Chapter 17. P-Groups and Soluble Groups

A p-group G is a group of order p^n, for some prime p and integer n. A soluble group is a group whose derived series reaches the identity subgroup. Both p-groups and soluble groups can be described by pc presentations, and can be studied using algorithms which exploit the properties of the series of subgroups associated with the pc presentation. Both classes of groups arise in special cases of advanced algorithms for permutation groups, and often those special cases are best dealt with by computing a pc presentation and then utilising an algorithm exploiting the pc presentation. This chapter briefly describes computation with pc presentations for p-groups and soluble groups. We restrict our attention to finite groups.

Presentations

A *presentation* of a group G in terms of generators and relations consists of a set X (of generators) and a set R of relations u=v, where u, v are words in X. For example, the presentation

$$< a, b \mid a^2 = b^3 = (ab)^4 = 1 >$$

of S_4 specifies the generators a, b and the relations a^2=1, b^3=1, and $(ab)^4$=1, where 1 represents the empty word (which corresponds to the identity element of the group). The words can include the inverses of the generators.

The group described by the presentation is isomorphic to a quotient of the *free* group F generated by X. The elements of F are equivalence classes of the words in X. Two words are equivalent if and only if they *freely reduce* to the same word. Free reduction repeatedly replaces all subwords of the form $x^{-1}x$ and xx^{-1} by the empty word until no such subwords exist. Hence, $bab^{-1}aa^{-1}ba$ freely reduces to baa. So the only identification of words in F is determined by the group axiom: $g \times g^{-1} = g^{-1} \times g$ = identity, for all elements g. Multiplication in F is performed by concatenating a word from each equivalence class (and then possibly freely reducing the word to obtain a class representative). The inverse of a word $x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}$ is the word $x_n^{-k_n} x_{n-1}^{-k_{n-1}} \cdots x_1^{-k_1}$.

The relations of the presentation of a group G specify further identifications of words of F. The relation u=v says that u and v represent the same element, or that uv^{-1} represents the identity element. In terms of the Cayley graph of G, the word uv^{-1} represents a loop (about every node). However, the consequences or deductions of the relations must also be taken into account. These are all the loops in the Cayley graph that can be constructed from the loops given by the relations. These are precisely those obtained by tracing a sequence of loops (corresponding to group multiplication), tracing a loop in the reverse direction (corresponding to inversion), and tracing a word w to some node g then tracing a loop l about g and then tracing w^{-1} back to the starting node (corresponding to the conjugate wlw^{-1}). Let N be the normal closure of R; that is, the subgroup of F generated by the relators uv^{-1}, where u=v is in R, and all consequences of these relators. Then N is a normal subgroup of F since it is closed under conjugation. The group G described by the presentation is the quotient F/N.

PC Presentations of Soluble Groups

A *power-commutation presentation* (pc presentation) of a finite soluble group G has the form

$$\begin{aligned} < a_1, a_2, \ldots, a_n \mid & a_i^{\rho(i)} = u_i, \ 1 \le i \le n, \\ & a_j a_i = a_i v_{ij}, \ 1 \le i < j \le n >, \end{aligned}$$

where each $\rho(i)$ is a positive integer, and u_i, v_{ij} are words in $\{a_{i+1}, a_{i+2}, \ldots, a_n\}$. (These are also called *AG systems* or *power-conjugate presentations.*) For example, the symmetric group S_4 has a pc presentation

$$\begin{aligned} < a_1, a_2, a_3, a_4 \mid & a_1^2 = a_3, a_2^3 = a_3^2 = a_4^2 = 1, \\ & a_2 a_1 = a_1 a_2^2 a_3, a_3 a_1 = a_1 a_3, a_4 a_1 = a_1 a_3 a_4, \\ & a_3 a_2 = a_2 a_4, a_4 a_2 = a_2 a_3 a_4, a_4 a_3 = a_3 a_4 >, \end{aligned}$$

where a_1 to a_4 correspond to the permutations (1,2,3,4), (1,2,3), (1,3)(2,4), and (1,2)(3,4) respectively.

The group

$$D_{12} = < a_1, a_2 \mid a_1^2 = a_2^6 = 1, a_2 a_1 = a_1 a_2^5 >$$

is the dihedral group of order 12 where a_2 corresponds to (1,2,3,4,5,6) and a_1 corresponds to (2,6)(3,5).

A pc presentation defines a chain of subgroups

$$G = G(1) \ge G(2) \ge \cdots \ge G(n) \ge G(n+1) = < \textit{identity} >,$$

where $G(i) = < a_i, a_{i+1}, \ldots, a_n >$. The relations $a_j a_i = a_i v_{ij}$ are equivalent to $a_i^{-1} a_j a_i = v_{ij}$. Since $v_{ij} \in G(i+1)$, the subgroup $G(i+1)$ is a *normal* subgroup of $G(i)$. We call such a chain of subgroups a *subnormal series* of G. The relation $a_i^{\rho(i)} = u_i$ and $u_i \in G(i+1)$ says that the quotient group $G(i)/G(i+1)$ is generated by $a_i G(i+1)$ and that the order of the quotient divides $\rho(i)$. (In most cases, we want the order to be precisely $\rho(i)$ - such pc presentations are said to be *consistent* - however, for an arbitrary pc presentation we can only say that the order divides $\rho(i)$.)

If each $\rho(i)$ is prime, then we say the series is *prime-step*. We can always refine a pc presentation to a prime-step series by introducing new generators which correspond to powers of the generators a_i where $\rho(i)$ is composite. For example, for D_{12} we introduce a_3 corresponding to a_2^2 and obtain the pc presentation

$$\begin{aligned} < a_1, a_2, a_3 \mid & a_1^2 = 1, a_2^2 = a_3, a_3^3 = 1, \\ & a_2 a_1 = a_1 a_2^5, a_3 a_1 = a_1 a_3^2, a_3 a_2 = a_2 a_3 > \end{aligned}$$

or instead of $a_1 a_2^5$ we could have written $a_1 a_2 a_3^2$.

The relations of a pc presentation allow any word in the generators to be reduced to a *normal word*

$$a_1^{k_1} a_2^{k_2} \cdots a_n^{k_n}, \text{ where } 0 \le k_i < \rho(i),$$

where the generators come in order and the exponents lie in a restricted range.

The relation $a_i^{\rho(i)} = u_i$ says that a subword $a_i^{n_i}$, where n_i is positive, can be written as $a_i^{k_i}(u_i)^m$ where $n_i = m\rho(i) + k_i$ and $0 \le k_i < \rho(i)$.

- The relation $a^{\rho(i)} = u_i$ says that a subword a_i^{-1} can be written as $a_i^{\rho(i)-1}u_i^{-1}$ since multiplying on the left by a_i gives the identity.
- The relation $a_j a_i = a_i v_{ij}$ for $i < j$ says that a subword $a_j a_i$ where the generators are not in order can be rewritten as $a_i v_{ij}$ where a_i is in order relative to all the generators in v_{ij}.

If the presentation is consistent, then the normal words will be unique; that is, every word in $\{a_1, a_2, ..., a_n\}$ reduces to only one normal word. Then we say the normal words are *canonical*. The order of the group is then $\rho(1)\times\rho(2)\times\cdots\times\rho(n)$, the number of normal words.

The *derived subgroup* (or commutator subgroup) of a group G is the subgroup generated by all commutators $[a,b]$ ($= a^{-1}b^{-1}ab$) where $a,b \in G$. It is written $[G,G]$ or $D(G)$. It is the smallest normal subgroup of G such that the quotient of G by the normal subgroup is abelian. The *derived series* of G is defined by

a. $D^0(G) = G$, and

b. $D^{c+1}(G) = [D^c(G), D^c(G)]$, for each $c \geq 0$.

