Computing automorphism groups

Ákos Seress

June 2011, Istanbul

Two basic ways to input a group:

1) generators–relators

$G = \langle a, b \mid a^2 = b^5 = 1, b^a = b^{-1} \rangle$

2) "concrete" representation, with generating permutations or matrices (over finite fields)

$G = \langle (1,5,2,6), (1,2)(3,4)(5,6) \rangle$

Theorem (Cayley)

Every finite abstract group can be described as a finitely presented group, and as a permutation group or matrix group.

Representations given by Cayley's theorem (Cayley tables, regular representations) are useless from a computational point of view. We want to work with groups that are too big to list.

Which representation to choose?

Finitely presented groups

Finitely presented groups are tough customers. Given a presentation $G = \langle X \mid R \rangle$, the fundamental question is the solution of the word problem: does a given word w in X represent the identity of G?

Theorem (Novikov, Boone)

The word problem is undecidable (no recursive algorithm to solve).

To handle a finitely presented group, we usually try to construct a permutation representation and switch to permutation group methods.
One notable exception: Finite solvable groups have polycyclic presentations and the word problem is solvable. Many further algorithms, including automorphism group computations. In the special case of p-groups, more efficient versions.

Permutation and matrix groups

Decidability is not a problem: everything is finite (provided the matrix groups are defined over finite fields).

For permutation groups, large library of efficient nearly linear-time algorithms, including many subroutines needed for automorphism group computations. However,
gap> AutomorphismGroup(G);
does not use the best theoretical algorithm.

Matrix group algorithms are less developed. So far, there is no practical attempt to compute automorphism groups, but the same algorithm as in the permutation group case works, provided we have subroutines for certain tasks, as in the previous paragraph.

Automorphism group computation is the most complex permutation group algorithm so far. Along the way, we have to switch to matrix representations, use polycyclic representation methods for certain factor groups.
To describe the algorithm from scratch would require a semester-long course; necessarily, we have to handwave a lot.

We encounter many interesting algorithmic and mathematical problems that are useful for a broader spectrum of algorithmic tasks.

Automorphism groups

Example

- $G = \langle (1,2),(3,4),(5,6)\rangle \cong \mathbb{Z}_2^3$, $\mathrm{Aut}(G) \cong \mathrm{GL}(3,2)$. Computations lead out from the input permutation representation.
- $$G = \langle (1,2)(3,4)(5,6)(7,8),(1,3)(2,4)(5,7)(6,8),\\ (1,5)(2,6)(3,7)(4,8)\rangle \cong Z_2^3$$

  ("useless" regular representation).

  $\mathrm{Aut}(G) \cong N_{S_8}(G)/C_{S_8}(G)$ computable in the input representation, but computing the normalizer in S_n of a regular subgroup is a notoriosly difficult case of normalizer computations.

We describe automorphisms by giving the images of generators of G.
$G = \langle X \rangle = \langle x_1, \ldots, x_k \rangle$,
$\mathrm{Aut}(G) = \langle \varphi_1, \ldots, \varphi_m \rangle$,
$\varphi_i : [x_1, \ldots, x_k] \mapsto [y_{i,1}, \ldots, y_{i,k}]$ for some $y_{i,j} \in G$.

To find the φ_i-image of an arbitrary $g \in G$, we have to solve the constructive membership problem in G: express g in terms of X (as a straight-line program). Readily done in permutation groups and polycyclic groups; recent major progress in matrix groups.

To find the image of $g \in G$ under an arbitrary $\varphi \in \mathrm{Aut}(G)$, solve the constructive membership problem in $\mathrm{Aut}(G)$, combine with method in previous paragraph.

Straigth-line programs

straight-line program from Y to g

sequence of expressions $w_1, \ldots, w_m$
w_i: symbol for some $y \in Y$ or
$w_i = (w_j, w_k)$ for some $j, k < i$ or
$w_i = (w_j, -1)$ for some $j < i$

Evaluation:
$\mathrm{eval}(w_j, w_k) = \mathrm{eval}(w_j)\mathrm{eval}(w_k)$
$\mathrm{eval}(w_j, -1) = \mathrm{eval}(w_j)^{-1}$
$\mathrm{eval}(w_m) = g$

$Y = \{y\}, \quad g = y^{1024}$
$w_1 = y, \ w_2 = (w_1, w_1), \ w_3 = (w_2, w_2), \ldots, w_{11} = (w_{10}, w_{10})$

Group actions and orbits

Let G be a group acting from the right on a set X:

$$A : X \times G \to X \qquad (\text{"action function"})$$

with $A(x,1) = x$ and $A(x,gh) = A(A(x,g),h)$ for all $x \in X$ and all $g, h \in G$.

Notation

Write xg for $A(x,g)$ and xgh for $A(x,gh) = A(A(x,g),h)$.
Write xG for $\{xg \mid g \in G\}$ and call A transitive if $xG = X$.
Call xG the orbit of x under G.
Call $\mathrm{Stab}_G(x) := \{g \in G \mid xg = x\} \leq G$ the stabilizer.
Write gH for $\{gh \mid h \in H\}$ and Hg for $\{hg \mid h \in H\}$.

Example

For $H < G$,

- H acts on G by $A : G \times H \to G, (g,h) \mapsto gh$.
- G acts on the right cosets $X := \{Hg \mid g \in G\}$ by $A : X \times G \to X, (Hg, g') \mapsto Hgg'$.

Orbit-Stabilizer Theorem

Theorem (Orbit-Stabilizer)

Let G act transitively on X and let $S := Stab_G(x)$ for some $x \in X$. Then $|G| = |X| \cdot |S|$ and

$$\begin{array}{ccc} \{Sg \mid g \in G\} & \longrightarrow & X \\ Sg & \longmapsto & xg \end{array}$$

is well-defined and is a bijection.

Proof:

- If $Sg = Sg'$ then $g' = sg$ for some $s \in S$, thus $xg = xsg = xg'$. So F is well-defined.
- F is surjective, because the image is the orbit xG.
- If $xg = xg'$, then gg'^{-1} fixes x and thus lies in S. Thus $g' = sg$ for some $s \in S$ and F is injective.

Algorithm: EnumerateOrbit

Input: $G = \langle g_1, \ldots, g_k \rangle$ acting on X and $x \in X$.

1. Assign list $L := [x]$ and $i := 1$
2. While $i \leq$ Length(L) do
3. For j in $[1, 2, \ldots, k]$ do
4. Assign $y := L[i]g_j$
5. If $y \notin L$ then
6. Append y to the end of L
7. Assign $i := i + 1$

Fact (Correctness and termination)

