A hybrid quotient algorithm

Alexander Hulpke and Alice C. Niemeyer

Draft of November 2, 2005

Abstract

We describe an algorithm which takes as input a finitely presented group \( F \), a finite confluent rewriting system for a quotient \( G \) and a prime \( p \), and returns as output a finite confluent rewriting system for the largest extension of an elementary abelian group in characteristic \( p \) by \( G \) which is still an epimorphic image of \( F \).

1 Introduction

Finitely presented groups can present a challenge for computational methods in group theory. For example, in many finitely presented groups it is not possible to determine whether two given words represent the same group element, see \([\text{Nov55}]\). However, some classes of finitely presented groups can be represented in such a way that the word problem is soluble. For example, polycyclic groups admit a particular kind of presentation in which the word problem can be solved. A natural question is thus, given a finitely presented group of which we know (or suspect) that it has a “computationally nice” presentation, can we find this presentation?

Previously, this question has been addressed for several types of polycyclic groups. Examples of algorithms which determine polycyclic presentations for certain polycyclic quotients of finitely presented groups include the methods for determining finite quotients of prime power order, see \([\text{HN80}]\), nilpotent quotients, see \([\text{Sim94, Sect. 11.5}]\), \([\text{Nic96}]\), or other polycyclic quotients, see \([\text{Bru99, ENP, Lee84, Lo98, Lo98b, Nie94, Nie95, Ple87, Wam77, Weg92}]\).

Here we consider a more general scenario where we are given a finite confluent rewriting system for a certain quotient of a finitely presented group and we wish to compute a rewriting system for larger quotient of the given finitely presented group, namely an extension of a module by the given quotient. The algorithm presented here is a generalization of the methods described in \([\text{ENP}]\) and \([\text{Nie95}]\).

Our algorithm repeats a basic step. Given a finite confluent rewriting system for a quotient \( G \) of a finitely presented group \( F \), the algorithm first computes a presentation for a module \( M \), such that the extension of \( M \) by \( G \) is isomorphic to a quotient of \( F \). An algorithm by Linton called vector enumerator \([\text{Lin93}]\) is employed to compute for the given \( G \)-module a vector space generating set and the action of the generators of \( G \) in the confluent rewriting
system on the vector space. We use the output of the vector enumerator to compute a
confluent rewriting system for the extension of $G$ by $M$ and obtain in this fashion a finite
confluent rewriting system for a larger quotient of our given finitely presented group.

We will start by describing some conventions we use. Section 2 then describes the actual
algorithm together with its justification in the form of a single step. Section 3 deals with the
initial step and describes how the algorithm would get started. Section 4 gives a completely
worked-out example. Section 5 finally describes a very basic implementation.

1.1 Conventions:

The necessary background on finite confluent rewriting systems can be found in [Sim94].
Here we only mention enough to establish our notation.

We use bold letters to indicate a collection of objects of one type.

If $\mathbf{x} = \{x_1, \ldots, x_n\}$ is an alphabet, we denote the set of words in this alphabet by $\mathbf{x}^*$. We use fraktur letters to denote words and write $\mathbf{w}(\mathbf{x})$ to denote a word in $\mathbf{x}_1$. We suppose further that $\mathbf{x}^*$ is endowed with a reduction order; for example the length-plus-lexicographic
order, in which words of equal length are ordered lexicographically and shorter words proceed
longer ones.

If $\mathcal{R}$ is a rewriting system on $\mathbf{x}^*$ and $u, v \in \mathbf{x}^*$, we write $u \stackrel{\mathcal{R}}{\longrightarrow} v$, if $v$ can be obtained
from $u$ by applying a sequence of rules from $\mathcal{R}$, and $u \stackrel{\mathcal{R}^*}{\longrightarrow} v$ if $u \stackrel{\mathcal{R}}{\longrightarrow} v$ and there is no rule in
$\mathcal{R}$ that can be applied to $v$. If the rewriting system is confluent such an element $v$ is
uniquely determined by $u$ and is called $\text{reduced}$. There is an algorithm, the \text{Knuth Bendix Procedure for Strings},
which computes for a word $u$ in the finite confluent rewriting system $\mathcal{R}$ a reduced word $v$
such that $u \stackrel{\mathcal{R}^*}{\longrightarrow} v$ (see [Sim94, Proposition 5.1]).

