# Computational Group Theory

Charles C. Sims

October 5, 1998

Computational group theory has a history going back more than 80 years. It is a collaborative effort of researchers in a wide range of areas in both mathematics and computer science. The field has produced important algorithmic ideas and large software packages that have been used to obtain valuable results in mathematics and in other disciplines. The algorithmic achievements include the development of complicated algorithms designed to have the best possible asymptotic complexity and algorithms of exponential or perhaps unknown complexity that are nevertheless extremely useful in actual computations. There are also techniques for attempting to study instances of problems that are known not have have algorithmic solutions in general.

This article is intended as a survey of the field for the nonexpert. Thus it is written to provide a mathematician, physicist, or chemist who has encountered a particular group with guidance about the kinds of information one should expect to be able to determine about the group and in some cases to offer insight into the size of problems that can be attacked with current hardware. The article also is designed to show the computer scientist unfamiliar with the field the rich variety of problems concerning the complexity of group-theoretic algorithms.

The reader is introduced to several data structures that are unique to computational group theory. These include bases and strong generating sets for permutation groups and coset tables used to describe finitely generated subgroups of free groups. Other themes are the role of the classification of finite simple groups in the development of the subject and efforts to cope with the fact that many interesting problems do not have general algorithmic solutions.

A number of other surveys of computational group theory have appeared

over the years. This article draws on many of them, including [3], [8], [24], [35], and [15].

# 1 A crash course in group theory

The reader is assumed to have a knowledge of abstract algebra at roughly the level of an introductory undergraduate course. Thus the following terms will be used without definition: *group, commutative* or *abelian group, subgroup, normal subgroup, quotient group, generating set, cyclic group, homomorphism, isomorphism, permutation, even permutation, orbit, coset, Sylow subgroup, center, conjugacy class* of elements or of subgroups, *centralizer, normalizer,* and *simple group.*

We will also assume a basic familiarity with linear algebra, particularly linear algebra over finite fields. For each power $q$ of a prime, the field with $q$ elements will be denoted $\mathrm{GF}(q)$.

Many properties of groups are defined using series of subgroups. Let $G$ be a group and let

$$G = G_1 \supseteq G_2 \supseteq \cdots \supseteq G_k \supseteq G_{k+1} = 1$$

be a series of subgroups of $G$. The series said to be *subnormal* if $G_{i+1}$ is normal in $G_i$ for $1 \leq i \leq k$. A *composition series* is a subnormal series in which each quotient group $G_i/G_{i+1}$ is a (nontrivial) simple group. If $G$ has a composition series, the simple groups which occur are called the *composition factors* of $G$. They do not depend on the particular composition series. The group $G$ is *solvable* if it has a subnormal series in which the quotients $G_i/G_{i+1}$ are abelian. If $G$ has a subnormal series with cyclic quotients, then $G$ is said to be *polycyclic.*

The $G_i$ form a *normal series* if each is a normal subgroup of $G$. If in this case each quotient $G_i/G_{i+1}$ is in the center of $G/G_{i+1}$, then $G$ is *nilpotent.* All finitely generated nilpotent groups are polycyclic and all polycyclic groups are solvable. Every finite solvable group is polycyclic. If $p$ is a prime, then a $p$-group is a group in which all elements have orders which are powers of $p$. All finite $p$-groups are nilpotent.

Let $n$ be a positive integer. The *symmetric group*, which consists of all permutations of the set $\{1, 2, \ldots, n\}$, will be denoted $\Sigma_n$ and $A_n$ will denote the *alternating subgroup* of $\Sigma_n$, the set of all even permutations. If

$K$ is a field, then the *general linear group* $\mathrm{GL}(n, K)$ is the group of all $n$-by-$n$ invertible matrices with entries in $K$. If $K = \mathrm{GF}(q)$, then we write $\mathrm{GL}(n, q)$. The group $\mathrm{GL}(n, \mathbb{Z})$ is the subgroup of $\mathrm{GL}(n, \mathbb{Q})$ consisting of the invertible integer matrices whose inverses also have integer entries. These are the integer matrices with determinant $\pm 1$.

To every set $X$ there is associated the *free group* $F(X)$ on $X$. Intuitively, $F(X)$ is "the most general group generated by $X$". Formally we proceed as follows: Let $X^{-1}$ be a set disjoint from $X$ and having the same cardinality as $X$. Let $x \mapsto x^{-1}$ be a bijection of $X$ onto $X^{-1}$. Define $(x^{-1})^{-1}$ to be $x$. A word over $X$ is a finite sequence $u = u_1 \cdots u_r$, where the $u_i$ are in $X \cup X^{-1}$. The word $u^{-1}$ is defined to be $u_r^{-1} \cdots u_1^{-1}$. The word $u$ is reduced if it contains no subwords of the form $xx^{-1}$ or $x^{-1}x$. The product of two words is defined to be their concatenation. We may take $F(X)$ to be the set of reduced words. If $u$ and $v$ are reduced words, then there are unique words $a$, $b$, and $c$ such that $u = ac$, $v = c^{-1}b$ and $c$ has maximal length. The product of $u$ and $v$ in $F(X)$ is defined to be $ab$. The identity element of $F(X)$ is the empty word.

Here is an example with $X = \{x, y\}$: If $u = xyx^{-1}yxy^{-1}$ and $v = yx^{-1}y^{-1}x^{-1}yx$, then $a = xyx^{-1}$, $b = x^{-1}y$, and $c = yxy^{-1}$. Thus the product of $u$ and $v$ in $F(X)$ is $ab = xyx^{-1}x^{-1}y$.

Let $R$ be a subset of $F(X)$. The subgroup of $F(X)$ generated by $R$ may not be normal in $F(X)$. However, if we form the subgroup $N$ generated by all conjugates of elements of $R$, then $N$ is normal in $F(X)$. The group $G = F(X)/N$ is denoted $\langle X \mid R \rangle$. The pair $(X, R)$ is called a *presentation* for $G$ by generators and relators. The presentation is *finite* if both $X$ and $R$ are finite. If $X = \{x_1, \ldots, x_r\}$ and $R = \{U_1, \ldots, U_s\}$, then we write $G = \langle x_1, \ldots, x_r \mid U_1 = \cdots = U_s = 1 \rangle$. For example, the group $\langle x, y \mid x^2 = y^3 = (xy)^7 = 1 \rangle$ is the most general group generated by an element of order 2 and an element of order 3 such that the product of these elements has order 7.

# 2 Describing groups

Some computational questions in group theory refer to many groups at once. For example, in [29] it is shown that the answer to the question "How many nonisomorphic groups of order 256 are there?" is 56092. However, more commonly a single group $G$ is specified and questions about the structure of

$G$ are asked.

Computational group theory is divided into subfields based on the way the group $G$ is specified. The questions asked and the techniques used to answer them depend critically on the nature of the specification of $G$.

Here are four ways of specifying a group $G$.

1. One can define a finite subset $S$ of some previously specified group $M$ and say that $G$ is the subgroup of $M$ generated by $S$. Frequently $M$ is a symmetric group $\Sigma_n$, a general linear group $\mathrm{GL}(n, K)$, or a free group $F(X)$ for some finite set $X$. In this context, the first problem one encounters is the *membership problem*: Given an element $g$ of $M$, is $g$ in $G$?

