© DISCRETE MATHEMATICS 5 (1973) 105–129. North-Holland Publishing Company

CONSTRUCTION OF DEFINING RELATORS FOR FINITE GROUPS*

John J. CANNON

Department of Pure Mathematics, University of Sydney, Sydney, N.S.W. 2006, Australia

Received 1 June 1972**

Abstract. Given a faithful representation of a group G of order up to 10^4, we describe an algorithm, based on the notion of the graph of G, for constructing a concise presentation for G. This technique may be generalized to give a semialgorithm which is usually successful in finding presentations for groups of order up to 10^6.

1. Introduction

Suppose G is a finite group for which a faithful permutation or matrix representation is known. A problem which often arises is to construct a set of defining relations (presentation) for G with respect to a given set of generators. For example, while it is a comparatively simple matter to show that the two permutations

$$a = (1\ \ 2),\ b = (1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7)$$

generate the symmetric group S_7 of order 5040, it is a much more difficult task to construct a corresponding set of defining relations, such as

$$a^2 = b^7 = (ab)^6 = (ab^{-1}ab)^3 = (ab^{-2}ab^2)^2 = I,$$

for the group.

Not only do we wish to be able to construct sets of defining relations from faithful permutation or matrix generators of G but we usually insist in addition that the resulting set of defining relations be concise both in the sense that the number of relations in the set be small and the actual

* This research was supported by a grant from the Australian Research Grants Committee.
** Original version received 1 December 1971.

relations be as uncomplicated as possible. No practical hand method of constructing defining relations for an arbitrary group appears to be known.

The only non-trivial application of computers to this problem appears to have been the use of programs implementing the Todd-Coxeter algorithm to check whether sets of relations holding in a group are indeed defining. (The Todd-Coxeter algorithm [6] determines the index of a subgroup H in a group G, given defining relations for G and a set of words generating H.) For some examples of this type of work see [6]. Otherwise, computers have been merely used to multiply permutations or matrices together to assist hand computations of defining relations. A sophisticated example of such a program has been described in [4].

In this paper, we describe an efficient machine algorithm for constructing fairly concise sets of defining relations for groups of order up to 10^8. Our method is based on the notion of the graph of a group modulo a subgroup (defined in Section 2). If the subgroup is taken as the identity, then a subset S of the complete set of circuits passing through any node of the graph will correspond to a concise set of defining relations for the group. The crux of our algorithm is an efficient technique for identifying this subset of circuits. Because of storage limitations this method is not directly applicable to groups of order greater than 10^4 and so in Section 6 we describe an inductive version of the method which can be applied to much larger groups but which may occasionally fail.

In [3, Chapter 3], a method of constructing defining relations is described which also makes use of the fact that relators correspond to circuits in the graph of the group over the identity. However, their method of constructing S involves topological techniques (such as symmetrically embedding a graph in a surface) which cannot be easily realized algorithmically.

2. Theory

We begin by defining the graph of a group modulo a subgroup and summarising some of its properties. Throughout this paper, we shall assume that the identity element is excluded from sets of generators $X = \{g_1, ..., g_r\}$ of a group. We shall write X^{-1} to denote the set $\{g_1^{-1}, ..., g_r^{-1}\}$.

Definition 1. Let H be a subgroup of the group G and suppose

$$G = Hc_1 + \dots + Hc_t,$$

where we shall always assume that $c_1 = I$, the identity of G. We associate a directed graph $\Gamma(G|H)$ with the group G, generating set $X = \{g_1, \dots, g_r\}$ for G and subgroup H of G as follows: With each coset Hc_i of H in G associate a node α_i of $\Gamma(G|H)$ so that there is a one-to-one correspondence between the cosets of H and the nodes of $\Gamma(G|H)$. Two nodes, α_i and α_j of $\Gamma(G|H)$ are joined by an edge ϵ_{ij} directed from α_i to α_j if and only if

$$Hc_i g_k = Hc_j, \text{ for some } g_k \in X.$$

The edge ϵ_{ij} is called a g_k-edge. The graph $\Gamma(G|H)$ is called the *graph of G modulo H.*

In the literature, this graph is sometimes called a *Schreier diagram.* If $H = \{I\}$, the graph is called a *Cayley diagram* or *colour group* in older works but we shall refer to it simply as the *graph of G* and denote it by $\Gamma(G)$. Basically, the graph of G is a very compact means of representing the multiplication table of G.

For convenience, we shall suppose that the node α_i of $\Gamma(G|H)$ corresponding to coset Hc_i is simply labelled by the integer i. Thus if i is any node of $\Gamma(G|H)$ and $\bar{w}$ is an element of G written as a word in the g's, we write $\pi(i, \bar{w})$ for the path in $\Gamma(G|H)$ beginning at node i and defined by $\bar{w}$. On the other hand if $\epsilon_1, \dots, \epsilon_t$ is the sequence of edges corresponding to some path π in $\Gamma(G|H)$ and if $s_j \in X \cup X^{-1}$ is the label of edge ϵ_j, we say that the word $s_1 \dots s_t$ is the *word corresponding to path* π.

We summarise a number of elementary properties of these graphs in the following lemma. In each case the proof is obvious.

Lemma 1. *Suppose that G is a group with generating set* $X = \{g_1, \dots, g_r\}$, *F is the free group on X and* $\Gamma(G|H)$ *is the graph of G modulo some subgroup H. Suppose further that under the natural homomorphism of F into G, the image of an element* $w \in F$ *is* $\bar{w}$.

(i) *For every node i in* $\Gamma(G|H)$ *and every word* $w \in F$, *the path* $\pi(i, \bar{w})$ *exists in* $\Gamma(G|H)$ *and is unique. Thus we denote the unique end node of the path* $\pi(i, \bar{w})$ *by* $(i)\bar{w}$.

(ii) *The graph $\Gamma(G|H)$ is connected.*

(iii) *If for any $w \in F$, $(1)\bar{w} = i$, then $\bar{w} \in Hc_i$. So any such $\bar{w}$ can be taken as our coset representative c_i.*

(iv) *If $\bar{w} = I$ is a relation holding in G and i is any node, then $(i)\bar{w} = i$, so that $\bar{w}$ determines a circuit at every node of $\Gamma(G|H)$. In particular, if*

$$R_1(g_1, ..., g_r) = ... = R_s(g_1, ..., g_r) = I$$

is a presentation for G, then each relator forms a circuit around each node of $\Gamma(G|H)$.

(v) *If $(i)\bar{w} = i$ then $c_i\bar{w}c_i^{-1} \in H$. In particular, if $H = \{I\}$, then $\bar{w} = I$,* i.e., *$\bar{w}$ is a relator.*

(vi) *If $H = \{I\}$, the sets of circuits at any nodes i and j are identical.*

A more extensive discussion of the properties of $\Gamma(G)$ together with examples of group graphs, may be found in [8].

We next recall the definition of fundamental circuit.

