

Last Time

Finitely Presented groups

Quotient algorithms — give homomorphisms

Pre-images under homomorphisms give subgroups.

Representing Subgroups

Work with homomorphisms to represent subgroup $U < G$ as pair (φ, U_φ) with $\varphi: G \rightarrow P$ for a (finite, permutation) group P . Can do well

- ▶ Normalizer, (Normal) closure
- ▶ Intersection (Embed in direct product)
- ▶ Subgroup of Subgroup (Embed in wreath product)
- ▶ pre-images of arbitrary subgroups


```
gap> s:=Intersection(k,k2);
```

```
Group(<fp, no generators known>)
```

```
gap> Index(g,s);
```

```
126647579520
```

```
gap> q:=DefiningQuotientHomomorphism(s);
```

```
[ a, b, c ] ->
```

```
[ (1,260,241)(2,240,177)(3,91,130)(4,234,203)
```

```
gap> p:=Image(q);
```

```
<permutation group of size 126647579520 with 3 generators>
```

```
gap> m:=MaximalSubgroupClassReps(p);;List(m,x->Index(p,x));
```

```
[ 114, 6328, 6441, 30058, 30058, 266, 1045, [...]
```

```
gap> u:=PreImage(q,m[4]);
```

```
Group(<fp, no generators known>)
```

```
gap> Index(g,u);
```

```
30058
```

```
gap> AbelianInvariants(u);
```

```
[ 2 ]
```

```
gap> AbelianInvariants(m[4]);
```

```
[ 2 ]
```


Subgroups And Coset Tables

Vice versa, if we have a subgroup $U < G$, given by generators, we want to find the index $[G:U]$, or even better the permutation action of G on the cosets of U .

If U is trivial, this determines $|G|$ – test for order!

Tool: Coset table. Columns correspond generators of G and inverses, rows are cosets of U . Entry in row i , generator a is number of coset image i^a .

Will also allow us to construct presentations for subgroups.

Coset Enumeration

The *process* of coset enumeration (Todd-Coxeter method) builds the coset table by defining cosets systematically as images, and deducing equality based on relators and subgroup generators.

It is the one of the first (1936) articles in CGT — at that time the calculation was performed by hand on paper.

Besides coset table, use two kinds of auxiliary tables:

One per relator (rows for every coset), one per subgroup generator (only one row).

Coset Enumeration

The *process* of coset enumeration (Todd-Coxeter method)

by finding cosets systematically as based on relators and

Because of impossibility results
this cannot be an algorithm
with predictable runtime.

It is the one of the first (1936) articles in CGT — at that time the calculation was performed by hand on paper.

Besides coset table, use two kinds of auxiliary tables:

One per relator (rows for every coset), one per subgroup generator (only one row).

Coset Enumeration

The *process* of coset enumeration (Todd-Coxeter method)

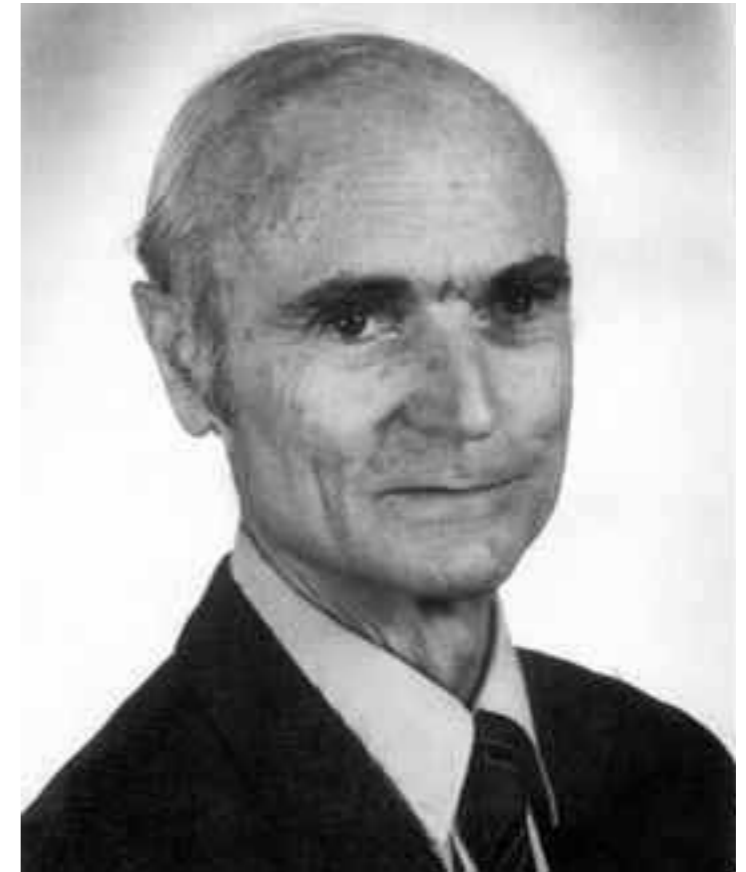
builds the
images,
subgroup

It is the
the calc

Besides



John Todd, 1908-1994



H.S.M. Coxeter, 1907-2003

One per relator (rows for every coset), one per subgroup
generator (only one row).

Coset Enumeration

The *process* of coset enumeration (Todd-Coxeter method) builds the coset table by defining cosets systematically as images, and deducing equality based on relators and subgroup generators.

It is the one of the first (1936) articles in CGT — at that time the calculation was performed by hand on paper.