2. One can define $G$ by a finite presentation.

3. One can define $G$ to be the group of all automorphisms of some algebraic or combinatorial structure. Possible combinatorial structures are graphs and block designs. Galois groups are automorphism groups of field extensions.

4. One can give a *black box* description of $G$. Black box descriptions are an abstraction of the minimal information one would need to begin computing in a finite group. They arise in algorithms for studying a finite group given by a set of generators. A black box description involves the following:

   - A positive integer $N$, which is assumed to be $O(\log |G|)$.
   - An encoding of elements of $G$ by bit strings of length $N$. By this is meant a bijection between $G$ and a subset of the bit strings of length $N$. The details of the encoding are "hidden". In particular, the set of bit strings which are encodings of elements of $G$ need not be known initially. All that one knows is that $|G| \leq 2^N$.
   - Two oracles or "black boxes". Let $U$ and $V$ be the bit strings corresponding to two elements $g$ and $h$ of $G$. Given $U$, the first oracle returns the bit string for $g^{-1}$. Given $U$ and $V$, the second oracle returns the bit string for $gh$.
   - The bit strings corresponding to a generating set for $G$. (Note that without these generators there would be no way to begin

computing in $G$. Using the two oracles, one can start with the generators and construct more elements of $G$ as products and inverses of previously obtained elements.)

When the way $G$ is specified makes it obvious that $G$ is finite, then the existence of algorithms for studying properties of $G$ is usually not an issue. The question becomes how hard is it to determine the answer to a particular question. This is not literally true, however. For example, one might ask, of the homomorphisms of $G$ into some infinite group $H$, which ones are essentially different in some sense. Since there might be infinitely many such homomorphisms, it is not clear that there is a finite description of the answer.

Some specifications define infinite groups or groups for which it is not immediately clear whether they are finite or infinite. In this case, the first issue is: Does there exist an algorithm to answer a particular question? If the answer is no, then discussions of practicality or efficiency are irrelevant.

It is now known that there are natural computational questions about groups which can not be answered algorithmically. There may be instances of the questions which can be answered, but there is no algorithm which is guaranteed to provide an answer in every case. In fact, most computational questions about finitely presented groups turn out not to have algorithmic solutions.

Except in groups given by finite presentations, questions about individual elements of a group tend not to be particularly challenging. Frequently one asks questions involving large sets of elements, either in the statement of the question or in its answer. For example, elements $g$ and $h$ of $G$ may be given and we want to determine all the elements of $G$ which commute with $g$ and to describe the elements which conjugate $g$ to $h$. The answer to the first question is the centralizer $C$ of $g$ in $G$, which is a subgroup of $G$. The answer to the second question is either the empty set or a coset of $C$. For each method of describing a group it is useful to have data structures which permit the efficient representation of subgroups.

# 3 A brief history

The definition of a group was not formalized until late in the 19th century. The first algorithmic questions about groups appeared almost immediately. The subject of computational group theory is generally considered to have

originated with three questions posed in [11] by Dehn in 1911. All three concern groups defined by finite presentations $(X, R)$. Elements of such a group are described by words over $X$.

**The Word Problem.** Let $G = \langle X|R\rangle$ and let $U$ be a word over $X$. Does $U$ represent the identity element of $G$?

**The Conjugacy Problem.** Let $G = \langle X|R\rangle$ and let $U$ and $V$ be words over $X$. Do $U$ and $V$ represent conjugate elements of $G$?

**The Isomorphism Problem.** Let $G = \langle X|R\rangle$ and $H = \langle Y|S\rangle$. Are $G$ and $H$ isomorphic groups?

All three of Dehn's questions turned out not to have algorithmic solutions. The word problem was shown undecidable by Novikov in [28]. A finite presentation $(X, R)$ has been constructed for which the word problem is undecidable. The word problem is problem is the special case of the conjugacy problem in which the word $V$ is taken to be empty. Thus the conjugacy problem is undecidable too. The isomorphism problem was shown to be undecidable in [32]. Even the case in which $H$ is a group of order 1 is undecidable.

Many other properties of finitely presented groups have been shown to be undecidable. For example, it is not possible to determine in general whether $G = \langle X|R\rangle$ is finite, nilpotent, or solvable.

Dehn showed the word problem does have a solution for a class of presentations which arise naturally in topology. The algorithm given by Dehn is now known to work for large classes of "small cancellation groups", groups in which distinct relators have no long subwords in common. With certain fairly natural probability measures on presentations, a random presentation turns out to define a small cancellation group and hence to have a solvable word problem. Unfortunately, the presentations which arise in many areas of research are not random.

One of the early positive contributions to computational group theory was [37]. That paper describes a technique now known as coset enumeration. Coset enumeration is one of the most used tools for studying finitely presented groups. It is discussed below.

The first digital computers were used primarily for numerical computation. However, in the mid-1940's Alan Turing suggested that a proposed new computer might be used to answer questions in group theory.

Attempts to implement coset enumeration began in the early 1950's, making coset enumeration one of the first nonnumerical procedures to be programmed. Unfortunately, computer memories of the time were too small to permit machines to go beyond what human experts could do by hand.

Perhaps the first mathematician to make computational group theory the primary focus of his professional activity was Joachim Neubüser. His first paper with a computational content [23] appeared in 1960.

The first conference to include a substantial portion of computational group theory was held in Oxford in 1967. Its proceedings [17] is the first volume to contain a collection of papers in this field. The first conference to be devoted entirely to computational group theory was held in Durham, England, in 1982. The proceedings [2] of that conference is still a useful reference.

The most important result of the 20th century in finite group theory is the classification of finite simple groups, which was completed in the early 1980's. The classification has had an impact on computational group theory in two important ways.

Beginning in the mid 1960's, new simple groups were discovered at fairly regular intervals. The existence of these large groups was difficult to demonstrate. Efforts to provide computer proofs of the existence of new simple groups provided strong motivation to improve algorithmic techniques, particular techniques for working with permutation groups.

Early algorithms for studying finite groups used relatively elementary theoretical results. However, once the classification of finite simple groups was finished, a new generation of algorithms began to appear which required the classification in their proofs of correctness or in the analysis of their complexity. These include many of the algorithms for studying finite permutation groups which have the best asymptotic complexity.

As important as the classification of simple groups is, there are large parts of finite group theory which are unaffected by it. In a certain sense most finite groups are $p$-groups, in fact, groups of order a power of 2. Finite $p$-groups lend themselves to computation more readily than general finite groups. Around 1970 there began line of algorithmic development related to finite $p$-groups. These techniques now constitute another major tool.

Until the 1970's, most research in computational group theory was undertaken by individuals trained as group theorists. This changed dramatically with the appearance of [21], which demonstrated the close connection between permutation group algorithms and efforts to solve the graph isomor-

phism problem, one of the most famous problems in theoretical computer science. Until computer scientists became actively involved, the primary focus of research in computational group theory consisted of efforts to solve specific problems with the then available hardware. Issues of complexity were not usually considered.

Computer scientists brought a new perspective to the subject. They insisted that researchers specify carefully what they meant by good algorithms. With their distinct point of view they constructed new data structures and algorithms with goal of obtaining better asymptotic complexity. It frequently turned out that these structures and algorithms provided practical improvements for people interested in actual machine computation. A health rivalry continues between the the two "camps" within computational group theory. Two conferences have been held at DIMACS with the specific purpose of encouraging dialogues between group theorists and computer scientists about algorithmic questions. The proceedings of these conferences [13, 14] make interesting reading. Another conference, sponsered by jointly by DIMACS and the Geometry Center, also produced a useful proceedings [1].