Definition 2. Let Γ be a graph having p edges and q nodes and suppose further that Δ is any spanning tree of Γ. An edge of Γ not in Δ is called a *chord.* The set of $p - q + 1$ circuits obtained by adding the $p - q + 1$ chords to Δ one at a time is called the *fundamental system of circuits* relative to Δ. A circuit in the fundamental system of circuits is called a *fundamental circuit.*

If A and B are two sets whose elements are edges of graph Γ, we define the *sum* $A + B$ of A and B to be the set

$$(A \cup B) - (A \cap B).$$

The *edge-disjoint union* of circuits means the union of a set of circuits having no common edges. It is well known that the set of all circuits and edge-disjoint unions of circuits in a graph Γ forms a vector space over GF(2), the so-called *circuit space* of Γ. Further, the fundamental system of circuits relative to a spanning tree is a basis for the circuit space of Γ (see [7]). This leads to the following important result [3]:

Lemma 2. *The relators corresponding to a fundamental system of circuits belonging to $\Gamma(G)$ are sufficient to define G.*

Let us call a circuit belonging to the fundamental system of circuits relative to spanning tree Δ of $\Gamma(G)$ a *Δ-circuit* and a relator corresponding to a Δ-circuit a *Δ-relator.* The complete set of Δ-relators will be called a *Δ-system* for G.

A Δ-system will contain $(r-1)|G|+1$ relators and, except in the case of the very smallest groups, will form a highly redundant set of defining relators. The heart of our technique for constructing non-redundant sets of defining relations is an algorithm for colouring all the edges of all those Δ-circuits in $\Gamma(G)$ whose corresponding Δ-relators can be deduced from a given set of Δ-relators.

Our strategy for constructing a non-redundant set D of defining relations for a finite group G thus becomes clear. First the graph $\Gamma(G)$ of G, relative to the given set of generators, is constructed and a spanning tree Δ for $\Gamma(G)$ found. Initially, we suppose that only those edges of Γ corresponding to the edges of Δ are coloured. Now each Δ-circuit γ is processed as follows:

(a) If some edge of γ is uncoloured we add the corresponding Δ-relator to D (initially D is empty). Now we colour all the edges of Δ-circuits corresponding to all Δ-relators implied by the relators of D by repeatedly applying the following *colouring rule* until no more edges of $\Gamma(G)$ can be coloured: If a circuit corresponding to any relator of D, starting at any node of $\Gamma(G)$, contains a single uncoloured edge, then colour this edge. When the colouring procedure has finished, we proceed to examine the next Δ-relator.

(b) If all the edges of γ are coloured we simply proceed to the next Δ-relator.

A circuit is said to be *coloured* if all its edges are coloured. If an s_j-edge is coloured, we automatically assume that the corresponding s_j^{-1}-edge is coloured. This means that if the circuit corresponding to relator R is coloured, then the circuit corresponding to R^{-1} is also coloured. We now establish that the above colouring rule will colour precisely those Δ-circuits corresponding to Δ-relators derivable from D. First, however, we shall make precise the notion of a relator being derivable from D.

Definition 3. Suppose $D = \{R_1, R_2, ..., R_m\}$ is a set of relators of a group G. A relator R is said to be *derivable* from D if it can be transformed into the identity by a finite number of applications of the following rules:

(i) Insert one of the relators $R_1, R_1^{-1}, R_2, R_2^{-1}, ..., R_m, R_m^{-1}$ or one of the trivial relators between any two consecutive symbols of R or at either end of R.

(ii) Delete one of the relators $R_1, R_1^{-1}, R_2, R_2^{-1}, ..., R_m, R_m^{-1}$ or one of the trivial relators if it forms a block of consecutive symbols in R.

Lemma 3. *Suppose that the coloured edges of $\Gamma(G)$ correspond either to the edges of the spanning tree Δ or to Δ-circuits corresponding to Δ-relators derivable from D. If R is a relator of D such that, for some node i of $\Gamma(G)$, the path $\pi(i, R)$ contains a single uncoloured edge ϵ, then the Δ-relator corresponding to the Δ-circuit including ϵ is derivable from D.*

Proof. Suppose $R = s_1 .... s_t$, $s_u \in X \cup X^{-1}$ for $u = 1, ..., t$ and that ϵ is an s_j-edge joining nodes i and k, being directed from i to k. Let $\pi(1, ps_jq)$ be the Δ-circuit containing ϵ with both the paths $\pi(1,p)$ and $\pi(k, q)$ coloured. We need to prove that ps_jq is a relator in G using only the relators of D.

Using rule (i) of Definition 3,

$$ps_jq = ps_{j-1}^{-1} .... s_1^{-1}s_1 .... s_{j-1}s_js_{j+1} .... s_ts_t^{-1} .... s_{j+1}^{-1}q$$

$$= ps_{j-1}^{-1} .... s_1^{-1}s_t^{-1} .... s_{j+1}^{-1}q = R', \text{ say.}$$

The second step is possible by rule (ii) of Definition 3, because $s_1 .... s_t$ is a relator of D. Since ϵ is the only uncoloured edge of $\pi(i, R)$, the circuit $\pi(1, ps_{j-1}^{-1} .... s_1^{-1}s_t^{-1} .... s_{j+1}^{-1}q)$ is coloured and so by assumption, R' is a relator derivable from D. Thus ps_jq is a relator derivable from D.

Lemma 4. *If all possible edges of $\Gamma(G)$ have been coloured by the colouring rule, using a set of relators D, then any Δ-relator whose Δ-circuit contains an uncoloured edge cannot be derived from D.*

Proof. Assume that the lemma is false and let R be a Δ-relator derivable from D which contains an uncoloured edge ϵ. This means that there is a finite chain of relators

$$R = R_1, R_2, ..., R_u = R',$$

where $R' \in D$ or is the identity, such that R_{i+1} is obtained from R_i by the application of one of the rules of Definition 3.

Corresponding to this chain of relators we have a chain of circuits beginning with $\pi(1, R)$ and ending with $\pi(1, R')$. Now the circuit $\pi(1, R)$ contains a single uncoloured edge, while all the edges of $\pi(1, R')$ are coloured. An application of rules (i) or (ii) to R_i has the effect on the circuit $\pi(1, R_i)$ of introducing or removing either a block of coloured edges or a pair of adjacent uncoloured edges (corresponding to the trivial relator $s_j s_j^{-1}$).

However, since $\pi(1, R)$ has one uncoloured edge while $\pi(1, R')$ has all its edges coloured, it is clear that we cannot get from $\pi(1, R)$ to $\pi(1, R')$ by the rules of Definition 3. So our assumption that R is derivable from D must be false.

3. Construction of the graph of a group modulo a subgroup

The algorithm for constructing defining relations depends upon the availability of efficient methods for constructing the group graph with respect to some generating set. In this section, we shall discuss the more general problem of constructing the graph $\Gamma(G|H)$ of G modulo some subgroup H as this shall be needed in Section 6.

We begin by stating a straightforward algorithm for constructing the edge table T of $\Gamma(G|H)$, simultaneously with a set of coset representatives for H in G, given a set of permutation or matrix generators $E(1), \ldots, E(r)$ for G. It is convenient to actually construct the *extended edge table* T which is the $|G:H| \times 2r$ array whose entries are defined by

$$T(i, j) = k \text{ if } (i)E(j) = k \qquad \text{for } j = 1, \ldots, r;$$

$$T(i, j) = l \text{ if } (i)E(j-r)^{-1} = l \qquad \text{for } j = r+1, \ldots, 2r.$$

Each row of T corresponds to a coset of H. In particular, row 1 of T corresponds to H itself.