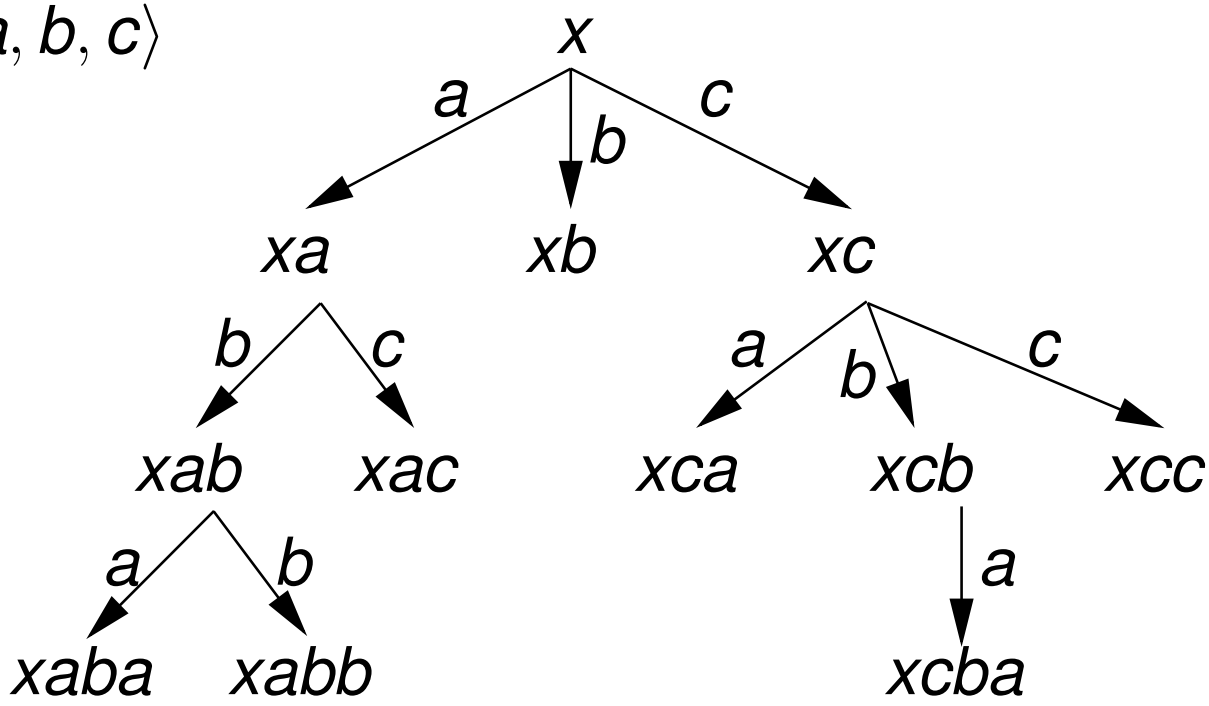
If this terminates, then L contains the complete orbit xG.
If xG is finite, then the algorithm terminates.

Comment (Performance)

Crucial: Check efficiently if $y \notin L$.
$\Longrightarrow$ use hashing technique

Breadth first search — Schreier tree

$G = \langle a, b, c \rangle$



Tree is discovered row by row.

For each point we get a word in the generators.

These words are shortest possible!

Computing the stabilizer

Assume we run the standard orbit algorithm for $X = xG$.

Fact (Schreier generators)

Whenever we apply a generator g to a point xw (w a word in the generators) and find that $y := xwg$ is already known, it must be of the form xw' for a known word w'. Then wgw'^{-1} fixes x and thus is contained in $Stab_G(x)$.

Theorem (Schreier's Lemma)

All these wgw'^{-1} together generate $Stab_G(x)$.

Problem

There can be many such Schreier generators.

The Orbit-Stabilizer-Algorithm

Algorithm: EnumerateOrbitWithStabilizer

Input: $G = \langle g_1, \ldots, g_k \rangle$ acting on X and $x \in X$.

1. Assign list $L := [x]$ and $i := 1$ and $S := \{1\}$
2. While $i \leq$ Length(L) do
3. For j in $[1, 2, \ldots, k]$ do
4. Assign $y := L[i]g_j$
5. If $y \notin L$ then
6. Append y to the end of L
7. else
8. Assign $S := \langle S, wg_jw'^{-1} \rangle$
9. Assign $i := i + 1$

Permutation Groups

Notation: Let Σ_n be the group of all permutations of $[n] := \{1, 2, \ldots, n\}$.

We use cycle notation:

$(1, 3, 4)(2, 5)$ maps $1 \mapsto 3 \mapsto 4 \mapsto 1$ and $2 \mapsto 5 \mapsto 2$.

We concatenate left before right:

$(1, 2)(2, 3) = (1, 3, 2)$.

Orbits and stabilizer cosets

Theorem (Orbit-Stabilizer)

Let G act on X and let $S := Stab_G(x)$ for some $x \in X$. Let $S \setminus G := \{Sg \mid g \in G\}$. Then $|G| = |xG| \cdot |S|$ and

$$\begin{array}{rccc} F: & S\setminus G & \longrightarrow & xG \\ & Sg & \longmapsto & xg \end{array}$$

is well-defined and is a bijection.

Fact

We can read off in which S-coset g lies by looking at xg.

Bases and Strong Generating Sets

(C. Sims, late 1960's)
Let $G = \langle T \rangle \leq \Sigma_n$, that is, G acts on $[n] = \{1, 2, \ldots, n\}$.

Definition (Base)

$X = (x_1, \ldots, x_k) \subset [n]$ is base for G:
pointwise stabilizer $G_X = 1$
$G = G^{[1]} \geq G^{[2]} \geq \cdots \geq G^{[k+1]} = 1$ point stabilizer chain
$G^{[i]} = G_{(x_1,\ldots,x_{i-1})}$

Definition (Strong Generating Set)

S is strong generating set (SGS):
S is a generating set for G and for
$S_i := G^{[i]} \cap S$,
$G^{[i]} = \langle S_i \rangle$

Example of base and SGS

$G = \langle (1,5,2,6), (1,2)(3,4)(5,6) \rangle$
$X = (1,3)$
$G > G_1 > G_{13} = 1$

$T = \{(1,5,2,6), (1,2)(3,4)(5,6)\}$ is not an SGS
$T \cap G_1 = \emptyset$, does not generate $G_1 \neq 1$.

$S := \{(1,5,2,6), (1,2)(3,4)(5,6), (3,4)\}$ is an SGS

The full data structure: base, SGS, transversals

$G = \langle T \rangle \leq \Sigma_n$, with

base $X = (x_1, \ldots, x_k) \subset [n]$ and SGS S,
point stabilizer chain $G = G^{[1]} \geq G^{[2]} \geq \cdots \geq G^{[k+1]} = 1$
$S_i = S \cap G^{[i]}$, $G^{[i]} = \langle S_i \rangle$

If a base and SGS are known:
For each $i \leq k$, we can compute the orbit
$O_i := x_i G^{[i]} = x_i \langle S_i \rangle$ and Schreier trees coding how to access elements of O_i from x_i.
In particular, we have transversals T_i for $G^{[i]}$ mod $G^{[i+1]}$ and we know the orbit lengths $\ell_i := |x_i \langle S_i \rangle|$.

The group order

Since we know all orbit lengths, we know the indices $[G^{[i]} : G^{[i+1]}] = |G^{[i]}|/|G^{[i+1]}|$.

Fact

We know the group order

$$|G| = \underbrace{[G^{[1]} : G^{[2]}]}_{\ell_1} \cdot \underbrace{[G^{[2]} : G^{[3]}]}_{\ell_2} \cdot \cdots \cdot \underbrace{[G^{[k]} : G^{[k+1]}]}_{\ell_k}.$$

Decomposition of group elements

Even better, the orbit algorithm provided Schreier trees and thus words in the stong generators to reach the orbit points.

