An Introduction to Computational Group Theory

Ákos Seress

Can one rotate *only one* corner piece in Rubik's cube? What are the energy levels of the buckyball molecule? Are the graphs on Figure 1 isomorphic? What is the Galois group of the polynomial $x^8 + 2x^7 + 28x^6 + 1728x + 3456$? What are the possible symmetry groups of crystals? These are all questions which, directly or in a not so obvious way, lead to problems in computational group theory.

Algebraic structures are well suited for machine computations. One reason for that is that we can describe large objects very concisely by a set of generators: for example, 50 bits are enough to define $GL_5(2)$, a group of order 9999360, by two 0-1 matrices of size 5×5. Even more importantly, often we can find a generating set which reflects the structure of the group so that structural and quantitative properties can be read off easily.

Computational group theory (CGT) is one of the oldest and most developed branches of computational algebra. Although most general-purpose symbolic algebra programs can handle groups to a certain extent, there are two systems which are particularly well suited for computations with groups: GAP and MAGMA. Also, numerous stand-alone programs and smaller systems are available.

GAP can be obtained by anonymous ftp from servers on three continents; the addresses can be found on the World Wide Web page `http://www-groups.dcs.st-and.ac.uk/`.

For the availability of MAGMA, please see the World Wide Web page `http://www.maths.usyd.edu.au:8000/comp/magma/`.

The important subareas of CGT correspond to the most frequently used representations of groups: permutation groups, matrix groups, and groups defined by generators and relators, as well as to perhaps the most powerful tool for the investigation of groups, representation theory. Also, there are specialized and more efficient algorithms for special classes such as nilpotent or solvable groups. In this survey in each subarea we attempt to indicate the basic ideas and the size of jobs which can be handled by the current systems on a reasonable machine. Of course, we cannot be comprehensive here. Also, because of space restrictions, our reference list consists only of surveys, conference volumes, books, and journal special issues. Individual results are referenced in the text only if they appear in these volumes; most of the others can be traced back from these sources. An extended version of this article, with complete references, can be obtained from `http://www.math.ohio-state.edu/~akos/` or `http://www.math.rwth-aachen.de/~Akos.Seress/`. E. O'Brien's database of papers on group theory, including

Ákos Seress is associate professor of mathematics at The Ohio State University, Columbus, Ohio. His e-mail address is `akos@math.ohio-state.edu`.

Partially supported by NSF Grant CCR-9503430 and by the Alexander von Humboldt Foundation.

Acknowledgement: the author is indebted to G. Havas, D. Holt, W. Kantor, K. Lux, J. Neubüser, and E. O'Brien for their helpful comments. The part of the section "Polyclyclic Groups" about quotient group methods was written by J. Neubüser.

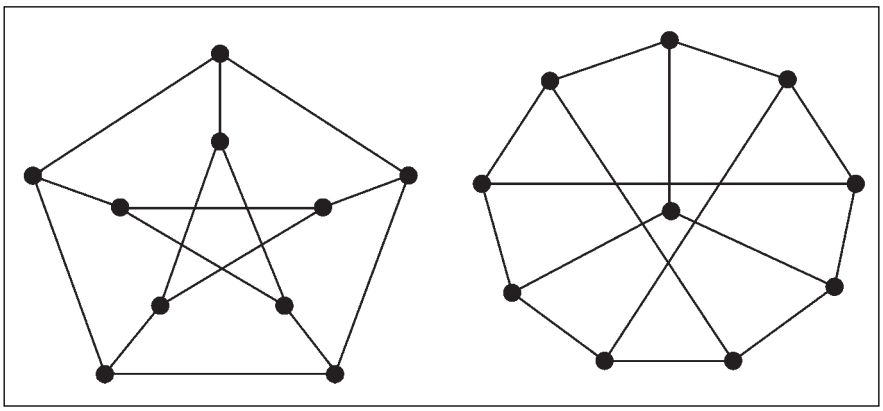

Figure 1.

a lot of references to CGT, is available via `http://www.math.auckland.ac.nz/~obrien/`.

We start with some historical remarks. Algorithmic questions permeated group theory from its inception in the last century. As examples, consider that Galois's work was inspired by the solvability of equations. Thirty years later, Mathieu announced the existence of the 5-transitive group M_{24}, but he needed twelve years to find the "clarity and elegance necessary to present it." Had he access to a computer, this period could probably have been shortened significantly. Jordan, Hölder, Cole, and others could also have used the machine in their quest to classify groups of small order.

The "official" starting date of CGT may be pinned down in 1911, when Dehn proposed the solution of the word problem, Find an algorithm to decide whether, in a group defined by a finite set of abstract generators and relators, a word in the generators represents the identity. Dehn's question was motivated by topological considerations; even today it is hard to draw a sharp border between combinatorial group theory and topology. The flourishing of CGT started in the sixties, when, for example, the basic methods for permutation group manipulation and the computation of character tables were established, and term rewriting procedures were introduced. Not much later, the first large applications, such as Sims's existence proof for Lyons's sporadic simple group, arose, and the development of the first integrated system, the Aachen-Sydney Group System, started. Since then the area has been growing rapidly, both in terms of the design, implementation, and application of algorithms, as well as in the number of mathematicians involved in this development. Nowadays, some of the major lines of development are the integration of consequences of the classification of finite simple groups and methods suggested by complexity theoretical considerations into practical algorithms and the systematic use of randomization. A more detailed history is in [11].