Algorithm 1. Construct graph.

r – number of generators of G.

E – array whose elements are the given generators of G.

T – extended edge table.

F – a $[G:H]$-dimensional vector in which a set of canonical coset representatives for H will be stored. The definition of a canonical representative will be discussed below.

i and j index the rows and columns of T, respectively.

m – number of coset representatives generated so far.

(i) [Initialize] $i \leftarrow 0$; $F(1) \leftarrow I$ (the coset representative for coset H); $m \leftarrow 1$.

(ii) [Increment row count] $i \leftarrow i+1$. If $i > m$, exit (the construction of $\Gamma(G|H)$ is complete). Else $j \leftarrow 0$.

(iii) [Increment column count] $j \leftarrow j+1$. If $j > r$ go to (ii) (row finished). Else,

(iv) [Compute $T(i, j)$] Find the canonical representative x for coset $F(i)E(j)$. If for some l, $1 \leq l \leq m$, $x = F(l)$, then $T(i, j) \leftarrow l$, $T(l, r+j) \leftarrow i$ (inverse entry). Go to (iii). Otherwise, $m \leftarrow m+1$, $T(i, j) \leftarrow m$, $T(m, r+j) \leftarrow i$, $F(m) \leftarrow x$ (store new coset representative). Go to (iii).

The two critical operations in this algorithm are the determination of the canonical representative of a coset and the location of a canonical coset representative among the elements of F (the first two operations of step (iv)). Let us discuss the second of these first. A very efficient way of doing this lookup is to store the elements of F in a hash table [9].

A simple and effective means of computing the hash address is the following. Assuming that a group element is stored as a (right-justified) packed integer string spread over several machine words, the contents of the machine words containing a group element are simply multiplied together (ignoring overflow) to form a single length product. The hash address of the group element is then taken as the remainder when this product is divided by the size of the hash table. To avoid right to left zero propagation each intermediate product is shifted one place to the right. Provided that the hash table is never more than seventy percent full, it has been found that, on the average, less than two comparisons are necessary in order to locate an element in the hash table.

A solution to the first problem is not so easy and it is convenient to distinguish three situations. If H is the identity, then there is a single element in each coset so that there is no problem in this case. If H is not the identity but is small enough so that all its elements can be stored, the following procedure can be used. We suppose the elements of G, as represented in the machine, are ordered and if $H = \{x_1, ..., x_n\}$, take the smallest of the elements $x_1 g, ..., x_n g$ as the canonical representative for coset Hg. This is not quite as inefficient as it seems, for it is only necessary to form a short initial segment of many of the products $x_i g$ in order to rule them out as being the canonical representative. However, the technique cannot be used on groups of order greater than 10^6.

If G is represented as a permutation group, then we may use some ideas of Sims [11] to define a canonical coset representative which may be cheaply computed without storing the elements of H. Following Sims let G be a permutation group on the set Ω. A *base* for G is a sequence $Z = \alpha_1, ..., \alpha_t$ of points such that the only element of G fixing all of the α_i is the identity. Suppose $Z = \alpha_1, ..., \alpha_t$ is a base for G and let $G^{(i)}$ be the stabilizer of $\alpha_1, ..., \alpha_{i-1}$. Then $G^{(1)} = G$ and $G^{(t+1)} = I$. Let U_i be a set of right coset representatives for $G^{(i+1)}$ in $G^{(i)}$ and let X_i be a set of generators for $G^{(i)}$. If a set of generators for $G^{(i)}$ is known, it is a simple matter to write down a set of coset representatives U_i for $G^{(i+1)}$ in $G^{(i)}$. Then by Schreier's theorem [5], $G^{(i+1)}$ is generated by

$$\{u^{(i)}x^{(i)}\phi\,(u^{(i)}x^{(i)})^{-1} \mid u^{(i)} \in U_i, x^{(i)} \in X_i\},$$

where $\phi(g)$ is the choosen representative for the coset containing g.

A knowledge of the sets U_i enables us to write any permutation of G in a unique form. For if $g \in G$, there exists a $g_1 \in U_1$ such that gg_1^{-1} fixes α_1, a $g_2 \in U_2$ such that $gg_1^{-1}g_2^{-1}$ fixes α_2 and so on. Eventually we find elements $g_i \in U_i$, $i = 1, ..., t$ such that

$$gg_1^{-1}g_2^{-1} \dots g_t^{-1} = I,$$

i.e.,

$$(1) \qquad g = g_t g_{t-1} \dots g_1 .$$

We now return to our problem of assigning a cheaply computable canonical representative to the cosets of subgroup H in a permutation group G. The solution described below is due to Richardson. Let $\alpha_1, ..., \alpha_t$ be a base for H and assume that the elements of Ω are ordered in some manner. Using Sims' algorithm, the sets U_i are constructed for the stabilizer chain

$$H = G^{(1)} > G^{(2)} > \dots > G^{(t)} > G^{(t+1)} = I.$$

A permutation x of G is the canonical representative for coset Hg provided that it is the minimal element of Hg with respect to the ordering $\leq$ which is defined as follows: If $y, z \in G$, then $y \leq z$ if and only if there exists a k, $1 \leq k \leq t + 1$ such that $\alpha_i^y = \alpha_i^z$ for $i = 1, ..., k - 1$ and $\alpha_k^y < \alpha_k^z$ (if $k \leq t$). While $\leq$ is not a total ordering on G, it may be shown that it is a total ordering on any coset Hg of H in G.

The canonical representative x of coset Hg is easily calculated. Select h_1 to be an element of U_1 such that $\alpha_1^{h_1 g} \leq \alpha_1^{ug}$, for all $u \in U_1$. Similarly, select $h_2 \in U_2$ such that $\alpha_2^{h_2 h_1 g} \leq \alpha_2^{u h_1 g}$, for all $u \in U_2$, and continue in this way to eventually obtain the required canonical representative, $x = h_t h_{t-1} \ldots h_1 g$. Note that since $G^{(i)}$ stabilizes $\alpha_1, \ldots, \alpha_{i-1}$, we have

$$\alpha_j^{h_i \ldots h_1 g} = \alpha_j^{h_j \ldots h_1 g}, \text{ for } j = 1, \ldots, i-1.$$

This canonical representative is fairly cheap to compute as a typical base will seldom contain more than 10 letters. Using such a scheme it is possible to compute graphs in very large permutation groups indeed, the main limitation being the storage space need to store $\Gamma(G|H)$. Note that if we compute the sets U_i for G as well as H, we may write a canonical representative x in the G-canonical form of equation (1), so that it is then not necessary to actually store the canonical coset representatives for H in G (the F vector in Algorithm 1).

