Verifying Nilpotence†

CHARLES C. SIMS

Department of Mathematics, Rutgers University, New Brunswick, NJ 08903, U.S.A.

(Received 15 July 1985)

This paper describes a new procedure, based on string rewriting rules, for verifying that a finitely presented group $G$ is nilpotent. If $G$ is not nilpotent, the procedure may not terminate. A preliminary computer implementation of the procedure has been used to prove a theorem about minimal presentations of free nilpotent groups of class 3. Finally, it is shown that the ideas presented here may be combined with work of Baumslag et al. (1981) to prove that the polycyclicity of a finitely presented group can be verified.

There are many computational problems in group theory which are known not to have algorithmic solutions. The word problem for finitely presented groups is probably the best known example. Let $G = \langle X | R \rangle$ be a finitely presented group. That is, $G$ is the quotient of the free group $F$ on the finite set $X$ by the smallest normal subgroup $N$ containing the elements defined by the finite set $R$ of words. There is no algorithm which can always decide, given $R$, whether a word $W$ represents the identity of $G$, or equivalently, whether $W$ defines an element of $N$. However, if $W$ represents the identity of $G$, this fact can be verified. We simply form in a systematic way products of conjugates of elements of $R$ and their inverses and then freely reduce the results. If $W = 1$ in $G$, then eventually the free reduction of $W$ will appear. If $W \neq 1$ in $G$, then we will go on forming products forever.

Even if we had good reason to believe that $W$ represents the identity of $G$, we would probably not attempt the procedure just described because experience teaches us that we would almost certainly have to wait a very long time to see our conjecture confirmed.

However, there are other more sophisticated procedures which we might use to try to convince ourselves or someone else that $W = 1$ in $G$. In fact, there are a number of procedures which “solve” certain “unsolvable” problems in group theory when the answer has a particular form and which have been found useful enough to be given computer implementations. The most frequently used is coset enumeration.

The main purpose of this paper is to point out that the nilpotence of a finitely presented group can be verified, to describe a procedure which may be useful in verifying nilpotence, and to report on some investigations using a computer implementation of this procedure. In the concluding section, we point out that results obtained here may be combined with a theorem of Baumslag et al. (1981) to show that the polycyclicity of a finitely presented group can be verified.

The notation and terminology used here are reasonably standard. One possible source of confusion needs to be mentioned. If $X$ is a set, then the free monoid on $X$ is the set $M$

† This research was supported in part by the National Science Foundation.

0747-7171/87/030231 + 17 $03.00/0 © 1987 Academic Press Inc. (London) Ltd.
of finite sequences of elements of $X$. Multiplication in $M$ is catenation. Elements of $M$ are
called words in $X$. However, in the context of groups the phrase "a word in $X"$ usually
means a finite sequence of elements of $X \cup X^{-1}$, where $X^{-1}$ is a set of formal inverses for
the elements in $X$. The empty word will be denoted $\theta$ and the identity element of a group
will be denoted $1$.

1. Coset Enumeration

Let $G = \langle X | \mathcal{R} \rangle$ be a finitely presented group and let $\mathcal{S}$ be a finite set of words defining
generators for a subgroup $H$ of $G$. In general, we cannot decide whether $H$ has finite index
in $G$. However, if $|G: H|$ is finite, then the procedure, or rather the family of procedures,
called coset enumeration can determine $|G : H|$. Several versions of coset enumeration
have been given computer implementations and used extensively. Details can be found in
Cannon et al. (1973) and Neubüser (1982) and in the references given there. For our
purposes we shall need only a general overview of coset enumeration.

Let $F$ be the free group on $X$. Any finitely generated subgroup $K$ of $F$ can be described
by an array $T$ of integers called a coset table. See Sims (1984) for the definition. Given $T$,
we can decide whether a word $W$ defines an element of $K$ and whether $K$ has finite index
in $F$. Given $\mathcal{R}$ and $\mathcal{S}$, we construct the table $T_0$ corresponding to the subgroup $K_0$ of $F$
generated by the elements of $\mathcal{S}$. We then systematically add conjugates of elements of $\mathcal{R}$
to $K_0$ to form an increasing sequence $K_0, K_1, \ldots$ of subgroups of $F$ corresponding to
coset tables $T_0, T_1, \ldots$. If $H$ has finite index in $G$, then this sequence stops with some
subgroup $K_n$ and coset table $T_n$ and $|G : H | = |F : K_n|$. Suppose $W$ is a word which defines
an element $w$ of $H$. Even if $H$ has infinite index in $G$, some $K_i$ will contain $w$ and so we
can verify membership in $H$. If the coset enumeration procedure is modified to keep track
of more information, then it is possible to determine a word in $\mathcal{S}$ which defines $w$. Taking
$\mathcal{S}$ to be the empty set, we have a procedure for verifying that $w = 1$.

Here is a list of verifications which can be performed using procedures based on coset
enumeration. We assume that $G = \langle X | \mathcal{R} \rangle$ and that $H$ is generated by the elements defined
by the finite set $\mathcal{S}$ of words.

1. Verify that $G$ is trivial.
2. Verify that $G$ is finite and compute $|G|$.
3. Verify that $H$ is trivial.
4. Verify that $|G : H|$ is finite and compute it.
5. Verify that a word $W$ in $X$ defines an element of $H$ and express that element as a
   word in $\mathcal{S}$.
6. Verify that $H$ is normal in $G$.

Note that $H$ is normal in $G$ if and only if $x^{-1}hx$ and $xhx^{-1}$ are in $H$ for all generators $h$
of $H$ and all $x$ in $X$. Thus, verification 6 reduces to a finite number of verifications of
type 5.

If the statement one is attempting to verify is actually false, then the procedures will not
terminate. Even if the statement is true, there is no way to make a simple $a$ priori estimate
of how long the verification will take.

2. Rewriting Processes

A number of authors have applied the ideas of term rewriting processes to the study of
algebraic systems. The first use of a computer implementation of such a process is
Verifying Nilpotence 233

described in Knuth & Bendix (1970). For monoids and groups, the simpler concept of a string rewriting process suffices for many purposes. A convenient statement of the main result concerning the application of these techniques to finitely presented groups and monoids may be found in Gilman (1979). Le Chenadec (1985) provides a more recent reference on these topics.

Let $M$ be the free monoid on a finite set $X$. We assume that $M$ is well-ordered by a relation $>$ which is translation invariant. That is, for all $U$, $V$, and $W$ in $M$, if $U > V$, then $WU > WV$ and $ UW > VW$. It follows that $U \geq \emptyset$ for all $U$ in $M$. For if $\emptyset > U$, then $U = U\emptyset > U^2$. By induction, we see that the sequence $U^i, i \geq 1$, is strictly decreasing and this contradicts our assumption that $M$ is well-ordered. It also follows that a word is greater than any of its proper subwords. We assume further that we have an algorithm which can decide, given words $U$ and $V$, whether $U > V$, $U = V$, or $V > U$.

Let us call an ordered pair $(L, R)$ of words with $L > R$ a rewriting rule. We say that $L$ is the left side of the rule and $R$ is the right side. Suppose $\mathcal{R}$ is a set of rewriting rules. Following Gilman (1979), we let $T(\mathcal{R})$ be the set of left sides of the elements of $\mathcal{R}$, we let $I(\mathcal{R})$ be the two-sided ideal of $M$ generated by $T(\mathcal{R})$, and we let $S(\mathcal{R})$ be the complement $M - I(\mathcal{R})$. We also denote by $\alpha(\mathcal{R})$ the two-sided congruence on $M$ generated by $\mathcal{R}$. The monoid $G$ defined by the relations $L = R$ with $(L, R)$ in $\mathcal{R}$ is the set of congruence classes of $\alpha(\mathcal{R})$.