Besides coset table, use two kinds of auxiliary tables:

One per relator (rows for every coset), one per subgroup generator (only one row).

Relator And Subgroup Tables

The subgroup and relator tables will be used to ensure:

- $1^s = 1$ for every generator s of U (1 is the coset U)
- $x^r = x$ for any coset x and any relator r . (Alternatively: For any coset representative g , grg^{-1} is also a relator.)

We do so by *tracing through* images of cosets, as soon as they are defined in the coset table. (The computer does so without need to write down the actual tables.)

Example

Let $G = \langle a, b \mid a^2 = b^3 = (ab)^5 = 1 \rangle$,
 and $U = \langle a, a^b \rangle$.

	a	a ⁻¹	b	b ⁻¹
1				

Relator Tables:

	a	a	b	b	b
1	1	1	1	1	1

	a	b	a	b	a	b	a	b	b
1	1								1

Subgroup Tables:

	a	b ⁻¹	a	b
1	1	1	1	1

Result

Let $G = \langle a, b \mid a^2 = b^3 = (ab)^5 = 1 \rangle$,
 and $U = \langle a, a^b \rangle$.

	a	a ⁻¹	b	b ⁻¹
1	1	1	2	3
2	4	4	3	1
3	3	3	1	2
4	2	2	5	6
5	6	6	6	4
6	5	5	4	5

Index 6

Permutations:

$$a \rightarrow (2,4)(5,6)$$

$$b \rightarrow (1,2,3)(4,5,6)$$

Coset Reps:

$$2 = 1b$$

$$3 = 1b^{-1}$$

$$4 = 1ba$$

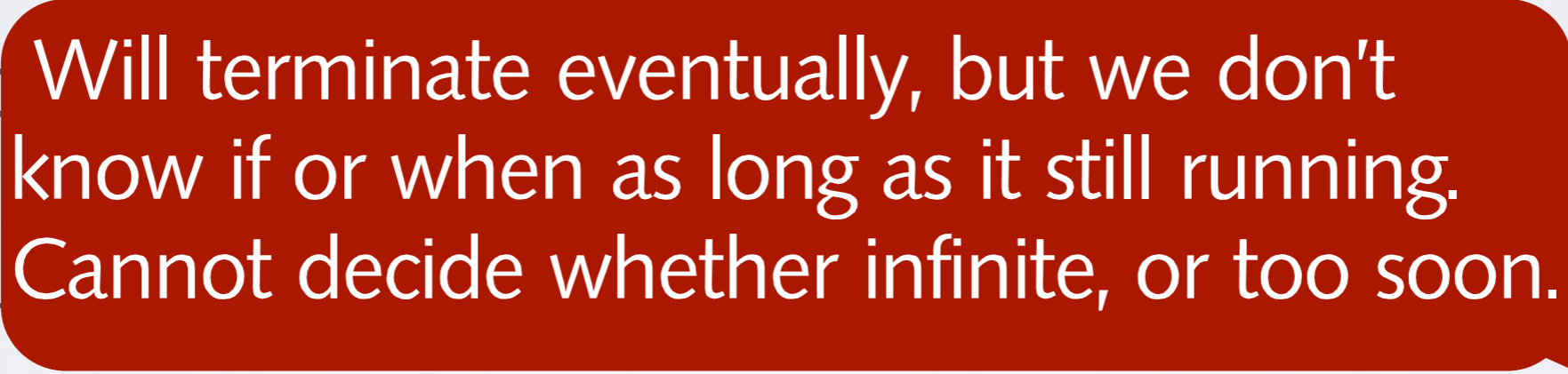
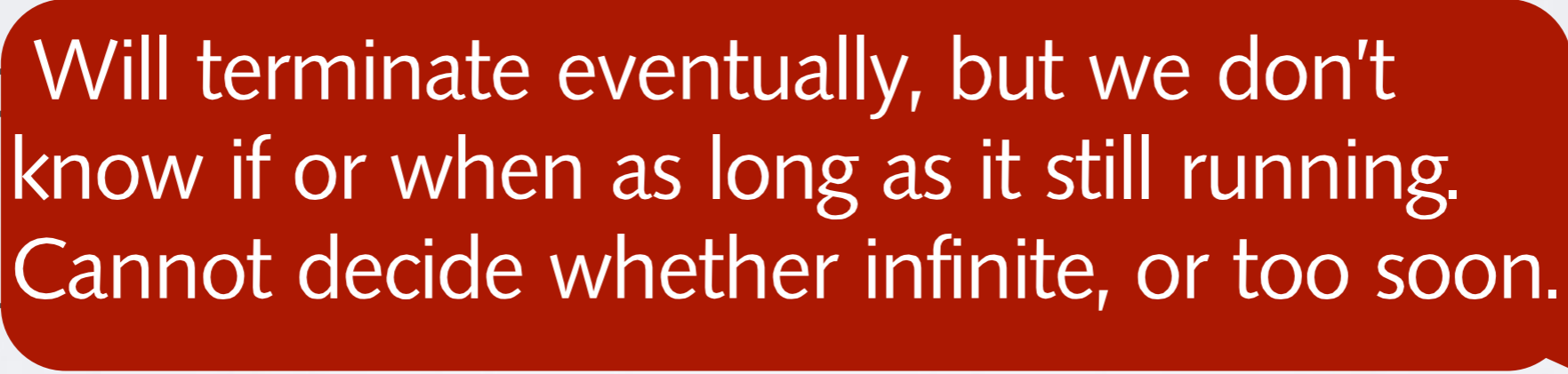
$$5 = 1bab$$

$$6 = 1bab^{-1}$$

Remarks

- A consequence ($x^g=y$) can be a *coincidence* of two cosets y,z that were previously considered separate. When processing these, typically further coincidences arise.
- Cosets are considered different unless the conditions require equality.
- If the coset table is complete, it must be correct.
- We cannot bound runtime.
- If index is finite, and systematic definitions, will terminate.