4. Construction of a minimal spanning tree

Our algorithms for constructing defining relations for a group G require that the spanning tree Δ for $\Gamma(G|H)$ be minimal in the sense that the path in Δ from node 1 to any node i be of minimum length. This is so that the relators we produce will be as short as possible. In this section, we describe a simple but fast algorithm for constructing a minimal spanning tree Δ for $\Gamma(G|H)$, given the extended edge table T for $\Gamma(G|H)$.

The distance between two nodes of a graph is defined to be the length of a path of minimum length joining them. The algorithm proceeds as follows. If Σ_j denotes the set of nodes of $\Gamma(G|H)$ at precisely distance j from node 1, suppose that a minimal tree Δ_{t-1} has been constructed containing the nodes of $\Sigma_0 \cup \Sigma_1 \cup \ldots \cup \Sigma_{t-1}$. Now Δ_{t-1} is extended to include the nodes of Σ_t by examining all nodes α adjacent to each node β of Σ_{t-1} and adding the edge joining α to β to the tree if α is not already in the tree.

The algorithm outputs Δ as a set of coset representatives stored recursively in an $n \times 3$ array W, where n is the number of rows of T. $W(p, 2)$ contains the number, q say, of a coset representative c_q and

$W(p, 3)$ contains the name, j say, of an element $s_j \in X \cup X^{-1}$ such that $c_p = c_q s_j$, where the length of the word c_q is less than the word c_p. in particular, c_1 is the identity. $W(p, 1)$ is a bookkeeping entry which links together the rows of W in the order in which they are generated. If $W(p, 1)$ is zero, then node p is not yet included in the spanning tree.

Algorithm 2. Construct a minimal spanning tree.

n – number of rows of T.

s – number of columns of T.

k – counts the number of coset representatives which have thus far been constructed.

i – current row of T being examined.

j – current column of T being examined.

l – links the rows of T together in the order in which they are to be examined.

(i) [Initialize] Zero the array W; $W(1, 2) \leftarrow 1$; $i \leftarrow 1$; $j \leftarrow 0$; $k \leftarrow 1$; $l \leftarrow 1$.

(ii) [Examine all nodes adjacent to node i] $j \leftarrow j + 1$. If $j > s$, go to (iv). (End of T row reached, so all nodes adjacent to i have been examined.) If $W(T(i, j), 1) \neq 0$ or $T(i, j) = l$, go to (ii). Node $T(i, j)$ is already in spanning tree.)

(iii) [Add node $T(i, j)$ to the spanning tree] $k \leftarrow k + 1$; $W(T(i, j), 2) \leftarrow i$; $W(T(i, j), 3) \leftarrow j$; $W(l, 1) \leftarrow T(i, j)$; $l \leftarrow T(i, j)$. Go to (ii).

(iv) [Check if finished] If $k = n$, finish. Else $i \leftarrow W(i, 1)$, $j \leftarrow 0$, and go to (ii).

5. Single stage presentation algorithm

We can now describe the basic algorithm for constructing a set of defining relators for a finite group G.

Algorithm 3. Single stage presentation algorithm.

(i) Construct the extended edge table T for $\Gamma(G)$, using Algorithm 1. While it is not theoretically necessary to use the extended edge table, its use by the minimal spanning tree algorithm leads to shorter words representing the group elements, and consequently to more concise relators.

(ii) Construct the array W containing a minimal spanning tree Δ for $\Gamma(G)$ using Algorithm 2. The $|G|$ words c_i corresponding to the set of paths of Δ beginning at the root (node 1) and ending at each node form a set of word representatives for the $|G|$ elements of G. (By construction, this set of words is actually a left Schreier system for I in G, although this fact is not explicitly used.) Thus the effect of Algorithm 2 is to set up the array W containing a set of word representatives for the elements of G.

Since Δ is a minimal spanning tree with respect to path length from 1 to each node, the words c_i will be as short as possible. Initially, all the edges of $\Gamma(G)$ are supposed uncoloured. Concurrently with the construction of Δ, we colour all those edges of $\Gamma(G)$ which are also edges of Δ, together with their inverses. (If ϵ is an s_i-edge joining nodes j and k, directed from j to k, then the edge inverse to ϵ is the s_i^{-1}-edge joining nodes j and k, directed from k to j. Recall that in the extended edge table an edge and its inverse have distinct representations).

Now the set of all remaining uncoloured edges of $\Gamma(G)$ (excluding the inverse edges) is precisely the set of chords of $\Gamma(G)$ relative to Δ, so that as each one of these edges is coloured a new fundamental circuit becomes coloured.

(iii) Construct a new relator. Let D denote a set of defining relators. Initially D is empty. We define the *distance* $d(\epsilon)$ *of an edge* ϵ of $\Gamma(G)$ from node 1 (relative to Δ) as follows. Suppose ϵ joins nodes j and k. Then $d(\epsilon)$ is defined to be the sum of the lengths of the words c_j and c_k. If ϵ is an uncoloured edge at minimum distance from node 1, then the word corresponding to the fundamental circuit which becomes coloured after ϵ is coloured is taken as the next relator to be added to D.

Specifically, suppose the edge ϵ is labelled s_i and joins nodes j and k, being directed from j to k. Let $c_j = s_{j_1} \,.....\, s_{j_p}$ and $c_k = s_{k_1} \,.....\, s_{k_q}$. Then we add relator

$$R = s_{j_1} \,.....\, s_{j_p} s_i s_{k_q}^{-1} \,.....\, s_{k_1}^{-1}$$

to D. By construction, R is a Δ-relator. The minimal property of Δ firstly ensures that R is the shortest possible relator which cannot be derived from the other relators of D, and secondly prevents the occurrence of expressions of the form $s_t s_t^{-1}$ in R.

If there are no remaining uncoloured edges of $\Gamma(G)$, then, by Lemmas 2 and 3, D comprises a set of defining relators for G and the algorithm terminates. Otherwise, each time a new relator is added to D we colour the edge ϵ and go to step (iv).

(iv) Derivation of the implications of the relators of D. Each time a new relator is added to D, we colour in edges of $\Gamma(G)$ to ensure that all fundamental circuits corresponding to all those Δ-relators derivable from D have all their edges coloured. This is done using the colouring rule of Section 2. If a circuit corresponding to any relator of D, beginning at any node of $\Gamma(G)$, contains a single uncoloured edge, then colour this edge. When no more edges of $\Gamma(G)$ can be coloured by repeated applications of this rule we return to step (iii). The use of this rule was justified in Lemmas 3 and 4. Methods of implementing this colouring rule are discussed below.

We now consider the efficiency of this algorithm. The construction of $\Gamma(G)$ involves $r|G|$ group element multiplications and not more than $2r|G|$ comparisons of group elements, where we assume that the hash table of elements is such that, on the average, less than two comparisons are required per lookup. The work involved in forming Δ and actually finding the relators is insignificant. The most critical part of the algorithm is the colouring-in procedure (step (iv)).