Finitely Presented Groups

Let $G = \langle E|\mathcal{R}\rangle$ be a presentation for a group $G : E = \{g_1, \dots, g_n\}$ is a finite set of generators, and $\mathcal{R} = \{r_1 = 1, \dots, r_m = 1\}$ is a set of defining relations. Each r_i is a word, using the generators in E and their inverses. The basic questions are to decide whether G is finite and to determine whether a given word represents the identity of G.

By the celebrated result of Novikov and Boone, these questions are *undecidable*: they cannot be answered by a recursive algorithm. Nevertheless, because of the practical importance of the problem, a lot of effort is devoted to the development of methods for investigating finitely presented groups.

One basic method is the *Todd-Coxeter coset enumeration procedure*. Given $G = \langle E|\mathcal{R}\rangle$ and $H = \langle h_1, \dots, h_k\rangle$, where $H \le G$ and each h_j is a word in the generators of G and their inverses, our goal is to compute the permutation representation of G on the right cosets of H.

We set up a *coset table*: this is a matrix M, where the rows are labelled by positive integers, representing cosets of H, and the columns are labelled by the elements of $\overline{E} := \{g_1, \dots, g_n, g_1^{-1}, \dots, g_n^{-1}\}$. The entries (if defined) are positive integers, $M(k, g) = l$, if we know that $kg = l$ for the cosets k, l and for $g \in \overline{E}$. Originally, we have a $1 \times |\overline{E}|$ table with no entries, where 1 denotes the coset $H \cdot 1$. As new cosets are defined, we add rows to the coset table.

Of course, we have to detect when two words, defining different rows of the table, actually belong to the same coset of H. To this end, for each relation $r_i = g_{i_1} g_{i_2} \cdots g_{i_t}$, we also maintain a *relation table*. This is a matrix M_i, with rows labelled by the cosets $1, 2, \dots$, as defined in M, and columns labelled by the elements of the sequence $(g_{i_1}, g_{i_2}, \dots, g_{i_t})$. The entry $M_i(k, g_{i_j})$, if defined, is the number of the coset $kg_{i_1} \cdots g_{i_j}$. Initially, we have $M_i(k, g_{i_t}) = k$ for each row number k, since $r_i = 1$ in G. Whenever a new coset is defined, we fill all entries of the relation tables that we can.

Finally, for each generator $h_j = g_{j_1} \cdots g_{j_t}$ of H, we maintain a *subgroup table*. This is a matrix S_j with only one row, corresponding to the coset $H \cdot 1$, and columns labelled by the factors of h_j. The rule for filling entries is the same as for the M_i; originally, $S_j(1, g_{j_t}) = 1$, since $Hh_j = H$.

When the last entry is filled in a row of a relation table or a subgroup table, we also get an extra piece of information, $kg = l$, for some cosets k, l and $g \in \overline{E}$. This is called a *deduction*. If the entry $M(k, g)$ is not yet defined, then we fill the entries $M(k, g)$, $M(l, g^{-1})$, and all possi-

ble entries in the relation and subgroup tables; this way, we may get further deductions. If $M(k,g)$ is already defined but $l' := M(k,g) \neq l$, then we realize that l, l' denote the same coset of H. This is called a *coincidence*. We replace all occurrences of l, l' by the smaller of these two numbers and fill the entries of the tables that we can. This may lead to further deductions and coincidences. The process stops when all entries of the coset table, the relation tables, and subgroup tables are filled.

CT	g_1	g_2
1	2	3
2	1	4
3		1
4		2

RT	g_1	g_2	g_1	g_2	g_1	g_2
1	2	4			3	1
2	1	3			4	2
3			4	2	1	3
4			3	1	2	4

ST	g_1	g_2	g_1	g_2
1	2	4	3	1

Figure 2.

We illustrate these ideas by enumerating $G = \langle g_1, g_2 | g_1^2 = 1, g_2^2 = 1, (g_1 g_2)^3 = 1 \rangle \cong S_3$ on the cosets of the subgroup $H = \langle g_1 g_2 g_1 g_2 \rangle$ of order 3. Since both generators are involutions, we have $\overline{E} = E$. Also, we maintain only one relation table, corresponding to $(g_1 g_2)^3 = 1$; the other two relators tell us that at the definition of new cosets, we should multiply previous cosets by g_1, g_2 alternatingly. Figure 2 shows the coset table CT, relation table RT, and subgroup table ST after the definition of the cosets $1 := H, 2 := 1g_1, 3 := 1g_2, 4 := 2g_2$. At that moment, the last entry (in the second column) of ST is filled and we get the deduction $4g_1 = 3$, which also implies $3g_1 = 4$. Then all entries in CT are known, and we can complete RT; this leads to the coincidences $1 = 4$ and $2 = 3$.

