DISSERTATION

ARITHMETIC IN GROUP EXTENSIONS USING A PARTIAL AUTOMATON

Submitted by

Ellen Ziliak

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2010

COLORADO STATE UNIVERSITY

May 11, 2010

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY ELLEN ZILIAK ENTITLED ARITHMETIC IN GROUP EXTENSIONS USING A PARTIAL AUTOMATON BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Jeff Achter

Anton Betten

Ross McConnell

Advisor: Alexander Hulpke

Department Head: Gerhard Dangelmayr

ABSTRACT OF DISSERTATION

ARITHMETIC IN GROUP EXTENSIONS USING A PARTIAL AUTOMATON

The purpose of this paper is to describe the structure of an extension group G which has a normal subgroup K and a quotient group $Q \cong G/K$. To describe the structure of G concretely, we want to be able to do arithmetic in G based on the arithmetic done in both the normal subgroup K and the quotient group Q. We will begin by looking at the 2-cohomology group which is the standard approach for working on this problem. This will lead us to questions concerning storage which we would like to reduce.

Therefore we will consider the case where our groups are finitely presented and see how storage may be reduced. During this reduction we will see that it will be necessary to be able to rewrite words in a free group as a product of generators of the normal subgroup K. We begin by looking at current approaches to this problem, which requires computing an (augmented) coset table.

If we will let Q be a finite group for which we also have a presentation $< S|R >$, (i.e. $Q \cong F/N$ with $F =< S >$ and N the normal closure of R in F). We assume that Q does not have a confluent rewriting system. We want to rewrite a word in S, representing the identity in Q as a product of conjugates in R. Such rewriting

can be done using an (augmented) coset table for N in F which can be visualized in a graph by a coset automaton, also called the full Cayley Graph. Tracing in the graph through words in F will allow us to rewrite these words as a product of generators of N. The difficulty that arises in this approach lies in storing and constructing the augmented coset table. Instead we will construct an object called a partial automaton which is a subgraph of the coset automaton. The partial automaton will have the property that it contains a loop for every relator in R. We will first show that this graph can be used to reconstruct the coset automaton, which means it contains the same information as the coset automaton even though it is much smaller.

Our next step will be to use the partial automaton to rewrite words in N as a product of conjugates in R. Since the partial automaton is much smaller than the coset automaton, and it does not contain doubly labeled edges as an augmented coset automaton would it requries substantially less memory to store.

A word in N is represented by a loop in the coset automaton, therefore if we wish to rewrite this word as a product of conjugates of relators, we essentially want to describe this larger loop as a product of smaller loops. Where we will restrict our smaller loops to be loops in the partial automaton. To do this rewrting we place the partial automaton locally at different states in the coset automaton until we cover the entire loop. By placing the partial automaton at different states in the graph we will then the conjugate of relators. Unfortunately we cannot just place the partial automaton arbitrarily at different states, because we would have many different choices of the conjugates of relators we could choose. Instead we

must use one further tool, which is the fact that our normal subgroup N is itself a free group. Therefore N has a free generating set, where the generators of N are conjugates of relators. With this generating set we can rewrite words in N uniquely as a product of the generators.

We will therefore, use the partial automaton to compute the generators of the free generating set for N and then use these generators to rewrite our word in N as a product of conjugate of relators. By using the partial automaton to do this rewriting we can quickly do rewriting in much larger examples.

This algorithm has been implemented in GAP and to suggest the improvement we rewrote several words in the group PSp_6 which is a group of order $1,451,520$. The partial automaton had 145 states and after some initial set up which will be described in the paper the run time for this rewriting took less than a half a second per word.

Ellen Ziliak
Department of Mathematics
Colorado State University
Fort Collins, CO 80523
Summer 2010

TABLE OF CONTENTS

Chapter 1

PREFACE

The purpose of this thesis is to explore arithmetic in group extensions. We will be considering extensions G of K by Q where K is a normal subgroup of G and Q is the quotient group. We will also assume that Q is given as a finitely presented group, so $Q \cong F/N$. The standard approach to arithmetic in group extensions which we will describe in Chapter 2 uses the 2-cohomology group. However, we will see that this approach will require us to know the value of 2-cocycles for every pair of elements in our quotient group Q. This requirement would be an issue if we wanted to compute extensions generically in a quotient algorithm where each value of the 2-cocycle would then be a variable. Therefore our goal will be to reduce the number of values of 2-cocycles we need to store.

In Chapter 3 we will show that we can reduce our storage requirement if we assume we have certain information about the normal subgroup K, and quotient group Q. First if we have a confluent rewriting system for each group, we will show that we only need to store a value of the 2-cocycle for each relator of Q. This is a significant reduction on the storage requirement, however the condition that we have a confluent rewriting system may be too strong. Since it might not always be feasible to get a confluent rewriting system, since confluent rewriting systems can

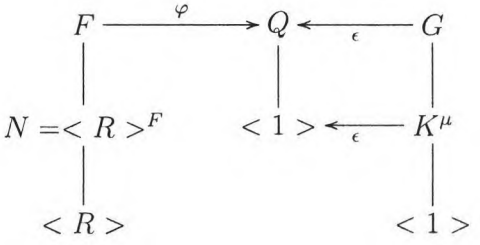
get very large. If we instead assume we have another representation for Q and K, such as a permutation representation where we can compute a normal form, then in this case, the problem can still be solved. It turns out, in this case all we need is an algorithm which takes an element in N, the normal closure of the relators for Q and writes this element as a product of generators for N.

In Chapter 4, we describe an object called the augmented coset automaton which is a graph that can be used to do this rewriting. However the difficulty in using the augmented coset automaton arises in storing and constructing this graph for large groups. We will instead use a similar object called a coset automaton which contains information about elements in N, but does not contain information about how to write those elements as a product of generators for N. In Chapter 5, we will then construct an object called the partial automaton, which will be a much smaller subgraph of the coset automaton. We will show a partial automaton contains enough information to reconstruct the coset automaton without introducing any coincidences.

Finally, in Chapter 6, we will show how a partial automaton can be used to rewrite elements in N as a product of generators for N. We will then be able to use the description in Chapter 3 to explore arithmetic in group extensions. In Chapter 7, we will discuss an adaptation to this algorithm, which will reduce the storage requirement even further. A final example which actually does arithmetic in group extensions will be the content of Chapter 8.

Since this description will include reference to several groups we will give the following diagram which shows how those groups are related. In the diagram

$F = F(S)$ will be a free group, such that Q is a finitely presented quotient group $Q \cong F/N$ which is given by the presentation $Q =< S|R >$. We will define N to be the normal closure in F of the group generated by the relators $< R >$. We will let K^{μ} be a normal subgroup of G, where G is an extension of K by Q.



Chapter 2

GROUP EXTENSIONS

Since we want to explore arithmetic in group extensions we will begin with the formal definition of a group extension. The following description is detailed in (R96, Chapter 11).

Definition 1 *If K and Q are groups, an **extension** of K by Q is a short exact sequence*

$$1 \longrightarrow K \xrightarrow{\mu} G \xrightarrow{\epsilon} Q \longrightarrow 1$$

*that means μ is a monomorphism and ϵ is an epimorphism. We will call the group G as an **extension group**.*

With this definition it is not obvious that for given K and Q, groups G with this condition exist. Therefore we begin with an example of a group G that will always exist. Let $\xi : Q \to Aut(K)$ be a homomorphism, we know such a homomorphism ξ will always exists since it could be trivial. Given ξ, the semidirect product $G = Q \ltimes_\xi K$ is an extension of K by Q. To prove this we will begin by considering the elements of the semidirect product as a set of pairs (q, k), where $q \in Q$ and $k \in K$. In this case μ can be defined by $k^\mu = (1, k)$ for all $k \in K$. We will then define ϵ such that for all $g = (q, k) \in G$, we have $g^\epsilon = (q, k)^\epsilon = q$. Using these definitions we compute $im(\mu) = (1, k)$ for every $k \in K$, and $ker(\epsilon) = \{(q, k)$ such that

$q = 1\}$. From these calculations we find $im(\mu) = ker(\epsilon)$, and therefore we have an exact sequence. We conclude that $G = Q \ltimes_\xi K$ is an example of an extension group.

For fixed groups Q and K the set of extension groups are in general not unique. However we can define an equivalence relation on extensions, such that the equivalence classes correspond to isomorphic extension groups.

Definition 2 *Let* $1 \longrightarrow K \xrightarrow{\mu} G \xrightarrow{\epsilon} Q \longrightarrow 1$ *and* $1 \longrightarrow \bar{K} \xrightarrow{\bar{\mu}} \bar{G} \xrightarrow{\bar{\epsilon}} \bar{Q} \longrightarrow 1$ *be two group extensions. A* ***morphism*** *(*α*,* β*,* γ*) from one extension to the other is defined by homomorphisms* α, β*, and* γ *such that the following diagram commutes.*

$$\begin{array}{ccccccccc} 1 & \longrightarrow & K & \xrightarrow{\mu} & G & \xrightarrow{\epsilon} & Q & \longrightarrow & 1 \\ & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \beta} & & \downarrow{\scriptstyle \gamma} & & \\ 1 & \longrightarrow & \bar{K} & \xrightarrow{\bar{\mu}} & \bar{G} & \xrightarrow{\bar{\epsilon}} & \bar{Q} & \longrightarrow & 1 \end{array}$$

We now characterize when two extensions are isomorphic when there exists a morphism $(\alpha, \beta, 1)$ such that α is an isomorphism. We will essentially prove that with these conditions β will be an isomorphism so that $G \cong \bar{G}$ and therefore the two extension groups are isomorphic.

Theorem 3 *If there is a morphism of the form* $(\alpha, \beta, 1)$ *such that* α *is an isomorphism of groups then* β *is an isomorphism and therefore the two extension groups are isomorphic. Also, the set of morphisms* $(\alpha, \beta, 1)$ *such that* α *is an isomorphism gives an equivalence relation on extension groups.*

Proof: Let us assume $(\alpha, \beta, 1)$ *is given where* α *is an isomorphism. We begin by showing* β *is an isomorphism, then we can conclude by definition that we have*

isomorphic group extensions. After that we will show that the morphism $(\alpha, \beta, 1)$ *gives an equivalence relation. Consider the following diagram:*

$$\begin{array}{ccccccccc} 1 & \longrightarrow & K & \xrightarrow{\mu} & G & \xrightarrow{\epsilon} & Q & \longrightarrow & 1 \\ & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \beta} & & \downarrow{\scriptstyle 1} & & \\ 1 & \longrightarrow & \bar{K} & \xrightarrow{\bar{\mu}} & \bar{G} & \xrightarrow{\bar{\epsilon}} & \bar{Q} & \longrightarrow & 1 \end{array}$$

By the definition of a morphism we have that β *is a homomorphism.*

Next we consider $x \in ker(\beta)$. *Since* μ *is a monomorphism there exists a unique* $j \in K$ *such that* $j^{\mu} = x$. *Hence if we calculate* $(j)^{\mu\beta} = x^{\beta} = 1$, *which by applying* $(\alpha, \beta, 1)$, *is a morphism we know that* $\mu\beta = \alpha\bar{\mu}$ *which implies* $j^{\alpha\bar{\mu}} = 1$. *Therefore as* α *and* $\bar{\mu}$ *are one to one we can conclude that* $\alpha\bar{\mu}$ *is one to one, this implies that* $j = 1$. *We can therefore conclude that* $x = 1$ *and hence the* $ker(\beta) = 1$, *so by definition* β *is one to one.*

Now we show that β *is onto. Let* $y \in \bar{G}$, *then since* $\alpha\bar{\mu}$ *is one to one there exists a unique* $k \in K$ *such that* $k^{\alpha\bar{\mu}} = y$. *Applying* $(\alpha, \beta, 1)$ *which is a morphism, gives* $\alpha\bar{\mu} = \mu\beta$ *implies that* $k^{\mu\beta} = y$. *This gives* $k^{\mu} \in G$ *such that* $(k^{\mu})^{\beta} = y$ *which shows* β *is onto. We conclude that* β *is an isomorphism.*

The final part of the proof requires that we show that the set of morphisms $(\alpha, \beta, 1)$ *such that* α *is an isomorphism form an equivalence relation on extension groups. We will say that group extensions* $1 \longrightarrow K \xrightarrow{\mu} G \xrightarrow{\epsilon} Q \longrightarrow 1$ *and* $1 \longrightarrow \bar{K} \xrightarrow{\bar{\mu}} \bar{G} \xrightarrow{\bar{\epsilon}} \bar{Q} \longrightarrow 1$ *are equivalent, denoted by* $G \sim \bar{G}$, *if there exists a morphism* $(\alpha, \beta, 1)$ *where* α *is an isomorphism. We begin by showing this relation is reflexive. Consider the morphism*

$(1,1,1)$ *where* α *and* β *are the identity map. The identity map is an isomorphism, so* $(1,1,1)$ *is a morphism between group extensions* $1 \longrightarrow K \xrightarrow{\mu} G \xrightarrow{\epsilon} Q \longrightarrow 1$ *and* $1 \longrightarrow K \xrightarrow{\mu} G \xrightarrow{\epsilon} Q \longrightarrow 1$ *so that* $G \sim G$*. Therefore, we conclude* $\sim$ *is reflexive.*

Next we consider two group extensions $1 \longrightarrow K \xrightarrow{\mu} G \xrightarrow{\epsilon} Q \longrightarrow 1$ *and* $1 \longrightarrow \bar{K} \xrightarrow{\bar{\mu}} \bar{G} \xrightarrow{\bar{\epsilon}} \bar{Q} \longrightarrow 1$ *, such that there exists a morphism* $(\alpha,\ \beta,\ 1)$ *between them. Then since* α *is an isomorphism by definition and we showed* β *is an isomorphism as well, we have a morphism* $(\alpha^{-1},\ \beta^{-1},\ 1)$ *that goes in the reverse direction between these two group extensions. Also the diagram commutes, since* $\mu\beta = \alpha\bar{\mu}$ *implies* $\alpha^{-1}\mu = \bar{\mu}\beta^{-1}$*, and similarly* $\beta\bar{\epsilon} = \epsilon$ *implies* $\bar{\epsilon} = \beta^{-1}\epsilon$*. Therefore we can conclude that the relation* $\sim$ *is symmetric.*

Finally if we have three group extensions $1 \longrightarrow K \xrightarrow{\mu} G \xrightarrow{\epsilon} Q \longrightarrow 1$ *,* $1 \longrightarrow \bar{K} \xrightarrow{\bar{\mu}} \bar{G} \xrightarrow{\bar{\epsilon}} \bar{Q} \longrightarrow 1$ *, and* $1 \longrightarrow \bar{\bar{K}} \xrightarrow{\bar{\bar{\mu}}} \bar{\bar{G}} \xrightarrow{\bar{\bar{\epsilon}}} \bar{\bar{Q}} \longrightarrow 1$ *such that there are morphisms* $(\alpha,\ \beta,\ 1)$ *and* $(\delta,\ \gamma,\ 1)$ *between them, where* α *and* δ *are isomorphisms. We can conclude also by our first argument that* β *and* γ *are also isomorphisms. Then we have the following diagram:*

$$\begin{array}{ccccccccc}
1 & \longrightarrow & K & \xrightarrow{\mu} & G & \xrightarrow{\epsilon} & Q & \longrightarrow & 1 \\
 & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \beta} & & \downarrow{\scriptstyle 1} & & \\
1 & \longrightarrow & \bar{K} & \xrightarrow{\bar{\mu}} & \bar{G} & \xrightarrow{\bar{\epsilon}} & \bar{Q} & \longrightarrow & 1 \\
 & & \downarrow{\scriptstyle \delta} & & \downarrow{\scriptstyle \gamma} & & \downarrow{\scriptstyle 1} & & \\
1 & \longrightarrow & \bar{\bar{K}} & \xrightarrow{\bar{\bar{\mu}}} & \bar{\bar{G}} & \xrightarrow{\bar{\bar{\epsilon}}} & \bar{\bar{Q}} & \longrightarrow & 1
\end{array}$$

Since the composition of isomorphisms is again an isomorphism, we conclude that $(\alpha\delta,\ \beta\gamma,\ 1)$ *is a morphism. Next, since we assume* $(\alpha,\beta,1)$ *and* $(\delta,\gamma,1)$ *are*

morphisms we have $\alpha\bar{\mu} = \mu\beta$ *and* $\delta\bar{\bar{\mu}} = \bar{\mu}\gamma$. *We combine these identities to get* $\alpha\delta\bar{\bar{\mu}} = \alpha\bar{\mu}\gamma = \mu\beta\gamma$. *A similar argument shows the other half of the diagram commutes, so we can conclude relation* $\sim$ *is transitive. Therefore we have shown that the relation* $\sim$ *is an equivalence relation.*□

Our next step will be to determine how to describe group extensions for fixed groups Q and K. If we assume $1 \longrightarrow K \xrightarrow{\mu} G \xrightarrow{\epsilon} Q \longrightarrow 1$ is an extension of K and Q so the maps μ and ϵ are fixed, then we choose another function, $\tau : Q \to G$ such that $\tau\epsilon = 1$ and τ is one to one. We will call τ the transversal function, the purpose of τ will be to choose a representative for each coset in Q. Obviously in most cases this choice of representative will not be unique, however we will require that $1_Q^\tau = 1_G$.

There are two functions related to τ: first if $q \in Q$, then define the action $k \mapsto k^{q^\tau}$ by $k \mapsto ((k^\mu)^{q^\tau})^{\mu^{-1}}$. In other words, we are using k^{q^τ} as a shorthand for $((k^\mu)^{q^\tau})^{\mu^{-1}}$. Using this function we get a map $Q \to Aut(K)$ which is defined by $q \to (k \to k^{q^\tau})$. Notice that this map is not necessarily a homomorphism. Let $q, p \in Q$, then $k^{(q\cdot p)^\tau} = ((k^\mu)^{(q\cdot p)^\tau})^{\mu^{-1}}$ and $k^{q^\tau} \cdot k^{p^\tau} = ((k^\mu)^{q^\tau})^{\mu^{-1}} \cdot ((k^\mu)^{p^\tau})^{\mu^{-1}} = ((k^\mu)^{q^\tau p^\tau})^{\mu^{-1}}$. Since τ is not a homomorphism we do not know that the coset representative for $(qp)^\tau$ is the same as the coset representative for $q^\tau p^\tau$. However, if K is abelian this is no longer an issue, since $(qp)^\tau$ and $q^\tau p^\tau$ can only differ by an element in K and since the two representatives only differ by a conjugate of an element in K this difference will cancel out. Therefore in the case where K is abelian the map $q \to (k \to k^{q^\tau})$ is a homomorphism.

The second function maps $Q \times Q \to K$ and this map describes how far τ is from a homomorphism. If τ is a homomorphism, then the extension group G is the semidirect product. For $p, q \in Q$ we define $m_{p,q} \in K$ such that $p^\tau q^\tau = (pq)^\tau m_{p,q}$ and we call $m_{p,q}$ a **2-cocycle** which is a function which maps $QxQ \to K$ defined by $(p, q) \mapsto m_{p,q}$. Using this definition we can now derive some general properties of the 2-cocycles.

Theorem 4 ***Properties of 2-cocycles***

1 $m_{1,q} = 1$ *and* $m_{q,1} = 1$

2 Let $k \in K^\mu$ *and* $p, q \in Q$ *then* $(k^{p^\tau})^{q^\tau} = (k^{(pq)^\tau})^{m_{p,q}}$

3 For $a, b, c \in Q$ *we have* $m_{a.bc} \cdot m_{b,c} = m_{ab,c} \cdot (m_{a,b})^{c^\tau}$

Proof: For (1) since $1^\tau = 1$ *we have* $1^\tau \cdot q^\tau = q^\tau \cdot 1$ *and* $q^\tau \cdot 1^\tau = q^\tau \cdot 1$. *Next for (2)*

$$(k^{p^\tau})^{q^\tau} = (q^\tau)^{-1}(k^{p^\tau})(q^\tau)$$

$$= (q^\tau)^{-1}(p^\tau)^{-1}(k)(p^\tau)(q^\tau)$$

$$= (p^\tau q^\tau)^{-1}(k)(p^\tau q^\tau)$$

$$= ((pq)^\tau m_{p,q})^{-1}(k)((pq)^\tau m_{p,q})$$

$$= m_{p,q}^{-1}((pq)^\tau)^{-1}(k)((pq)^\tau)m_{p,q}$$

$$= (k^{(pq)^\tau})^{m_{p,q}}$$

To prove property (3) we use associativity, since $(a^\tau b^\tau)c^\tau = a^\tau(b^\tau c^\tau)$ *we have by definition*

$$(a^\tau b^\tau)c^\tau = (ab)^\tau m_{a,b}c^\tau$$

$$= (ab)^\tau(c^\tau c^{-\tau})m_{ab}c^\tau$$

$$= (ab)^\tau c^\tau (m_{a,b})^{c^\tau}$$

$$= (abc)^\tau m_{ab,c}(m_{a,b})^{c^\tau}$$

On the other hand

$$a^\tau(b^\tau c^\tau) = a^\tau (bc)^\tau m_{b,c}$$

$$= (abc)^\tau m_{a,bc} m_{b,c}$$

Equating both sides we conclude that $m_{a,bc} \cdot m_{b,c} = m_{ab,c} \cdot (m_{a,b})^{c^\tau}$ □

A theorem (R96, Prop 11.1.4) by Schreier states that these two functions define an extension.

Theorem 5 ***(Schreier)***

Given groups Q,K and two maps $Q \to Aut(K)$ *and* $Q \times Q \to K$ *such that the three 2-cocycle properties are fulfilled, then* $G = \{(q,k) | q \in Q, k \in K\}$ *is a group. The multiplication is defined by* $(q,k),(p,n) \in G$*, then* $(q,k) \cdot (p,n) = (qp, m_{q,p} k^{p^\tau} n)$*. This group G has normal subgroup* $K \cong \{(1,K)\} \lhd G$ *and* $G/K \cong Q$*.*

Proof: We begin by first showing that G is a group. The identity element in G is (1,1). If we let $(q,k) \in G$*, then* $(q,k)(1,1) = (q, m_{q,1} k \cdot 1) = (q,k)$ *and* $(1,1)(q,k) = (q, m_{1,q} 1^q \cdot k) = (q,k)$*. Next the inverse of* (q,k) *is* $(q^{-1}, (k^{(q^\tau)^{-1}})^{-1} (m_{q,q^{-1}})^{-1})$*. We check this by considering*

$$(q,k)(q^{-1}, (k^{(q^\tau)^{-1}})^{-1} (m_{q,q^{-1}})^{-1})$$

$$= (qq^{-1}, m_{q,q^{-1}} k^{(q^\tau)^{-1}} (k^{(q^\tau)^{-1}})^{-1} (m_{q,q^{-1}})^{-1}) = (1,1)$$

Finally we need to show that multiplication in this group is associative.
Let $(q,k),(p,n),(h,l) \in G$*. Then consider*

$$((q,k)(p,n))(h,l) = (qp, m_{q,p} k^{p^\tau} n)(h,l)$$

$$= (qph, m_{qp,h}(m_{q,p}k^{p^\tau}n)^{h^\tau} \cdot l)$$

$$= (qph, m_{qp,h}(m_{q,p})^{h^\tau}(k^{p^\tau})^{h^\tau}n^{h^\tau}l)$$

$$(qph, m_{q,ph}m_{p,h}(k^{p^\tau})^{h^\tau}n^{h^\tau}l)$$

Recall the definition of $(k^{p^\tau})^{h^\tau} = (k^{(ph)^\tau})^{m_{p,h}}$.

$$(qph, m_{q,ph}m_{p,h}(k^{(ph)^\tau})^{m_{p,h}}n^{h^\tau}l)$$

$$= (qph, m_{q,ph}k^{(ph)^\tau}(m_{p,h})n^{h^\tau}l)$$

$$= (q,k)(ph, m_{p,h}n^{h^\tau}l)$$

$$= (q,k)((p,n)(h,l))$$

We conclude G *is a group. Next we can see clearly that* $K \cong (1,K)$ *so we only need to show* $(1,K) \lhd G$. *Let* $(p,n) \in G$ *and* $(1,k) \in (1,K)$, *we consider*

$$(p,n)^{-1}(1,k)(p,n) = (p^{-1}, (n^{(p^\tau)^{-1}})^{-1}m_{p,p^{-1}})(p, m_{1,p}k^{p^\tau}n)$$

$$= (1, m_{p,p^{-1}}((n^{(p^\tau)^{-1}})^{-1}m_{p,p^{-1}})^{p^\tau}k^{p^\tau}n) \in (1,K)$$

Therefore $(1,K) \lhd G$. *To show* $Q \cong G/K$ *we need a homomorphism* $\epsilon : G \to Q$ *with* $ker(\epsilon) \cong K$ *then by the first isomorphism theorem* $G/K \cong Q$. *Let* $\epsilon : G \to Q$ *such that* $(q,k)^\epsilon = q$. *First we show that* ϵ *is a homomorphism, let* $(q,k),(p,n) \in G$ *then* $((q,k)(p,n))^\epsilon = (qp, m_{q,p}k^{p^\tau}n)^\epsilon = qp = (q,k)^\epsilon(p,n)^\epsilon$. *Also* ϵ *is clearly onto* Q, *and* $ker(\epsilon) = \{(q,k) \in G | (q,k)^\epsilon = 1_Q\} = (1,K)$, *so by the first isomorphism theorem* $G \cong Q/K$.$\square$

Our next step will be to show an example of how to describe all group extensions. In this example we want to compute all possible extensions when

$Q \cong K \cong C_2$ where $C_2 =< a|a^2 = 1 >$. First notice that $Aut(C_2) =< 1 >$, since the automorphism group is trivial $k^{q^\tau} = k^1$ for all $q \in Q$. Next we consider the second map, the 2-cocycles. Here we must have $m_{1,1} = 1, m_{1,a} = 1$, and $m_{a,1} = 1$ by our first property. Therefore the only thing we have left to consider is the value of $m_{a,a}$.

We have only two options for this value since $K \cong C_2$, either $m_{a,a} = 1$ or $m_{a,a} = a$. Before we see what group extensions correspond to these 2-cocycles we must first check that both options satisfy the 2-cocycle properties. We only have properties (2) and (3) left to check since we begin with property (1) satisfied. For property (2), let $k \in C_2^\mu$ and $p, q \in C_2$, then $(k^{p^\tau})^{q^\tau} = (k^{(pq)^\tau})^{m_{p,q}}$. This property holds since $k^{q^\tau} = k$, which gives $k = (k)^{m_{p,q}} = m_{p,q}^{-1} k m_{p,q} = m_{p,q}^{-1} m_{p,q} k = k$ since in this example K is abelian. Next we must show that for all $a, b, c \in Q$ property (3) $m_{x,yz} m_{y,z} = m_{xy,z} (m_{x,y})^{z^\tau}$ holds. In the case where $m_{a,a} = 1$, this property is trivially satisfied since $1 \cdot 1 = 1 \cdot 1$. Now suppose $m_{a,a} = a$, then since $Q = C_2$ we only have two choices for x,y,z they can be a or 1. The following chart verifies this identity holds for each choice of x,y, and z.

x	y	z	$m_{x,yz} m_{y,z}$	$m_{xy,z} (m_{x,y})^{z^\tau}$
1	1	1	$1 \cdot 1$	$1(1)^1$
1	1	a	$1 \cdot 1$	$1(1)^1$
1	a	1	$1 \cdot 1$	$1(1)^1$
a	1	1	$1 \cdot 1$	$1(1)^1$
1	a	a	$1 \cdot a$	$a(1)^1$
a	1	a	$a \cdot 1$	$a(1)^1$
a	a	1	$a \cdot 1$	$1(a)^1$
a	a	a	$1 \cdot a$	$1(a)^1$

This chart shows $m_{a,a} = a$ is also a valid choice since property (3) holds for all possibilities of a,b and c.

In the first case, if $m_{a,a} = 1$, then the value of all 2-cocycles is one. The multiplication in G is defined by $(q,k)(e,n) = (qe, k^{e^\tau} n) = (qe, kn)$, since $Aut(C_2) =< 1 >$. In this case we have τ is a homomorphism, and therefore $G = C_2 \times C_2$. Generally this would give the semidirect product, if the $Aut(K)$ is nontrivial.

In the second case where $m_{a,a} = a$, to identify this group let us look at the order of elements. Consider $x = (a, 1)$, then $x^2 = (a^2, m_{a,a} 1^a \cdot 1) = (a^2, a) = (1, a) \neq (1, 1)$. Next we consider $x^3 = (a, m_{a,1} 1^1 a) = (a, a) \neq (1, 1)$, and $x^4 = (a^2, m_{a,a} a^1 \cdot 1) = (a^2, a^2) = (1, 1)$, so we conclude G has an element of order 4. Since $|G| = 4$ we can conclude that $G \cong C_4$ in this case.

We will now restrict ourselves to the case where K is abelian. In this case, as we mentioned before the first map $Q \to Aut(K)$ is a homomorphism. We can think of K as a Q-module where the action of Q on K is given by the map $q \to (k \to k^{q^\tau})$. Also, when we are working with 2-cocycles property (2): $(k^{p^\tau})^{q^\tau} = (k^{(pq)^\tau})^{m_{p,q}}$ is no longer necessary, since $(k^{p^\tau})^{q^\tau} = (k^{(pq)^\tau})$, because the 2-cocycle $m_{p,q}$ cancels off when K is abelian. Therefore in the case where K is abelian, 2-cocycles only have to have property (1) and (3). We will begin by looking at 2-cocycles and show that they form a group called $Z^2(Q, K)$, the group of **2-cocycles**.

Theorem 6 *For a particular action $Q \to Aut(K)$ the set of possible 2-cocycles form a group $Z^2(Q, K)$, the group of 2-cocycles. The arithmetic in $Z^2(Q, K)$ is defined by $(mm^*)_{a,b} = m_{a,b} \cdot m^*_{a,b}$, where m and m^* are different 2-cocycles for the*

extension of K by Q, which correspond to the maps $\tau : Q \to G$ and $\tau^* : Q \to G$ respectively.