There are two obvious ways of doing this. The simplest method is to make a number of passes over $\Gamma(G)$, where each pass involves applying every relator of D to every node. The procedure terminates when no edges have been coloured during an entire pass. Much of the time no new edges are coloured during the second pass and it is rare for any more edges to be coloured during the third pass, so that if a is the average number of passes each time step (iv) is executed, we have $2 < a < 3$. If the total number of defining relators constructed is t, then the total number of relator cycles traced is easily seen to be

$$\tfrac{1}{2}at(t+1)|G| \sim t^2|G|.$$

This is the method that we have used in our implementation of the presentation algorithms.

A more complicated colouring-in procedure involves keeping a list C of those edges which become coloured as the result of constructing a new relator in step (iii). When step (iv) is entered, C contains the single edge ϵ which is coloured to give the new relator. Each edge in C is examined in turn to see if it forms part of some relator cycle containing a single uncoloured edge. Each such edge is coloured and added to C to await its turn for examination. It is not known how much more efficient this procedure is over the one described in the previous paragraph.

Coloured edges of $\Gamma(G)$ may be conveniently represented in the machine by simply negating the corresponding entry in array T. Thus if the s_i-edge ϵ joining nodes j and k is coloured, then $T(j, s_i)$ is set equal to $-k$.

The algorithms described in this paper have been implemented in ANSI Fortran (with an exception to be noted below) on the CDC6600 computer as part of the Sydney Group Theory System. The only machine dependent feature is a suite of routines (known collectively as the Stack Handler) through which all arrays are accessed. The Stack Handler provides dynamic storage allocation and the ability to process packed arrays whose field sizes are known only at run time. Packing is desirable for the components of a matrix or permutation representation of a group element, as these components usually only occupy a few bits each, and also for the elements of the array T. Run times quoted in this paper then refer to this Fortran implementation running on the CDC6600.

Examples 1. We give five examples of presentations constructed by Algorithm 3. Given the standard representations of the familiar small groups (e.g. $S_4, S_5, S_6, A_4, A_5, A_6$, $\mathrm{PSL}_2(7)$ etc.), the algorithm finds the usual presentations so that we shall only give one such example. In each case, we reproduce the relators in the order in which they were constructed by the algorithm.

(1) The symmetric group S_4 of order 24 generated by

$$a = (1\ \ 2)$$
$$b = (2\ \ 3\ \ 4)$$

is presented as

$$a^2 = b^3 = (ba)^4 = 1.$$

Execution time was 0.24 seconds of which 0.13 seconds was taken up by Algorithms 1 and 2.

(2) The simple group $\mathrm{PSL}_2(11)$ of order 660 generated by

$$a = (1\ \ 11\ \ 2\ \ 5\ \ 7)(10\ \ 3\ \ 6\ \ 4\ \ 9)$$
$$b = (2\ \ 11\ \ 3\ \ 10\ \ 9)(6\ \ 1\ \ 5\ \ 4\ \ 8)$$

is presented as

$$(ab)^2 = a^5 = b^5 = (ba^{-1})^6 = ba^2b^2a^{-1}ba^{-1}b^{-2}a^{-2}ba^{-1} = 1.$$

Execution time was 16 seconds of which 6 seconds was taken up by Algorithms 1 and 2.

(3) The symmetric group S_7 of order 5040 generated by

$$a = (1\ \ 2)$$
$$b = (1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7)$$

is presented as

$$a^2 = b^7 = (ba)^6 = (bab^{-1}a)^3 = (b^2ab^{-2}a)^2 = 1.$$

N.B. This presentation for S_7 has one less relator than the corresponding one given in Coxeter and Moser [3; p. 137]. Execution time was 128 seconds of which 43 seconds were taken up by Algorithms 1 and 2.

(4) The following two permutations (due to John McKay) generate the unitary simple group $U_3(3)$ of order 6048:

$$\begin{aligned} a = {} & (1\ 5\ 7\ 3\ 12\ 24\ 11)(2\ 23\ 4\ 27\ 13\ 14\ 26) \\ & (6\ 20\ 18\ 8\ 25\ 21\ 28)(9\ 10\ 17\ 15\ 22\ 16\ 19) \\ b = {} & (3\ 4)(5\ 17\ 7\ 16\ 8\ 20\ 6\ 13) \\ & (9\ 19\ 11\ 14\ 12\ 18\ 10\ 15) \\ & (21\ 23\ 26\ 28\ 24\ 22\ 27\ 25) \end{aligned}$$

This group is presented as

$$\begin{aligned} (ba^{-1})^3 = a^7 = b^8 &= ba^2baba^{-1}b^{-1}ab^2a^{-1} \\ &= bab^{-3}a^{-1}b^{-4}a^{-2} = 1. \end{aligned}$$

Execution time was 320 seconds of which 80 seconds were taken up by Algorithms 1 and 2. This example exhibits a behaviour occasionally observed when the algorithm is applied to larger groups where a large number of colouring passes (12 in this case) are required after the last relator is added to D. This suggests that one should perhaps modify Algorithm 3 slightly so that if more than three colouring passes are required after the addition of a relator of D, instead of doing the extra colouring passes, one should attempt to show that D is defining using the Todd-Coxeter algorithm. If that strategy had been used on this example, execution time would have been reduced to about 200 seconds.

(5) The following four permutations generate the Mathieu simple group M_{11} of order 7920 (see [11]).

$$\begin{aligned} a &= (2\ 6)(3\ 5)(4\ 7)(9\ 10) \\ b &= (1\ 5\ 7)(2\ 9\ 4)(3\ 8\ 10) \\ c &= (1\ 11)(2\ 7)(3\ 5)(4\ 6) \\ d &= (2\ 5)(3\ 6)(4\ 7)(11\ 12) \end{aligned}$$

The group is presented as

$$\begin{aligned} a^2 &= c^2 = d^2 = b^3 = (ca)^2 = (da)^2 = (dc)^3 = (db^{-1})^4 \\ &= (ba)^2\, db^{-1}ab^{-1}d = abab^{-1}adb^{-1}db^{-1} \\ &= abcbacb^{-1}cb^{-1} = 1. \end{aligned}$$

Execution time was 550 seconds of which 125 seconds were taken up by Algorithms 1 and 2. After the addition of the last relator 10 colouring passes were required, so that if the strategy suggested at the end of the last example were adopted, the execution time for this example would drop to about 370 seconds.

We may summarise the behaviour of the algorithm for this example in Table 1, where column 3 contains the total colouring time after the addition of the indicated relator to D.

Table 1

Relator added to D	No. of colouring passes	Colouring time in seconds
b^3	2	5.5
$(ba)^2$	2	10.0
$(da)^2$	2	14.4
$(dc)^3$	2	19.7
$(db^{-1})^4$	2	25.4
$(ba)^2db^{-1}ab^{-1}d$	2	31.0
$abab^{-1}adb^{-1}db^{-1}$	2	37.3
$abcbacb^{-1}cb^{-1}$	10	268.4

6. Two stage presentation algorithm

The algorithm of Section 5 is not directly applicable to groups much greater than 10,000 because of storage requirements. In addition, the execution time starts to become significant at this stage. So we consider the possibility of constructing a presentation for G from the graphs $\Gamma(G|H)$ and $\Gamma(H)$, where H is some subgroup of G, rather than from the graph $\Gamma(G)$. Since $\Gamma(G)$ is not always determined by $\Gamma(G|H)$ and $\Gamma(H)$, we cannot expect that such a procedure would work in every case. However, with a little care in the choice of H, the following semialgorithm will usually manage to construct a set of defining relators for G. We assume that we are given a set of permutation or matrix generators $\{g_1, ..., g_r\}$ for G and a set of words $\{h_1, ..., h_s\}$ in the g's, generating the subgroup H. Let $[G:H] = f$.