If $\mathcal{R}$ is finite, we can define a rewriting process which, given a word $U$, computes a word $V$ in $S(\mathcal{R})$ such that $(U, V)$ is in $\alpha(\mathcal{R})$, that is, $U$ and $V$ define the same element of $G$.

Algorithm rewrite($\mathcal{R}, U$)
begin
set $V = U$;
while $V$ is not in $S(\mathcal{R})$ do begin
write $V$ as $ALC$, where $A$ and $C$ are in $M$ and $L$ is in $T(\mathcal{R})$;
let $(L, R)$ be in $\mathcal{R}$;
set $V = ARC$;
end;
return $V$;
end.

In general there is more than one decomposition of $V$ as $ALC$ and more than one choice for $R$, so there are potentially many ways to rewrite $U$. However, since $ALC > ARC$ and $>$ is a well-ordering, the process eventually terminates. If $V$ depends only on $U$ and not on the choices made in the rewriting process, then we say that $\mathcal{R}$ is confluent. In this case $S(\mathcal{R})$ is a transversal for $\alpha(\mathcal{R})$.

For us, the main result on rewriting processes is the following theorem. See, for example, Gilman (1979).

**Theorem 1.** Let $\mathcal{R}$ be a finite set of rewriting rules. It is possible to decide whether $\mathcal{R}$ is confluent. There is a procedure which will compute a finite confluent set $\mathcal{R}'$ of rewriting rules such that $\alpha(\mathcal{R}') = \alpha(\mathcal{R})$, provided such a set $\mathcal{R}'$ exists.

The procedure of Theorem 1 is called the Knuth–Bendix procedure for strings. If no set $\mathcal{R}'$ exists, then the procedure does not terminate. We can always assume that whenever $(L, R)$ is in $\mathcal{R}'$, then $R$ is in $S(\mathcal{R}')$ and every proper subword of $L$ is in $S(\mathcal{R}')$. With this
assumption, $\mathcal{R}$ is unique. This $\mathcal{R}$ will be called the normalised confluent rewriting system defined by $\mathcal{R}$ and the particular translation invariant well-ordering.

We shall provide a sketch of one version of the Knuth–Bendix procedure for strings. Descriptions of more efficient versions may be found in Le Chenadec (1985). The input to the procedure consists of the initial set $\mathcal{R}$ of rewriting rules and the translation invariant well-ordering $\succ$. Our version of the procedure uses four subroutines.

Subroutine $\texttt{ADD}(P, Q)$ (* $P$ and $Q$ are words. *)
begin
if $P > Q$ then
add $(P, Q)$ to $\mathcal{R}$
else if $Q > P$ then
add $(Q, P)$ to $\mathcal{R}$;
end;

Subroutine $\texttt{RIGHT}$ (* Rewrite right sides of rules. *)
begin
select words $A$, $B$, $C$, $D$, and $E$ such that $(A, BCD)$ and $(C, E)$ are in $\mathcal{R}$;
replace $(A, BCD)$ by $(A, BED)$;
end;

Subroutine $\texttt{LEFT}$ (* Rewrite left sides of rules. *)
begin
select words $A$, $B$, $C$, $D$, and $E$ such that $(ABC, D)$ and $(B, E)$ are distinct elements of $\mathcal{R}$;
delete $(ABC, D)$ from $\mathcal{R}$;
\texttt{ADD}(AEC, D);
end;

Subroutine $\texttt{OVERLAP}$ (* Produces new rules by overlapping left sides. *)
begin
select words $A$, $B$, $C$, $D$, and $E$ such that $B \neq \emptyset$ and both $(AB, D)$ and $(BC, E)$ are in $\mathcal{R}$;
let $P = \text{REWRITE}(\mathcal{R}, DC)$;
let $Q = \text{REWRITE}(\mathcal{R}, AE)$;
$\texttt{ADD}(P, Q)$;
end.

To perform the Knuth–Bendix procedure, we apply $\texttt{RIGHT}$, $\texttt{LEFT}$, and $\texttt{OVERLAP}$ repeatedly until no changes in $\mathcal{R}$ can be produced. The final value of $\mathcal{R}$ is the output. In order to guarantee termination when a finite confluent rewriting system exists, the order of the applications must be specified more precisely. One way is to say that changes with $\texttt{RIGHT}$ are to be tried first, then changes with $\texttt{LEFT}$, then changes with $\texttt{OVERLAP}$. In $\texttt{OVERLAP}$, the quintuples $(A, B, C, D, E)$ are to be tried in the order of the length of $W = ABC$. The name $\texttt{OVERLAP}$ comes from the fact that in $W$ the two left sides $AB$ and $BC$ overlap. When $A$, $B$, $C$, $D$, and $E$ are clear, we shall refer to this operation as overlapping $AB$ and $BC$.

If $X$ is a finite set, then there are many translation invariant well-orderings on the free monoid $M$ on $X$. To select one, we first choose the restriction to $X$. The most commonly used ordering on $M$ is the one in which words are ordered first by length and then lexicographically according to the selected order on $X$. However, as we shall see, other orderings can be useful.
Suppose \(|X| > 0\). We shall now define an ordering \(\gg\) on \(M\) which we shall call the collected ordering. If \(U \gg V\), then we shall say that \(U\) is less collected than \(V\) or that \(V\) is more collected than \(U\). We first choose a linear ordering of \(X\). Let \(U\) be a nonempty word. Set \(U \gg \emptyset\). Let \(y = y(U)\) be the largest element of \(X\) occurring as a factor in \(U\) and write \(U\) in the form
\[U = A_0 y A_1 y \ldots y A_r,\]
where each \(A_i = A_i(U)\) is a word which does not involve \(y\). Set \(r(U) = r\). If \(V\) is another nonempty word, we say \(U \gg V\), provided one of the following conditions holds:

(a) \(y(U)\) is greater than \(y(V)\).
(b) \(y(U) = y(V)\) and \(r(U) > r(V)\).
(c) \(y(U) = y(V)\), \(r(U) = r(V)\), and for some \(i\) with \(1 \leq i \leq r(U)\) we have \(A_i(U) \gg A_i(V)\) and \(A_j(U) = A_j(V)\) for \(1 \leq j < i\).

Because of (c), this definition is recursive.

As an example, let us suppose that \(X = \{a, b, c\}\) and \(a \gg b \gg c\). Then
\[baca \gg caba \gg b^{100}.\]

It is not hard to see that \(\gg\) is a well-ordering. Checking translation invariance is slightly tedious, but straightforward.

