Computing finite soluble quotients

ALICE C. NIEMEYER

March 1993

Abstract

A finite soluble quotient algorithm which computes power conjugate presentations for finite soluble quotients of finitely presented groups is described. A version of this algorithm has been implemented in **C** and is available as the ANU Soluble Quotient Program.

Polycyclic presentations are a natural way of describing polycyclic groups as they exhibit a polycyclic series of the group. From a computational point of view they are also very useful, since they allow the computation (by collection) of a normal word for every element in the group. This enables the computation of products and inverses of group elements. Algorithms using such descriptions for polycyclic groups are an integral part of the computational group theory systems Cayley (Cannon, 1984) and GAP (Schönert et al., 1993). Not every soluble group is polycyclic, but every finite soluble group is polycyclic. Baumslag, Cannonito, and Miller (1981a, 1981b) describe an algorithm which decides whether a given soluble group is polycyclic. It has been partly implemented by Sims (1990).

In this paper attention is restricted to finite soluble groups. Often a finite soluble group arises as a quotient of a finitely presented group. The task of a soluble quotient algorithm is to compute a polycyclic presentation for such a quotient. Clearly there are soluble quotient algorithms, but the problem is to describe an algorithm performing well in practice. A number of proposals for a finite soluble quotient algorithm have been made – these include those by Wamsley (1977), Leedham-Green (1984) and Plesken (1987). The latter has been developed, analysed and implemented by Wegner (1992).

Here the aim is to outline a new finite soluble quotient algorithm which computes a *power conjugate presentation* for a finite soluble quotient of a finitely presented group. New features are the use of a vector enumerator and the intermediate presentations considered. With respect to the latter it has some similarities to the p-quotient algorithm (Havas & Newman, 1980) and the nilpotent quotient algorithm (Nickel, in preparation). A version of this algorithm has been implemented in **C** and is available as the ANU Soluble Quotient Program. A more detailed description (including proofs) of the algorithm is in preparation.

The attention now is on the details of the algorithm. To begin with the kinds of polycyclic presentations to be considered are described more precisely. Let G be a finite soluble group and let the series $G = G_0 \geq G_1 \geq \cdots \geq G_n = \{1\}$ be a composition series for G with cyclic factors of prime order. Choose elements $a_i \in G$ for $1 \leq i \leq n$ such that $G_{i-1} = \langle G_i, a_i \rangle$; let p_i be the order of the factor G_{i-1}/G_i. Then $A = \{a_1, \ldots, a_n\}$ is a generating set for G and the set

$$R = \{a_i^{p_i} = v_{ii},\ a_k^{a_j} = v_{jk} \mid 1 \leq i \leq n,\ 1 \leq j < k \leq n\},$$

where v_{ij} is a word in the generators $a_{j+1}, \ldots, a_n$, is a defining set of relations for G. The presentation $\{A \mid R\}$ is called a *power conjugate presentation* for G. On the other hand, a power conjugate presentation $\{A \mid R\}$ of a group G exhibits the composition series $G = G_0 \geq G_1 \geq \cdots \geq G_n = \langle 1 \rangle$, where $G_{i-1} = \langle a_i, \ldots, a_n \rangle$ for $1 \leq i \leq n$. A word $w(a_1, \ldots, a_n)$ in the generators is *normal* if it is of the form $a_1^{e_1} \cdots a_n^{e_n}$ with $0 \leq e_i < p_i$. Collection relative to a power conjugate presentation (see e.g. Leedham-Green & Soicher, 1990) consists of computing a normal word representing the same element of G as a given arbitrary word in the generators. Multiplication of two elements of G amounts to computing a normal word for their product, given by concatenation, and the inversion of a group element amounts to computing a normal word for its formal inverse. In general, there may exist many normal words representing a given group element. If the normal word is unique, the power conjugate presentation is *consistent*. In this case two group elements are equal only if they are represented by the same normal word. For the finite soluble group G the order is then $\prod_{i=1}^{n} p_i$. It will be assumed from now on that the words v_{ij} in a power conjugate presentation are normal.