A group G is defined to be *soluble* if and only if some term of the derived series of G is the identity subgroup. The derived series is a subnormal series since $D(G)$ is normal in G. It is also a *normal series* since each term $D^c(G)$ is normal in G. Furthermore, each quotient $D^c(G)/D^{c+1}(G)$ is abelian. Each abelian group H can be written as a direct product of cyclic groups of prime power order. Hence, H has a normal series

$$H = H(1) \geq H(2) \geq \cdots \geq H(m) \geq H(m+1) = < identity >,$$

where the quotients $H(i)/H(i+1)$ are *elementary abelian*. Therefore, the derived series of G can be refined to a normal series with elementary abelian quotients, and this *normal* series can be further refined to a prime-step *subnormal* series. This leads to the following definition:

A *conditioned* pc presentation of a finite soluble group G is a consistent pc presentation of the form

$$< a_1, a_2, ..., a_n \mid a_i^{p(i)} = u_i,\ 1 \leq i \leq n,$$
$$a_j a_i = a_i v_{ij},\ 1 \leq i < j \leq n >,$$

where

1. $p(i)$ is a prime,
2. u_i, v_{ij} are normal words in $\{a_{i+1}, a_{i+2}, ..., a_n\}$, and
3. there exists a *normal* series

$$G = N_1 \geq N_2 \geq \cdots \geq N_r \geq N_{r+1} = < identity >,$$

of G where each quotient N_i/N_{i+1} is elementary abelian, and each N_i has the form $< a_{n(i)}, a_{n(i)+1}, \cdots, a_n >$ for some integer $n(i)$.

Every finite soluble group has a conditioned pc presentation, and the extra properties are very useful when computing in a finite soluble group.

The example for S_4 is conditioned where $N_1 = G$, $N_2 = < a_2, a_3, a_4 >$, $N_3 = < a_3, a_4 >$, and $N_4 = < identity >$. The prime-step pc presentation for D_{12} with the relation $a_2 a_1 = a_1 a_2 a_3^2$ is conditioned where $N_1 = G$, $N_2 = < a_2, a_3 >$, $N_3 = < a_3 >$, and $N_4 = < identity >$.

PC Presentations of p-Groups

A *power-commutator presentation* (pc presentation) of a finite p-group G is a presentation of the form

$$< a_1, a_2, \ldots, a_n \mid a_i^{\rho(i)} = u_i,\ 1 \le i \le n, \\ [a_j, a_i] = v_{ij}^*,\ 1 \le i < j \le n >,$$

where each $\rho(i)$ is a power of the prime p, u_i are words in $\{a_{i+1}, a_{i+2}, \ldots, a_n\}$, and v_{ij}^*, are normal words in $\{a_{j+1}, a_{j+2}, \ldots, a_n\}$.

The relations $[a_j, a_i] = v_{ij}^*$ can be rewritten as $a_j a_i = a_i a_j v_{ij}^*$ and allow words to be reduced to normal words (as was the case for pc presentations of soluble groups). The pc presentation is *consistent* if every word reduces to a unique normal word. The order of the group G is then $\rho(1) \times \rho(2) \times \cdots \times \rho(n)$. If the pc presentation is prime-step - that is, each $\rho(i) = p$ - then $|G| = p^n$.

The pc presentations of p-groups are related to descending central chains of subgroups. A chain

$$G_0 \ge G_1 \ge \cdots \ge G_c \ge G_{c+1} \ge \cdots$$

of subgroups of a group G is a *descending central chain* of subgroups of G if

a. $G_0 = G$, and

b. $G_{c+1} \ge [G_c, G]$, for each c.

The *lower central chain* of G is defined by $\gamma_0(G) = \gamma_1(G) = G$ and $\gamma_{c+1}(G) = [\gamma_c(G), G]$ for $c \ge 1$. A group is *nilpotent* if some $\gamma_{c+1}(G) = \{\text{identity}\}$. All p-groups are nilpotent.

The *lower exponent-p-central* chain for a prime p is

$$G = P_0(G) \ge P_1(G) \ge \cdots \ge P_c(G) \ge P_{c+1}(G) \ge \cdots$$

where $P_{c+1}(G) \ge [P_c(G), G] \left[P_c(G)\right]^p$, for each c.

(Warning: There is much inconsistency in the literature about whether series begin at the zero-th term or the first term.)

Some important properties of these chains are summarised now.

1. Every descending central series is a *subnormal* series; that is G_{c+1} is normal in G_c.
2. Every descending central series is a *normal* series; that is G_{c+1} is normal in G.
3. Every descending central series is a *central* series; that is G_c/G_{c+1} is contained in the centre of G/G_{c+1}.
4. Each of the quotients G_c/G_{c+1} of a descending central series is abelian.
5. Each of the quotients $P_c(G)/P_{c+1}(G)$ of the lower exponent-p-central series is an elementary abelian p-group.

A pc presentation of a finite p-group G is *conditioned* if it is prime-step and defines a refinement of the lower exponent-p-central series. That is, it has the form

$$\begin{aligned} < a_1, a_2, \ldots, a_n \mid a_i^p &= u_i, \ 1 \le i \le n, \\ [a_j, a_i] &= v^*_{ij}, \ 1 \le i < j \le n > \end{aligned}$$

where u_i are words in $\{a_{i+1}, a_{i+2}, \ldots, a_n\}$, and v^*_{ij} are normal words in $\{a_{j+1}, a_{j+2}, \ldots, a_n\}$. Furthermore, there are integers $w(c)$ such that

$$P_c(G) \ = \ < a_{w(c)}, a_{w(c)+1}, \cdots, a_n >$$

for each c.

The *weight* of a generator a_i (denoted $wt(a_i)$) is the value c such that $w(c) \le i < w(c+1)$. The generators of weight 0 are the *defining generators*. There are d ($= w(1) - 1$) of them.

For example, the following pcp for the quaternion group of order 8:

$$\begin{aligned} < a_1, a_2, a_3 \mid a_1^2 &= a_3, a_2^2 = a_3, a_3^2 = 1, \\ [a_2,a_1] &= a_3, [a_3,a_1] = 1, [a_3,a_2] = 1 > \end{aligned} \tag{Q8}$$

is conditioned with $w(0) = 1$, $w(1) = 3$ and $w(2) = 4$. So $d = 2$.

For example, the group of order 2^6 given by the conditioned pcp

$$\begin{aligned} < b_1, b_2, b_3, b_4, b_5, b_6 \mid b_2^2 &= b_4, b_3^2 = b_5, b_4^2 = b_5, \\ [b_2,b_1] &= b_3, [b_3,b_1] = b_5, [b_3,b_2] = b_6, [b_4,b_1] = b_5 b_6 > \end{aligned} \tag{G64}$$

- it is usual to omit powers and commutators which are trivial - has $d = 2$, $w(0) = 1$, $w(1) = 3$, $w(2) = 5$, and $w(3) = 7$.

The defining generators are important because any algorithm which needs to compute an action by G only needs to use the action of the defining generators and not the action of all n generators.

The algorithms for p-groups that we will consider in this chapter do not use the weights of the generators or the fact that the presentation is conditioned. However, such presentations are necessary for many algorithms.

Collection

A *collection process* is an algorithm for determining an equivalent normal form of a word in $\{a_1, a_2, \ldots, a_n\}$. The algorithms look for subwords which prevent the word being normal and rewrite the subword. Table 1 gives (A) a list of (minimal) subwords which violate the property of being normal, and (B) the corresponding rewrite of the subword.