Proof: The identity element of this group is the 2-cocycle defined by $m_{a,b} = 1$ for all $a, b \in Q$. This 2-cocycle will occur in the case where τ is a homomorphism which gives the semidirect product as the extension.

Next if, $m_{a,b}$ defines a 2-cocycle, then $m_{a,b}^{-1}$ also defines a 2-cocycle. To prove this we will need to show that the two properties of a 2-cocycle hold for $m_{a,b}^{-1}$. We begin with property (1): $m_{1,q}^{-1} = (m_{1,q})^{-1} = (1)^{-1} = 1$ for all $q \in Q$. Similarly $m_{q,1}^{-1} = 1$, so we conclude that property (1) holds for $m_{a,b}^{-1}$. Next we must show that property (3) holds for $m_{p,q}^{-1}$ which comes from the fact that property (3) holds for $m_{p,q}$. Since $m_{a,bc}m_{b,c} = m_{ab,c}(m_{a,b})^{c^\tau}$ by taking inverses we can conclude that $m_{b,c}^{-1}m_{a,bc}^{-1} = (m_{a,b}^{-1})^{c^\tau}m_{ab,c}^{-1}$, and since K is abelian we have $m_{ab,c}^{-1}(m_{a,b}^{-1})^{c^\tau} = m_{a,bc}^{-1}m_{b,c}^{-1}$ which gives us property (3).

The next step to this proof is to show that the 2-cocycles are closed under multiplication in $Z^2(Q, K)$. We begin first by showing that the product of two 2-cocycles is again a 2-cocycle. Let m, m^* be two 2-cocycles along with their associated maps $\tau : Q \to G$ and $\tau^* : Q \to G$ respectively, then we must show that the two properties of 2-cocycles hold for $m \cdot m^*$. Consider $(m \cdot m^*)_{1,q} = m_{1,q} \cdot m_{1,q}^* = 1 \cdot 1 = 1$ and similarly $(m \cdot m^*)_{q,1} = 1$, so we can conclude that property (1) holds. For property (3) consider

$$(mm^*)_{a,bc}(mm^*)_{b,c} = m_{a,bc}m_{a,bc}^*m_{b,c}m_{b,c}^*$$

$$= m_{a,bc}m_{b,c}m_{a,bc}^*m_{b,c}^*$$

$$= m_{ab,c}(m_{a,b})^{c^\tau} m^*_{ab,c}(m^*_{a,b})^{c^{\tau^*}}$$

Since K is abelian

$$= m_{ab,c}m^*_{ab,c}(m_{a,b})^{c^\tau}(m^*_{a,b})^{c^{\tau^*}}$$

The map $Q \to G$ which corresponds to mm^* is $\tau \cdot \tau^*$ the product of the two maps.

$$= m_{ab,c}m^*_{ab,c}(c^\tau)^{-1}(m_{a,b})(c^\tau)(c^{\tau^*})^{-1}(m^*_{a,b})(c^{\tau^*})$$

$$= m_{ab,c}m^*_{ab,c}(c^{\tau^*})^{-1}(c^\tau)^{-1}(m_{a,b})(m^*_{a,b})(c^\tau)(c^{\tau^*})$$

$$= m_{ab,c}m^*_{ab,c}(c^\tau c^{\tau^*})^{-1}(m_{a,b})(m^*_{a,b})(c^\tau c^{\tau^*})$$

$$= m_{ab,c}m^*_{ab,c}(c^{\tau\cdot\tau^*})^{-1}(m_{a,b})(m^*_{a,b})(c^{\tau\cdot\tau^*})$$

$$= (mm^*)_{ab,c}((mm^*)_{a,b})^{(c^{\tau\cdot\tau^*})}$$

Therefore we can conclude that the product of 2-cocycles is again a 2-cocycle. Now if we have three 2-cocycles m, m^*, and $\overline{m}$, then since for all $p, q \in Q$, we have $m_{p,q}, m^*_{p,q}$ and $\overline{m_{p,q}}$ are elements of K we see that clearly $(m_{p,q}m^*_{p,q})\overline{m_{p,q}} = m_{p,q}(m^*_{p,q}\overline{m_{p,q}})$ by the associativity of K. Hence we can conclude that $Z^2(Q,K)$ is a group.$\square$

We have established that the set of 2-cocycles form a group $Z^2(Q,K)$, the group of 2-cocycles. Our next step it to consider the function $\tau : Q \to G$ along with the associated 2-cocycle function m, which is an integral part of the definition of a given 2-cocycle. Suppose we are given a different map $\tau^* : Q \to G$ along with its associated 2-cocycle function m^*, where $q^{\tau^*} = q^\tau \cdot c_q$ and $c_q : Q \to K$. We can assume that τ^* has this form because a different transversal function will just change the coset representatives. We now want to see how $m_{a,b}$ is related to $m^*_{a,b}$.

Let us consider $a^{\tau^*}b^{\tau^*}$ in two different ways: first there is a 2-cocycle m^* so by definition $a^{\tau^*}b^{\tau^*} = (ab)^{\tau^*}m^*_{a,b}$. Now since $f^{\tau^*} = f^\tau \cdot c_f$ we get $a^{\tau^*}b^{\tau^*} = (ab)^\tau c_{ab}m^*_{a,b}$. If instead we use the fact that $f^{\tau^*} = f^\tau \cdot c_f$ first we get

$$a^{\tau^*}b^{\tau^*} = a^\tau c_a b^\tau c_b$$

$$= a^\tau (b^\tau b^{-\tau}) c_a b^\tau c_b$$

$$= a^\tau b^\tau (c_a)^{b^\tau} c_b$$

$$= a^\tau b^\tau (c_a)^b c_b$$

For ease of notation we will leave off the τ on $(c_a)^b$, however b is the image of the element b in the group G under the map τ.

$$= (ab)^\tau m_{a,b}(c_a)^b c_b$$

Now setting these two equations equal we get

$$m^*_{a,b} = c^{-1}_{ab} m_{a,b}(c_a)^b c_b = m_{a,b} c^{-1}_{ab}(c_a)^b c_b$$

since all the elements are in K which is an abelian group. Notice that m^* differs from m by something that depends on c. We will now show that the difference between m and m^* describes a subgroup of $Z^2(Q, K)$.

Definition 7 *Let* $B^2(Q,K) = \{c^{-1}_{ab}(c_a)^b c_b | c : Q \to K\} \subseteq \{Q \times Q \to K\}$ *this set is called the* ***2-coboundries****.*

Lemma 8 $B^2(Q,K) \leq Z^2(Q,K)$

Proof: First we show $B^2(Q,K) \subseteq Z^2(Q,K)$*. To do this we need to show that the functions* $c : Q \to K$ *fulfill the two 2-cocycle properties.*

Given $c : Q \to K$, *let* $m_{a,b} = c_{ab}^{-1}(c_a)^b c_b$, *then* $m_{1,q} = c_q^{-1}(c_1)^q c_q = c_1 = 1$ *since* $1^{\tau^*} = 1 = 1^\tau \cdot c_1$. *Similarly* $m_{q,1} = c_q^{-1}(c_q)^1 c_1 = 1$, *so we have fulfilled the requirements of property (1). For property (3) we have*

$$m_{a,bc}m_{b,c} = c_{abc}^{-1}(c_a)^{bc}c_{bc}c_{bc}^{-1}(c_b)^c c_c$$

$$= c_{abc}^{-1}((c_a)^b c_b)^c c_c$$

$$= c_{abc}^{-1}(c_{ab})^c(c_{ab}^{-1})^c((c_a)^b c_b)^c c_c$$

$$= c_{abc}^{-1}(c_{ab})^c c_c(c_{ab}^{-1}(c_a)^b c_b)^c$$

$$= m_{ab,c}(m_{a,b})^c$$

$$= m_{ab,c}(m_{a,b})^{c^\tau}$$

We now add in the map τ *which was suppressed in our computations to be consistent with our previous definition. Therefore we can conclude that* $B^2(Q,K) \subseteq Z^2(Q,K)$.

Finally we need to show that $B^2(Q,K)$ *is a subgroup of* $Z^2(Q,K)$. *The argument for closure and inverses of elements in* $B^2(Q,K)$ *is the same as the argument given in Theorem 6 which did not depend on how* m *and* m^* *were defined, so letting* $m_{a,b} = c_{ab}^{-1}(c_a)^b c_b$ *and* $m_{a,b}^* = d_{ab}^{-1}(d_a)^b d_b$ *for two maps* $c : Q \to K$ *and* $d : Q \to K$ *will not change the argument. Therefore we can conclude that* $B^2(Q,K) \leq Z^2(Q,K)$.□

Definition 9 *Let* $H^2(Q,K) = Z^2(Q,K)/B^2(Q,K)$, *then* $H^2(Q,K)$ *is called the* ***second cohomology group*** *for the action of* Q *on* K.

This group $H^2(Q,K)$ is the group we want to compute when we are trying to describe all extensions groups for a fixed Q and K.

Theorem 10 *Let Q and K be fixed along with a fixed action $Q \to Aut(K)$, then extension groups up to morphism are in bijection with $H^2(Q,K)$ for the same action.*

Proof: We begin by recalling Schreier's Theorem 5 which states we can represent all extension groups $G = \{(q,k)|q \in Q, k \in K\}$ along with two maps $Q \to Aut(K)$ defined by $q \to (k \to k^{q^\tau})$ and the map $Q \times Q \to K$ defined by $(p,q) = m_{p,q}$. Let E be the set of equivalence classes of extension groups G for a fixed Q and K. We will construct a bijection between E and the group $Z^2(Q,K)$. The map will take the group G that has 2-cocycles defined by $m_{a,b}$ to the 2-cocycle defined to be $m_{a,b}$ for each $a,b \in Q$. To show this is a bijection we need to show this map is one to one and onto. If we call the bijection φ then $ker(\varphi) = \{G \in E|(G)^\varphi = 1\}$, so if $G \in ker(\alpha)$ we have $m_{a,b} = 1$ for all $a,b \in Q$. Then the multiplication in G for $(q,k),(p,n) \in G$ is defined as $(q,k)\cdot(p,n) = (qp, m_{q,p}k^{p^\tau}n) = (qp, k^{p^\tau}n)$. This is the definition of a group extension $G = Q \ltimes_\xi K$ where $\xi = k^{q^\tau} : Q \to Aut(K)$ is the given action. Since this is the identity element in E, we have α is injective. Our next step is to show α is onto. Let $m_{a,b}$ be a 2-cocycle in $Z^2(Q,K)$, then using Schreier's theorem we can then construct a group extension G with $m_{a,b}$ being the 2-cocycles. Therefore the map is onto.

Our next step is to relate this bijection to the equivalence classes of group extensions. In Theorem 3 we showed that two group extensions $G \sim \overline{G}$ were in the same equivalence class if we had a morphism $(\alpha,\beta,1)$ such that α is an

isomorphism. Consider the following diagram:

$$\begin{array}{ccccccccc} 1 & \longrightarrow & K & \xrightarrow{\mu} & G & \xrightarrow{\epsilon} & Q & \longrightarrow & 1 \\ & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \beta} & & \downarrow{\scriptstyle 1} & & \\ 1 & \longrightarrow & \bar{K} & \xrightarrow{\bar{\mu}} & \bar{G} & \xrightarrow{\bar{\epsilon}} & \bar{Q} & \longrightarrow & 1 \end{array}$$

Now we want to consider what happens to an element $(q,k) \in G$ under the isomorphism β. We will let elements of $G = \{(q,k) | q \in Q^{\tau}, k \in K^{\mu}\}$ and elements of $\bar{G} = \{[q,k] | q \in \bar{Q}^{\bar{\tau}}, k \in \bar{K}^{\bar{\mu}}\}$ so we can distinguish the different groups. We begin by considering an element of $K^{\mu} = (1,k)$ under the map β: $(1,k)^{\beta} = [1,k]$ where $[1,k] \in \bar{K}^{\bar{\mu}}$. We are using the same letter k in this case since $K \cong \bar{K}$. In reality the element $k = k^{\alpha}$, but for convenience we will just call the element k. Next we consider an element $(q,1)^{\beta} = [q,c_q]$ where $c_q \in \bar{K}^{\bar{\mu}}$. The element must look like this since commutativity of the diagram gives $\beta\bar{\epsilon} = \epsilon$, and ϵ takes an element $(q,k)^{\epsilon} = q$ where $\bar{\epsilon}$ is defined so that $[q,k]^{\bar{\epsilon}} = q$. Therefore by looking at both of our previous computations we can conclude that $(q,k)^{\beta} = [q, c_q k]$. Now we want to consider the effect of β on pairs of elements in G. Let $(q,k),(p,n) \in G$, then we consider

$$((q,k)\cdot(p,n))^{\beta} = (qp, m_{q,p}k^{p^{\tau}}n)^{\beta} = [qp, c_{qp}m_{q,p}k^{p^{\tau}}n]$$

or similarly since β is a homomorphism we could multiply these two elements in $\bar{G}$ instead

$$(q,k)^{\beta}\cdot(p,n)^{\beta} = [q,c_q k]\cdot[p,c_p n] = [qp, \overline{m_{q,p}}(c_q k)^{p^{\bar{\tau}}}c_p n]$$

Now equating $((q,k)(p,n))^{\beta} = (q,k)^{\beta}(p,n)^{\beta}$ we get $c_{qp}m_{q,p}k^{p^{\tau}}n = \overline{m_{q,p}}(c_q k)^{p^{\bar{\tau}}}c_p n$. Since K is abelian, we can manipulate this equation to get $m_{p,q} = \overline{m_{p,q}}c_{qp}^{-1}(c_q)^{p^{\tau}}c_p$ which is exactly our 2-coboundry condition. Which implies two extension groups G and $\overline{G}$ are in the same equivalence class exactly when there 2-cocycles differ by

an element in $B^2(Q,K)$. *Therefore* φ *actually gives a bijection between the group extensions and the group of 2-cocycles* $H^2(Q,K)$.$\square$

Now that we have shown that the equivalence classes of group extensions are in bijection with the elements of the group $H^2(Q,K)$, this gives us a way to describe extension groups. However, since our primary goal will be to do arithmetic in an extension group G for a fixed Q and K, we will need more information then just the equivalence classes of group extensions. We will need to know how the maps μ, ϵ, and τ are defined.

One issue we mentioned that arises is that for a given τ we would need to know the values of the 2-cocycles $m_{a,b}$ for every pair of a, b elements in Q to compute the product $a^\tau b^\tau$ of two elements in the extension group G. We would like to look for an equivalence version of τ where we can choose τ so we have a good way to describe the value of the 2-cocycles $m_{a,b}$. Ideally we would like to choose τ so that we can reduce the number of values of 2-cocycles we must store.

If we have a concrete extension group G for a fixed K and Q, then the maps μ and ϵ are fixed. We would like to define τ so that as many as possible of the values of the 2-cocycles are trivial. Since we do not always have a split extension, we know that most of the time we cannot make all of the 2-cocycles trivial. The claim is that one way we can do this is to assume we have some way to compute a normal form (NF) to represent every element in Q uniquely as a product of the generators of Q. The choice of the normal form word can be arbitrary though we would require that if $w = NF(u) = vx$, then the $NF(v) = v$. In other words,

the normal form of a subword of a word in normal form is that subword. If $NF(w) = x_1x_2\cdots x_n$, then define $(x_1x_2\cdots x_n)^\tau = x_1^\tau x_2^\tau \cdots x_n^\tau \cdot 1$ so the values of the 2-cocycles when forming normal form words are trivial. In this way we have made τ as close to a homomorphism as possible without G being the semidirect product. Since extensions are uniquely defined by μ and ϵ making a choice for the form of τ does not effect the extension, so we are free to define τ as we wish.

With this assumption on τ, if we assume we have a black box operation which takes a word in Q and puts it in normal form, then for any product of words in Q we can compute the value of the 2-cocycles. For example, take any product of words $u \cdot v$ in Q, we can input each word into the black box $\overline{u} = NF(u)$, $\overline{v} = NF(v)$, and $\overline{(u \cdot v)} = NF(u \cdot v)$ and this gives us a way to compute the value of the 2-cocycles. Since $\overline{(u \cdot v)}^\tau = \overline{u}^\tau \cdot \overline{v}^\tau$, we therefore have $u^\tau v^\tau = (uv)^\tau \overline{u} \cdot \overline{v} \cdot \overline{(u \cdot v)}^{-1}$ so the value of the 2-cocycle $m_{u,v} = \overline{u} \cdot \overline{v} \cdot \overline{(u \cdot v)}^{-1}$.

Chapter 3

FINITELY PRESENTED GROUPS

Now that we have a description that reduces the amount of information we must store to do arithmetic in a fixed group extension G. Our next step will be to consider the case where Q and K are finitely presented groups. We begin by giving a formal definition of a finitely presented group. We will let $F = F(S)$ be a free group.

Definition 11 *Let R be a nonempty subset of F, the* ***normal closure*** *N of R in F is the intersection of all the normal subgroups of F which contain R. Often N is denoted by R^F.*

By this definition it is clear N is the smallest normal subgroup of F containing R. Now we will give a precise definition of a finitely presented group. In (DF99, Sect 6.3) a finitely presented group is defined in the following way.

Definition 12 *Let S be a subset of a group G, where G is a finitely presented group.*

1 *A* ***presentation*** *for G is a pair $< S|R >$, where R is a set of words in $F(S)$ (i.e. the free group F with generators S) such that the normal closure N of R in $F(S)$ equals the kernel of the homomorphism $\pi : F(S) \to G$ (where π extends the identity map from S to S). The elements of S are called the* ***generators*** *and those of R are called the* ***relators*** *of G.*

2 *We say G is* ***finitely generated*** *if there is a presentation* $< S|R >$ *such that S is a finite set, and we say G is* ***finitely presented*** *if there is a presentation* $< S|R >$ *where both S and R are finite sets.*

Now we will discuss how one computes a normal form for a word in a finitely presented group. We begin by taking a presentation with relators R, a set of words $r_i = \epsilon$. We can then we manipulate these relators to make them into rewriting rules. For example, if a group G has a relator $xy = \epsilon$ then we can rewrite this as a rule $y \to x^{-1}$. To know which way the arrow must go, we must have a well ordering on elements of the free group F. We then must have the property that the left hand side of the arrow must represent a larger word with respect to the well ordering then the right hand side of the arrow. In this way we will define our normal form word as the smallest representative for a word. In general if $R =< r_1 = r_2 = \ldots = r_n = \epsilon >$, then a rewriting system for R could be $\tilde{R} = \{r_1 \to \epsilon, r_2 \to \epsilon, \ldots, r_n \to \epsilon\}$. If we can use this rewriting system to compute a normal form and the order in which we apply the rewriting rules does not matter then we say the rewriting system is confluent. For a more precise definition of confluence see (S94).

Definition 13 *A rewriting system is* ***confluent*** *if the order in which one applies the rewriting rules to compute a normal form does not effect the resulting word.*

We begin by supposing we have a confluent rewriting system for K and Q, then we can develop a rewriting system for G, as long as we store a value of the 2-cocycle for each relator for the group Q. Before we prove this claim let us look at an example. Suppose the confluent rewriting system for $Q = V_4 =< r, s|r^2 \to 1, s^2 \to 1, rs \to sr >$ and $K = C_2 =< a|a^2 \to 1 >$, then to get a rewriting system

for G the extension of K by Q we will begin with the rules for Q and lift them to G by using the values of the 2-cocycles.

We need to know the values of the 2-cocycles $m_{r,r} = a, m_{s,s} = 1$, and $m_{r,s} = a$ to get a rewriting system for G. It should be pointed out that the left hand side of the rewriting rules do not need to be words of length 2, but rather the $m_{r,r}$ are 2-cocycles in the sense that the image of the right hand side of the rule under τ is equal to the left hand side of the rule under τ times the 2-cocycle in G. These choices for the values of the 2-cocycles uniquely determine τ, so the extension group G is now fixed. Consider first the relator $r^2 \to 1$. Under τ we have:

$$r^\tau r^\tau = (r^2)^\tau m_{r,r}$$

Therefore we can conclude that $r^2 \to a$ is a relator in G. We will get two other relators using this procedure, and we add the relators for K to get the following rewriting system for G, $\{r^2 \to a, s^2 \to 1, rs \to sra$, and $a^2 \to 1\}$. Using Tietze Transformations we can change our presentation for G from $< r, s, a | r^2 = a, s^2 = 1, rs = sra, a^2 = 1 >$ to $< r, s | r^4 = 1, s^2 = 1, rs = sr^3 >$ which is the presentation for D_8. Now if we want to use our rewriting system for G, we can compare our answer with the words in D_8 to see that we in fact get the correct solution. The final step necessary to compute the normal form in G, is that normal form words will have elements which are images under Q^τ before elements which are images under K^μ. Therefore if we have a word in G such that this is not the case, for example $k \cdot q$ where $k \in K^\mu$ and $q \in Q^\tau$, we will apply the rule $k \cdot q \to qk^q$ where k^q is the action on K by Q defined previously. With these rules we can rewrite any word in G in normal form. For example, if we wish to rewrite $rss \in G$.

$$\underline{rs}s \to sr\underline{a}s \to srsa^s = srsa.$$

We conclude $srsa^s = srsa$ since the action on a is trivial because $Aut(K) =< 1 >$. Then

$$\underline{sr}\underline{s}a \rightarrow \underline{ss}raa \rightarrow r\underline{aa} \rightarrow r$$

Therefore we can conclude that the normal form for rss is r. Indeed these are the same elements in D_8.

Theorem 14 *If G is a finitely presented group, such that $K^\mu \lhd G$ and $Q \cong G/K^\mu$ and we also we assume a confluent rewriting system already exists for K and Q. Then if we know the values of the 2-cocycles for each rewriting rule in Q we can develop a confluent rewriting system for G.*

Proof: Suppose the confluent rewriting system for Q is given by $Q =< x_1, x_2, \ldots x_r | R_1 \rightarrow L_1, R_2 \rightarrow L_2, \ldots, R_k \rightarrow L_k >$, *and suppose we have the values of the 2-cocycles $m_{R_1}, m_{R_2}, \ldots, m_{R_k}$ for each relator in Q. These elements are values of 2-cocycles in the sense that each $m_{R_j} \in K^\mu$, and $(R_j)^\tau = L_j^\tau m_{R_j}$ is how multiplication is defined in G. Also, suppose that K has a confluent rewriting system $K =< y_1, \ldots, y_m | K_1 \rightarrow J_1, K_2 \rightarrow J_2, \ldots K_n \rightarrow J_n >$. We can then develop a confluent rewriting system for G, with respect to the wreath product order. Here by wreath product order we mean that we will compare the Q parts of a word first, and the K parts of the word second. If there are elements in the word that are images of K^μ before images of Q then those words are larger then all other words where the Q parts come first. Consider an element in G given by $g = g_1 g_2 \cdots g_n$. Since G is an extension of K by Q for each g_i, if $g_i^\epsilon = 1$, then $g_i \in K^\mu$, otherwise $g_i \in Q^\tau$. One step in the rewriting process will be to move all $g_i \in Q^\tau$ to the left. To do this suppose $g_i \in Q^\tau$ and $g_{i-1} \in K^\mu$ then $g_{i-1} g_i \rightarrow g_i g_{i-1}^{g_i}$, we can conclude $g_{i-1}^{g_i} \in K^\mu$ since $K^\mu \lhd G$. In this way we can rewrite $g = q_1 \cdots q_l k_1 \cdots k_m$, where*

$q_i \in Q^\tau$ *and* $k_i \in K^\mu$. *Now, since the rewriting systems for* Q *and* K *are confluent we will apply the rules mapped under* τ *and* μ *respectively to each part of* g *in any order we wish. Each time we apply a rule for a subword of* $q_1 \cdots q_l$ *we must multiply by the appropriate value of the 2-cocycle. Once no further rules can be applied we say the resulting word is the normal form of* g. *This rewriting system is confluent since the rewriting systems were derived from were confluent, so the order in which we apply the rules does not matter.* □

In the previous theorem we are only required to store a value of the 2-cocycle for each relator, but what if we want to describe the second cohomology group? We would need to know a value of the 2-cocycle for each pair of elements in Q. We can show now with the rewriting system for G that every value of the 2-cocycle can be computed, so that it is only necessary to store a value of the 2-cocycle for each relator of Q.

Theorem 15 *Given a finitely presented group* G *such that* $K \lhd G$ *and* $Q \cong G/K$ *where we assume a confluent rewriting system already exists for* K *and* Q. *Then if we know values of the 2-cocycles for each relator in* Q *we can compute a value of the 2-cocycle for any pair of elements* $p, q \in Q$.

Proof: Suppose we want to compute $m_{p,q}$ *for* $p, q \in Q$. *We would begin by computing the normal form in* Q *for the words* p, q, *and* pq. *We will denote the normal form by* $\overline{p}$, $\overline{q}$, *and* $\overline{pq}$. *Then since we know* $m_{p,q}$ *is defined by* $p^\tau q^\tau = (pq)^\tau m_{p,q}$, *and since we can define* τ *such that it maps generators to generators, therefore the generators of* Q *are again in the generators of* G . *Hence we have the relation* $\overline{p} \cdot \overline{q} = \overline{pq} m_{p,q}$ *which implies that* $m_{p,q} = \overline{p} \cdot \overline{q} \cdot \overline{pq}^{-1}$. *In this way we can compute*

the value of the 2-cocycle for any pair of elements $p, q \in Q$. $\square$

Our next step will be to look at the case where we do not have a confluent rewriting system for Q. However, we will assume that we are able to obtain the normal form of elements in Q by some other procedure. For example, we could assume we also have a faithful permutation representation for Q, and we define the normal form word to be the smallest length-lexicographic representative. These representatives can be computed using enumeration of the elements of Q, and the following article (CFS90) describes how one can store two-bits for each element of Q.

We will now briefly describe the algorithm which one can use to compute the length-lexicographic representative of a given word from a faithful permutation representation. The algorithm will require that we enumerate all of the elements of the group Q by storing two lists: count and distance. Where the distance will be defined to be the length of the length-lexicographic representative of the word modulo 3. The list count is used to assign a unique integer in the range 0 to $|Q|-1$. Therefore each entry in count will uniquely represent a group element.

Now we will discuss how these lists are constructed. Initially the distance list D, will have $D[i] = 3$ for $1 \leq i \leq |Q| - 1$ and $D[0] = 0$ where we assume count(ϵ)=0. We will then update D so that if count(q)=i, then $D[i]$ is the length modulo 3 of the length-lexicographic representative of q. The elements are enumerated in count using an orbit algorithm where we order our generators according to the lexicographic ordering defined for the generators of Q. At the same time we

will update our list D, until all values of D are 0,1 or 2.

Finally to compute the length-lexicographic representation for $q \in Q$ once D has been constructed is now a simple task. First we define a parent of a node q to be any element which has distance to the identity modulo 3 to be one less that q. Notice this implies their may not be a unique parent for each $q \in Q$. The only node which has no parent will be ϵ the trivial word. Now to compute the length-lexicographic representative for $q \in Q$ we must choose a parent node as the successor of each new word we compute. Each time a new word is computed, the distance is diminished by one unit, and since ϵ has no parent the product will eventually terminate. To compute the parent of q we will run through the generators $\bar{x}$ of Q and check the values of $D[count(q\bar{x})]$. If q is not equal to ϵ, then there exists a parent node $q\bar{x}_1$ of q. This process is then repeated with q replaced by $q\bar{x}_1$. Eventually we will get $q\bar{x}_1 \cdots \bar{x}_k = \epsilon$, which implies $q = \bar{x}_k \cdot \bar{x}_1$, which is the word of shortest length which represents q. To guarantee this is the length-lexicographic representative we must test our generators of Q in reverse lexicographic order. Since that way the inverse of q will be as large as possible which implies q will be as small as possible.

We now conclude this description with an example. Let $Q = D_8 =< r = (1,2,3,4), s = (1,2)(3,4) >$ and the lexicographic ordering on the generators be $[r, s, r^{-1}, s^{-1}]$ then the length-lexicographic representatives of the elements of Q are $[1, r, s, r^{-1}, r^2, rs, sr, r^2 s]$. We have $count = [1,2,3,4,5,6,7,8]$ and $D = [0,1,1,1,2,2,2,0]$ and suppose we want the length-lexicographic representative of $q = rsr^{-1} = (1,4)(2,3)$. We will run through our generators in reverse lexicographic order $[s^{-1}, r^{-1}, s, r,]$ and compute $D(q\bar{x})$ for each of these. Initially we

have $D(q) = 0$, so we want to find a generator $\bar{x}$ such that $D(q\bar{x}) = 2$.

q	$\bar{x}$	**q** $\bar{x}$	**count(q$\bar{x}$)**	**D(q$\bar{x}$)**
(1,4)(2,3)	$s^{-1} = (1,2)(3,4)$	(1,3)(2,4)	4	2
(1,4)(2,3)	$r^{-1} = (1,4,3,2)$	(1,3)	6	2
(1,4)(2,3)	$s = (1,2)(3,4)$	(1,3)(2,4)	4	2
(1,4)(2,3)	$r = (1,2,3,4)$	(2,4)	5	2

All of these products have distance 2, but we choose s^{-1} because that is first in our reverse lexicographic ordering. Now we consider $qs^{-1} = (1,3)(2,4)$, where $D(qs^{-1}) = 2$ so we want to find a generator $\bar{x}$ such that $D(qs^{-1}\bar{x}) = 1$.

qs^{-1}	$\bar{x}$	**q** $s^{-1}\bar{x}$	**count(q**$s^{-1}\bar{x}$**)**	**D(q**$s^{-1}\bar{x}$**)**
(1,3)(2,4)	$s^{-1} = (1,2)(3,4)$	(1,4)(2,3)	7	0
(1,3)(2,4)	$r^{-1} = (1,4,3,2)$	(1,2,3,4)	1	1
(1,3)(2,4)	$s = (1,2)(3,4)$	(1,4)(2,3)	7	0
(1,3)(2,4)	$r = (1,2,3,4)$	(1,4,3,2)	3	1

Here only two generators work, but we choose r^{-1} since it comes first in the reverse lexicographic order. Next we consider $qs^{-1}r^{-1} = (1,2,3,4)$ where $D(qs^{-1}r^{-1}) = 1$, so we want to find a generator $\bar{x}$ such that $D(qs^{-1}r^{-1}\bar{x}) = 0$.

q$s^{-1}r^{-1}$	$\bar{x}$	**q** $s^{-1}r^{-1}\bar{x}$	**count(q**$s^{-1}r^{-1}\bar{x}$**)**	**D(q**$s^{-1}r^{-1}\bar{x}$**)**
(1,2,3,4)	$s^{-1} = (1,2)(3,4)$	(2,4)	5	2
(1,2,3,4)	$r^{-1} = (1,4,3,2)$	()	0	0
(1,2,3,4)	$s = (1,2)(3,4)$	(2,4)	5	2
(1,2,3,4)	$r = (1,2,3,4)$	(1,3)(2,4)	4	2

In this case there is only one choice for $\bar{x}$ which is r^{-1}, so we get $qs^{-1}r^{-1}r^{-1} = \epsilon$ which implies the length-lexicographic representative for $q = r^2s$. Since it is well known that a length-lexicographic order is a well-ordering, we know a smallest representative will always exist and be unique.

