# Computational methods in AGT

## (Minicourse on MAGMA)

## Rogla, Slovenia, June 2011

Marston Conder

University of Auckland

m.conder@auckland.ac.nz

## Outline of topics

#1 Overview of MAGMA and the kinds of structures it can deal with (incl. graphs, digraphs, Cayley graphs)

#2 Dealing with permutation groups, matrix groups, and groups of small order

#3 Finitely presented groups and their images/applications

#4 Question and Answer session
(Please give me some questions!)

PDF copies of the slides from each lecture will be available

# Lecture 1: Overview of MAGMA

**What is MAGMA?** A comprehensive software package for computations in algebra, number theory, algebraic geometry and algebraic combinatorics

– Developed in 1980s, distributed by University of Sydney

– Similar to GAP, distributed by University of St Andrews

– Can run under Mac, Linux or Windows operating systems

– Available for use by individuals/groups/departments

– Its use has been cited in papers 2918 times at last count

Web address: magma.maths.usyd.edu.au/magma

## How can MAGMA help you with your work?

– Analyse specific structures (e.g. a given graph or group)

– Generate all small examples

– Test conjectures on small examples

– Look for patterns among all small examples

– Provide better understanding of given kinds of structures

What's best of all is that MAGMA can often show the way to new theorems and (computer-free) proofs of them.

## What kinds of things can MAGMA deal with?

- Sets/sequences/mappings
- Groups (finite/infinite)
- Codes/lattices
- Modules & algebras
- Lie algebras
- Combinatorial Functions
- Convex polytopes/polyhedra
- Elliptic curves
- Graphs & digraphs
- Rings & fields
- Block designs
- Representations
- Coxeter systems
- Homology computations
- Affine/projective planes
- Sheaves, varieties etc.

... all kinds of discrete computations

## Command and input/output features

MAGMA can be used interactively, or by writing a list of commands into an input file and running the in the foreground or background.

MAGMA has a huge library of commands that can be used, and these can be found in an indexed manual that is available not only on the web at

magma.maths.usyd.edu.au/magma/handbook

but also via the command line, e.g. by typing

```
?Cayley graph
```

Advertisement: Conference and MAGMA Workshop

"Symmetries of discrete objects"

www.math.auckland.ac.nz/∼conder/SODO-2102

Queenstown, New Zealand, 12–17 February 2012

Temporary MAGMA licences available for all participants

## Defining/creating/generating graphs

A specific graph can be constructed by specifying

- the edges (as a set of unordered pairs),
- the neighbours of each vertex, or
- the adjacency matrix.

Graphs on up to 50,000 vertices can be defined in this way.

It is also possible to generate all graphs on up to a given (small) number of vertices satisfying certain conditions, e.g.

— with bounds on number of edges

— with bounds on minimum or maximum degree

— only connected graphs

— only bipartite graphs.

## Example: Constructing the Petersen graph

The Petersen graph is the complement of the triangular graph $T(5)$, and so its vertices can be identified with pairs $\{i, j\}$ where $1 \leq i < j \leq 5$, with adjacency given by empty intersection. Hence the graph can be constructed as follows:

```
V := SetToSequence(Subsets({1..5},2));
E := { {s,t} : s,t in [1..#V] | #(V[s] meet V[t]) eq 0 };
P := Graph<#V | E>;
```

Note that the first command could be replaced by this:

```
V := SetToSequence({1,2}^Sym(5));
```

— but that could give a different labelling of the vertices.

## Example: Constructing Cayley graphs

Let $G$ be a finite group, defined by some generating set $S$ (with $1_G \notin S$). The command `D := CayleyGraph(G);` gives the Cayley digraph for the pair $(G, S)$, viz. the graph with vertex-set $G$ and arcs of the form $(g, gs)$ for $g \in G$ and $s \in S$, while the command `X := UnderlyingGraph(CayleyGraph(G));` gives the undirected Cayley graph $X = \mathrm{Cay}(G, S)$.

Alternatively, the undirected Cayley graph $X = \mathrm{Cay}(G, S)$ can be constructed as follows:

```
edges := {{g,s} : g in G, s in S};
X := Graph<Order(G) | edges>;
```

## What can Magma tell you about a graph?

There are numerous obvious basic functions, e.g.