(A)	(B) soluble group	p-group	
a_i^{-1}	$a_i^{\rho(i)-1} u_i^{-1}$	$a_i^{\rho(i)-1} u_i^{-1}$	$1 \le i \le n$
$a_i^{\rho(i)}$	u_i	u_i	$1 \le i \le n$
$a_j a_i$	$a_i v_{ij}$	$a_i a_j v^*_{ij}$	$1 \le i < j \le n$

The outline of the algorithm is

Algorithm 1 : Outline of collection

```
Input: a pc presentation with generators $\{a_1, a_2, ..., a_n\}$;
       a word $w$ in $\{a_1, a_2, ..., a_n\}$;

Output: a normal word $w$ equivalent to the input word;

begin
  while $w$ is not normal do
    $w$ := the result of replacing a subword of $w$ which occurs in list (A) by
           its equivalent in list (B) ;
  end while;
end.
```

As an example consider the pcp of the quaternion group Q8. We will collect the word $a_3^{-1}a_2a_1a_2a_1^{-1}$ and highlight in bold the subword we are replacing at each step.

$a_3^{-1}a_2a_1a_2\mathbf{a_1^{-1}}$
$\mathbf{a_3^{-1}}a_2a_1a_2a_1a_3^{-1}$
$a_3\mathbf{a_2a_1}a_2a_1a_3^{-1}$
$a_3a_1a_2a_3\mathbf{a_2a_1}a_3^{-1}$
$\mathbf{a_3a_1}a_2a_3a_1a_2a_3a_3^{-1}$
$a_1\mathbf{a_3a_2}a_3a_1a_2a_3a_3^{-1}$
$a_1a_2\mathbf{a_3^2}a_1a_2a_3a_3^{-1}$
$a_1\mathbf{a_2a_1}a_2a_3a_3^{-1}$
$a_1^2a_2\mathbf{a_3a_2}a_3a_3^{-1}$
$a_1^2a_2^2\mathbf{a_3^2}a_3^{-1}$
$a_1^2\mathbf{a_2^2}a_3^{-1}$
$a_1^2a_3\mathbf{a_3^{-1}}$
$a_1^2\mathbf{a_3^2}$
$\mathbf{a_1^2}$
a_3

The collection processes always terminate. To show this we order the words and show that each iteration of the collection process results in a word that is "more normal" than its predecessor. The ordering $\gg$ is called the *collected ordering* (or syllable ordering). First, order the generators and their inverses so that

$$a_1^{-1} \gg a_1 \gg a_2^{-1} \gg a_2 \gg \cdots \gg a_n^{-1} \gg a_n.$$

Then specify that all nonempty words are greater than the empty word. If u is a non-empty word, define $y(u)$ to be the largest generator (or inverse) y in u. Write u as

$$u = A_0yA_1y \cdots yA_r$$

where the generator y does not occur in the subwords A_i. We write $A_i(u)$ and $r(u)$ to denote A_i and r when it is important to specify the word u. Then we define $u \gg v$ for non-empty words u, v if and only if either

1. $y(u) \gg y(v)$,
2. $y(u) = y(v)$ and $r(u) > r(v)$,
3. $y(u) = y(v)$, $r(u) = r(v)$, and for some j, $0 \le j \le r$, $A_j(u) \gg A_j(v)$ and $A_i(u) = A_i(v)$ for $i = 0,1,2...,j-1$.

The ordering $\gg$ is a well-ordering and is *translation invariant.* That is, if $u \gg v$ then $wu \gg wv$ and $uw \gg vw$ for all words w. This implies that u is greater than any subword of u.

From the definition of $\gg$ we can see that each word in list (A) is greater than the corresponding entry in list (B). Since $\gg$ is translation invariant, this means that the result of one replacement is less than its input word. So each iteration of the collection process decreases the word w (with respect to $\gg$) and makes it "more normal".

An efficient collection process is important. Efficiency depends on the strategy followed in choosing which subword to replace, and upon good implementations. The implementation might preprocess the pc presentation if one expects to perform many collections. The main strategies that have been studied are

- collection *from the right* in which the subword that is replaced is always the rightmost one;
- collection *to the left* in which the leftmost occurrence of the greatest (with respect to $\gg$) uncollected generator (or inverse) y is moved to the left by the choice of a suitable subword; (A generator is "uncollected" if it is in a subword which occurs in list (A).)
- collection *from the left* in which the subword that is replaced is always the leftmost one.

The treatment of inverses during the collection process is often simplified by first "clearing inverses" - that is, recursively applying the first row of Table 1 to eliminate inverses - before following one of the above strategies. We will not follow that practice in our examples, but as a consequence we sometimes require a substep to clear an inverse before the collection strategy can be applied.

We will repeat the collection of the word $a_3^{-1}a_2a_1a_2a_1^{-1}$ in Q8 with each of these strategies and highlight in bold the subword we are replacing at each step.

Collection from the right:

$a_3^{-1}a_2a_1a_2\mathbf{a_1^{-1}}$
$a_3^{-1}a_2a_1a_2a_1\mathbf{a_3^{-1}}$
$a_3^{-1}a_2a_1\mathbf{a_2a_1}a_3$
$a_3^{-1}a_2a_1a_1a_2\mathbf{a_3a_3}$
$a_3^{-1}a_2\mathbf{a_1a_1}a_2$
$a_3^{-1}a_2\mathbf{a_3a_2}$
$a_3^{-1}\mathbf{a_2a_2}a_3$
$a_3^{-1}\mathbf{a_3a_3}$
$\mathbf{a_3^{-1}}$
a_3

Collection to the left:

$a_3^{-1}a_2a_1a_2\mathbf{a}_1^{-1}$
$a_3^{-1}\mathbf{a}_2\mathbf{a}_1a_2a_1a_3^{-1}$
$\mathbf{a}_3^{-1}\mathbf{a}_1a_2a_3a_2a_1a_3^{-1}$
... in two substeps ... first a_3^{-1} replaced by a_3
$a_1a_3a_2a_3\mathbf{a}_2\mathbf{a}_1a_3^{-1}$
$a_1a_3a_2\mathbf{a}_3\mathbf{a}_1a_2a_3a_3^{-1}$
$a_1a_3\mathbf{a}_2\mathbf{a}_1a_3a_2a_3a_3^{-1}$
$a_1\mathbf{a}_3\mathbf{a}_1a_2a_3a_3a_2a_3a_3^{-1}$
$\mathbf{a}_1\mathbf{a}_1a_3a_2a_3a_3a_2a_3a_3^{-1}$
$a_3\mathbf{a}_3\mathbf{a}_2a_3a_3a_2a_3a_3^{-1}$
$\mathbf{a}_3\mathbf{a}_2a_3a_3a_3a_2a_3a_3^{-1}$
$a_2a_3a_3a_3\mathbf{a}_3\mathbf{a}_2a_3a_3^{-1}$
$a_2a_3a_3\mathbf{a}_3\mathbf{a}_2a_3a_3a_3^{-1}$
$a_2a_3\mathbf{a}_3\mathbf{a}_2a_3a_3a_3a_3^{-1}$
$a_2\mathbf{a}_3\mathbf{a}_2a_3a_3a_3a_3a_3^{-1}$
$\mathbf{a}_2\mathbf{a}_2a_3a_3a_3a_3a_3a_3^{-1}$
$a_3a_3a_3a_3a_3a_3\mathbf{a}_3^{-1}$
$\mathbf{a}_3\mathbf{a}_3a_3a_3a_3a_3a_3$
$\mathbf{a}_3\mathbf{a}_3a_3a_3a_3$
$\mathbf{a}_3\mathbf{a}_3a_3$
a_3

Collection from the left:

$\mathbf{a}_3^{-1}a_2a_1a_2a_1^{-1}$
$\mathbf{a}_3\mathbf{a}_2a_1a_2a_1^{-1}$
$a_2\mathbf{a}_3\mathbf{a}_1a_2a_1^{-1}$
$\mathbf{a}_2\mathbf{a}_1a_3a_2a_1^{-1}$
$a_1a_2\mathbf{a}_3\mathbf{a}_3a_2a_1^{-1}$
$a_1\mathbf{a}_2\mathbf{a}_2a_1^{-1}$
$a_1a_3\mathbf{a}_1^{-1}$
$a_1a_3a_1\mathbf{a}_3^{-1}$
$a_1\mathbf{a}_3\mathbf{a}_1a_3$
$\mathbf{a}_1\mathbf{a}_1a_3a_3$
$\mathbf{a}_3\mathbf{a}_3a_3$
a_3

The strategy should not only minimise the number of replacements but also control the length of the intermediate words. Current empirical and theoretical evidence strongly suggests that collection from the left is best in general.