We would like to develop a procedure that takes a word in a fixed extension group G and maps this element to its normal form representative which was defined in Theorem 14. The first rewriting rule will be the rule which moves an element in Q^τ past an element in K^μ. The procedure to do this operation will be defined as it was in Theorem 14. Once we have our word in G written as $g = g_1 \cdots g_l k_1 \cdots k_m$, where $q_i \in Q^\tau$ and $k_i \in K^\mu$. We then want a procedure which takes the product $q_1 \cdots q_l$ and calculates the normal form for this representative. We then want a procedure that would compute the 2-cocycles from the 2-cocycles given for the relators of Q in a nice way. Our final step will then be to compute the normal form in K of the product of the 2-cocycles and $k_1 \cdots k_m$. The product of these two normal forms will give us the normal form for the word in G.

Our next step will be to give a more detailed description on how the rewriting process will work. Suppose $g \in G$ such that $g = w \cdot v$ and $w, v \in Q^\tau$ then our first step will be to compute the normal form of $(w \cdot v)^\epsilon$ in Q. Since we assume we have a way to compute the normal form of elements in Q, suppose $u \in Q$ such that $u = NF((w \cdot v)^\epsilon)$. Now we can compute the 2-cocycle needed to compute the normal form of $g \in G$. Since $g = ((w \cdot v)^\epsilon)^\tau = u^\tau m_{w,v}$ and by Theorem 15 $m_{w,v} = w \cdot v \cdot u^{-1}$ where $m_{w,v} \in K^\mu$.

However, since we assume we are only storing 2-cocycles for the relators of Q our next step is to give a way to compute the value of $m_{w,v}$ for arbitrary elements w, v in normal form from the stored 2-cocycles. The way we compute this value is to rewrite $w \cdot v \cdot u^{-1}$ as a product of conjugates of the relators for Q. We will do this rewriting in the free group F, where $Q \cong F/N$, so that $m_{w,v} = \prod_{i=1}^{n} r_i^{x_i}$ where

r_i is a relator for Q and $x_i \in F$. To make this more clear, let $\varphi : F \to Q$ be the natural homomorphism, then we want $((m_{w,v})^{\epsilon})^{\varphi^{-1}} = \prod_{i=1}^{n} r_i^{x_i}$.

Then if we consider the map $\tilde{\varphi} : F \to \tilde{G}$ where $\tilde{G} \leq G$ and $G =< \tilde{G}, \tilde{K} >$ and $\tilde{K} \leq K$, we get that $\tilde{\varphi}$ is related to φ such that the following diagram commutes.

$$F \xrightarrow{\tilde{\varphi}} \tilde{G} \xrightarrow{\beta} G \xrightarrow{\epsilon} Q, \quad F \xrightarrow{\varphi} Q$$

In G we can rewrite $g = w \cdot v = u^{\tau}((\prod_{i=1}^{n} m_{r_i}^{x_i})^{\tilde{\varphi}})^{\beta}$ where each m_{r_i} is a 2-cocycle which was stored for a relator r_i. The only part of this argument which may not be clear, is that the map $\beta : \tilde{G} \to G$ will not effect the 2-cocycles. However, an element of G can be written as a product of words in $\tilde{G}$ and $\tilde{K}$ therefore to put a word in normal form we will have to move elements of $\tilde{K}$ to the right. During this process we will not add any new 2-cocycles, so everything needed to rewrite is known. Therefore, the fact that $\tilde{\varphi}$ maps to $\tilde{G}$ instead of G does not effect the number of values of 2-cocycles we need to store.

Now that we have outlined the basic procedure for rewriting a word in G, if we do not have a confluent rewriting system. We see that all we have left to do is to describe an algorithm which rewrites a word in F as a product of conjugates of the relators given for Q.

Chapter 4

COSET AUTOMATA

The standard approach used to rewrite a word in F, which lies in N as a product of conjugates of relators, is to construct an augmented coset table for N in F. We give a brief description of this process and refer the reader to (S94) for further details.

The first step will be to construct a coset table. The fundamental idea behind this algorithm is that we perform an orbit algorithm on the cosets of a subgroup N. This gives the images of generators of F under a permutation representation, which we will list in a table called the coset table. Notice this construction will require us to have subgroup generators of N, since the relators alone do not generate a normal subgroup of F.

One issue that may arise during this construction is that we might define two different cosets which are really the same since we do not have an element test for N. We will notice this duplicate definition when we trace through each generator for N starting at the trivial coset. When two different definitions arise we say we have a coincidence, and we set the two cosets equal to each other updating the

table in the process.

We now show an example of a coset table for $F/N = S_3$ where the free group $F =< a, b >$, and $N =< a^2, b^2, (ab)^3, (b^2)^a, ((ab)^3)^a, (a^2)^b, ((ab)^3)b, (b^2)^{ab} >$ the normal closure of $R =< a^2, b^2, (ab)^3 >$.

	a	a^{-1}	b	b^{-1}
1	2	2	3	3
2	1	1	4	4
3	5	5	1	1
4	6	6	2	2
5	3	3	6	6
6	4	4	5	5

One thing to note is that the numbering of cosets depends on the generators of the subgroup N, therefore in general the numbering of a coset table is not unique. One can run a standardization process after the coset table is computed to make the table unique, however since this standardization is not necessary for the following description we will only refer the reader to (S94) for further details.

Our next step will be to describe how to compute an object which we will call the augmented coset table. An augmented coset table is a coset table where we attach to each image in the coset table the word in N which corresponds to that image. Since the generators of F are not the generators of N, we cannot tell how an image is defined as a word in the generators of N in the standard coset table. The purpose of the augmented coset table is to give this information.

Definition 16 *We say a relator R* ***traces through*** *a coset table $\mathcal{C}$ from the state γ if $\gamma^R = \gamma$ in $\mathcal{C}$.*

The algorithm for computing an augmented coset table requires that each time an image is defined by tracing completely through a generator of N, we attach to that image the name of the corresponding generator of N. We can then use the augmented coset table if we want to rewrite a word in N, as a product of generators in N. We do this by tracing through the images of the augmented coset table collecting the augmented words. The resulting word will give the word rewritten in N as a product of generators of N. This algorithm works for any generating set for N, therefore in particular we can choose a generating set in which the generators are conjugates of relators for Q. Let us look at an example. Suppose $Q = S_3$ as before, and let the generators of $N =< x_1 = a^2, x_2 = b^2, x_3 = (ab)^3, x_4 = (b^2)^a, x_5 = ((ab)^3)^a, x_6 = (a^2)^b, x_7 = ((ab)^3)^b, x_8 = (b^2)^{ab} >$, then the augmented coset table for F/N is given below:

	a	a^{-1}	b	b^{-1}
1	2	$x_1^{-1}2$	3	$x_2^{-1}3$
2	$x_1 1$	1	4	$x_1 x_4^{-1} x_1^{-1} 4$
3	5	$x_2 x_6^{-1} x_2^{-1} 5$	$x_2 1$	1
4	6	$x_1 x_4 x_5^{-1} x_2 x_6 x_8 x_3^{-1} 6$	$x_1 x_4 x_1^{-1} 2$	2
5	$x_2 x_6 x_2^{-1} 3$	3	$x_2 x_6 x_8 x_3^{-1} 6$	$x_2 x_6 x_3^{-1} 6$
6	$x_3 x_8^{-1} x_6^{-1} x_2^{-1} x_5 x_4^{-1} x_1^{-1} 4$	4	$x_3 x_6^{-1} x_2^{-1} 5$	$x_3 x_8^{-1} x_6^{-1} x_2^{-1} 5$

[1]

From this augmented table if we wanted to rewrite $ab^{-1}a^{-1}bab^{-1}$ as a product of generators for N we trace

$$1^{ab^{-1}a^{-1}bab^{-1}} = 2^{b^{-1}a^{-1}bab^{-1}}$$

$$= x_1 x_4^{-1} x_1^{-1} 4^{a^{-1}bab^{-1}}$$

[1]It should be noted in the augmented table we switched to writing the 2-cocycle on the left, this was done so that inverse 2-cocycles are easier to read. If we left the 2-cocycles on the right as they were in the description of group extensions, then inverse 2-cocycles would be conjugated by the corresponding generator.

$$= x_1 x_4^{-1} x_1^{-1} x_1 x_4 x_5^{-1} x_2 x_6 x_8 x_3^{-1} 6^{bab^{-1}}$$

$$= x_1 x_5^{-1} x_2 x_6 x_8 x_3^{-1} x_3 x_6^{-1} x_2^{-1} 5^{ab^{-1}}$$

$$= x_1 x_5^{-1} x_2 x_6 x_8 x_6^{-1} x_2^{-1} x_2 x_6 x_2^{-1} 3^{b^{-1}}$$

$$= x_1 x_5^{-1} x_2 x_6 x_8 x_2^{-1} 1$$

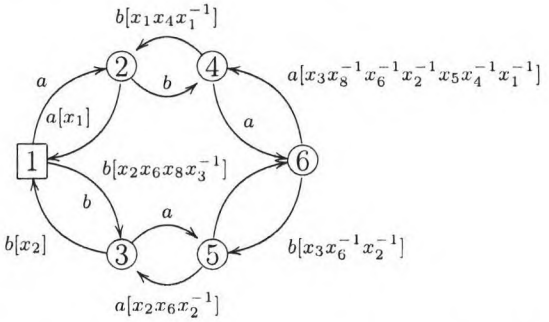
Therefore we can rewrite $ab^{-1}a^{-1}bab^{-1} = x_1 x_5^{-1} x_2 x_6 x_8 x_2^{-1}$.

Now we want to do is describe a nice way to visualize an augmented coset table and that is with a directed graph called a coset automaton.

Definition 17 *A **coset automaton** for a group $Q \cong F/N$ is a directed graph that consists of a set of vertices Σ one for each coset fN, and edges E. An edge is defined from coset i to coset j if in the coset table there exists a generator x of F; where the image of i under x is j, denoted $i^x = j$. We then label this edge with the generator x. We will identify the initial coset as the identity element. It should be noted that this is just a graphical representation of the coset table.*

We assign a secondary label of the image in the generators of N to each edge which we will call the augmented edge. If the graph has augmented edges, then the graph is called an **augmented coset automaton**. Therefore the augmented coset automaton for our example of S_3, with $N =< x_1 = a^2, x_2 = b^2, x_3 = (ab)^3, x_4 = (b^2)^a, x_5 = ((ab)^3)^a, x_6 = (a^2)^b, x_7 = ((ab)^3)^b, x_8 = (b^2)^{ab} >$ looks like the following figure. To make the image easier to read we will not include inverse edges, as they

can be determined by tracing a positive edge in reverse.



At this point it might also be useful to introduce a few definitions from graph theory to make the rest of the description more clear. These hold because a coset automaton is essentially a digraph where states are vertices.

Definition 18 *A **walk** in a digraph Γ is an alternating sequence of vertices and edges, beginning and ending with a vertex, where each vertex is incident to both the edge that precedes it and the edge that follows it in that sequence, and where the vertices that precede and follow an edge are the vertices of that edge.*

Definition 19 *When we **trace through** a word w through the coset automaton, we begin at the trivial coset and define a walk where we follow the edge labeled by the first letter of w. Then from the state at which we arrive we leave following the second letter of w. This process will continue until we have followed the edge labeled by the last letter of w, at which point we will arrive at some state in the coset automaton.*

Definition 20 *A **path** in Γ is a walk in which no vertices (states) and thus no edges are repeated.*

Definition 21 *A **loop** is a path which begins and ends at the same vertex (state) in the graph Γ.*

Now with these definitions, we will now prove that coset enumeration is a valid procedure, and therefore if we have a finite group Q then coset enumeration will terminate. The following result comes from (M64).

Theorem 22 *Assume the index of H in Q is finite. Then the coset enumeration procedure as defined in (N82) for H in Q will eventually terminate.*

Proof: Suppose we have an enumeration procedure where the following 5 properties are fulfilled:

(i) $1 \in \Sigma$ and the coset representative for 1 is the empty word ϵ

(ii) $\alpha^x = \beta$ if and only if $\beta^{x^{-1}} = \alpha$

(iii) If $\alpha^x = \beta$, then if the coset representative for α is A, and the coset representative for β is B, we have the coset HAx=HB

(iv) For all $\alpha \in \Sigma$ if $1^A = \alpha$ then any other word V=A in Q/H. We must have $1^V = \alpha$

(v) Let $\overline{\Sigma}$ be the set of elements in Σ that represent unique cosets, then for each $\alpha \in \overline{\Sigma}$ and for each $x \in S \cup S^{-1}$ we have α^x is at some point defined.

If these properties are fulfilled we will show that the enumeration procedure will eventually terminate. We notice that any valid strategy for coset enumeration will fulfill these five properties.

Suppose the coset enumeration procedure for Q/H does not terminate. If the enumeration procedure has these five properties fulfilled, then for each $\alpha \in \overline{\Sigma}$ at some point α^x is defined as $\alpha^x = \beta$. Since α is in $\overline{\Sigma}$ we know α represents a unique

coset in Q/H. Now we do not know if $\beta \in \overline{\Sigma}$ and therefore represents a unique coset. However, at some point we will either determine β is unique or $\beta = \gamma$, where γ is a coset that has been previously defined. So we have $\gamma < \beta$. Therefore, since α^x can only decrease it will eventually become stable. Thus $\overline{\Sigma}$ must be infinite, because otherwise the coset table would full with $\Sigma = \overline{\Sigma}$.

We can now define an infinite full coset table in which the rows are indexed by the elements $\alpha \in \overline{\Sigma}$ and the entries of α^x are their stable values. Property (v) ensures that the coset table gives a permutation representation of the group Q/H. Since the coset table is infinite and full we can therefore conclude that the action of Q, by right multiplication on the cosets of H, has infinite degree. But this contradicts the assumption that $|Q : H|$ is finite, since Q is a finite group and H is a subgroup of Q. Therefore the coset enumeration procedure for Q/H will eventually terminate.□

Notice that the proof of this theorem does not give a bound on how long it will take to terminate, it only shows that eventually the procedure will terminate. In fact, it has been proven that there cannot be an algorithm which would terminate with a given bound, for any bound one gives there exists a presentation for which the coset enumeration procedure violates this bound.

In the example we gave of the augmented coset automaton, the generators of the group N included conjugates of the relators as well the relators themselves. We will now show in an example that conjugates are really necessary to represent every word in N. If we consider $< R >$ the group generated by the relators of Q, then if

$F/ < R >$ generated Q we could write every element in Q as a product of elements in R. To do this rewriting we would apply coset enumeration to construct a coset table for $F/ < R >$ which would give the multiplication table for Q. We will see however in the following example that enumeration will not generally terminate, which will allow us to conclude that the subgroup $< R >$ has infinite index in F. We will then show instead that we need to construct the coset table for F/N to get the multiplication table for Q.

We will let $Q = D_8 =< r, s | r^4 = s^2 = s^{-1}rsr = 1 >$ then $F =< r, s >$, the free group generated by r and s. The subgroup $< R >=< r^4, s^2, s^{-1}rsr >$. The only constraint in defining new cosets will be that each generator of R must map the trivial coset to itself. After defining the first 6 cosets we will get the following edge table:

	r	r^{-1}	s	s^{-1}
1	2	3	4	4
2	5	1		
3	1	5		6
4	6		1	1
5	3	2		
6		4	3	

At this point each relator will map the trivial coset to itself so we have no further constraints. Now notice that by defining the image of any coset representative under increasing powers of r will always define a new coset. Therefore we can conclude that the coset enumeration for this example will never terminate with a full edge table, and hence in this case every word in Q cannot be written as a product of elements of R.

Since we have seen that we cannot rewrite every word in Q as a product of relators, we now want to show that every word in Q can algorithmically be written as a product of conjugates of relators. In other words we want to show the algorithm returns the multiplication table for Q given by a coset automaton constructed for F/N.

Definition 23 *A coset automaton is* ***full****, if for every state* $\sigma \in \Sigma$ *and for each* $x \in S \cup S^{-1}$ *the image state* σ^x *is uniquely defined.*

We will now prove that a coset automaton is full if and only if every coset is mapped to itself under each relator.

Definition 24 *We say a coset automaton is* ***relator closed****, if it has the property that each coset is mapped to itself under every relator and inverse relator.*

Theorem 25 *A coset automaton for a finite group Q is full if and only if it is relator closed.*

Proof: First if a coset automaton for Q is full, then for each $\sigma \in \Sigma$ *and each* $x \in S \cup S^{-1}$ *the image state* σ^x *is defined. Therefore at each state* $\sigma \in \Sigma$ *and for each relator or inverse relator r, the walk* σ^r *is defined. Suppose there exists a* $\sigma \in \Sigma$ *such that* $\sigma^r \neq \sigma$ *so that the coset automaton would not be relator closed. Define* $\sigma = 1^S$ *where S is a word in Q, so we have* $1^{Sr} \neq 1^S$*. This implies that* $SrS^{-1} \neq \epsilon$ *in Q, but since r is a relator, or an inverse of a relator, we have* $r = \epsilon$ *so* $SrS^{-1} = SS^{-1} = \epsilon$*, a contradiction. Therefore* $\sigma^r = \sigma$ *for all r in the relators or their inverses. Hence if a coset automaton for Q is full, then it is relator closed.*

For the converse we will show that the same 5 properties identified in Theorem 22 are fulfilled when we have a coset automaton that is relator closed. We could then conclude by Theorem 22 that the coset enumeration will terminate, and the coset automaton for Q is full.

We begin with property (i): the coset 1 is defined and 1 represents the trivial coset. This is verified by the definition of a coset automaton. Next property (ii): $\alpha^x = \beta$ if and only if $\beta^{x^{-1}} = \alpha$. This property holds because every time we define an edge, the inverse edge is also defined to guarantee each relator and its inverse maps every coset to itself. For property (iii): If $\alpha^x = \beta$ where $1^A = \alpha$ and $1^B = \beta$, then we need to show that the word represented by $Ax{=}B$. Since $\alpha^x = \beta$ this means $1^{Ax} = 1^B$ which implies that $1^{AxB^{-1}} = 1$, therefore $AxB^{-1} = \epsilon$ the identity element in Q. This tells us that $Ax = B$ as words. Now we look at property (iv): for all $\alpha \in \Sigma$, if α is defined there exists a word A such that $1^A = \alpha$. Now if in Q there is another word $V{=}A$, we must have $1^V = \alpha$. By definition $\alpha = 1^A = 1^V$ so $1^V = \alpha$. Therefore, we have verified properties (i)-(iv) which was a fairly trivial verification.

The only nontrivial property we need to verify is property (v): for each $\alpha \in \Sigma$ and each $x \in S \cup S^{-1}$ we have α^x will at some stage be defined. Let $\alpha = 1^A$, then if α^x is not defined at some point there exists an element AxB^{-1} of N, where $Q \cong F/N$, since $AxB^{-1} \in N$ we know that $AxB^{-1} = \epsilon$ in F/N. We are assuming this element cannot be traced through the coset automaton. Now since N is generated by relators $\{r_1, \ldots, r_n\}$, and conjugates of the relators where we have the property that for each $\gamma \in \Sigma$ that $\gamma^{r_i} = \gamma$, the word AxB^{-1} can be written as a product of conjugates of relators since it is an element of N. So $AxB^{-1} = r_{i_1}^{g_i} \cdots r_{i_k}^{g_k}$.

If we let $\gamma_i = 1^{g_i^{-1}}$, *then* $1^{AxB^{-1}} = 1^{r_{i_1}^{g_1} \cdots r_{i_k}^{g_k}} = 1^{g_1^{-1} r_{i_1} g_1} \cdots 1^{g_k^{-1} r_{i_k} g_k} = \gamma_1^{r_{i_1}} \cdots \gamma_k^{r_{i_k}}$. *Since the coset automaton is relator closed this implies* 1^{Ax} *must be defined to be* 1^B, *since each* $\gamma_j^{r_{i_j}}$ *is defined to be* γ_j. *Therefore, if the coset automaton is relator closed, then the coset automaton is full as long as* Q *is finite.* □

The problem we had with $F/ < R >$ is that the coset automaton we constructed for $F/ < R >$ will not be relator closed. Now we will prove a result on when termination occurs in the free group case.

Theorem 26 *Let F=F(S) be a free group. Let* $Q =< S|R >$ *be a finitely presented finite group. Then coset enumeration will terminate for the free group* F *with normal subgroup* $N=< R >^F$.

Proof: Since Q *is a finite group let* H *be the trivial subgroup. Then by theorem 22 coset enumeration for* Q/H *will terminate. Upon termination the coset table will be full, so therefore the coset automaton for* Q *will be relator closed.*

Now consider $N =< R >^F$ *the normal closure of* $< R >$. *Since* $< R >$ *is the group generated by the relators, coset enumeration will have the property that every relator maps the trivial coset to itself. The generating set for the subgroup* N *is larger than the generating set for* $< R >$ *in that they also include conjugates of relators. Therefore, coset enumeration will force each of these conjugates of relators to map the trivial coset to itself. However, the element we conjugate by* g *will have a unique coset representative* $\overline{g}$ *in the coset enumeration, therefore by adding this conjugate we are forcing the coset represented by* $\overline{g}^{-1}$ *to map to itself under each relator. Hence, coset enumeration for* F/N *will require the coset automaton*

to be relator closed, which therefore gives a full coset automaton. Also the coset automaton for F/N will be isomorphic to the coset automaton for Q/H, which is finite, so therefore this enumeration will terminate. □

Chapter 5

PARTIAL AUTOMATA

Our first goal will be to see if it is possible to construct the coset automaton for F/N without having to compute a generating set for the subgroup N. Notice in the proof of Theorem 26, we argued that for the coset automaton to be full it must be relator closed. Therefore, if we know the path taken by each relator from the trivial coset we can conclude that the entire automaton should have the same set of paths defined from each state. Our next step will be to prove that this holds.

Before we give the proof, we must first define what it means for an automaton to be isomorphic to itself when we consider different initial states.

Definition 27 *If we let $\mathcal{C}$ be the full coset automaton for F/N with initial state 1 the trivial coset and let $\mathcal{D}$ be the same coset automaton with a different initial state γ, then we say $\mathcal{C} \cong \mathcal{D}$ if there exists a one to one function $f : States(\mathcal{C}) \to States(\mathcal{D})$ such that any pair of states α, β which are adjacent $\alpha^x = \beta$ if and only if $f(\alpha)^x = f(\beta)$.*

Theorem 28 *Define $\mathcal{C}$ and $\mathcal{D}$ as they were defined in the previous definition. Then $\mathcal{C} \cong \mathcal{D}$*

Proof: We need to construct a one to one function $f : States(\mathcal{C}) \to States(\mathcal{D})$ *such that any pair of states* α, β *that are adjacent* $\alpha^x = \beta$ *if and only if* $f(\alpha)^x = f(\beta)$.

Since $\mathcal{D}$ *is the same coset automaton beginning at a different state* $\gamma \neq 1$, *then there exists an element* $G \in F/N$ *such that* $1^G = \gamma$. *We will then define* $f : States(\mathcal{C}) \to States(\mathcal{D})$ *such that if* $\mu \in States(\mathcal{C})$ *such that* $1^M = \mu$ *we have* $f(\mu) = (1^G)^M$.

Now we need to show that this function f is one to one. Suppose $1^M = \mu$ *and* $1^N = \nu$ *and* $f(\mu) = f(\nu)$, *then we conclude* $(1^G)^M = (1^G)^N$. *We can manipulate this equation to get* $(1^G)^{MN^{-1}G^{-1}} = 1$ *which means* $GMN^{-1}G^{-1} = \epsilon$ *in* F/N. *Hence* $GM = GN$ *so we conclude that* $M = N$. *Therefore* $\mu = \nu$ *so f is one to one.*

Next let $\alpha \in States(\mathcal{D}) = States(\mathcal{C})$, *then there exists an* $A \in F/N$ *such that* $1^A = \alpha$. *Now by definition of a full coset automaton, there exists a* $\beta \in States(\mathcal{C})$ *such that* $\beta = 1^{G^{-1}A}$, *since for any word* $GA^{-1} \in F/N$ *there must be a path in* $\mathcal{C}$ *from 1 to* GA^{-1}. *We then have* $f(\beta) = (1^G)^{G^{-1}A} = 1^A = \alpha$. *Therefore we can conclude that f is onto.*

Finally we must show if there exist a pair of states $\alpha, \beta \in States(\mathcal{C})$ *such that* $\alpha^x = \beta$ *then* $f(\alpha)^x = f(\beta)$. *Suppose* $\alpha^x = \beta$ *then there exists* $A, B \in F/N$ *such that* $1^A = \alpha$ *and* $1^B = \beta$, *so therefore* $1^{AxB^{-1}} = 1$. *Now suppose* $f(\alpha)^x = \delta$ *where* $\delta \neq f(\beta) = (1^G)^B$ *and* $1^D = \delta$. *Then* $f(\alpha) = (1^G)^A$, *so* $(1^G)^{AxD^{-1}} = 1$ *which implies that* $GAxD^{-1} = \epsilon$. *We can manipulate this equation to* $G = Dx^{-1}A^{-1}$.

Also $1^{AxB^{-1}} = 1$ *which implies* $AxB^{-1} = \epsilon$ *therefore* $xB^{-1} = A^{-1}$. *Plugging in this equation for* G *we get* $G = Dx^{-1}xB^{-1} = DB^{-1}$. *Therefore* $D = GB$ *which implies that* $1^D = 1^{GB} = f(\beta)$. *A contradiction, hence* $f(\alpha)^x = f(\beta))$ *so we have an isomorphism.* □

From Theorem 28 it now makes sense to construct an object which we will call a partial automaton, that describes only the paths taken by each relator in R in the full coset automaton. In the rest of the description that follows we will assume that we have a fixed free group $F = F(S)$ and a finite finitely presented group $Q =< S|R >$. Also we assume $\mathcal{C}$ is the full coset automaton.

Definition 29 *A* ***partial automaton*** $\mathcal{A}$ *is a subgraph of the full coset automaton, including the initial state 1, with the property that each relator from the initial coset traces through in* $\mathcal{A}$.

With this definition, it is clear that for some states in $\mathcal{A}$ we won't have all the edges which leave that state in $\mathcal{C}$ defined in $\mathcal{A}$. We will therefore classify the states into two categories: complete and incomplete.

Definition 30 *The* ***complete*** *states of* $\mathcal{A}$ *will be those which have a known edge for every element in* $S \cup S^{-1}$. *The* ***incomplete*** *states* α *will be those where there exists an element* x *in* $S \cup S^{-1}$ *such that* α^x *is unknown.*

Our goal will be to show that the partial automaton contains enough information to reconstruct the full coset automaton from it by using the isomorphism established in Theorem 28 and updating the partial automaton until all of the edges are complete. We can assume without loss of generality, that 1 the trivial

coset is complete, which means that we will require our presentation to contain at least one relator that starts with each element of $S \cup S^{-1}$. If this is not initially the case, then cyclic conjugates of the given relators can be added to the presentation until this property holds. The necessity of this condition will become clear in the following lemma.

Lemma 31 *A partial automaton is full if every state is complete.*

Proof: To show a partial automaton is a full coset automaton we must show that for all $\sigma \in \Sigma$ and for all $x \in S \cup S^{-1}$, the image state σ^x is uniquely defined. If every state in the partial automaton is complete, we know for all states σ in the partial automaton σ^x is uniquely defined for all $x \in S \cup S^{-1}$. Therefore, we only need to show that there does not exist a σ in the full coset automaton that is not in our partial automaton. If β is a state in the full coset automaton, then there exists a word $B \in F/N$ such that $1^B = \beta$. If $B = b_1 \cdots b_k$ then we know 1^{b_1} must have been defined, since we assume the trivial state is complete which implies that the state 1^{b_1} must exist in our partial automaton. Then by the completeness of each state we have that $1^{b_1 b_2}$ must also be defined so that $1^{b_1 b_2}$ must also be a state in the partial automaton. Continuing in this fashion we find that $1^B = \beta$ must be a state in the partial automaton. Therefore if every state in the partial automaton is complete, then the partial automaton is full. □

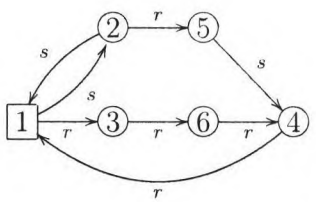
Our next step will be to describe how we can construct a particular partial automaton for $Q \cong F/N$ which we will use to reconstruct the full automaton. We will begin by constructing an initial partial automaton by labeling the cosets to determine the edges up to the point where each relator maps the trivial coset to

itself. When defining new cosets we will use the element test in Q to ensure we have a unique definition for each coset. Therefore we are guaranteed that we have not defined the same coset twice.