Let G be a group and p a prime. The series $G = \mathcal{P}_0^p(G) \geq \mathcal{P}_1^p(G) \geq \cdots$ with $\mathcal{P}_i^p(G) = [\mathcal{P}_{i-1}^p(G), G] \left(\mathcal{P}_{i-1}^p(G)\right)^p$ for $i \geq 1$ is called the *lower exponent-p central series* of G. If there exists an integer $c \geq 0$ such that $\mathcal{P}_c^p(G) = \langle 1 \rangle$, then the smallest such integer is called the *exponent-p class* of G.

Those finite soluble quotients of a finitely presented group G for which the algorithm to be presented computes a power conjugate presentation are described now more precisely. Let $\mathcal{L} = [(p_1, c_1), \ldots, (p_k, c_k)]$ be a list of pairs consisting of a prime, p_i, and a non-negative integer, c_i, with $p_i \neq p_{i+1}$. For $1 \leq i \leq k$ and $0 \leq j \leq c_i$ define the list $\mathcal{L}_{i,j} = [(p_1, c_1), \ldots, (p_{i-1}, c_{i-1}), (p_i, j)]$. Define $\mathcal{L}_{1,0}(G) = G$. For $1 \leq i \leq k$ and $1 \leq j \leq c_i$ define the subgroups

$$\mathcal{L}_{i,j}(G) = \mathcal{P}_j^{p_i}(\mathcal{L}_{i,0}(G))$$

and for $1 \leq i < k$ define the subgroups

$$\mathcal{L}_{i+1,0}(G) = \mathcal{L}_{i,c_i}(G)$$

and $\mathcal{L}(G) = \mathcal{L}_{k,c_k}(G)$. Note that for $j < c_i$

$$\mathcal{L}_{i,j}(G) \geq \mathcal{L}_{i,j+1}(G).$$

The chain of subgroups

$$G = \mathcal{L}_{1,0}(G) \geq \mathcal{L}_{1,1}(G) \geq \cdots \geq \mathcal{L}_{1,c_1}(G) = \mathcal{L}_{2,0}(G) \geq \cdots \geq \mathcal{L}_{k,c_k}(G) = \mathcal{L}(G)$$

is called the *soluble $\mathcal{L}$-series* of G. For a given i the series $\mathcal{L}_{i,0}(G) \geq \cdots \geq \mathcal{L}_{i,c_i}(G)$ is an initial segment of the lower exponent-p central series of $\mathcal{L}_{i,0}(G)$ and $\mathcal{L}_{i,0}(G)/\mathcal{L}_{i,c_i}(G)$ is a p_i-group of exponent p_i-class at most c_i. The soluble quotient algorithm presented here computes a power conjugate presentation for the quotient $G/\mathcal{L}(G)$ which exhibits a composition series of this quotient which is a refinement of the soluble $\mathcal{L}$-series.

For mathematical and algorithmic reasons power conjugate presentations with an added feature are considered. Let $\{\mathcal{A} \mid \mathcal{R}\}$ be a power conjugate presentation for a finite soluble group H with $\mathcal{A} = \{a_1, \ldots, a_n\}$. Let d be the minimal number of generators in $\mathcal{A}$ required to generate H. Assume there exists a d-element subset $\mathcal{X}$ of $\mathcal{A}$ such that $\mathcal{X}$ generates H and for each generator $a \in \mathcal{A}\backslash\mathcal{X}$ there is at least one relation of $\mathcal{R}$ having a as the *last* generator on the right hand side and occurring with exponent 1. Choose exactly one of these relations and call it the *definition* of a. The presentation $\{\mathcal{A} \mid \mathcal{R}\}$ together with chosen definitions for the generator in $\mathcal{A}\backslash\mathcal{X}$ is called *labelled.* Let G be a group with generating set $\{g_1, \ldots, g_b\}$ and τ an epimorphism of G onto H. For $i = 1, \ldots, b$ let w_i be the normal word equivalent to $\tau(g_i)$. If $a \in \mathcal{X}$ is the *last* generator in at least one w_i occurring with exponent 1 and we have chosen one such w_i, we call this the *definition of* a. For each $a \in \mathcal{X}$ there is a maximal k and a maximal c such that $a \in \mathcal{L}_{k,c}(H)$. We call τ a *labelled* epimorphism if each generator $a \in \mathcal{X}$ has a definition $\tau(g_i) = w_i$ and a is the only generator which occurs in w_i and does lie in $\mathcal{L}_{k,c}(H)$. If $\{\mathcal{A} \mid \mathcal{R}\}$ is a labelled power conjugate presentation for the group H and τ a labelled epimorphism from G to H, then every generator in $\mathcal{A}$ has a definition either as an image under τ or as a relation in $\mathcal{R}$. Further we can read off a preimage in G for each $a \in \mathcal{A}$ under τ. Thus we can compute a preimage in H for each $g \in G$ under τ.