These words form transversals: We have elements $t_j^{(i)}$ for $1 \le i \le k$ and $1 \le j \le \ell_i$ with:

$$G^{[i]} = \bigcup_{j=1}^{\ell_i} G^{[i+1]} t_j^{(i)} \qquad \text{(disjoint union)}.$$

Therefore, $g \in G$ can be written uniquely in the form

$$g = t_{j_k}^{(k)} \cdot t_{j_{k-1}}^{(k-1)} \cdots t_{j_1}^{(1)}$$

for some numbers $j_1, j_2, \ldots, j_k$ with $1 \le j_i \le \ell_i$ for all i.

Yet better, the stabilizer chain allows us to read off these numbers!

Sifting

Assume $g \in G$, thus:

$$g = t_{j_k}^{(k)} \cdot t_{j_{k-1}}^{(k-1)} \cdots t_{j_1}^{(1)}$$

for $j_1, j_2, \ldots, j_k$ with $1 \le j_i \le \ell_i$ for $1 \le i \le k$.

Since the first $k-1$ of these lie in $G^{[2]}$, they all fix x_1.

Thus $x_1 g \in O_1$ depends only on j_1 and not on the other j_i!

So, we compute $x_1 g \in O_1$, look it up and determine j_1.

Now

$$g_2 := g t_{j_1}^{(1)^{-1}} = t_{j_k}^{(k)} \cdot t_{j_{k-1}}^{(k-1)} \cdots t_{j_2}^{(2)}$$

fixes x_1 and thus lies in $G^{[2]}$.

We can now compute $x_2 g_2 \in O_2$ and determine $j_2, \ldots, j_k$ inductively.

This procedure is called sifting.

Membership test

If we sift an element $g \notin G$ (for example, if $G \leq \Sigma_n$ and $g \in \Sigma_n \setminus G$), then something will go wrong:

- if $x_1 g \notin O_1 = x_1 G$ then we proved that $g \notin G$,
- if $x_1 g = x_1 t_j^{(1)}$ for some $1 \leq j \leq \ell_1$ then,

  since $t_j^{(1)} \in G$ and $x_1 g t_j^{(1)\,-1} = x_1$, we have

  $$g \in G \iff g t_j^{(1)\,-1} \in G^{[2]}.$$

  $\Longrightarrow$ Inductively, we test membership in the stabilizer.

Theorem (Stabilizer chain and sifting)

Given $G \leq \Sigma_n$, an element $g \in \Sigma_n$ and a stabilizer chain for G with its orbits and Schreier trees, we can sift g, and

- *either prove that $g \notin G$,*
- *or write g constructively in a unique way as product of transversal elements*

$$g = t_{j_k}^{(k)} \cdot t_{j_{k-1}}^{(k-1)} \cdots t_{j_1}^{(1)}.$$

How to compute a stabilizer chain

The basic idea:

Given $G = \langle T \rangle \leq \Sigma_n$, we can pick the first base point as any $x_1 \in [n]$ that is not fixed by T.

The Orbit-Stabilizer-Algorithm computes the first orbit O_1 and transversal T_1 of the data structure, and Schreier generators for the group $G^{[2]} = G_{x_1}$.

Pick $x_2 \in [n]$ not fixed by $G^{[2]}$; repeat.

Problem: Blowup in the number of Schreier generators. It is possible to reorganize the procedure to avoid the blowup, and obtain a polynomial-time algorithm.

Theorem (Sims)

Let $G = \langle T \rangle \leq \Sigma_n$. A base and strong generating set for G can be computed in time bounded by

$$O(n^2 \log^3 |G| + |T| n^2 \log |G|).$$

Theorem (Babai, Cooperman, Finkelstein, Seress)

Let $G = \langle T \rangle \leq \Sigma_n$ and d an arbitrary constant. A guess for a strong generating set for G can be computed in time bounded by

$$O(n \log n \log^4 |G| + |T| n \log |G|)$$

so that the probability for a wrong output is $\leq 1/n^d$.

Complexity measure of algorithms

time unit: computing the image xg of some $x \in [n]$ under some $g \in \Sigma_n$

memory unit: storing an integer $\leq n$.

Input length: $|T|n$
Running time (and memory usage) depends not only on input length, but on the size of the object the input generates.

In addition, the BCFS algorithm is randomized, and it may give incorrect answer (with probability controlled by the user). Still, preferable when $\log |G|$ is small compared to n.

No worry in GAP: in default setting, all permutation group algorithms give guaranteed correct answers (using various verification procedures to check the output).

Classification of randomized algorithms

randomized algorithm is Monte Carlo: output may be wrong, with probability controlled by user

randomized algorithm is Las Vegas: output is always correct, may report failure with probability controlled by user

Small-base groups

Definition (Small-base family)

A family $\mathcal{F}$ of permutation groups is called a family of small-base groups, if there is a constant c such that each $G \in \mathcal{F}$ of degree n satisfies $\log |G| \leq \log^c n$.

Fact

For any non-redundant base X for a group G, (i.e. the stabilizer chain defined by X is strictly decreasing),

$$\frac{\log |G|}{\log n} \leq |X| \leq \log |G|.$$

Example (Small-base families)

- All permutation representations of (almost) simple groups, except alternating and symmetric groups.
- Primitive groups G, unless $\mathrm{Soc}(G)$ contains large alternating composition factors.

Nearly linear-time algorithms

Definition (Nearly linear-time)

A permutation group algorithm is called nearly linear-time if for inputs $G \leq \Sigma_n$, the running time is $O(n \log^c |G|)$ for some absolute constant c.

For small-base group inputs, nearly linear-time algorithms run in $O(n \log^{c'} n)$ time, which is the computer science definition of nearly linear time.

In practice, most computations are performed with small-base group inputs.
In general, $\log |G|$ can be proportional to n (or even slightly worse). Nearly linear-time algorithms still run in polynomial time on such inputs, but there are more efficient algorithms to handle arbitrary inputs.

Black-box groups

Definition

Black-box group representation of a group G:

- Group elements are represented by words in an alphabet A.
- Given $g, h \in G$, oracle to compute words representing gh, g^{-1}, and decide $g = 1$?

Definition

A black-box group algorithm is an algorithm that uses only black-box operations.

Why lose information?

(i) sometimes we cannot use more info (e.g. random element generation).

(ii) Permutation group elements as words in strong generators: faster group operation than permutation multiplication.

Permutation groups as black-box groups

Suppose a base X, an SGS S, and Schreier trees coding the transversals T_i in the stabilizer chain were computed for some $G \leq \Sigma_n$.

Images of base points determine the elements of G uniquely.
If $X^g = [x_1^g, \ldots, x_k^g] = X^h = [x_1^h, \ldots, x_k^h]$ for some $g, h \in G$ then gh^{-1} fixes X pointwise, implying $g = h$.

X^g is a very short encoding of $g \in G$ but: given X^g and X^h, we cannot easily compute X^{gh}.

Compromise: Each element of G has normal form

$$g = t_{j_k}^{(k)} \cdot t_{j_{k-1}}^{(k-1)} \cdots t_{j_1}^{(1)} \qquad (*)$$

for $j_1, j_2, \ldots, j_k$ with $1 \leq j_i \leq \ell_i$ for $1 \leq i \leq k$. Writing the $t_{j_i}^{(i)}$ as words in the strong generators, we get a black-box group representation of G over the alphabet S.

Representation of group elements is unique. Need to show how to compute products and inverses. To this end, we also need to compute and store the permutations $S^{-1} := \{s^{-1} \mid s \in S\}$. In fact, we may assume $S = S^{-1}$.