First we will see an example, look at the partial automaton which we will use for $D_8 =< r, s | r^4 = s^2 = s^{-1}rsr = 1 >$. By defining edges up to the point where each relator maps the trivial coset to itself, and we have guaranteed each state defined represents a unique element in Q gives us the following table.

	r	r^{-1}	s	s^{-1}
1	2	3	4	4
2	5	1		
3	1	5		6
4	6		1	1
5	3	2		
6		4	3	

This table defines an automaton with 6 states which looks like the following figure where we will not write in the inverse edges so the picture is easier to read:



Our next step will be to give an algorithm which uses the partial automaton to reconstruct the full coset automaton. This is clearly a desirable property if we want to be able to use the partial automaton for rewriting. Once we show this reconstruction can be achieved, our next step will be to show how the partial automaton can be used to do the same rewriting described using the augmented

coset automaton.

Before we give a detailed description of the algorithm to reconstruct the full coset automaton $\mathcal{C}$ from the partial automaton $\mathcal{A}$, we will give a brief description abstractly of what the algorithm will do.

To begin this description we will introduce a function which will help clarify the process. The partial automaton $\mathcal{A}$ consists of a set of paths for each relator in R. Since $\mathcal{C}$ is a full coset automaton, we have proven in Theorem 25 that $\mathcal{C}$ is relator closed. Therefore we can define a function: $\mathcal{A}$ embeds into $\mathcal{C}$ under the map which takes the state 1 to the state σ, which we will represent by $\mathcal{A} \subseteq_{1\to\sigma} \mathcal{C}$. If we define $\sigma = 1^S$, where S is an element of the free group F, then the map $\mathcal{A} \subseteq_{1\to\sigma} \mathcal{C}$ takes any other state $\beta = 1^B$ in $\mathcal{A}$ to 1^{SB} in $\mathcal{C}$. Since $\mathcal{C}$ is full this map is well-defined.

Our aim is to start with $\mathcal{A}$ and add information to this graph until we have all of $\mathcal{C}$. To do this we begin with our partial automaton $\mathcal{A}$, and we let $\mathcal{B} = \mathcal{A} \subseteq_{1\to\sigma} \mathcal{C}$ be a new partial automaton where we add any information obtained by embedding $\mathcal{A}$ to $\mathcal{C}$ under the map $1 \to \sigma$. However, when we do this construction we assume $\mathcal{C}$ is not already known. Therefore for each β a state in $\mathcal{A}$ where β is defined so that $1^B = \beta$, if we cannot determine uniquely that $1^{SB} \neq \gamma$ where γ is a state already known in $\mathcal{A}$, then we do not label 1^{SB} in $\mathcal{B}$. The claim is that if we repeat this process at every state σ in $\mathcal{A}$ we will reconstruct $\mathcal{C}$. Notice that here the set of states in σ in $\mathcal{A}$ could grow as we update our partial automaton. The proof of this will be given after the formal description.

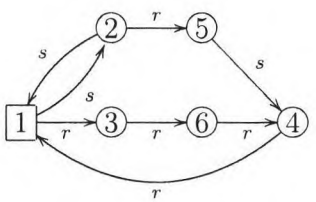
To begin the formal description we will define a function $p(\mathcal{A}, \gamma, w)$ which takes as input the partial automaton $\mathcal{A}$, an initial state γ, and a word w. The function maps words in F to states in $\mathcal{A}$ by tracing the letters of the word w through the partial automaton $\mathcal{A}$, and returning the state at which we end. We define $p(\mathcal{A}, \gamma, w) = \emptyset$ if there exists a letter x in w, where $w = DxA$, such that $\gamma^D = \delta$ and δ^x is currently undefined in $\mathcal{A}$.

In our example, the initial partial automaton $\mathcal{A}$ has $p(\mathcal{A}, 1, r^2) = 6$, but $p(\mathcal{A}, 1, sr^2) = \emptyset$. To begin reconstructing the full coset automaton we will go through each existing state α, and try to set $p(\mathcal{A}, \alpha, r) = \alpha$ for each r a relator, or an inverse of a relator. If this equality holds for each state $\alpha \in \Sigma$, then our partial automaton is full by Theorem 25, and therefore we would have reconstructed $\mathcal{C}$ the full coset automaton from $\mathcal{A}$.

We must test our states in a systematic way. To begin we test states whose coset representatives are words of the smallest size. Since we have chosen a well ordering on F, we will choose the element which is minimal with respect to this well ordering as our starting point. Once we have ensured that all relators and their inverses map that state to itself for that coset, we will continue by choosing the next state whose coset representative is the next largest size, where the size of a coset representative is defined by the well-ordering of Q. Eventually we will run through every state, at which point the updated partial automaton $\mathcal{A}$ will be relator closed. We can therefore conclude by Theorem 25, $\mathcal{A}$ will be full. Also since $\mathcal{A}$ is full by Lemma 31 every state is complete. At this point we will have reconstructed $\mathcal{C}$.

Suppose α is the first state such that there exists a relator or inverse of a relator r, where $p(\mathcal{A}, \alpha, r) \neq \alpha$. Then $r = BxD^{-1}$, where $p(\mathcal{A}, \alpha, B) = \beta$, but $p(\mathcal{A}, \alpha, Bx) = \emptyset$. We must determine if $p(\mathcal{A}, \alpha, Bx) = \sigma$ for some $\sigma \in \Sigma$, or if $p(\mathcal{A}, \alpha, Bx)$ is a new state. Essentially we are trying to determine if this edge should go to a state we already know exists or if it goes to a new state. This test is the same as defining an updated partial automaton where $\alpha = 1^A$ $\mathcal{B} = \mathcal{A} \subseteq_{1 \to \alpha} \mathcal{C}$ and determining if 1^{ABx}, is equal to some other state in $\mathcal{A}$, or if 1^{ABx} represents a new unique state.

Therefore our first step is to test if $p(\mathcal{A}, \alpha, Bx) = \sigma$ for some $\sigma \in \Sigma$. To determine this we will test for all σ currently a state in $\mathcal{A}$ and all relators and inverses r, letting $p(\mathcal{A}, \sigma, r) = \sigma$ to see if we can determine the edge $p(\mathcal{A}, \alpha, Bx)$. Let us first see how this would work in our example. We begin with the partial automaton $\mathcal{A}$ which looks like the following figure.



At the state 2 we cannot trace the relator r^4 through $\mathcal{A}$. Since $p(\mathcal{A}, 2, r) = 5$ and $p(\mathcal{A}, 2, r^2) = \emptyset$, we get stuck at 2^{r^2}. Therefore, we run through each state with every relator to determine if 2^{r^2} is equal to a state already in $\mathcal{A}$, or if this edge goes to a state that we do not know yet. The following table gives this computation by listing $p(\mathcal{A}, \alpha, r)$, the path taken by the relator r both forward and backwards (which accounts for the inverse of the relator), and whether any new edges are

defined.

$p(\mathcal{A}, \alpha, r)$	**Path Forward**	**Path Backward**	**New Edges**
$p(1, \mathcal{A}, r^4) = 1$	$1 \to 3 \to 6 \to 4 \to 1$	$1 \to 4 \to 6 \to 3 \to 1$	None
$p(1, \mathcal{A}, s^{-1}rsr) = 1$	$1 \to 2 \to 5 \to 4 \to 1$	$1 \to 4 \to 5 \to 2 \to 1$	None
$p(1, \mathcal{A}, s^2) = 1$	$1 \to 2 \to 1$	$1 \to 2 \to 1$	None
$p(2, \mathcal{A}, r^4) = \emptyset$	$2 \to 5 \to ? \to ? \to 2$	$2 \to ? \to ? \to 5 \to 2$	None
$p(2, \mathcal{A}, s^{-1}rsr) = \emptyset$	$2 \to 1 \to 3 \to ? \to 2$	$2 \to ? \to 3 \to 1 \to 2$	None
$p(2, \mathcal{A}, s^2) = 2$	$2 \to 1 \to 2$	$2 \to 1 \to 2$	None
$p(3, \mathcal{A}, r^4) = 3$	$3 \to 6 \to 4 \to 1 \to 3$	$3 \to 1 \to 4 \to 6 \to 3$	None
$p(3, \mathcal{A}, s^{-1}rsr) = \emptyset$	$3 \to ? \to 2 \to 1 \to 3$	$3 \to 1 \to 2 \to ? \to 3$	None
$p(3, \mathcal{A}, s^2) = \emptyset$	$3 \to ? \to 3$	$3 \to ? \to 3$	None
$p(4, \mathcal{A}, r^4) = 4$	$4 \to 1 \to 3 \to 6 \to 4$	$4 \to 6 \to 3 \to 1 \to 4$	None
$p(4, \mathcal{A}, s^{-1}rsr) = \emptyset$	$4 \to 5 \to ? \to 6 \to 4$	$4 \to 6 \to ? \to 5 \to 4$	None
$p(4, \mathcal{A}, s^2) = 4$	$4 \to ? \to 4$	$4 \to (5) \to 4$	$4^s = 5$, $5^{s^{-1}} = 4$
$p(5, \mathcal{A}, r^4) = \emptyset$	$5 \to ? \to ? \to 2 \to 5$	$5 \to 2 \to ? \to ? \to 5$	None
$p(5, \mathcal{A}, s^{-1}rsr) = 5$	$5 \to 4 \to 1 \to 2 \to 5$	$5 \to 2 \to 1 \to 4 \to 5$	None
$p(5, \mathcal{A}, s^2) = 5$	$5 \to ? \to 5$	$5 \to (4) \to 5$	$5^s = 4$, $4^{s^{-1}} = 5$
$p(6, \mathcal{A}, r^4) = 6$	$6 \to 4 \to 1 \to 3 \to 6$	$6 \to 3 \to 1 \to 4 \to 6$	None
$p(6, \mathcal{A}, s^{-1}rsr) = \emptyset$	$6 \to ? \to ? \to 3 \to 6$	$6 \to 3 \to ? \to ? \to 6$	None
$p(6, \mathcal{A}, s^2) = \emptyset$	$6 \to ? \to 6$	$6 \to ? \to 6$	None

At this point we have verified that $2^{r^2} \neq \sigma$ for any $\sigma \in \Sigma$, and we have found two new edges $4^s = 5$, $5^s = 4$ along with their inverses. We will now give a formal algorithm which describes this test.

Algorithm 32 ***ExistentEdges***

Input*: Partial Automaton $\mathcal{A}$, an edge α^x that is undefined.*

Output: *An updated partial automaton $\mathcal{A}$ if new edges get defined and α^x if it gets defined, otherwise $\alpha^x = \emptyset$ if it does not get defined.*

1 for $r \in R$

2 for $\sigma \in \Sigma$

3 If a new edge is computed by checking if $p(\mathcal{A}, \sigma, r) = \sigma$ update $\mathcal{A}$ with the new edge. If α^x is defined as γ, then return $\alpha^x = \gamma$

4 If α^x is not defined return $\alpha^x = \emptyset$

The only part of this algorithm which might not be clear is step 3, in this step we might define new edges. We must check that these new edge definitions are correct, and therefore the new edges defined match the edges in $\mathcal{C}$.

Theorem 33 *The edges defined in step 3 of ExistentEdges, will correspond to the edges in the full coset automaton $\mathcal{C}$, determined by the map $\mathcal{A} \subseteq_{1\to\sigma} \mathcal{C}$.*

Proof: Suppose in step 3 we add the edge $\sigma^x = \delta$, where δ is defined such that $1^D = \delta$ and $1^S = \sigma$. To add the edge $\sigma^x = \delta$ in step 3 there must have been a state $\gamma = 1^G$, and a relator AxT^{-1} such that $\gamma^{AxT^{-1}} = \gamma$, with $\gamma^A = \sigma$ and $\gamma^T = \delta$. In the full coset automaton $\mathcal{C}$, since AxT^{-1} is a relator we know $(AxT^{-1})^{G^{-1}}$ is an element of N, therefore in $\mathcal{C}$ we must have $1^{GAxT^{-1}G^{-1}} = 1$ or $1^{GAx} = 1^{GT}$. This implies $\gamma^{Ax} = \sigma^x = \gamma^T = \delta$ so $\sigma^x = \delta$ in $\mathcal{C}$. Therefore the edges defined in step 3 of ExistentEdges will correspond to the edges in the full coset automaton $\mathcal{C}$ determined by the map $\mathcal{A} \subseteq_{1\to\sigma} \mathcal{C}$. □

The next thing we must prove is that if in the full coset automaton $\mathcal{C}$ we have $\alpha^x = \beta$, where α and β are states already in $\mathcal{A}$, then ExistentEdge will define $\alpha^x = \beta$.

Theorem 34 *If $\alpha^x = \beta$ in $\mathcal{C}$, where α, β are states in $\mathcal{A}$, then ExistentEdge will return $\alpha^x = \beta$.*

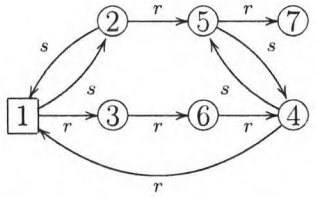
Proof: Suppose $\alpha = 1^A$ and $\beta = 1^B$ are states initially in $\mathcal{A}$ such that the edge $\alpha^x = \beta$ is not yet in $\mathcal{A}$, but $\alpha^x = \beta$ is an edge in $\mathcal{C}$. Then $1^{AxB^{-1}} = 1$ in $\mathcal{C}$, so $AxB^{-1} = \epsilon$. Since AxB^{-1} is not a relator because it is not a path in $\mathcal{A}$ this implies

that AxB^{-1} can be written as a product of conjugates of relators since it is an element of N. Therefore if $< R >=< r_1, \ldots r_n >$, we have $AxB^{-1} = r_{i_1}^{g_{i_1}} \cdots r_{i_k}^{g_{i_k}}$. If we define $\gamma_j = 1^{g_{i_j}^{-1}}$ then the walk $1^{AxB^{-1}} = 1^{r_{i_1}^{g_{i_1}} \cdots r_{i_k}^{g_{i_k}}} = \gamma_1^{r_{i_1}} \gamma_2^{r_{i_2}} \cdots \gamma_k^{r_{i_k}}$. So the path which has a label the word AxB^{-1} can be traced by tracing through the relators at each state γ_i.

Now we must show that there exists a state $\gamma_j \in \mathcal{A}$ such that by tracing through the relators r_{i_j} we define $\alpha^x = \beta$ with ExistentEdge. Since $AxB^{-1} = r_{i_1}^{g_{i_1}} \cdots r_{i_k}^{g_{i_k}}$ we know $A = r_{i_1}^{g_{i_1}} \cdots r_{i_{m-1}}^{g_{i_{m-1}}} g_{i_m}^{-1} P$ where $g_{i_m}^{-1} P$ is a prefix of $r_{i_m}^{g_{i_m}}$ and let $PxQ = r_{i_m}$. Then the claim is that ExistentEdge will define the edge $(\gamma_m^P)^x = \gamma_m^{Q^{-1}}$, and therefore α^x will then be defined.

Since by hypothesis $\alpha^x = \beta$ is an edge between two existent states we know γ_m^P and $\gamma_m^{Q^{-1}}$ are two states in $\mathcal{A}$. Therefore, since each of these states are known in $\mathcal{A}$ we can label the path from γ_m to γ_m^P by the letters of P, and similarly for Q^{-1}. Therefore in ExistentEdge we will see the path between γ_m^P and $\gamma_m^{Q^{-1}}$ close with $(\gamma_m^P)^x = \gamma_m^{Q^{-1}}$. At this point the edge α^x will be defined to be β another state in $\mathcal{A}$. □

At this point if we do not find an existent edge we know that $\alpha^x = \gamma$, where γ is a new state. In our example, since 2^{r^2} was not defined, we will define $2^{r^2} = 5^r = 7$, where 7 is a new state. At this point our partial automaton will look like the following figure.



The following algorithm will define a new state after we have determined that a missing edge does not go to a state that already exists.

Algorithm 35 ***NewState***

Input: *The partial automaton $\mathcal{A}$, a state α, and a word S, where S is defined to be the shortest subword of a relator such that $p(\mathcal{A}, \alpha, S) = \emptyset$, and we have already tested that $\alpha^S \neq \sigma$ for all $\sigma \in \Sigma$.*

Output: *An updated partial automaton $\mathcal{B}$ where $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$ and the states of $\mathcal{B}$ match the states of $\mathcal{A}$ except for the new state $p(\mathcal{A}, \alpha, S) =$ a new state.*

1 Let $S = s_1 \cdots s_k$

2 $\beta = p(\mathcal{A}, \alpha, s_1 \cdots s_{k-1})$

3 $p(\mathcal{B}, \alpha, S) = \gamma$ *where* $\gamma \notin \Sigma$ *(A new state)*

4 $\gamma^{s_k^{-1}} = \beta$ *in* $\mathcal{B}$

Next we will show NewState works correctly, by showing that NewState never defines a new state which is equivalent to a state already in $\mathcal{A}$.

Theorem 36 *NewState will never define a new state which is equivalent to a state which already exists in $\mathcal{A}$.*

Proof: In NewState we define $\beta^{s_k} = \gamma$ *and* $\gamma^{s_k^{-1}} = \beta$*, only after we test with ExistentEdge that* β^{s_k} *is not defined to be a state in* Σ*. Therefore* $\beta^{s_k} \neq \sigma$ *for any* σ *already in* Σ*, since in Theorem 34 we proved that all edges between existent states will be defined by ExistentEdge.* □

The construction makes it clear that the partial automaton $\mathcal{A}$ is always a subset of the full coset automaton $\mathcal{C}$, however we formalize this in the following theorem.

Theorem 37 *The partial automaton* $\mathcal{A}$ *is always a subset of the full coset automaton* $\mathcal{C}$*, assuming the original cosets in* $\mathcal{A}$ *are unique.*

Proof: Initially $\mathcal{A}$ *is the partial automaton constructed with the property that for all* $r \in R$ $1^r = 1$*. Since this property holds in the full coset automaton* $\mathcal{C}$ *and all of our initial states are unique, we have initially* $\mathcal{A} \subseteq \mathcal{C}$*.*

Next we call the algorithm ExistentEdges which will only add new edges to $\mathcal{A}$*. By Theorem 33 we showed that the new edges defined will always match the edges in* $\mathcal{C}$*. Therefore we will still have* $\mathcal{A} \subseteq \mathcal{C}$*.*

Then we consider the algorithm NewState where we add a new state and two new edges. First we must show our new state must be in $\mathcal{C}$*. Since* $\mathcal{C}$ *is a full coset automaton we know that there exists a state* β*, such that for each existing state* α *and for each* $x \in S \cup S^{-1}$ *the image of* $\alpha^x = \beta$ *will be defined. Now since we proved in Theorem 34, that* $\beta \neq \sigma$ *for all* $\sigma \in \Sigma$ *currently in* $\mathcal{A}$*, we know a new state* $\beta \in \mathcal{C}$ *is a unique state in* $\mathcal{C}$ *different from those in* $\mathcal{A}$*. Also since* $\alpha^x = \beta$ *in*

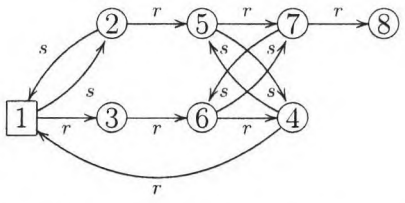
$\mathcal{C}$ we know that the edges $\alpha^x = \beta$ and $\beta^{x^{-1}} = \alpha$ added by NewState must also be in $\mathcal{C}$. Therefore $\mathcal{A} \subseteq \mathcal{C}$ and hence the partial automaton $\mathcal{A}$ is always a subset of the full coset automaton $\mathcal{C}$.$\square$

After calling NewState if $\mathcal{B}$ is not full we find the first state whose coset representative is the smallest in the well-ordering for Q, and this coset is not relator closed in $\mathcal{A}$. We then set $\mathcal{A} = \mathcal{B}$ and we will repeat the process again by calling ExistentEdges followed by NewState. This process will continue until $\mathcal{A}$ is full. Let us go back to our example. Again we see that the state 2 is not full since 2^{r^4} is not yet defined, therefore we call ExistentEdges. The following chart shows the result of this call.

$p(\mathcal{A}, \alpha, r)$	**Path Forward**	**Path Backward**	**New Edges**
$p(1, \mathcal{A}, r^4) = 1$	$1 \to 3 \to 6 \to 4 \to 1$	$1 \to 4 \to 6 \to 3 \to 1$	None
$p(1, \mathcal{A}, s^{-1}rsr) = 1$	$1 \to 2 \to 5 \to 4 \to 1$	$1 \to 4 \to 5 \to 2 \to 1$	None
$p(1, \mathcal{A}, s^2) = 1$	$1 \to 2 \to 1$	$1 \to 2 \to 1$	None
$p(2, \mathcal{A}, r^4) = \emptyset$	$2 \to 5 \to 7 \to ? \to 2$	$2 \to ? \to 7 \to 5 \to 2$	None
$p(2, \mathcal{A}, s^{-1}rsr) = \emptyset$	$2 \to 1 \to 3 \to ? \to 2$	$2 \to ? \to 3 \to 1 \to 2$	None
$p(2, \mathcal{A}, s^2) = 2$	$2 \to 1 \to 2$	$2 \to 1 \to 2$	None
$p(3, \mathcal{A}, r^4) = 3$	$3 \to 6 \to 4 \to 1 \to 3$	$3 \to 1 \to 4 \to 6 \to 3$	None
$p(3, \mathcal{A}, s^{-1}rsr) = \emptyset$	$3 \to ? \to 2 \to 1 \to 3$	$3 \to 1 \to 2 \to ? \to 3$	None
$p(3, \mathcal{A}, s^2) = \emptyset$	$3 \to ? \to 3$	$3 \to ? \to 3$	None
$p(4, \mathcal{A}, r^4) = 4$	$4 \to 1 \to 3 \to 6 \to 4$	$4 \to 6 \to 3 \to 1 \to 4$	None
$p(4, \mathcal{A}, s^{-1}rsr) = 4$	$4 \to 5 \to 7 \to 6 \to 4$	$4 \to 6 \to (7) \to 5 \to 4$	$7^s = 6$, $6^{s^{-1}} = 7$
$p(4, \mathcal{A}, s^2) = 4$	$4 \to 5 \to 4$	$4 \to 5 \to 4$	None
$p(5, \mathcal{A}, r^4) = \emptyset$	$5 \to 7 \to ? \to 2 \to 5$	$5 \to 2 \to ? \to 7 \to 5$	None
$p(5, \mathcal{A}, s^{-1}rsr) = 5$	$5 \to 4 \to 1 \to 2 \to 5$	$5 \to 2 \to 1 \to 4 \to 5$	None
$p(5, \mathcal{A}, s^2) = 5$	$5 \to 4 \to 5$	$5 \to 4 \to 5$	None
$p(6, \mathcal{A}, r^4) = 6$	$6 \to 4 \to 1 \to 3 \to 6$	$6 \to 3 \to 1 \to 4 \to 6$	None
$p(6, \mathcal{A}, s^{-1}rsr) = \emptyset$	$6 \to 7 \to ? \to 3 \to 6$	$6 \to 3 \to ? \to 7 \to 6$	None
$p(6, \mathcal{A}, s^2) = \emptyset$	$6 \to ? \to 6$	$6 \to (7) \to 6$	$6^s = 7$, $7^{s^{-1}} = 6$

At this point we have added two new edges and their inverses $7^s = 6$ and $6^s = 7$. However 2^{r^3} is not defined to be an existent state, so we call NewState

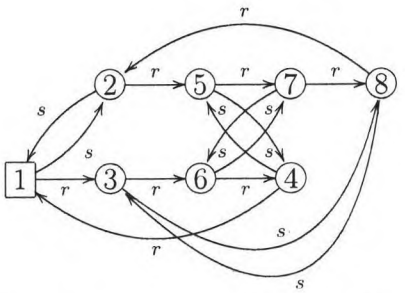
which will define $2^{r^3} = 5^{r^2} = 7^r = 8$. The following diagram shows the current state of $\mathcal{B}$.



Still $\mathcal{B}$ is not full since 2^{r^4} is still not defined, so we set $\mathcal{A} = \mathcal{B}$ and call ExistentEdges. This calculation is given by the following table.

$p(\mathcal{A}, \alpha, r)$	**Path Forward**	**Path Backward**	**New Edges**
$p(1, \mathcal{A}, r^4) = 1$	$1 \to 3 \to 6 \to 4 \to 1$	$1 \to 4 \to 6 \to 3 \to 1$	None
$p(1, \mathcal{A}, s^{-1}rsr) = 1$	$1 \to 2 \to 5 \to 4 \to 1$	$1 \to 4 \to 5 \to 2 \to 1$	None
$p(1, \mathcal{A}, s^2) = 1$	$1 \to 2 \to 1$	$1 \to 2 \to 1$	None
$p(2, \mathcal{A}, r^4) = 2$	$2 \to 5 \to 7 \to 8 \to 2$	$2 \to 8 \to 7 \to 5 \to 2$	$8^r = 2$, $2^{r^{-1}} = 8$
$p(2, \mathcal{A}, s^{-1}rsr) = 2$	$2 \to 1 \to 3 \to ? \to 2$	$2 \to (8) \to 3 \to 1 \to 2$	$3^s = 8$, $8^{s^{-1}} = 3$
$p(2, \mathcal{A}, s^2) = 2$	$2 \to 1 \to 2$	$2 \to 1 \to 2$	None
$p(3, \mathcal{A}, r^4) = 3$	$3 \to 6 \to 4 \to 1 \to 3$	$3 \to 1 \to 4 \to 6 \to 3$	None
$p(3, \mathcal{A}, s^{-1}rsr) = 3$	$3 \to 8 \to 2 \to 1 \to 3$	$3 \to 1 \to 2 \to (8) \to 3$	$8^s = 3$, $3^{s^{-1}} = 8$
$p(3, \mathcal{A}, s^2) = 3$	$3 \to 8 \to 3$	$3 \to 8 \to 3$	None
$p(4, \mathcal{A}, r^4) = 4$	$4 \to 1 \to 3 \to 6 \to 4$	$4 \to 6 \to 3 \to 1 \to 4$	None
$p(4, \mathcal{A}, s^{-1}rsr) = 4$	$4 \to 5 \to 7 \to 6 \to 4$	$4 \to 6 \to 7 \to 5 \to 4$	None
$p(4, \mathcal{A}, s^2) = 4$	$4 \to 5 \to 4$	$4 \to 5 \to 4$	None
$p(5, \mathcal{A}, r^4) = 5$	$5 \to 7 \to 8 \to 2 \to 5$	$5 \to 2 \to 8 \to 7 \to 5$	None
$p(5, \mathcal{A}, s^{-1}rsr) = 5$	$5 \to 4 \to 1 \to 2 \to 5$	$5 \to 2 \to 1 \to 4 \to 5$	None
$p(5, \mathcal{A}, s^2) = 5$	$5 \to 4 \to 5$	$5 \to 4 \to 5$	None
$p(6, \mathcal{A}, r^4) = 6$	$6 \to 4 \to 1 \to 3 \to 6$	$6 \to 3 \to 1 \to 4 \to 6$	None
$p(6, \mathcal{A}, s^{-1}rsr) = 6$	$6 \to 7 \to 8 \to 3 \to 6$	$6 \to 3 \to 8 \to 7 \to 6$	None
$p(6, \mathcal{A}, s^2) = 6$	$6 \to 7 \to 6$	$6 \to 7 \to 6$	None

At this point the partial automaton $\mathcal{B}$ is full since each coset is relator closed and the full coset automaton is given by the following figure.



Our next step will be to give an algorithm which computes a full coset automaton from a partial automaton $\mathcal{A}$.

Algorithm 38 ***FullAutomaton***

Input: *A partial automaton* $\mathcal{A}$

Output: *A full coset automaton constructed from the partial automaton* $\mathcal{A}$.

1 for $\alpha \in \Sigma$ *such that the coset representative for* α *is the smallest in the well-ordering for* Q.

2 if there exists a relator or inverse of a relator r *such that* $p(\mathcal{A}, \alpha, r) = \emptyset$ *then*

3 $ExistentEdges(\mathcal{A}, \alpha)$

4 if $p(\mathcal{A}, \alpha, r) = \emptyset$ *then there exists a subword* S *of* r *of shortest length such that* $p(\mathcal{A}, \alpha, S) = \emptyset$

5 $\mathcal{B}$=$NewState(\mathcal{A}, \alpha, S)$

6 $\mathcal{A} = \mathcal{B}$

6 return $\mathcal{A}$

Now we will prove if $Q \cong F/N$ is a finite group, then FullAutomaton will terminate.

Theorem 39 *Given Q is a finite group, the algorithm FullAutomaton will terminate.*

Proof: We need to show at some point α^x is defined for all $\alpha \in \Sigma$, and for all $x \in S \cup S^{-1}$, at which point FullAutomaton will return an updated $\mathcal{A}$, the full coset automaton. Suppose there exists an $\alpha \in \Sigma$, such that α^x is undefined. Then we have two cases, first $\alpha^x = \beta$ where β is already in the partial automaton $\mathcal{A}$. Then by Theorem 34: $\alpha^x = \beta$ will be defined by ExistentEdges. In the other case $\alpha^x = \beta$, where β is not already in $\mathcal{A}$, then since Q is finite, and by Theorem 36 we always define unique cosets, at some point we must define β. We will therefore define α^x, using either ExistentEdge or NewState. Therefore if Q is a finite group, FullAutomaton will terminate with a full coset automaton, since FullAutomaton constructs a coset automaton which is relator closed.