Consistency

In the terminology of rewriting systems, a consistent pcp is an example of a *complete* system of rewrite rules. A rewrite rule is a pair (L,R) of words, and a rewrite system is a set of rewrite rules. A system defines a rewriting process on words where an occurrence of L as a subword may be replaced by R until no more replacements are possible. The result of rewriting is called a normal word. A rewrite system is *complete* if the process of rewriting always terminates, and for each word it returns a unique answer irrespective of the order of replacements of R for L. One checks completeness by guaranteeing termination of the rewriting process, and by checking that each *critical pair* rewrites to the same normal word. A critical pair is a minimal example of a word w which can be rewritten in more than one way (together with the pair of rewrite rules (L_1, R_1) and (L_2, R_2) such that L_1 and L_2 are subwords of w).

The rewrite rules for a pc presentation are the pairs (L,R), where L is an entry of list (A) and R is the corresponding entry of list (B). Collection is the rewriting process. For pcp's termination of rewriting is automatic. The critical pairs are determined by considering the minimal words which display an overlap of the subwords in list (A). For a finite soluble group, these overlaps are

(C) - finite soluble group	
$a_k a_j a_i$	$1 \le i < j < k \le n$
$a_j^{p(j)} a_i$	$1 \le i < j \le n$
$a_j a_i^{p(i)}$	$1 \le i < j \le n$
$a_i^{p(i)+1}$	$1 \le i \le n$

Each of these critical pairs has precisely two subwords in list (A). Each subword should be rewritten using the entry in list (B) and then collected to a normal word. It is not important which strategy is used to complete the collection. However, each of the two possible ways of beginning the collection process must be tried separately. The two resulting normal words are equal if the pcp is consistent.

For a finite p-group with a prime-step pc presentation, the overlaps are

(C) - finite p-group	
$a_k a_j a_i$	$1 \le i < j < k \le n$
$a_j^p a_i$	$1 \le i < j \le n$
$a_j a_i^p$	$1 \le i < j \le n$
a_i^{p+1}	$1 \le i \le n$

For cases where we know $a_1, a_2, ..., a_d$ are the defining generators of a p-group, the list of overlaps to check can be reduced to

(C') - finite p-group	
$a_k a_j a_i$	$1 \le i < j < k \le n$ and $i \le d$
$a_j^p a_i$	$1 \le i < j \le n$ and $i \le d$
$a_j a_i^p$	$1 \le i < j \le n$
a_i^{p+1}	$1 \le i \le n$

The proof that this reduced list suffices is difficult (and not explained here).

Elements and Subgroups of p-Groups

For ease of exposition, this section restricts discussion to the context of finite p-groups described by a consistent pcp. The algorithms and techniques carry over to finite soluble groups with only minor changes.

The canonical form of an element as a normal word

$$\prod_{i=1}^{n} a_i^{\varepsilon(i)} \text{ with } 0 \le \varepsilon(i) < p.$$

means that we can establish a mapping

$$\exp : G \rightarrow V(n,p)$$

between G and the n-dimensional vector space $V(n,p)$ over Z_p by defining

$$\prod_{i=1}^{n} a_i^{\varepsilon(i)} \mapsto (\varepsilon(1), \varepsilon(2), ..., \varepsilon(n)).$$

So the elements of the group can be represented as vectors. For example, in the quaternion group Q8 an element would be represented in a computer by an array

$\varepsilon(1)$	$\varepsilon(2)$	$\varepsilon(3)$

where $\varepsilon(1), \varepsilon(2), \varepsilon(3)$ take values 0 or 1.

The operations on elements are performed using word operations and collection.

- Multiply g and h by collecting $a_1^{\varepsilon(1)(g)} a_2^{\varepsilon(2)(g)} \cdots a_n^{\varepsilon(n)(g)} a_1^{\varepsilon(1)(h)} a_2^{\varepsilon(2)(h)} \cdots a_n^{\varepsilon(n)(h)}$;
- Invert g by collecting $a_n^{-\varepsilon(n)(g)} a_{n-1}^{-\varepsilon(n-1)(g)} \cdots a_1^{-\varepsilon(1)(g)}$;
- Test g and h for equality by comparing $(\varepsilon(1)(g), \varepsilon(2)(g), ..., \varepsilon(n)(g))$ and $(\varepsilon(1)(h), \varepsilon(2)(h), ..., \varepsilon(n)(h))$;
- Calculate the order $|g|$ of an element g as follows

```
order := 1;
for $i$ := 1 to $n$ do
  if $\varepsilon(i)(g) \neq 0$ then
    order := order $\times p$;    (* for soluble case use $p(i)$ *)
    $g := g^p$;
  end if;
end for;
```

Define the *leading coefficient* of an element g to be the first non-zero entry of its corresponding vector, and define the *leading index* to be the index of the leading coefficient in the vector. Denote these by $lc(g)$ and $li(g)$ respectively.

A subgroup H of G has an induced series defined by

$$H(c) = H \cap G(c) = H \cap \langle a_c, a_{c+1}, ..., a_n \rangle$$

where there may be duplicate terms in the series. We can eliminate these terms and determine a canonical set of generators for a subgroup H as follows.

Definition: A sequence $[h_1, h_2, ..., h_r]$ of elements of H is called a *canonical generating sequence* (cgs) relative to a fixed pcp of G if and only if

i. the sequence $h_1, h_2, ..., h_r$ of elements defines a prime-step subnormal series of H;

ii. $li(h_i) > li(h_j)$ for $i > j$;

iii. $lc(h_i) = 1$ for $i = 1, 2, ..., r$;

iv. $\exp(h_j)[li(h_i)] = 0$ for $i \neq j$.

(If only conditions (i) and (ii) hold, then the sequence is called a *generating sequence* (gs).)

This definition requires that the exponent vectors $\exp(h_i)$ form a row-echelonized matrix of a form like

$$\begin{bmatrix} \exp(h_1) \\ \exp(h_2) \\ \exp(h_3) \end{bmatrix} = \begin{bmatrix} 0\,1\,*\,*\,0\,*\,*\,0\,* \\ 0\,0\,0\,0\,1\,*\,*\,0\,* \\ 0\,0\,0\,0\,0\,0\,0\,1\,* \end{bmatrix}$$

Let $[h_1, h_2, ..., h_r]$ be a cgs of H. Define $H_i = \langle h_i, h_{i+1}, ..., h_r \rangle$ for $1 \leq i \leq r+1$. Then

1. for each $h \in H$ there exists a unique h_i with $li(h) = li(h_i)$; Furthermore, $h \in H_i$.
2. if $h, k \in H$ and $li(h) = li(k) = li(h_i)$ then $hH_{i+1} = kH_{i+1}$ if and only if $lc(h) = lc(k)$;
3. the cgs of H is uniquely determined with respect to a fixed pcp of G;
4. the order of H is p^r.

If the sequence is a gs then (1), (2) and (4) still hold.

The cgs can be formed by applying non-commutative Gaussian elimination to the vectors of an arbitrary generating set of H. The algorithm is similar to row-echelonizing a matrix over a finite field, however, our subgroup is closed under p-th powers and commutators of the generating set.

