

## 5.1 Hyperbolic Groups

In this section, we survey fundamental results and concepts related to hyperbolic groups, which is a class of groups introduced by Gromov [Gro87]. Our central goal is to demonstrate that word hyperbolicity implies automaticity.

Recall that the notion of a $(k, \epsilon)$-quasi-geodesic was introduced in Definition 3.3.18. To motivate this section, we now present a connection between quasi-geodesics and automatic groups.

**Theorem 5.1.1** (Accepted Implies Quasi-Geodesic)**.** *Let $G$ be an automatic group with an automatic structure $(A, L)$. Suppose for every element $g \in G$, the number of accepted representatives $\overline{g} \in L$ is bounded. Then there exists a constant number $N$ such that any path in the Cayley graph $\Gamma(G, A)$ corresponding to an accepted string is an $(N, N)$-quasi-geodesic.*

*Proof.* Firstly, we replace $L$ by its prefix closure and we observe that it still satisfies the finiteness condition by Theorem 4.2.17. We let $w = ruv$ be an accepted string and let $u'$ be a geodesic path in the Cayley graph $\Gamma(G, A)$ from $\overline{r}$ to $\overline{ru}$, we have to show that $|u'|$ is within a bounded difference from $|u|$. For $0 \leq i \leq |u'|$, we let $r_i$ be an accepted word representing $\overline{ru'(i)}$, since $L$ is prefix closed, we can take $r_0 = r$ and $r_{|u'|} = ru$. Now note that Lemma 4.1.8 provides us a constant $N$ such that the difference in length between $r_0$ and $r_{|u'|}$ is at most $N \cdot \max\{1, |u'|\}$, Therefore, we have

$$|r| + |u| = |ru| \leq |r| + N(|u'| + 1)$$

which further implies

$$|u'| \leq |u| \leq N(|u'| + 1)$$

hence the result follows. □

**Definition 5.1.2** (Riemannian Metric)**.** Let $M$ be a smooth manifold. A *Riemannian metric* on $M$ is a smoothly chosen inner product $g : T_pM \times T_pM \to \mathbb{R}$ on each of the tangent spaces $T_pM$ of $M$.

**Definition 5.1.3** (Riemannian Manifold)**.** A *Riemannian manifold* is a pair $(M, g)$, where $M$ is a smooth manifold and $g$ is a Riemannian metric on $M$.

Next, we see one lemma due to Morse, which is implicit in [Mor24] as part of the proof of Mostow's Rigidity Theorem, the proof can be found in [Thu22]:

**Lemma 5.1.4.** *Let $M$ be a compact manifold with strictly negative sectional curvature. Then there exists a constant $c = c(k)$ such that any finite $(k, \epsilon)$-quasi-geodesic $\alpha$ in the universal cover $\widetilde{M}$ of $M$ lies within the $c$-neighborhood of the geodesic segment joining the endpoints of $\alpha$.*

**Remark 5.1.5.** Note that this result does not extend to spaces that are not negatively curved.

By using this Lemma, we can prove the following theorem:

**Theorem 5.1.6** (Negative Curvature)**.** *Let $M$ be a compact manifold with strictly negative sectional curvatures ane let $S$ be a generating set for the fundamental group $G = \pi_1(M)$. Then the set of geodesic words in the Cayley graph $\Gamma(G, S)$ is a regular language, and it is part of an automatic structure for $G$. In particular, the fundamental group $G$ is automatic.*

The modified proof due to Farb can be found in [Far92, Theorem 3], while the original proof by Cannon appears in [Can84]. The original proof introduced the notion of *cone type*, based on the idea that geodesics in a hyperbolic group exhibit only finitely many asymptotic behaviors—namely, finitely many cone types.