Chapter 6

PRODUCT OF CONJUGATES

Now that we have seen how the partial automaton can be used to construct the full coset automaton $\mathcal{C}$ for $Q \cong F/N$ where $N =< R >^F$, our next step will be to see how the partial automaton can be used for rewriting in group extensions. In Chapter 3, we described an algorithm which assumed we were able to obtain a normal form for elements in Q and K by some other means. For example, we could assume we also have a permutation representation for Q and K, and we define the normal form word to be the smallest length-lexicographic representative. We have described previously how this representative can be found using an algorithm which requires only the storage of two bits for each element in Q. The algorithm for rewriting in group extensions also required we find a way to take as input a word in N and express that word as a product of conjugates of relators. This step was required so we could compute the value of the 2-cocycles from the 2-cocycles we are storing for each relator of Q. Therefore in this chapter we will discuss how the partial automaton can be used to write words in N as a product of generators of N. We will begin by making two assumptions about the presentation for Q. The first assumption is, that our relators have the property that no relator is a proper subword of another relator. Our second assumption is that, no generator occurs only once in one relator. If this was the case we would apply the Tietze

Transformation which removes both that generator from the free group F and the relator in which that generator occurred from the set of relators. Since this is a Tietze Transformation this manipulation will not change the isomorphism class of the group defined by our presentation.

We will begin by describing a partial automaton $\mathcal{P}$ that is a graph representing all cyclic permutations of the relators given in the presentation for Q.

Definition 40 *We will call a partial automaton a* ***cyclically closed automaton*** *abbreviated* ***cc automaton*** *and denoted by $\mathcal{P}$ if it is a partial automaton and its relators are defined to be all cyclic permutations of the set of relators given for Q. Then the set of walks which can be traced through $\mathcal{P}$ will be defined by this set of relators.*

From this definition, we can now assume when we are talking about the relators for the cc automaton $\mathcal{P}$, that these relators consist of the relators given for Q along with all cyclic permutations of these relators. Now we will use the cc automaton $\mathcal{P}$ to compute a free generating set for N.

Definition 41 *A* ***free generating set*** *or* ***free basis*** *S for the normal subgroup N consists of a set of elements in N that generate N, with the property that every element of N can be written uniquely as a product of elements in the free generating set S.*

From (LS70, Prop 2.2) it follows that given a free group with an arbitrary generating set T, one can remove generators from this set to get a free generating set S. Therefore, since N is the group generated by all conjugates of relators, we will

then construct a free generating set S which consists of some conjugates of relators. Once this set is constructed we will use it to rewrite every element in N uniquely as a product of elements in S. In other words, we can use this set which consists of conjugates of relators, to uniquely rewrite every element of N as a product of conjugates of relators. Once this rewriting has been done we can use our previous description to do arithmetic in a group extension. To begin constructing S we must first construct a spanning tree for $\mathcal{C}$ the full coset automaton. Recall from our previous description that we are assuming we have a normal form for Q given. We can use the normal form for each word in Q to define a spanning tree for the full coset automaton $\mathcal{C}$.

Definition 42 *A spanning subtree $\mathcal{T}$ of $\mathcal{C}$ is* ***geodesic*** *in $\mathcal{C}$ if for any vertex v in $\mathcal{T}$ the label of the path from the identity to v is the normal form for the element in Q.*

With the geodesic tree we can now construct a free basis S for N. We begin by defining two types of edges for the graph of $\mathcal{C}$.

Definition 43 *Suppose the digraph $\mathcal{C}$ has the edge $a \rightarrow^g b$ then this edge is a* ***positive edge*** *for a since the arrow has initial vertex a, and a* ***negative edge*** *for b since b is the terminal vertex of the arrow.*

It is all about perspective, a positive edge from one vertex is a negative edge from another. We will call $vert(\mathcal{T})$ the set of vertices in the geodesic tree $\mathcal{T}$, and we will call $edge(\mathcal{T})$ the set of edges in the geodesic tree $\mathcal{T}$. Consider the set of edges $X = \{p_e | p_e \in edge(\mathcal{C} - \mathcal{T}), e \in vert(\mathcal{T})\}$, where the vertex e is the initial vertex in $\mathcal{T}$ such that each p_e is the label of a positive edge which leaves e, and the edge p_e is in $\mathcal{C} - \mathcal{T}$. The following lemma says that the cardinality of the set X is equal to the rank of N.

Lemma 44 *Let $\mathcal{C}$ be the full coset automaton and let $\mathcal{T}$ be a geodesic spanning tree in $\mathcal{C}$, then the number of positive edges in $\mathcal{C}$ outside $\mathcal{T}$, given by $|X|$, is equal to the rank(N).*

Proof: By Nielsen-Schreier (LS70, Prop 3.9) we have

$$rank(N) = [F : N](rank(F) - 1) + 1$$

$$= [F : N](rank(F)) - ([F : N] - 1)$$

In $\mathcal{C}$ we have $[F : N]$ vertices, each with rank(F) positive edges. Therefore there are [F:N]rank(F) positive edges in $\mathcal{C}$. Our tree $\mathcal{T}$ has one vertex for every group element in $Q \cong F/N$, which gives $[F : N] - 1$ edges. Therefore, the size of the set X is equal to the rank(N).

With this lemma we now have enough information to construct a free basis for N which will consist of conjugates of the relators given for Q. In practice we will not need to compute the entire set defined by the following algorithm, but rather, we will only construct a subset of this set necessary to do rewriting of a given word. In this construction we will associate to each positive edge $p_e \in X$ a unique element of N. It will be easier in this algorithm for conjugates of relators to have the form crc^{-1} which we will write as ${}^{c}r$ to denote that we are conjugating by c^{-1}. It should be noted that ${}^{c}r = r^{c^{-1}}$ so these are still elements of N.

The basic idea behind the algorithm is that we will compute a generator for each $p_b \in X$. We will have that p_b is a positive edge leaving a vertex b in the geodesic tree $\mathcal{T}$. We will define the label of a vertex to be equal to the word created by adjoining the labels of the path in $\mathcal{T}$ which begins at 1 and ends at b. For

example, suppose $b = b_1 \dots b_k$ then in the geodesic tree there is a path from the initial vertex 1 to the vertex b where the labels on the edges of this path are $b_1 \dots b_k$.

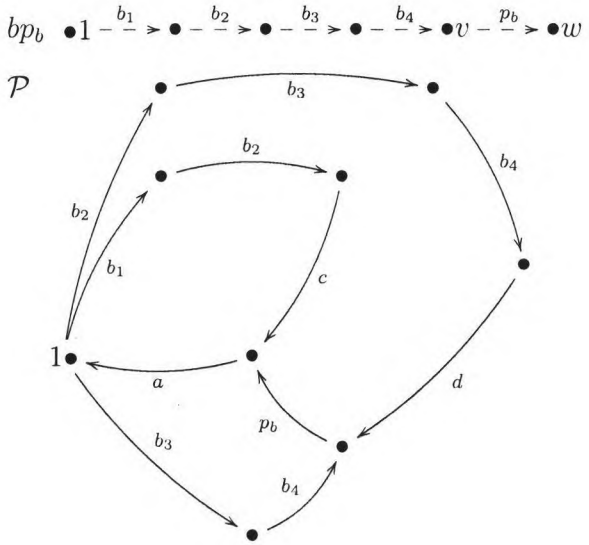
Now, since generators in S are conjugates of relators and we are only storing $\mathcal{P}$ which does not contain all conjugates of relators, we need to use the fact that a conjugate of a relator is a loop in $\mathcal{C}$ starting at a different vertex. With this in mind, and the fact that we need a generator for each positive edge $p_b \in X$, where b leaves the tree $\mathcal{T}$ from the vertex $b = b_1 b_2 \cdots b_k$, we will begin by tracing the word bp_b in $\mathcal{P}$. If we can trace completely through this word in $\mathcal{P}$ then we know the generator is a relator in $\mathcal{P}$. Otherwise, the generator must be a conjugate of a relator. We will now try to find a conjugate of a relator that has as prefix bp_b which we know exists since we know that there is a path in $\mathcal{P}$ that contains the edge p_b. In particular, we know that p_b is an edge leaving the initial vertex, since we assume the initial vertex of $\mathcal{P}$ is complete. We can then identify this path in $\mathcal{P}$ which contains possibly some suffix of b, along with p_b and then set some prefix of b to be the conjugating element. In practice if bp_b does not trace through $\mathcal{P}$ then our next step will be to let our conjugating element be b_1 and we try to trace $b_2 \cdots b_k p_b$ through $\mathcal{P}$ to see if we can define a relator. If not we will make our conjugating element be $b_1 b_2$ and repeat our process.

Once we find a path $b_i \cdots b_k p_b$ which is a path in $\mathcal{P}$ starting at the identity, we then can identify a relator by choosing any path back in $\mathcal{P}$ to the identity which is by definition a loop. By definition of a loop we will not allow ourselves to reverse any of the edges $b_i, \dots, b_k, p_b$ in this process. We know such a path exists, since $\mathcal{P}$ is a graph which consists of a union of loops for each relator and cyclic conjugates

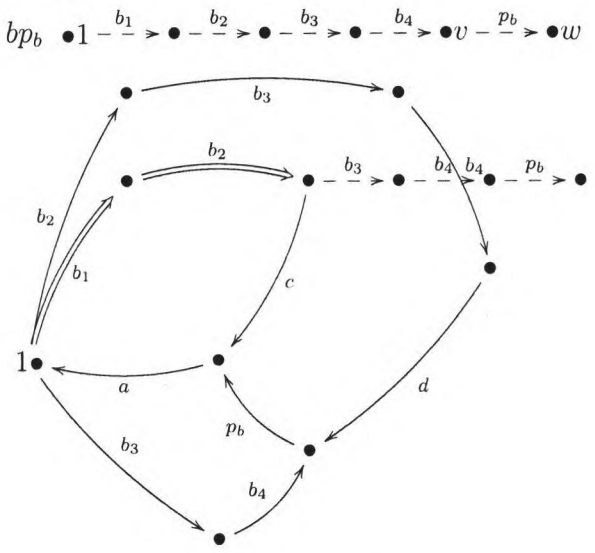
of the relators. There is a possibility of some choice involved on our path back and we will discuss the issues that arise with this choice in a moment. Since we assume the initial vertex is complete, we know at a minimum p_b is a path in $\mathcal{P}$ and b could be our conjugating element.

Before we go on to discuss the choice involved in the path back let us look at an example so we can visualize this description, in the following figure images of the path bp_b will be marked with $\dashrightarrow$ and the cc automation $\mathcal{P}$ whose lines will be labeled with $\longrightarrow$. We will then look at what happens when we place the cc automaton at each part of the word to find our conjugating element and the relator. Edges that appear in both graphs will be labeled with $\Longrightarrow$, and finally the path that finishes the relator will be labeled with edges that look like $\rightsquigarrow$. To make the display easier to read we will let $k = 4$.

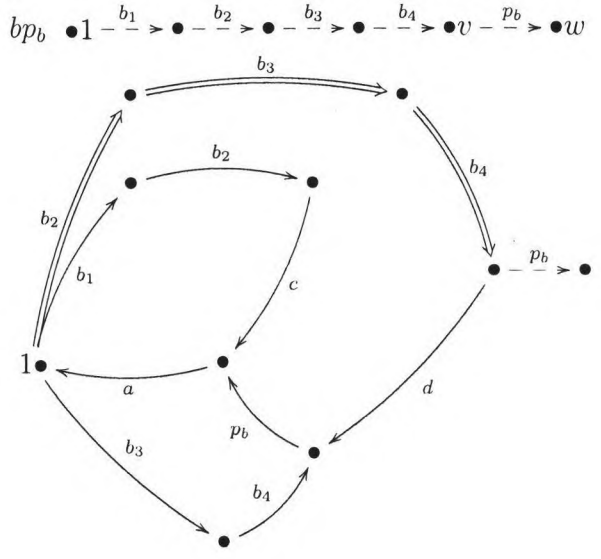
(1) Initially we assume we have the word $bp_b = b_1b_2b_3b_4p_b$ and the cc automaton $\mathcal{P}$ given in the figure below.



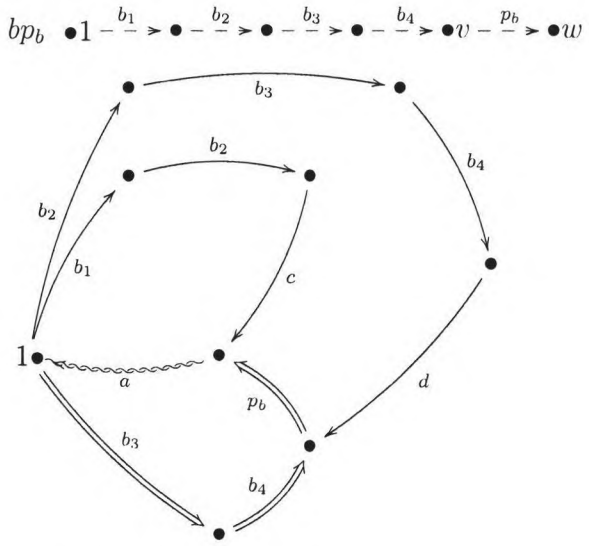
(2) We begin by tracing bp_b in the cc automaton $\mathcal{P}$. We see that the path b_1b_2 is in $\mathcal{P}$, however since $b_3 \neq c$ we cannot continue this path so we know we have a conjugate of a relator instead.



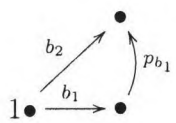
(3) At this point our conjugating element will be b_1 and we begin tracing the word $b_2b_3b_4p_b$ in $\mathcal{P}$, however again we find that we cannot trace through this entire path since $d \neq p_b$ so we will make our conjugating element longer.



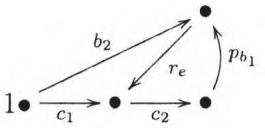
(4) Now our conjugating element is b_1b_2 and we see that the entire path $b_3b_4p_b$ traces through $\mathcal{P}$. We then choose a path back to the trivial coset which identifies the relator $b_3b_4p_ba$. The element of N which we will choose as a generator is ${}^{b_1b_2}(b_3b_4p_ba)$.



Our next step will be to discuss the issues that arise in our choice of path back to this initial vertex. The only issue we may have would be if we choose the same element as a generator for two different positive edges $p_b \in X$, then our set of generators would not generate all of N, since the generating set would not have the correct size. To see how our choice of path back might cause this redundancy, let us think about how two positive edges might be related. We begin by considering a positive edge p_{b_1} which leaves a vertex b_1. This edge p_{b_1} must map to another vertex b_2 in $\mathcal{T}$, since $\mathcal{T}$ is a spanning tree for $\mathcal{C}$. Therefore we have the following picture.

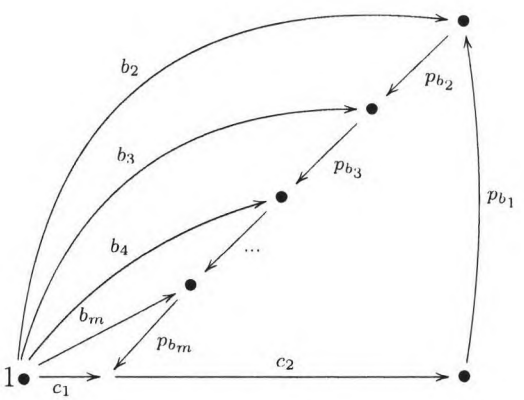


The next step in this process would be to compute the conjugating element c_1, such that $b_1 = c_1 c_2$ and $c_2 p_{b_1}$ is a path in $\mathcal{P}$. At this point we must choose a path back to the initial vertex in $\mathcal{P}$, we will call this path back r_e so that $c_2 p_{b_1} r_e$ is a loop in $\mathcal{P}$, and therefore represents a relator. We now have the following picture.



If two different edges p_e, p_b are given the same element of N as a generator then we say that we have a redundancy of generators. Now the path r_e is the only place where a choice is involved where we could cause a redundancy of generators can occur. To understand this potential redundancy let us consider two cases for the form of r_e. The first case is that $r_e = p_{b_2} p_{b_3} \cdots p_{b_m}$, or in other-words r_e is a path which consists of only positive edges.

In this case, we have $r_e = p_{b_2} p_{b_3} \cdots p_{b_m}$, a path consisting of all positive edges. We will then have the following picture, where we will assume that the edge p_{b_i} will leave the vertex b_i.



In this case, we claim ${}^{c_1}(c_2 p_{b_1} r_e) =^{b_i} ((c_2 p_{b_1} r_e)^{c_2 p_{b_1} p_{b_2} \cdots p_{b_{i-1}}})$. In other words, this relation shows that two conjugates of cyclic permutations of our original relator are equal, therefore we need to be sure not to define a cyclic permutation of this relator for one of the other positive edges p_{b_i}. To prove this we will use the fact that $b_i = b_1 p_{b_1} \cdots p_{b_i}$ in Q.

$$
\begin{aligned}
{}^{c_1}(c_2 p_{b_1} r_e) &= c_1 c_2 p_{b_1} r_e c_1^{-1} \\
&= c_1 c_2 p_{b_1} p_{b_2} \cdots p_{b_{i-1}} p_{b_i} p_{b_{i+1}} \cdots p_{b_m} c_1^{-1} \\
&= b_1 p_{b_1} p_{b_2} \cdots p_{b_{i-1}} p_{b_i} p_{b_{i+1}} \cdots p_{b_m} c_1^{-1} \\
&= b_i p_{b_i} p_{b_{i+1}} \cdots p_{b_m} c_1^{-1}
\end{aligned}
$$

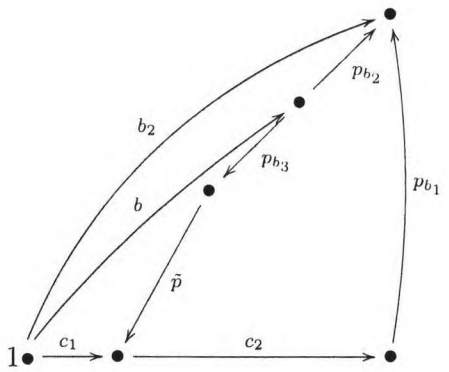
Now since $b_1 = c_1 c_2$ which implies that $c_1 = b_i (c_2 p_{b_1} \cdots p_{b_{i-1}})^{-1}$ which gives us

$$
\begin{aligned}
&= b_i p_{b_i} p_{b_{i+1}} \cdots p_{b_m} (b_i (c_2 p_{b_1} \cdots p_{b_{i-1}})^{-1})^{-1} \\
&= b_i p_{b_i} p_{b_{i+1}} \cdots p_{b_m} c_2 p_{b_1} \cdots p_{b_{i-1}} b_i^{-1} \\
&=^{b_i} (p_{b_i} p_{b_{i+1}} \cdots p_{b_m} c_2 p_{b_1} \cdots p_{b_{i-1}}) \\
&=^{b_i} ((c_2 p_{b_1} r_e)^{c_2 p_{b_1} p_{b_2} \cdots p_{b_{i-1}}})
\end{aligned}
$$

Therefore we must be careful not to choose the cyclic conjugate of the relator $c_2 p_{b_1} r_e$ for any of the other positive edges $p_{b_2}, \ldots, p_{b_m}$.

Now we will consider the second case that r_e contains at least one negative edge. For convenience we will assume that the negative edge occurs as the first letter, so $r_e = p_{b_2}^{-1} p_{b_3} \tilde{p}$. In this case p_{b_2}, p_{b_3} will both leave the same vertex $b \in \mathcal{T}$.

We will have the following picture.



At this point, since p_{b_2} is a positive edge leaving b, and not b_2 we will not have any issue with redundancy leaving b_2. It turns out that we get two issues from the positive edges leaving the vertex b. We get that ${}^{c_1}(c_2 p_{b_1} r_e) =^b ((c_2 p_{b_1} r_e)^{c_2 p_{b_1} p_{b_2}^{-1}})$, which implies that $({}^{c_1}(c_2 p_{b_1} r_e))^{-1} = ({}^b((c_2 p_{b_1} r_e)^{c_2 p_{b_1} p_{b_2}^{-1}}))^{-1}$ so we cannot define the inverse of this generator for the positive edge p_{b_2}. Again the proof of equality between these two equations stems from the fact that $b = b_1 p_{b_1} p_{b_2}^{-1}$ in Q.

$$
\begin{aligned}
{}^{c_1}(c_2 p_{b_1} r_e) &= c_1 c_2 p_{b_1} p_{b_2}^{-1} p_{b_3} \tilde{p} c_1^{-1} \\
&= b_1 p_{b_1} p_{b_2}^{-1} p_{b_3} \tilde{p} c_1^{-1} \\
&= b p_{b_3} \tilde{p} c_1^{-1}
\end{aligned}
$$

Since $b_1 = c_1 c_2$ this implies that $c_1 = b(c_2 p_{b_1} p_{b_2}^{-1})^{-1}$

$$
\begin{aligned}
& b p_{b_3} \tilde{p} c_2 p_{b_1} p_{b_2}^{-1} b^{-1} \\
=^b & \ (p_{b_3} \tilde{p} c_2 p_{b_1} p_{b_2}^{-1}) \\
=^b & \ ((c_2 p_{b_1} r_e)^{c_2 p_{b_1} p_{b_2}^{-1}})
\end{aligned}
$$

Therefore again we must be careful not to choose the cyclic conjugate of the relator $c_2 p_{b_1} r_e$ for any positive edge p_{b_i} in r_e. For any negative edge, no matter where it

occurs in the word r_e, we must not choose the inverse of the cyclic conjugate of the relator $c_2 p_{b_1} r_e$.

Definition 45 *We define the* ***redundancy condition*** *to be the set of redundancies of either type one where $r_e = p_{b_2} p_{b_3} \cdots p_{b_m}$ so that r_e is a path which consists of only positive edges or of type two where r_e contains at least one negative as edge. If we have a redundancy of generators because one of the two types of redundancies has occurred we will say that the redundancy condition has been violated.*

With this definition we now give a theorem which states that the only way we can have a redundancy of generators is if the redundancy condition is violated.

Theorem 46 *The generators constructed from $\mathcal{P}$ are unique if we make sure to not violate the redundancy condition.*

Proof: Suppose we have two positive edges p_b and p_e where p_e is not one of the edges on the path back to the identity chosen for p_b. Then we would not violate the redundancy condition, so we will then show that no redundancy will occur. The claim is that the two generators constructed from $\mathcal{P}$ for these two positive edges are distinct. Suppose we choose ${}^c(r)$ for p_b where the prefix of ${}^c(r)$ is bp_b and we choose ${}^d(q)$ for p_e where the prefix of ${}^d(q)$ is ep_e, then we claim ${}^c(r) \neq {}^d(q)$ which says that the two generators constructed are distinct.

Assume by contradiction that ${}^c(r) = {}^d(q)$ which implies

$$crc^{-1} = dqd^{-1}.$$

Then we can manipulate this equation to get

$$r = c^{-1}dqd^{-1}c = {}^{c^{-1}d}(q)$$

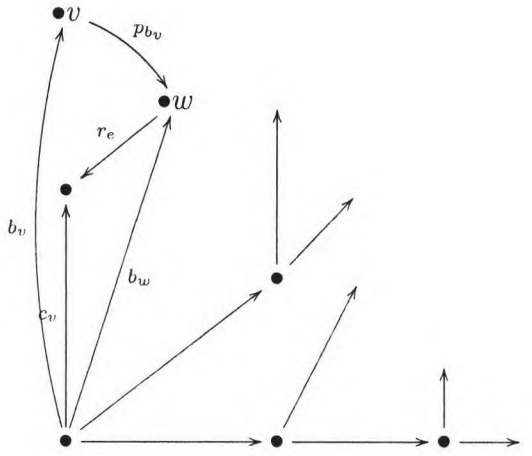
Therefore we can conclude that r is a conjugate of q. Since these are both loops in $\mathcal{P}$ this will allow us to conclude that r is a cyclic conjugate of q. We make this conclusion since we assumed originally that in $\mathcal{P}$ the only loops which are conjugates of relators are cyclic conjugates of the original relators for Q by requiring that no relator of Q is a proper subword of another. Therefore we have a contradiction to our assumption that ${}^{c}(r) =^{d} (q)$. Hence the only way the redundancy condition can be violated is if we get a redundancy of the form described by case one or case two.
□

As a consequence of this theorem if we now choose generators which avoid violating the redundancy condition we know that the free generators we construct will be unique. Now we will give an algorithm to compute a free generating set. We will see later how this algorithm can be adapted so we do not need to store all of the previously computed generators to check for a possible redundancy.

Algorithm 47 ***FreeGen***

Input: *A geodesic tree $\mathcal{T}$, a generating set for the free group F, and a cc automaton $\mathcal{P}$.*

Output: *A free generating set for N given as conjugates of relators.*



1 For each vertex v in $\mathcal{T}$ let b_v be the label of the path in $\mathcal{T}$ from 1 to v.

2 Compute all positive edges $p_{b_v} \in edge(\mathcal{C} - \mathcal{T})$ where $b_v \in edge(\mathcal{T})$.

3 For all p_{b_v} do

4 For each edge p_{b_v} it must have a terminal vertex w in $\mathcal{T}$. Let b_w be the label of the path in $\mathcal{T}$ from 1 to w. Then $b_v p_{b_v} b_w^{-1}$ is a loop in the full automaton $\mathcal{C}$, so it is a word in N.

5 Let c_v be the prefix of b_v such that $c_v^{-1} b_v p_{b_v}$ is the prefix of a relator r in $\mathcal{P}$. In other words, we can trace through $c_v^{-1} b_v p_{b_v}$ in $\mathcal{P}$, then by tracing this path back to the trivial coset in $\mathcal{P}$ we will identify a loop in $\mathcal{P}$ whose edges define the relator $r = c_v^{-1} b_v p_{b_v} r_e$.

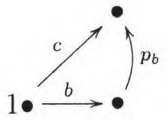
6 Add to the list of generators ${}^{c_v}r$. Also check that we are defining a unique element of N, by checking that the redundancy condition is not violated by this assignment. If the redundancy condition is violated then we want to choose a different relator r, or make c_v a longer prefix of $b_v p_{b_v}$.

7 return the list of free generators ${}^{c_v}r$.

We will call the list of generators returned from the algorithm FreeGen S. The next obvious question is whether or not this algorithm works. The only part which might still be unclear is that we will always have another choice of element in N that we can choose which we can associate to each positive edge.

Theorem 48 *It is possible to choose a generator for each* $p_b \in X$ *such that the redundancy condition is not violated.*

Proof: We begin by choosing an edge $p_b \in X$ *such that b is the label in* $\mathcal{T}$ *of the path from the trivial coset to the initial vertex of* p_b. *We know that this edge must map to another state in* $\mathcal{T}$ *which we will call c. Therefore we have the following picture to describe this situation.*



We want to show that regardless of previous choices we can still find a generator for p_b *that does not violate teh redundancy condition. We will prove the existence of a generator by contradiction. Therefore, we suppose instead that using the algorithm FreeGen we cannot find any path in* $\mathcal{P}$ *that does not violate the redundancy condition.*

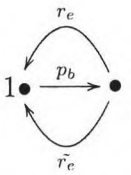
Our first step will be to show if this is the case, then our conjugating element for p_b *must be b. We will prove this by assuming no shorter prefix of b can be chosen as the conjugating element without the generator computed from FreeGen violating the redundancy condition. To prove this, suppose our conjugating element*

is b_1 a prefix of b such that $b = b_1 b_2$ and we cannot find any relator $b_2 p_b r_e$ such that the redundancy condition is not violated. Then in step 6 of FreeGen we make our prefix b_1 longer and search again. Therefore, the only way we can not make the prefix longer is if b is the conjugating element. Since we are assuming no generator can be found, this will be the case.

Now that we can assume that b is our conjugating element the next step will be to prove there exists at least two paths in $\mathcal{P}$ which begin at p_b and trace back through as a loop to the trivial coset 1. Since we know that the trivial coset 1 is complete, we know there exists at least one path in $\mathcal{P}$ which begins with p_b. Now we claim that in fact there exists at least two paths in $\mathcal{P}$ which begin with p_b. We will prove this by contradiction. Therefore, we will assume that there is only one path in $\mathcal{P}$ that begins with p_b and that path is $p_b r_e$. Now since $\mathcal{P}$ contains all cyclic permutations of relators this implies that p_b is a generator of F such that there exists only one relator for Q which contains p_b only once. We can guarantee this, since if p_b or p_b^{-1} had occurred in any other relator, then a cyclic conjugate of that relator or its inverse would begin with p_b and this would lead to another path in $\mathcal{P}$ which begins with p_b. However, we originally made an assumption that Tietze Transformations would be applied to the set of relators for Q which would have removed p_b from the set of generators for F in this case. Therefore, we can conclude that there exists at least two paths in $\mathcal{P}$ such that the path begins with p_b.

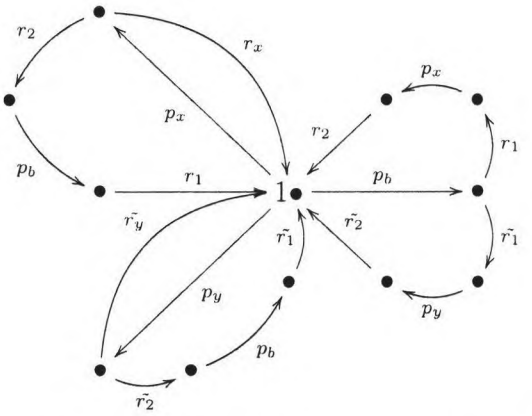
The final step of the proof is to show that there will always be a generator of N we can choose that will not violate the redundancy condition. In our previous step, we showed that there are at least two paths $p_b r_e$ and $p_b \tilde{r_e}$ in $\mathcal{P}$ which begin

with p_b. Now, we want to show that if a cyclic conjugate, or the inverse of a cyclic conjugate, of both of these paths have already been chosen for generators of other positive edges then we can guarantee the existence of other paths in $\mathcal{P}$ which begin with p_b that we can choose from. We begin by assuming that there are only 2 paths in $\mathcal{P}$ which begin with p_b, then we know a subset of $\mathcal{P}$ must look like the following figure.