Algorithm 2 : Non-commutative Gaussian Elimination

```
Input: a set $W$ of elements of $G$;

Output: a cgs $\bar{h} = [h_1, h_2, ..., h_r]$ of $H = <W>$;

procedure reduce_and_insert( $g$ : element; var $\bar{h}$ : sequence of elements );
(* $\bar{h} = [h_1, h_2, ..., h_r]$ has the property that $li(h_i) < li(h_j)$ for $i < j$.
 The element $g$ is echelonized against $\bar{h}$ and its reduced form
 (possibly) inserted in the sequence. *)

begin (* reduce_and_insert *)
  for $i = 1$ to $r$ do
    if $li(g) = li(h_i)$ then
      $g := gh_i^{-\exp(g)[li(h_i)]}$; (* now $li(g) > li(h_i)$ *)
    else
      make a gap in the sequence between $h_{i-1}$ and $h_i$ for new $i$-th generator;
      insert $g$ as the new $i$-th generator;
    end;
  end;
  if $g \neq identity$ then append $g$ to $\bar{h}$; end;
end; (* reduce_and_insert *)

begin
  $\bar{h}$ := empty sequence;
  for each $g \in W$ do
    reduce_and_insert( $g$, $\bar{h}$ );
    for each $h$, $k$ in $\bar{h}$ do
      reduce_and_insert( $h^p$, $\bar{h}$ ); reduce_and_insert( $[h,k]$, $\bar{h}$ );
    end;
  end;

  for each $h$ in $\bar{h}$ with $lc(h) \neq 1$ do
    replace $h$ by $h^m$ where $lc(h^m) = 1$;
  end;

  for each $h$, $k$ in $\bar{h}$ with $li(k) > li(h)$ and $\exp(h)[li(k)] \neq 0$ do
    replace $h$ by $hk^{-\exp(h)[li(k)]}$;
  end;
end.
```

Let us consider the example of $H = <b_2b_4b_5, b_4b_6>$ in the group (G64). The first thing the algorithm does is to insert the first generator $b_2b_4b_5$ as the first entry h_1 of the cgs. We have the matrix

$$\begin{bmatrix} 0\,1\,0\,1\,1\,0 \end{bmatrix}$$

The algorithm then considers $h_1{}^2 = b_4 b_5$ and inserts it as h_2.

$$\begin{bmatrix} 0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,1\,1\,0 \end{bmatrix}$$

The commutator $[h_1,h_2]$ is the identity, but $h_2{}^2 = b_5$, which is inserted as h_3.

$$\begin{bmatrix} 0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,1\,0 \end{bmatrix}$$

The commutators $[h_1,h_3]$ and $[h_2,h_3]$ are the identity, as is $h_3{}^2$.

The algorithm now considers the second generator $b_4 b_6$. When inserting this element it is reduced against h_2 by forming $b_4 b_6 h_2 = b_6$. This is inserted as h_4.

$$\begin{bmatrix} 0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1 \end{bmatrix}$$

All commutators with h_4 are trivial, as is $h_4{}^2$, so we have an echelonized form of the subgroup generators satisfying conditions (1) and (2). We need to enforce conditions (3) and (4) to get the final form.

$$\begin{bmatrix} 0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1 \end{bmatrix}$$

The subgroup has order 2^4 and cgs $[b_2, b_4, b_5, b_6]$.

This linear algebra view of subgroups and straightforward application of non-commutative Gaussian elimination allow us to easily

- test membership of an element $g \in G$ in a subgroup H of G;
- form a cgs for the subgroup $<H,g>$ where $g \in G$;
- form a cgs for the subgroup $<H,K>$ where H and K are subgroups of G;
- test containment $K \leq H$ of subgroups of G;
- form the normal closure of a subgroup;
- form a commutator subgroup $[H,K]$ of subgroups of G, and hence various series of G such as the derived series and lower central series.

The reduction of an element $g \in G$ against a cgs of a subgroup H as in the procedure reduce_and_insert determines a canonical (left) coset representative of gH. If $g \in H$ it also determines the normal word for g with respect to the cgs. Thus a cgs for H determines a pcp for H, and if H is normal it determines a pcp for G/H.

Conjugacy Classes of Elements of p-Groups

Two elements h and k of G are *conjugate* if there exists and element $g \in G$ such that $g^{-1}hg = k$. This defines an equivalence relation and the equivalence classes are called *conjugacy classes*. The centralizer of an element h is defined as

$$C_G(h) = \{ c \mid c \in G, c^{-1}hc = h \}.$$

The elements in the conjugacy class of h are in one-to-one correspondence with the cosets $gC_G(h)$. The determination of the conjugacy classes and the corresponding centralizers is often an important precursor to other algorithms.

The algorithm which computes the conjugacy classes works down the chain of factor groups $G/G(i)$. The step of extending the class information of $G/G(i)$ to that of $G/G(i+1)$ relies on the fact that $G(i)/G(i+1)$ is central in $G/G(i+1)$. For each conjugacy class we store the information

- a representative h of the class, and
- a gs $[c_1, c_2, ..., c_m]$ for the centralizer $C_G(h)$.

Note that groups of order p or p^2 are abelian, so each element lies in a conjugacy class of its own, and its centralizer is the whole group. So in considering the inductive step from $G/G(i)$ to $G/G(i+1)$ we will assume the following information is given

1. a p-group $L = G/G(i)$ of order $p^{i-1} \geq p^3$,
2. a generator a_iG_{i+1} for the minimal normal subgroup $N = G(i)/G(i+1)$ of L, and
3. for each conjugacy class of the factor group L/N a representative and a gs of its centralizer in L/N.

The following result presents the justification for the inductive step.

Proposition : Let L be a p-group of order $\geq p^3$. Let N be a minimal normal subgroup of L and h an arbitrary element of L. Let K be the union of conjugates of the coset hN under L/N. Moreover, let $c_1, c_2, ..., c_{m+1} \in L$ such that $[c_1N, c_2N, ..., c_mN]$ is a gs of the centralizer $C_{L/N}(hN)$ and that $<c_{m+1}> = N$.

Then one of the following cases applies.

I. If $[h,c_j] = 1$ for $1 \leq j \leq m$, then K splits into p conjugacy classes under L with representatives h, hc_{m+1}, ..., hc_{m+1}^{p-1}, respectively. All these representatives have the same centralizer, and the elements $[c_1, c_2, ..., c_{m+1}]$ form a gs for it.

II. If $[h,c_j] \neq 1$ for some j, $1 \leq j \leq m$, then K consists of just the conjugates of h in L. Choose the largest such value of j. The for each l, $1 \leq l < j$, there is an integer $k(l)$ with $0 \leq k(l) < p$ such that $[h,c_l] = [h,c_j]^{k(l)}$. Let

$$c_l^* = \begin{cases} c_l c_j^{-k(l)} & \text{for } 1 \leq l < j, \\ c_{l+1} & \text{for} j \leq l \leq m. \end{cases}$$

Then the elements $c_1^*, c_2^*, ..., c_m^*$ form a gs for the centralizer $C_L(h)$. □

For example with the quaternion group of order 8 we begin by noting that $G/G(3)$ is abelian of order p^2 and has classes

	class representative	centralizer gs
(1)	$1G(3)$	$a_1G(3), a_2G(3)$
(2)	$a_1G(3)$	$a_1G(3), a_2G(3)$
(3)	$a_2G(3)$	$a_1G(3), a_2G(3)$
(4)	$a_1a_2G(3)$	$a_1G(3), a_2G(3)$

The classes of G are determined by one inductive step. For (1) we have case I, so we get two classes with representatives 1 and a_3 both with a_1, a_2, a_3 as the gs for their centralizer. For (2) we have case II with $j = 2$ and get one class with representative a_1 and gs a_1, a_3 for its centralizer. For (3) we have case II with $j = 1$ and get one class with representative a_2 and gs a_2, a_3 for its centralizer. For (4) we have case II with $j = 2$ and get one class with representative a_1a_2 and gs $a_1a_2a_3, a_3$ for its centralizer. The final result is

class representative	centralizer gs
1	a_1, a_2, a_3
a_3	a_1, a_2, a_3
a_1	a_1, a_3
a_2	a_2, a_3
a_1a_2	$a_1a_2a_3, a_3$

The algorithm is very efficient - quite capable of handling groups of order 2^{50}. A variation of the algorithm computes the centralizer of a given element.