Remarks

- A consequence ($x^g=y$) can be a *coincidence* of two cosets y,z that were previously considered separate. When processing these, typically further coincidences arise.
- Cosets are considered different unless the conditions require equality.
- If the c  Will terminate eventually, but we don't know if or when as long as it still running.
- We ca  Cannot decide whether infinite, or too soon.
- If index is finite, and systematic definitions, will terminate.

Subgroup Presentations

Given coset table for U , select representatives t_i for coset i .

Schreier generator for coset i , generator j is: $t_i g_j (t_i g_j)^{-1}$.

(Can be trivial if $t_i g_j = t_k$.)

n cosets, m generators for G , there are $n-1$ definitions, thus:
 $nm - (n-1) = n(m-1) + 1$ nontrivial Schreier generators.

We can rewrite any element of G that lies in subgroup as word in Schreier generators, let ρ be this rewriting.

Subgroup Presentations

Rewriting:

Given a word in one generator set, express it into another generator set by following a set of substring replacement.

The proof of SCHREIER's theorem describes such a process. For example, if $1^a=2$ and $2^b=1$, using $t_1=id$:

$$ab = t_1 a b = t_1 a t_2^{-1} t_2 b = s_{1a} t_2 b = s_{1a} t_2 b t_1^{-1} = s_{1a} s_{2b}$$

n cosets, m

$nm - (n-1) = n(m-1) + 1$ nontrivial Schreier generators.

We can rewrite any element of G that lies in subgroup as word in Schreier generators, let ρ be this rewriting.

Reidemeister's Theorem



Kurt Reidemeister, 1893-1971

For any

Theorem
given by

Proof: C
arbitrary
conjugates

rep. t_i , clearly $1 = t_i r t_i^{-1} \in U$.

in Schreier generators is

$$\langle r, (t_i r t_i^{-1}), 1 \leq i \leq n, 1 \leq j \leq m \rangle$$

are identity. Take an
can write as product of
has form $r g$.

Write $g = t_i^{-1} q$, $q \in U$. Then $r g = q^{-1} t_i r t_i^{-1} q = (t_i r t_i^{-1}) q$.

Note: Variant (Modified Todd-Coxeter) gives presentation
in user-selected generators, often worse looking result.

Reidemeister's Theorem

For any relator r of G , any coset rep. t_i , clearly $1 = t_i r t_i^{-1} \in U$.

Theorem: A presentation for U in Schreier generators is given by:

$$\langle s_{i,j} \text{ nontrivial} \mid \rho(t_i r_k t_i^{-1}), 1 \leq i \leq n, 1 \leq j \leq m \rangle$$

This is sloppy notation! Formally distinguish between elements of free group, elements of G , and symbols for new presentation.

Note: Variant (Modified Todd-Coxeter) gives presentation in user-selected generators, often worse looking result.

Reidemeister's Theorem

For any relator r of G , any coset rep. t_i , clearly $1 = t_i r t_i^{-1} \in U$.

Theorem: A presentation for U in Schreier generators is given by:

$$\langle s_{i,j} \text{ nontrivial} \mid \rho(t_i r_k t_i^{-1}), 1 \leq i \leq n, 1 \leq j \leq m \rangle$$

Proof: Clearly all these relations are identity. Take an arbitrary relation $w=1$ in G , then can write as product of conjugates of relators, each factor has form r^g .

Write $g=t_i^{-1}q$, $q \in U$. Then $r^g = q^{-1} t_i r t_i^{-1} q = (t_i r_k t_i^{-1})^q$.

Note: Variant (Modified Todd-Coxeter) gives presentation in user-selected generators, often worse looking result.

In Practice

Read Later

Store in augmented coset table the accumulated Schreier generators that would give element (not just coset) equality. (For every entry that is not a **coset definition**.)

	a	a ⁻¹	b	b ⁻¹
1	1	1	2	3
2	4	4	3	1
3	3	3	1	2
4	2	2	5	6
5	6	6	6	4
6	5	5	4	5

Coset Reps:

$$t_1=1$$

$$t_2=b$$

$$t_3=b^{-1}$$

$$t_4=ba$$

$$t_5=bab$$

$$t_6=bab^{-1}$$

Schreier Generators:

$$c=t_1at_1^{-1}=a$$

$$d=t_2bt_3^{-1}=b^3$$

$$e=t_3at_3^{-1}=b^{-1}ab$$

$$f=t_4at_2^{-1}=baab^{-1}$$

$$g=t_5at_6^{-1}=bababa^{-1}b^{-1}$$

$$h=t_5bt_6^{-1}=babbbba^{-1}b^{-1}$$

$$i=t_6at_5^{-1}=bab^{-1}ab^{-1}a^{-1}b^{-1}$$

In Practice

Read Later

Store in augmented coset table the accumulated Schreier generators that would give element (not just coset) equality. (For every entry that is not a **coset definition**.)