Taking minimal care in defining new cosets (namely, if a coset k is defined, then sooner or later we define kg for all $g \in \overline{E}$), it is guaranteed that the algorithm terminates if $|G : H| < \infty$. However, there is no recursive function of $|G : H|$ and the input length which would bound the number of cosets defined during the procedure. It is easy to give presentations of the trivial group such that no commonly used variant of the Todd-Coxeter algorithm can handle them. This, and different coset enumeration strategies, are discussed, for example, in [13, Ch. 5]. A very accessible, elementary description of the methods is in [10]. Success mostly depends on the number of entries defined in the coset table rather than $|G : H|$. There are instances of successful enumeration with $|G : H| > 10^6$.

If we do not have a candidate for a small index subgroup H in G, we may try programs which find some or all subgroups of G with index at most a given bound n. These programs consider all coset tables with at most n rows and use a backtrack search to eliminate those which are not consistent with the given relators. Depending on the complexity of the presentation, in some cases we can expect success for values of n up to about 100.

An alternative method to coset enumeration is the *Knuth-Bendix term-rewriting procedure* [13]. We collect a list of pairs of words (u, v) such that u, v represent the same element of G. These pairs are called *rewriting rules*, since we can replace a word $w_1 u w_2$ by $w_1 v w_2$. The goal is to collect a *confluent* system of rules: no matter in which order the rules are applied, every word in the generators is converted into a unique normal form. Although the usual problems of undecidability arise, Knuth-Bendix methods can sometimes solve the word problem for infinite groups, which can almost never be done by coset enumeration techniques.

An interesting recent development is the definition and the algorithmic handling of *automatic groups*. These are groups with solvable word problem and include important group classes occurring in topology, such as the fundamental groups of compact hyperbolic and Euclidean manifolds and of hyperbolic manifolds of finite volume, word hyperbolic groups, and groups satisfying various small cancellation properties.

If a presentation for a subgroup $H \leq G$ is needed, Schreier's subgroup lemma may be used to obtain generators for H.

Lemma 1.1. Let T be a right transversal for H in $G = \langle S \rangle$, and, for $g \in G$, let $\overline{g}$ denote the element of T such that $g \in H\overline{g}$. Then

$$\{ts\overline{ts}^{-1} : t \in T, s \in S\}$$

generates H.

Using the fact that words representing subgroup elements can be rewritten as products of Schreier generators, one may obtain a presentation for H [13, Ch. 6]. This presentation is usually highly redundant, and it can be shortened by applying the so-called *Tietze transformations* [13, Ch. 1]. It is often worthwhile to do Tietze transformations interactively, guiding the computer to the type of presentation we try to achieve.

Polycyclic Groups

In the rest of this survey, we shall almost exclusively deal with groups for which the undecidability of the word problem vanishes. This will clearly be the case with finite groups given, for example, as permutation or matrix groups, but

it is also the case for groups given by polycyclic presentations.

A group is called *polycyclic* if it has a finite subnormal series of the form

$$G = G_1 \rhd G_2 \rhd \cdots \rhd G_n \rhd G_{n+1} = 1, \tag{1}$$

with cyclic factors G_i/G_{i+1}. If, for $1 \le i \le n$, we pick $a_i \in G_i \setminus G_{i+1}$ such that $G_i = \langle G_{i+1}, a_i \rangle$, then each $g \in G$ can be written uniquely in the form $g = a_1^{e_1} a_2^{e_2} \cdots a_n^{e_n}$, where $e_i \in \mathbb{Z}$ for infinite cyclic factors G_i/G_{i+1}, and $0 \le e_i < |G_i : G_{i+1}|$ for finite factor groups. This description of g is called a *collected word* for g. For finite polycyclic groups, we also have

$$\begin{aligned} a_i^{|G_i:G_{i+1}|} &= a_{i+1}^{\varepsilon_{i;i+1}} a_{i+2}^{\varepsilon_{i;i+2}} \cdots a_n^{\varepsilon_{i;n}}, \\ &\qquad \text{for } 1 \le i \le n, \\ a_j a_i &= a_i a_{i+1}^{\varepsilon_{i,j;i+1}} a_{i+2}^{\varepsilon_{i,j;i+2}} \cdots a_n^{\varepsilon_{i,j;n}}, \\ &\qquad \text{for } 1 \le i < j \le n. \end{aligned} \tag{2}$$

These are the relators for the so-called *power-conjugate presentation* (pcp) for G. (For infinite polycyclic groups, similar relators have to be added to (2) for the inverses of the a_i of infinite order.)

The pcp's are very compact, but still relatively easily manageable representations of polycyclic groups. For finite solvable groups, as shown by the work of Laue, Neubüser, and Schoenwaelder in [1]; Celler; Eick in [7]; Glasby and Slattery in [2]; Mecky; Neubüser; M. Smith; and Wright, pcp's can be used to determine virtually all structural properties. Among others, conjugacy classes of elements, complements to normal subgroups, normalizers, and Frattini subgroups can be computed, and it is possible to find automorphism groups. For these tasks, if there are algorithms in other representations of groups, they can handle only groups of much smaller order. There are examples of structural exploration of polycyclic groups with $\log |G|$ in the thousands; currently, groups of that size cannot be handled in any other representation.