Sylow Subgroups of Soluble Groups

Let G be a finite soluble group described by a conditioned pc presentation with generators $a_1, a_2, ..., a_n$, subnormal series

$$G = G(1) \geq G(2) \geq \cdots \geq G(n) \geq G(n+1) = < identity >,$$

where $G(i) = < a_i, a_{i+1}, ..., a_n >$, and normal series

$$G = N_1 \geq N_2 \geq \cdots \geq N_r \geq N_{r+1} = < identity >,$$

with elementary abelian quotients where $N_i = < a_{n(i)}, a_{n(i)+1}, \cdots, a_n >$ for some integer $n(i)$. Let $p(i) = |G(i) : G(i+1)|$ be prime.

Let p be a prime. The aim of this section is to present an algorithm to compute a Sylow p-subgroup S of the soluble group G. The algorithm recursively works down the normal series by computing a Sylow subgroup of G/N_2, G/N_3,..., G/N_r, G; and works up the subnormal series when extending a Sylow subgroup of G/N_i to one for G/N_{i+1}. Working up the subnormal series, the algorithm uses the cyclic extension technique: it finds an element $g \in G(i)-G(i+1)$ (when $p = p(i)$) which normalizes a Sylow subgroup S of $G(i+1)$ and such that $g^p \in S$. The element g is in the coset $a_iG(i+1)$; however, a_i itself while it does normalize $G(i+1)$ may not normalize S. Hence, the algorithm finds an element y which conjugates S^{a_i} to S and then chooses g to be a suitable power of a_iy. The element y exists as any two Sylow p-subgroups of $G(i+1)$ are conjugate. The element a_iy normalizes S.

Algorithm 3 : Sylow Subgroup of a Soluble Group

```
function pc_sylow( $G$ : group; $p$ : prime ) : subgroup;
(* Given a finite soluble group $G$ with a conditioned pc presentation,
and a prime $p$, return a Sylow $p$-subgroup of $G$ *)
begin (* pc_sylow *)
    if $p$ does not divide $|G|$ then
        result is <identity>;
    else
        let $S/N_r$ := pc_sylow( $G/N_r, p$ ) and let $[h_1, h_2, ..., h_t]$ be a cgs for $S$;
        if $N_r$ is a $p$-group then
            result is $S$;
        else
            $M$ := pc_sylow( $<h_2, h_3, ..., h_t>, p$ );
            if $p_{li(h_1)} \neq p$ then
                result is $M$;
            else
                $y$ := conj_elt( $G(li(h_1)+1), M^{h_1}, M, p$ );
                let $q$ be integer coprime to $p$ such that $|h_1 y| = p^m q$;
                result is $< (h_1 y)^q, M >$;
            end if;
        end if;
    end if;
end; (* pc_sylow *)

function conj_elt( $G, H, K$ : group; $p$ : prime ) : element;
(* Given a finite soluble group $G$ with a conditioned pc presentation,
a prime $p$, and Sylow $p$-subgroups $H = <h_1, h_2, ..., h_t>, K = <k_1, k_2, ..., k_t>$
of $G$ such that $H/N_r = K/N_r$, return an element $g \in N_r$ such that $H^g = K$ *)
begin (* conj_elt *)
    if ($N_r$ is a $p$-group) or ($H/N_r = N_r/N_r$) then
        result is identity;
    else
        $j := li(k_1)+1$; $L := <k_2, k_3, ..., k_t>$; (* $= K \cap G(j)$ *)
        $y$ := conj_elt( $G(j), H \cap G(j), L, p$ );
        $h := h_1{}^y$;
        if $h^{-1} k_1$ is a $p$-element then
            result is $y$;
        else
            let $|h^{-1} k_1| = p^m q$, where $q$ is coprime to $p$;
            find integers $a, b$ such that $ap^m \equiv 1 \bmod q$ and $b(-p) \equiv_b 1 \bmod q$;
            $x := (h^{-1} k_1)^{ap^m}$;        $z := \left[ x^h (x^2)^{h^2} ... (x^{p-1})^{h^{p-1}} \right]$;
            result is $yz$;
        end if;
    end if;
end; (* conj_elt *)
```

The element z in *conj_elt* is required to conjugate h to an element of the coset k_1L, so that it conjugates $<h, L>$ to $<k_1, L>$. Hence, we want an element $l \in L$ such that $z^{-1}hz = k_1l$, or equivalently $[h, z] = h^{-1}z^{-1}hz = h^{-1}k_1l$. The map $z \mapsto [h,z]$ is a linear transformation of N_r regarded as a vector space. The formula in *conj_elt* for z provides a solution to the equation $[h,z] \in h^{-1}k_1L$ (though we will not prove this fact).

As an example, consider the fourth group G of Chapter 10. The group G has degree 21 and is generated by

a=(1,8,9)(2,11,15)(3,10,12)(4,14,19)(5,16,17)(6,21,20)(7,13,18),
b=(9,18,20)(12,19,17), and
c=(10,21,11)(13,16,14).

It has order 27,783=$3^4 \times 7^3$. It is soluble. The following elements of G :

a_1=a=(1,8,9)(2,11,15)(3,10,12)(4,14,19)(5,16,17)(6,21,20)(7,13,18),
a_2=(2,3,6)(4,7,5),
a_3=(8,13,21)(10,14,16),
a_4=b=(9,18,20)(12,19,17),
a_5=(1,2,3,4,6,5,7),
a_6=(8,14,13,10,16,11,21), and
a_7=(9,19,18,12,17,15,20).

define an isomorphism between G and the group described by the pcp

$$\begin{aligned}< a_1, a_2, a_3, a_4, a_5, a_6, a_7 \mid\ & a_1^3 = a_2^3 = a_3^3 = a_4^3 = a_5^7 = a_6^7 = a_7^7 = 1,\\ & a_2a_1 = a_1a_3a_6^5,\ a_3a_1 = a_1a_4,\ a_4a_1 = a_1a_2a_5^6,\\ & a_5a_1 = a_1a_6^5,\ a_6a_1 = a_1a_7,\ a_7a_1 = a_1a_5^3,\\ & a_3a_2 = a_2a_3,\ a_4a_2 = a_2a_4,\ a_5a_2 = a_2a_5^2,\ a_6a_2 = a_2a_6,\ a_7a_2 = a_2a_7,\\ & a_4a_3 = a_3a_4,\ a_5a_3 = a_3a_5,\ a_6a_3 = a_3a_6^2,\ a_7a_3 = a_3a_7,\\ & a_5a_4 = a_4a_5,\ a_6a_4 = a_4a_6,\ a_7a_4 = a_4a_7^2,\\ & a_6a_5 = a_5a_6,\ a_7a_5 = a_5a_7,\ a_7a_6 = a_6a_7\ >.\end{aligned}$$

This is a conditioned pcp with normal series $N_1 = G$, $N_2 = <a_2,a_3,\ldots,a_7>$, $N_3 = <a_5,a_6,a_7>$, and $N_4 = <$*identity*$>$. N_3 is an elementary abelian group of order 7^3; N_2/N_3 is an elementary abelian group of order 3^3, and N_1/N_2 is a cyclic group of order 3.