We solve a slightly more general problem. Given any word w over the alphabet $S \cup S^{-1}$ that multiplies to some $g \in G$, we can compute the normal form $(*)$ of g (and w) by sifting as a word:

Sifting, revisited

Algorithm: SiftAsWord

Input: Base X, SGS $S = S^{-1}$, and transversals T_i for the point stabilizer chain, encoded as Schreier trees with labels in S. A word w over the alphabet S.

1. Assign $z := w$ and $L := []$
2. For $1 \leq i \leq \mathrm{Length}(X)$ do
3. Compute $y_i := x_i z$ by tracing images of points along z
4. Look up $t_{j_i}^{(i)} \in T_i$ with $x_i t_{j_i}^{(i)} = y_i$
5. Append $\left(t_{j_i}^{(i)}\right)^{-1}$ to z
6. Add $t_{j_i}^{(i)}$ to L
7. return the concatenation of elements of L in reverse order

Permutation groups as black-box groups, II

Given the normal forms w_g, w_h of $g, h \in G$, sifting as words the concatenation $w_g w_h$ and w_g^{-1} gives the normal forms of gh and g^{-1}.

Advantage: The BCFS algorithm computes an SGS so that depth of Schreier trees for the T_i is $O(\log |G|)$.

With that data structure, group operations in the black-box group G take $O(\log^c |G|)$ time. For small-base groups, group operations are much faster than permutation multiplication. In a nearly linear-time algorithm, we are allowed to use $O(n)$ group operations.

Disadvantage: In the black-box representation of G, we lose all information stored in the cycle structure and action of elements. We cannot compute element orders, orbits, blocks of imprimitivity, etc. Only pure black-box algorithms are allowed.

Composition series: credits

First algorithm to use the classification of simple groups (CFSG) (Luks, 1981) but there is an elementary version (Beals).

Many versions, contributions by Babai, Beals, Cannon, Holt, Kantor, Luks, Neumann, Seress.

Composition series: reduction to the perfect, primitive case

To compute a composition series for some $G = \langle T \rangle \leq \Sigma_n$, we may apply homomorphisms

$$\mathrm{Ker}(\varphi) \leftarrow G \xrightarrow{\varphi} H$$

for an appropriate $\varphi : G \to H$ to break up the problem to two smaller problems on $\mathrm{Ker}(\varphi)$ and $\mathrm{Im}(\varphi)$.

Example (Combinatorial reductions)

- G is intransitive, $H = \mathrm{Sym}(O)$ for an orbit O of G.
- G is transitive but imprimitive, $H = \mathrm{Sym}(\mathcal{B})$ for a block system $\mathcal{B} = B_1 \cup \cdots \cup B_m$ of G.

Further reduction: G primitive but not perfect. Compute G', fill up G/G' with cyclic composition factors.

Remaining case: G primitive and perfect. We ran out of cheap reductions.

Primitive groups

Definition

A block system for a transitive $G \leq \Sigma_n$ is a partition $\mathcal{B}$ of $[n] = B_1 \cup B_2 \cup \cdots \cup B_k$ so that each $g \in G$ permutes the parts in $\mathcal{B}$.
The group G is called primitive if the only block systems for G are the trivial ones: $|\mathcal{B}| = 1$ or $|\mathcal{B}| = n$.

Equivalent definition: G is transitive, point stabilizers are maximal subgroups of G. More generally, in a transitive group $G \leq \Sigma_n$, blocks containing a fixed $\omega \in [n]$ are in one-to-one correspondence with the subgroups H containing the point stabilizer G_ω.

Primitive groups, II

Fact

Nontrivial normal subgroups of a primitive $G \leq \Sigma_n$ are transitive on $[n]$.

More generally, in a transitive group $G \leq \Sigma_n$, the orbits of a normal $N \lhd G$ are blocks of imprimitivity for G.

O'Nan–Scott theorem: description of primitive groups based on their socle, and the point stabilizer of the socle.

We need some preparatory steps.

Centralizers

Fact

- $N \lhd G \implies C_G(N) \lhd G$.
- $G \leq \Sigma_n$ *transitive* $\implies C := C_{\Sigma_n}(G)$ *is semiregular:* $C_\omega = 1$ *for all* $\omega \in [n]$. *Moreover,* $\omega C =$ *fixed point set of* G_ω.

Corollary

- G *transitive, abelian* $\implies C_{\Sigma_n}(G) = G$ *and* G *is regular.*
- $G \leq \Sigma_n$ *primitive* $\implies Z(G) = 1$, *unless* n *is prime and* $G = Z_n$.
- $G \leq \Sigma_n$ *primitive,* $N \lhd G$, $C_G(N) \neq 1 \implies N, C_G(N)$ *are regular.*

Subdirect products

Definition

$T_1, \ldots, T_m$ are nonabelian simple groups.
$G \leq T_1 \times \cdots \times T_m$ is called a subdirect product if $\pi_i(G) = T_i$ for all $i \leq m$ (π_i: projection to i^{th} coordinate).

Fact

If $G \leq T_1 \times \cdots \times T_k$ is a subdirect product then there exists a partition $[m] = I_1 \cup \cdots \cup I_\ell$:

$$G = \mathrm{Diag}_{i \in I_1}(T_i) \times \cdots \times \mathrm{Diag}_{i \in I_\ell}(T_i).$$

In particular, all T_i in the same index set I_k are isomorphic.

Socles

Definition

The socle $\mathrm{Soc}(G)$ of a group G is the subgroup generated by all minimal normal subgroups of G.

Fact

- *Minimal normal subgroups are characteristically simple: products of isomorphic simple groups.*
- *Minimal normal subgroups intersect trivially, and centralize each other.*

The O'Nan–Scott theorem

Proven in the late 1970's; Jordan should have known it ...

Let $G \leq \Sigma_n$ be primitive.

Case 1: There exists a minimal normal subgroup N of G, $C_G(N) \neq 1$. Then N is regular.

(i) If N is abelian then $\mathrm{Soc}(G) = N = C_G(N) \cong Z_p^d$, $n = p^d$.
N can be identified with the vector space $\mathrm{GF}(p)^d$, point stabilizer G_ω is an irreducible matrix group $G_\omega \leq \mathrm{GL}(d, q)$.

(ii) If N is nonabelian then there are exactly two minimal normal subgroups of G, and they are isomorphic: left- and right-regular representations of $T_1 \times \cdots \times T_m$, $T_i \cong T$ for some nonabelian simple T, $n = |T|^m$.

The O'Nan–Scott theorem, II

Case 2: For all N minimal normal in G, $C_G(N) = 1$. Then G has unique minimal normal subgroup

$$\mathrm{Soc}(G) = N \cong T_1 \times \cdots \times T_m,$$

$T_i \cong T$ for some nonabelian simple T.

G acts transitively on $\{T_1, \ldots, T_m\}$ by conjugation, $G \leq \mathrm{Aut}(T) \wr \Sigma_m$. $m = 1$ is possible.

(i) $N_\omega = 1$. Then N is regular; cannot happen for all T, m. Smallest possibility: $n = 60^6 = 46656000000$.

In practice, we do not meet such G; in theory: frequently G can be handled by fact that $G \leq H$, H primitive with two regular normal subgroups.