Now the input and output of the finite soluble quotient algorithm are given precisely. The finite soluble quotient algorithm described here takes as input:

1) a finite presentation $\{g_1, \ldots, g_b \mid r_1(g_1, \ldots, g_b), \ldots, r_m(g_1, \ldots, g_b)\}$ for a group G;
2) a list $\mathcal{L} = [(p_1, c_1), \ldots, (p_k, c_k)]$, where p_i is a prime, $p_i \neq p_{i+1}$ and c_i a positive integer.

The output is:

1) a labelled power conjugate presentation for $H = G/\mathcal{L}(G)$ exhibiting a composition series refining the soluble $\mathcal{L}$-series of this quotient;
2) a labelled epimorphism $\tau : G \twoheadrightarrow H$.

Our algorithm proceeds by computing power conjugate presentations for the quotients $G/\mathcal{L}_{i,j}(G)$ in turn. Without loss of generality assume that so far a power conjugate presentation for $G/\mathcal{L}_{i,j}(G)$ has been computed for $j < c_i$. We compute a power conjugate presentation for $G/\mathcal{L}_{i,j+1}(G)$. The group $\mathcal{L}_{i,j}(G)/\mathcal{L}_{i,j+1}(G)$ is a p_i-group. The basic step takes as input:

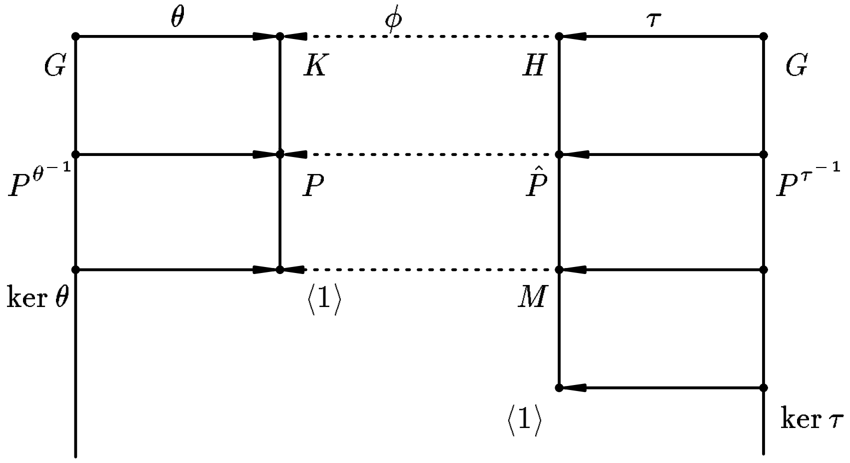
1) the finite presentation for G;
2) a labelled power conjugate presentation for the finite soluble quotient $K \cong G/\mathcal{L}_{i,j}(G)$ of G with $j < c_i$ which refines the $\mathcal{L}$-series of K.
3) a labelled epimorphism $\theta : G \twoheadrightarrow K$.

The output is:

1) a labelled power conjugate presentation for the finite soluble group $G/\mathcal{L}_{i,j+1}(G)$, denoted by H, exhibiting a composition series refining the $\mathcal{L}$-series of H;
2) an epimorphism $\phi : H \twoheadrightarrow K$;
3) a labelled epimorphism $\tau : G \twoheadrightarrow H$ with $\tau\phi = \theta$.