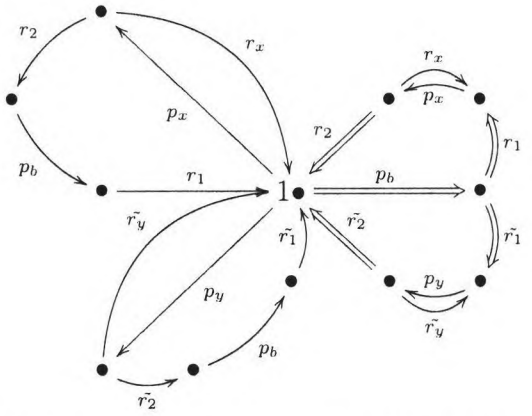


Since we assumed that some cyclic conjugate or the inverse of a cyclic conjugate, of both $p_b r_e$ and $p_b \tilde{r_e}$ have already been chosen we must look at a couple of cases depending on whether or not the inverse of a cyclic conjugate or a cyclic conjugate was chosen. First, we will suppose a cyclic conjugate was chosen for both positive edges so there exists two positive edge p_x and p_y such that $p_b r_e = p_b r_1 p_x r_2$ and $p_b \tilde{r_e} = p_b \tilde{r_1} p_y \tilde{r_2}$. In this case we know there exists edges x,y in $\mathcal{T}$ such that ${}^x(p_x r_2 p_b r_1)$ and ${}^y(p_y \tilde{r_2} p_b \tilde{r_1})$ were chosen as generators. We also know there exists at least two paths in $\mathcal{P}$ which begin with p_x and p_y, so we know there are other paths $p_x r_x$ and $p_y \tilde{r_y}$ which must be in $\mathcal{P}$, so the cc automaton contains the paths $p_x r_2 p_b r_1$, $p_x r_x$, $p_y \tilde{r_2} p_b \tilde{r_1}$, and $p_y \tilde{r_y}$. Therefore a subset of $\mathcal{P}$ must look like the following figure. It should be noted that we may have two edges in the following graph which are in fact the same edges in $\mathcal{P}$. This discrepancy will not change the

argument, so for clarity we will draw the diagram ignoring this possibility.



From this figure since $\mathcal{P}$ contains all cyclic conjugates of relators we see two possibly new loops which begin with p_b, and they are $p_b r_1 r_x^{-1} r_2$ and $p_b \tilde{r_1} \tilde{r_y}^{-1} \tilde{r_2}$. We will now update our subset of $\mathcal{P}$ by drawing in these loops as if they are different from the other loops, however it remains to show that these loops are in fact new.



To determine if these are new loops, suppose that in F

$$p_b r_1 r_x^{-1} r_2 = p_b r_1 p_x r_2$$

which means we are assuming these are exactly the same words. Then this would imply in F $r_x^{-1} = p_x$ which would mean $p_x r_x$ is not a loop in $\mathcal{P}$ which is a contradic-

tion. Similarly we can conclude that $p_b\tilde{r_1}\tilde{r_y}^{-1}\tilde{r_2} \neq p_b\tilde{r_1}p_y\tilde{r_2}$. *Next, if either of these loops are new then we have found other paths in* $\mathcal{P}$ *to choose from. Therefore, we suppose instead that they are not new which means*

$$p_b r_1 r_x^{-1} r_2 = p_b \tilde{r_1} p_y \tilde{r_2}$$

and

$$p_b \tilde{r_1} \tilde{r_y}^{-1} \tilde{r_2} = p_b r_1 p_x r_2$$

in F *so that we have not found any new paths in* $\mathcal{P}$, *just a renaming of the paths we already knew.*

Since $p_b r_1 r_x^{-1} r_2 = p_b \tilde{r_1} p_y \tilde{r_2}$ *and* $p_b \tilde{r_1} \tilde{r_y}^{-1} \tilde{r_2} = p_b r_1 p_x r_2$ *then we can suppose without loss of generality that* $\tilde{r_1} = r_1\overline{r_1}$ *and* $\tilde{r_2} = \overline{r_2}r_2$ *or that* $\tilde{r_1}$ *and* $\tilde{r_2}$ *are the longer words. If this was not the case and* r_1 *or* r_2 *was the longer word then we would have a similar equality for* r_1 *and* r_2 *so the argument will not change. From the identity* $p_b \tilde{r_1} \tilde{r_y}^{-1} \tilde{r_2} = p_b r_1 p_x r_2$ *we get*

$$\overline{r_1} \tilde{r_y}^{-1} \overline{r_2} = p_x$$

which implies

$$\overline{r_1} = p_x \overline{r_2}^{-1} \tilde{r_y}$$

From $p_b r_1 r_x^{-1} r_2 = p_b \tilde{r_1} p_y \tilde{r_2}$ *we get*

$$r_1 r_x^{-1} r_2 = r_1 \overline{r_1} p_y \overline{r_2} r_2$$

which implies

$$r_x^{-1} = \overline{r_1} p_y \overline{r_2}.$$

Plugging in the identity for $\overline{r_1}$ *we get*

$$r_x^{-1} = p_x \overline{r_2}^{-1} \tilde{r_y} p_y \overline{r_2}$$

which implies $r_x = (p_x \overline{r_2}^{-1} \tilde{r_y} p_y \overline{r_2})^{-1} = \overline{r_2}^{-1} p_y^{-1} \tilde{r_y}^{-1} \overline{r_2} p_x^{-1}$. *Therefore the loop in* $\mathcal{P}$ *of* $p_x r_x$ *which begins with* p_x *leaving the trivial coset 1, becomes*

$$p_x \overline{r_2}^{-1} p_y^{-1} \tilde{r_y}^{-1} \overline{r_2} p_x^{-1}$$

which is not a loop which begins at the trivial coset, but rather a loop which leaves the vertex p_x. *This is a contradiction, so one of these paths must be new.*

Now in the previous argument we assumed that both of the generators already chosen were cyclic conjugates of the two possible generators for p_b. *If we instead had an inverse of a cyclic conjugate of a generator chosen the argument would be exactly the same, except we would replace* p_x *by* p_x^{-1}. *We will still find that there exists at least one new path that begins with* p_b.

Finally we claim that one of these new paths we just found has been chosen for another positive edge p_z, *then there will be other paths we will be able to choose from. Suppose any of these new paths was already chosen for a different positive edge. We can assume for example that* $p_b r_1 r_x^{-1} r_2$ *was chosen for a positive edge* p_z *such that* $p_b r_1 r_x^{-1} r_2 = p_b \overline{r_1} p_z \overline{r_2}$ *then again we get another possibly new relator* $p_b \overline{r_1} r_z^{-1} \overline{r_2}$ *in* $\mathcal{P}$. *Again by a similar argument as before we can show that either this relator must be new or the other new relator we found previously must be new. Eventually since only* $rank(N) - 1$ *generators can be assigned before the generator for* p_b *we will have to find a new path which has not been used. Therefore it is*

always possible to choose a generator for each $p_b \in X$ *such that the redundancy condition is not violated.* □

Now that we have guaranteed the existence of a choice of generators our next step will be to prove that the set S returned by FreeGen is a basis for N. We will begin by showing the list returned has the correct size.

Theorem 49 *The set S will have cardinality rank(N).*

Proof: By Lemma 44 the number of positive edges in $\mathcal{C} - T = |X| = rank(N)$. *Therefore since we add a unique generator for each positive edge in* $\mathcal{C} - T$ *we are only adding* $rank(N)$ *generators, so the cardinality of S will be rank(N).* □

Our next step is to show S generates N. To show this we will show that if we build a graph from S which is essentially a union of Cayley graphs for each generator in S that this graph can be used to construct a graph which is isomorphic to $\mathcal{C}$ the full coset automaton. We will call Y the **alphabet for a graph** and Y will consist of the set of generators of F along with there inverses. The following description comes from (KM00).

Definition 50 *Let* Γ_1 *be a directed graph (digraph), where there exists at least one vertex v in* Γ_1 *and at least two edges in* Γ_1 *labeled by* $y \in Y$, *then we say* Γ_2 *results from* ***folding*** Γ_1 *if in* Γ_2 *there exists a vertex* $\tilde{v}$ *the image of v in* Γ_2 *and only one path labeled by* $y \in Y$, *where the terminal vertices of the edges labeled by y in* Γ_1 *are identified as the same vertex in* Γ_2. *The rest of* Γ_2 *remains isomorphic to* Γ_1. *Therefore a folding identifies a set of duplicate edges in* Γ_1.

The following lemma gives us a few more details about how a folded graph looks, the proof is trivial and will not be included.

Lemma 51 *Let Γ be a digraph obtained from folding a graph $\tilde{\Gamma}$. Let v be a vertex of Γ and $\tilde{v}$ be the corresponding vertex of $\tilde{\Gamma}$. Then the following hold.*

1 *If $\tilde{\Gamma}$ is connected then Γ is connected.*

2 *Let p be a path from $\tilde{v}$ to $\tilde{v}$ in $\tilde{\Gamma}$ with label w. Then the edgewise image of p in Γ is a path from v to v with label w.*

3 *If the digraph $\tilde{\Gamma}$ is finite, then a folding always decreases the number of edges in $\tilde{\Gamma}$ by one.*

We will say a graph Γ is a **folded graph** if it results from applying all possible foldings to a graph $\tilde{\Gamma}$. The next theorem states that there exists a unique folded graph resulting from applying all possible foldings to a given digraph.

Theorem 52 *Suppose $\tilde{\Gamma}$ is a digraph, then there exists a unique folded graph Γ which is the graph which results from applying all possible foldings to $\tilde{\Gamma}$*

Proof: Suppose that both Γ and Ω are distinct graphs that result from applying all possible foldings to $\tilde{\Gamma}$. Now since both Γ and Ω result from folding $\tilde{\Gamma}$, consider the vertex $\tilde{v} \in vert(\tilde{\Gamma})$, along with path p from $\tilde{v} \to \tilde{v}$ in $\tilde{\Gamma}$ with edge labels w. We know there exists a vertex $v \in \Gamma$, and $\overline{v} \in \Omega$ along with the corresponding images of the p which map $v \to v$ and $\overline{v} \to \overline{v}$ respectively in Γ and Ω, with edge labels w. Let $f : vert(\Gamma) \to vert(\Omega)$ be defined so that $f(v) = \overline{v}$ and for any other vertex $u \in vert(\Gamma)$ which lies on a path leaving v where the label of that path is x written $v^x = u$ have as image $f(u) = f(v^x) = f(v)^x = \overline{v}^x$. Since paths in Γ are defined as

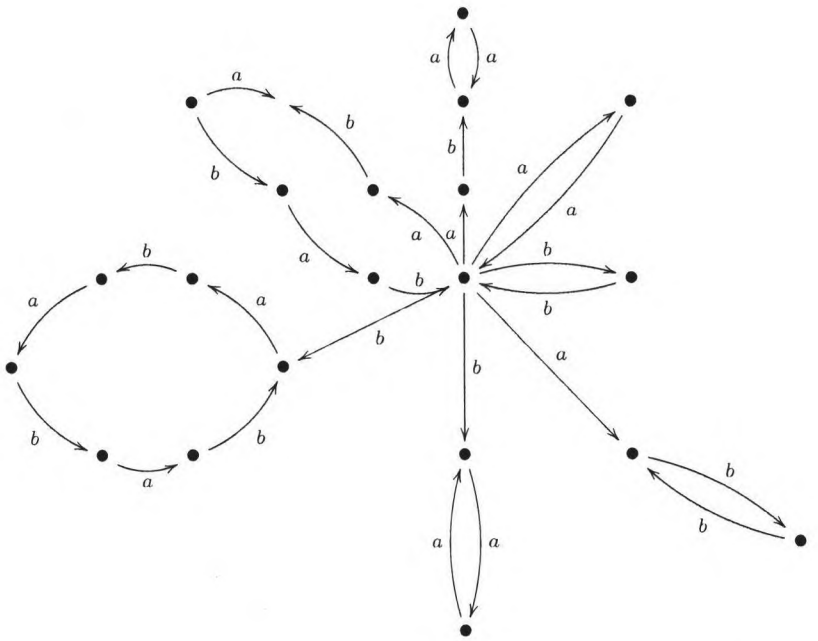
images of paths in $\tilde{\Gamma}$ we know this map is well defined. Also we see that f preserves paths, therefore if f is one-to-one we can conclude that $\Gamma \cong \Omega$ and therefore there exists a unique folded graph for $\tilde{\Gamma}$. Suppose $\overline{u} = \overline{v}$ in $\tilde{\Gamma}$, which means that the image of $\tilde{u}$ and $\tilde{v}$ in Ω results in the same vertex in Ω. Then suppose $u \neq v$ where u and v are the corresponding images of $\tilde{u}$ and $\tilde{v}$ in Γ. Since $\overline{u} = \overline{v}$ this implies that a folding was applied in Ω to make that equality, and since Γ is a folded graph this folding must have been applied there as well, so $u = v$. Which implies that f is one to one and $\Gamma \cong \Omega$. $\square$

Our next step will be to precisely define a graph which we will call the bouquet created by the generators of S.

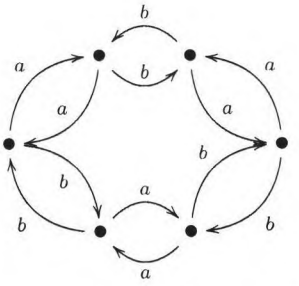
Definition 53 *We define the **bouquet** $\mathcal{B}$ for S as the digraph with the property that for every element S a Cayley graph is formed and the bouquet is then the union of these Cayley graphs where the initial vertex is identified.*

To better understand this definition we will consider an example, let $Q = S_3 \cong F/N$ then a set S which generates N is $S = \{a^2, b^2, {}^a b^2, {}^b a^2, {}^b (ababab), {}^{ab} a^2, ababab\}$.

The bouquet $\mathcal{B}$ looks like the following figure.



Next we will show that if we apply all possible foldings to this figure, so that $\mathcal{B}$ is folded, we will get the full coset automaton $\mathcal{C}$. Let us first see how this works in our example, below is the figure for $\mathcal{B}$ folded.



We can see that this diagram is isomorphic to the full coset automaton $\mathcal{C}$ for $S_3 =< a, b | a^2 = b^2 = (ab)^3 >$, which was given earlier. Now that we have seen in an example that $\mathcal{B}$ folded is isomorphic to $\mathcal{C}$, our next step will be to prove this

result in general. To prove that when we fold $\mathcal{B}$ we get a graph isomorphic to $\mathcal{C}$ we must first describe the properties a graph must have to be the full coset automaton $\mathcal{C}$ for F/N. The following description comes from (S94).

Definition 54 *A vertex v in a digraph Γ is **accessible** if there exists a path in Γ from the trivial vertex 1 to v. We define Γ_a to be the set of vertices in Γ which are accessible. We say Γ is **accessible** if $\Gamma = \Gamma_a$.*

It should be noted that from this definition a digraph Γ which is accessible is the same as a connected graph. We will define $\Gamma_{\mathcal{B}}$ to be the graph that results from folding the bouquet $\mathcal{B}$. Our next step will be to show that $\Gamma_{\mathcal{B}}$ is accessible.

Lemma 55 *The digraph $\Gamma_{\mathcal{B}}$ is accessible.*

Proof: First each vertex in Γ comes from a vertex in the bouquet $\mathcal{B}$. The bouquet $\mathcal{B}$ is accessible because each element of S is a loop from the trivial coset back to the trivial coset. Therefore $\Gamma_{\mathcal{B}}$ is also accessible.

The next property of a coset automaton $\mathcal{C}$ is that $\mathcal{C}$ is deterministic.

Definition 56 *A digraph Γ is **deterministic** if*

a It has only one initial vertex

b The label of each edge is nonempty.

c For every vertex v in Γ, and for each $y \in Y$ there exists at most one edge leaving v labeled x in Γ.

Now we will show $\Gamma_{\mathcal{B}}$ which is the folded bouquet $\mathcal{B}$ is deterministic.

Lemma 57 *The folded bouquet $\Gamma_{\mathcal{B}}$ resulting from $\mathcal{B}$ is deterministic.*

Proof: The only part of this definition that is not clear is item c that for every vertex v in $\Gamma_{\mathcal{B}}$ and for each y a negative element of Y, that there exists an edge leaving v labeled by y in $\Gamma_{\mathcal{B}}$. We know by the construction of $\mathcal{B}$ that for all positive y in Y, and all v in $\Gamma_{\mathcal{B}}$ we have the edge leaving v labeled y defined in $\Gamma_{\mathcal{B}}$. Now since we have the same number of positive edges as negative edges and each positive edge $v^y = w$ corresponds to a negative edge $w^{y^{-1}} = v$, we only need to show that we cannot have $w^{y^{-1}} = v$ and $w^{y^{-1}} = u$ where $v \neq u$. If this cannot happen, then since each inverse edge goes to a unique vertex we can conclude for all vertices v and all $y \in Y$ there is an edge leaving v labeled by y in $\Gamma_{\mathcal{B}}$. Suppose $w^{y^{-1}} = v, w^{y^{-1}} - u$ where $v \neq u$, then since $\Gamma_{\mathcal{B}}$ is folded we have both edges leaving w under y^{-1} are the same edge, which means the vertices $v = u$ which is a contradiction. Therefore each inverse edge goes to a unique vertex and we can conclude that $\Gamma_{\mathcal{B}}$ is deterministic.

Our next step is to show that any directed graph with the following properties is a coset automaton.

Lemma 58 *A **coset automaton** is a directed graph Γ such that the following properties hold.*

a Γ is accessible

b Γ is deterministic

c The initial vertex is the same as the terminal vertex

d If there is an edge $v^y = w$ in Γ, then $w^{y^{-1}} = v$ is another edge in Γ.

Proof: To show that these 4 properties define a coset automaton, we must show that given a digraph Γ *with properties a-d, then* Γ *is a coset automaton. First we give the definition of a coset automaton which we gave earlier. We defined a coset automaton as a directed graph that consists of a set of vertices* Σ *one for each coset fN, and edges E. An edge is defined from coset i to coset j if in the coset table there exists a generator y of F; where the image of i under y denoted* $i^y = j$. *We then label this edge with the generator y. We will identify the initial coset as the identity element. Therefore, we will show that a graph with these 4 properties defines a coset table, and that coset table will correspond to a coset automaton. We will number the vertices of the digraph, choosing one of them to be the trivial vertex, then we will label our coset table so that the rows are numbered by these vertices. We will label the columns of the table with the set of elements in Y for the graph. Then since the graph* Γ *is accessible we know that the there exists a path in* Γ *that connects the trivial vertex to each other vertex in* Γ. *Therefore we can record the definition of each vertex as an image in our coset table. Next, since* Γ *is deterministic we know that each row of the table will be full, and therefore we know that our graph* Γ *will define a coset table. Hence this coset table corresponds to a coset automaton. Therefore these four properties completely define a coset automaton.*

Our final step will therefore be to show that $\Gamma_{\mathcal{B}}$ the directed graph resulting from folding $\mathcal{B}$ is a coset automaton. If we show this, then since $\mathcal{C}$ is defined to be the coset automaton for F/N we can conclude $\Gamma_{\mathcal{B}} \cong \mathcal{C}$. If this property holds, then since the loops of $\mathcal{B}$ were defined to be the elements of S and we showed that this graph when folded was isomorphic to the full coset automaton $\mathcal{C}$ we can conclude

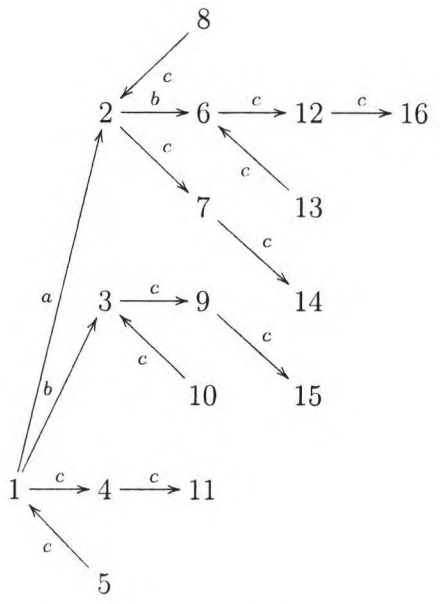
that S is a generating set for N. The reason we can make this conclusion is that we know by definition of the full Cayley graph every element of N is represented by a loop in $\mathcal{C}$. Since we can use S to construct $\mathcal{C}$, and all loops in a folded graph are loops of the original graph $\mathcal{B}$ or a product of loops in the original graph $\mathcal{B}$ when the edges are identified. We can therefore conclude that every element of N can be written as a product of elements in S, and therefore S is a generating set for N.

Theorem 59 *The digraph $\Gamma_{\mathcal{B}}$ is isomorphic to the coset automaton $\mathcal{C}$.*

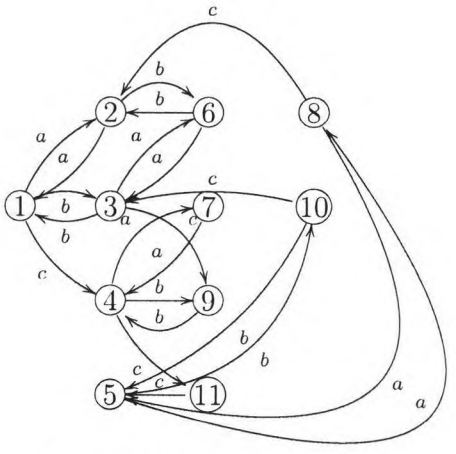
Proof: In Lemma 55 we showed $\Gamma_{\mathcal{B}}$ is accessible. In Lemma 57 we showed $\Gamma_{\mathcal{B}}$ is deterministic. By definition of $\mathcal{B}$ we have that $\Gamma_{\mathcal{B}}$ has the trivial coset as the initial and terminal vertex, since each loop begins and ends at that vertex. Therefore $\Gamma_{\mathcal{B}}$ has the same initial vertex as the terminal vertex. Finally we define inverse edges in $\Gamma_{\mathcal{B}}$ such that if $v^y = w$ then $w^{y^{-1}} = v$. Therefore $\Gamma_{\mathcal{B}}$ is a coset automaton for F/N which means $\Gamma_{\mathcal{B}} \cong \mathcal{C}$. $\square$

Our next step will be to show an example where we compute a free generating set for the group $Q =< a, b, c | a^2 = b^2 = abab = acac^{-1} = bcbc^{-1} = c^4 >$ which is a transitive group of order 16. We will begin by computing the geodesic tree and the partial automaton $\mathcal{P}$. Below are these two objects.

Geodesic Tree T



CC Automata $\mathcal{P}$



Now we will give the list with label of edge to vertex in tree v, positive edge leaving that vertex pv, and the generator for each. Again we are using the convention that $r^c = crc^{-1}$.

v	p_v	Generator
a	a	a^2
b	a	$baab$
b	b	b^2
c	a	$cac^{-1}a$
c	b	$cbc^{-1}b$
c^{-1}	a	$c^{-1}aca$
c^{-1}	b	$c^{-1}bcb$
ac^{-1}	a	$ac^{-1}ac$
ac^{-1}	b	${}^{a}(c^{-1}bcb)$
ab	a	$abab$
ab	b	$abba$
ac	a	$acac^{-1}$
ac	b	${}^{a}(cbc^{-1}b)$
bc	a	${}^{b}(cac^{-1}a)$
bc	b	$bcbc^{-1}$
bc^{-1}	a	${}^{b}(c^{-1}aca)$
bc^{-1}	b	$bc^{-1}bc$
cc	a	${}^{c}(cac^{-1}a)$
cc	b	${}^{c}(cbc^{-1}b)$
cc	c	c^4
abc	a	${}^{ab}(cac^{-1}a)$
abc	b	${}^{a}(bcbc^{-1})$
abc^{-1}	a	${}^{ab}(c^{-1}aca)$
abc^{-1}	b	${}^{a}(bc^{-1}bc)$
acc	a	${}^{ac}(cac^{-1}a)$
acc	b	${}^{ac}(cbc^{-1}b)$
acc	c	${}^{a}(c^4)$
bcc	a	${}^{bc}(cac^{-1}a)$
bcc	b	${}^{bc}(cbc^{-1}b)$
bcc	c	${}^{b}(c^4)$
abcc	a	${}^{abc}(cac^{-1}a)$
abcc	b	${}^{abc}(cbc^{-1}b)$
abcc	c	${}^{ab}(c^4)$

We can test using GAP that the set of generators given in the table is in fact a generating set for N.

Now that we have described an algorithm to construct a free generating set S for N, our next step will be to show how we can use this algorithm to rewrite a word as a product of elements of S. We will take our word find the first element in our product, then we will update our word by dividing off the element of S making our word a shorter product of elements of S at each step. We will also construct elements of S that we need while we do the rewriting process.

Algorithm 60 ***ProdConj***

Input: *A geodesic tree $\mathcal{T}$, a generating set for the free group F, a cc automaton $\mathcal{P}$, and a word $w \in N$ to be rewritten.*

Output: *The word w rewritten as a product of free generators.*

1 S:=[]; prod:=[]

2 while $w \neq \epsilon$ do

3 Trace w in $\mathcal{T}$, let b_v be the vertex in $\mathcal{T}$ where we cannot trace any further.

4 Let p_{b_v} be the first letter of $b_v^{-1}w$. Then p_{b_v} is an edge in $\mathcal{C}$ that is not in $\mathcal{T}$, so p_{b_v} is either a positive edge or a negative edge.

5 If p_{b_v} is a positive edge then do steps 4-6 in FreeGen, update S to include the new generator. If the generator is already in S we won't add a duplicate but we will make sure to use that generator. $w = generator^{-1} \cdot w$, Add(prod, generator)

6 If p_{b_v} is a negative edge, then $p_{b_v}^{-1}$ is a positive edge leaving $b_v = NormalForm(b_v p_{b_v})$. Apply steps 4-6 in FreeGen with $p_{b_v} = p_{b_v}^{-1}$. Update S to include the new generator. Let $w = generator \cdot w$, Add(prod, $generator^{-1}$)

7 return prod a list of generators which multiplied together gives w.

The only part of this algorithm which is not completely clear is that the algorithm will terminate.

Theorem 61 *ProdConj terminates.*

Proof: Since $w \in N$ it can be written uniquely as a product of elements in S. Therefore we have $w = \prod_{i=1}^{n}({}^{c_i}r_i)$. The algorithm begins by finding ${}^{c_1}r_1$ as it was defined as a generator. We then update $w = \prod_{i=2}^{n}({}^{c_i}r_i)$ which is a shorter product. Therefore since we find a generator at each step the algorithm terminates. The reason why we can conclude that we are in fact finding the correct element of S for the product, is that we compute the generator of S which we divide off in exactly the same way we computed elements of S when we defined them, so these elements will be the same. Since we showed that S generates N we know that every element can be rewritten in this way, so the algorithm ProdConj terminates. □

Chapter 7

ADAPTATION

Our next step will be to reduce our storage requirement even further. The current algorithm requires storing generators of S as we rewrite words, but also if we want to rewrite several words for the same group, we would need to store the generators of S found at each previous call as well, which would amount to constructing all of S in several calls to ProdConj.

Therefore, we wish to modify the way we compute generators for the set S by storing a list of paths that can be taken locally in the cc automaton $\mathcal{P}$ for each vertex v. The list will be called the **decision list** and it will consist of an ordered list of loops in $\mathcal{P}$ which pass through the given vertex and return to the trivial set. By storing this local information in the cc automaton, we can prevent violation of the redundancy condition without the need to store the set S which describes globally the generators of N in the full Cayley graph.

It should be noted that by definition of a loop the walk taken by any loop in $\mathcal{P}$ will never pass through any vertex more than once, and therefore the walk will also never pass through any edge more than once. Therefore, for each path ordering the decision list will be a finite list. To ensure we do not choose the same

element as a generator for two positive edges, we must test that the redundancy condition is not violated by the first loop in the decision list, if it is we will then move on to choose the second loop in the decision list. We will continue in this fashion until we find a generator which does not violate the redundancy condition.

Let us recall how redundancies could occur from our previous description. We essentially broke them down into two cases where we have a relator $c_2 p_{b_1} r_e$ with conjugating element c_1. In the first case we had $r_e = p_{b_2} p_{b_3} \cdots p_{b_m}$ is a path of positive edges. In the second case $r_e = p_{b_2}^{-1} p_{b_3} \tilde{p}$ is a path which contains at least one negative edge, and so we could draw pictures we assumed the negative edge occurs first, since if it occurred at a different point in the word the argument was the same. From our previous analysis we concluded that for any positive edge p_{b_i} we must be careful not to choose the cyclic conjugate $(c_2 p_{b_1} r_e)^{(c_2 p_{b_1} r_{e_1} \cdots r_{e_{i-2}})}$ where $r_{e_1} \cdots r_{e_{i-2}}$ is the prefix of r_e of length i-2. In the case of negative edges $p_{b_i}^{-1}$ we needed to be careful not to choose the inverse of the cyclic conjugate $((c_2 p_{b_1} r_e)^{(c_2 p_{b_1} r_{e_1} \cdots r_{e_{i-1}})})^{-1}$ as a relator. We also showed that it is possible to choose a relator which does not cause these redundancies in Theorem 48.

Essentially what we will do is adapt the definition of our generators, so that the generator for any positive edge $p_b \in X$ that leaves the vertex $b \in \mathcal{T}$ is the conjugate of the first loop in the decision list where the conjugating element is smaller then all of the other conjugating elements computed for any other cyclic conjugate of that relator that could violate the redundancy condition. In the case where there is no loop in the decision list where this property holds, we will update the decision list for other vertices by removing a relator to ensure this property

holds. Now we will describe an algorithm which computes generators which fulfill our adapted definition of generators.