For $p = 7$ the computation first calculates a Sylow 7-subgroup S/N_3 of G/N_3. The result is $S = N_3$ since G/N_3 has order 3^4. Then it returns the result N_3 as a Sylow 7-subgroup of G since N_3 is a 7-group.

For $p = 3$, the computation is more involved. Since G/N_3 is a 3-group, the call *pc_sylow*(G/N_3, 3) returns G/N_3, so $S = G$. N_3 is not a 3-group, so we calculate *pc_sylow*($<a_2,a_3,\ldots,a_7>$, 3), which is $M = <a_2,a3,a_4>$ of order 3^3. The index p_1 is 3, so we need to extend M to a Sylow 3-subgroup of G. The element h_1 is a_1, and M^{a_1} is $<a_2a_5^6,a_3a_6^5,a_4>$. We call *conj_elt* to find the element y conjugating M^{a_1} to M.

In *conj_elt*, the recursion works up the chain of subnormal subgroups considering the values $j = 4, 3, 2$. For $j = 4$, it decides that the identity conjugates $<a_4>$ to $<a_4>$. For $j = 3$, it decides that a_6^5 conjugates $<a_3a_6^5, L>$ to $<a_3, L>$, where $L = <a_4>$, because $h = a_3a_6^5$, $h^{-1}k_1 = a_6^2$ of order 7, $m = 0$, $q = 7$, $a = 1$, $b = 2$, $x = a_6^2$ and $z = a_6^5$. For $j = 2$, it decides that a_5^6

conjugates $<a_2a_5^6, L>$ to $<a_2, L>$, where $L = <a_3,a_4>$, because $h = a_2a_5^6$, $h^{-1}k_1 = a_5$ of order 7, $m = 0$, $q = 7$, $a = 1$, $b = 2$, $x = a_5$ and $z = a_5^6$. Hence, the element y conjugating M^{a_1} to M is $a_5^6a_6^5$.

So *pc_sylow* returns

$$< a_1a_5^6a_6^5, a_2, a_3, a_4 >$$

as the Sylow 3-subgroup of G. As a permutation,

$$a_1a_5^6a_6^5 = (1,11,15)(2,10,12)(3,14,19)(4,21,20)(5,13,18)(6,16,17)(7,8,9).$$

Summary

This chapter has defined the concepts associated with pc presentations for p-groups and soluble groups. It has briefly described a few algorithms based on pc presentations.

Exercises

(1/Easy) Determine all the conditioned pc presentations (as a finite soluble group) of the symmetries of the square.

(2/Easy) Determine all the conditioned pc presentations (as a finite p-group) of the symmetries of the square.

(3/Moderate) Consider the given pc presentation for S_4, the symmetric group of degree 4. Use a suitably modified form of the non-commutative Gaussian algorithm to compute a cgs of $<a_1, a_4>$, a cgs of $<a_2,a_1a_2^2a_4>$, and a cgs for $<a_2,a_3a_4>$.

(4/Easy) Compute the conjugacy classes of the symmetries of the square.

(5/Easy) Compute a Sylow 2-subgroup of S_4.

Bibliographical Remarks

The use of Cayley graphs and presentations to represent groups is widespread in combinatorial group theory. The correspondence between loops and relations is also well known. The classic books H.S.M. Coxeter and W.O.J. Moser, **Generators and Relations for Discrete Groups**, Springer-Verlag, Berlin, 1965, and W. Magnus, A. Karass, and D. Solitar, **Combinatorial Group Theory**, Interscience, New York, 1966 contain much more on Cayley graphs, relations, and presentations.

A vast range of algorithms for p-groups and soluble groups have been developed, implemented, and described by R. Laue, J. Neubüser, and U. Schoenwaelder, *"Algorithms for finite soluble groups and the SOGOS system"*, **Computational Group Theory**, M.D. Atkinson, (ed.), Academic Press, London, 1984, pp.105-135. They either use the approach of the conjugacy class algorithm of extending a result from $G/G(i)$ to $G/G(i+1)$, or consider the action of subgroups and compute appropriate orbits and stabilisers (as in the normalizer algorithm). This is a good paper to read as an introduction to the area.

Further work on algorithms for soluble groups and p-groups has been done by J.J. Cannon and C.R. Leedham-Green in forthcoming publications, and by Glasby, Slattery, Conlon, and Neubüser and Mecky in S.P. Glasby, *"Constructing normalisers in finite soluble groups"*, J. Symbolic Computation **5** (1988) 285-294; S.P. Glasby and M.C. Slattery, *"Computing* *intersections and normalizers in soluble groups"*, J. Symbolic Computation **9** (1990) 637-651; S.B. Conlon, *"Computing modular and projective character degrees of soluble groups"*, J. Symbolic Computation **9** (1990) 551-570; S.B. Conlon, *"Calculating characters of p-groups"*, J. Symbolic Computation **9** (1990) 535-550; M. Mecky and J. Neubüser, *"Some remarks on the computation of conjugacy classes of soluble groups"*, Bulletin Australian Math. Soc. **40** (1989) 281-292.

AG-systems for 2-groups were implemented by J. Neubüser, *"Bestimmung der Untergruppenverbände endlicher p-Gruppen auf einer programmgesteuerten elektronischen Dualmaschine"*, Numerische Mathematik **3** (1961) 271-278; and for general finite soluble groups by Lindenberg and Jürgensen in W. Lindenberg, *"Über eine Darstellung von Gruppenelementen in digitalen Rechenautomaten"*, Numerische Mathematik **4** (1962) 151-153; W. Lindenberg, *"Die Struktur eines Übersetzungsprogramm zur Multiplikation von Gruppenelementen in digitalen Rechenautomaten"*, Mitteilungen des Rhein-Westfälischen Institut für Instrumentalische Mathematik Bonn **2** (1963) 1-38; and H. Jürgensen, *"Calculation with the elements of a finite group given by generators and defining relations"*, **Computational Problems in Abstract Algebra**, J. Leech (ed.), Pergamon Press, Oxford, 1970, pp.47-57. There is no universal agreement on notation and terminology. We follow the terminology for pc presentations in M.F. Newman, *"Calculating presentations for certain kinds of quotient groups"*, **SYMSAC '76** (Proc. 1976 ACM Symp. on Symbolic and Algebraic Computation, Yorktown Heights, 1976), R.D. Jenks (ed.), ACM, New York, 1976, pp.2-8.

An introduction to rewriting is B. Buchberger and R. Loos, *"Algebraic simplification"*, **Computer Algebra: Symbolic and Algebraic Computation**, (2nd edition), B. Buchberger, G.E. Collins, R.G.K. Loos (editors), Springer Verlag, Wien, 1983, pp.11-43.