If during the basic step it is discovered that $\mathcal{L}_{i,j}(G) = \mathcal{L}_{i,j+1}(G)$, then $\mathcal{L}_{i+1,0}(G)$ is set to $\mathcal{L}_{i,j}(G)$.

The basic step is illustrated by the following diagram, where the input is described on the left and the output is described on the right. Put $p = p_i$, let P denote $\mathcal{L}_{i,0}(K)$, and $\hat{P}$ denote $\mathcal{L}_{i,0}(H)$. If $j = 0$ then P is trivial. The elementary abelian p-group $\ker\phi$ is denoted by M. The group $\hat{P}$ acts trivially on M, thus $\hat{P}$ is a central extension of P by M, and $\hat{P}$ is a p-group of exponent-p class at most one larger than the exponent-p class of P.



The basic step is now described in more detail. Let $\{A \mid R\}$ be the input consistent power conjugate presentation for K, where $A = \{a_1, \ldots, a_n\}$ and

$$R = \{a_i^{p_i} = v_{ii},\ a_k^{a_j} = v_{jk} \mid 1 \le i \le n,\ 1 \le j < k \le n\}.$$

Then $\{A \mid R\}$ is a power conjugate presentation for K with respect to a composition series which refines the soluble $\mathcal{L}$-series. Therefore there is an r such that $\{a_1P, \ldots, a_rP\}$ generates

K/P and $\{a_{r+1}, \ldots, a_n\}$ generates P. In order to obtain a power conjugate presentation for H a presentation $\{\hat{A} \mid \hat{R}\}$ for an extension, $\hat{K}$, of K which has H as a factor group, is determined first. The group $\hat{K}$ can be viewed as a generalised covering group and the presentation $\{\hat{A} \mid \hat{R}\}$ as a generalised power conjugate presentation. Let s be the sum of the number of relations in R which are not definitions and the number of generators of G whose images under θ are not definitions. Note that $s = (n-1)n/2 + b$. Introduce new generators $\{y_1, \ldots, y_s\}$ and define $\hat{A} = \{a_1, \ldots, a_n\} \cup \{y_1, \ldots, y_s\}$. We obtain $\hat{R}$ in the following way:

1) initialise $\hat{R}$ to contain all relations of R which are definitions;
2) modify each non-defining relation $u = v$ of R to read $u = vy_t$ for some $t \in \{1, \ldots, s\}$ and add the modified relation to $\hat{R}$, different non-defining relations are modified by different y_t;
3) add all relations of the form $[y_i, y_j^g] = 1$ for all normal $g = w(a_1, \ldots, a_r)$ and all relations $y_i^p = 1$ to $\hat{R}$ for $1 \leq i, j \leq s$;
4) add all relations $y_i^{a_j} = y_i$ for $j > r$ to $\hat{R}$ for $1 \leq i \leq s$.

Define a map σ from $\{g_1, \ldots, g_b\}$ to the group $\hat{K}$ by $g_i^\sigma = g_i^\theta y_t$ if g_i^θ is non-defining and $g_i^\sigma = g_i^\theta$ if g_i^θ is defining. This uses up the remaining elements of $\{y_1, \ldots, y_s\}$.

The subgroup $M = \ker \phi$ can be characterised as follows. It is the maximal $\mathbb{F}_p K$-module by which K can be extended so that P acts trivially on M and the extension is an epimorphic image of G and, where P is non-trivial, has the same generator number as K. Thus M is an $\mathbb{F}_p(K/P)$-module. Let Y be the free $\mathbb{F}_p(K/P)$-module on $\{y_1, \ldots, y_s\}$. Then M is a homomorphic image of Y. The kernel of the homomorphism from Y onto M can be computed effectively. In order to see this, the group $\hat{K}$ is studied in more detail. Consider the subgroup $\langle y_1, \ldots, y_s \rangle$ of $\hat{K}$ embedded into the additive group of Y.

One can collect in the group $\hat{K}$ relative to $\{\hat{A} \mid \hat{R}\}$. The definition of a normal word can be generalised for this presentation in the following way. A word in $\hat{A}$ is *normal* if it is of the form $w(a_1, \ldots, a_n) \cdot \Pi_{i=1}^s y_i^{f_i}$, where $w(a_1, \ldots, a_n)$ is a normal word in $\{a_1, \ldots, a_n\}$ and f_i is an element of $\mathbb{F}_p(K/P)$. The following steps can be applied to any non-normal word in $\hat{K}$.