The O'Nan–Scott theorem, III

(ii) N_ω is a subdirect product of N.
$m = k\ell$, $n = |T|^{(k-1)\ell}$, and after renumbering of the T_i

$$N_\omega = \mathrm{Diag}(T_1 \times \cdots \times T_k) \times \cdots \times \mathrm{Diag}(T_{m-k+1} \times \cdots \times T_m).$$

(iii) $N_\omega = (T_1)_\omega \times \cdots (T_m)_\omega$,
$1 < (T_i)_\omega < T_i$, $n = |T_1 : (T_1)_\omega|^m$.

Large-base primitive groups

Theorem (Cameron (1981), using CFSG)

$G \leq \Sigma_n$, G primitive $\Longrightarrow$ G is small-base, unless:
$n = \binom{r}{k}^m$, $\mathrm{Soc}(G) = A_r^m$, $G \leq \Sigma_r \wr \Sigma_m$
$[n]$ can be identified with m-sequences of k-subsets of $[r]$.

Case 2(iii) of the O'Nan–Scott theorem, with T alternating, and $|T_1 : (T_1)_\omega|$ small.

Detection of large alternating factors

First question: Is G a giant (alternating or symmetric in natural action)?

Theorem (Jordan)

If a primitive group $G \leq \Sigma_n$ contains a p-cycle for some prime $3 \leq p \leq n-3$ then G contains A_n.

Algorithm: TESTGIANT

Input: $G = \langle T \rangle$ acting transitively on $[n]$

repeat $c \log n$ times

$r := PseudoRandom(G)$

If Order(r) is divisible by some prime $n/2 < p < n - 2$

then return G is giant

else return G is probably not giant

Proof of correctness:

If "good" r is found then $s := r^{(n-p)!}$ is a p-cycle

s cannot act on non-trivial block system

output "giant" is guaranteed to be correct

If G is giant then a random element of G is good with probability $\approx 1/\log n$

output "not giant" is probably correct

Detection of large alternating factors, II

If G is not giant, test for Cameron-type large-base groups:

Construct mr subsets of $[n]$ such that G permutes them as the natural imprimitive action of $\Sigma_r \wr \Sigma_m$.

Can be done by considering orbital graphs of G (action on $[n] \times [n]$). If construction fails: with high probability, G is a small-base, primitive, perfect group.

Composition series

Input: $G \leq \Sigma_n$ primitive, perfect (small-base).

Major reduction step:

- Find generators for a proper normal subgroup; or
- Find a faithful permutation representation of G on $\leq n/2$ points; or
- Prove that G is simple.

If $N \lhd G$ is found: compute index i in point stabilizer chain $G = G^{[1]} > G^{[2]} > \cdots > G^{[m+1]} = 1$, $G^{[j]} = G_{(\beta_1,\ldots,\beta_{j-1})}$, so that

$$G^{[i]}N = G, \quad G^{[i+1]}N < G.$$

New permutation domain Ω: orbits of $G^{[i+1]}N$, within $\beta_i G^{[i]}$. Action of $G^{[i]}$ on Ω is computable, N fixes Ω pointwise, so action of $G = \langle G^{[i]}, N \rangle$ is known.

$\varphi : G \to \mathrm{Sym}(\Omega)$ computable homomorphism, with nontrivial image, kernel; reduction to smaller problems.

O'Nan–Scott theorem, algorithmic version

$G \leq \Sigma_n$ primitive. Then at least one of the following holds.

- G has a normal subgroup N with $C_G(N) \neq 1$.
- G has a normal subgroup of index $\leq n$.
- G is perfect, has a unique minimal subgroup $N = N_1 \times \cdots \times N_m$ and G acts as A_m on $\{N_1, \ldots, N_m\}$ by conjugation.
  Three subcases as before, by the structure of N_ω.
- G is simple.

Advantage: Alternating action on $\{N_1, \ldots, N_m\}$ gives information on suborbit lengths (orbit lengths of G_ω).

Case of normal subgroup with small index

Taking $O(n)$ random elements of G, one of them is in a proper normal subgroup with high probability. For efficiency: consider G as black-box group, construct random elements as words in strong generators.

Problem: Given $g_1, \ldots, g_k$ elements of G, one of them in a proper normal subgroup. How to construct one element in a proper normal subgroup?

Beals trick: Replace g_1 and g_2 by $[g_1, g_2]$. Shorter list, and if one of g_1, g_2 in proper normal subgroup then so is $[g_1, g_2]$. Iterate.

Difficulty: $[g_1, g_2] = 1$.
Compute normal closure C of $\langle g_1 \rangle$. If $[g_1^t, g_2] \neq 1$ for some generator g_1^t of C then use $[g_1^t, g_2]$. If $[g_2, C] = 1$ then $C \neq G$ because $g_2 \notin Z(G)$; output g_1.

Other cases of computational O'Nan–Scott theorem

Take 2-point stabilizer $G_{\alpha\beta}$ with $|G : G_{\alpha\beta}| < n \log^2 n$; embed into maximal subgroup K containing a normal subgroup of G.

Of course, I oversimplified. E.g., special considerations for Frobenius groups.

We also may end up in maximal subgroup K with G acting faithfully on cosets of K; but in this case $|G : K| \leq n/2$.

A side remark

We can upgrade SGS constructions to a Las Vegas algorithm (guaranteed correct output).

Theorem (Kantor, Seress (1999))

Base, SGS in $O(n \log^c |G|)$ Las Vegas time, provided there are no ${}^2G_2$, ${}^2F_4$ composition factors.

Compute composition series by a Monte Carlo algorithm, verify correctness by constructively recognizing composition factors.

Newer development: ${}^2F_4$ composition factors are OK.
Missing step: Is there a short presentation of the groups ${}^2G_2$?

Some characteristic subgroups

Every finite group has a series of characteristic subgroups

$$1 \leq \mathrm{Rad}(G) \leq \mathrm{Soc}^*(G) \leq \mathrm{PKer}(G) \leq G$$

$\mathrm{Rad}(G)$: largest solvable normal subgroup

$\mathrm{Soc}^*(G)/\mathrm{Rad}(G)$: socle of $G/\mathrm{Rad}(G)$
$\mathrm{Soc}^*(G)/\mathrm{Rad}(G) \cong T_1 \times \cdots \times T_k$
T_i nonabelian simple, G permutes them by conjugation

$\mathrm{PKer}(G)$: kernel of this permutation action
$\mathrm{PKer}(G)/\mathrm{Soc}^*(G) \leq \mathrm{Out}(T_1) \times \cdots \times \mathrm{Out}(T_k)$ solvable (CFSG)

Composition series containing the special subgroups

For many algorithms (including automorphism group computations) it is beneficial to construct a chief series containing $\mathrm{Rad}(G), \mathrm{Soc}^*(G), \mathrm{PKer}(G)$.

Major step: Construction of $\mathrm{Rad}(G)$, permutation representation for $G/\mathrm{Rad}(G)$.

For arbitrary $N \lhd G \leq \Sigma_n$, G/N may not have faithful permutation representation of size polynomial in n. Fortunately, $G/\mathrm{Rad}(G)$ has a faithful representation on n points.

Construction of $\mathrm{Rad}(G)$

Suppose $N \lhd G \leq \Sigma_n$, $N \cong Z_p^m$ elementary abelian.
Define new permutation domain Ω for G, $|\Omega| = n$:
$[n] = O_1 \cup \ldots \cup O_k$: orbits of N.
N acts regularly on each O_i, $N|_{O_i}$ consists of $|O_i|$ permutations Ω_i; let $\Omega = \bigcup_{i=1}^{k} \Omega_i$.