	a	a ⁻¹	b	b ⁻¹
1	c 1	c ⁻¹ 1	2	3
2	4	f ⁻¹ 4	d 3	1
3	e 3	e ⁻¹ 3	1	d ⁻¹ 2
4	f 2	2	5	6
5	g 6	i ⁻¹ 6	h 6	4
6	i 5	g ⁻¹ 5	4	h ⁻¹ 5

Coset Reps:

$$t_1=1$$

$$t_2=b$$

$$t_3=b^{-1}$$

$$t_4=ba$$

$$t_5=bab$$

$$t_6=bab^{-1}$$

Schreier Generators:

$$c=t_1at_1^{-1}=a$$

$$d=t_2bt_3^{-1}=b^3$$

$$e=t_3at_3^{-1}=b^{-1}ab$$

$$f=t_4at_2^{-1}=baab^{-1}$$

$$g=t_5at_6^{-1}=bababa^{-1}b^{-1}$$

$$h=t_5bt_6^{-1}=babbba^{-1}b^{-1}$$

$$i=t_6at_5^{-1}=bab^{-1}ab^{-1}a^{-1}b^{-1}$$

Example, Continued

Read Later

Now trace through every relator, at every coset (i.e. conjugate relators) and collect Schreier generators.

	a		a ⁻¹		b		b ⁻¹	
1	c	1	c ⁻¹	1		2		3
2		4	f ⁻¹	4	d	3		1
3	e	3	e ⁻¹	3		1	d ⁻¹	2
4	f	2		2		5		6
5	g	6	i ⁻¹	6	h	6		4
6	i	5	g ⁻¹	5		4	h ⁻¹	5

	a ²	b ³	(ab) ⁵
1	c ²	d	cgfde
2	f	d	gfdec
3	e ²	d	ecgfd
4	f	h	fdecg
5	gi	h	gfdec
6	ig	h	(ih) ⁵

Example, Continued

Read Later

Now trace through every relator, at every coset (i.e. conjugate relators) and collect Schreier generators.

	a	a ⁻¹	b	b ⁻¹
1	c 1	c ⁻¹ 1	2	3
2	4	f ⁻¹ 4	d 3	1
3	e 3	e ⁻¹ 3	1	d ⁻¹ 2
4	f 2	2	5	6
5	g 6	i ⁻¹ 6	h 6	4
6	i 5	g ⁻¹ 5	4	h ⁻¹ 5

	a ²	b ³	(ab) ⁵
1	c ²	d	cgfde
2	f	d	gfdec
3	e ²	d	ecgfd
4	f	h	fdecg
5	gi	h	gfdec
6	ig	h	(ih) ⁵

$$U = \langle c, d, e, f, g, h, i \mid c^2, d, cgfde, f, e^2, h, gi, ig, (ih)^5 \rangle$$

$$= \langle c, e, g \mid c^2, e^2, cge, g^{-5} \rangle = \langle c, e \mid c^2, e^2, (ec)^5 \rangle$$

Example, Continued

Read Later

Now trace through every relator, at every coset (i.e. conjugate relators) and collect Schreier generators.

Dihedral group of order 10:

$$= \langle c, d \mid c^2, d^5, dc^{-1}dc^{-1} \rangle \text{ with } d=ec,$$

$$\text{thus } cd^{-1} = dc^{-1} = dc$$

Group G has order 60.

$$U = \langle c, d, e, f, g, h, i \mid c^2, d, cgfde, f, , h, gi, ig, (ih)^5 \rangle$$

$$= \langle c, e, g \mid c^2, e^2, cge, g^{-5} \rangle = \langle c, e \mid c^2, e^2, (ec)^5 \rangle$$

In GAP

The easiest way too handle this process of rewriting a presentation for a subgroup is to use functions `IsomorphismFpGroup`, respectively (for user generators) `IsomorphismFpGroupByGenerators` that both produce homomorphisms to new fp groups.

```
gap> g:=f/ParseRelators(f,"a3,b5,(ab)2");;
gap> u:=Subgroup(g,[g.1,g.1^g.2]);;Index(g,u);
6
gap> hom:=IsomorphismFpGroup(u);
[<[[1,1]]|a>,<[[1,-1,2,-1]]|a^-1*b^-1*a^-1*b>]->[F1,F2]
gap> RelatorsOfFpGroup(Image(hom));
[ F1^2, (F2^-1*F1)^2, F2^5 ]
gap> hom:=IsomorphismFpGroupByGenerators(u,
GeneratorsOfGroup(u));
[ a, b^-1*a*b ] -> [ F1, F2 ]
```


In GAP

What on Earth is this?

The algorithm does not multiply out words, but stores *straight line programs*: $1,-1,2,-1$ is $\text{gen}_1^1 * \text{gen}_2^{-1}$. These are elements stored as straight line programs and printing as program|word. For all practical purposes, treat them like an ordinary element.

```
gap> g:=f/ParseRelators(f,"b,b5,(ab)2");;
gap> u:=Subgroup(g,[g.1,g.1^g.2]);;Index(g,u);
6
gap> hom:=IsomorphismFpGroup(u);
[<[[1,1]]|a>,<[[1,-1,2,-1]]|a^-1*b^-1*a^-1*b>]->[F1,F2]
gap> RelatorsOfFpGroup(Image(hom));
[ F1^2, (F2^-1*F1)^2, F2^5 ]
gap> hom:=IsomorphismFpGroupByGenerators(u,
GeneratorsOfGroup(u));
[ a, b^-1*a*b ] -> [ F1, F2 ]
```