1) replace a word $y_j^f y_i^{f'}$ with $f, f' \in \mathbb{F}_p(K/P)$ and $i < j$ by the word $y_i^{f'} y_j^f$;
2) replace a word $y_i^f y_i^{f'}$ with $f, f' \in \mathbb{F}_p(K/P)$ by the word $y_i^{f+f'}$;
3) replace a word $y_j^f a_i$ with $f \in \mathbb{F}_p(K/P)$ by the word $a_i y_j^{(fa_i)}$;
4) replace a word $a_i^{p_i}$ for $1 \leq i \leq n$ by the word v, where $a_i^{p_i} = v$ is a relation in $\hat{R}$;
5) replace a word $a_j a_i$ for $i < j$ by the word $a_i v$, where $a_j^{a_i} = v$ is a relation in $\hat{R}$.

In each step a word is replaced by a word representing the same element of $\hat{K}$. After applying a finite number of these steps to any non-normal word it is replaced by a normal word. Rules 1), 2) and 3) use the fact that Y is an $\mathbb{F}_p(K/P)$-module. Note that 4) and 5) resemble collection steps in a collection algorithm, where the power conjugate presentation is used to determine the replacement.

The following theorem, which is sometimes referred to as the "Consistency Theorem", allows us to describe the kernel of the homomorphism from Y onto M in a way suitable for computation. It considers certain non-normal words in $\hat{K}$. A non-normal word containing subwords in parentheses is replaced by a normal word by a collection process which applies at least one collection step to each subword in parentheses before proceeding. A non-normal word containing subwords in square brackets is replaced by a normal word by a collection process which replaces the subwords in square brackets by normal words before proceeding.

Theorem

Let Y be the free $\mathbb{F}_p(K/P)$-module on $\{y_1, \ldots, y_s\}$ and $\{\hat{A} \mid \hat{R}\}$ the presentation for the extension $\hat{K}$ of K as defined above. Let S be the set of elements of Y obtained by collecting

$$\begin{aligned}
&\left[(a_k\, a_j)\, a_i\right] \left[a_k\, (a_j\, a_i)\right]^{-1} && \text{for} \quad 1 \le i < j < k \le n,\\
&\left[(a_k^p)\, a_j\right] \left[a_k^{p-1}\, (a_k\, a_j)\right]^{-1} && \text{for} \quad 1 \le j < k \le n,\\
&\left[(a_j\, a_i)\, a_i^{p-1}\right] \left[a_j\, (a_i^p)\right]^{-1} && \text{for} \quad 1 \le i < j \le n,\\
&\left[(a_i^p)\, a_i\right] \left[a_i\, (a_i^p)\right]^{-1} && \text{for} \quad 1 \le i \le n.
\end{aligned}$$

Let $T = \{r_i(g_1^\sigma, \ldots, g_b^\sigma) \mid 1 \le i \le m\}$ be the set of elements of Y obtained by evaluating the relators of G in the images of the generators of G under the map σ.

M is isomorphic to $Y/(S \cup T)\mathbb{F}_p(K/P)$.

A proof for a very similar theorem can be found in Sims (to appear).

The technique of vector enumeration is used to compute a basis for the $\mathbb{F}_p(K/P)$-module M needed to obtain a power conjugate presentation for F/S. Its use in this context has been suggested by Leedham-Green (private communication, 1991). Here an implementation of such an algorithm, called vector enumerator, developed by Linton (1991) is used. In this context his vector enumerator takes as input a consistent power conjugate presentation for K, the set of free generators for Y, and the set $S \cup T$. The output is an $\mathbb{F}_p$-basis for M; an expression in this basis for the image under the generators of K/P of any basis element; and expressions for the images of the $\mathbb{F}_p(K/P)$-generators of Y in terms of the basis elements. This output is used to obtain a consistent power conjugate presentation for the extension H of K by M, an epimorphism τ from G to H and an epimorphism ϕ from H to K. In some cases additional work is necessary to transform the presentation and the epimorphism τ into a labelled presentation and a labelled epimorphism.