We indicate two basic methods, which are used in almost all algorithms dealing with finite solvable groups. The first one is to exploit analogies with linear algebra. Each collected word can be represented as an integer vector of length n, the coordinates being the exponents of the a_i. The relators (2) imply that if w_1, w_2 are collected words and the first m positions of w_1 are 0, then the first m positions in the collected word for $w_1 w_2$ are the same as in w_2. Thus, given a list of generators for a subgroup $H \le G$, a noncommutative version of Gaussian elimination can be used to obtain generators $h_1, \ldots, h_k$ for H in row-echelon form. (Besides the usual steps of row reduction, powers and commutators using the newly obtained vectors have to be formed, corresponding to the construction of Schreier generators in Lemma 1.1.) Then each $h \in H$ can be written uniquely in the form

$$h = h_1^{d_{i_1}} h_2^{d_{i_2}} \cdots h_k^{d_{i_k}}, \tag{3}$$

with exponents $0 \le d_{i_j} < |G_{i_j} : G_{i_{j+1}}|$. The procedure can be used for membership testing in H as well, by attempting to factorize any given $g \in G$ as in (3). When membership testing is available, we can compute normal closures of subgroups and derived series and lower central series, and handle homomorphisms. This method can be extended to infinite polycyclic groups, and we shall see an analogous procedure for permutation groups in the next section.

The other basic method is to consider larger and larger factor groups of G. We construct a subnormal chain (1), which is a refinement of a chain of normal subgroups $G = N_1 \rhd N_2 \rhd \cdots \rhd N_m = 1$, with N_i/N_{i+1} elementary abelian. Then we use induction to extend information from G/N_i to G/N_{i+1}. Since G/N_i acts as a group of linear transformations on N_i/N_{i+1}, often linear algebra methods can be used.

Given the usefulness of pcp's, one may try to construct one from other representations. In most applications, the groups arise as finitely presented groups, and there is a collection of algorithms for finding pcp's of certain factor groups.

The first of these was Macdonald's pQ-algorithm for finding p-quotients of a finitely presented group. This was soon turned into a powerful tool by Newman and used for the study of *Burnside groups*. Let F_n be the free group of rank n and q a power of a prime. The Burnside group $B(n,q)$ is the largest factor group of F_n with all elements of order dividing q. $B(n,q)$ may be infinite, but, as was recently proved by Zelmanov, it has a largest finite factor group $\overline{B}(n,q)$. The first big success of the p-quotient method was the determination of $|\overline{B}(4,4)| = 2^{422}$ by Alford, Havas, and Newman. (In this particular instance of Burnside groups, we have $B(4,4) = \overline{B}(4,4)$.) For small primes p, present implementations can determine quotients of Burnside groups of order p^k with k in the thousands.

The family of available quotient group algorithms nowadays includes an NQ-method that determines possibly infinite nilpotent quotients of a finitely presented group by Nickel, two different SQ-methods for finding finite solvable factor groups by Plesken and Niemeyer, and a PCQ-method for finding infinite polycyclic factor groups by Sims in [2] and by Lo in [7]. All of these algorithms proceed by induction. As a

first step, an epimorphism $\varphi : G \to G/G'$ is constructed. A polycyclic presentation for G/G' can be obtained by adding to the relators of G that all pairs of generators commute and computing the cyclic decomposition of the resulting abelian group. Suppose that an epimorphism φ is known onto a nilpotent or polycyclic group H. Then an attempt is made to lift φ to an epimorphism of G onto an extension group K of H by a normal subgroup N that is central in the case of pQ and NQ or just abelian in the other cases.

The methods for finding such N and K and for lifting φ differ in the various quotient methods. In the case of pQ, they amount to solving linear equations over GF(p) that on one hand represent confluence conditions for the pcp presentation and on the other hand stem from the relations for G. In the case of NQ, congruences occur instead of equations, while in the case of SQ's, modular representations of H are considered. Finally, in the case of PCQ's, Gröbner basis techniques come into play. It should be no wonder that the time requirement for NQ, and especially for SQ and PCQ, is much higher than for pQ, and so applications of these are far more restricted.

Permutation Groups

The situation is quite satisfactory in the case of permutation group algorithms as well: most structural properties of permutation groups of reasonable degree and order can be readily computed. Working with permutation representations is usually the method of choice when studying simple groups and their subgroup structure, provided that the simple group has a permutation representation of sufficiently small degree. "Reasonable" and "sufficiently small" degree, of course, depend on the available software and hardware; currently, it is in the low hundred thousands, while the logarithm of the group order may be in the low hundreds.

This is the area of CGT where the complexity analysis of algorithms is the most developed. The reason for that is the connection with the celebrated *Graph Isomorphism Problem*. The decisive result in establishing the connection is Luks's polynomial time algorithm for the isomorphism testing of graphs with bounded valence, where the isomorphism problem is reduced to finding setwise stabilizers of subsets in the permutation domain of groups with composition factors of bounded size. This result not only established a link between complexity theory and CGT but provided new methodology for permutation group algorithms.

Because of the approach from two different points of view, complexity theoretic and practical, algorithms for numerous tasks were developed independently in the two contexts. In the last five years, a remarkable convergence of the approaches occurred: for example, in GAP, most of the permutation group library uses implementations of algorithms with the fastest known asymptotic running times.