In GAP

The easiest way too handle this process of rewriting a presentation for a subgroup is to use functions `IsomorphismFpGroup`, respectively (for user generators) `IsomorphismFpGroupByGenerators` that both produce homomorphisms to new fp groups.

```
gap> g:=f/ParseRelators(f,"a3,b5,(ab)2");;
gap> u:=Subgroup(g,[g.1,g.1^g.2]);;Index(g,u);
6
gap> hom:=IsomorphismFpGroup(u);
[<[[1,1]]|a>,<[[1,-1,2,-1]]|a^-1*b^-1*a^-1*b>]->[F1,F2]
gap> RelatorsOfFpGroup(Image(hom));
[ F1^2, (F2^-1*F1)^2, F2^5 ]
gap> hom:=IsomorphismFpGroupByGenerators(u,
GeneratorsOfGroup(u));
[ a, b^-1*a*b ] -> [ F1, F2 ]
```

Simplification

The result of rewriting is often messy and contains “stupid” generators that e.g. duplicate others. Thus reduce heuristically by applying *Tietze Transformations*:

- ▶ Add/Delete relator that is word in other relators
- ▶ Add/Delete generator that occurs only in one relator

(In fact GAP already does some initial simplification on its own, before even returning the homomorphism.)

`IsomorphismSimplifiedFpGroup` does so more boldly.

Calculating Group Order

To determine a groups order, rewrite the presentation to a subgroup (e.g. Low Index, Quotient pre-image of subgroup, generated by some elements) and use this.

Maybe iterate.

In particular: Cyclic subgroup, generated by one element — presentations in one generator are easy.

If group is finite this will *eventually* terminate.

Finding Presentations For Group

For a (permutation) group G , find a presentation:

1. Investigate multiplication structure, find some relators, prove that they define no larger group.
2. Take homomorphism from free group, compute kernel generators via stabilizer chain.
3. Combine presentations of normal subgroup and factor group (similar to how a pc presentation works).

Simple Nonabelian Factors

Use Classification of Finite Simple Groups:

Cyclic: done.

Alternating: Presentation on $(1,2,3,4,\dots,n)$, $(1,3,2,4,5,\dots,n)$
(odd n , variant for even n) from COXETER, MOSER book.

Sporadic: Ad-hoc presentations found by hand.

Lie Type (e.g. GL for finite fields): STEINBERG presentations.
These are not *short* in the storage size of a generating set.

Short Presentations

For complexity analysis — if the presentation grows too much, evaluating a presentation will take too much time — a lot of recent work has gone towards finding short presentations of groups of Lie type:

▶ Lie Type of rank >1 (STEINBERG 1962, BABAI, GOODMAN, KANTOR, LUKS, PÁLFY, 1997)

▶ $\text{PSL}_2(q)$ (TODD 1936)

▶ Suzuki groups (SUZUKI 1964)

▶ $\text{PSU}_3(q)$ (H., SERESS 2001)

Only the Ree groups ${}^2\text{G}_2(q)$ remain ...

4

Permutation And Matrix Groups

Testing Random Stabilizer Chains

Can compute a stabilizer chain quickly by random methods.

If we are unlucky, the chain describes a set that is *too small*.

Calculate a presentation for this group (From Composition Series and presentations for simple factors, combine)

If we were unlucky, the resulting presentation describes a *smaller* group.

Detect this by evaluating the presentation on G . If it is too tight (or anything failed) we know stabilizer chain was wrong.

Otherwise we have certified the stabilizer chain to be correct!

Almost Linear Time Stab.Chain

1. Compute stabilizer chain, using only random subproducts of Schreier generators. (Quick)
2. Determine composition series. (see below)
3. (Short — **for good complexity**) presentation for whole group, from composition factors. (Rewrite in original generators.)
4. If anything failed, or group does not satisfy presentation, the random calculation failed — add further random Schreier generators. Otherwise proven correct.

Towards Composition Series

Thus need composition series.

Want to construct a composition series

$G = G_0 > G_1 > \dots > G_k = \langle 1 \rangle$, that is $G_{i+1} \triangleleft G_i$ and G_i/G_{i+1} simple.

Use homomorphism kernels to split; Either:

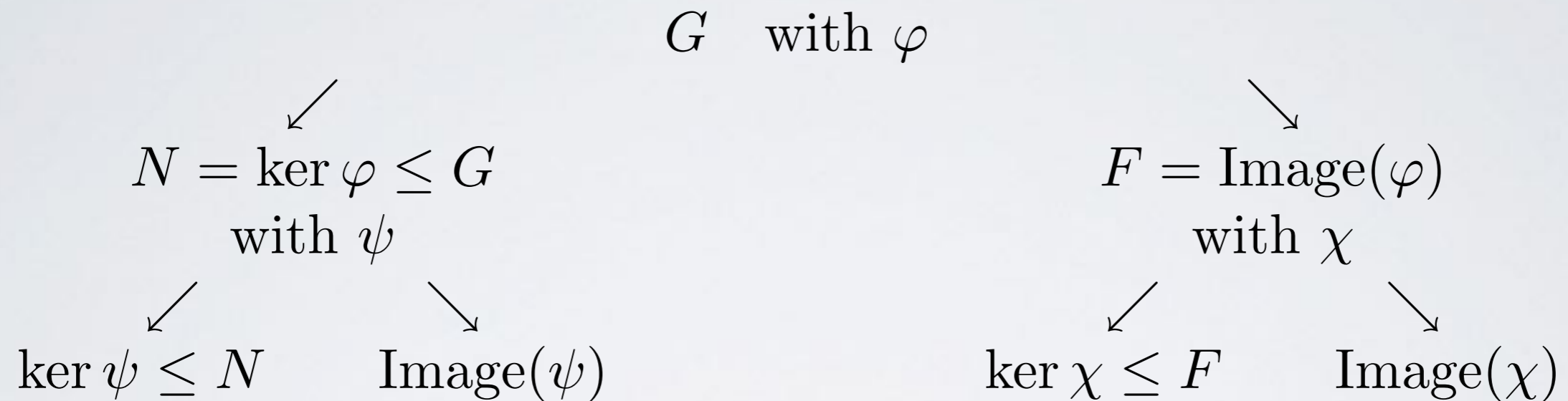
► Find a homomorphism φ on G with “*smaller*” (by size or required effort) image.