Two other major collections of papers on computational group theory were published as special issues [9] and [10] of the Journal of Symbolic Computation.

# 4 Permutation groups

The algorithmic theory of permutation groups is one of the most developed areas of computational group theory. Space does not permit giving here either a complete discussion of the available algorithms or a complete bibliography. An elementary introduction to the subject can be found in [7]. The survey [15] provides a good overview of recent results. A more complete treatment [34] is in preparation.

One of the first lessons learned about machine computation with permutation groups is that, although cycle notation is useful for hand computation, it is not a good representation for elements of $\Sigma_n$ in a computer. In most cases, an element $g$ of $\Sigma_n$ is represented by a vector of length $n$ whose $i$-th component is the image $i^g$ of $i$ under $g$. Thus the number of bits need to describe our element $g$ is $n \log n$. However, it is traditional to consider $n$ to be the size of a permutation.

The first question one normally asks about a permutation group $G$ is

whether or not $G$ is transitive or, more generally, what the orbits of $G$ are. Finding the orbit , of $G$ containing a point $\alpha$ is easy, but it is basic. One usually needs to construct not only , but also, for each $\gamma$ in , , an element $v(\gamma)$ in $G$ such that the image of $\alpha$ under $v(\gamma)$ is $\gamma$. Suppose $G$ is given by a generating set $S$. Then we initialize the orbit , to $\{\alpha\}$ and set $v(\alpha)$ equal to the identity. Then we consider the elements $\beta$ of , in the order they are discovered and for each $x$ in $S \cup S^{-1}$ we compute $\gamma = \beta^x$. If $\gamma$ is not already known to be in , , we add $\gamma$ to , and define $v(\gamma)$ to be $v(\beta)x$. An easy but fundamental fact is that $|, |$ is the index $|G : G_\alpha|$ of the stabilizer $G_\alpha$ of $\alpha$ in $G$.

If $G$ has only one orbit, then $G$ is *transitive*. In this case we can ask if $G$ is imprimitive. That is, we can ask if there is a nontrivial equivalence relation on $\Omega_n$ which is preserved by $G$. This question can be answered in time $O(n^2)$.

Subgroups of $\Sigma_n$ can have orders which are exponential in $n$. Thus we can not solve the membership problem in polynomial time for a subgroup $G$ of $\Sigma_n$ by listing the elements of $G$. A better data structure is needed.

A *base* for $G$ is a sequence $B = \beta_1, \ldots, \beta_r$ of elements of $\Omega_n$ such that the only element of $G$ fixing each $\beta_i$ is the identity. An element of $G$ is determined by what it does to $B$. If $G$ contains $A_n$, then any base for $G$ must have at least $n - 2$ elements. However, many interesting groups have bases of length $O(\log^c n)$ for small values of $c$.

The group $\Sigma_n$ induces a permutation group $T(n)$ on the set , of two-element subsets of $\Omega_n$. If $n = 3m$, where $m$ is an integer, then the following sequence of two-element sets is a base for $T(n)$:

$$C = \{1,2\},\ \{2,3\},\ \{4,5\},\ \{5,6\},\ \ldots,\ \{3m-2,3m-1\},\ \{3m-1,3m\}.$$

If an element $g$ of $\Sigma_n$ fixes the sets $\{1,2\}$ and $\{2,3\}$, then it fixes 1, 2, and 3. Thus if $g$ all $2m$ of these sets, $g$ must be the identity. Small modifications of this sequence form bases in the case that $n$ is not congruent to 0 modulo 3.

Fix a base $B = \beta_1, \ldots, \beta_r$ for a permutation group $G$. For $1 \leq i \leq r+1$ let $G^{(i)}$ denote the subgroup of $G$ fixing $\beta_1, \ldots, \beta_{i-1}$. Then

$$G^{(1)} = G \qquad \text{and} \qquad G^{(r+1)} = 1.$$

For $1 \leq i \leq r$ let $\Delta_i$ be the orbit of $G^{(i)}$ containing $\beta_i$ and let $U_i$ be a set of right coset representatives for $G^{(i+1)}$ in $G^{(i)}$. Since $G^{(i+1)}$ is the stabilizer of $\beta_i$ in $G^{(i)}$, it follows that $|\Delta_i| = |G^{(i)} : G^{(i+1)}|$ and that every element of $G$

can be uniquely written in the form $u_r \cdots u_1$, where $u_i$ is in $U_i$. The $\Delta_i$ are referred to as the *basic orbits* relative to $B$. If we know the basic orbits and sets $U_i$, we can decide membership in $G$ using a straightforward algorithm which runs in time $O(n^2)$.

If we take $G$ to be the group $T(3m)$ on , with the base $C$, then $r = 2m$ and $\Delta_1$ is all of , . The set $\Delta_2$ is the set of $2(n-2)$ two-element sets which intersect $\{1, 2\}$ in one point.

Suppose that we are given a generating set $S$ for our group $G$ with base $B$. Then it is easy to find $\Delta_1$. However, finding the other $\Delta_i$ is more difficult, since we do not immediately know generating sets for the subgroups $G^{(i)}$ with $i > 1$. If $S$ contains generators for all the subgroups $G^{(i)}$, then we say $S$ is a *strong generating set* relative to $B$.

We can get generators for the stabilizers of points using the following theorem.

**Theorem 1.** *Let $\Delta$ be an orbit of a permutation group $G$, let $\delta$ be a point in $\Delta$ and for each $\gamma$ in $\Delta$ let $v(\gamma)$ be an element of $G$ taking $\delta$ to $\gamma$. Suppose that $G$ is generated by a set $S$. Then $G_\delta$ is generated by*

$$T = \{v(\gamma)sv(\gamma^s)^{-1} \mid \gamma \in \Delta,\ s \in S\}.$$

The generating set $T$ in Theorem 1 can be much larger than $S$. To use this theorem iteratively, we must show that a large generating set for a permutation group can be "boiled down" to one of manageable size. A permutation group on a set with $n$ elements can always be generated by fewer than $n$ elements. Here we exhibit an algorithm for producing a generating set of size $O(n^2)$. The algorithm uses variables $u_{ij}$ for $1 \le i < j \le n$. Either $u_{ij}$ is nil or it is a permutation taking $i$ to $j$.

```
Procecure BOIL(S)
  Initialize all u_ij to nil.
  For g in S do
    h := g;
    For i from 1 to n − 1 do
      j := i^h;
      If j > i then
        If u_ij is nil then u_ij := h; break;
        else h := h u_ij^{-1}; fi;
      fi;
```

od; od;
Return the set of $u_{ij}$ which are not nil;
end.

If $T$ is BOIL($S$), then $S$ and $T$ generate the same group and $|T|$ is $O(n^2)$. The running time for BOIL is $O(|S|n^2)$.

Suppose we are given a generating set $S$ for a permutation group $G$ on $\Omega_n$. Assume that $|S|$ is $O(n^2)$. The sequence $B = 1, 2, \ldots, n-1$ is a base for $G$. We can obtain the basic orbits $\Delta_i$ and generating sets $S_i$ for the subgroups $G^{(i)}$ by defining $S_1$ to be $S$, computing $\Delta_1$ as sketched above, and computing a generating set $T$ for $G^{(2)} = (G^{(1)})_1$ using Theorem 1. Then $S_2$ is taken to be BOIL($T$). Since $|S_2|$ is $O(n^2)$, we can proceed inductively.