If $\mathbf{x}$ and $\mathbf{y}$ are alphabets, the (automatically disjoint) union $\mathbf{x} \cup \mathbf{y}$ is an alphabet. An
element in $(\mathbf{x} \cup \mathbf{y})^*$ can be written in the (unique) form $\mathbf{w}_0(\mathbf{x})y_1\mathbf{w}_1(\mathbf{x})y_2\mathbf{w}_2(\mathbf{x})\cdots y_i\mathbf{w}_i(\mathbf{x})$ with $y_1 \in \mathbf{y}$ and $\mathbf{w}_i(\mathbf{x}) \in \mathbf{x}^*$. If $\prec_1$ is an order on $\mathbf{x}^*$ and $\prec_2$ is an order on $\mathbf{y}^*$ the
wreath product order $\prec_1 \wr \prec_2$ is defined as follows. Let $v = v_0y_1v_1y_2v_2\cdots y_nv_n$ and $w = w_0z_1w_1z_2w_2\cdots z_nw_n$ be words in $(\mathbf{x} \cup \mathbf{y})^*$ with $\mathbf{w}_i, v_i \in \mathbf{x}^*$ and $y_i, z_i \in \mathbf{y}$. Then $v(\prec_1 \wr \prec_2 )w$ if either $y_1y_2\cdots y_n \prec_2 z_1z_2\cdots z_n$ or $y_1y_2\cdots y_n = z_1z_2\cdots z_n$ and $v_0v_1\cdots v_n$ precedes
$w_0w_1\cdots w_n$ with respect to the lexicographic ordering on $(\mathbf{x}^*)^r+1$ induced by $\prec_1$.

Let $G$ be a group, $\mathbf{g}$ a generating sequence for $G$, and $\mathbf{x}$ an alphabet of the same cardinality
and $\mathcal{R} = \{\mathbf{w}_{i,1}(\mathbf{x}) \rightarrow \mathbf{w}_{i,2}(\mathbf{x}) \mid 1 \leq i \leq r\}$ a finite confluent rewriting system. Then we can
evaluate a word $\mathbf{w}(\mathbf{x})$ in the group generators $\mathbf{g}$ by replacing each $x_i$ in $\mathbf{w}(\mathbf{x})$ by $g_i$. This way
we obtain an element of $G$ denoted by $\mathbf{w}(\mathbf{g})$. We say that the finite confluent rewriting system $\mathcal{R}$ describes $G$ in terms of $\mathbf{g}$ if the reduced words in $\mathbf{x}^*$ are in one-to-one correspondence
to the elements of $G$ via this evaluation. In particular, as $\mathcal{R}$ is finite and confluent, every
element of $G$ can be identified uniquely with a word $\mathbf{w}(\mathbf{x})$ and thus also with the word $\mathbf{w}(\mathbf{g})$. 
2 A rewriting system for an extension

2.1 Setup

Let $F$ be a finitely presented group of rank $\ell$, $G$ a group with a finite confluent rewriting system (e.g. a finite group), and $\varphi: F \rightarrow G$ an epimorphism. We say that $\psi$ is a lifting of $\varphi$ if there is a group $H$ and epimorphism $\nu: H \rightarrow G$ such that $\psi: F \rightarrow H$ is an epimorphism and $\varphi = \psi \nu$. A lifting $\psi$ of $\varphi$ is a module lifting if $M = \ker \nu$ is an $(H/M)$-module. For simplicity we identify $H/M$ with $G$ and say that $M$ is a $G$-module.

Our algorithm computes a confluent rewriting system for the largest module lifting of $\varphi$ in the sense that any other module lifting $\hat{\varphi}$ satisfies that $\hat{\varphi}(F)$ is an epimorphic image of $\varphi(F)$.

Let $\mathfrak{g} = \{g_1, \ldots, g_n\}$ be a generating set for $G$ and $\mathfrak{x} = \{x_1, \ldots, x_n\}$ an alphabet of the same cardinality. Let $\mathcal{R} = \{w_{i,1}(\mathfrak{x}) \rightarrow w_{i,2}(\mathfrak{x}) \mid 1 \leq i \leq r\}$ be a finite confluent rewriting system for $G$ in terms of $\mathfrak{g}$ with respect to a reduction order $\prec$. Then we can express the images under $\varphi$ of the free generators in $F$ uniquely as words in $\mathfrak{g}$. Let $f_i^\varphi = b_i(\mathfrak{g})$ for $1 \leq i \leq \ell$.

The group $H$ is generated by $M$ together with the representatives $\mathfrak{h} = \{h_1, \ldots, h_n\}$ for the generators $\mathfrak{g}$ of the factor group $G$, where $\nu(h_i) = g_i$. As $\mathfrak{h}$ also has the same cardinality as $\mathfrak{x}$, we can evaluate the words $w(\mathfrak{x})$ in $\mathfrak{h}$ and obtain elements of $H$. As a quotient of $F$, the group $H$ is generated as by the images of the generators of $F$ under $\psi$. The image of the free generator $f_i$ of $F$ under $\psi$ differs from $b_i(\mathfrak{h})$ by an element $m_i$ of $M$, that is $f_i^\psi = b_i(\mathfrak{h})m_i$. We call $m_i$ the tail of the image $f_i^\psi$. Let $\{m_1, \ldots, m_\ell\}$ be the set of tails (so the generator images are $B = \{b_i(\mathfrak{h})m_i \mid i\}$).