3. Polycyclic Groups

A group \(G\) is said to be polycyclic if it possesses a series of subgroups
\[G = G_1 \supseteq G_2 \supseteq \ldots \supseteq G_{m+1} = 1\]
such that \(G_{i+1}\) is normal in \(G_i\) and \(G_i/G_{i+1}\) is cyclic, \(1 \leq i \leq m\). Let us call such a series a polycyclic series for \(G\). Suppose \(G\) is polycyclic and the subgroups \(G_i\) form a polycyclic series. For \(1 \leq i \leq m\), let \(a_i\) be an element of \(G_i\) mapping onto a generator of \(G_i/G_{i+1}\). Then \(G_i = \langle a_1, \ldots, a_m \rangle\). We shall say that a finite sequence of generators for \(G\) obtained in this way is a polycyclic sequence of generators. If \(1 \leq i < j \leq m\), then \(a_i^{-1} a_j a_i\) and \(a_i a_j a_i^{-1}\) are in \(G_{i+1}\). If \(G_i/G_{i+1}\) is finite of order \(n_i\), then \(a_i^{n_i}\) is also in \(G_{i+1}\). To avoid awkward phrases like “provided \(n_i\) is defined”, let us set \(n_i = 0\) when \(G_i/G_{i+1}\) is infinite. Any element of \(G\) can be expressed as \(a_1^{n_1} \ldots a_m^{n_m}\) and the \(a_i\) are unique if we assume that \(0 \leq n_i < n_i\) when \(n_i > 0\). The process of computing this normal form for an element of \(G\) given by an arbitrary word in the \(a_i\) is called collection. Information about efficient collection algorithms may be found in Havas & Nicholson (1976).

A polycyclic group \(G\) has many nice properties. It is solvable and Hopfian and all subgroups are finitely generated. Given generators for a subgroup \(H\) of \(G\), we can decide membership in \(H\). If \(a_1, \ldots, a_m\) is a sequence of generators for \(G\) and \(b_1, \ldots, b_m\) are any elements of \(G\), then we can decide whether the map \(a_i \mapsto b_i\) extends to an automorphism \(\sigma\) of \(G\) and, if so, we can compute \(\sigma^{-1}\).

Now let \(a_1, \ldots, a_m\) be a sequence of abstract generators, let \(n_1, \ldots, n_m\) be a sequence of nonnegative integers, and for \(1 \leq i \leq m\) and \(1 \leq j \leq m\) let \(U_i, V_{ij}, W_{ij}\) be words in \(a_{i+1}, \ldots, a_m\) and their inverses. If \(n_i = 0\), we shall assume that \(U_i\) is the empty word. The
group $G$ generated by $a_1, \ldots, a_m$ and defined by the relations

\[
\begin{align*}
    a_i^n &= U_i, & 1 \leq i \leq m, \\
    a_i^{-1} a_i a_i &= V_{ij}, & 1 \leq i < j \leq m, \\
    a_i a_j a_i^{-1} &= W_{ij}, & 1 \leq i < j \leq m, \\
\end{align*}
\]  

(\ast)

is polycyclic and $a_1, \ldots, a_m$ is a polycyclic generating sequence for $G$. If all the $n_i$ are positive, then the relations with right side $W_{ij}$ may be omitted and the group defined is still polycyclic. However, if some of the $n_i$ are 0, then these additional relations are necessary. For example, $\langle a, b|a^{-1}ba=b^2 \rangle$ is not polycyclic and has no largest polycyclic quotient. A presentation of the form (\ast) will be called a polycyclic presentation.

Suppose $G$ is defined by (\ast) and set $G_i = \langle a_1, \ldots, a_m \rangle$. The order of $G_i/G_{i+1}$ may be finite even if $n_i = 0$ and may be less than $n_i$ if $n_i$ is positive. If $G_i/G_{i+1}$ has order $n_i$ when $n_i > 0$ and is infinite when $n_i = 0$, then (\ast) is called a consistent polycyclic presentation. In this case, every element of $G$ is expressible uniquely as $a_1^{s_1} \cdots a_m^{s_m}$, where $0 \leq s_i < n_i$ if $n_i > 0$.

Assume (\ast) is consistent. Then in $G$ we have

\[
\begin{align*}
    a_i a_i^{-1} &= 1, & 1 \leq i \leq m, \\
    a_i^{-1} a_i &= 1, \\
    a_i^n &= U_i, & n_i > 0, \\
    a_i^{-1} &= a_i^{n_i-1} U_i^{-1}, \\
    a_j a_i &= a_i V_{ij}, \\
    a_j^{-1} a_i &= a_i^{-1} V_{ij}^{-1}, \\
    a_j a_i^{-1} &= a_i^{-1} W_{ij}, \\
    a_j^{-1} a_i^{-1} &= a_i^{-1} W_{ij}^{-1}. \\
\end{align*}
\]  

(\ast\ast)

If we order the free monoid $M$ on $X = \{a_1, a_1^{-1}, \ldots, a_m, a_m^{-1}\}$ using the collected ordering $\gg$ with

\[
    a_1^{-1} \gg a_1 \gg a_2^{-1} \gg a_2 \gg \ldots \gg a_m^{-1} \gg a_m,
\]

then each word on the left of one of these relations is greater than the word on the right. Considered as rewriting rules, these relations suffice to rewrite or collect any word into standard form. Thus, this set of rules is confluent.

**THEOREM 2.** Suppose $G = \langle X|\mathcal{R} \rangle$ is a finitely presented group. If $G$ is polycyclic and $X = \{a_1, \ldots, a_m\}$, where $a_1, \ldots, a_m$ is a polycyclic sequence of generators, then this fact can be verified.

**PROOF.** Let $\mathcal{R}$ be the set of rewriting rules consisting of the pairs $(S, \emptyset)$ with $S$ in $\mathcal{I}$ together with the rules $(a_i a_i^{-1}, \emptyset)$ and $(a_i^{-1} a_i, \emptyset)$, $1 \leq i \leq m$. Because the set $\mathcal{R}'$ of rules in (\ast\ast) is a confluent set with $\alpha(\mathcal{R}') = \alpha(\mathcal{R})$, the procedure of Theorem 1 will terminate, and assuming the output is normalised, will give $\mathcal{R}$. If the conjectured statement is false, then either the procedure of Theorem 1 will not terminate, or it will terminate with a rewriting system which does not have the form (\ast\ast). $\square$
Some applications of Theorem 2 are described in Section 6.

We turn now to the problem of transforming a presentation \((*)\) into a consistent presentation of the same type. The general method of Theorem 2 could be used, but there are more efficient techniques for this special case. We start with a simple observation.

**Theorem 3.** If \((*)\) is consistent, then the relations

\[ a_i a_j a_i^{-1} = W_{ij}, \quad 1 \leq i < j \leq m, \]

are redundant.

**Proof.** Assume \((*)\) is consistent. The relation \(a_i a_j a_i^{-1} = W_{ij}\) is equivalent to \(a_i^{-1} W_{ij} a_i = a_j\). Using only the relations \(a_i^{-1} a_k a_i = V_{ik}, i < k \leq m\), we can find a word \(T\) in \(a_{i+1}, \ldots, a_m\) and their inverses such that \(T\) and \(a_i^{-1} W_{ij} a_i\) define the same element of the group. The relation \(T = a_j\) is a relation on \(a_{i+1}, \ldots, a_m\), and because of consistency is a consequence of the relations in \((*)\) which involve only these generators. Thus \(a_i a_j a_i^{-1} = W_{ij}\) is redundant. \(\square\)

Here are two simple examples illustrating Theorem 3. Let

\[ K = \langle a, b | a^{-1} b a = b^{-1}, a b a^{-1} = b^{-1} \rangle. \]

This presentation is a consistent polycyclic presentation and the second relation is derivable from the first. Now let

\[ L = \langle a, b | a^{-1} b a = b^3, a b a^{-1} = b^7 \rangle. \]

This is a polycyclic presentation but it is not consistent. In \(L\) we have

\[ b = a^{-1} b^7 a = (a^{-1} b a)^7 = b^{21}, \]

so \(b^{20} = 1\). Neither defining relation for \(L\) is redundant. The presentation

\[ b^{20} = 1, \quad a^{-1} b a = b^3, \quad a b a^{-1} = b^7 \]

is consistent and the third relation is a consequence of the other two.