Let $G$ be the fundamental group of a compact Riemannian manifold $M$, and let $S$ be a generating set for $G$. Then there is a natural copy of the Cayley Graph $\Gamma(G, S)$ in the

universal cover $\widetilde{M}$ of $M$. To be more specific, we choose a basepoint $m \in M$ and a lift $\widetilde{m}$ of $m_0$ in the universal covering space $\widetilde{M}$, then we construct a graph whose vertices and edges correspond to the Cayley graph $\Gamma(G, S)$, but embedded into $\widetilde{M}$ as follows:

1. For each deck transformation $g \in G$, we place a vertex at the point $g(\widetilde{m})$ in $\widetilde{M}$.
2. If $g$ and $g \cdot s$ are connected by an edge in the Cayley graph $\Gamma(G, S)$, then in $\widetilde{M}$, we connect the points $g(\widetilde{m})$ and $gs(\widetilde{m})$ with a geodesic segment.

It is a fundamental fact that the geodesic paths in the Cayley graph $\Gamma(G, S)$ are actually quasi-geodesics in the universal cover $\widetilde{M}$. Thus Theorem 5.1.6 is actually a particular case of a more general result involving the word hyperbolic groups.

**Theorem 5.1.7** (Word Hyperbolicity Implies Automaticity)**.** *Let $G$ be a word hyperbolic group with a finite set of semigroup generators $A$ that is closed under inversion. Then the set of geodesics over $A$ forms a regular language, which constitutes part of an automatic structure for $G$.*

*Proof.* Given two geodesic strings $u$ and $v$ such that $d(\overline{u}, \overline{v}) = 1$, we obtain a geodesic triangle where the three side lengths are $|u|$, $|v|$, and either 1 or 0. Since the Cayley graph of $G$ is a hyperbolic space, it follows that the paths $\widehat{u}$ and $\widehat{v}$ are within a Hausdorff distance of at most $(\delta + 1)$ from each other. Hence, the automaticity of $G$ follows from Theorem 4.2.4. □

Recall the definition of strongly geodesically automatic groups (Definition 4.2.3). The following is a stronger result due to Cannon and Papasoglu [Pap95, Theorem 1, 2]:

**Theorem 5.1.8.** *A group $G$ is hyperbolic if and only if it is strongly geodesically automatic.*

In summary, the automatic structure of a word hyperbolic group follows from the fact that any two geodesic words are separated by a uniformly bounded distance.

## 5.2 Euclidean Groups

In this section, we examine a class of groups distinct from word hyperbolic groups—namely, the Euclidean groups. These are discrete cocompact subgroups of the isometry group of Euclidean space in any dimension.

With further analysis, we can show that Euclidean groups admit a biautomatic structure that simultaneously satisfies prefix-closedness, representation by geodesics, and the uniqueness property, making them particularly well-behaved.

Recall that Theorem 4.2.20 states that the direct product of two automatic groups is automatic. It follows immediately that every finitely generated abelian group is automatic, since any such group can be expressed as the product of a finite group and finitely many copies of $\mathbb{Z}$ by the Fundamental Theorem of Finitely Generated Abelian Groups.

In fact, we can conclude the following statement from either Theorem 4.2.20 or Theorem 4.2.6:

**Proposition 5.2.1.** *Finitely generated abelian groups are biautomatic.*

Thus, we can narrow the problem: it suffices to study the automatic structure of $\mathbb{Z}$. The simplest automatic structure for $\mathbb{Z}$ is given by the regular expression $(x)^* \vee (x^{-1})^*$, where $x$ is a generator of $\mathbb{Z}$.