Algorithm 4. Two stage presentation algorithm.

(i) Using Algorithm 3, construct a presentation for H. Then rewrite these relators as words in the g's and call the set of rewritten relators D.

(ii) Using Algorithm 1, construct the extended edge table T for the graph $\Gamma(G|H)$.

(iii) Construct a minimal spanning tree Δ for $\Gamma(G|H)$ using Algorithm 2. This sets up the array W containing a set of coset representatives $\{c_1, ..., c_f\}$ for H in G.

(iv) Using the same colouring rule as in Algorithm 3, step (iv), colour edges of $\Gamma(G|H)$ according to the relators of D. It should be noted that, since fundamental circuits in $\Gamma(G|H)$ do not necessarily correspond to fundamental circuits in $\Gamma(G)$, this colouring rule may, on the one hand, miss colouring circuits in $\Gamma(G|H)$ corresponding to relators which can be derived from D and, on the other hand, it may colour circuits of $\Gamma(G|H)$ corresponding to relators which cannot be derived from D.

While it is possible to give fairly simple colouring rules which avoid colouring circuits corresponding to relators independent of D, such rules have the drawback that they often result in the generation of a large number of redundant relators. Colouring rules which manage to avoid both difficulties apparently have to be of considerably greater complexity than the one suggested here. The current simple-minded rule works quite well in practice so that we have not felt it necessary to undertake the complicated programming involved in a complicated colouring rule.

(v) Construct a new relator. As in Algorithm 3, step (iii), we look for an uncoloured edge ϵ at minimum distance from node 1. Suppose ϵ is labelled s_i and joins nodes j and k, being directed from j to k. Hence $c_j s_i c_k^{-1} \in H$. To get a relator from this, we use the given representation of G to obtain $c_j s_i c_k^{-1}$ as an element, h say, of the representation. Now there are two ways of expressing h as a word in the generators of H. In general, one saves the elements of H and the corresponding word table W_H, for $\Gamma(H)$ (storing them on disc between uses if necessary). Then one can simply look h up in the list of elements of H and read the corresponding word off the W_H array.

Alternatively, if G is represented as a permutation group, one may write h as a word in the generators of H using the canonical form given by equation 1 in Section 3. While this method requires the storage of little information about H it sometimes results in an unnecessarily complicated word for h.

Having found h as a word in the generators of H, it is immediately rewritten as a word, w say, in the generators of G. Suppose $c_j = s_{j_1} \,....\, s_{j_p}$, $c_k = s_{k_1} \,....\, s_{k_q}$ and $w = s_{m_1} \,....\, s_{m_u}$. Then the relation $c_j s_i c_k^{-1} w^{-1} = 1$ is written in the form

$$s_{j_1} \,....\, s_{j_p} s_i s_{k_q}^{-1} \,....\, s_{k_1}^{-1} s_{m_u}^{-1} \,....\, s_{m_1}^{-1}$$

and added to D. Then we return to step (iv). If all the edges of $\Gamma(G|H)$ are coloured, we go to the next step.

(vi) Using the Todd-Coxeter algorithm [6], determine if D is a set of defining relators for G. At this stage the relators of D may or may not be sufficient to define G. We check this by attempting to enumerate the cosets of H in the group defined by D using a Todd-Coxeter program [2]. If the program finds that H has index $[G:H]$ in this group, then the relators of D certainly define G. On the other hand, if the Todd-Coxeter program finds an index greater than $[G:H]$ or, if after a reasonable time has passed (determined by experience), it has not obtained any index, we assume that the relators of D are not defining.

The set of relators produced by Algorithm 4 will sometimes contain a small number of redundant relators. If desired, these may be removed with the aid of the Todd-Coxeter algorithm.

At this stage, little is known about how the embedding of H in G is correlated with the success or failure of Algorithm 4 to construct a

Table 2
Performance of the two-stage presentation algorithm

Group	Remarks	Order	H	$\|H\|$	$[G:H]$	No. of generators of G	No. of relators constructed	Success (√) or failure (X)
$14G1$	$\mathrm{PSL}_2(13)$	1092	$\langle a_{13}, b_{13}^2\rangle$	78	14	3	11	√
$12G2$	$\mathrm{PGL}_2(11)$	1320	$\langle a_{11}, b_{11}\rangle$	110	18	3	9	√
$8G5$		1344	$\langle a_7, c_7\rangle$	168	8	3	7	X
$10G7$		1440	$\langle a_9, c_9\rangle$	72	20	3	12	√
$16G13$		1920	$\langle a_{15}, a_{16}\rangle$	80	24	3	6	X
$14G2$	$\mathrm{PGL}_2(13)$	2184	$\langle a_{13}, b_{13}\rangle$	156	14	3	8	√
$16G15$		2880	$\langle a_{15}e_{15}, a_{16}\rangle$	240	12	3	6	X
$18G1$	$\mathrm{PSL}_2(17)$	2448	$\langle a_{17}, b_{17}^2\rangle$	136	18	3	8	√
$17G6$	$\mathrm{PSL}_2(16)$	4080	$\langle a_{15}e_{15}, a_{16}\rangle$	240	17	3	9	√
$18G2$	$\mathrm{PGL}_2(17)$	4896	$\langle a_{17}, b_{17}\rangle$	272	18	3	9	√
$13G7$	$\mathrm{PSL}_3(3)$	5616	$\langle a_{13}\rangle$	13	432	2	6	√
$16G16$		5760	$\langle a_{15}e_{15}, a_{16}\rangle$	240	24	3	6	X
$16G17$		5760	$\langle a_{15}, b_{15}\rangle$	360	16	3	12	√
$20G2$	$\mathrm{PGL}_2(19)$	6840	$\langle a_{19}, b_{19}\rangle$	342	20	3	9	√
$17G7$		8160	$\langle a_{15}e_{15}, a_{16}\rangle$	240	34	3	12	√
$12G3$	M_{11}	7920	$\langle a_{10}^2, b_{10}\rangle$	60	132	3	9	√
$16G18$		11,520	$\langle a_{15}, c_{15}\rangle$	720	16	3	11	√
$17G8$		16,320	$\langle a_{15}e_{15}, a_{16}\rangle$	240	68	3	13	√
$17G8$		16,320	$\langle a_{15}e_{15}, a_{16}, e_{16}\rangle$	960	17	3	15	√
$16G19$		40,320	$\langle b_{15}, d_{15}\rangle$	2520	16	3	11	√

presentation for G. In many situations where Algorithm 4 currently fails the reason is that, while $\Gamma(G|H)$ and $\Gamma(H)$ contain sufficient information to present G, much of this information is destroyed by the crude colouring rule used in step (iv). The main heuristic rule that emerges from use of the algorithm is that the larger the index of H in G the more likely the algorithm will be successful. Usually larger indexes also result in fewer redundant relators and more elegant relators. In Table 2, we summarise the performance of Algorithm 4 with respect to twenty presentations taken at random from Sim's table of primitive groups of degrees not exceeding 20 [11]. In most cases the subgroup H was taken to the stabilizer of a point so that $[G:H]$ is often small compared to $|H|$. The notation is the same as that used in [11].