The relations \(a_i a_j a_i^{-1} = W_{ij}\) are needed primarily to insure that for a given \(i\) the elements \(V_{i,i+1}, V_{i,i+2}, \ldots, V_{i,m}\) generate \(\langle a_{i+1}, \ldots, a_m \rangle\).

The procedure we shall describe here for converting a presentation \((*)\) into a consistent presentation of the same type is only a slight generalisation of the reduction algorithm described in Havas & Newman (1980). It is included here for completeness. We first discuss testing for consistency.

If \(m = 1\), then we have only the relation \(a_1^i = 1\), and this is a consistent presentation. By induction on \(m\), we may assume that the relations on \(a_2, \ldots, a_m\) are a consistent polycyclic presentation for a group \(H\). We first check whether the map \(a_i \rightarrow V_{ij}, 2 \leq j \leq m\), extends to a homomorphism \(\sigma\) of \(H\) into itself. This is done by testing whether the \(V_{ij}\) satisfy the defining relations for \(H\). By Theorem 3, the relations \(a_i a_j a_i^{-1} = W_{ij}, 2 \leq i < j \leq m\), are redundant and do not need to be checked. If we have a homomorphism, then we test whether \(\sigma\) maps \(H\) onto \(H\) by checking whether \(\sigma(W_{ij}) = a_j, 2 \leq j \leq m\). Since \(H\) is polycyclic and therefore Hopfian, if \(\sigma\) is surjective, then \(\sigma\) is an automorphism. If \(n = n_1 = 0\), then the presentation is consistent. If \(n > 0\), then we must also check whether \(\sigma\) fixes \(U_j\) and whether \(\sigma^n\) is the inner automorphism of \(H\) induced by \(U_j\). See pages 128 and 129 of Zassenhaus (1958) for the relevant information about cyclic extensions.

If \((*)\) is not consistent, then in the process of testing consistency we will find two
words \( P = a_1^{\alpha_1} \ldots a_m^{\alpha_m} \) and \( Q = a_1^{\beta_1} \ldots a_m^{\beta_m} \) which should be equal but are not. We modify (\( \ast \)) using the following algorithm.

Algorithm MODIFY

begin
while \( P \neq Q \) do begin
convert the relation \( P = Q \) to one of the form \( a_i^e = R \), where \( e > 0 \), \( R \) is a word in \( a_{i+1}, \ldots, a_m \) and their inverses, and \( e < n_i \) if \( n_i > 0 \);
if \( n_i = 0 \) then begin
set \( n_i = e \) and \( U_i = R \);
stop;
end;
(\( \ast \) now \( 0 < e < n_i. \) \( \ast \))
set \( n = n_i \) and \( U = U_i \);
repeat
write \( n = qe + r \) with \( 0 \leq r < e \);
set \( T \) equal to the result of collecting \( UR^{-e} \);
if \( r \neq 0 \) then
set \( n = e \), \( U = R \), \( e = r \), \( R = T \);
until \( r = 0 \);
set \( n_i = e \) and \( U_i = R \);
set \( P = r \) and \( Q = \emptyset \);
end;
end.

When MODIFY stops, we have a new polycyclic presentation which defines the same group and either some \( n_i \) has been changed from 0 to a positive value or some positive \( n_i \) has been given a smaller positive value. We now check the new presentation for consistency. This process must stop after a finite number of iterations with a consistent presentation.

4. Nilpotent Groups

If \( G \) is a group, then the terms \( \Gamma_i(G) \) of the lower central series of \( G \) are defined recursively by \( \Gamma_1(G) = G \) and \( \Gamma_{i+1}(G) = [G, \Gamma_i(G)] \), \( i \geq 1 \). If \( \Gamma_i(G) = 1 \) for some \( i \), then \( G \) is nilpotent. The class of \( G \) is the smallest integer \( c \) such that \( \Gamma_{c+1}(G) = 1 \). If \( G \) is finitely generated and nilpotent, then \( G \) is polycyclic. In fact, \( G \) has a polycyclic series

\[
G = G_1 \supseteq \ldots \supseteq G_{m+1} = 1
\]

which is also a central series. That is, each \( G_i \) is normal in \( G \) and \( G_i/G_{i+1} \) is central in \( G/G_{i+1} \). If \( a_1, \ldots, a_m \) is a polycyclic sequence of generators obtained from such a series, then for \( 1 \leq i < j \leq m \) we have \( a_i^{-1}a_ja_i = a_{ij} \), where \( t_{ij} \) is in \( G_{i+1} \). Such a polycyclic generating sequence will be called central.

Let \( X \) be a finite set with \( |X| > 1 \), let \( X^{-1} \) be a set of formal inverses for the elements of \( X \), and let \( M \) be the free monoid on \( X \cup X^{-1} \). The free group \( F \) on \( X \) is the set of equivalence classes of \( M \) under free equivalence. When no confusion can occur, we shall refer to elements of \( F \) by elements of \( M \) which define them.

A basic sequence of commutators is an infinite sequence \( c_1, c_2, \ldots \) of elements of \( F \), each of which has associated with it a positive integer \( w_i \) called its weight, satisfying the following conditions:
(1) If \( i < j \), then \( w_i \leq w_j \).
(2) The commutators of weight 1 in the sequence are the elements of \( X \) in some order.
(3) If \( w_k > 1 \), then \( c_k = [c_j, c_i] \) for some \( i \) and \( j \) such that \( i < j \) and \( w_k = w_i + w_j \). If \( w_j \geq 2 \) so that \( c_j = [c_q, c_p] \), then \( p < i \).
(4) Every commutator \([c_j, c_i]\) which could occur under condition 3 does occur exactly once.

The term "sequence of basic commutators" is more common, but being basic is a property of the sequence, not of the individual commutators.

There are infinitely many basic sequences of commutators. The terms of weight at most 6 in one sequence for \( X = \{a, b\} \) are:

\[
\begin{align*}
c_1 &= a, & c_{13} &= [c_7, c_2],
c_2 &= b, & c_{14} &= [c_6, c_2],
c_3 &= [c_2, c_1], & c_{15} &= [c_5, c_4],
c_4 &= [c_3, c_1], & c_{16} &= [c_6, c_3],
c_5 &= [c_3, c_2], & c_{17} &= [c_7, c_3],
c_6 &= [c_4, c_1], & c_{18} &= [c_8, c_3],
c_7 &= [c_4, c_2], & c_{19} &= [c_6, c_1],
c_8 &= [c_5, c_2], & c_{20} &= [c_{11}, c_2],
c_9 &= [c_4, c_4], & c_{21} &= [c_{12}, c_2],
c_{10} &= [c_5, c_3], & c_{22} &= [c_{13}, c_2],
c_{11} &= [c_6, c_1], & c_{23} &= [c_{14}, c_2],
c_{12} &= [c_6, c_2].
\end{align*}
\]

Here \( c_1 \) and \( c_2 \) have weight 1, \( c_3 \) has weight 2, \( c_4 \) and \( c_5 \) have weight 3, \( c_6, c_7, \) and \( c_8 \) have weight 4, \( c_{9}, \ldots, c_{14} \) have weight 5, and \( c_{15}, \ldots, c_{23} \) have weight 6.