What about the free abelian group on two generators, $x$ and $y$? We consider the following example:

**Example 5.2.2.** For the free abelian group generated by $x$ and $y$, its automatic structure can be given by the regular expression $((x)^* \vee (x^{-1})^*)((y)^* \vee (y^{-1})^*)$, which coincides with the **ShortLex** language on the ordered alphabet $\{x, x^{-1}, y, y^{-1}\}$. If we use the same generators but in different order: $\{x, y, y^{-1}, x^{-1}\}$, the corresponding **ShortLex** language is symmetric, given by the regular expression

$$((x)^*(y)^*) \vee ((x)^*(y^{-1})^*) \vee ((y)^*(x^{-1})^*) \vee ((y^{-1})^*(x^{-1})^*)$$

Using this idea, we can generalize the automatic structure to any finitely generated abelian groups:

**Theorem 5.2.3** (Abelian Symmetric Automation)**.** *Let $G$ be a finitely generated abelian group, then it admits a automatic structure.*

*Proof.* We decompose $G$ into the product of a finite group $H$ and a free abelian group $F$ on generators $x_1, ..., x_k$. We consider the ordered alphabet $A = \{x_1, ..., x_k, x_k^{-1}, ..., x_1^{-1}\}$. Then

the language $L = \mathbf{ShortLex}(F, A)$ will give us a symmetric automation for $F$, and we get the automatic structure for $G$ from the language $HLH$ over the alphabet $H \cup A$. □

One might ask whether abelianness implies **ShortLex**-automaticity. The answer is yes—Holt [ECH+92, Theorem 4.3.1] proved the following result:

**Theorem 5.2.4.** *Let $G$ be a finitely generated abelian group. Then $G$ is **ShortLex**-automatic with respect to any ordered set of semigroup generators.*

Bieberbach [Bie12] proved that virtually abelian groups can be realized as discrete groups of isometries of Euclidean spaces $\mathbb{R}^n$ for some $n$, the proof can be obtained from [Cha86], thus we call virtually abelian groups *Euclidean groups*. Now, applying Theorem 5.2.3 together with Theorem 4.2.23, we conclude the following result:

**Corollary 5.2.5.** *Euclidean groups are automatic.*

Even more, Thurston, Epstein, and Levy [ECH+92, Corollary 4.2.4] demonstrated that Euclidean groups are biautomatic:

**Theorem 5.2.6.** *Any Euclidean group admits a prefix-closed biautomatic structure consisting of unique geodesic representatives.*

We conclude this section by presenting an example of a Euclidean group that is **ShortLex**-automatic with respect to a certain set of generators but far from being strongly geodesically automatic. Moreover, by reversing the order of the same generators, the group ceases to be **ShortLex**-automatic. This example is due to John Sullivan and Cannon.

**Example 5.2.7.** Let $G$ be the wreath product of $\mathbb{Z}$ with $\mathbb{Z}_2$. This group is generated by translations and the diagonal flip in $\mathbb{Z} \times \mathbb{Z}$. A group presentation for $G$ is given by

$$G = \langle x, y, z | x^2, zyz^{-1}y^{-1}, yxz^{-1}x^{-1} \rangle,$$

where $x$ represents the flip, while $y$ and $z$ correspond to unit translations along two coordinate axes in $\mathbb{Z}^2$. We observe that $G$ is biautomatic since it contains $\mathbb{Z} \times \mathbb{Z}$ as a subgroup of index 2. Moreover, one can show that $G$ is **ShortLex**-automatic with respect to the ordered set of generators

$$A = \{z^{-1}, z, y^{-1}, y, x\}$$

but it ceases to be **ShortLex**-automatic when the order of the generators is reversed, i.e., when considering

$$A' = \{x, y, y^{-1}, z, z^{-1}\}$$

Furthermore, if we extend the generating set by adding $y^2$ and $yz$, the set of geodesic strings over

$$A'' = \{x, y, y^{-1}, z, z^{-1}, y^2, y^{-2}, yz, z^{-1}y^{-1}\}$$

is no longer a regular language.

## 5.3 Nilpotent Groups

So far, we have focused on the theory of automatic groups and key examples of important groups that exhibit automaticity. However, it is equally important to examine non-examples. In particular, we demonstrate here that nilpotent groups are not, in general, automatic. However, every nilpotent group has a decidable word problem, which further demonstrates that automaticity is not a reliable indicator of whether a group has a solvable word problem.