The basic ideas of permutation group manipulation are due to Sims. A *base* for $G \le \mathrm{Sym}(\Omega)$ is a sequence $B = (\beta_1, \dots, \beta_m)$ of points from Ω such that the pointwise stabilizer $G_B = 1$. The sequence B defines a subgroup chain

$$G = G^{[1]} \ge G^{[2]} \ge \cdots \ge G^{[m]} \ge G^{[m+1]} = 1 \tag{4}$$

where $G^{[i]} = G_{(\beta_1,\dots,\beta_{i-1})}$. A *strong generating set* (SGS) relative to B is a generating set S for G with the property that

$$\langle S \cap G^{[i]} \rangle = G^{[i]} \quad \text{for } 1 \le i \le m+1. \tag{5}$$

Given an SGS, the orbits $\beta_i^{G^{[i]}}$ and transversals R_i for $G^{[i]}$ mod $G^{[i+1]}$ can be easily computed. As an analogue of (3) in the section "Polycyclic Groups", every $g \in G$ can be written uniquely in the form

$$g = r_m r_{m-1} \cdots r_1 \tag{6}$$

with $r_i \in R_i$. Factorizing elements in this form is called *sifting* and may be considered as a permutation group version of Gaussian elimination. It can be used for membership testing; computing normal closures of subgroups, derived series, and lower central series; handling homomorphisms; and finding the pointwise stabilizer of any subset of Ω.

We sketch the basic idea of an SGS construction. Given $G = \langle T \rangle$, we maintain a sequence $B = (\beta_1, \dots, \beta_k)$ of already-known elements of a base and approximations S_i for a generating set of the stabilizer $G_{(\beta_1,\dots,\beta_{i-1})}$. We say that the data structure is *up-to-date below level* j if

$$\langle S_i \rangle_{\beta_i} = \langle S_{i+1} \rangle \tag{7}$$

holds for all $j < i \le k$. It is easy to see that $\bigcup S_i$ is an SGS for G if and only if it generates G and the data structure is up-to-date below level 0.

In the case when the data structure is up-to-date below level j, we compute a transversal R_j for $\langle S_j \rangle$ mod $\langle S_j \rangle_{\beta_j}$. Then we test whether (7) holds for $i = j$. By Lemma 1.1, this can be done by sifting the Schreier generators obtained from R_j and S_j in the group $\langle S_{j+1} \rangle$. If all Schreier generators are in $\langle S_{j+1} \rangle$, then we say that the data structure is up-to-date below level $j-1$; otherwise we add a nontrivial Schreier generator h to S_{j+1} and say that the data structure is up-to-date below level $j+1$. In the case $j = k$, we also choose a new base point β_{k+1} from the support of h.

We start the algorithm by choosing $\beta_1 \in \Omega$ which is moved by at least one generator in T and setting $S_1 := T$. At that moment, the data structure is up-to-date below level 1; the algorithm terminates when the data structure becomes up-to-date below level 0.

A second generation of algorithms uses divide-and-conquer techniques by utilizing the imprimitivity block structure of the input group, thereby reducing the problems to primitive groups. We mention Luks's and P. Neumann's results on computing a composition series, the Sylow subgroup constructions by Kantor, and the asymptotically fastest deterministic SGS construction by Babai, Luks, and Seress. The latter runs almost a factor n^2 faster than the asymptotically fastest versions of Sims's original method. The required group-theoretic arsenal for these algorithms includes consequences of the classification of finite simple groups and, in the case of the Sylow subgroup constructions, detailed knowledge of the classical simple groups. (We mention, however, that a composition series can be computed without using the simple group classification by a result of Beals.)

The running time of most algorithms for $G = \langle S \rangle \leq S_n$ does not depend only on the input length $|S|n$ but also on the order of G. Moreover, in groups of current interest, it frequently happens that the degree of G is in the tens of thousands or even higher, so even a $\Theta(n^2)$ algorithm may not be practical. On the other hand, $\log |G|$ is often small. Therefore, a recent trend is to search for algorithms with running time of the form $O(n|S|\log^k |G|)$. These are called *nearly linear algorithms*, since their running time is $O(n \log^c n)$ if $\log |G|$ is bounded from above by a polylogarithmic function of n. This happens, for example, for all permutation representations of finite simple groups except the alternating ones.

There is a large library of nearly linear algorithms by Babai, Beals, Cooperman, Finkelstein; Luks and Seress in [7]; Morje in [7]; and Schönert. Roughly, everything that can be done in polynomial time can also be done in nearly linear time in groups which do not have composition factors of exceptional Lie type. The price we pay for the speedup is that most algorithms are random *Monte Carlo*: they may return an incorrect answer, with probability of error bounded by the user. Although there is a recent theoretical result by Kantor and Seress that upgrades all nearly linear time algorithms to Las Vegas (i.e., to guaranteed correct output) for groups with no exceptional Lie type composition factors, the practicability of this method is not demonstrated yet.