Recurse to image, kernel. Or:

► Prove that G is simple.

Composition Tree

The resulting data structure is called a *Composition Tree*.



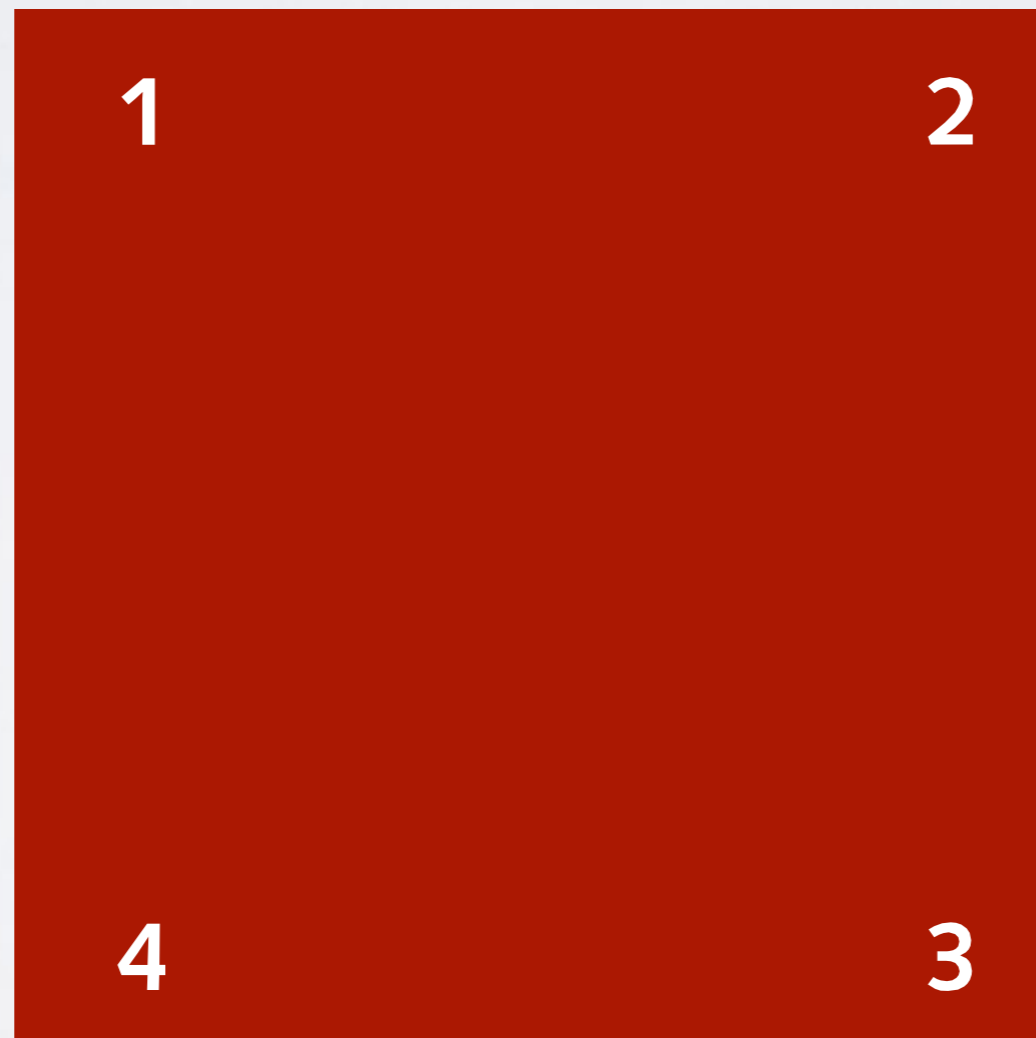
Natural Source of Homomorphisms: Group Actions, in particular from permutation action.

(Same concept for other groups but different actions.)

If intransitive: Action on one orbit.

Blocks (AKA imprimitivity)

G acting (transitively) on Ω , a *block system* is a G -invariant partition \mathcal{B} of Ω , i.e. $\Omega = \cup_{B \in \mathcal{B}} B$ but $B_i \cap B_j = \emptyset$. Thus G also acts on \mathcal{B} .



Blocks (AKA imprimitivity)

G acting (transitively) on Ω , a *block system* is a G -invariant partition \mathcal{B} of Ω , i.e. $\Omega = \bigcup_{B \in \mathcal{B}} B$ but $B_i \cap B_j = \emptyset$. Thus G also acts on \mathcal{B} .