We start by example of the three-dimensional Heisenberg group, which is the simplest non-abelian nilpotent group, and it is not automatic.

**Example 5.3.1** (Three Dimensional Heisenberg Group)**.** Consider the three-dimensional Heisenberg group $H_3$, which has presentation

$$H_3 = \langle a, b, c \mid [a, b]c^{-1}, [a, c], [b, c] \rangle$$

as stated in [Far92, Page 13] and proved in [ECH+92, Example 8.1.1], $H_3$ has a cubic isoperimetric function, indicating that it does not have a quadratic isoperimetric inequality, and therefore, by Theorem 4.1.9 the three-dimensional Heisenberg group is not automatic.

Recall that Theorem 4.1.9 establishes that automatic groups satisfy a quadratic isoperimetric inequality. Thus by Theorem 4.1.9 and from the example of the three-dimensional Heisenberg group discussed above, we observe that a group failing to satisfy a quadratic isoperimetric inequality cannot be automatic.

This naturally leads to the question: is automaticity equivalent to satisfying a quadratic isoperimetric inequality? The answer is negative, as Thurston proved that the nilpotent groups

$$\langle a_1, b_1, ..., a_n, b_n, c \mid [a_i, b_i]c^{-1}, [a_i, c], [b_i, c], [u_i, u_j]\rangle$$

where $u_i$ and $u_j$ represent $a_i$ or $b_i$ and $i \neq j$, have quadratic isoperimetric inequalities if $n \geq 2$.

The nilpotent groups are generally not automatic, as proved by Holt and stated on [Far92, Page 13], the following result holds:

**Theorem 5.3.2.** *Let $G$ be a finitely generated nilpotent group. Suppose $G$ is automatic. Then it is has a finite index subgroup that is abelian.*

Speaking of growth function of nilpotent groups, we have the well known following result. For a proof, see [ECH$^+$92, Lemma 8.2.6]. The following result was also used as part in the proof of Theorem 5.3.2.

**Lemma 5.3.3.** *A finitely generated nilpotent group $G$ has polynomial growth.*

## 5.4 Baumslag–Solitar Groups

In this section, we examine another class of non-examples of automatic groups—namely, the Baumslag-Solitar groups, as observed by Thurston.

**Theorem 5.4.1.** *Let $G_{p,q}$ denote the Baumslag-Solitar group with the presentation*

$$G_{p,q} = \langle x, y \mid yx^p y^{-1} x^{-q}\rangle$$

*Then $G_{p,q}$ is automatic if and only if $p = q$; otherwise, it is not automatic.*

The case $p = q$ is established by bounding the uniform distances of paths in the corresponding Cayley graph of the Baumslag-Solitar groups. By applying the Knuth-Bendix procedure, we obtain a normal form for $G_{p,p}$. Combining these results, we observe that the language of normal forms of elements of $G_{p,p}$ constitutes part of an automatic structure.

On the other hand, the non-automaticity of $G_{p,q}$ for $p \neq q$ follows from Theorem 4.1.9. Specifically, we show that $G_{p,q}$ does not satisfy a quadratic isoperimetric inequality by

constructing a sequence of loops whose lengths grow linearly while their combinatorial areas increase exponentially.

Additionally, Gersten proved the following result regarding the class of isoperimetric inequalities satisfied by Baumslag-Solitar groups [Ger92]:

**Proposition 5.4.2.** *The Baumslag-Solitar groups $G_{p,q}$ satisfy an exponential isoperimetric inequality fir $p \neq q$.*

# Bibliography


[BH99] Martin R. Bridson and André Haefliger. *Metric spaces of non-positive curvature*, volume 319 of *Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1999.

[Bie12] L. Bieberbach. Über die bewegungsgruppen der euklidischen räume. (zweite abhandlung.) die gruppen mit einem endlichen fundamentalbereich. *Mathematische Annalen*, 72:400–412, 1912.