The time of the $i$-th iteration of this procedure is dominated by the time needed to apply BOIL. The initial generating set $T$ for $G^{(i+1)}$ has size $O(n^3)$. Thus applying BOIL takes time $O(n^5)$. Since $n-1$ iterations may be needed, the time needed to compute the strong generating set is $O(n^6)$.

It is not hard to improve this approach to $O(n^5)$, but "breaking the exponent 5 barrier" has been possible only using the classification of finite simple groups. The best deterministic algorithm known runs in time $O(n^4 \log^c n)$ for some constant $c$. Details, along with historical information, can be found in [34].

The Rubik Cube group is a permutation group on 48 points. Showing that the order of this group is 43252003274489856000 takes practically no time at all. For groups with relatively short bases, it is now possible to find strong generating sets even for degrees in the millions.

Let $G$ be a permutation group given by a generating set $S$. A *nearly linear time* algorithm answering a question about $G$ is one that runs in time $O(|S|n \log^c |G|)$ for some fixed constant $c$. If $G$ has a base of length $O(\log^d n)$, then the order of $G$ is $O(n^{\log^d n})$. Thus $\log |G|$ is $O(\log^{d+1} n)$. Therefore in this case a nearly linear time algorithm runs in time $O(|S|n \log^r n)$, where $r = c(d+1)$. This is not much larger than the time needed to read in the data for the problem.

Most known nearly linear time algorithms are randomized and thus have a small probability of error. There are randomized nearly linear time algorithms for finding orbits and blocks of imprimitivity of a group $G$, for computing the order of $G$, for deciding membership in $G$, and even for constructing a composition series for $G$.

There are still some algorithmic questions about permutation groups

which are not known to have polynomial time solutions. For example, the problem of finding generators for the intersection of two permutation groups is equivalent to graph isomorphism and may not have a polynomial time solution.

Another class of difficult problems are those which ask about properties of specific generating sets of a permutation groups. As remarked above, it is easy to determine the order of the Rubik Cube group and thus to compute the number of possible configurations the cube can have. However, we do not yet know how to find out the minimum number of moves which is needed to return an arbitrary configuration to the starting configuration. That is, we do not know the minimum number of factors needed to express an arbitrary element of the Rubik Cube group as a product of the generators corresponding to the rotations of the six faces of the cube.

# 5 Matrix groups

It is natural to consider a group $G$ generated by a set of invertible matrices.

Analogous to the problems of determining orbits and deciding primitivity for permutation groups, one has the following questions:

- Does $G$ act reducibly on the underlying vector space? That is, does there exist a proper nontrivial subspace which is invariant under the action of $G$?

- If $G$ acts irreducibly, does $G$ act imprimitively? That is, can the vector space be expressed as a nontrivial direct sum of subspaces in such a way that the elements of $G$ permute the direct summands among themselves?

As with any group defined as a subgroup, one has the membership problem for $G$. One would also like to be able to say something about the structure of $G$. Is $G$ finite? If so, what is its order and what are its composition factors? Is $G$ solvable or nilpotent?

What can be said about $G$ depends very much on the field over which $G$ is defined.

Let us first consider the case in which $G$ is a subgroup of $\mathrm{GL}(n, q)$ for some positive integer $n$ and some prime power $q$. The size of a matrix for analyzing complexity is $n^2 \log q$.

Computer scientists are quick to point out that there are some very difficult problems even when $n = 1$. For example, we do not know how to solve the membership problem in polynomial time in this case. This problem is usually called the discrete log problem. Given two nonzero elements $a$ and $b$ of $\mathrm{GF}(q)$, is $b$ a power of $a$? If so, find an integer $r$ such that $b = a^r$. The order of the multiplicative group of $\mathrm{GF}(q)$ is $q - 1$ and we do not know how to compute the prime factorization of $q - 1$ in polynomial time. Thus we can not determine in polynomial time the possible orders of subgroups of $\mathrm{GL}(1, q)$.

Group theorists tend not to worry about the discrete log problem as much as computer scientists. Group theorists are most often interested in the case in which $q$ is relatively small. They are prepared to assume that

$$|\mathrm{GL}(n, q)| = q^{\binom{n}{2}} \prod_{i=1}^{n} (q^i - 1)$$

can be factored into primes. This means that the prime factors of the numbers $q^i - 1$ can be determined.

A remarkably simple but extremely useful algorithm for deciding whether $G$ is reducible and if so, for finding an invariant subspace was found by R. Parker, who christened it the Meat-Axe. This algorithm has been improved by Holt and Rees.

So far it has not been possible to find a data structure analogous to a base and strong generating which permits a polynomial time solution to the membership problem even when the discrete log problem is not the obstacle. Researchers are currently focusing on what has been dubbed the "matrix group recognition project". This is a project to develop algorithms, possibly randomized, for providing information about the nonabelian composition factors of $G$ and about the maximal subgroups of $\mathrm{GL}(n, q)$ which contain $G$. An introduction to this project can be found in [27].

Now suppose that $G$ is a finitely generated subgroup of $\mathrm{GL}(n, K)$, where $K$ is an infinite field. The main case which has been studied is the one in which $K$ is a finite extension of the rational numbers. If $m$ is the degree of $K$ over $\mathbb{Q}$, then any vector space of dimension $n$ over $K$ is a vector space of dimension $mn$ over $\mathbb{Q}$. This gives an embedding of $\mathrm{GL}(n, K)$ into $\mathrm{GL}(mn, \mathbb{Q})$. Thus it suffices to consider the case $K = \mathbb{Q}$.

If $n \geq 4$, then the membership problem for finitely generated subgroups of $\mathrm{GL}(n, \mathbb{Q})$ is undecideable. The group $\mathrm{GL}(2, \mathbb{Q})$ contains a subgroup isomorphic to the free group $F$ on two generators. Since $\mathrm{GL}(4, \mathbb{Q})$ contains a

copy of $\mathrm{GL}(2,\mathbb{Q}) \times GL(2,\mathbb{Q})$, it follows that $\mathrm{GL}(4,\mathbb{Q})$ contains a copy of $F \times F$. The subgroup membership problem for finitely generated subgroups of $F \times F$ is equivalent to the word problem for finitely presented groups and hence is undecideable.

Somewhat surprisingly, it is possible to decide whether a finitely generated subgroup of $\mathrm{GL}(n,\mathbb{Q})$ is finite.

Any polycyclic group is isomorphic to a subgroup of $\mathrm{GL}(n,\mathbb{Z})$ for some $n$ and all solvable subgroups of $\mathrm{GL}(n,\mathbb{Z})$ are polycyclic. Thus there is a close connection between the algorithmic theories of polycyclic groups and solvable subgroups of $\mathrm{GL}(n,\mathbb{Z})$.

More information about algorithms referred to in this section can be found in [6] and [30].

# 6 Black box groups

Black box groups are not just of theoretical interest. At times, when one is studying a permutation groups $G$, one may have subgroups $H$ and $K$ of $G$ with $K$ normal in $H$. The quotient $H/K$ may not have a description as a permutation group and may have to be treated as a black box group. In fact it is sometimes useful to consider $G$ itself to be a black box group.