Fact

G acts by conjugation on Ω. Kernel of this action is elementary abelian p-group, containing N.

Construction of $\mathrm{Rad}(G)$, II

Lemma

Suppose $N \lhd G$ maximal normal, $\mathrm{Rad}(N) = 1$.

(i) *If G/N is nonabelian then $\mathrm{Rad}(G) = 1$.*

(ii) *If G/N is cyclic then $\mathrm{Rad}(G) = C_G(N)$ (either trivial, or isomorphic to G/N).*

Note: Centralizer of a normal subgroup can be computed in nearly linear time.

Construction of $\mathrm{Rad}(G)$, III

Algorithm: RADICAL

Input: $G = \langle g_1, \ldots, g_k \rangle \leq \Sigma_n$.

1. Compute composition series
$$1 \lhd N_1 \lhd \cdots \lhd N_{m-1} \lhd N_m = G.$$
2. Find smallest i with $\mathrm{Rad}(N_i) \neq 1$.
[[$\mathrm{Rad}(N_i) \cong Z_p, \mathrm{Rad}(N_i) \lhd \lhd G$]]
3. Compute normal closure $N := \langle \mathrm{Rad}(N_i)^G \rangle$.
[[N is a p-group]]
4. In N, find elementary abelian subgroup K, $K \lhd G$.
5. Construct permutation representation for G, with K in kernel.
6. Iterate.

Automorphism group of $G/\mathrm{Rad}(G)$

Recall:

$$1 \leq \mathrm{Rad}(G) \leq \mathrm{Soc}^*(G) \leq \mathrm{PKer}(G) \leq G$$

and

$$\mathrm{Soc}(G/\mathrm{Rad}(G)) = T_1 \times \cdots \times T_k,$$

all T_i nonabelian simple.

$$\mathrm{Aut}(G/\mathrm{Rad}(G)) \leq \left(\prod_{i=1}^{k} \mathrm{Aut}(T_i)\right) \cdot \left(\prod_{j=1}^{\ell} \Sigma_{k_j}\right)$$

containing $\prod \mathrm{Inn}(T_i) \cong \prod T_i$.

Automorphism group of $G/\mathrm{Rad}(G)$, II

We need to recognize the groups T_i constructively: find identification with standard copy of T_i. Can be done in nearly linear time.
The full automorphism group of the standard copy of T_i can be constructed.
We can work in the factor group

$$\left(\prod_{i=1}^{k} \mathrm{Out}(T_i)\right) . \left(\prod_{j=1}^{\ell} \Sigma_{k_j}\right) .$$

"Work" means: Permutation representation for $(\prod \mathrm{Out}(T_i)).(\prod \Sigma_{k_j})$ can be constructed and enough to construct the normalizer of $G/\mathrm{Soc}^*(G)$ in this group.

Automorphism group of G

Construct $1 = L_0 < L_1 < \cdots < L_m = \mathrm{Rad}(G)$ satisfying

(i) L_i char G;

(ii) L_{i+1}/L_i elementary abelian (so L_{i+1}/L_i can be considered as a vector space V_i and G acts on V_i by conjugation, as a subgroup $G_i \leq \mathrm{GL}(V_i)$);

(iii) G_i is an irreducible matrix group.

Satisfying (i),(ii) and construction of the matrix group G_i is easy: refine derived series of $\mathrm{Rad}(G)$.

For (iii), apply the matrix group program MeatAxe, to compute a composition series of the G_i-module V_i.

Proceeding downwards on the series of factor groups

$$G/L_m = G/\mathrm{Rad}(G), G/L_{m-1}, \ldots, G/L_0 = G,$$

Computation of $\mathrm{Aut}(G)$ reduced to:

Problem

L char G, L elementary abelian, $\mathrm{Aut}(G/L)$ is known. Compute $\mathrm{Aut}(G)$.

Solution of the same problem is needed in the polycyclic case, the two cases merge.

Polycyclic groups

Definition

A group G is called polycyclic if there exists a chain of subgroups

$$G = G_1 \geq G_2 \geq \cdots \geq G_n \geq G_{n+1} = 1$$

so that $G_{i+1} \lhd G_i$ and G_i/G_{i+1} is cyclic for $1 \leq i \leq n$.

The group G may be infinite. Polycyclic groups are the solvable groups with the additional property that every subgroup is finitely generated.

Example

$G = A_4 = \langle (1,2)(3,4), (1,3)(2,4), (1,2,3) \rangle$. Then
$G = G_1$, $G_2 = \langle (1,2)(3,4), (1,3)(2,4) \rangle$,
$G_3 = \langle (1,2)(3,4) \rangle$, $G_4 = 1$ is a polycyclic series.

PCGS (PolyCyclic Generating Sequence)

Let G be polycyclic, with polycyclic series $G = G_1 \geq G_2 \geq \cdots \geq G_n \geq G_{n+1} = 1$.

Definition

A sequence of elements $X = [x_1, \ldots, x_n]$ is called a PCGS if $\langle x_i, G_{i+1} \rangle = G_i$ for $1 \leq i \leq n$.

Definition

With G, X as above, the sequence $R(X) = [r_1, \ldots, r_n]$, with $r_i := [G_i : G_{i+1}] \in \mathbb{Z}^+ \cup \{\infty\}$, is called the sequence of relative orders.

r_i is the smallest power k so that $x_i^k \in G_{i+1}$.

Example

$X = [(1,2,3),(1,2)(3,4),(1,3)(2,4)]$ is a PCGS for A_4, with relative orders $R(X) = [3,2,2]$. In this case, the order of x_i is the i^{th} relative order.

Example

$$X = \left[\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 1 & -1 \\ 0 & -1 \end{pmatrix}\right]$$

is a PCGS for the infinite dihedral group D_∞, with relative orders $R(X) = [2,\infty]$.

G is finite if and only if all r_i are finite.

If G is finite then

$$|G| = \prod_{r_i \in R(X)} r_i,$$

the product of the relative orders.

Length of a PCGS

The group G does not determine uniquely the length of a PCGS for G.

Example

Let $G = D_8 = \langle (1,2,3,4), (1,3) \rangle$. Then

- $X_1 = [(1,2)(3,4), (1,3), (2,4)]$, $R(X_1) = [2,2,2]$;
- $X_2 = [(1,2)(3,4), (1,2,3,4)]$, $R(X_2) = [2,4]$;
- $X_3 = [(1,2,3,4), (1,2)(3,4), (1,3)(2,4)]$, $R(X_3) = [2,2,2]$

are PCGS for G.

In $X_3 = [x_1, x_2, x_3]$, the relative order $r_1 = 2$, but $|x_1| = 4$.

Decomposition of group elements

Lemma

Let $X = [x_1, \ldots, x_n]$ be a PCGS for a group G. Then every $g \in G$ can be written uniquely in the form $g = x_1^{e_1} x_2^{e_2} \cdots x_n^{e_n}$, with exponents e_i satisfying $0 \leq e_i < r_i$ if r_i is finite; $e_i \in \mathbb{Z}$ if $r_i = \infty$.

Proof: Analogue of sifting in permutation groups.
Powers of x_i: left coset representatives for G_i mod G_{i+1}.
There is a unique coset $x_1^{e_1} G_2$ containing g.
$x_1^{-e_1} g \in G_2$; proceed by induction.