[Can84] James W. Cannon. The combinatorial structure of cocompact discrete hyperbolic groups. *Geom. Dedicata*, 16(2):123–148, 1984.

[Ced89] Judith N. Cederberg. *A course in modern geometries*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1989.

[Cha86] Leonard S. Charlap. *Bieberbach groups and flat manifolds*. Universitext. Springer-Verlag, New York, 1986.

[Chi09] Ian Chiswell. *A course in formal languages, automata and groups*. Universitext. Springer-Verlag London, Ltd., London, 2009.

[ECH+92] David B. A. Epstein, James W. Cannon, Derek F. Holt, Silvio V. F. Levy, Michael S. Paterson, and William P. Thurston. *Word processing in groups*. Jones and Bartlett Publishers, Boston, MA, 1992.

[EHR91] D. B. A. Epstein, D. F. Holt, and S. E. Rees. The use of Knuth-Bendix methods to solve the word problem in automatic groups. volume 12, pages 397–414. 1991. Computational group theory, Part 2.

[EIFZ96] David B. A. Epstein, Anthony R. Iano-Fletcher, and Uri Zwick. Growth functions and automatic groups. *Experiment. Math.*, 5(4):297–315, 1996.

[ES00] D. B. A. Epstein and P. J. Sanders. Knuth-Bendix for groups with infinitely many rules. *Internat. J. Algebra Comput.*, 10(5):539–589, 2000.

[Far92] Benson Farb. Automatic groups: a guided tour. *Enseign. Math. (2)*, 38(3-4):291–313, 1992.

[Ger92] S. M. Gersten. Dehn functions and $l_1$-norms of finite presentations. In *Algorithms and classification in combinatorial group theory (Berkeley, CA, 1989)*, volume 23 of *Math. Sci. Res. Inst. Publ.*, pages 195–224. Springer, New York, 1992.

[Gil95] Robert H. Gilman. Automatic groups and string rewriting. In *Term rewriting (Font Romeux, 1993)*, volume 909 of *Lecture Notes in Comput. Sci.*, pages 127–134. Springer, Berlin, 1995.

[Gro87] M. Gromov. Hyperbolic groups. In *Essays in group theory*, volume 8 of *Math. Sci. Res. Inst. Publ.*, pages 75–263. Springer, New York, 1987.

[HRR17] Derek F. Holt, Sarah Rees, and Claas E. Röver. *Groups, languages and automata*, volume 88 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 2017.

[Lee18] John M. Lee. *Introduction to Riemannian manifolds*, volume 176 of *Graduate Texts in Mathematics*. Springer, Cham, second edition, 2018.

[LS77] Roger C. Lyndon and Paul E. Schupp. *Combinatorial group theory*, volume Band 89 of *Ergebnisse der Mathematik und ihrer Grenzgebiete [Results in Mathematics and Related Areas]*. Springer-Verlag, Berlin-New York, 1977.

[Mor24] Harold Marston Morse. A fundamental class of geodesics on any closed surface of genus greater than one. *Trans. Amer. Math. Soc.*, 26(1):25–60, 1924.

[Pap95] P. Papasoglu. Strongly geodesically automatic groups are hyperbolic. *Invent. Math.*, 121(2):323–334, 1995.

[Sip96] Michael Sipser. *Introduction to the Theory of Computation*. PWS Pub. Co., Boston, MA, first edition, 1996.

[Thu97] William P. Thurston. *Three-dimensional geometry and topology. Vol. 1*, volume 35 of *Princeton Mathematical Series*. Princeton University Press, Princeton, NJ, 1997.

[Thu22] William P. Thurston. *The geometry and topology of three-manifolds. Vol. IV*. American Mathematical Society, Providence, RI, [2022] ©2022. Edited and with a preface by Steven P. Kerckhoff and a chapter by J. W. Milnor.