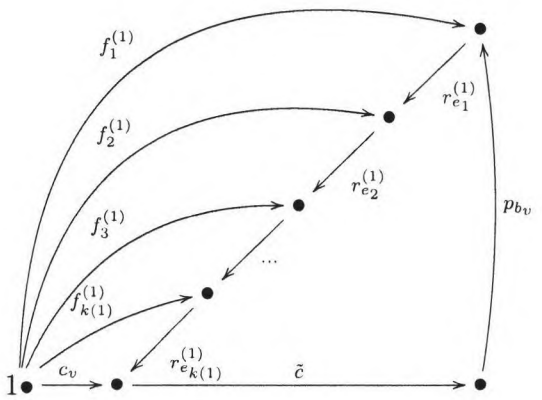
To help make it more clear which decision list we are looking at we introduce some notation.

Definition 62 *Let $p_b \in X$ be a positive edge leaving the vertex b in $\mathcal{T}$. Then in the algorithm FreeGen we computed c, the shortest prefix of the word bp_b such that $c^{-1}bp_b$ was a path in $\mathcal{P}$. Define $[bp_b]$ to be the vertex in $\mathcal{P}$ where we end after tracing $c^{-1}bp_b$ through $\mathcal{P}$.*

To prevent a redundant choice of generator we must make sure the path we choose back to the trivial vertex in $\mathcal{P}$ will not assign a relator where a cyclic conjugate of this relator or its inverse will be assigned later to a different positive edge.

We will begin by supposing we have a word w in Q that we wish to rewrite uniquely as a product of generators in S. We will then find the vertex b_v and the positive edge p_{b_v} as they were defined in the algorithm ProdConj. Next we compute the conjugating element c_v as defined in the algorithm FreeGen. At this point we can look at the decision list for $[b_v p_{b_v}]$ which will contain a list of relators which begin with a prefix $c_v^{-1}b_v p_{b_v}$. The relators will then have different suffixes $r_e^{(i)}$, where the superscript denotes that this is the i^{th} suffix such that each $c_v^{-1}b_v p_{b_v} r_e^{(i)}$ is a relator in $\mathcal{P}$. To help with this description we will draw a picture

for $r_e^{(1)} = r_{e_1}^{(1)} r_{e_2}^{(1)} \cdots r_{e_{k(1)}}^{(1)}$.



Now, for each $r_{e_j}^{(1)}$ if it is a positive edge in X we have a potential redundant definition with a cyclic conjugate of $c_v^{-1} b_v p_{b_v} r_e^{(1)}$. If instead $r_{e_j}^{(1)}$ is a negative edge then we have a potential redundant definition with the inverse of a cyclic conjugate of $c_v^{-1} b_v p_{b_v} r_e^{(1)}$ where $(r_{e_j}^{(1)})^{-1}$ is an edge in X. We begin by deciding if a potential redundancy can occur. To do this we must first compute the path ordering for each $[f_i^{(1)} r_{e_i}^{(1)}]$ where $r_{e_i}^{(1)}$ is an edge in X and see if the appropriate cyclic conjugate of $r_e^{(1)}$ is in its decision list. If the cyclic conjugate of $r_e^{(1)}$ is not in its decision list for any $r_{e_i}^{(1)}$ that correspond to positive edges in X, then we are free to choose $r_e^{(1)}$. Otherwise, we need to decide which of these positive edges should get $c_v^{-1} b_v p_{b_v} r_e^{(1)}$ or a cyclic conjugate as a relator. To make this decision we will compare the words c_v and the corresponding conjugating elements for the $r_{e_i}^{(1)}$ that are elements of X with respect to the ordering determined by our normal form. Whichever of these words is smallest will be assigned the appropriate cyclic conjugate of $c_v^{-1} b_v p_{b_v} r_e^{(1)}$ or its inverse. If for example, the vertex $f_i^{(1)}$ which gets assigned this relator has two positive edges $(r_{e_{i-1}}^{(1)})^{-1}$ and $r_{e_i}^{(1)}$ leaving it then the smaller of the two words $(r_{e_{i-1}}^{(1)})^{-1}$ or $(r_{e_i}^{(1)})$ will be assigned the relator. We do not actually care about

this tie break, unless $c_v = b_v$ and $(r^{(1)}_{e_{k(1)}})^{-1}$ is a positive edge so we need to determine if $b_v p_{b_v}$ or $b_v(r^{(1)}_{e_{k(1)}})^{-1}$ gets the relator $c_v^{-1} b_v p_{b_v} r^{(1)}_e$ or its inverse respectively.

If c_v is not the smallest word in the list $c_v, f^{(1)}_1, f^{(1)}_2, \cdots f^{(1)}_{k(1)}$ and the words that are smaller have a cyclic conjugate of $c_v^{-1} b_v p_{b_v} r^{(1)}_e$ as a relator in their decision list, then we cannot choose the relator $c_v^{-1} b_v p_{b_v} r^{(1)}_e$ as a relator for the positive edge $p_{b_v} \in X$. We will then go on to consider the relator $c_v^{-1} b_v p_{b_v} r^{(2)}_e$ and repeat the process.

Eventually we will either find a relator which works, or we will make the conjugating element c_v longer if possible, and repeat the process. If we still do not find a relator which works and $c_v = b_v$ then we will choose the shortest ending $r^{(i)}_e$ as our relator as long as the path orderings $[f^{(i)}_1 r^{(i)}_{e_1}], \cdots [f^{(i)}_{k(i)} r^{(i)}_{e_{k(i)}}]$ which contain a cyclic conjugate of $p_{b_v} r^{(i)}_e$ have decision list which contain more than one relator. This could happen, for example if b_v is the largest word in Q. We will then assign as the generator for $b_v p_{b_v}$ the element ${}^{c_v}(p_{b_v} r^{(i)}_e)$. Next we will remove the respective cyclic conjugates from the decision list of the path orderings $[f^{(i)}_1 r^{(i)}_{e_1}], \cdots [f^{(i)}_{k(i)} r^{(i)}_{e_{k(i)}}]$ which contain a cyclic conjugate of $p_{b_v} r^{(i)}_e$. Therefore we will guarantee that an overlap cannot occur. The only thing that may not be clear at this point is that a unique choice of generator exists for each positive edge $p_e \in X$.

Theorem 63 *The adapted algorithm uses a unique generator for each positive edge $p_e \in X$.*

Proof: Since we know from Theorem 48 that a unique relator exists for each $p_e \in X$, and we will only remove a relator from the decision list of a path ordering if the list

has more than one relator in it, we will be guaranteed that there is still a relator left that we can choose for each $p_e \in X$. □

Our next step will be to formally write up the adapted ProdConj algorithm.

Algorithm 64 ***ProdConjAdapt***

Input: *A geodesic tree $\mathcal{T}$, a generating set for the free group F, a cc automaton $\mathcal{P}$, and a word $w \in N$ to be rewritten.*

Output: *The word w rewritten as a product of free generators.*

1 S:=[]; prod:=[]

2 while $w \neq \epsilon$ do

3 Trace w in $\mathcal{T}$, let v be the vertex in $\mathcal{T}$ where we cannot trace any further. Let b_v be a prefix of w, which is a path from 1 to v in $\mathcal{T}$.

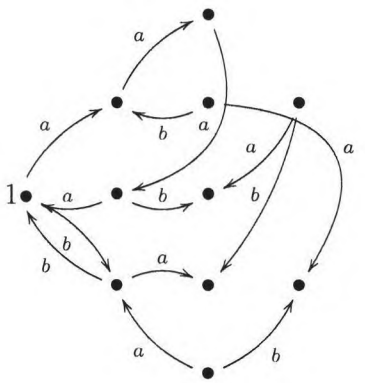
4 Let p_{b_v} be the first letter of $b_v^{-1}w$. Then p_{b_v} is an edge in $\mathcal{C}$ that is not in $\mathcal{T}$, so p_{b_v} is either a positive edge or a negative edge.

5 If p_{b_v} is a positive edge then do steps 4-6 in Free Gen. We will choose the relator from the decision list for $[b_v p_{b_v}]$ such that we do not cause a redundancy. Once we have a unique generator we will update w. $w = generator^{-1} \cdot w$, Add(prod, generator)

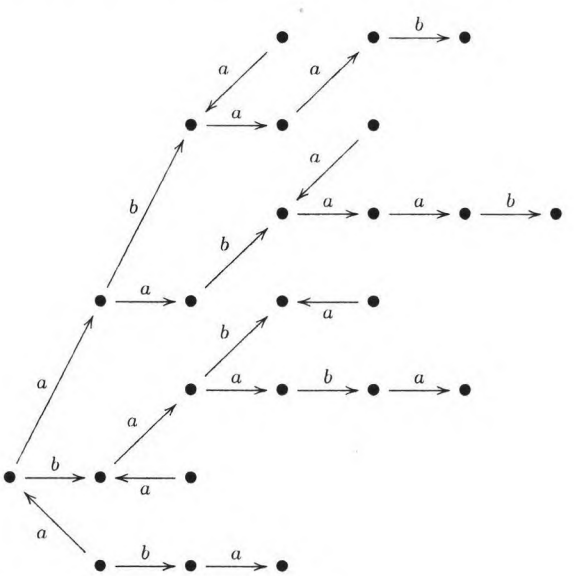
6 If p_{b_v} is a negative edge, then $p_{b_v}^{-1}$ is a positive edge leaving $b_v = NormalForm(b_v p_{b_v})$. Apply steps 4-6 in FreeGen with $p_{b_v} = p_{b_v}^{-1}$. We will choose the relator from the decision list for $[b_v p_{b_v}]$ such that we do not cause a redundancy. Let $w = generator \cdot w$, Add(prod, $generator^{-1}$)

7 return prod a list of generators which multiplied together gives w.

Finally we will end this description with an example. Consider $Q \cong S_4$ where S_4 is given as the finitely presented group $S_4 =< a, b | a^4 = b^2 = a^{-1}ba^{-1}ba^{-1}b >$. Then the cc automaton $\mathcal{P}$ is given by the following figure:



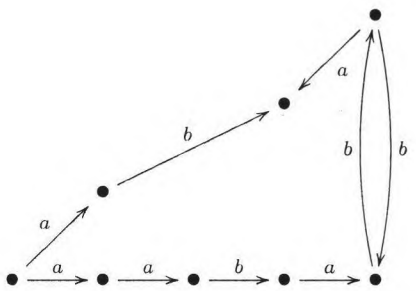
The geodesic tree $\mathcal{T}$ is then given by the following figure:



Next we will give the list of path orderings followed by their decision lists.

Path Ordering	**Decision List**
$[a]$	$[\ a^4,\ ab^{-1}ab^{-1}ab,\ ab^{-1}ab^{-1}ab^{-1},$ $a^3ba^{-1}ba^{-1}b, a^3ba^{-1}ba^{-1}b^{-1}\ ,\ ab^{-1}ab^{-1}a^2b^{-1}ab^{-1}a]$
$[ab^{-1}]$	$[\ ab^{-1}ab^{-1}ab,\ ab^{-1}ab^{-1}ab^{-1},\ ab^{-1}ab^{-1}a^2b^{-1}ab^{-1}a]$
$[a^{-1}b]$	$[a^{-1}ba^{-1}ba^{-1}b,\ a^{-1}ba^{-1}ba^{-1}b^{-1},$ $a^{-1}ba^{-1}ba^{-2}ba^{-1}ba^{-1}\ ]$
$[ba]$	$[bab^{-1}ab^{-1}a,\ bab^{-1}ab^{-1}a^{-3},$ $bab^{-1}ab^{-1}a^{-2}b^{-1}ab^{-1}ab]$
$[b^{-1}a]$	$[b^{-1}ab^{-1}ab^{-1}a,\ b^{-1}ab^{-1}ab^{-1}a^{-3},$ $b^{-1}ab^{-1}ab^{-1}a^{-2}b^{-1}ab^{-1}ab^{-1}]$
$[ba^{-1}]$	$[ba^{-1}ba^{-1}ba^{-1},\ ba^{-1}ba^{-1}ba^3,\ ba^{-1}ba^{-1}ba^2ba^{-1}ba^{-1}b$ $]$
$[b^{-1}a^{-1}]$	$[b^{-1}a^{-1}ba^{-1}ba^{-1},\ b^{-1}a^{-1}ba^{-1}ba^3,$ $b^{-1}a^{-1}ba^{-1}ba^2ba^{-1}ba^{-1}b^{-1}]$
$[a^{-1}ba^{-1}]$	$[a^{-1}ba^{-1}ba^{-1}b,\ a^{-1}ba^{-1}ba^{-1}b^{-1},$ $a^{-1}ba^{-1}ba^{-2}ba^{-1}ba^{-1}\ ]$
$[ba^{-1}b]$	$[ba^{-1}ba^{-1}ba^{-1},\ ba^{-1}ba^{-1}ba^3,\ ba^{-1}ba^{-1}ba^2ba^{-1}ba^{-1}b$ $]$
$[b^{-1}a^{-1}b]$	$[b^{-1}a^{-1}ba^{-1}ba^{-1},\ b^{-1}a^{-1}ba^{-1}ba^3,$ $b^{-1}a^{-1}ba^{-1}ba^2ba^{-1}ba^{-1}b^{-1}]$

In our example, suppose $b_1 = a^2ba$ is an edge label in $\mathcal{T}$ with positive edge $p_{b_1} = b$. Then we compute $f_1^{(1)} = NormalForm(a^2bab) = aba^{-1}$. Next we determine using $\mathcal{P}$ that $[b_1 p_{b_1}] = [b]$, and look at the decision list for this path ordering, the first relator is b^2. We will have the following picture to describe what is happening, if we choose b^2.



Therefore, the only redundant definition which can occur would be with the positive edge $p_b = b$ leaving the vertex aba^{-1} which is a shorter word. However, when we compute the path ordering for $[aba^{-1}b]$ we get that $[aba^{-1}b] = [ba^{-1}b]$ and b^2 is not a relator in the decision list for $ba^{-1}b$, so therefore we can choose ${}^{a^2ba}(b^2)$ as a generator for $b_1 p_{b_1}$ without any risk that this relator causes a redundancy.

Chapter 8

ARITHMETIC IN A GROUP EXTENSION USING THE PARTIAL AUTOMATON

In this chapter, we will describe in detail how to do arithmetic in a group extension. Recall in Chapter 3, we described an algorithm which did arithmetic in a group extension, assuming we had a normal form for Q and K along with a procedure which rewrites words in N as a product of conjugates of relators. In the previous chapter we gave an algorithm that rewrote words in N as a product of conjugates of relators. However, we have expanded the set of relators to include all cyclic conjugates of the relators for Q. Unfortunately, since we only want to store a 2-cocycle for the relators given for Q we still have one further rewriting step to apply. The claim is that if we get a loop in $\mathcal{P}$ which is not a relator for Q, that loop in $\mathcal{P}$ can be rewritten only using the relators of Q. This rewriting can be done using the partial automaton $\mathcal{A}$ which only has loops defined for the given relators of Q.

To make this rewriting unique we must make two further assumptions about our presentation. The first is that our presentation does not include two relators of the form $abc = 1$ and $adc = 1$. If our presentation does include relators of this form we will apply the Tietze Transformation which keeps the first relator $abc = 1$

and changes the second relator to $bd^{-1} = 1$. Second, if our presentation contains two relators of the form $abde = 1$ and $bfea = 1$ we will apply the Tietze Transformation which keeps the first relator $abcde = 1$ and changes the second relator to $df^{-1} = 1$. Assuming these transformations have been applied we can uniquely rewrite all loops in $\mathcal{P}$ using the partial automaton $\mathcal{A}$ so that the loops in $\mathcal{P}$ can be rewritten as a product of conjugates of the given relators for Q.

To begin describing this algorithm, or first claim is that every loop in $\mathcal{P}$ is either a cyclic conjugate of a relator, or a product of a cyclic conjugates of relators.

Theorem 65 *There exists an algorithm which will rewrite every loop in $\mathcal{P}$ as a product of cyclic conjugates of relators.*

Proof: To prove this we will show that the folded graph Γ obtained from applying all possible foldings to the bouquet $\mathcal{B}$ defined to have one loop for each cyclic conjugate of a relator is isomorphic to $\mathcal{P}$. We will do this by constructing a function $f : vert(\Gamma) \to vert(\mathcal{P})$, with the property that if x is the label of an edge in Γ, and γ is the label of a vertex in Γ then $f(\gamma^x) = f(\gamma)^x$. Suppose $\gamma \in vert(\Gamma)$, then there exists a path from 1 to γ in Γ with label G, so that $1^G = \gamma$ in Γ. Then we claim there exists a vertex σ in $\mathcal{P}$, such that $1^G = \sigma$. Since G is a path in Γ, we know that G must be the prefix of a cyclic conjugate of a relator or the inverse of a cyclic conjugate of a relator. Since $\mathcal{P}$ is defined so that all cyclic conjugates of relators and there inverses must trace through $\mathcal{P}$ starting at 1, we know that G must be a path in $\mathcal{P}$. We then define $f(\gamma) = \sigma$. Then clearly f is onto $vert(\mathcal{P})$. Next consider

$ker(f) = \{\gamma \in vert(\Gamma) | f(\gamma) = 1 \text{ where } 1 \in vert(\mathcal{P})\}$

$= \{\gamma \in vert(\Gamma)$ where G is the label of the path in Γ from 1 to $\gamma | 1^G = 1$ in $\mathcal{P}\}$

$= \{1 \in vert(\Gamma)\}$

Therefore we conclude that f is a one to one map so $\mathcal{P} \cong \Gamma$. $\square$

Next we give an algorithm which rewrites any path in $\mathcal{P}$ as a product of cyclic conjugates of relators given for Q.

Algorithm 66 ***ProdCyclicConj***

Input: *A geodesic tree* $\mathcal{T}$*, a partial automaton* $\mathcal{A}$*, and a loop L in the cc automaton* $\mathcal{P}$ *to be rewritten.*

Output: *The loop L rewritten as a product of cyclic conjugates of relators.*

1 S:=[]; prod:=[]

2 while $L \neq \epsilon$ *do*

3 Trace w in $\mathcal{T}$*, let* b_v *be the vertex in* $\mathcal{T}$ *where we cannot trace any further.*

4 Let p_{b_v} *be the first letter of* $b_v^{-1}w$*. Then* p_{b_v} *is an edge in* $\mathcal{C}$ *that is not in* $\mathcal{T}$*, so* p_{b_v} *is either a positive edge or a negative edge.*

5 If p_{b_v} *is a positive edge then do steps 4-6 in FreeGen, except on step 5 choose the path back so that we have a cyclic conjugate of a relator and in step 6 we do not need to check for redundancy. Update S to include the new generator. If the generator is already in S we won't add a duplicate but we will make sure to use that generator.* $w = generator^{-1} \cdot w$*, Add(prod, generator)*

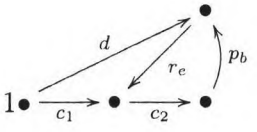
6 *If* p_{b_v} *is a negative edge, then* $p_{b_v}^{-1}$ *is a positive edge leaving* $b_v = NormalForm(b_v p_{b_v})$. *Apply steps 4-6 in FreeGen with* $p_{b_v} = p_{b_v}^{-1}$ *except on step 5 choose the path back so that we have a cyclic conjugate of a relator and in step 6 we do not need to check for redundancy. Update* S *to include the new generator. Let* $w =$ $generator \cdot w$, *Add(prod,* $generator^{-1}$*)*

7 *return prod a list of generators which multiplied together gives* w.

It is clear that if this algorithm terminates the result will be correct, therefore the only part we need to prove is that the algorithm will always terminate.

Theorem 67 *ProdCyclicConj will always terminate.*

Proof: Since by theorem 65 we proved that every loop L *in* $\mathcal{P}$ *can be written as a product of cyclic conjugates of relators or their inverses we know that* $L = \prod_{i=1}^{n} r_i^{x_i}$ *where the* r_i *are relators or inverses of relators for* Q *and the* $x_i \in F$ *such that* $r_i^{x_i}$ *is a cyclic conjugate of* r_i. *We will show that the algorithm will uniquely compute this product so the algorithm will terminate with* $\prod_{i=1}^{n} r_i^{x_i}$ *returned. Let us suppose* L *begins with* bp_b *where* b *is a prefix of* K *that is in* $\mathcal{T}$ *and* p_b *is a positive edge leaving* $\mathcal{T}$. *Also, suppose* c_1 *is a prefix of* bp_b *such that* $c_1^{-1} bp_b$ *traces through* $\mathcal{A}$, *and let* $b = c_1 c_2$. *Then we have the following picture.*



The claim is that there exists a unique r_e *such that* ${}^{c_1}(c_2 p_b r_e)$ *is the first cyclic conjugate of a relator* L. *We will prove this by contradiction, so first suppose there*

exists a different path r in $\mathcal{A}$ such that ${}^{c_1}(c_2p_br)$ is also a cyclic conjugate of a relator. Let $r_e = r_{e_1}r_{e_2}\cdots r_{e_n}$ and $r = r_1\cdots r_m$ then we have

$$
\begin{aligned}
{}^{c_1}(c_2p_br_e) &= c_1c_2p_br_{e_1}\cdots r_{e_n}c_1^{-1} \\
&= r_{e_i}\cdots r_{e_n}c_2p_br_{e_1}\cdots r_{e_{i-1}}
\end{aligned}
$$

since this is a cyclic conjugate of a relator and

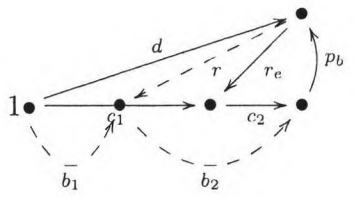
$$
\begin{aligned}
{}^{c_1}(c_2p_br) &= c_1c_2p_br_1\cdots r_mc_1^{-1} \\
&= r_j\cdots r_mc_2p_br_1\cdots r_{j-1}
\end{aligned}
$$

Now since $c_1c_2p_b = c_1c_2p_b$ we conclude

$$r_{e_i}\cdots r_{e_n} = r_j\cdots r_m = a$$

Let $r_{e_1}\cdots r_{e_{i-1}} = x$ and $r_1\cdots r_{j-1} = y$ then we have two relators $c_2p_bxa = 1$ and $c_2p_bya = 1$ in $\mathcal{A}$. However, we assumed that this could not happen by our first assumption on the relators for Q. So therefore we cannot have another path in $\mathcal{A}$ leaving c_2p_b.

The only other way we could get a different choice for a cyclic conjugate of a relator is if there exists another prefix b_1 of bp_b such that $b_1^{-1}bp_b$ traced through $\mathcal{A}$ and $b = b_1b_2$ so that ${}^{b_1}(b_2p_br)$ is another cyclic conjugate of a relator where $b_1 \neq c_1$. We would then have the following picture.



We suppose without loss of generality that b_1 is a prefix of c_1 so that $c_1 = b_1 d$ and $b_2 = dc_2$. Then if we again let $r = r_1 \cdots r_m$ we have

$$^{b_1}(b_2 p_b r) = b_1 b_2 p_b r b_1^{-1}$$

$$= b_1 d c_2 p_b r b_1^{-1}$$

$$= r_j \cdots r_m d c_2 p_b r_1 \cdots r_{j-1}$$

and similarly

$$^{c_1}(c_2 p_b r_e) = c_1 c_2 p_b r_e c_1^{-1}$$

$$= b_1 d c_2 p_b r_e d^{-1} b_1^{-1}$$

$$r_{e_i} \cdots r_{e_m} c_2 p_b r_{e_1} \cdots r_{e_{i-1}}$$

So therefore $r_j \cdots r_m d = r_{e_i} \cdots r_{e_m} = ad$. We then have the following 2 relators for Q:

$$d c_2 p_b y a = 1$$

and

$$c_2 p_b x a d = 1$$

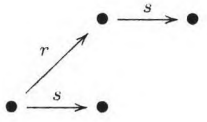
However, we assumed that this could not happen by the second assumption on the relators for Q, so therefore we only have one choice for the first cyclic conjugate of a relator for L and hence $r_1^{x_1} =^{c_1} (c_2 p_b r_e)$. We will continue this process, and eventually ProdCyclicConj will return the finite product $\prod_{i=1}^{n} r_i^{x_i}$ since at no step will we have any choice involved. □

Now that we have described a process which rewrites all loops in $\mathcal{P}$ as a product of cyclic conjugates of relators our final step is to show an example where we

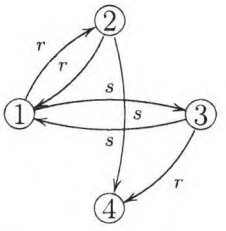
do arithmetic in a group extension.

In this example, suppose $K \cong C_2 =< a|a^2 = 1 >$ where the list of elements in normal form are $\{1, a\}$. Let $Q \cong V_4 =< r, s|r^2 = 1, s^2 = 1, rsr^{-1}s^{-1} = 1 >$ where the list of elements in normal form are $\{1, r, s, rs\}$. If we choose 2-cocycles $m_{r^2} = a$, $m_{s^2} = 1$, and $m_{rsr^{-1}s^{-1}} = a$ we have determined our extension group G, which in this case is D_8. We do not actually need to know G to do arithmetic. It turns out that since Q is so small the cc automaton $\mathcal{P}$ is the full coset automaton. Therefore, we only need to do the second step of the rewriting which uses the partial automaton $\mathcal{A}$ and the geodesic tree $\mathcal{T}$ to do the rewriting. We show these two graphs below.

$\mathcal{T}$

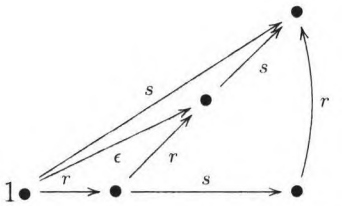


$\mathcal{A}$

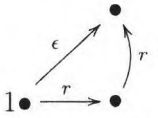


Suppose we wish to compute the normal form of $rsar$ in G. Then our first step would be to move the a which is an image of K to the right, so $rsar \rightarrow rsra^r = rsra$ since $Aut(K) =< 1 >$. Next we compute the normal form of rsr in Q: $NormalForm_Q(rsr) = s$. Then we need to compute the 2-cocycles so we must rewrite the product $rsrs^{-1}$ as a product of conjugates of relators for Q. We see that rs is an edge in $\mathcal{T}$, so r is our positive edge in X. Since rsr does not trace through $\mathcal{A}$ we have r is our conjugating element. The following diagram depicts

this situation.



We see in $\mathcal{A}$ we only have one choice of path back from sr, and that is $s^{-1}r^{-1}$. Therefore our first conjugate of a relator is ${}^{r}(srs^{-1}r^{-1}) = ({}^{r}(rsr^{-1}s^{-1}))^{-1}$. We update our loop $rsrs^{-1}$ by dividing off $({}^{r}(rsr^{-1}s^{-1}))^{-1}$ we get $({}^{r}(rsr^{-1}s^{-1}))rsrs^{-1} = r^2$. Then since r is an edge in $\mathcal{T}$ and r is a positive edge we get the following diagram.



From this we get our second conjugate of a relator, so that we have

$$rsrs^{-1} = ({}^{r}(rsr^{-1}s^{-1}))^{-1}r^2.$$

Now to compute the normal form of $rsra$ in G we have

$$NormalForm_G(rsra) = NormalForm_Q(rsr)\cdot NormalForm_K(({}^{r}(m_{rsr^{-1}s^{-1}}))^{-1}m_{r^2}\cdot a)$$

$$= s \cdot NormalForm_K({}^{r}(a^{-1})(a)(a))$$

$$= s \cdot NormalForm_K(a^{-1}(a)(a))$$

Since $Aut(C_2) =< 1 >$

$$= s \cdot a$$

We can check our result using the confluent rewriting system for D_8 which we gave earlier: $D_8 =< r, s, a | r^2 \to 1, s^2 \to 1, rs \to sra, a^2 \to 1 >$. Therefore we have

$$\underline{rs}ar \to sr\underline{aa}r \to s\underline{r^2} \to sa$$

Chapter 9

LARGER EXAMPLES

As a final step we will show a larger example, since it is not obvious from our previous work that the new algorithm reduces storage. The author has implemented the algorithm in GAP and this implementation has been used to do the following computations. Let $Q = PSL_3(4) =< a, b | a^2 = b^4 = (ab)^7 = (ab^2)^5 = (abab^2)^7 = 1 >$ and $K = C_3 =< c | c^3 = 1 >$. In this case if we choose the 2-cocycles $m_{a^2} = 1$, $m_{b^4} = 1$, $m_{(ab)^7} = c$, $m_{(ab^2)^5} = 1$ and $m_{(abab^2)^7} = c$ we get the non-split extension group $G = SL_3(4)$.