Suppose \( c_1, c_2, \ldots \) is any basic sequence of commutators in \( F \). For \( k \geq 1 \), the quotient \( \Gamma_k(F)/\Gamma_{k+1}(F) \) is a finitely generated free abelian group with a basis given by the images of the \( c_i \) of weight \( k \). The group \( G = F/\Gamma_{k+1}(F) \) is nilpotent of class \( k \) and the images of the \( c_i \) of weight less than or equal to \( k \) form a polycyclic sequence of generators for \( G \). The group \( G \) is called the free nilpotent group of class \( k \) on \( X \). The commutator collection process described in Section 11.1 of Hall (1959) allows us to determine a consistent polycyclic presentation for \( G \).

For \( k \geq 1 \), the subgroup \( \Gamma_k(F) \) is the normal closure in \( F \) of the left normed commutators \([u_1, \ldots, u_k]\), where the \( u_i \) range over \( X \). The number of such commutators is \(|X|^k \). A finitely presented group \( H = \langle X | R \rangle \) is nilpotent of class at most \( k - 1 \) if and only if all of these commutators are trivial in \( H \). Thus we can verify that \( H \) is nilpotent of class at most \( k - 1 \) using coset enumeration techniques as described in Section 1. Since we can simulate by a single procedure the effect of testing \( H \) for nilpotence of class at most \( k \) for each \( k \geq 1 \), we can verify that \( G \) is nilpotent. However, this approach is much too inefficient to be useful. It is hoped that the procedure described in Section 5 is more practical.

It is reasonable to ask whether \( \Gamma_k(F) \) is the normal closure of the commutators of weight \( k \) in a basic sequence of commutators. In general the answer is not known. Some partial results concerning this question are discussed in Section 6.

5. Nilpotent Quotients

Let \( G = \langle X | R \rangle \) be a finitely presented group. For any \( k > 0 \) we can compute a consistent polycyclic presentation for the nilpotent group \( G/\Gamma_{k+1}(G) \), the largest class \( k \)
quotient of \( G \). This observation is implicit in Chen et al. (1958), where it was first observed that the structure of the abelian groups \( \Gamma_j(G)/\Gamma_{j+1}(G) \) could be determined.

The simplest algorithm to state for computing \( G/\Gamma_{k+1}(G) \) is probably the following:

**Algorithm NILQUOT**

begin
  construct a consistent polycyclic presentation for the free nilpotent group of class \( k \) on \( X \);
  for each \( R \) in \( \mathcal{B} \) do begin
    let \( P \) be the result of collecting \( R \);
    set \( Q = \emptyset \);
    apply the procedure \textsc{modify} of Section 3;
  end;
  apply the consistency algorithm to the resulting presentation;
end.

If \( |X| \) and \( k \) are small, this approach is not unreasonable. However, in most cases the number of generators in the initial polycyclic presentation will be very large and it is likely that most \( n_i \) in the final presentation will be 1, indicating that the corresponding generators are redundant. A better approach would be to use an algorithm similar to the one for computing \( p \)-quotients described in Havas & Newman (1980). However, we shall not pursue this point here.

We can now describe our proposed procedure for verifying that \( G = \langle X | \mathcal{B} \rangle \) is nilpotent. It is based on the observation that \( G \) is nilpotent of class at most \( k \) if and only if the commutators of weight at most \( k \) in some basic sequence of commutators form a central polycyclic generating sequence for \( G \).

**Algorithm NILPOT**

begin
  compute the quotients \( G/\Gamma_{k+1}(G) \) for increasing \( k \) until \( \Gamma_{k+2}(G) = \Gamma_{k+1}(G) \);
  for this \( k \), add to \( X \) the commutators of weight 2 through \( k \) in some basic sequence of commutators and add the definitions of these commutators to \( \mathcal{B} \);
  use the procedure of Theorem 2 to verify that we have a central polycyclic generating sequence for \( G \);
end.

In practice, one would probably add in the second step only those commutators which are not redundant in the presentation obtained in the first step.

### 6. An Application

The techniques of Section 5 can be used to study presentations of free nilpotent groups. More precisely, suppose \( c_1, c_2, \ldots \) is a basic sequence of commutators in the free group \( F \) on a finite set \( X \). Let \( \mathcal{B}_k \) denote the set of \( c_i \) having weight \( k \). We would like to know whether the normal closure \( N_k \) of \( \mathcal{B}_k \) in \( F \) is \( \Gamma_k(F) \). It is not hard to show that \( N_k = \Gamma_k(F) \) if and only if \( G_k = \langle X | \mathcal{B}_k \rangle = F/N_k \) is nilpotent. In fact the largest nilpotent quotient of \( G_k \) is \( F/\Gamma_k(F) \). Thus, if \( G_k \) is nilpotent, then its class is \( k-1 \).

We should really write \( G_k(c_1, \ldots) \) and \( N_k(c_1, \ldots) \), since these groups may depend on the basic sequence of commutators chosen. We shall show that for small \( k \) the isomorphism type of \( G_k \) depends only on \( k \) and \( r = |X| \). This is obvious for \( k = 1 \) and
$k = 2$. The set $\mathcal{A}_1$ is $X$, so $N_1$ is $F$ and $G_1$ is trivial. Given two distinct elements $x$ and $y$ of $X$, either $[x, y]$ or $[y, x]$ is in $\mathcal{A}_2$. Thus $G_2$ is abelian and $N_2$ is $\Gamma_2(F)$.

Let us fix one basic sequence of commutators. If $c$ and $d$ are terms in this sequence, then we shall write $c > d$ if $c$ occurs later in the sequence than $d$.

Given the order on the commutators of weight 1, that is, on the elements of $X$, there are certain commutators which will always occur in the basic sequence. For example, let $L_k$ be the set of left normed commutators $[x_1, \ldots, x_k]$, where the $x_i$ are in $X$ and $x_1 > x_2 \leq x_3 \ldots \leq x_k$. Then $L_k$ is a subset of $\mathcal{A}_k$.

To describe other commutators which are always in $\mathcal{A}_k$, we shall introduce the notion of a pattern. For every positive integer $k$, the symbol $[k]$ is a pattern of weight $k$. Let $u$ and $v$ be patterns of weights $m$ and $n$, respectively. Then $[u, v]$ is a pattern of weight $m + n$ provided:

(a) $m \geq n \geq 2$.
(b) If $u = [p, q]$, then the weight of $q$ is at most $n$.

For example, $[6]$, $[[4], [[2], [2]]]$, and $[[[4], [3]], [4]]$ are patterns of weights 6, 8, and 11, respectively. To reduce the number of brackets, we shall drop the brackets around single integers when they are part of a larger pattern. Thus, the second example will be written $[4, [2, 2]]$. We shall also introduce left-normed patterns. Suppose $m_1 \geq m_2 \leq m_3 \leq \ldots \leq m_r$ are integers with $m_2 \geq 2$. The pattern $[m_1, \ldots, m_r]$ is defined recursively to be $[[m_1, \ldots, m_{r-1}], m_r]$. Thus, the third example can be written $[4, 3, 4]$.