Most algorithms related to black box groups are randomized. An important class of such algorithms are algorithms for recognizing simple groups. The idea is to decide whether the black box group $G$ is isomorphic to a given simple group $H$ and, if it is, to exhibit an isomorphism. It is surprising that polynomial time randomized recognition algorithms exist. In some cases, additional information about the group $G$ is required, such as the ability to determine orders of elements quickly. See [15] for more details.

# 7 Abelian groups

One class of groups in which computation is relatively easy is the class of finitely generated abelian groups. Any such group is isomorphic to a quotient group of a free abelian group $M = \mathbb{Z}^n$, the direct sum of copies of the additive group of integers.

A subgroup $H$ of $M$ is finitely generated and thus may be described as the group generated by the rows of an $m$-by-$n$ integer matrix $A$. Given $A$, the

first question, as usual, is to decide membership in $H$. This is usually done by replacing $A$ by its (row) Hermite normal form by applying integer row operations to $A$. Sometimes one needs to know an element $P$ of $\mathrm{GL}(m, \mathbb{Z})$ such that $PA$ is in Hermite normal form. It is important to keep the entries in $P$ as small as possible. Polynomial time algorithms for computing Hermite normal form are known.

Sometimes lattice reduction is used instead of computing Hermite normal forms. The LLL lattice reduction algorithm is slow, but it sometimes produces multipliers $P$ with quite small entries. See [36]. Other approaches to computing nice bases of subgroups of $\mathbb{Z}^n$ are discussed in [**?**].

# 8 Polycyclic groups

Although computation with infinite groups usually leads quickly to unsolvable problems, there is one interesting class which provides an exception. This is the class of polycyclic-by-finite groups, groups which possess a polycyclic subgroup with finite index. All such groups have finite presentations and most computational problems concerning these groups have algorithmic solutions. In some cases, however, it is not known whether there are algorithms which are practical with current hardware.

Since space is limited, the discussion here will be restricted to polycyclic groups. The finite subgroup at the top of a polycyclic-by-finite group adds only technical difficulties. A good general reference on polycyclic groups is [33].

An alternative characterization of polycyclic groups gives some insight into the reason polycyclic groups are nice from a computational point of view. A group $G$ is polycyclic if and only if it is solvable and all subgroups are finitely generated. Finite generation of subgroups is equivalent to the ascending chain condition on subgroups. As noted previously, we are frequently looking for a subgroup $H$ of $G$. Suppose that we know a subgroup $K$ of $H$. If we have a procedure which either confirms that $K = H$ or produces an element $h$ in $H$ but not in $K$, then we are guaranteed to be able to find $H$. If $K \neq H$, then we replace $K$ by the subgroup generated by $K$ and the element $h$ produced by the procedure. The ascending chain condition implies that we will only be able to iterate this step a finite number of times.

Suppose that $G$ is a polycyclic group and let

$$G = G_1 \supseteq G_2 \supseteq \cdots \supseteq G_k \supseteq G_{k+1} = 1$$

be a polycyclic series for $G$. Since each group $G_i/G_{i+1}$ is cyclic, for $1 \leq i \leq k$ we can find an element $a_i$ such that the coset $a_i G_{i+1}$ is a generator for $G_i/G_{i+1}$. By an easy induction, an element of $G$ may be expressed in the form $a_1^{x_1} \cdots a_k^{x_k}$, where the $x_i$ are integers. If $G_i/G_{i+1}$ is finite of order $m_i$, then we can assume that $0 \leq x_i < m_i$. With this condition, the exponents $x_i$ are uniquely determined by $g$. The $k$-tupe $(x_1, \ldots, x_k)$ is called the *exponent vector* of $g$ and $a_1^{x_1} \cdots a_k^{x_k}$ is called the *normal form* for $g$.

Let $I$ be the set of those indices $i$ for which $G_i/G_{i+1}$ is finite. For $1 \leq i < j \leq k$ and each choice of $\alpha$ and $\beta$ in $\{1, -1\}$ the element $a_i^{-\alpha} a_j^{\beta} a_i^{\alpha}$ is in $G_{i+1}$. Thus there are words $U(i, j, \alpha, \beta)$ over $\{a_{i+1}, \ldots, a_k\}$ such that

$$a_j^{\beta} a_i^{\alpha} = a_i^{\alpha} U(i, j, \alpha, \beta).$$

If $i$ is in $I$, then $a_i^{m_i}$ is in $G_{i+1}$. It follows that there are words $V_i$ and $W_i$ over $\{a_{i+1}, \ldots, a_k\}$ such that

$$a_i^{m_i} = V_i \qquad \text{and} \qquad a_i^{-1} = a_i^{m_i - 1} W_i.$$

To these relations it is useful to add the redundant relations

$$a_i a_i^{-1} = 1 \qquad \text{and} \qquad a_i^{-1} a_i = 1,$$

for $1 \leq i \leq k$.

Not only do these relations give a presentation for $G$, they permit the computation of normal forms. If $W$ is a word over the $a_i$ which is not in normal form, then $W$ contains as a subword the left side of one of these relations. If we replace that left side with the corresponding right side and iterate, we will eventually obtain a word in normal form representing the same element of the group as $W$. This process is an example of rewriting, which is discussed below.

The rewriting approach to computing with elements of polycyclic groups is usually called *collection* by analogy with commutator collection introduced by P. Hall.

Subgroups of our polycyclic group $G$ can be represented by integer matrices which are analogous to the matrices in Hermite normal form which represent subgroups of $\mathbb{Z}^n$. Let $H$ be a subgroup of $G$. For $1 \leq i \leq k$ let $H_i = H \cup G_i$. Let $J$ be the set of indices $i$ such that $H_i \neq H_{i+1}$, that is, $H_i$ is not contained in $G_{i+1}$. For $i$ in $J$ we can chose an element $h_i$ in $H_i$ whose image generates the image of $H_i$ in the cyclic group $G_i/G_{i+1}$. If $i$ is in

$I$, then we can assume that $x_i$, the first nonzero exponent of $h_i$ is a proper divisor of $m_i$. The elements $h_i$ with $i$ in $J$ generate $H$ and the matrix whose rows are the exponent vectors for the $h_i$ gives a good description of $H$. With this matrix, or equivalently a knowledge of the $h_i$, we decide membership in $H$ and decide equality of cosets.

Finite solvable groups form an important subclass of polycyclic groups. If $G$ is a finite solvable group, then one normally chooses the polycyclic series to be a composition series. In this case the integers $m_i$ are all primes.

Algorithms for many constructions in finite solvable groups are available in both of packages Magma and GAP discussed in Section 10. These include computing centralizers, normalizers, and intersections of subgroups and constructing Sylow subgroups. One can also find representatives for the conjugacy classes of elements. Which computations are feasible depends on the order of the group and the type of computation. Intersections can be found in very large groups, but complete information about conjugacy classes of elements can be difficult in groups with orders in the millions.

For infinite polycyclic groups algorithms for most constructions have been shown to exist, but many of these algorithms are not practical. See [4]. A practical algorithm for deciding conjugacy of elements in a polycyclic group would be quite interesting.

For nilpotent polycyclic groups the situation is better. In [19] practical algorithms for computing intersections and normalizers are described.

# 9 Finitely presented groups

Most computational questions about finitely presented groups are undecideable in general. One of the few exceptions are questions about certain quotient groups. This section describes some of the tools available to attempt to study a finitely presented group. Details can be found in [36].