Suppose we wish to compute the normal form of the randomly chosen word $w = ba^{-1}b^2ab^2aba^{-1}b^{-1}c$ in G. Then we begin by computing the normal form of $\tilde{w} = ba^{-1}b^2ab^2aba^{-1}b^{-1}$ in Q. The $NormalForm_Q(\tilde{w}) = b^{-1}ab^2ab^{-1}ab^{-1}$, therefore

$$v = \tilde{w} \cdot NormalForm_Q(\tilde{w})^{-1}$$

$$= ba^{1-}b^2ab^2aba^{-2}ba^{-1}b^{-2}a^{-1}b \in N$$

Our first step will be to use the cc automaton $\mathcal{P}$ and the geodesic tree $\mathcal{T}$ to rewrite v as a product of generators for N. In this example, since $|Q| = 20,160$ the geodesic tree has 20,160 vertices which is much to large to include a drawing of here. The cc automaton is significantly smaller containing only 469 vertices,

however again its image is difficult to draw. Using the adapted algorithm we rewrite

$$v =^{b} (a^2)\cdot(bab^2ab^2ab^2ab)\cdot(ab)^{-7}\cdot^{b^{-1}}(ab^3(a^{-1}b^{-1})^6)^{-1}\cdot^{b^{-1}ab^2}(a^{-2})\cdot^{b^{-1}ab^2ab^{-1}a}(a^{-2})$$

We can check by hand that this result is correct. However, at this point a few of the elements in this product are not just conjugates of relators given for Q. We will identify these elements by underlining them.

$$^{b}(a^2)\cdot\underline{(bab^2ab^2ab^2ab)}\cdot(ab)^{-7}\cdot\underline{^{b^{-1}}(ab^3(a^{-1}b^{-1})^6)^{-1}}\cdot^{b^{-1}ab^2}(a^{-2})\cdot^{b^{-1}ab^2ab^{-1}a}(a^{-2})$$

The underlined elements are cyclic conjugates or products of cyclic conjugates of the relators given for Q so they need to be rewritten as a product of conjugates of the relators given for Q. To do this rewriting we use the partial automaton $\mathcal{A}$ which only has loops for each relator. This graph only contains 89. Using $\mathcal{A}$ to rewrite v we get:

$$v =^{b} (a^2)\cdot^{b}((ab^2)^5)\cdot^{-b}(a^{-2})\cdot^{b^{-1}a}(b^{-4})\cdot^{b^{-1}ab^2}(a^{-2})\cdot^{b^{-1}ab^2ab^{-1}a}(a^{-2})$$

Now at this point our final step is to use this rewriting to compute the normal form in G of w.

$$NormalForm_G(w) = NormalForm_Q(\tilde{w})$$

$$\cdot NormalForm_K(^{b}(m_{a^2})\cdot^{b}(m_{(ab^2)^5})\cdot^{-b}(m_{a^{-2}})\cdot^{b^{-1}a}(m_{b^{-4}})\cdot^{b^{-1}ab^2}(m_{a^{-2}})\cdot^{b^{-1}ab^2ab^{-1}a}(m_{a^{-2}})\cdot c)$$

$$= b^{-1}ab^2ab^{-1}ab^{-1}\cdot NormalForm_K(^{b}(1)\cdot^{b}(1)\cdot^{-b}(1)\cdot^{b^{-1}a}(1)\cdot^{b^{-1}ab^2}(1)\cdot^{b^{-1}ab^2ab^{-1}a}(1)\cdot c)$$

Now in this case $Aut(C_3) \cong C_2$, however both automorphisms map $1 \mapsto 1$, so the product in K simplifies to c. Therefore we conclude that the $NormalForm_G(w) = b^{-1}ab^2ab^{-1}ab^{-1}c$. We have now seen in this example that using the partial automaton to do arithmetic in group extensions allows us to save on storage so we can

work with larger groups.

To show the power of this algorithm, we also ran it on a much larger example, for $Q = PSp_6(2) =< a, b | a^2 = b^7 = (ab)^9 = (a^{-1}b^1a^{-1}b^{-1}a^{-1}b^{-1}ababab)^2 = (a^{-1}b^{-1}ab)^3 = (a^{-1}b^{-2}ab^2)^2 >$ which is a group of order $1,451,520$ elements. At this size one can argue that rewriting can hardly be done by constructing an augmented coset automaton. The cc automaton has 145 states and the partial automaton has only 52 states. The only part of our algorithm that took any time was the construction of the geodesic tree and decision list. We rewrote several words that are elements of N using the algorithm and calculated the run time to rewrite these words once those two lists were in place and the run time for this rewriting took about a half a second. The results of rewriting using the algorithm implemented in GAP are included below.

$word = a^{-3}ba^{-2} \cdot ba^{-3}b^{-1} \cdot a^{-1}b^2a^{-1} \cdot ba^{-2}b^{-1} \cdot ab^{-2}a^{-1} \cdot ba^{-1}b^{-2}a^{-1}$

$\Longrightarrow (a^{-2}) \cdot (a^{-2}) \cdot ({}^{(ab)}(a^{-2})) \cdot ({}^{(ab^2)}(a^{-2})) \cdot ({}^{(ab^2)}(a^{-2})) \cdot ({}^{(ab^2ab^{-1})}(a^{-2})) \cdot ({}^{(ab^2ab^{-1}ab^2)}(a^{-2})) \cdot ({}^{(ab^2ab^{-1}ab^2ab)}(a^{-2})) \cdot ({}^{(ab^2ab^{-1}ab^2)}(a^2))$ (Rewritten in $\mathcal{P}$)

$\Longrightarrow (a^{-2}) \cdot (a^{-2}) \cdot ({}^{(ab)}(a^{-2})) \cdot ({}^{(ab^2)}(a^{-2})) \cdot ({}^{(ab^2)}(a^{-2})) \cdot ({}^{(ab^2ab^{-1})}(a^{-2})) \cdot ({}^{(ab^2ab^{-1}ab^2)}(a^{-2})) \cdot ({}^{(ab^2ab^{-1}ab^2ab)}(a^{-2})) \cdot ({}^{(ab^2ab^{-1}ab^2)}(a^2))$ (Rewritten in $\mathcal{A}$)

$word = b^{-1}a^{-2}b \cdot a^{-1}b^{-1}a \cdot b^{-1}ab \cdot a^2b^{-2}a^{-4} \cdot ba^{-1}b \cdot a^{-1}ba^{-1}$

$\Longrightarrow ({}^{(b^{-1})}(a^{-2})) \cdot (a^{-2}) \cdot ({}^{(ab^{-1}ab^{-1}ab)}(a^2)) \cdot ({}^{(ab^{-1}ab^{-1}ab^{-1})}(a^{-2}) \cdot ({}^{(ab^{-1}ab^{-1}ab^{-1})}(a^{-2}))$ (Rewritten in $\mathcal{P}$)

$\Longrightarrow ({}^{(b^{-1})}(a^{-2})) \cdot (a^{-2}) \cdot ({}^{(ab^{-1}ab^{-1}ab)}(a^2)) \cdot ({}^{(ab^{-1}ab^{-1}ab^{-1})}(a^{-2}) \cdot ({}^{(ab^{-1}ab^{-1}ab^{-1})}(a^{-2}))$ (Rewritten in $\mathcal{A}$)

$word = a^{-4}b^{-5}a^{-1} \cdot b^2a^{-1}b^2 \cdot a^{-1}b^{-2}a \cdot b^2a^{-1}b^3 \cdot a^{-1}b^{-2}$

$\Longrightarrow (a^{-2})\cdot(a^{-2})\cdot(b^{-7})\cdot({}^{(b^2)}(a^{-2}))\cdot({}^{(b^2a)}(b^7))\cdot({}^{(b^2ab^{-3})}(b^{-2}a^{-1}b^2a^{-1})^2)\cdot({}^{(b^2ab^{-3}ab^{-2})}(a^2))$

(Rewritten in $\mathcal{P}$)

$\Longrightarrow (a^{-2})\cdot(a^{-2})\cdot(b^{-7})\cdot({}^{(b^2)}(a^{-2}))\cdot({}^{(b^2a)}(b^7))\cdot({}^{(b^2ab^{-5}a^{-1}b^2)}(a^{-2}))\cdot({}^{(b^2ab^{-3})}(b^{-2}a^{-1}b^2a)^2)\cdot$

$({}^{(b^2ab^{-3})}(a^{-2}))\cdot({}^{(b^2ab^{-3}ab^{-2})}(a^2))$ (Rewritten in $\mathcal{A}$)

$word = aba^{-2} \cdot b^{-1}a^{-1}b^{-3} \cdot a^2ba \cdot b^2a^{-2}b^5 \cdot a^{-1}b^2$

$\Longrightarrow ({}^{(ab)}(a^{-2}))\cdot({}^{(b^{-3})}(a^2))\cdot({}^{(b^{-2})}(a^2))\cdot(b^{-2}a^{-1}b^2a^{-1}b^{-2}a^{-1}b^2a^{-1})\cdot({}^{(ab^{-2})}(a^2))\cdot(a^2)\cdot$

$(a^{-1}b^{-2}a^{-1}b^2a^{-1}b^{-2}a^{-1}b^2)\cdot({}^{(b^{-2}a)}(b^7))$ (Rewritten in $\mathcal{P}$)

$\Longrightarrow ({}^{(ab)}(a^{-2}))\cdot({}^{(b^{-3})}(a^2))\cdot({}^{(b^{-2})}(a^2))\cdot({}^{(b^{-2}a^{-1}b^2)}(a^{-2}))\cdot(b^{-2}a^{-1}b^2ab^{-2}a^{-1}b^2a)\cdot a^{-2}\cdot$

$({}^{(ab^{-2})}(a^2))\cdot(a^2)\cdot({}^{(a^{-1}b^{-2})}(a^{-2}))\cdot(a^{-1}b^{-2}ab^2a^{-1}b^{-2}ab^2)\cdot({}^{(b^{-2}a^{-1})}(a^{-2}))\cdot({}^{(b^{-2}a)}(b^7))$

(Rewritten in $\mathcal{A}$)

$word = ab^2a^3 \cdot b^{-1}a^{-1}b^3 \cdot a^2b^3a \cdot b^{-1}a^{-2}b \cdot a^{-1}ba^{-1} \cdot ba^{-1}b^{-2}a^{-1}$

$\Longrightarrow ({}^{(ab^2)}(a^2))\cdot({}^{(ab^2ab^{-1})}(a^{-2}))\cdot({}^{(ab^2ab^{-1}ab^3)}(a^2))\cdot({}^{(ab^2ab^{-1}a)}(b^7))\cdot({}^{(ab^2ab^{-1}ab^{-1}ab^{-1})}(a^{-2}))$

(Rewritten in $\mathcal{P}$)

$\Longrightarrow ({}^{(ab^2)}(a^2))\cdot({}^{(ab^2ab^{-1})}(a^{-2}))\cdot({}^{(ab^2ab^{-1}ab^3)}(a^2))\cdot({}^{(ab^2ab^{-1}a)}(b^7))\cdot({}^{(ab^2ab^{-1}ab^{-1}ab^{-1})}(a^{-2}))$

(Rewritten in $\mathcal{A}$)

$word = aba^{-1}\cdot b^{-4}ab^{-1}\cdot a^2b^{-1}a\cdot b^{-1}ab\cdot aba^{-2}\cdot b^{-1}a^{-1}b^{-1}\cdot a^{-1}ba^{-1}\cdot b^2a^{-1}b^{-3}\cdot a^{-1}b^{-1}a^{-1}$

$\Longrightarrow ({}^{(ab)}(a^{-2}))\cdot({}^{(aba)}(b^{-7}))\cdot({}^{(abab^3ab^{-1})}(a^2))\cdot({}^{(abab)}(b^2ab^{-2}ab^2ab^{-2}a))\cdot({}^{(abab)}(a^{-2}))\cdot$

$({}^{(abababb^2)}(a^{-2}))\cdot({}^{(abababb^2ab^{-3}abab)}(a^{-2}))\cdot({}^{(abababb^2)}(a^2))\cdot({}^{(abab)}(a^2))\cdot({}^{(abab)}(a^{-1}b^2a^{-1}b^{-2})^2)$

(Rewritten in $\mathcal{P}$)

$\Longrightarrow ({}^{(ab)}(a^{-2}))\cdot({}^{(aba)}(b^{-7}))\cdot({}^{(abab^3ab^{-1})}(a^2))\cdot({}^{(abab^3ab^{-2})}(a^2))\cdot({}^{(abababb^2)}(b^{-2}a^{-1}b^2a)^2)\cdot$

$({}^{((ab)^2ab^2)}(a^{-2}))\cdot({}^{((ab)^2ab^2ab^{-3}abab)}(a^{-2}))\cdot({}^{((ab)^2ab^2)}(a^2))\cdot({}^{(abab^3a)}(a^{-1}b^{-2}ab^2)^2)\cdot$

$({}^{(abab^3ab^{-2})}(a^{-2}))$ (Rewritten in $\mathcal{A}$)

$word = a^4b^2a^6 \cdot b^{-1}a^{-1}b \cdot a^{-1}ba^{-2} \cdot b^{-1}a^{-1}b^{-1} \cdot a^{-1}b^{-1}$

$\Longrightarrow (a^2) \cdot (a^2) \cdot ({}^{(b^2)}(a^2)) \cdot ({}^{(b^2)}(a^2)) \cdot ({}^{(b^2)}(a^2)) \cdot ({}^{(b)}(a^{-2})) \cdot ({}^{(bab)}(a^{-2})) \cdot ({}^{(babab)}(a^{-2})))$

(Rewritten in $\mathcal{P}$)

$\Longrightarrow (a^2) \cdot (a^2) \cdot ({}^{(b^2)}(a^2)) \cdot ({}^{(b^2)}(a^2)) \cdot ({}^{(b^2)}(a^2)) \cdot ({}^{(b)}(a^{-2})) \cdot ({}^{(bab)}(a^{-2})) \cdot ({}^{(babab)}(a^{-2})))$

(Rewritten in $\mathcal{A}$)

$word = ab^{-3}a \cdot b^{-1}ab \cdot a^{-1}ba^{-3} \cdot b^{-1}a^{-1}b \cdot a^2b^{-1}a^{-1} \cdot ba^{-1}b^{-1} \cdot a^{-1}b^{-1}a^{-1} \cdot ba^{-1}b^3a^{-1}$

$\Longrightarrow ({}^{(ab^{-3}ab^{-1}ab)}(a^{-2})) \cdot ({}^{(ab^{-3}ab^{-1}abab)}(a^{-2})) \cdot ({}^{(ab^{-3}ab^{-1}abab)}(a^{-2})) \cdot ({}^{(ab^{-3}ab^{-1}ababab^{-1})}(a^{-2})) \cdot$
$({}^{(ab^{-3}ab^{-1}ab)}(abab^{-1})^3) \cdot ({}^{(ab^{-3}ab^{-1})}(a^2)) \cdot ({}^{(ab^{-3}a)}(b(b^{-2}a^{-1}b^2a^{-1})^2b^{-1})) \cdot ({}^{(ab^{-3}abab^{-2}ab)}(a^{-2})) \cdot$
$({}^{(ab^{-3}abab^{-2}abab)}(a^2)) \cdot ({}^{(ab^{-3}abab^{-2}ab)}(a^2)) \cdot ({}^{(ab^{-3}a)}(bab^{-2}ab^2ab^{-2}ab)) \cdot ({}^{(ab^{-3}ab^{-1})}(a^{-2})) \cdot$
$({}^{(ab^{-3}ab^{-1}ab)}(ba^{-1}b^{-1}a^{-1})^3)$ (Rewritten in $\mathcal{P}$)

$\Longrightarrow ({}^{(ab^{-3}ab^{-1}ab)}(a^{-2})) \cdot ({}^{(ab^{-3}ab^{-1}abab)}(a^{-2})) \cdot ({}^{(ab^{-3}ab^{-1}abab)}(a^{-2})) \cdot ({}^{(ab^{-3}ab^{-1}ababab^{-1})}(a^{-2})) \cdot$
$({}^{(ab^{-3}ab^{-1}abab)}(a^2)) \cdot ({}^{(ab^{-3}ab^{-1}ab^2a)}((a^{-1}b^{-1}ab)^3) \cdot ({}^{(ab^{-3}ab^{-1}ab^2ab^{-1}a^{-1}b)}(a^2)) \cdot ({}^{(ab^{-3}ab^{-1}ab^2)}(a^2)) \cdot$
$({}^{(ab^{-3}aba)}(a^{-1}b^{-2}ab^2)^2) \cdot ({}^{(ab^{-3}abab^{-2})}(a^{-2})) \cdot ({}^{(ab^{-3}abab^{-2}ab)}(a^{-2})) \cdot ({}^{(ab^{-3}abab^{-2}abab)}(a^2)) \cdot$
$({}^{(ab^{-3}abab^{-2}ab)}(a^2)) \cdot ({}^{(ab^{-3}ab)}(a^2)) \cdot ({}^{(ab^{-3}ab)}(a^{-1}b^{-2}ab^2)^2) \cdot ({}^{(ab^{-3}ab^{-1}a^{-1}b^2)}(a^2)) \cdot ({}^{(ab^{-3}ab^{-1})}(a^{-2})) \cdot$
$({}^{(ab^{-3}ab^{-1}ab^2a^{-1}b^{-1})}(a^{-2})) \cdot ({}^{(ab^{-3}ab^{-1}aba^{-1}ba)}(a^{-1}b^{-1}ab)^3) \cdot ({}^{(ab^{-3}ab^{-1}aba^{-1}bab^{-1})}(a^{-2})) \cdot$
$({}^{(ab^{-3}ab^{-1}ab)}(a^{-2}))$ (Rewritten in $\mathcal{A}$)

$word = ba^{-1}b \cdot aba^{-1} \cdot ba^{-1}b^2 \cdot aba \cdot b^{-1}ab \cdot a^{-1}b^{-1}a^{-1} \cdot b^{-1}a^{-1}b \cdot a^{-1}ba^{-1} \cdot b^{-1}a^{-1}b^2 \cdot$
$a^{-1}ba^{-1}$

$\Longrightarrow ({}^{(b)}(a^{-2})) \cdot ({}^{(babab)}(a^{-2})) \cdot ({}^{(bababab)}(a^{-2})) \cdot (ba)^9 \cdot (a^{-2}) \cdot ({}^{(ab^{-1})}(a^{-2})) \cdot ({}^{(ab^{-1}ab^{-1})}(a^{-2})) \cdot$
$({}^{(ab^{-1}ab^{-1}ab^{-1})}(a^{-2})) \cdot ({}^{((ab^{-1})^4)}(a^{-2})) \cdot ({}^{(ab^{-1}ab^{-1}ab^{-1}ab^{-1}ab)}(abab^{-1})^3) \cdot ({}^{(ab^{-1}ab^{-1}ab^{-1}ab^{-1})}(a^2)) \cdot$
$({}^{((ab^{-1})^3ab)}(b^{-2}a^{-1}b^2a^{-1})^2) \cdot ({}^{(ab^{-1}ab^{-1}ab^{-1}abab^{-2}ab)}(a^{-2})) \cdot ({}^{(ab^{-1}ab^{-1}ab^{-1}abab^{-2}abab)}(a^{-2})) \cdot$

$({}^{(ab^{-1}ab^{-1}ab^{-1}ab)}(a^2)) \cdot ({}^{(ab^{-1}ab^{-1}ab^{-1})}(a^2)) \cdot ({}^{(ab^{-1}ab^{-1})}(a^2)) \cdot ({}^{(ab^{-1}ab^{-1})}(a^{-1}b^{-1}a^{-1}b)^3)$

(Rewritten in $\mathcal{P}$)

$\Longrightarrow ({}^{(b)}(a^{-2}))\cdot({}^{(bababab)}(a^{-2}))\cdot({}^{(bababab)}(a^{-2}))\cdot({}^{(b)}(ab)^9)\cdot(a^{-2})\cdot({}^{(ab^{-1})}(a^{-2}))\cdot({}^{(ab^{-1}ab^{-1})}(a^{-2}))\cdot$
$({}^{(ab^{-1}ab^{-1}ab^{-1})}(a^{-2}))\cdot({}^{(ab^{-1}ab^{-1}ab^{-1}ab^{-1})}(a^{-2}))\cdot({}^{(ab^{-1})^4abab)}(a^2))\cdot({}^{(ab^{-1})^4ab^2a)}(a^{-1}b^{-1}ab)^3)\cdot$
$({}^{(ab^{-1})^4ab^2ab^{-1}a^{-1}b)}(a^2))\cdot({}^{(ab^{-1})^4ab^2)}(a^2))\cdot({}^{((ab^{-1})^3aba)}(a^{-1}b^{-2}ab^2)^2)\cdot({}^{((ab^{-1})^3abab^{-2})}(a^{-2}))\cdot$
$({}^{(ab^{-1}ab^{-1}ab^{-1}abab^{-2}ab)}(a^{-2}))\cdot({}^{(ab^{-1}ab^{-1}ab^{-1}abab^{-2}abab)}(a^{-2}))\cdot({}^{((ab^{-1})^3ab)}(a^2))\cdot({}^{((ab^{-1})^3)}(a^2))\cdot$
$({}^{(ab^{-1}ab^{-1})}(a^2))\cdot({}^{((ab^{-1})^2a^{-1}b^{-1})}(a^{-2}))\cdot({}^{((ab^{-1})^2)}(a^{-1}b^{-1}ab)^3)\cdot({}^{((ab^{-1})^2b^{-1}a^{-1}bab^{-1})}(a^{-2}))\cdot$
$({}^{((ab^{-1})^2b^{-1})}(a^{-2}))$ (Rewritten in $\mathcal{A}$)

$word = bab^{-1} \cdot a^2b^{-6}a^{-1} \cdot b^{-1}a^{-2}$

$\Longrightarrow ({}^{(bab^{-1})}(a^2)) \cdot ({}^{(ba)}(b^{-7})) \cdot (a^{-2})$ (Rewritten in $\mathcal{P}$)

$\Longrightarrow ({}^{(bab^{-1})}(a^2)) \cdot ({}^{(ba)}(b^{-7})) \cdot (a^{-2})$ (Rewritten in $\mathcal{A}$)

$word = a^{-1}b^2a^{-1}\cdot b^{-1}ab^2\cdot ab^2a^{-1}\cdot ba^3b\cdot a^{-2}b^{-1}a^{-1}\cdot b^{-1}a^{-1}b^{-2}\cdot a^{-1}b^{-2}a^{-1}\cdot ba^{-1}b^{-2}a^{-1}$

$\Longrightarrow (a^{-2})\cdot({}^{(ab^2)}(a^{-2}))\cdot({}^{(ab^2ab^{-1}ab^2ab^2)}(a^{-2}))\cdot({}^{(ab^2ab^{-1}ab^2ab^2ab)}(a^2))\cdot({}^{(ab^2ab^{-1}ab^2ab^2abab)}(a^{-2}))$

(Rewritten in $\mathcal{P}$)

$\Longrightarrow (a^{-2})\cdot({}^{(ab^2)}(a^{-2}))\cdot({}^{(ab^2ab^{-1}ab^2ab^2)}(a^{-2}))\cdot({}^{(ab^2ab^{-1}ab^2ab^2ab)}(a^2))\cdot({}^{(ab^2ab^{-1}ab^2ab^2abab)}(a^{-2}))$

(Rewritten in $\mathcal{A}$)

$word = b^{-1}a^{-1}b\cdot a^{-1}ba\cdot ba^{-2}b^2\cdot a^{-1}b^{-1}a^{-2}\cdot ba^{-1}b^{-3}\cdot a^{-1}b^{-1}a^{-1}\cdot b^{-1}a^{-1}b$

$\Longrightarrow ({}^{(b^{-1})}(a^{-2}))\cdot({}^{(b^{-1}ab)}(a^{-2}))\cdot({}^{(b^{-1}abababab)}(a^{-2}))\cdot({}^{(b^{-1}abababab^3)}(a^{-2}))\cdot({}^{(b^{-1}abababab^3ab^{-1})}(a^{-2}))$

(Rewritten in $\mathcal{P}$)

$\Longrightarrow ({}^{(b^{-1})}(a^{-2}))\cdot({}^{(b^{-1}ab)}(a^{-2}))\cdot({}^{(b^{-1}abababab)}(a^{-2}))\cdot({}^{(b^{-1}abababab^3)}(a^{-2}))\cdot({}^{(b^{-1}abababab^3ab^{-1})}(a^{-2}))$

(Rewritten in $\mathcal{A}$)

$word = b^{-2}ab^{-1} \cdot a^{-1}b^{-1}a^{-2} \cdot b^{-1}a^{-1}b^{-1} \cdot ab^2a^2 \cdot b^2a^{-2}b^3 \cdot a^{-1}ba^{-1} \cdot b^2a^{-1}b \cdot a^{-1}b^2$

$\Longrightarrow ({}^{(b^{-2}ab^{-1})}(a^{-2})) \cdot ({}^{(b^{-2}ab^{-1}ab^{-1})}(a^{-2})) \cdot ({}^{(b^{-2}ab^{-1}ab^{-2})}(a^{-2})) \cdot ({}^{(b^{-2}ab^{-1}ab^{-2}ab^{-1}ab^2)}(a^2)) \cdot$
$({}^{(b^{-2}ab^{-1}ab^{-2}ab^{-1}a)}(b^7)) \cdot ({}^{(b^{-2}ab^{-1}ab^{-2}ab^{-1}ab^{-3})}(a^{-2}))$ (Rewritten in $\mathcal{P}$)

$\Longrightarrow ({}^{(b^{-2}ab^{-1})}(a^{-2})) \cdot ({}^{(b^{-2}ab^{-1}ab^{-1})}(a^{-2})) \cdot ({}^{(b^{-2}ab^{-1}ab^{-2})}(a^{-2})) \cdot ({}^{(b^{-2}ab^{-1}ab^{-2}ab^{-1}ab^2)}(a^2)) \cdot$
$({}^{(b^{-2}ab^{-1}ab^{-2}ab^{-1}a)}(b^7)) \cdot ({}^{(b^{-2}ab^{-1}ab^{-2}ab^{-1}ab^{-3})}(a^{-2}))$ (Rewritten in $\mathcal{A}$)

$word = a^{-1}b^{-1}a^2 \cdot b^{-2}a^{-1}b \cdot ab^{-1}a^{-2} \cdot b^3ab^{-1} \cdot a^2ba^{-1} \cdot b^{-2}a^{-1}b^{-1} \cdot a^{-1}b^3a^{-1}$

$\Longrightarrow (a^{-2}) \cdot ({}^{(ab^{-1})}(a^2)) \cdot ({}^{(ab^{-3})}(a^{-2})) \cdot ({}^{(ab^{-3}ab)}(a^2)) \cdot ({}^{(ab^{-3})}(a^2)) \cdot ({}^{(ab^{-2})}(b^{-1}a^{-1}ba^{-1})^3) \cdot$
$({}^{(ab^{-2}ab^{-1}ab)}(a^2)) \cdot ({}^{(ab^{-2}ab^{-1})}(a^2)) \cdot ({}^{(ab^{-2})}(a^2)) \cdot ({}^{(ab^{-2})}(a^{-1}b^{-1}a^{-1}b)^3) \cdot ({}^{(ab^{-3}abab^2ab^{-1})}(a^2))$
(Rewritten in $\mathcal{P}$)

$\Longrightarrow (a^{-2}) \cdot ({}^{(ab^{-1})}(a^2)) \cdot ({}^{(ab^{-3})}(a^{-2})) \cdot ({}^{(ab^{-3}ab)}(a^2)) \cdot ({}^{(ab^{-3})}(a^2)) \cdot ({}^{(ab^{-2}b^{-1}a^{-1}b)}(a^{-2})) \cdot$
$({}^{(ab^{-2})}(b^{-1}a^{-1}ba)^3) \cdot ({}^{(ab^{-2}a^{-1}b^{-1}ab)}(a^{-2})) \cdot ({}^{(ab^{-2})}(a^{-2})) \cdot ({}^{(ab^{-2}ab^{-1})}(a^2)) \cdot ({}^{(ab^{-2})}(a^2)) \cdot$
$({}^{(ab^{-2}a^{-1}b^{-1})}(a^{-2})) \cdot ({}^{(ab^{-2})}(a^{-1}b^{-1}ab)^3) \cdot ({}^{(ab^{-3}a^{-1}bab^{-1})}(a^{-2})) \cdot ({}^{(ab^{-3})}(a^{-2})) \cdot$
$({}^{(ab^{-3}abab^2ab^{-1})}(a^2))$ (Rewritten in $\mathcal{A}$)

$word = a^2b^{-1}a^{-5} \cdot ba^2b^{-1} \cdot aba \cdot b^{-1}a^{-2}ba^{-1}$

$\Longrightarrow (a^2) \cdot ({}^{(b^{-1})}(a^{-2})) \cdot ({}^{(b^{-1})}(a^{-2})) \cdot ({}^{(b^{-1})}(a^{-2})) \cdot ({}^{(b^{-1}ab)}(a^2)) \cdot ({}^{(b^{-1})}(a^2)) \cdot ({}^{(ab^{-1})}(a^{-2}))$
(Rewritten in $\mathcal{P}$)

$\Longrightarrow (a^2) \cdot ({}^{(b^{-1})}(a^{-2})) \cdot ({}^{(b^{-1})}(a^{-2})) \cdot ({}^{(b^{-1})}(a^{-2})) \cdot ({}^{(b^{-1}ab)}(a^2)) \cdot ({}^{(b^{-1})}(a^2)) \cdot ({}^{(ab^{-1})}(a^{-2}))$
(Rewritten in $\mathcal{A}$)

Therefore, we can conclude that the algorithm actually works well even on extremely large groups.

Bibliography

Cooperman, G, Finkelstein, L, & Sarawagi, N. (1990). *Applications of cayley graphs.* Proceedings of the 8th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 367-378.

Dehn, M. (1912). *Über unendliche diskontinuierliche gruppen.* Math. Annalen, 71, 116-144.

Dehn, M. (1912). *Transformation der kurven auf zweiseitigen flächen.* Math. Annalen, 72, 413-421.

Dummit, D S, & Foote, R M. (1999). *Abstract algebra.* Danvers, MA: Prentice Hall.

Holt, D F, Eick, B, & O'Brien, E A. (2005). *Handbook of computational group theory.* Chapman and Hall/ CRC.

Kapovich, I, & Myasnikov, A. (2001). *Stallings foldings and subgroups of free groups.* Journal of Algebra, 608-668.

Lyndon, R C. (1966). *On denh's algorithm.* Math. Annalen, 166, 208-228.

Lyndon, R C, & Schupp, P E. (1970). *Combinatorial group theory.* Springer.

Mendelsohn, N S. (1964). *An Algorithmic solution for a word problem in group theory.* Canadian J. Math, 16, 509-16.

Neubüser, J. An elementary introduction to coset table methods in computational group theory. In *Groups - St Andrews* 1981, volume 71 of *London Math. Soc. Lecture Note Ser.*, pages 1-45, Cambridge, 1982. Cambridge University Press.

Robinson, D J S. (1996). *A Course in the theory of Groups.* New York: Springer-Verlag.

Sims, C C. (1994). *Computation with finitely presented groups.* Cambridge University Press.