Suppose $g_ig_j = g_k$ in $G$. According to extension theory [Hup67, Satz I.14.1] the product of the corresponding representatives $h_i$, $h_j$ in $H$ differs from the representative $h_k$ of the product (in the factor group) by an element of $M$ called a cofactor. Thus in $H$, both sides of a rewriting rule do not necessarily evaluate to the same element of $H$, but we have that

\[ w_{i,1}(\mathfrak{h}) = w_{i,2}(\mathfrak{h})c_i, \]

for some cofactors ($c_i \in M$). To simplify notation later on we set $m_{\ell+i} := c_i$ for $1 \leq i \leq r$.

A generating set for $H$ is:

- The representatives $\mathfrak{h}$ (which yield $n$ generators);
- The tails $m_1, \ldots, m_\ell$ and the cofactors $m_{\ell+1}, \ldots, m_{\ell+r}$;
- Conjugates of the $m_i$ in $H$. (As $M$ itself is abelian, the conjugation needs to be done only with products of the $h_i$, so there is one conjugate for each element of $G$. If $G$ is finite, this gives $n + (\ell + r) \times |G|$ generators in total).

We have to distinguish between elements of $F$, $G$, $H$ and $M$, and their images, or chosen preimages under $\varphi$, $\nu$ and $\psi$ as well as their representations as elements in various rewriting systems. For ease of notation we introduce some conventions which we summarize in Tables 2.1 and 2.1.
2.2 A new rewriting system

Our next goal is to define a rewriting system $S$ of $H$ which "exhibits" $R$. The idea is that $S$ should contain some generators $\gamma$ which correspond to the representatives $h$ and some, $\omega$, which correspond to a generating set for $M$. The reduction order for $S$ should be derived from the reduction order on $R$. Further the rules in $S$ should contain a sub-collection of rules which reduce to the rules in $R$ if $y_i$ is replaced by $x_i$ and the elements of $\omega$ are deleted.

To represent the generators of $H$, we choose an alphabet that contains letters $\gamma = \{y_1, \ldots, y_n\}$ (that stand for the representatives $h_i$) and letters $\omega = \{\omega_1, \ldots, \omega_\ell\}$ as $H$-module

Cofactors $\omega_{i,1}(h) = \omega_{i,2}(h)\omega_{i+1}$

Conjugates by reduced words

Table 1: Group elements and corresponding alphabet letters

<table>
<thead>
<tr>
<th>Group</th>
<th>group elements</th>
<th>alphabet letters</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>$f_i$</td>
<td>$x_i$</td>
<td>$F = \langle f_1, \ldots, f_\ell \rangle$</td>
</tr>
<tr>
<td>$G$</td>
<td>$g_i$</td>
<td>$x_i$</td>
<td>$G = \langle g_1, \ldots, g_n \rangle$</td>
</tr>
<tr>
<td>$H$</td>
<td>$h_i$</td>
<td>$y_i$</td>
<td>$H = \langle M, h_1, \ldots, h_n \rangle$</td>
</tr>
<tr>
<td>$M$</td>
<td>$m_i$</td>
<td></td>
<td>$M = \langle m_1, \ldots, m_{\ell+r} \rangle$ as $H$-module</td>
</tr>
<tr>
<td>$m_{i+1}$</td>
<td></td>
<td>$z_{i,m}$</td>
<td>Conjugates by reduced words</td>
</tr>
<tr>
<td>$m_i^{w(h)}$</td>
<td></td>
<td>$z_{j,m}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Epimorphisms

$\phi: F \rightarrow G; f_i^\phi = b_i(g)$

$\psi: F \rightarrow H; f_i^\psi = b_i(h)m_i$

$\nu: H \rightarrow G; h_i^\nu = g_i$

2.2 A new rewriting system

The extension rules $T_0$:

The first collection of rules, $T_0$, describes $H$ as an extension of a $G$-module $M$ by $G$:
Each rule \( w_{i,1}(x) \rightarrow w_{i,2}(x) \) in \( \mathcal{R} \) gives rise to a corresponding rule
\[
\begin{align*}
  w_{i,1}(y) & \rightarrow w_{i,2}(y)z_{i+1,1}
\end{align*}
\] (2)
in \( \mathcal{T}_0 \), reflecting the introduction of cofactors in Equation (1),
(The elements \( z_{i,\mathfrak{w}} \) for nontrivial \( \mathfrak{w} \) will represent conjugates.)

Let \( p \) be a prime and the characteristic of the \( G \)-module \( M \). The following rules in \( \mathcal{T}_0 \) certainly all represent the identity in \( H \) and describe \( M \) as a \( G \)-module.

\[
\begin{align*}
  z_{j,0}z_{i,\mathfrak{w}} & \rightarrow z_{i,\mathfrak{w}}z_{j,0} & \text{if } z_{i,\mathfrak{w}} < z_{j,0}, \quad (3)
  (z_{i,\mathfrak{w}})^p & \rightarrow 1, \quad (4)
  z_{i,\mathfrak{p}} \cdot w(y) & \rightarrow w(y)z_{i,u} & \text{if } w(x) \text{ is reduced in } \mathcal{R} \text{ and } v(x) \cdot w(x) \xrightarrow{\mathcal{R}} u(x). \quad (5)
\end{align*}
\]

(In practice we do not list these rules explicitly but incorporate them in the rewriting process.)