## 9.1 Coset enumeration

Computation of products of elements in a free group $F = F(X)$ is very easy. Most other calculations with elements of $F$ pose few difficulties. For example, the conjugacy problem for $F$ has an efficient solution. Given two reduced words $U$ and $V$ over $X$, one writes $U$ as $ABA^{-1}$ and $V$ as $CDC^{-1}$ with the words $A$ and $C$ of maximal length. Then $U$ and $V$ are conjugate in $F$ if and

only the words $B$ and $D$ have the same length and are cyclic permutations of each other.

Subgroups of free groups are free. Most computational questions about a finitely generated subgroup $H$ of $F$ have algorithmic solutions. The solution to the membership problem for $H$ is particularly important. There are in fact two approaches to this problem. One uses Nielsen reduction of words and the other uses coset tables. The coset table approach will be sketched here. Nielsen reduction in described in [22].

A coset table relative to $X$ is an array $T$ whose rows are indexed by a finite set $\Omega$ of positive integers, whose columns are indexed by $X \cup X^{-1}$, and whose entries are elements of $\Omega$. Some entries may not be defined. The following conditions must hold:

1. The set $\Omega$ contains 1.

2. If $k$ is the entry in row $i$ and column $x$, then the entry in row $i$ and column $x^{-1}$ is $k$.

3. The graph $\mathcal{G}$ with vertex set $\Omega$ and labeled edges consisting of the triples $(i, x, j)$ such that $j$ is an entry in the $i$-th row and $x$-th column of $T$ is connected.

Here is an example of a coset table relative to $X = \{x, y\}$.

| | $x$ | $x^{-1}$ | $y$ | $y^{-1}$ |
|---|---|---|---|---|
| 1 | 2 | | 2 | 3 |
| 2 | 4 | 1 | | 1 |
| 3 | | | 1 | 5 |
| 4 | | 2 | 5 | |
| 5 | 5 | 5 | 3 | 4 |

Given an element $i$ of $\Omega$ and a word $u$ over $X$, there is at most one path in $T$ which starts at $i$ and has the property that the product of the labels on the edges of the path is $u$. If $j$ is the end point of this path, then we write $j = i^u$. In the above table, if $u = xyxy^2$, then $2^u = 1$.

The set of reduced words $u$ such that $1^u = 1$ is a finitely generated subgroup of $F$, which we denote $\mathcal{H}(T)$. Deciding membership in $\mathcal{H}(T)$ is linear in the length of the input word. There is a natural notion of isomorphism of coset tables and isomorphic tables define the same subgroup of $F$. Every finitely generated subgroup $H$ of $F$ is defined by a coset table and all coset

tables $T$ such that $H = \mathcal{H}(T)$ and the cardinality of $\Omega$ is minimal are isomorphic. For any coset table $T$, the subgroup $\mathcal{H}(T)$ has finite index in $F$ if and only if every entry in $T$ is defined. In this case $|F : H| = |\Omega|$.

In its simplest form, *coset enumeration* refers to the construction of a coset table $T$ such that $\mathcal{H}(T) = H$, where $H$ is given as the subgroup of $F$ generated by a finite set $\mathcal{S}$. The basic step in coset enumeration consists of the following: Given a coset table $T$ and an element $g$ of $F$, find a coset table $T_1$ such that $\mathcal{H}(T_1)$ is the subgroup generated by $\mathcal{H}(T)$ and $g$.

To find $T_1$, we write the word $g$ as a product of subwords $g = uvw$, where $i = 1^u$ and $j = 1^{w^{-1}}$ are defined, the length of $u$ is maximal, and subject to this the length of $w$ is maximal.

If the length of $v$ is 1, so $v$ is in $X \cup X^{-1}$, then we change $T$ by making $j$ the entry in row $i$ and column $v$ and making $i$ the entry in row $j$ and column $v^{-1}$.

If $v$ has length greater than 1, let $x$ be the first factor of $v$. We add a new row to $T$ indexed by a new index $k$ and add entries so that $i^x = k$ and $k^{x^{-1}} = i$. Then we recompute $u$, $v$, and $w$.

If $v$ is empty, then either $i = 1$ and $w$ is empty or $i \neq j$. In the first case, $g$ is in $H$ and no change to $T$ is needed. If $i \neq j$, then we must identify $i$ and $j$. More precisely, we find the finest equivalence relation $\sim$ on $\Omega$ such that $i \sim j$ and whenever $x$ is in $X \cup X^{-1}$, $r \sim s$ and both $r^x$ and $s^x$ are defined in $T$, then $r^x \sim s^x$. Let $\Omega_1$ be the set of elements in $\Omega$ which are first in their $\sim$-class. We can take $T_1$ to be the coset table with rows indexed by $\Omega_1$ such that $i^x = j$ in $T_1$ if and only if there are elements $r$ and $s$ in $\Omega$ such that $i \sim r$, $s \sim j$ and $r^x = s$ in $T$. This construction is known as the coincidence procedure. The fact that it can be carried out efficiently is what makes coset enumeration such a useful algorithm.

Now let $(X, R)$ be a finite presentation for a group $G$. There is a homomorphism of $F$ onto $G$ with kernel equal to N, the subgroup of $F$ generated by the conjugates of the elements of $R$. There is a one-to-one correspondence between subgroups of $G$ and subgroups of $F$ which contain $N$. If the subgroup $H$ of $G$ corresponds to the subgroup $K$ of $F$ then $|G : H| = |F : K|$.

Suppose that $X$ and two finite subsets $\mathcal{R}$ and $\mathcal{S}$ are given. Let $G = \langle X \mid R \rangle$ and let $H$ be the subgroup of $G$ generated by the image of $\mathcal{S}$. If $|G : H|$ is finite, then it is possible to verify this fact and to compute $|G : H|$. However, if $|G : H|$ is infinite, then there is no algorithm which is guaranteed to demonstrate that fact.

In its more general form, coset enumeration refers to a family of related

procedures for attemptying to verify that $|G : H|$ is finite. Suppose that $|G : H|$ is finite and let $K$ be the inverse image of $H$ in $F$. Then $|F : K|$ is finite and $K$ is generated by $\mathcal{S}$ and a finite set of conjugates of elements of $\mathcal{R}$. All variants of coset enumeration proceed by constructing the coset table for the subgroup $L$ of $F$ generated by $\mathcal{S}$ and larger and larger finite sets of conjugates of elements of $\mathcal{R}$ until $|F : L|$ is finite. As noted above, this occurs when every entry in the coset table $T$ for $L$ is defined. Let $\Omega$ be the set of integers indexing the rows of $T$. For each element $i$ of $\Omega$ we chose one word $U$ such that $1^U = i$ in $T$. We add all elements of the form $URU^{-1}$ to the generators of $L$, where $R$ ranges over the elements of $\mathcal{R}$. This may cause $L$ to get bigger. At this point $L = K$.

Efforts to program coset enumeration go back at least to 1953. Despite the intervening 45 years, we are still finding ways to carry out the procedure more quickly or with less memory. Currently it is possible to work with coset table having tens of millions of rows.

## 9.2 The Knuth-Bendix procedure for strings

The Knuth-Bendix procedure for strings is a special case of the very general technique of universal algebra described in [16]. It provides an alternative to coset enumeration for studying finitely presented groups. With coset enumeration the basic data structure is the coset table. With the Knuth-Bendix procedure for strings the basic data structure is the rewriting system.