Example of decomposition

Example

Let $G = D_8 = \langle (1,2,3,4), (1,3) \rangle$.
$X_3 = [(1,2,3,4), (1,2)(3,4), (1,3)(2,4)]$, $R(X_3) = [2,2,2]$ is a PCGS for G.
$G_1 = G$, $G_2 = \langle (1,2)(3,4), (1,3)(2,4) \rangle$,
$G_3 = \langle (1,3)(2,4) \rangle$, $G_4 = 1$.

Let e.g. $g = (1,4,3,2)$. Then
$x_1^{-1}(1,4,3,2) = (1,3)(2,4) \in G_2$, so $e_1 = 1$.
$x_2^{-0}(1,3)(2,4) = (1,3)(2,4) \in G_3$, so $e_2 = 0$.
$x_3^{-1}(1,3)(2,4) = () \in G_4$, so $e_3 = 1$.

$g = x_1^1 x_2^0 x_3^1$

Normal form

Definition

The expression $g = x_1^{e-1} x_2^{e_2} \cdots x_n^{e_n}$ is the normal form of g, with exponent vector $\exp_X(g) = [e_1, e_2, \ldots, e_n]$ with respect to X.

Expressing certain powers and conjugates of elements of X in normal form, we obtain a presentation for G.

Polycyclic presentation

Definition

A presentation $\langle x_1, \ldots, x_n \mid R \rangle$ is called a polycyclic presentation if there exist $r_i \in \mathbb{N} \cup \{\infty\}$, for $1 \leq i \leq n$, so that R consists of the relations

(a) $x_i^{r_i} = x_{i+1}^{a_{i,i+1}} \cdots x_n^{a_{i,n}}$ if r_i is finite, $1 \leq i \leq n$

(b) $x_j^{-1} x_i x_j = x_{j+1}^{b_{i,j,j+1}} \cdots x_n^{b_{i,j,n}}$ for $1 \leq j < i \leq n$

(c) $x_j^{-1} x_i x_j = x_i x_{i+1}^{c_{i,j,i+1}} \cdots x_n^{c_{i,j,n}}$ for $1 \leq i < j \leq n$

Notation

$G = \mathrm{Pc}\langle x_1, \ldots, x_n \mid R \rangle$

(Pc presentation = power-conjugate presentation)

PCGS vs polycyclic presentation

Theorem

Every PCGS determines a unique polycyclic presentation.

Caution with the converse!

Any given $\mathrm{Pc}\langle x_1, \ldots, x_n \mid R\rangle$ defines a group with PCGS $X = [x_1, \ldots, x_n]$ but the relative orders $R(X) = [r_1, \ldots, r_n]$ may be smaller than the exponents in part (a) of the presentation.

Two different normal words may represent the same group element. Called inconsistent presentation.

Example of inconsistent presentation

Example

$$G = \mathrm{Pc}\langle x_1, x_2, x_3 \mid x_1^2 = x_2, x_2^3 = x_3,$$
$$x_2^{x_1} = x_2, x_3^{x_1} = x_2x_3, x_3^{x_2} = x_3,$$
$$x_1^{x_2} = x_1x_3, x_1^{x_3} = x_1, x_2^{x_3} = x_2\rangle$$

We may expect $R(X) = [2, 3, \infty]$.

But: $x_1^{x_2} = x_1x_3$, implying $[x_1, x_2] = x_3$.
Also $x_2^{x_1} = x_2$, implying $[x_2, x_1] = 1$. Hence $x_3 = 1$.

Hence $x_3^{x_1} = x_2x_3$ implies $1 = x_2$.

$R(X) = [2, 1, 1]$

There are efficient methods to determine whether a Pc presentation is consistent (and make it consistent, if it is not).

PCGS, polycyclic presentation for p-groups

Exponent-p lower central series of a p-group G:
$P_1 = G$, $P_{j+1} = [G, P_j]P_j^p$ for $j \geq 1$.
$P_j \lhd G$, P_j/P_{j+1} elementary abelian for all j

We work with PCGS refining the exponent-p lower central series.
Advantage: for such PCGS $X = [x_1, \ldots, x_n]$, all relative orders are p and
$x_j^{-1} x_i x_j = x_i x_{i+1}^{c_{i,j,i+1}} \cdots x_n^{c_{i,j,n}}$
holds for both $i < j$ and $j < i$.

Definition

A power-commutator presentation is a presentation with relators

(a) $x_i^p = x_{i+1}^{\alpha_{i,i+1}} \cdots x_n^{\alpha_{i,n}}$, $0 \leq \alpha_{i,k} < p$ for $1 \leq i \leq n$

(b) $[x_j, x_i] = x_{j+1}^{\beta_{i,j,j+1}} \cdots x_n^{\beta_{i,j,n}}$ for $1 \leq i < j \leq n$

Writing words in normal form

Given a PCGS $X = [x_1, \ldots, x_n]$ for G and a polycyclic presentation $G = \mathrm{Pc}\langle X \mid R\rangle$, we know that every word w in X represents a group element g and g has normal form $g = x_1^{e_1} \cdots x_n^{e_n}$.

Theoretical justification is an analogue of sifting in permutation groups, but no computational analogue: there is no easy way to find which coset $x_1^{e_1} G_2$ contains g (and so no easy way to find the exponents e_i).

The method to write words in normal form is called collection.

Collection

If a word w is in normal form then it is called collected; otherwise w is uncollected.

Uncollected words contain occurrences of minimal non-normal subwords: subwords u of w that are one of the following forms.

(a) $x_i x_j$, $x_i^{-1} x_j$, $x_i x_j^{-1}$, or $x_i^{-1} x_j^{-1}$ with $i > j$

(b) x_i^a for $r_i \neq \infty$ and $a \notin \{1, 2, \ldots, r_i - 1\}$.

One step of collection: replace a minimal non-normal subword by a collected one.
Collection: Repeat, until all minimal non-normal subwords are gone.

Collection strategies

Collection to the left (P. Hall): push occurrences of x_1 to the left, to get $w' = x_1^{e_1} w''$, where w'' is a word in $x_2, \ldots, x_n$. Proceed recursively.

Collection from the left (Leedham-Green, Soicher; Vaughan-Lee): replace the leftmost minimal non-normal subword.

Collection from the right (Havas, Nicholson): replace the rightmost minimal non-normal subword.

Theorem

No matter which order we process minimal non-normal subwords, collection terminates.

Practical observation: Collection from the left is most efficient.

Example of collection

$$G = D_{16} = \mathrm{Pc}\langle x_1, x_2, x_3, x_4 \mid x_1^2 = 1, x_2^2 = x_3x_4, x_3^2 = x_4, x_4^2 = 1, [x_2, x_1] = x_3, [x_3, x_1] = x_4, [x_3, x_2] = 1, [x_4, x_1] = 1, [x_4, x_2] = 1, [x_4, x_3] = 1\rangle$$

Collect the word $w = x_3x_2x_1$.