Note that by the definition of the wreath product ordering \( \prec_Z \preceq Y \), the right hand sides of the rules (5) are indeed reduced words. (This is essentially the reason for choosing such an ordering.)

**Remark 6.** We work here in finite characteristic \( p \). By permitting negative exponents it would be possible to work in characteristic 0, leaving out rules (4). In this case we would find the largest \( \mathbb{Z} \)-module below \( G \).

Using these rules, we can rewrite a word in \( (y \cup z)^* \) as a product of a word \( a(y) \) (the \( y \)-part) and a word \( b(z) \) (the \( z \)-part). Furthermore, \( a(x) \) is reduced with respect to \( \mathcal{R} \), and \( b(z) \) is reduced in the obvious way.

**Lemma 7.** The monoid \( A \) generated by \( y \cup z \) and defined by the relations (2) – (5) is a group.

**Proof.** It is sufficient to show that every element of \( A \) has at least one inverse.

Let \( a \in A \), represented by the word \( w(y)v(z) \). As \( G \) is a group, there exists a word \( u(y) \), such that \( w(g) \cdot u(g) \) is the identity in \( G \). Thus the product \( w(y)v(z) \cdot u(y) \) must reduce to an element with trivial \( y \)-part. On the other hand, every element represented by a word in \( z^* \) clearly has an inverse because of rules (4). Multiplying \( u(y) \) with a word for this inverse thus yields a right inverse \( r \) for \( a \).

By a similar argument, \( a \) must have a left inverse \( l \). The calculation \( r = lr = (la)r = l(ar) = ll = l \) then shows that both are the same. \( \square \)

It is easily seen by a similar argument that also all factor monoids of \( A \) are groups.

While theory shows the equality of left- and right-inverse, a concrete calculation has to take some care, as the rewriting process might produce different words (with the same \( y \)-part) for both. We therefore set both \( z \)-parts equal and add these relations to \( \mathcal{T}_0 \).
The confluence rules $T_1$: 

By Lemma 7 we obtain a rewriting system with generators $y \cup z$ and rules $T_0$ for a quotient group of $G$. However, we want to ensure that we obtain a confluent rewriting system. Hence we introduce additional rules $T_1$ to our rewriting system.

We can achieve confluence by testing for local confluence. By [Sim94, Proposition 3.1] it is sufficient to consider overlaps of left sides of rules from the rewriting system.

**Lemma 8.** The only overlaps we need to test are overlaps of rules of type (2) with rules type (2).

**Proof.** It is easily seen that an overlap of rules of type (3) or (4) with rules of type (3) or (4) always reduce in both ways to the same reduced word.

An overlap of a rule of type (3) with a rule of type (5) is of the form

\[ z_{j,i}w(y)z_{i,w} \rightarrow w(y)z_{j,i,w}z_{i,w} \]

and the reduced words are equal. A similar argument holds for overlaps of rules of type (4) with rules of type (5). Rules of type (5) cannot overlap with rules of type (5), as $y$-part and $z$-part collide. Rules of type (2) cannot overlap with rules of type (3) or (4). Neither can they overlap with rules of type (5) because if $y$ occurs in (5) then $x$ is reduced in $R$, i.e. cannot admit any reduction with rules obtained from $R$.

Thus the only situation, where local confluence is not automatically guaranteed is for words of the form $uvw$ with nontrivial $v$ for which there are two different rules, namely $uv \rightarrow x$ and $uv \rightarrow y$, both of type (2).

We therefore reduce both products $xw$ and $uy$ with the rules in $T_0$. As $R$ is confluent, the $y$-parts of the resulting words must agree, so confluence imposes an equality condition on the $z$-parts of these words. We collect all these equality conditions in a list $T_1$.

We therefore know that the rules $T_0 \cup T_1$ yield a confluent rewriting system. The group given by it has a quotient isomorphic to $G$, obtained by mapping $z_{i,w} \mapsto 1$.

**The quotient rules $T_2$:**

Next, we want to ensure that the monoid represented by this rewriting system is a quotient of $F$. Thus we have to consider the images under $\psi$ of the generators of $F$. As $f_i^b = b_i(h)m_i$ in $H$ for $1 \leq i \leq \ell$, we use generators $z_{i,x}$ for $1 \leq i \leq \ell$ and represent $f_i^b$ by $b_i(y)z_{i,x}$. Let $\tilde{B} := \{b_i(y)z_{i,x} \mid 1 \leq i \leq \ell\}$.

For each defining relation $w(f)$ of $F$, we evaluate $w(\tilde{B})$ as a word in $(y \cup z)^*$, and reduce it using the rules in $T_0$. As the images in $G$ fulfill the relations, the $y$-part of the reduced words must vanish yielding a reduced word in $z^*$, which must be trivial in $H$. We collect these reduced forms in a list $T_2$. 