Rewriting systems are defined with the aid of special orderings on words. Let $X$ be a finite set. A *reduction ordering* on the set $M$ of group words over $X$ is an well ordering $<$ of $M$ with the property that the empty word is less than any other word and, for all words $A$, $B$, $U$, and $V$, if $U < V$, then $AUB < AVB$. A rewriting system relative to a reduction ordering is a set $\mathcal{R}$ of ordered pairs of words such that for each element $(V, U)$ of $\mathcal{R}$ we have $V > U$. Elements of $\mathcal{R}$ are called rewriting rules.

Given a rewriting system $\mathcal{R}$, we shall be interested in the group $G$ generated by $X$ and defined by the relations $V = U$ for all $(V, U)$ in $\mathcal{R}$. It is useful to assume that $\mathcal{R}$ contains the rules $(xx^{-1}, 1)$ for all $x$ in $X \cup X^{-1}$. The corresponding relations are redundant in the context of groups but not in the context of monoids.

Once we have fixed a rewriting system $\mathcal{R}$, we can rewrite a group word $W$ over $X$ using the following procedure:

Procecure REWRITE($W$)
  $P := W$;
  While $P$ contains the left side of a rule in $\mathcal{R}$ do
    Let $P = AVB$, where $A$ and $B$ are words and $(V, U)$ is in $\mathcal{R}$;
    $P := AUB$;
  od;
  Return $P$;
END.

Because an occurrence of the left side of a rule $(V, U)$ is replaced by the right side, we often write the rule in the form $V \to U$.

Each time $P$ is changed in REWRITE, the new value is less than the old with respect to $<$. Since $<$ is a well ordering, $P$ can change only finitely often. Thus the procedure must terminate. The image of $P$ in $G$ does not change during the execution of REWRITE. Therefore the word returned by REWRITE defines the same element $g$ of $G$ as $W$ and is *reduced* with respect to $\mathcal{R}$ in the sense that it contains no left side of a rule as a subword. Executing REWRITE poses no problems if $\mathcal{R}$ is finite. If $\mathcal{R}$ is infinite, then it may still be possible to execute REWRITE.

It would be nice if the result of REWRITE depended only on $g$. Unfortunately this need not be the case. In the operation of REWRITE there may be many choices for the words $A$ and $B$ and the rule $(V, U)$. The final result may depend on those choices. Thus there may be many reduced words which define the same element of $G$.

The rewriting system $\mathcal{R}$ is called *confluent* if the word returned by the call REWRITE($W$) depends only on $W$ and not on the choices made during execution. When this is the case, each element of $G$ is defined by a unique reduced word. Thus, when $\mathcal{R}$ is confluent and we can execute REWRITE, then we can solve the word problem for $G$.

If $\mathcal{R}$ is finite, then there is a finite test for confluence. The test either confirms that $\mathcal{R}$ is confluent or produces a new rule $(V, U)$ which must be added to $\mathcal{R}$ if confluence is to hold. The Knuth-Bendix procedure for strings proceeds by iterating the test for confluence, adding any new rules produced. The procedure may not terminate, but if it does, the resulting rewriting system is confluent and gives a solution of the word problem for $G$.

The Knuth-Bendix procedure is most easily run which $<$ is a *lenlex* ordering. That is, words are first ordered by length and then lexicographically according to a specified ordering of the elements of $X \cup X^{-1}$. However, other

orderings are frequently needed. For example, the polycyclic presentation for a polycyclic group defined above is a confluent rewriting system with respect to an ordering on words where it is possible to have $U < V$ even if $U$ is longer than $V$.

The following is a confluent rewriting system:

$$aA \to 1, \quad Aa \to 1, \quad bB \to 1, \quad Bb \to 1,$$
$$bA \to Abb, \quad bba \to ab, \quad Ba \to baB, \quad BA \to ABB.$$

Here the "case convention" is being used. That is, $A$ and $B$ represent $a^{-1}$ and $b^{-1}$, respectively. Again the ordering $<$ is not compatible with length. The group defined by this rewriting system is infinite since all words of the form $a^i$ with $i > 0$ are reduced.

## 9.3 Quotient groups

Let $G$ be a group given by a finite presentation $(X, R)$. Despite the unsolvability of many computational questions about $G$, there are algorithms which are guaranteed to construct certain nilpotent quotients of $G$. There are also procedures for attempting to determine certain other solvable quotients of $G$.

The most important case is the determination of the largest abelian quotient $G/G'$. Suppose that $|X| = n$ and $X = \{x_1, \ldots, x_n\}$. There is a homomorphism $f$ from $F = F(X)$ to $\mathbb{Z}^n$ which maps $x_i$ to the $i$-th standard basis vector of $\mathbb{Z}^n$. The map $f$ is clearly surjective and the kernel of $f$ is $F(X)'$. It is not hard to show that $G/G'$ is isomorphic to the quotient $\mathbb{Z}^n/M$, where $M$ is the subgroup of $Z^n$ generated by the image of $R$ under $f$. Using the methods of Section 7, we can determine the structure of $G/G'$.

More generally, for any $s > 0$ we can determine a polycyclic presentation for $P = G/\gamma_s(G)$ and thus compute effectively in this quotient of $G$. An algorithm for computing this presentation is described in [25].

A special case of nilpotent quotients are quotients which are $p$-groups. Very powerful algorithms have been developed to compute quotients of $G$ which are $p$-groups for some given prime $p$. Quotients of order $p^n$ with $n$ in the thousands are often within reach.

The quotients of $G$ by terms of the derived series other than the first need not be polycyclic and cannot in general be computed. There are two approaches for computing finite solvable quotients of $G$. They are described in [26] and [31].

The only approach available for computing infinite nonnilpotent quotients of $G$ is given in [20]. This method is based on the fact that if $P$ is a polycyclic group, then right ideals in the group ring $\mathbb{Z}(P)$ are finitely generated. It is possible to generalize the Gröbner basis methods described in Section ??? to submodules of finitely generated free right modules over $\mathbb{Z}(P)$. Let $N$ be a normal subgroup of $G$ such that $G/N$ is isomorphic to $P$. Then $N/N'$ is a finitely generated right module over $\mathbb{Z}(P)$ it is possible to give a finite presentation for this module and using the Gröbner basis algorithm one can decide whether $N/N'$ is finitely generated as an abelian group. In this way one can decide whether $G/N'$ is polycyclic. If it is, then one can find a consistent polycyclic presentation for it.

## 9.4 The Reidemeister-Schreier algorithm

Let $G$ be a finitely presented group and let $H$ be a subgroup of $G$. It is possible to describe a presentation for $H$. If the index of $H$ in $G$ is finite, then this presentation is finite. The Reidemeister-Schreier algorithm is one method for constructing presentations for subgroups of finitely presented groups.

One of the few ways available to prove that $G$ is infinite is to find a subgroup $H$ of finite index such that $H/H'$ is infinite, which can be confirmed by finding the Reidemeister-Schreier presentation for $H$ and then using the methods of the previous subsection to determine the structure of $H/H'$.

Even if the presentation for $G$ is quite manageable and the index of $H$ is only a few hundred, the presentations for $H$ obtained are very complicated. It is usually necessary to find some way of simplifying the presentation before it can be used further. Two approaches to simplification are the Knuth-Bendix method for strings and the systematic use of Tietze transformations.