The ANU Soluble Quotient Program is an implementation of this algorithm written in **C**. It is available from the author. The table below gives the results of sample runs of the program. Under "Order" the order of the computed soluble quotient is listed and under "Time" the cpu time in seconds that the algorithm took on a Sparc Station 10/31 is given.

Presentation	$\mathcal{L}$	Order	Time
$\{a, b \mid a^3, b^6, (ab)^6, (a^{-1}b)^6\}$	$[(3,2),(2,2)]$	$2^{182}\cdot 3^3$	218
$\{a, b \mid (ab)^2 b^{-6}, a^4 b^{-1} ab^{-9} a^{-1} b\}$	$[(2,1),(3,1),(2,2),(3,2)]$	$2^4 \cdot 3^4$	59
$\{a, b \mid ab^2(ab^{-1})^2, (a^2 b)^2 a^{-1} ba^2 (bab)^{-1}\}$	$[(3,1),(2,2),(5,2)]$	$2^3 \cdot 3 \cdot 5^3$	77
$\{a, b \mid ab^2 a^{-1} b^{-1} a^3 b^{-1}, ba^2 b^{-1} a^{-1} b^3 a^{-1}\}$	$[(3,2),(2,2)]$	$2^8 \cdot 3^3$	2

In the first example 134 seconds of the 218 seconds were spent in the vector enumerator.

ACKNOWLEDGMENTS: I thank Dr L.G. Kovács, Dr C.R. Leedham-Green, Dr M.F. Newman, Dr Werner Nickel and Dr E.A. O'Brien for many encouraging discussions and generous help.

References

G. Baumslag, C.F. Miller III, F.B. Cannonito (1981a), "Computable algebra and group embeddings.", *J. Algebra*, **69**, 186–212.

G. Baumslag, C.F. Miller III, F.B. Cannonito (1981b), "Some recognizable properties of solvable groups", *Math. Z.*, **178**, 289–295.

John J. Cannon (1984), "An Introduction to the Group Theory Language, Cayley", *Computational Group Theory*, (Durham, 1982), pp. 145–183. Academic Press, London, New York.

George Havas and M.F. Newman (1980), "Application of computers to questions like those of Burnside", *Burnside Groups*, Lecture Notes in Math., **806**, (Bielefeld, 1977), pp. 211–230. Springer-Verlag, Berlin, Heidelberg, New York.

C.R. Leedham-Green (1984), "A Soluble Group Algorithm", *Computational Group Theory*, (Durham, 1982), pp. 85–101. Academic Press, London, New York.

C.R. Leedham-Green and L.H. Soicher (1990), "Collection from the left and other strategies", *J. Symbolic Comput.*, **9**, 665–675.

S.A. Linton (1991), "Constructing Matrix Representations of Finitely Presented Groups", *J. Symbolic Comput.*, **12**(4 & 5), 427–438.

Werner Nickel (in preparation), "A Nilpotent Quotient Algorithm".

W. Plesken (1987), "Towards a Soluble Quotient Algorithm", *J. Symbolic Comput.*, **4**, 111–122.

Martin Schönert et al. (1993), GAP – *Groups, Algorithms and Programming, version 3 release 2.* RWTH, Aachen: Lehrstuhl D für Mathematik.

Charles C. Sims (1990), "Implementing the Baumslag-Cannonito-Miller Polycyclic Quotient Algorithm", *J. Symbolic Comput.*, **9**(5 & 6), 707–723.

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

J.W. Wamsley (1977), "Computing soluble groups", A. Dold & B. Eckmann (Eds.), *Group Theory*, Lecture Notes in Math., **573**, (Canberra 1975), pp. 118–125. Springer-Verlag, Berlin, Heidelberg, New York.

Alexander Wegner (1992), *The Construction of Finite Soluble Factor Groups of Finitely Presented Groups and its Application*, PhD thesis. St. Andrews.

Alice C. Niemeyer
alice@pell.anu.edu.au
School of Mathematical Sciences
Australian National University
Canberra ACT 0200
Australia.