6
Remark 9. The relators for \( F \) or the words \( b_i \) may involve inverses. In this case we use the method of the proof of lemma 7 to determine words for inverses.

If we take the relations in \( T_0 \cup T_1 \cup T_2 \), they describe a group that is a quotient of \( F \).

The uniqueness rules \( T_3 \):

There is however one bit of ambiguity left: If we would change the representatives \( h_i \) by elements of \( M \), the resulting changed generating set is also compatible with \( T_0 \cup T_1 \cup T_2 \). We therefore prescribe which representatives to chose: In \( G \), each of the generators \( g_i \) can be written as a word in the images of the free generators: \( g_i = u_i(b_1(g), \ldots, b_n(g)) \).

We enforce that the generators for \( H \) behave in the same way by putting \( h_i = u_i(B) \). In the presentation, this yields the relator \( u_i(B)/y_i \). Again, we rewrite these relators using \( T_0 \). Again the \( y \)-parts vanish as the relations hold in the factor group \( G \), and we obtain another set \( T_3 \) of words in \( z^* \).

Remark 10. This condition can also be viewed as ensuring that the map \( \psi : F \to H \) is surjective: Clearly they ensure that the generators \( h_1, \ldots, h_l \) are in the image of \( \psi \). Then \( f_i^v \) is represented by \( b_i(y)z_i \) for \( i \leq \ell \), we have that the elements represented by \( z_{i,1} \) are in the image. Finally the extended rewriting rules show that the also the elements represented by \( z_{i,1} (i > \ell) \) are contained in the image.

Remark 11. Note that we also face a similar problem in other quotient algorithms. For example, in the soluble quotient algorithm \([ENP]\) the problem is overcome by using labeled quotient systems.

The group \( \tilde{H} \) presented by \( T_0 \cup T_1 \cup T_2 \cup T_3 \) now is a quotient of \( F \), it has an epimorphism \( \tilde{\nu} : H \to G \), and the kernel of \( \tilde{\nu} \) is an abelian group. On the other hand, every group fulfilling these conditions has to be a quotient of \( \tilde{H} \). Thus \( T_0 \cup T_1 \cup T_2 \cup T_3 \) is a presentation for \( H \).

### 2.3 A Module Presentation

Let us now consider the set of “new” rules \( T_1 \cup T_2 \cup T_3 \). Suppose we have a \( G \)-module \( N \) together with elements \( n_i \in N \) such that, substituting \( n_i^{w(g)} \) for \( z_{i,1} \) the rules \( T_1 \cup T_2 \cup T_3 \) are fulfilled.

Then, using the rules from \( T_0 \) to describe the 2-cohomology, we can form an extension of \( N \) by \( G \) such that this extension is a quotient of \( F \) and has a quotient \( G \).

This means that the \( G \)-module defined by \( T_1 \cup T_2 \cup T_3 \) is the largest module fulfilling these properties and thus is isomorphic to \( M \).

We now write the rules in the \( z_{i,1} \) additively in the module generators \( \{ z_{i,1} | 1 \leq i \leq l+r \} \) and their images under words representing elements of \( G \).

For example a relation \( z_3.a \cdot z_5 \cdot z_7 \cdot w \to z_2.a \) would become \( z_3^u + 5w + 2z_7^w - z_2^a \).

This way we can consider \( T_1 \cup T_2 \cup T_3 \) (together with a presentation of the acting group \( G \)) as a module presentation for \( M \).
We now use the *vector enumerator* algorithm by Linton [Lin93] to compute a vector space basis for $M$. In our context the vector enumerator takes the module presentation as input and terminates if and only if $M$ is finitely generated, whence the output is:

1. a vector space basis $v = \{v_1, \ldots, v_d\}$ for $M$,
2. a matrix describing the action of each $g \in G$ on $M$ with respect to $v$,
3. an expression in $v$ for the elements represented by $\{z_{i,1} \mid 1 \leq i \leq \ell + r\}$.

### 2.4 A rewriting system for $H$

Finally we want to obtain a confluent rewriting system for the extension $H$ of $M$ by $G$. We thus can compute the following words:

- For $z \in z$ denote by $bc(z)$ a word in $v^*$, representing the same element of $M$.
- For a word $w \in x^*$ and $v_i \in v$ we denote by $bc(v_i^w|g)$ a word in $v^*$ representing the image $m_i^w$ given by the action of $G$ on $M$.

We now create a new rewriting system, $S$, in $y \cup v^*$.

\[
\begin{align*}
  w_{i,1}(y) & \rightarrow w_{i,2}(y)bc(z_{i+1,1}) \\
  v_i v_j & \rightarrow v_j v_i \quad \text{if } i > j, \\
  v_i \cdot w(y) & \rightarrow w(y)bc(v_i^w|g) \quad \text{if } w \text{ is reduced with respect to } R \\
  v_i^{p_i} & \rightarrow 1 \quad \text{(orders of the generators)}
\end{align*}
\]

This system describes an extension of $M$ by $G$, as its rules are fulfilled by the elements $h_i$ and $m_j$ of $H$ it describes $H$.