For future reference, we list the patterns of weight up to 9.

\[
\begin{align*}
[1], & \quad [8], \quad [4, 4], \\
[2], & \quad [2, 2, 4], \quad [4, [2, 2]], \\
[3], & \quad [[2, 2], [2, 2]], \quad [5, 3], \\
[4], & \quad [3, 2, 3], \\
[2, 2], & \quad [6, 2], \quad [4, 2, 2], \\
[5], & \quad [2, 2, 2], \quad [3, 2], \\
[6], & \quad [5, 4], \\
[3, 3], & \quad [3, 2, 4], \\
[4, 2], & \quad [5, [2, 2]], \\
[2, 2, 2], & \quad [[3, 2], [2, 2]], \quad [6, 3], \\
[7], & \quad [3, 3, 3], \\
[4, 3], & \quad [4, 2, 3], \\
[2, 2, 3], & \quad [2, 2, 2], \\
[5, 2], & \quad [7, 2], \\
[3, 2, 2], & \quad [5, 2, 2], \\
& \quad [3, 2, 2, 2].
\end{align*}
\]

For each pattern $u$, we define the set $\mathcal{B}(u)$ of commutators belonging to $u$ as follows: If $u = [k]$, where $k$ is an integer, then $\mathcal{B}(u) = L_k$. If $u = [v, w]$, then $\mathcal{B}(u)$ is the set of
commutators \([a, b, c]\), where \(a\) is in \(\mathcal{B}(v)\) and \(b\) is in \(\mathcal{B}(w)\). For example, if \(u = [2, 2, 2]\), then \(\mathcal{B}(u)\) is the set of all \([a, b, c]\), where \(a, b,\) and \(c\) are in \(\mathcal{L}_2\).

For \(k \geq 1\), let \(\mathcal{R}_k\) be the union of the sets \(\mathcal{B}(u)\), where \(u\) ranges over the patterns of weight \(k\). It is easy to see that any commutator occurring in our basic sequence belongs to a unique pattern. Thus \(\mathcal{R}_k \subseteq \mathcal{S}_k\).

**Theorem 4.** To show that the isomorphism type of \(G_k\) depends only on \(k\) and \(r = |X|\), it suffices to prove that \(\mathcal{R}_k\) is always contained in \(N_k\).

**Proof.** Let \(c_1, c_2, \ldots\) and \(d_1, d_2, \ldots\) be two basic sequence of commutators in \(F\). Any permutation of \(X\) extends to an automorphism of \(F\). Applying such an automorphism, we may assume that \(c_i = d_i, 1 \leq i \leq r\). Thus, the sets \(\mathcal{R}_k\) are the same for the two sequences. If \(\mathcal{R}_k\) is contained in \(N_k\), then \(N_k\) is the normal closure of \(\mathcal{R}_k\) and hence \(N_k\) is the same for both sequences. Thus \(G_k\) is the same also. \(\square\)

For many patterns \(u\) of weight \(k\) we can prove that \(\mathcal{B}(u)\) is always contained in \(\mathcal{R}_k\). We shall call such patterns basic patterns.

**Theorem 5.** For \(k \geq 1\), the pattern \([k]\) is basic. Suppose that \(u\) is a pattern of the form \([v, w]\), where \(v\) and \(w\) are basic patterns and \(v\) has greater weight than \(w\). If \(v\) is \([p, q]\), then assume also that \(w\) has greater weight than \(q\). Then \(u\) is basic.

The proof of Theorem 5 is a straightforward induction on the weights of the patterns involved. Repeated application of Theorem 5 shows, for example, that \([6, 2], [3, 2]\) is a basic pattern.

**Lemma 6.** Suppose \(i \geq j \geq 2\) and \(u\) is the pattern \([i, j]\). Then \(\mathcal{B}(u)\) is contained in \(N_{i+j}\).

**Proof.** Let \(a\) be in \(\mathcal{L}_i\) and let \(b\) be in \(\mathcal{L}_j\). If \(i > j\), then \(a > b\) and \([a, b]\) is in \(\mathcal{R}_{i+j}\). Suppose \(i = j\). If \(a = b\), then \([a, b]\) is trivial. If \(a > b\), then \([a, b]\) is in \(\mathcal{R}_{i+j}\). If \(b > a\), then \([b, a] = [a, b]^{-1}\) is in \(\mathcal{R}_{i+j}\). In every case, \([a, b]\) is in \(N_{i+j}\). \(\square\)

**Theorem 7.** For \(k \leq 5\), the isomorphism type of \(G_k\) depends only on \(k\) and \(r\).

**Proof.** Theorem 5 and Lemma 6 suffice to show that for every pattern \(u\) of weight \(k\) at most \(5\), the set \(\mathcal{B}(u)\) is contained in \(N_k\). The theorem follows by Theorem 4. \(\square\)

We shall now show that the groups \(G_k\) are nilpotent for \(k \leq 4\). In doing so, we shall invoke the following elementary facts.

**Theorem 8.** Let \(G\) be a group and suppose \(Z\) is a subgroup of the centre of \(G\). If \(G/Z\) is nilpotent, then \(G\) is nilpotent.

**Theorem 9.** Suppose \(G\) is a group generated by a set \(X\). If for all \(k\)-element subsets \(Y\) of \(X\) the subgroup generated by \(Y\) is nilpotent of class at most \(k-1\), then \(G\) is nilpotent of class at most \(k-1\).

**Corollary 10.** To show that \(G_k\) is nilpotent, it is enough to consider the case \(|X| \leq k\).
Verifying Nilpotence 243

PROOF. Suppose \(|X| > k\). If \(Y\) is a \(k\)-element subset of \(X\), then the subsequence of \(c_1, c_2, \ldots\) consisting of those commutators which involve only generators in \(Y\) is a basic sequence of commutators for the free group \(F_1\) on \(Y\). Thus \(R_k\) contains the elements of weight \(k\) in a basic sequence of commutators for \(F_1\). By assumption, \(\langle Y \rangle\) is nilpotent of class at most \(k-1\). Thus \(G_k\) is nilpotent by Theorem 9. □

In what follows, we shall write \(a = c_1, b = c_2, c = c_3, \ldots\). The cases \(k = 1\) and \(k = 2\) have already been considered.

Suppose that \(k = |X| = 3\). We have the following six commutators in \(R_1 \cup R_2\):

\[
\begin{align*}
a, & \quad d = [b, a], \\
b, & \quad e = [c, a], \\
c, & \quad f = [c, b].
\end{align*}
\]

In \(G_3\), the following eight relations hold:

\[
\begin{align*}
[d, a] = 1, & \quad [e, b] = 1, \\
[d, b] = 1, & \quad [e, c] = 1, \\
[d, c] = 1, & \quad [f, b] = 1, \\
[e, a] = 1, & \quad [f, c] = 1.
\end{align*}
\]

Since \(d\) and \(e\) are central, by Theorem 8 we may set \(d\) and \(e\) equal to 1 without changing whether the group is nilpotent. But with \(d = e = 1\), we see that \(a\) is central and so \([f, a] = 1\). Therefore \(f\) is central and we may set \(f = 1\). The resulting group is abelian and hence \(G_3\) is nilpotent.