Basic facts:

- All blocks have the same size.
- Trivial block systems: $\{\Omega\}$ and singleton sets. If only these two: G *primitive* (otherwise *imprimitive*).
- Block systems are in bijection with subgroups $\text{Stab}_G(\omega) \leq S \leq G$, S is stabilizer of block with ω .
- If $N \triangleleft G$ the orbits of N form a block system.

Primitive Groups

If no blocks, G is primitive. The O'NAN-SCOTT theorem describes the possible structure

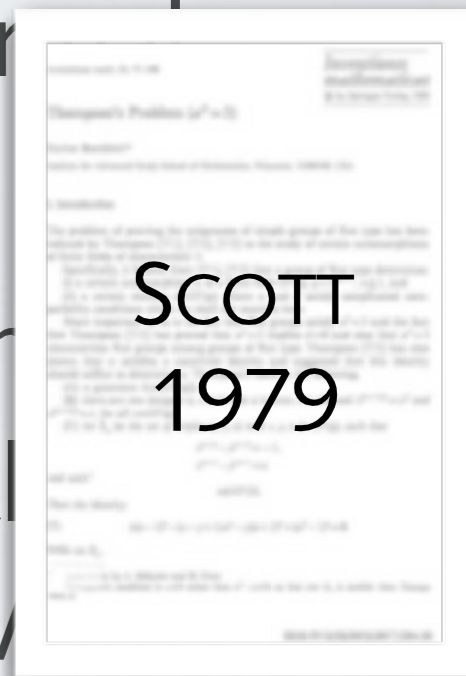
Key component: The $Soc(C)$ subgroup generated by



Leonard Scott, *1942



Michael O'Nan, *1943



Primitive Groups

If no blocks, G is *primitive*. The O'NAN-SCOTT theorem describes the possible structure.

Key component: The *Socle* $\text{Soc}(G)$, subgroup generated by all minimal normal subgroups.

Lemma: $\text{Soc}(G)$ is direct product of minimal normal subgroups.

Proof: Take $M \leq \text{Soc}(G)$, $M \triangleleft G$ maximal with this property. If $M \neq \text{Soc}(G)$ there exists $N \triangleleft G$, minimally normal, $N \not\leq M$. Thus $M \cap N = \langle 1 \rangle$ and $\langle M, N \rangle = M \times N \leq \text{Soc}(G)$ is larger, Contradiction.

Actions To Get Composition Tree

- ▶ If intransitive, act on orbit. *Otherwise:*
- ▶ If imprimitive, act on blocks (as sets). *Otherwise:*
- ▶ If primitive, socle not simple: act on socle factors.
Otherwise:
- ▶ If not simple: $\text{Out}(S)$ is small, solvable. *Otherwise:*
- ▶ Constructive recognition (isomorphism) to simple group. (Use CFSG!). Later.

In GAP

When working with permutations groups, knowing a groups order speeds up calculations immensely!

There is `CompositionSeries` (also `DerivedSeries`, `ElementaryAbelianSeries`, ...) and `DisplayCompositionSeries`.

`Blocks(group, domain, action)` finds a block system (also: `IsPrimitive`), `AllBlocks` finds representatives for all block systems.

Same Idea For Matrix Groups

For Matrix groups (and others) the stabilizer chain approach can be infeasible because of long orbits.

Thus use same approach of homomorphisms to decompose the group and get a composition series as basic data structure.

Matrix Actions

To repeat strategy from permutation groups, need actions for matrix groups. For example (MeatAxe) action on submodule or quotient module.

Need actions for irreducible groups:

Permutation of basis vectors

Permutation	Matrix
intransitive	reducible
imprimitive	<i>induced</i>

Matrix Actions

To repeat strategy from permutation groups, need actions for matrix groups. For example (M, V) or quotient module.

Need actions for irreducible groups.

Permutation of basis vectors.

Matrix wreath product: A is of dimension m , B permutes n points. $A \wr B$ in dimension $n \cdot m$, Each copy of A is diagonal block, B permutes blocks.

Permutation	Matrix
intransitive	reducible
imprimitive	<i>induced</i>

Matrix Actions

Space whose basis is labelled with formal products $\underline{b} \otimes \underline{c}$, compatible action on both parts.

tion groups, need actions for (atAxe) action on submodule

ps:

Permutation of basis vectors

Tensor products

Permutation	Matrix
intransitive	reducible
imprimitive	<i>induced</i>

Permutation	Matrix
intransitive	<i>tensor product</i>
imprimitive	<i>tensor wreath</i>

Matrix Actions

Space whose basis is labelled with formal products $\underline{b} \otimes \underline{c}$, compatible action on both parts.

tion groups, need actions for (atAxe) action on submodule

ps:

Permutation of basis vectors

Tensor products

Permutation

Matrix

Permutation

Matrix

intransitive

reducible

intransitive

tensor product

imprimitive

induced

imprimitive

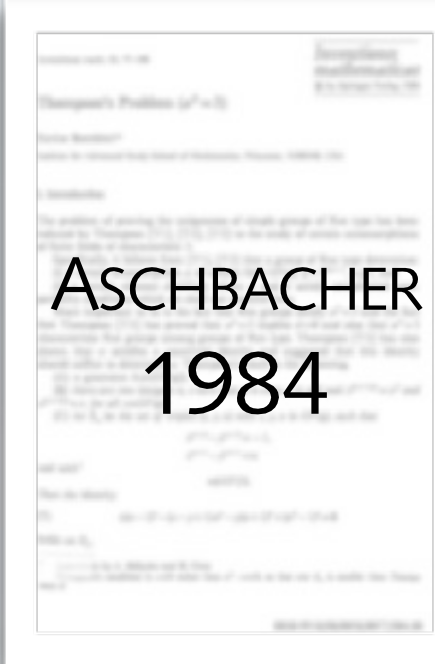
tensor wreath

Analog to O'NAN-SCOTT, ASCHBACHER's theorem describes geometric reductions for matrix groups.