If Algorithm 4 is unsuccessful, one simply chooses another subgroup K and tries again. If the relators found using K are also insufficient, one puts together the two sets of relators and, using the Todd-Coxeter program, tests whether this new set is defining. In practice, it has been found to be an extremely rare occurrence for these techniques to fail to present a group of order less than 10^6.

A useful feature incorporated in both the single stage and two stage presentation programs is the ability to input relators along with the generators of G. Not only does this enable one to construct presentations containing specified relators but it can often mean considerable savings in execution time. In the case of the two stage algorithm, user supplied relators (such as the orders of the generators) may help in the construction of a defining set.

The two stage algorithm is restricted by storage considerations to groups G and subgroups H such that $|H| < 10{,}000$ and $[G:H] < 10{,}000$. In the case of permutation groups, we have seen in Section 3 that it is possible to construct graphs $\Gamma(G|H)$ without storing the elements of H. Thus it is possible to present much larger groups by increasing the number of stages in Algorithm 4 and inducing a presentation for G up a chain of subgroups

$$I < H_1 < \ldots < H_t < G.$$

However, defining relators constructed by such an algorithm may become very long for the following two reasons: Firstly, unless the generators of the subgroups H_t can be given as very short words in the generators of G, the defining relators for the H_t will become lengthy when rewritten as words in the generators of G. Secondly, if as in step (v) of Algorithm 4, $c_j s_i c_k^{-1} w^{-1}$ is a relator constructed for H_t, then the word w may be excessivily long because the canonical form of equation (1) in Section 3 has to be used to obtain it.

Examples 2. We give five examples of presentations constructed using the two stage algorithm. In each case the relators up to the semicolon are those constructed for H.

(1) The group generated by

$$a = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13)$$
$$b = (2\ 3)(5\ 10)(7\ 11)(9\ 12)$$

is the simple group PSL_3 (3) of order 5616. Taking $H = \langle a \rangle$ we have $|H| = 13$, $[G:H] = 432$ and we get

$$a^{13} = 1;\ b^2 = (a^3b)^3 = [a, b]^3 = baba^2ba^{-2}ba^{-1}ba^{-2}ba^2$$
$$= ((ba)^2ba^{-3})^2 = 1.$$

Note that here $[G:H]$ is large relative to $|H|$.

(2) The group generated by

$$a = (1\ 15\ 7\ 5\ 12)(2\ 9\ 13\ 14\ 8)(3\ 6\ 10\ 11\ 4)$$
$$b = (1\ 7)(2\ 11)(13\ 12)(4\ 13)(5\ 10)(8\ 14)$$
$$c = (1\ 16)(2\ 3)(4\ 5)(6\ 7)(8\ 9)(10\ 11)(12\ 13)(14\ 15)$$

is a primitive group of order 11,520. Taking $H = \langle a, c \rangle$, we have $|H| = 720$, $[G:H] = 16$ and we get

$$b^2 = a^5 = [b, a]^3 = (bc)^6 = ((ba)^2(ba^{-1})^2)^2 = 1;$$
$$c^2 = ca^2b^{-1}cba^{-2} = cacba^{-1}ca^{-1}ca^2b^{-1}a^{-1}$$
$$= cbcab^{-1}a^{-1}b^{-1}cb^{-1}ab^{-1}a^{-1}b^{-1}$$
$$= cbca^{-1}b^{-1}aca^{-1}b^{-1}ab^{-1}$$
$$= ca^{-1}cbca^{-1}b^{-1}cb^{-1}ab^{-1}a = 1.$$

Here $[G:H]$ is small compared to $|H|$.

(3) Using the same four permutations for the Mathieu group M_{11} of order 7920, as in (5) of Examples 1, i.e.,

$$\begin{aligned}a&=(2\ \ 6)(3\ \ 5)(4\ \ 7)(9\ \ 10)\\ b&=(1\ \ 5\ \ 7)(2\ \ 9\ \ 4)(3\ \ 8\ \ 10)\\ c&=(1\ \ 11)(2\ \ 7)(3\ \ 5)(4\ \ 6)\\ d&=(2\ \ 5)(3\ \ 6)(4\ \ 7)(11\ \ 12)\end{aligned}$$

and taking $H = \langle a, b\rangle$ we have $|H| = 60$, $[G{:}H] = 132$, and we get

$$\begin{aligned}c^2 &= d^2 = cbcab^{-1}cb^{-1}ab = (dc)^3 = cb(da)^2b^{-1}c\\ &= cbdb^{-1}dbcbab(ab^{-1})^2 = 1;\ a^2 = b^3 = (ba)^5 = 1.\end{aligned}$$

Note that this is a more compact presentation than the one found using the single stage algorithm!

(4) The group generated by

$$\begin{aligned}a&=(1\ \ 4\ \ 5)(2\ \ 8\ \ 10)(3\ \ 12\ \ 15)(6\ \ 13\ \ 11)(7\ \ 9\ \ 14)\\ b&=(1\ \ 9\ \ 5\ \ 14\ \ 13\ \ 2\ \ 6)(3\ \ 15\ \ 4\ \ 7\ \ 8\ \ 12\ \ 11)\\ c&=(1\ \ 16)(2\ \ 3)(4\ \ 5)(6\ \ 7)(8\ \ 9)(10\ \ 11)(12\ \ 13)(14\ \ 15)\end{aligned}$$

is of order 40,320 (see [11]). Taking $H = \langle b, c\rangle$, we have $|H| = 2520$, $[G{:}H] = 16$ and we get

$$\begin{aligned}a^3 &= b^7 = (ba^{-1}b^{-1}a^{-1})^2 = (ab^2)^3 = (a^{-1}b)^5 = 1;\\ c^2 &= (ac)^3 = cb^2aca^{-1}b^{-2} = cba^{-1}b^{-1}a^{-1}cba^{-1}b^{-1}a^{-1}\\ &= ca^{-1}ba^{-1}b^{-1}ca^{-1}ba^{-1}b^{-1} = (cb^{-1})^2a^{-1}cab^2 = 1.\end{aligned}$$

Using the Todd-Coxeter algorithm the second last relator can be seen to be redundant and hence can be removed.