As the relations in $T_1$ were incorporated in the choice of the module basis, it is confluent for the wreath product order $\prec_V \wr \prec_V$, where $\prec_V$ is the lexicographic order on $v^*$.

As we also know images for the free generators and a decomposition of the $h_i$ as words in these images, we have the necessary input information to iterate the process.

### 3 Initial Quotient

As input to the algorithm, we need an epimorphism $\varphi : F \rightarrow G$ onto an initial quotient and a finite confluent rewriting system for $G$. Choosing $G = \langle 1 \rangle$, our algorithm would compute a soluble quotient of $F$, for which already specialized algorithms are available.

In some situations (for example for the quotients of finitely presented groups studied in [BIP00]), such a quotient $G$ is given by the setup of the problem. More commonly, one could use a low-index algorithm [Sim94, Section 5.4] to obtain a quotient onto a permutation group $G$ (the action on the cosets of the subgroup found) of small degree.

We also need a finite confluent rewriting system for this quotient $G$. Clearly it is possible to take a presentation for $G$ (obtained for example from the permutation representation
and use Knuth-Bendix completion. In general, however this yields rather complicated rewriting systems. Instead it is preferable to proceed analogous to [BGK+97] and use the composition structure of \(G\). For this we can proceed essentially as for rules (12)-(15) above:

Suppose that \(N \triangleleft G\) and that we have a confluent rewriting system for \(G/N\) and one for \(N\). By modifying the rules of \(G/N\) to incorporate cofactors from \(N\) (analogous (2)) and by adding commutator rules for generators of generators of \(G/N\) and generators of \(N\) (analogous (5)) we obtain a rewriting system for \(G\) with respect to a wreath product ordering. As the rules obtained from \(G/N\) bring elements modulo \(N\) into a reduced form, and the rules for \(N\) bring the \(N\)-part into reduced form, the obtained rewriting system is confluent. (We needed to enforce completion as for the rules \(T_0\) above as the rules on the generators \(z_{i,m}\) are in fact not confluent – The rules among elements of the factor group impose rules among these generators.)

4 An Example

As an example we consider the finitely presented group

\[
F := \langle a, b, c \mid a^2 = b^2 = c^2 = (ac)^2 = (ab)^3 = (bc)^3 = 1 \rangle
\]

(which is isomorphic \(S_4\)), together with the quotient \(G = S_3\) and epimorphism \(\varphi: F \to G\), \(a \mapsto (1,3), b \mapsto (1,2), c \mapsto (1,3)\) an epimorphism. We chose the generating set \(g_1 = (1,2)\) and \(g_2 = (2,3)\) for \(G\). In these generators, a confluent rewriting system for the length-plus-lexicographic order is given by the rules:

\[
R = \{ x_1^2 \to 1, \quad x_2^2 \to 1, \quad x_2x_1x_2 \to x_1x_2x_1 \},
\]

and we have the images

\[
a \mapsto g_1g_2g_1, \quad b \mapsto g_1, \quad c \mapsto g_1g_2g_1.
\]

We now append tails to all the rules in \(R\) to obtain the rules of form (2) of the rewriting system \(T_0\):

\[
x_1^2 \to z_1, \quad x_2^2 \to z_2, \quad x_2x_1x_2 \to x_1x_2x_1z_3.
\]

We consider all possible overlaps of rules, and obtain the following relations in \(T_1\):

<table>
<thead>
<tr>
<th>Overlap A</th>
<th>Overlap B</th>
<th>Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1^2x_1 \to z_1x_1 = x_1z_{1,x_1})</td>
<td>(x_1x_1^2 \to x_1z_1)</td>
<td>(z_1 = z_{1,x_1})</td>
</tr>
<tr>
<td>(x_2^2x_2 \to z_2x_2 = x_2z_{2,x_2})</td>
<td>(x_2x_2^2 \to x_2z_2)</td>
<td>(z_2 = z_{2,x_2})</td>
</tr>
<tr>
<td>(x_2x_1x_2 \to x_2x_1z_{1,x_1}z_3z_{3,x_2} = x_2(x_2x_1x_2) \to x_1x_2z_1z_3z_{3,x_1})</td>
<td>(z_{2,x_1,x_2} = z_1z_3z_{3,x_1})</td>
<td></td>
</tr>
<tr>
<td>(x_2x_1x_2 \to x_2x_1z_{1,x_1}z_3z_{3,x_2} = z_2)</td>
<td>(z_{1,x_2,x_1}z_3z_{3,x_2} = z_2)</td>
<td></td>
</tr>
<tr>
<td>(x_2x_1x_2 \to x_1z_{1,x_1}z_2z_{3,x_2} = x_2x_1(x_2x_1x_2) \to x_1z_1z_2z_{2,x_1}z_3)</td>
<td>(z_{1,x_2}z_2z_{3,x_1} = z_{1,x_2}z_{2,x_1}z_3)</td>
<td></td>
</tr>
</tbody>
</table>
We write these relations in the form of module relators, so for example the relation \( z_{2x_1x_2} = z_1z_3z_{3x_1} \) becomes \( z_1z_2^{-x_1x_2}z_3^{x_1} \).