Aschbacher's Theorem

Matrix group over finite fields fall in classes:

- 1) Reducible
- 2) Induced
- 3) Up to scalars, in smaller dimension over larger field.
- 4) Tensor product of non isomorphic submodules
- 5) Up to scalars, write over smaller field
- 6) Normalizing an extraspecial group
- 7) Tensor Wreath product
- 8) Stabilizing bilinear form (Sp, O, U)
- 9) Almost simple (ie. $T \leq G \leq \text{Aut}(T)$)



ASCHBACHER
1984

Matrix Group Recognition

Research project in CGT over last 15-20 years:

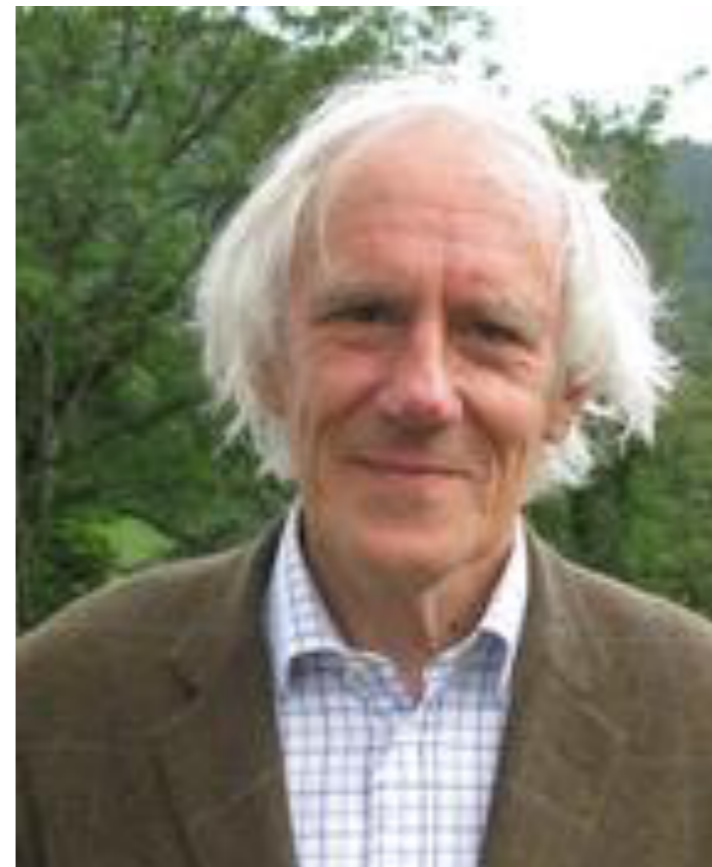
- ▶ Develop algorithms that will recognize the different classes and produce homomorphisms.
- ▶ If (almost) simple, recognize the group — isomorphism (type and actual bijective map) to *natural* representation.
- ▶ Presentations for simple group give generators of kernels. Get composition tree.
- ▶ In Las-Vegas (random selections+verification) poly. time.
- ▶ Implementations recently available.

Matrix Group Recognition

Research project in CGT over last 15-20 years:



Eamonn O'Brien



Charles Leedham-Green

► Implementations recently available.

Matrix Group Recognition

Research project in CGT over last 15-20 years:

- ▶ Develop algorithms that will recognize the different classes and produce homomorphisms.
- ▶ If (almost) simple, recognize the group — isomorphism (type and actual bijective map) to *natural* representation.
- ▶ Presentations for simple group give generators of kernels. Get composition tree.
- ▶ In Las-Vegas (random selections+verification) poly. time.
- ▶ Implementations recently available.

Matrix Group Recognition



Derek Holt



Max Neunhöffer



Ákos Seress, 1958-2013

kernelis. Get composition tree.

- ▶ In Las-Vegas (random selections+verification) poly. time.
- ▶ Implementations recently available.

Base Case: Simple Groups

After the reductions of ASCHBACHER's theorem we have an almost simple group. Due to SCHREIERS conjecture G''' is simple and $[G:G''']$ small. WLOG G simple, no reduction.

Assumption: We know G (we know the simple groups).

Find an isomorphism (called *constructive recognition* from G to the natural representation of the simple group (A_n as permutations, PSL as matrices up to scalars, ...)).

Then use known information from the literature (or stabilizer chain methods...) to obtain the result.

(Constructive) Recognition

First identify the type of G by looking at order statistics.

But G might be in *some* (horrible) matrix representation.

The elements of the natural representation act as matrices or permutations.

Build these from scratch by acting on suitable objects that correspond to the underlying geometry/combinatorics.

E.g. *transvections* $\underline{v} \mapsto \underline{v} + \underline{t}$ in place of vectors. Identify such elements from orders (also of products).

Model: Black-Box Groups

Conceptually this utilizes the model of a *black-box* group:

- ▶ Can form products, inverses
- ▶ Can test for equality
- ▶ Upper bound on group order.

To enable homomorphisms, find kernels, remember how elements were formed as products of generators.