Collection has a long history, going back to theoretical work of P. Hall, *"A contribution to the theory of groups of prime-power order"*, Proceedings of the London Mathematical Society (2) **36** (1934) 29-95. The first implementations followed Hall and used collection to the left: H. Felsch, **Die Behandlung zweier gruppen-theoretischer Verfahren auf elektronischen Rechenmaschinen**, Diplomarbeit, Kiel, 1960; J.M. Campbell and W.J. Lamberth, *"Symbolic and numeric computation in group theory"*, Proceedings of the Third Australian Computer Conference (Canberra, 1966), Australian Trade Publications, Chippendale, Australia, 1967, pp.293-296; E. Czyzo, *"An attempt of mechanization of Hall collecting process"*, Algorytmy **9** (1972) 5-17; E. Czyzo, *"An automatization of the commutator calculus"*, Algorytmy **10** (1973) 23-34; I.D. Macdonald, *"A computer application to finite p-groups"*, Journal of the Australian Mathematical Society **17** (1974) 102-112; and W. Felsch, **Ein commutator collecting Algorithmus zur Bestimmung einer Zentralreihe einer endlichen p-Gruppe**, Staatsexamensarbeit, Aachen, 1974. In 1961, Neubüser discovered that collection from the right was superior in terms of space and time usage. This strategy was used in the implementations of Neubüser, Lindenberg, Jürgensen cited above, and by T.W. Sag and J.W. Wamsley, *"On computing the minimal number of defining relations for finite groups"*, Mathematics of Computation **27** (1973) 361-368; J.W. Wamsley, *"Computation in nilpotent groups (theory)"*, (Proceedings of the 2nd International Conference on the Theory of Finite Groups, Canberra, 1973), M.F. Newman (editor), Lecture Notes in Mathematics **372**, Springer-Verlag, Berlin, 1974, pp.691-700; V. Felsch, *"A machine independent implementation of a collection algorithm for multiplication of group elements"*, **SYMSAC '76** (Proc. 1976 ACM Symp. on Symbolic and Algebraic Computation, Yorktown Heights, 1976), R.D. Jenks (ed.), ACM, New York, 1976, pp.159-166; and G. Havas and T. Nicholson, *"Collection"*, **SYMSAC '76** (Proc. 1976 ACM Symp. on Symbolic and Algebraic Computation, Yorktown Heights, 1976), R.D. Jenks (ed.), ACM, New York, 1976, pp.9-14. Collection from the right was independently discovered by Wamsley in 1972 in association with the nilpotent quotient algorithm (NQA). An important variation to this strategy is *combinatorial collection* for *p*-groups introduced by G. Havas and T. Nicholson, *"Collection"*, **SYMSAC '76** (Proc. 1976 ACM Symp. on Symbolic and Algebraic Computation, Yorktown Heights, 1976), R.D. Jenks (ed.), ACM, New York, 1976, pp.9-14. Another useful idea is that of pre-compiling or pre-processing the presentation if many collections are to be performed. A detailed description of this approach is V. Felsch, *"A machine independent implementation of a collection algorithm for multiplication of group elements"*, **SYMSAC '76** (Proc. 1976 ACM Symp. on Symbolic and Algebraic Computation, Yorktown Heights, 1976), R.D. Jenks (ed.), ACM, New York, 1976, pp.159-166. Collection from the left is championed in the work of C.R. Leedham-Green and L.H. Soicher, *"Collection from the left and other strategies"*, J. Symbolic Computation **9** (1990) 665-675. They present empirical and theoretical evidence that it is the best strategy for general use. In connection with the NQA, their conclusion is supported by the empirical evidence of M.R. Vaughan-Lee, *"Collection from the left"*, J. Symbolic Computation **9** (1990) 725-733, which also contains a detailed description of an implementation of collection. The termination of the collection process and the concept of a collected ordering is discussed in C.C. Sims, *"Verifying nilpotence"*, J. Symbolic Computation **3** (1987) 231-247.

Echelonization and canonical generating sequences of subgroups are due to M.F. Newman in unpublished work in 1977. The first implementation and description appeared in R. Laue, J. Neubüser, and U. Schoenwaelder, *"Algorithms for finite soluble groups and the SOGOS system"*, **Computational Group Theory**, M.D. Atkinson, (ed.), Academic Press, London, 1984, pp.105-135.

The first application of canonical generating sequences in a top-down approach was the determination of the conjugacy classes of elements of a *p*-group by V. Felsch and J. Neubüser, *"An algorithm for the computation of conjugacy classes and centralizers in p-groups"*, **EUROSAM '79** (Proc. European Symp. on Symbolic and Algebraic Manipulation, Marseille, June 26-28, 1979), E.W. Ng (ed.), Lecture Notes in Computer Science **72**, Springer-Verlag, Berlin, 1979, 452-465. It was noted in the SOGOS paper of 1984 that the algorithm could be extended to finite soluble groups described by a pc presentation. This was later done as part of SOGOS in 1986 by M. Mecky and later refined by M. Mecky and J. Neubüser, *"Some remarks on the computation of conjugacy classes of soluble groups"*, Bulletin Australian Math. Soc. **40** (1989) 281-292. An algorithm for the classes of a soluble group was also implemented by J.J. Cannon and M.C. Slattery in 1987 in the Cayley system.

The work on conjugacy classes of elements can be generalized to determine the characters of the group (or at least the degrees of the characters). See S.B. Conlon, *"Computing modular and projective character degrees of soluble groups"*, J. Symbolic Computation **9** (1990) 551-570; S.B. Conlon, *"Calculating characters of p-groups"*, J. Symbolic Computation **9** (1990) 535-550.

The algorithm for computing Sylow subgroups of a soluble group and its proof of correctness is presented in S.P. Glasby, *"Constructing normalisers in finite soluble groups"*, J. Symbolic Computation **5** (1988) 285-294. The algorithm follows ideas of W.M. Kantor.

For a p-group given by an arbitrary presentation, the nilpotent quotient algorithm (NQA) will produce a pc presentation. In fact it does more than this: Given an arbitrary presentation of a group G, a prime p, and an integer bound c on the class (that is, the length of the lower exponent-p central series), it returns a pc presentation for the largest quotient P of G which is a p-group and has class at most c. The NQA is due to I.D. Macdonald, who in 1971 had an implementation of the NQA using collection to the left. His version of the algorithm worked down the lower central series and extended the quotient by elementary abelian steps. See I.D. Macdonald, *"A computer application to finite p-groups"*, Journal of the Australian Mathematical Society **17** (1974) 102-112. J.W. Wamsley noticed the benefits of collection from the right, and together with Bayes and Kautsky implemented the algorithm in 1973 - see J.W. Wamsley, *"Computation in nilpotent groups (theory)"*, (Proceedings of the 2nd International Conference on the Theory of Finite Groups, Canberra, 1973), M.F. Newman (editor), Lecture Notes in Mathematics **372**, Springer-Verlag, Berlin, 1974, pp.691-700; A.J. Bayes, J. Kautsky, J.W. Wamsley, *"Computation in nilpotent groups (application)"*, (Proceedings of the 2nd International Conference on the Theory of Finite Groups, Canberra, 1973), M.F. Newman (editor), Lecture Notes in Mathematics **372**, Springer-Verlag, Berlin, 1974, pp.82-89. The best implementation to date is due to W.A. Alford, G. Havas, and M.F. Newman. Havas and Newman introduced many ideas such as using the lower exponent-p-central series, and extending the quotient by steps of size p. This work is described in M.F. Newman, *"Calculating presentations for certain kinds of quotient groups"*, **SYMSAC '76** (Proc. 1976 ACM Symp. on Symbolic and Algebraic Computation, Yorktown Heights, 1976), R.D. Jenks (ed.), ACM, New York, 1976, pp.2-8, and G. Havas and M.F. Newman, *"Application of computers to questions like those of Burnside"*, **Burnside Groups**, Lecture Notes in Mathematics **806**, Springer-Verlag, Berlin, 1980, pp.211-230. The implementation incorporates contributions of M.R. Vaughan-Lee, *"An aspect of the nilpotent quotient algorithm"*, **Computational Group Theory**, M.D. Atkinson, (ed.), Academic Press, London, 1984, pp.75-83, on reducing the number of critical pairs that must be checked in order to ensure consistency - that is, list (C').

Some of the algorithms based on echelonization of subgroup generators are analogues of algorithms from linear algebra. One such is the (original) algorithm to compute intersections, which is an analogue of the algorithm for finding the intersection of vector subspaces (see H.G. Zimmer, **Computational Problems, Methods, and Results in Algebraic Number Theory**, Lecture Notes in Mathematics **268**, Springer-Verlag, Berlin, 1972, pp. 24).