- `VertexSet(X)`
- `MinimumDegree(X)`
- `IsRegular(X)`
- `IsConnected(X)`
- `Diameter(X)`
- `SpanningForest(X)`
- `CliqueNumber(X)`
- `IndependenceNumber(X)`
- `ChromaticNumber(X)`
- `EdgeSet(X)`
- `MaximumDegree(X)`
- `IsBipartite(X)`
- `Components(X)`
- `Girth(X)`
- `IsPlanar(X)`
- `MaximumClique(X)`
- `MaximumIndependentSet(X)`
- `ChromaticPolynomial(X)`

But also some helpful algebraic functions, e.g.

- `AdjacencyMatrix(X)`
- `AutomorphismGroup(X)`
- `IsTransitive(X)`
- `IsSymmetric(X)`
- `IsDistanceTransitive(X)`
- `Spectrum(X)`
- `IsIsomorphic(X,Y)`
- `IsEdgeTransitive(X)`
- `IsPrimitive(X)`
- `IsDistanceRegular(X)`

Most of these depend on computation of the automorphism group of the graph, which is achieved using an interface with Brendan McKay's `nauty` package. The latter is also used in computing other parameters such as diameter, and in generating all graphs of small order (with specified properties).

## Demonstrations

An interactive MAGMA session will show how some of these things work, for various examples:

- the Petersen graph
- the complete bipartite graph $K_{20,11}$
- a random graph on 2011 vertices
- a random cubic graph on 100 vertices

These and subsequent demonstrations are on a fairly standard laptop, so exhibit close to typical behaviour of MAGMA.

## Cayley graphs

MAGMA can generate all Cayley graphs of small order.

One way to do this is to use MAGMA's database of groups of order up to 2000 (not including 1024).

To obtain all 3-valent Cayley graphs of order $n$, run through the list of all groups of order $n$, and for each such group $G$, consider every possibility for the generating set $S$, which is of one of two forms:

- a set $\{a, b, c\}$ of three involutions in $G$, or
- a set $\{x, y, y^{-1}\}$ where $o(x) = 1$ and $o(y) > 2$.

A demonstration will show how fast MAGMA can find these graphs, up to isomorphism. Another approach will be shown later (in the third lecture).

## Building on MAGMA

It is possible (and easy) to extend the functionality of MAGMA by writing your own procedures and functions that use the commands and databases within the MAGMA package.

For example, there's a nice method (due to Nick Wormald) for finding a Hamilton path or cycle in a connected 3-regular finite graph, and a version is available in MAGMA code. This starts with a short path, that you might like to imagine is a 'snake', which extends itself by adding a vertex adjacent to its head. It keeps growing until the head meets the tail, and if it does not yet give a Hamilton path or cycle, then the snake bites off the last piece of its tail and continues ...

A demonstration will show how well (and quickly) this works.

# Lecture 2: **Permutation groups & matrix groups**

The beginnings of computational group theory in the 1960s and 70s focussed on two main themes: coset enumeration in finitely-presented groups, and algorithms for determining the order and structure of finite permutation groups.

Recent efforts have concentrated on developing methods for determining the order and structure of finite linear groups — that is, subgroups of $\mathrm{GL}_n(R)$ where $R$ is a ring.

We will look at permutation groups and linear groups in this lecture, and finitely-presented groups in the next one.

## Defining/generating permutation groups

A permutation group is any subgroup of the symmetric group $S_n$ for some $n$.

This may be a specific known example (e.g. the cyclic group $C_n$, the dihedral group $D_n$, the alternating $A_n$, or $S_n$ itself, or PSL$(2, q)$ as a subgroup of $S_{q+1}$ in its action on the $q+1$ points of the projective line L$(q)$ = GF$(q) \cup \{\infty\}$.

Or it could be an arbitrary subgroup of $S_n$, or one chosen from the subgroups that are transitive, or that are primitive.

MAGMA has a database of all transitive groups of degree 2 to 30, and one of all primitive groups of degree 2 to 2499.

## What can Magma tell you about a permutation group?

There are numerous basic functions giving properties of a permutation group $G$ or an element $g \in G$, e.g.