Now let us consider the case \(k = 4\). In Havas & Richardson (1983) there is a proof due to Groves that \(G_4\) is nilpotent when \(|X| = 2\). We shall reprove Groves’ result using the techniques of Section 5. If \(|X| = 2\), then there are five elements in \(R_1 \cup R_2 \cup R_3\). They are:

\[
\begin{align*}
a, & \quad c = [b, a], \\
b, & \quad d = [c, a], \\
e, & \quad f = [c, b].
\end{align*}
\]

In \(G_4\) we have the following relations:

\[
\begin{align*}
[d, a] = 1, & \quad [d, b] = 1, \\
[e, b] = 1.
\end{align*}
\]

Since \(d\) is central, we may assume that \(d = 1\). We now apply the procedure of Theorem 1 in the free monoid generated by \(\{a, a^{-1}, b, b^{-1}, c, c^{-1}, e, e^{-1}\}\) with the rewriting rules

\[
\begin{align*}
ba = abc, & \quad cb = bce, \\
ca = ac, & \quad eb = be,
\end{align*}
\]

together with \(xx^{-1} = 1\) and \(x^{-1}x = 1\) for \(x = a, b, c, \text{ and } e\). Here rules are being written as relations. We use collected order with

\[
a^{-1} \gg a \gg b^{-1} \gg b \gg c^{-1} \gg c \gg e^{-1} \gg e.
\]

Overlapping \(ca\) and \(aa^{-1}\), we get the rule \(aca^{-1} = c\). Then overlapping \(aa^{-1}\) and \(aca^{-1}\), we obtain \(ca^{-1} = a^{-1}c\). Similarly, we get

\[
\begin{align*}
c^{-1}a = ac^{-1}, & \quad c^{-1}a^{-1} = a^{-1}c^{-1}, \\
eb^{-1} = b^{-1}e, & \quad e^{-1}b = be^{-1}, \\
e^{-1}b^{-1} = b^{-1}e^{-1}.
\end{align*}
\]
Overlapping \( cb \) and \( bb^{-1} \), then \( bcb^{-1}e \) and \( ee^{-1} \), and finally \( b^{-1}b \) and \( bcb^{-1} \), we obtain \( cb^{-1} = b^{-1}ce^{-1} \). We get the rules
\[
\begin{align*}
c^{-1}b &= b^{-1}e^{-1}c^{-1}, \\
c^{-1}b^{-1} &= b^{-1}e^{-1}c^{-1}, \\
b^{-1}a &= ab^{-1}ec^{-1}, \\
ba^{-1} &= a^{-1}bc^{-1}, \\
b^{-1}a^{-1} &= a^{-1}b^{-1}ce^{-1},
\end{align*}
\]
by overlapping the following 15 pairs of words in the order specified:
\[
\begin{align*}
cc^{-1}, cb; \\
c^{-1}c, cb^{-1}; \\
b^{-1}b, ba; \\
ba, aa^{-1}; \\
b^{-1}b, ba; \\
cc^{-1}, cb; \\
c^{-1}bce, ee^{-1}; \\
c^{-1}bc, cc^{-1}; \\
c^{-1}b^{-1}ce^{-1}, e^{-1}e; \\
c^{-1}b^{-1}c, cc^{-1}; \\
b^{-1}abc, cc^{-1}; \\
b^{-1}ab, bb^{-1}; \\
aba^{-1}c, cc^{-1}; \\
a^{-1}a, aba^{-1}; \\
b^{-1}a^{-1}bc^{-1}, c^{-1}c; \\
b^{-1}a^{-1}b, bb^{-1}.
\end{align*}
\]
We now overlap \( cb \) and \( ba \) to get \( bcea = abcec \). Next, we overlap \( b^{-1}b \) and \( bcea \). One of the rewritings involved is fairly complicated.
\[
\begin{align*}
b^{-1}abcec &= ab^{-1}e^{-1}bced = ab^{-1}ebe^{-1}c^{-1}cec \\
&= ab^{-1}ebe^{-1}ec = ab^{-1}ebc \\
&= ab^{-1}lec^{-1}ec = abec.
\end{align*}
\]
Here we have underlined the subwords replaced. We now have the rule \( cea = aec \). Overlapping \( c^{-1}c \) and \( cea \) gives \( ea = ac^{-1}ec \). Next, we overlap \( eb \) and \( ba \). The rewriting involved here is
\[
bea = bac^{-1}ec = abcc^{-1}ec = abec
\]
and
\[
eabc = ac^{-1}ecbc = ac^{-1}ebcec
= ac^{-1}bcecc = abe^{-1}c^{-1}ec.
\]
This gives the rule \( abe^{-1}c^{-1}ec = abec \). Now we overlap the following six pairs of words:
\[
\begin{align*}
abe^{-1}c^{-1}ec, cc^{-1}; \\
abe^{-1}c^{-1}ec, ee^{-1}; \\
abe^{-1}c^{-1}ec, ee^{-1}; \\
abe^{-1}c^{-1}ec, ee^{-1}; \\
abe^{-1}c^{-1}ec, ee^{-1}; \\
abe^{-1}c^{-1}ec, ee^{-1}.
\end{align*}
\]
The result is the rule \( ec = ce \). Next, we reduce the right side of the earlier rule \( ea = ac^{-1}ec \) to get \( ea = ae \). We could continue the Knuth–Bendix procedure, but at this point we know that \( e \) is central, so we can set \( e = 1 \). The resulting group is clearly nilpotent.

A preliminary computer implementation confirms the computations just performed more or less instantaneously.

We turn now to the case \( k = 4, |X| = 3 \). There are 14 commutators with weight at most 3 in a basic sequence.
\[
\begin{align*}
a, \\
b, \\
c, \\
d = [b, a], \\
e = [c, a], \\
f = [c, b], \\
g = [d, a],
\end{align*}
\]
\[
\begin{align*}
h = [d, b], \\
i = [d, c], \\
j = [e, a], \\
k = [e, b], \\
m = [e, c], \\
n = [f, b], \\
p = [f, c].
\end{align*}
\]
There are 18 relations defining $G_4$.

\[
\begin{align*}
[e, d] &= 1, & [j, a] &= 1, \\
[f, d] &= 1, & [j, b] &= 1, \\
[f, e] &= 1, & [j, c] &= 1, \\
[g, a] &= 1, & [k, b] &= 1, \\
[g, b] &= 1, & [k, c] &= 1, \\
[g, c] &= 1, & [m, c] &= 1, \\
[h, b] &= 1, & [n, b] &= 1, \\
[h, c] &= 1, & [n, c] &= 1, \\
\end{align*}
\]

Applying our work in the case $|X| = 2$ to the subgroups $\langle a, b \rangle$, $\langle a, c \rangle$, and $\langle b, c \rangle$, we can add the following relations:

\[
\begin{align*}
[g, d] &= [h, a] = [h, d] = [h, g] = 1, \\
[j, e] &= [m, a] = [m, e] = [m, f] = 1, \\
[n, f] &= [p, b] = [p, f] = [p, n] = 1.
\end{align*}
\]

The elements $g$, $h$, and $j$ are central and may be set equal to 1.