At present there are no polynomial time algorithms for some important tasks such as computing the centralizer of elements, the intersection of two subgroups, or setwise stabilizers of subsets of the permutation domain. These tasks are polynomially equivalent, and graph isomorphism can be reduced to them (see Luks in [6]). Therefore, it is suspected that no polynomial time algorithms exist.

Practical algorithms for these problems use *backtrack methods*. Let B be a fixed base of G. The images of base points uniquely determine the elements of G, while the image of an initial segment of B defines a coset of some $G^{[i]}$ (cf. (4)). The images of all initial segments define a partial order by inclusion, which is called the *search tree* for G. Traditional backtrack methods systematically examine the elements of the search tree. Especially valuable is when a node close to the root can be eliminated, because all elements of G less than this node (i.e., elements of the appropriate coset) can be excluded at once. A new generation of backtrack methods was developed by Leon in [3], based on ideas from B. McKay's graph isomorphism testing program *nauty*. Nowadays centralizers can be computed in groups of degree in the tens of thousands, provided that they have a small base. The computation of normalizers of subgroups seems to be harder: in theory, polynomial time equivalence with centralizers is not known, and practical computation is much more complicated.

Representation Theory

As John Conway writes in the introduction of the *Atlas* [5], "the ordinary character table is beyond doubt the most compendious way of conveying information about a group to the skilled reader." However, computing a character table from scratch is not an easy task. Therefore, GAP contains tables for the most frequently used groups, including all tables in the *Atlas* and tables for most of the maximal subgroups of sporadic groups.

A basic idea for computing the complex irreducible characters of a group G is due to Burnside. In the group algebra $\mathbb{C}G$ of G, the sum S_C of the elements of a conjugacy class C is in the center of $\mathbb{C}G$; in fact, taking these sums for all conjugacy classes, we get a basis for $Z(\mathbb{C}G)$. For a fixed conjugacy class C, the products $S_C S_{C_i}$ decompose as a linear combination of the S_{C_j}'s, and the coefficients, taken for all possible C_i, define a matrix M_C. Since the S_{C_j}'s are in $Z(\mathbb{C}G)$, the matrices M_{C_j} commute and are simultaneously diagonalizable. The irreducible characters can be easily computed from the entries in the diagonal forms.

To perform the computation sketched above, we must know the conjugacy classes and the class-multiplication coefficients, which boils

down to an algorithm to determine which conjugacy class a given group element belongs to. Dixon performs the diagonalization over prime fields $\mathbb{Z}_p$ for (large) primes p, and the result is lifted back to $\mathbb{C}$. Schneider in [2] introduces strategies to select a small collection of the M_{C_j}'s from which the new basis for the diagonal form can be computed. The Dixon-Schneider method can be used for groups of moderate order (about $|G| < 10^9$) and with not too many conjugacy classes (in the low hundreds).

For larger groups, we may try to compute the character table interactively (see Neubüser, Pahlings, Plesken in [1], and [9]). We may know some (not necessarily irreducible) characters from a matrix or a permutation representation of G, and there are machine commands to induce characters of subgroups, restrict characters of overgroups, and extend characters of factor groups. We may also take products of known characters (corresponding to tensor products of representations), and, if the power maps for conjugacy classes are known, we can compute the generalized characters defined by $\chi^{(n)}(g) := \chi(g^n)$.

Once a set of characters is collected, the hunt for the irreducible ones may begin. Taking scalar products, we may subtract multiples of the known irreducible characters to decrease the norms of the characters in our collection. Earlier methods tried to split characters as the sum of characters with smaller norms. Nowadays the most effective method seems to be the application of Lovász's basis reduction algorithm to the set of known characters to obtain a basis of the lattice $\mathcal{L}$ of the generalized characters which consists of elements of small norm. Then, following an idea of W. Plesken, we may consider embeddings of the basis vectors of $\mathcal{L}$ into the lattice $\mathbb{Z}^m$ as vectors with the indicated norm and find the standard basis of $\mathbb{Z}^m$, which corresponds to the irreducible characters, as linear combinations of the basis vectors of $\mathcal{L}$. Quite remarkably, despite the numerous possibilities for the embedding, this method usually quickly produces some irreducible characters.

The method of generating characters by hunting for irreducibles can be iterated: when a larger collection of irreducible characters is known, it may be worthwhile to induce characters of smaller subgroups or compute higher tensor powers. Initially, we may restrict the computations to rational characters because of the cost of handling the irrational numbers. When irrational characters are finally computed, we can use the fact that if n is the order of some $g \in G$, then each character value on g can be written as an integer linear combination of the powers of a primitive n^{th} root of unity.

The other large area of representation theory is the theory of modular representations. Brauer characters are even harder to compute than ordinary ones, so GAP also contains a library of Brauer character tables, including all tables in the modular extension of the *Atlas* [8].

In the case of representations over the complex numbers, most computations deal only with the character tables. On the other hand, over finite fields, the emphasis is also on the computation with the representations themselves. One of the reasons is that modular representations occur naturally in different areas of CGT: for example, when considering groups acting on the elementary abelian factors (section "Polycyclic Groups") or when studying matrix groups (section "Matrix Groups").