Next we define new images \( \bar{B} \):

\[
\begin{align*}
\bar{a} & \mapsto x_1x_2x_1z_4, \\
\bar{b} & \mapsto x_1z_5, \\
\bar{c} & \mapsto x_1x_2x_1z_6
\end{align*}
\]

and evaluate the relators of \( F \) to obtain \( T_2 \):

<table>
<thead>
<tr>
<th>Relator</th>
<th>Value</th>
<th>Algebra Relator</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a^2 )</td>
<td>( x_1x_2x_1z_4x_1x_2x_1z_4 )</td>
<td>( z_1^{1+x_1x_2}z_2^{x_1}z_4^{1+x_1x_2} )</td>
</tr>
<tr>
<td>( b^2 )</td>
<td>( x_1z_5x_1z_5 )</td>
<td>( z_1z_5^{1+x_1} )</td>
</tr>
<tr>
<td>( c^2 )</td>
<td>( x_1x_2x_1z_6x_1x_2x_1z_6 )</td>
<td>( z_1^{1+x_1x_1}z_2^{x_1}z_6^{1+x_1x_1} )</td>
</tr>
<tr>
<td>((ac)^2)</td>
<td>( (x_1x_2x_1z_4x_1x_2x_1z_6)^2 )</td>
<td>( z_1^{1+x_1+2x_2x_1}z_2^{x_1}z_4^{2x_1x_2x_1}z_6^{x_1} )</td>
</tr>
<tr>
<td>((ab)^3)</td>
<td>( (x_1x_2x_1z_4z_1z_5)^3 )</td>
<td>( z_1^{2+x_2+x_1x_2+x_2}x_1z_2z_3z_4^{x_1}+x_1x_2x_1z_5^{1+x_1x_2+x_2x_1} )</td>
</tr>
<tr>
<td>((bc)^3)</td>
<td>( (x_1z_5x_1x_2x_1z_6)^3 )</td>
<td>( z_1^{2+x_2+x_1x_2+x_2}x_1z_2z_3z_4^{x_1}+x_1x_2x_1z_5^{1+x_1x_2+x_2x_1} )</td>
</tr>
</tbody>
</table>

Finally, we decompose the generators of \( G \) as words in the images:

\[
\begin{align*}
g_1 &= b\varphi, \\
g_2 &= (a \cdot b \cdot a)\varphi
\end{align*}
\]

and use these relations as definitions for the \( x_i \). Reduction yields the relators \( T_3 \):

Equation | Relator
---|---
\( x_1 := x_1z_5 \) | \( z_5 \)
\( x_2 := x_1x_2x_1z_4x_1x_2x_1z_4 \) | \( z_1^{1+x_2+x_1x_2}x_1z_2z_3z_4^{x_1}z_5^{1+x_1x_2}z_5^{x_1x_2x_1} \)

We now feed the presentation for \( G \) and the relations in \( T_1 \cup T_2 \cup T_3 \) between the conjugates of the \( z_i \) as a GF(2)-module presentation in the vector enumerator [Lin91].

It returns that the module is two-dimensional, that basis coefficients for the \( z_i \) are given by

\[
\begin{align*}
z_1 &= (0,0), & z_2 &= (0,0), & z_3 &= (0,0), & z_4 &= (0,0), & z_5 &= (0,0), & z_6 &= (1,0),
\end{align*}
\]

and that matrices for the action of \( x_1 \) and \( x_2 \) on this basis are given by

\[
\begin{align*}
x_1 & \mapsto \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, & x_2 & \mapsto \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.
\end{align*}
\]

We therefore know that the largest elementary abelian 2-quotient of \( \ker \varphi \) is of order \( 2^2 \), that the corresponding quotient group of \( F \) is given by the (rewriting system) presentation

\[
\langle x_1, x_2, x_3, v_1, v_2 \mid x_1^2 \rightarrow 1, \quad x_2^2 \rightarrow 1 \quad x_2x_1x_2 \rightarrow x_1x_2x_1, \quad v_1^2 \rightarrow 1, \quad v_2^2 \rightarrow 1, \quad v_2v_1 \rightarrow v_1v_2, \quad v_1x_1 \rightarrow x_1v_2, \quad v_1x_2 \rightarrow x_2v_1v_2, \quad v_2x_1 \rightarrow x_1v_1, \quad v_2x_2 \rightarrow x_2v_2 \rangle,
\]

and that the corresponding lift of \( \varphi \) is given by

\[
\begin{align*}
\bar{a} & \mapsto x_1x_2x_1, \quad \bar{b} \mapsto x_1, \quad \bar{c} \mapsto x_1x_2x_1v_1.
\end{align*}
\]
5 Implementation

We have implemented a basic version of the described algorithm in the system GAP [GAP04]. Table 3 shows the results of some example runs. Group $G_1$ was defined by H. Heineken, $G_2$ and $G_3$ were pointed out by G. Rosenberger. $G_4$ is due to A. Cavicchioli. $G_6$ is obtained by simplifying the presentation of the Fibonacci group $F_8$.