We now feed the rules

\[
\begin{align*}
ba &= abd, & fe &= ef, \\
ca &= ace, & ic &= ci, \\
cb &= bce, & kb &= bk, \\
da &= ad, & k &= ck, \\
ab &= bd, & ma &= am, \\
dc &= cdi, & mc &= cm, \\
ea &= ae, & me &= em, \\
eb &= bek, & nb &= bn, \\
ec &= cem, & nc &= cn, \\
fb &= bfn, & nf &= fn, \\
fc &= csp, & pb &= bp, \\
ed &= de, & pc &= cp, \\
f &= df, & pf &= fp, \\
fd &= df, & pn &= np,
\end{align*}
\]

together with the rules $xx^{-1} = 1$ and $x^{-1}x = 1$ for each of the 11 remaining generators $x$, into the program referred to above. In slightly over one minute on a Sun 3/50 workstation the program returns a polycyclic presentation from which it is clear that this group, and hence $G_4$, is nilpotent.

When $X = \{a, b, c, d\}$, the initial presentation for $G_4$ involves 86 relations on 30 commutators. Applying results for the case $|X| = 3$ to the subgroups $\langle a, b, c \rangle$, $\langle a, b, d \rangle$, $\langle a, c, d \rangle$, and $\langle b, c, d \rangle$ and deleting generators which are obviously central, we get a presentation on 15 generators. In just under three and a half minutes on the Sun, this group is shown to be nilpotent.

As a result of these computations, we have proved the following theorem.

**Theorem 11.** Let $X$ be a finite set and let $\mathcal{R}_k$ be the set of commutators of weight $k$ in some basic sequence of commutators in the free group $F$ on $X$. If $k \leq 4$, then the normal closure of $\mathcal{R}_k$ in $F$ is $\Gamma_k(F)$. 
The case \( k = 5 \) has not been resolved. When \( |X|=2 \), the group \( G_5 \) is defined by twelve relations on eight commutators. The program verifies that \( G_5 \) is nilpotent in about 33 seconds. Work is continuing on larger values of \( |X| \), but a better implementation will probably be needed.

We can now extend Theorem 7.

**Lemma 12.** Suppose \( a, b, \) and \( c \) are in \( L_i, L_j, \) and \( L_k \), respectively. Then \( \Gamma_3(\langle a, b, c \rangle) \) is contained in \( N_{i+j+k} \).

**Proof.** We must show that \([a, b, c]\) is in \( N_{i+j+k} \). By Theorem 11, we may assume that \( a > b \leq c \). Under this assumption, \([a, b]\) is in \( R_{i+j} \). If \( c < [a, b] \), then \([a, b, c]\) is in \( R_{i+j+k} \). If \( c = [a, b] \), then \([a, b, c] = 1\). If \( c > [a, b] \), then \([c, [a, b]]\) is in \( R_{i+j+k} \) and \([a, b, c] = [c, [a, b]]^{-1}\) is in \( N_{i+j+k} \). In all cases, \([a, b, c]\) is in \( N_{i+j+k} \). □

**Lemma 13.** Suppose \( a, b, c, \) and \( d \) are in \( L_i, L_j, L_k, \) and \( L_l \), respectively. Then \( \Gamma_4(\langle a, b, c, d \rangle) \) is contained in \( N_{i+j+k+l} \).

**Proof.** By Theorem 11 we need only show that \([a, b, c, d]\) is in \( N_{i+j+k+l} \) when \( a > b \leq c \leq d \) and that \([[[a, b], [c, d]]]\) is in \( N_{i+j+k+l} \) when \( a > b, c > d, [a, b] > [c, d] \) and \( b \leq [c, d] \).

Assume first that \( a > b \leq c \leq d \). Then \([a, b]\) is in \( R_{i+j} \).

Case 1: \( c = [a, b] \). Here \([a, b, c] = 1\).

Case 2: \( c < [a, b] \). Here \([a, b, c]\) is in \( R_{i+j+k} \).

Case 2.1: \( d = [a, b, c] \). Here \([a, b, c, d] = 1\).

Case 2.2: \( d < [a, b, c] \). Here \([a, b, c, d]\) is in \( R_{i+j+k+l} \).

Case 2.3: \( d > [a, b, c] \). Here \([d, [a, b, c]]\) is in \( R_{i+j+k+l} \) and so

\[
[a, b, c, d] = [d, [a, b, c]]^{-1}
\]

is in \( N_{i+j+k+l} \).

Case 3: \( c > [a, b] \). Here \([c, [a, b]]\) is in \( R_{i+j+k} \) and \( d \) cannot equal \([c, [a, b]]\).

Case 3.1: \( d < [c, [a, b]] \). Here \([c, [a, b], d]\) is in \( R_{i+j+k+l} \). Thus, \( d \) and \([c, [a, b]]\) commute modulo \( N_{i+j+k+l} \), and therefore so do \( d \) and \([a, b, c] = [c, [a, b]]^{-1}\). Hence \([a, b, c, d] \) is in \( N_{i+j+k+l} \).

Case 3.2: \( d > [c, [a, b]] \). Here \([d, [c, [a, b]]]\) is in \( R_{i+j+k+l} \). Again, \( d \) and \([c, [a, b]]\) commute modulo \( N_{i+j+k+l} \) and \([a, b, c, d]\) is in \( N_{i+j+k+l} \).

Now assume that \( a > b, c > d, [a, b] > [c, d] \) and \( b \leq [c, d] \). Then \( [[a, b], [c, d]] \) is in \( R_{i+j+k+l} \). □

**Theorem 14.** For \( k \leq 9 \), the isomorphism type of \( G_k \) depends only on \( k \) and \( r \).

**Proof.** Let \( u \) be a pattern of weight \( k \leq 9 \). By Theorem 5 and Lemmas 6, 12, and 13, it follows that \( \mathcal{R}(u) \) is contained in \( N_k \). The theorem follows from Theorem 4.

### 7. Verifying Polycyclicity

For any group \( G \), the terms \( G^{(k)} \) in the derived series of \( G \) are defined by \( G^{(1)} = G \) and \( G^{(k+1)} = [G^{(k)}, G^{(k)}] \) for \( k > 1 \). In Baumslag et al. (1981) the following theorem is proved.
THEOREM 15. Given a positive integer \( k \) and a finite presentation for a group \( G \), it is possible to decide whether \( H = G/G^{(k)} \) is polycyclic. If \( H \) is polycyclic, then it is possible to construct a polycyclic presentation for \( H \).

The last sentence is implicit in the proof of Theorem 3.1 of Baumslag et al. (1981). We shall only need the result that it is possible to determine words in the generators of \( G \) that define elements of \( G \) which map onto the terms in a polycyclic generating sequence for \( H \).

THEOREM 16. It is possible to verify that a finitely presented group is polycyclic.

PROOF. Let \( G \) be given by a finite presentation and assume that \( G \) is polycyclic. We can verify this fact with the following procedure. We compute the quotients \( G/G^{(k)} \) for increasing \( k \). Each of these groups will be polycyclic. We continue until \( (G/G^{(k+1)})^{(k)} \) is trivial. Then \( G \) is in fact isomorphic to \( H = G/G^{(k)} \). Let \( x_1, \ldots, x_s \) be the generators for \( G \). Choose words \( W_1, \ldots, W_s \) which define elements of \( G \) mapping onto a polycyclic generating sequence of \( H \). Let \( y_1, \ldots, y_s \) be new abstract generators. Add the \( y_i \) to the generators of \( G \) and add the relations \( y_i = W_i, 1 \leq i \leq s \). Now use the procedure of Theorem 2 to verify that the sequence \( x_1, \ldots, x_s, y_1, \ldots, y_s \) is a polycyclic generating sequence for the group thus presented. \( \square \)

The author wishes to acknowledge the helpful suggestions of M. F. Newman, who read an early version of this paper.

References