Dealing with a modular representation $G \leq \mathrm{GL}_d(\mathbb{F})$, a fundamental problem is to find invariant submodules or to prove that G acts irreducibly. This can be done, for example, using a generalization by Holt and Rees of R. Parker's idea, the so-called Meat-Axe. Also, in studying the structure of large dimensional modules, sometimes the *condensation method* (see Ryba in [2], and [9]) can be applied. This means that given $G \leq \mathrm{GL}_d(\mathbb{F})$, we find a subalgebra $A \leq \mathbb{F}G$ and an A-module N of possibly much smaller dimension than M such that the submodule lattices of M and N are isomorphic. For example, Cooperman, Hiss, Lux, and Muller recently condensed a 976841775-dimensional module of the sporadic simple Thompson group to 1403 dimensions.

Matrix Groups

Matrix groups are important and very compact representations of groups, but pose serious computational problems. For groups over the integers, the membership problem is undecidable already in four dimensions. (However, on the positive side, the finiteness of matrix groups over $\mathbb{Z}$ can be determined by a practical polynomial time algorithm of Babai, Beals, and Rockmore.) There are difficulties with matrix groups defined over finite fields as well. Rewriting matrix groups as permutation groups on the underlying vector space, which was the early approach, results in an exponential blowup of the input size. Moreover, two basic ingredients, which were crucial for the efficient handling of solvable and permutation groups, are missing: there may be no subgroup chain with small indices, analogous to (1) or (4), and it is not clear how to generalize Gaussian elimination. Large fields also cause problems, since the discrete logarithm problem arises already for 1×1 matrices. Nevertheless, despite all these difficulties, there is a concentrated attack from both the practical and theoretical sides to determine the

structure of matrix groups over finite fields. Currently, this is the most active area of CGT.

Most effort concentrates on the *matrix recognition project*: given a matrix group $G \le \mathrm{GL}_d(q)$ by a list of generators, find at least one category of Aschbacher's classification to which G belongs. Aschbacher's theorem classifies subgroups of $\mathrm{GL}_d(q)$ into nine categories: roughly, modulo scalars G is an almost simple group or isomorphic to a subgroup of $\mathrm{GL}_d(q')$ for some $q'|q$, or there is a normal subgroup of G naturally associated with the action of G on the underlying vector space. Once a category is found, it can be used to explore the structure of the group further. For example, if the group acts imprimitively and we find a decomposition $V = V_1 \oplus V_2 \oplus \cdots \oplus V_m$ of the underlying vector space such that the V_i are permuted by G, then we can construct a homomorphism $\varphi : G \to S_m$. The image of φ is a permutation group of low degree, so it can be handled easily. In this case, the "naturally associated" normal subgroup is the kernel of φ; we can obtain generators for it and apply our methods recursively to the action on the lower-dimensional vector spaces V_i.

The first subproblem solved both in theory and practice is the recognition of the classical almost simple groups in their natural representation. Neumann and Praeger introduced the basic ideas. We say that a number is $\mathrm{ppd}(q,d)$ if it has a prime divisor p such that $p|q^d - 1$ and $p \nmid q^i - 1$ for $1 \le i \le d-1$. Neumann and Praeger give precise estimates for the proportion of elements in groups $G \ge \mathrm{SL}_d(q)$ whose order is $\mathrm{ppd}(q,d)$ and for those whose order is $\mathrm{ppd}(q,d-1)$. It turns out that among $5.5d$ randomly and independently chosen elements, both types occur with probability >0.99. On the other hand, the (short) list of subgroups containing both types of elements but not containing $\mathrm{SL}_d(q)$ is determined, and special tests are devised to eliminate them. This method was extended by Niemeyer and Praeger in a highly nontrivial way to the other classical groups. An alternative approach is described by Celler and Leedham-Green in [7]. Here, random elements of G are chosen, their order is determined, and groups of the Aschbacher classification which cannot contain elements of this order are eliminated.

The classical matrix groups can also be recognized *constructively*; the output is not only a proof that G is a classical group, but every element of G can be expressed effectively in terms of the given generating set. Celler and Leedham-Green handle the case $G \ge \mathrm{SL}_d(q)$ in the natural representation; Cooperman, Finkelstein, and Linton in [7] handle the case $G = \mathrm{SL}_d(2)$ in *any* representation; while Kantor and Seress handle all classical groups in any representation.

Concerning the other cases of the Aschbacher classification, the Meat-Axe (cf. section "Representation Theory") can be used to test irreducibility and absolute irreducibility of G. There are also programs by Holt, Leedham-Green, O'Brien, and Rees computing imprimitivity and tensor decompositions. The exploration of groups beyond classifying them in one of the Aschbacher classes has also started. The implementations of the algorithms indicated in the previous three paragraphs can handle groups of dimension in the low hundreds over moderately sized fields ($q < 2^{16}$).