Clearly all of these are small examples for which results could have been obtained with other methods. One obstacle for working with larger examples is that we currently lack implementations for good rewriting systems of simple groups that are not alternating. In these cases a rewriting system is simply obtained from a presentation.

Nevertheless an analysis of the runtime of the examples shows (unsurprisingly, as it is at the heart of the algorithm) that the effective run time is completely dominated by the module enumeration.

On the other hand the choice of prime does not seem to have major influence on the calculation (however it has impact on the amount of memory used).

We also noticed in the calculations that the module relations produced are in general rather sparse. It might be worth to use a Tietze-transformation-like process to reduce the number of module generators, and to shorten relations. The feasibility of such an approach remains to be tested.

6 Closing Remarks

An obvious question is about the performance of the algorithm and its dependence on the choice of the initial rewriting system for $G$. We have not studied this so far. One problem considering complexity considerations is the fundamental insolubility of the word problem [Nov55].

In practice, as mentioned above, at the moment the principal bottleneck is the vector enumerator. This is partly due to the fact that an existing implementation in C is problematic to compile on current machines. Also the implementation is not adapted to the fact that we do not only have a presentation for the acting group $G$, but in fact a confluent rewriting system. Certainly this offers possibilities for substantial improvement.

So far we also have not investigated the question of how to chose primes for the characteristic of the $G$-module $M$. By using a $\mathbb{Z}$-module, instead of a $\mathbb{Z}_p$-module for a fixed prime $p$, it would be possible to treat all primes simultaneously. On the other hand a generalization of the criterion used in newer versions of the SQ [Bru99].

A final question is on what to do with the result of the computation. In the same way that the family of polycyclic quotient algorithms give rise to polycyclic presentations [LNS84], one can consider the rewriting system computed by our algorithm as a particular kind of rewriting presentation. As the resulting group has a large radical, many of the recent algorithms for permutation groups become available. This opens an area of further work.
Group | Image of \( \varphi \) | Inv. | Prime | Module | Time (seconds)
--- | --- | --- | --- | --- | ---
\( G_1 = \langle x, y, z \mid [x, [y, z]] = z, [y, [y, z]] = x, [z, [z, x]] = y \rangle \) | \( A_5 \) | 2 | 2⁵ | 4
\( G_2 = \langle x, y, z \mid x^2 = y^5 = z^3 \) = (xy)^3 = (xz)^3 = (y^2z)^2 = 1 \rangle | \( A_5 \) | 2 \cdot 4^4 | 2 | 2⁵ | 1
\( G_3 = \langle x, y, z \mid x^2 = y^5 = z^3 \) = (xy)^3 = (xz)^2 = (y^2z)^3 = 1 \rangle | \( A_6 \) | 0^162^4 | 2 | 2⁰ | 959
\( G_4 = \langle a, b \mid aba^{-2}bab^{-1} \) = (b⁻¹a³b⁻¹a⁻³)²a = 1 \rangle | \( L_2(11) \) | 0^52^2 \cdot 5 | 2 | 2^5³ | 96367
\( G_5 = \langle a, b \mid a^2 = b^3 \) = (ab)^7 = 1 \rangle | \( L_2(7) \) | 0^6 | 2 | 2⁶ | 375
\( G_6 = \langle a, b \mid ba^{-2}ba^{-1}b^2ab^2a^{-1} \) \( L_2(7) \) | 2 \cdot 3^20\cdot 7^39^2 | 2 | 2¹ | 1057
\( = abab^2aba^2b^{-1}a = 1 \rangle \) | 3 | 3^2² | 690
\( = 7 \cdot 7^3 \) | 780
\( G_7 = \langle a, b, c \mid ac^{-1}bc^{-1}aba^{-1}b \) \( = abab^{-1}c^2b^{-1} \) SL₂(5) | 3^125^5 \cdot 11^4 | 5 | 5^⁴ | 423
\( = a^2b^{-1}(ca)^3c^{-1}b^{-1} = 1 \rangle \) | 11 | 11^⁴ | 494

Inv. are the abelian invariants of the commutator factor of \( \ker \varphi \). Module is the structure of the module found by the vector enumerator for the particular prime. Time is in seconds on a 2.4GHz Pentium 4 machine under Linux.

Table 3: Example calculations
Acknowledgments

The first author would like to thank the School of Mathematics and Statistics of the University of Western Australia and in particular Cheryl Praeger for their hospitality during a visit in the Summer of 2000, during which this project was started. The second author acknowledges the support of ARC Small Grant and a Fay Gale fellowship.

We also would like to thank Derek Holt for pointing out a better rewriting system for alternating groups.

References