# 10 Group-theoretic software

There are three software packages which offer general support for group theoretic computation. These are Magma, GAP, and Magnus. In addition, Maple provides some tools for group theory and there are many smaller packages which are either intended to be run by themselves or with one or more of the larger systems.

Magma with its predecessor system Cayley is the oldest of the existing

comprehensive systems. It provides support for algebraic number theory and several other areas besides group theory. Information about Magma can be found on its Web page, `http://???`. A license fee is involved.

GAP has been around for about ??? years. A major revision of the system is being tested as this article is written. The focus of GAP is more directly on group theory. Its Web page is `http://???`.

The system Magnus is much newer and is less developed than Magma and GAP. It is intended to support research on infinite groups. For example, while Magma and GAP only attempt to find presentations of subgroups of finite index in finitely presented groups, Magnus is able to produce partial presentations for subgroups of infinite index.

Among the more specialized packages are

# 11 Another perspective

The point of view up to this point has be to look at a class of groups and ask how hard it is to compute with elements and subgroups in groups belonging to that class. There is a growing body of research which looks at computation in groups quite differently. One make assumptions about the ease or difficulty of computing in a group and asks what this implies about the structure of the group.

The most common assumption is that the group is automatic. An *automatic group* is a group $G$ in which multiplication and comparison of elements can be performed using finite automata. The standard reference on automatic groups is [12].

The sets of elements of automatic groups are naturally described by families of words called regular languages. It is possible to consider classes of groups whose elements require more complicated types of languages for their description.

Sylow subgroups of permutation groups. Random elements Lower central series Derived series Degree

# References

[1] *Geometric and computational perspectives on infinite groups*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science,

vol. 25, Providence, AMS, 1996.

[2] Michael D. Atkinson (ed.), *Computational group theory*, London, New York, Academic Press, 1984.

[3] Laszlo Babai, *Computational complexity in finite groups*, Proceedings of the International Congress of Mathematicians, Kyoto, Japan, 1990, The Mathematical Society of Japan, 1991, p. what pages.

[4] Gilbert Baumslag, Frank B. Cannonito, Derek J. S. Robinson, and Dan Segal, *The algorithmic theory of polycyclic-by-finite groups*, Journal of Algebra **142** (1991), 118–149.

[5] Robert Beals, *Computing blocks of imprimitivity for small-base groups in nearly linear time*, In Finkelstein and Kantor [13], pp. 17–26.

[6] ———, *Towards polynomial time algorithms for matrix groups*, In Finkelstein and Kantor [14], pp. 31–54.

[7] Gregory Butler, *Fundamental algorithms for permutation groups*, Lecture Notes in Computer Science, vol. 559, Springer-Verlag, Berlin-Heidelberg-New York, 1991.

[8] J. Cannon and G. Havas, *Algorithms for groups*, Australian Computer Journal **24** (1992), 59–68.

[9] John Cannon (ed.), *Computational group theory I*, vol. 9, Journal of Symbolic Computation, no. 5 & 6, London, New York, Academic Press, 1990.

[10] John Cannon (ed.), *Computational group theory II*, vol. 12, Journal of Symbolic Computation, no. 4 & 5, London, New York, Academic Press, 1991.

[11] M. Dehn, *Ueber unendlichediskontinuierliche gruppen*, Math. Ann. **71** (1911), 116–144.

[12] D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, and W. P. Thurston, *Word processing in groups*, Academic Press, New York, 1992.

[13] Larry Finkelstein and William M. Kantor (eds.), *Groups and computation*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 11, Providence, AMS, 1993.

[14] Larry Finkelstein and William M. Kantor (eds.), *Groups and Computation II*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 28, Providence, AMS, 1997.

[15] William M. Kantor, *Simple groups in computational group theory*, Proceedings of the International Congress of Mathematicians, Berlin, 1998, Deutsche Mathematiker-Vereinigung, 1998, pp. 77–86.

[16] D. E. Knuth and P. B. Bendix, *Simple word problems in universal algebra*, Computational Problems in Abstract Algebra (John Leech, ed.), Pergamon Press, 1970, pp. 169–183.

[17] J. Leech (ed.), *Computational problems in abstract algebra, proceedings of a conference held at Oxford under the auspices of the Science Research Council, Atlas Computer Laboratory, 29. aug. to 2. sept. 1967*, Oxford, Pergamon Press, 1970.

[18] Jeffrey S. Leon, *Partitions, refinements, and permutation group computations*, In Finkelstein and Kantor [14], pp. 123–158.

[19] Eddie H. Lo, *Finding intersections and normalizers in finitely generated nilpotent groups*, Journal of Symbolic Computation **25** (1998), 45–59.

[20] ———, *A polycyclic quotient algorithm*, Journal of Symbolic Computation **25** (1998), 45–59.

[21] Eugene M. Luks, *Isomorphism of graphs of bounded valence and be tested in polynomial time*, Journal of Computer and System Sciences **25** (1982), 42–65.

[22] Roger C. Lyndon and Paul E. Schupp, *Combinatorial group theory*, Springer Verlag, 1977.

[23] J. Neubüser, *Untersuchungen des Untergruppenverbandes endlicher Gruppen auf einer programmgesteurten elektronischen Dualmaschine*, Numer. Math. **2** (1960), 280–292.

[24] _____, *An invitation to computational group theory*, Groups '93, Galway/St. Andrews, London Math. Soc. Lecture Note Series, vol. 212, Cambridge University Press, 1995, pp. 457–475.

[25] Werner Nickel, *Computing nilpotent quotients of finitely presented groups*, In *Geometric and Computational Perspectives on Infinite Groups* [1], pp. 175–191.

[26] Alice C. Niemeyer, *A finite soluble quotient algorithm*, Journal of Symbolic Computation **18** (1994), 541–561.

[27] Alice C. Niemeyer and Cheryl E. Praeger, *Implementing a recognition algorithm for classical groups*, In Finkelstein and Kantor [14], pp. 273–296.

[28] P. S. Novikov, *On the algorithmic unsolvability of the word problem in group theory*, Trudy Math. Inst. im. Steklov **44** (1955), 1–143, In Russian.

[29] E. A. O'Brien, *The groups of order 256*, Journal of Algebra **143** (1991), 219–235.

[30] Gretchen Ostheimer, *Algorithms for polycyclic-by-finite matrix groups*, In Finkelstein and Kantor [14], pp. 297–308.

[31] W. Plesken, *Towards a solvable quotient algorithm*, Journal of Symbolic Computation **4** (1987), 111–122.

[32] M. O. Rabin, *Recursive unsolvability of group theoretic problems*, Ann. of Math. **67** (1958), 172–194.

[33] D. Segal, *Polycyclic goups*, Cambridge University Press, Cambridge, 1983.

[34] Akos Seress, *Permutation group algorithms*, Cambridge University Press, Cambridge, in press.

[35] _____, *An introduction to computational group theory*, Notices of the American Mathematical Society **44** (1997), 671–679.

[36] Charles C. Sims, *Computation with finitely presented groups*, Cambridge University Press, 1994.

[37] J. A. Todd and H. S. M. Coxeter, *A practical method for enumerating cosets of finite abstract groups*, Proc. Edinburgh Math. Soc. **5** (1936), 26–34.