While randomization can often be used to speed up permutation group algorithms, it seems to be an indispensable tool for the study of matrix groups. In theory, concerning polynomial time construction the situation is satisfactory: Babai gives a Monte Carlo algorithm which, after a preprocessing phase consisting of $O(\log^5 |G|)$ group operations, constructs independent, nearly uniformly distributed elements at a cost of $O(\log |G|)$ group operations per random element. (Nearly uniform means that each element of G is selected with probability between $(1-\varepsilon)/|G|$ and $(1+\varepsilon)/|G|$ for some small ε.) In practice, the heuristic product replacement algorithm of Celler, Leedham-Green, Murray, O'Brien, and Niemeyer is used. This starts with a list $(g_1, g_2, \ldots, g_m)$ of generators; at each step, two indices i, j are selected randomly and one of g_i, g_j is replaced by the product $g_i g_j$. After K replacements as preprocessing, we start to output the newly created elements of the list as random elements of G. It is shown that if m is at least twice the size of a minimal generating set of G, then the limit distribution of the sequence $(g_1, g_2, \ldots, g_m)$ is uniform among all generating sequences of length m, and computational evidence shows that in applications for matrix groups of dimension in the low hundreds, K can be chosen around 100. Although the constructed random elements are not independent, the ratio of elements with properties relevant for the algorithms (for example, the ppd property) seems in experimental tests to be close to the ratio of such elements in the group.

We mention that some of the algorithms described in this section work in the more general context of *black box groups* (cf. the extended version of this survey on the World Wide Web).

Applications

We finish this survey by briefly mentioning some available databases, applications of CGT, and stand-alone programs.

In the nineteenth century a central problem of group theory was the classification of groups of a given order. Currently, GAP contains a library of the 174366 groups of order at most 1000, but

not 512 or 768 constructed by Besche and Eick. Among these, the 56092 groups of order 256 were found by O'Brien, and it is expected that there are more than a million groups of order 512.

The 42038 transitive permutation groups of degree at most 31, constructed by Hulpke, are also available. In principle this list can be used to compute Galois groups; the computations are practical up to degree 15. Nonaffine primitive permutation groups are listed up to degree 1000 by Dixon and Mortimer and solvable permutation groups up to degree 255 by Short. The list of 4-dimensional space groups by Brown, Bülow, Neubüser, Wondratschek, and Zassenhaus and conjugacy class representatives of irreducible maximal finite subgroups of $GL_n(\mathbb{Q})$ for $n \leq 24$ by Nebe and Plesken are now also online. In the section "Representation Theory" we already mentioned the availability of ordinary and modular character tables. Probably just the sizes of these databases indicate the computational difficulties obtaining them.

The applications of CGT in group theory are too numerous to list here. We just mention one of them: the construction of some of the sporadic finite simple groups by Sims and Norton.

Both GAP and MAGMA support the application of CGT to related fields such as other areas of abstract algebra, graph theory, and coding theory. MAGMA covers other areas of symbolic computation as well; it has a particularly strong number theory component. Finally, we mention two stand-alone programs that may be used to investigate finitely presented groups. QUOTPIC by Holt and Rees is a user-friendly graphical interface to display factor groups of fp-groups, while the system Magnus, in the beta-release stage, works with infinite fp-groups.

References

[1] MICHAEL D. ATKINSON, ed., *Computational group theory*, Academic Press, London, New York, 1984; Durham, 1982.

[2] JOHN CANNON, ed., *Computational group theory I*, J. Symbolic Comput. **9** (5-6) (1990).

[3] ____ , *Computational group theory II*, J. Symbolic Comput. **12** (4-5) (1991).

[4] JOHN CANNON and GEORGE HAVAS, *Algorithms for groups*, Australian Comput. J. **24** (1992), 51–60.

[5] J. H. CONWAY, R. T. CURTIS, S. P. NORTON, R. A. PARKER, and R. A. WILSON, *Atlas of finite groups*, Clarendon Press, Oxford, 1985.

[6] LARRY FINKELSTEIN and WILLIAM M. KANTOR, eds., *Groups and computation*, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 11, Amer. Math. Soc., Providence, RI, 1993.

[7] ____ , *Groups and computation II*, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 28, Amer. Math. Soc., Providence, RI, 1997.

[8] CHRISTOPH JANSEN, KLAUS LUX, RICHARD PARKER, and ROBERT WILSON, *An atlas of Brauer characters*, London Math. Soc. Monographs (N. S.), vol. 11, Clarendon Press, Oxford, 1995.

[9] KLAUS LUX and HERBERT PAHLINGS, *Computational aspects of representation theory of finite groups* (G. O. Michler and C. M. Ringel, eds.), Representation Theory of Finite Groups and Finite-Dimensional Algebras, Progr. Math., vol. 95, Birkhauser-Verlag, 1991, pp. 37–64.

[10] J. NEUBÜSER, *An elementary introduction to coset table methods in computational group theory* (C. M. Campbell and E. F. Robertson, eds.), Groups – St. Andrews 1981, London Math. Soc. Lecture Note Ser., vol. 71, Cambridge Univ. Press, 1982, pp. 1–45.

[11] ____, *An invitation to computational group theory* (C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, eds.), Groups '93 – Galway/St. Andrews, London Math. Soc. Lecture Note Ser., vol. 212, Cambridge Univ. Press, 1995, pp. 457–475.

[12] ÁKOS SERESS, *Nearly linear time algorithms for permutation groups: An interplay between theory and practice*, Acta Appl. Math. (1997), to appear.

[13] CHARLES C. SIMS, *Computation with finitely presented groups*, Cambridge Univ. Press, 1994.