(5) The group generated by

$$\begin{aligned}a&=(1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ \ 11)\\ b&=(2\ \ 5)(4\ \ 6)(3\ \ 11)(7\ \ 8)\\ c&=(2\ \ 6\ \ 5\ \ 4)(3\ \ 7\ \ 11\ \ 8)\\ d&=(1\ \ 12)(2\ \ 3)(4\ \ 9)(5\ \ 7)(6\ \ 8)(10\ \ 11)\end{aligned}$$

is the Mathieu group M_{12} of order 95,040. Taking $H = \langle a, b\rangle$, we have $|H| = 660$, $[G{:}H] = 144$ and we get

$$b^2 = (a^{-1}b)^5 = a^{11} = (ba^2ba^{-2})^2 = (a^{-3}b)^3 = 1;$$

$$\begin{aligned} c^2b = d^2 &= dcdc^{-1}a^{-1}ba^2ba^{-1}b = (ca)^2c^{-1}ba \\ &= ca^{-2}ba^{-1}c^{-1}b(a^{-1}b)^2aba^{-1} = dada^{-2}dba^2ba^{-1}b \\ &= cda^{-1}ba^{-2}dc^{-1}(a^{-1}b)^2a^{-4}b = dabac^{-1}a^{-1}d(a^{-1}b)^2a^{-4} = 1. \end{aligned}$$

Execution time for this example was 30 seconds.

7. Applications

The availability of defining relations for a group G means that the Todd-Coxeter algorithm may be used to enumerate cosets of subgroups of G. This enables us to carry out certain investigations of G which may otherwise be very difficult. For example, it is then a simple matter to compute the transitive permutation representation of G afforded by the cosets of a specified subgroup K.

The techniques of this paper may be used to produce an economical generating set for G. Suppose X is a set of elements known to generate G and that X contains r elements. The first step is to construct a set of defining relators for G on the set X. Then, using the Todd-Coxeter algorithm, we enumerate the cosets in G of all those subgroups of G generated by $r-1$ element subsets of X. If any of these subgroups has index 1 in G, then the corresponding subset X' of X generates G. We may in turn apply this process to all the $r-2$ element subsets of X' and continue in this way until either a sufficiently small generating set is found or it is not possible to proceed further.

If the set X does not contain a sufficiently small subset generating G, one may use the Todd-Coxeter algorithm to test small sets of words in the elements of X for the property of generating G. It is usually easy to find two or three words generating G in this way. Having found a sufficiently small generating set for G, we construct defining relators for G in terms of these generators.

To illustrate this procedure, we construct a 2-generator presentation for Qd(5), of order 3000, which is the group obtained by allowing the special linear group SL(2, 5) to act in the natural way on the two dimensional vector space over GF(5). From the definition of Qd(5) and using the generators and defining relations for SL(2, 5) given in [3, Section 7.5] we obtain

$$\begin{aligned}\mathrm{Qd}(5) = \mathrm{gp}\,\langle s, v, t, z, x, y \mid\, & s^5 = 1,\ v^{-1}sv = s^{-1},\\ & v^2 = t^2 = (st)^3 = (tv)^2 = (s^2tv)^3 = z,\ z^2 = 1,\\ & x^5 = y^5 = x^{-1}y^{-1}xy = 1,\ s^{-1}xs = x,\ s^{-1}ys = xy,\\ & t^{-1}xt = y,\ t^{-1}yt = x^{-1},\ v^{-1}xv = x^2, v^{-1}yv = y^3\rangle.\end{aligned}$$

Using the Todd-Coxeter algorithm we find that Qd(5) is generated by $\{vx, st^{-1}\}$. Taking the faithful permutation representation of Qd(5) afforded by the cosets of $\langle s, v, t\rangle$ and applying Algorithm 3, we obtain

$$\mathrm{Qd}(5) = \mathrm{gp}\,\langle a, b \mid a^4 = b^3 = (ab)^5 = (a^2b)^2a^{-1}ba^{-2}b^{-1}a^{-1}b = 1\rangle,$$

where $a = vx$ and $b = st^{-1}$.

Generally, presentations produced by Algorithms 3 and 4 are close to ideal from the point of view of the Todd-Coxeter algorithm. Thus given a presentation of G which causes the Todd-Coxeter algorithm to perform badly, we may use Algorithm 3 to construct a better presentation, provided that it is possible to enumerate the cosets of the identity using the original presentation.

Programs [1, 10] for investigating the structure of groups of moderate order need fast methods for constructing subgroup normalizers. The best method currently known, which does not depend upon the way the group is represented, involves the use of the Todd-Coxeter algorithm and hence requires that a set of defining relations be known for the group. This requirement was the initial motivation for the development of the algorithms described in this paper. Now, whenever a set of generators for a group G is input to the suite of routines constituting the Sydney Group Theory System, a presentation is automatically constructed for G.

Finally, we note that the ability to input relators along with the generators of G enables us to construct presentations for G involving specific relators. This is particularly useful if one is trying to find a family of related presentations for a family of related groups.

8. Conclusion

The relatively old notion of the graph of a group can be used as the basis for a straightforward yet powerful algorithm for constructing defining relations. While it is impractical to apply the method to groups much larger than 10,000, one may introduce the graph of a group modulo a subgroup and apply the method inductively over two or more stages. This means that the method is applicable to groups of order up to at least 10^8 and possibly in some cases to much larger groups.

The availability of these algorithms has already lead to a variety of applications in such areas as group structure programs, investigation of the behaviour of the Todd-Coxeter algorithm, calculation of Schur multipliers of groups and the determination of families of similar presentations for families of related groups.

Acknowledgement

This work grew out of conversations with Lucien A. Dimino while the author was visiting Bell Telephone Laboratories in 1970. I would like to express my appreciation to Lou Dimino and also to those in the Computing Science Research Centre who made the visit possible.

References

[1] J.J. Cannon, Computing local structure of large finite groups, in: G. Birkhoff and M. Hall Jr., eds., Computers in algebra and number theory, SIAM-AMS Proc., vol. 3 (Am. Math. Soc., Providence, R.I., 1971) 161–176.
[2] J.J. Cannon, L.A. Dimino, G. Havas and J.M. Watson, Implementation and analysis of the Todd-Coxeter algorithm, Math. Comp., to appear.
[3] H.S.M. Coxeter and W.O.J. Moser, Generators and relations for discrete groups, second edition (Springer, Berlin, 1965).
[4] J. Grover. L. Rowe and L. Wilson, Applications of coset enumeration, in: Proc. 2nd symp. on symbolic and algebraic manipulation, 1971 (Association for Computing Machinery, New York, 1971).
[5] M. Hall, Theory of groups (Macmillan, New York, 1959).
[6] J. Leech, Coset enumeration on digital computers, Proc. Cambridge Philos. Soc. 59 (1963) 257–267.
[7] C.L. Liu, Introduction to combinatorial mathematics (McGraw-Hill, New York, 1968).
[8] W. Magnus, A. Karass and D. Solitar, Combinatorial group theory (Interscience, New York, 1966).
[9] R. Morris, Scatter storage techniques, Comm. ACM, 11 (Jan. 1968) 38–44.
[10] J. Neubüser, Computing moderately large groups: some methods and applications, in: G. Birkhoff and M. Hall, Jr., eds., Computers in algebra and number theory, SIAM-AMS Proc., vol. 3 (Am. Math. Soc., Providence, R.I., 1971) 183–190.
[11] C.C. Sims, Computational methods in the study of permutation groups, in: J. Leech, ed., Computational problems in abstract algebra (Pergamon, London, 1970) 169–183.