To the left: $3\underline{21} = \underline{31}23 = 13\underline{42}3 = 1\underline{32}43 = 12\underline{34}3 = 12\underline{33}4 = 1\underline{24}4 = 12$

7 steps

From the right: $3\underline{21} = \underline{31}23 = 13\underline{42}3 = 13\underline{24}3 = 1\underline{32}34 = 12\underline{33}4 = 1\underline{24}4 = 12$

7 steps

From the left: $\underline{32}1 = \underline{23}1 = \underline{21}34 = 12\underline{33}4 = 1\underline{24}4 = 12$

5 steps

Checking consistency, p-group version

Theorem (Wamsley; Vaughan-Lee)

A power-commutator presentation on $[x_1, \ldots, x_n]$ *is consistent if and only if the following holds.*

- $(x_k x_j)x_i = x_k(x_j x_i)$ *for* $1 \le i < j < k \le n$
- $(x_j^{p-1} x_j)x_i = x_j^{p-1}(x_j x_i)$ *for* $1 \le i < j \le n$
- $(x_j x_i^{p-1})x_i = x_j(x_i^{p-1} x_i)$ *for* $1 \le i < j \le n$
- $(x_i x_i^{p-1})x_i = x_i(x_i^{p-1} x_i)$ *for* $1 \le i \le n$

Interpretation: Collect both sides of each equation, starting with the minimal non-normal subword indicated by the parenthesis. Check whether the results of the two collections equal.

Algorithms for p-groups

Many algorithms are available, and in a lot of cases they are faster than the ones for the permutation representations of the same abstract p-group.

Recall that for $G = \mathrm{Pc}\langle X \mid R\rangle$, the subgroup chain defined by the PCGS X is a refinement of the exponent-p lower central series $P_1 = G$, $P_{j+1} = [G, P_j]P_j^p$ for $j \geq 1$ and $P_j \lhd G$, P_j/P_{j+1} elementary abelian for all j.

Most algorithms follow the inductive principle:

- Solve the problem for $P_1/P2$
- Using the solution for P_1/P_k, solve the problem for P_1/P_{k+1}.

Examples: compute conjugacy classes, automorphism groups, find a consistent presentation from an inconsistent one.

Construction of automorphism groups

Problem remaining in the permutation group case, and the problem to be solved in the polycyclic case:

Problem

L char G, L elementary abelian, $\mathrm{Aut}(G/L)$ is known. Compute $\mathrm{Aut}(G)$.

General solution is quite nasty, involving cohomology computations. Case of p-groups is easier, both conceptually and computationally.

Automorphism group of p-groups

Basic algorithm: O'Brien, late 1980's
Refinements: Eick, Leedham-Green, O'Brien 2002

Algorithm: AUT-P-GROUP

Input: G p-group, Z char G, Z elementary abelian, central, $F = G/Z$, $\mathrm{Aut}(F)$.

Output: $\mathrm{Aut}(G)$.

G is given by a Pc presentation, refining the exponent p lower central series; Z is the last term of this series.

p-covering group of F

Definition

The p-covering group C of F is the largest elementary abelian, central Frattini extension of F: $M \lhd C$, $M \leq \Phi(C)$, $C/M \cong F$, M elementary abelian.

Formally, if F is d-generated as a factor group $\mathcal{F}_d/R$ of the d-generated free group $\mathcal{F}_d$, then $C = [R, \mathcal{F}_d]R^p$.

Construction of the p-covering group of F

$F = \langle x_1, \ldots, x_d \rangle$ is given by the PCGS $[x_1, \ldots, x_n]$ and Pc presentation $R = \{R_1, \ldots, R_m\}$.

(i) For each relator R_l, introduce new generator y_l.

(ii) Append y_l to end of R_l:
$x_i^p = x_{i+1}^{\alpha_{i,i+1}} \cdots x_n^{\alpha_{i,n}} y_l$ or
$[x_j, x_i] = x_{j+1}^{\beta_{i,j,j+1}} \cdots x_n^{\beta_{i,j,n}} y_l$

(iii) Add relators showing that $\langle y_1, \ldots, y_m \rangle$ is elementary abelian: $y_l^p = 1$, $[y_l, y_J] = 1$.

(iv) Add relators showing that $\langle y_1, \ldots, y_m \rangle$ is central: $[x_i, y_l] = 1$.

(v) Make this presentation consistent.

Connection between G and C

Both G and C are central extensions of $F = \langle x_1, \ldots, x_d \rangle$.

Define $\gamma : G \to F$, $\psi : C \to F$ natural homomorphisms to a factor group.

Let $\overline{x_i}$, $1 \leq i \leq d$, be arbitrary γ-preimage of x_i. Note that $\langle \overline{x_i} \mid 1 \leq i \leq d \rangle = G$ because $Z \leq \Phi(G)$.

Let x_i^*, $1 \leq i \leq d$, be arbitrary ψ-preimage of x_i. Note that $\langle x_i^* \mid 1 \leq i \leq d \rangle = C$ because $M \leq \Phi(C)$.

Define $\varepsilon : \begin{array}{c} C \to G \\ x_i^* \mapsto \overline{x_i} \end{array}$

Then ε is an epimorphism, $M\varepsilon = Z$.

Define $U = \ker(\varepsilon) \leq M$.

Action of $\mathrm{Aut}(F)$ on M and $\mathrm{Aut}(G)$

Each $\alpha \in \mathrm{Aut}(F)$ induces an automorphism $\alpha_M \in \mathrm{Aut}(M)$:

$m \in M \Rightarrow m = w(x_1^*, \ldots, x_d^*)$ for some word w.

$x_i^*\psi \in F$, $x_i^*\psi\alpha \in F$, define $h_i^* \in C$ so that $h_i^*\psi = x_i^*\psi\alpha$.

Then $m\alpha_M := w(h_1^*, \ldots, h_d^*) \in M$.
Action of $\mathrm{Aut}(F)$ on M is computable.

Computationally most difficult step: Compute $S := \mathrm{Stab}_{\mathrm{Aut}(F)}(U)$. Most improvements try to reduce the size of the orbit of U, by exploiting characteristic subgroups that have to be preserved by any automorphism.

Let $T = \{\beta \in \mathrm{Aut}(G) \mid \beta \text{ centralizes } F\}$. T is computable with reasonable effort.

Theorem

Let $\nu : \mathrm{Aut}(G) \to \mathrm{Aut}(F)$ be the natural homomorphism. Then $T = \ker(\nu)$ and $S = \mathrm{im}(\nu)$, so $\mathrm{Aut}(G) = TR$ for $R\nu = S$.

Conclusion

The main purpose of these lectures was to make the audience aware of the wonderful computer algebra tools available.

The computer algebra systems GAP and Magma may be used to formulate and check conjectures, create counterexamples, gain insights for proofs in many branches of discrete mathematics, number theory and algebra. Besides these two general-purpose programs, there are many other, more specialized systems.

Over a thousand citations in group theory, representation theory, topology, algebraic graph theory, designs, cryptography, large combinatorial searches, exotic (partial differential equations, music), teaching.

Literature

Survey article

Ákos Seress: An introduction to computational group theory, Notices Amer. Math. Soc. 44 (1997), 671–679.
http://www.math.osu.edu/~akos/Publications.html

Short overview, easy bedtime reading :−)

Literature, II

Books :

- Derek Holt, Bettina Eick, Eamonn O'Brien: Handbook of computational group theory, Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2005. ISBN 1-58488-372-3
  Suggested first book to read.
- Charles C. Sims, Computation with Finitely-presented Groups, Encyclopedia of Mathematics and its Applications, vol. 48, Cambridge University Press, Cambridge, 1994. ISBN 0-521-43213-8
- Ákos Seress, Permutation group algorithms, Cambridge Tracts in Mathematics, vol. 152, Cambridge University Press, Cambridge, 2003. ISBN 0-521-66103-X.