- `Degree(G)`
- `Orbits(G)`
- `AllPartitions(G)`
- `IsSymmetric(G)`
- `Order(G)`
- `IsTransitive(G)`
- `IsPrimitive(G)`
- `IsAlternating(G)`

- `Order(g)`
- `IsEven(g)`
- `Fix(g)`
- `CycleStructure(g)`
- `Sign(g)`
- `Degree(g)`

... and also many that deal with stabilizers, block systems, and so on, e.g.

- `Stabilizer(G,i)`
- `MinimalPartitions(G)`
- `WreathProduct(G,H)`
- `OrbitImage(G,O)`
- `BlocksImage(G,P)`
- `IsFrobenius(G)`
- `MaximalPartitions(G)`
- `IsSemiregular(G)`
- `OrbitKernel(G,O)`
- `BlocksKernel(G,P)`

where $i$ is a point, $O$ is an orbit, and $P$ is a $G$-invariant partition (block system), and $H$ is another permutation group.

Most of these functions depend on knowing a base and strong generating set for the permutation group $G$. These are key concepts introduced by Charles Sims in 1970.

## Base and strong generating set

Let $G$ be a permutation group on the set $X = \{1, 2, \ldots, n\}$.

A base for $G$ is an ordered set $B = (\beta_1, \beta_2, \ldots, \beta_r)$ of distinct points of $X$ such that the pointwise stabilizer $G_B = G_{\beta_1 \beta_2 \ldots \beta_r}$ of $B$ is trivial.

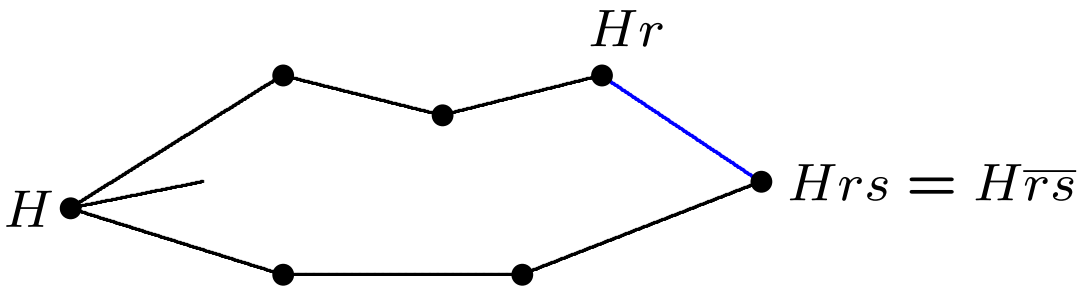
A strong generating set for $G$ with respect to the base $B$ is a subset $S \subseteq G$ such that for $0 \leq i \leq r$, the elements in $S$ that fix $B_i = (\beta_1, \beta_2 \ldots \beta_i)$ generate the pointwise stabilizer $G_{\beta_1 \beta_2 \ldots \beta_i}$ of $B_i$, that is, $\langle\, S \cap G_{\beta_1 \beta_2 \ldots \beta_i} \,\rangle = G_{\beta_1 \beta_2 \ldots \beta_i}$.

For example, for the dihedral group $D_n$ acting naturally on the vertices of a regular $n$-gon, a base consists of any two adjacent vertices, and then a strong generating set consists of a rotation of order $n$ and a reflection fixing the first vertex.

## Schreier-Sims algorithm

This is an algorithm for finding a base and strong generating set (BSGS) for any finite permutation group.

It was developed by Sims using Schreier's subgroup lemma:




*If $H$ is a subgroup of $G = \langle S \rangle$, and $R$ is a (right) transversal for $H$ in $G$, then $H$ is generated by $\{rs(\overline{rs})^{-1} : r \in R, s \in S\}$, where $\overline{rs}$ is the element of $R$ representing the coset $Hrs$.*

[This is important, and we'll see it again in the 3rd lecture]

The Schreier-Sims algorithm is fast and effective. And once a base and strong generating set are known, it is easy to derive a lot of other useful information about a permutation group $G$, e.g.

- the index $|G_{B_{i-1}} : G_{B_i}|$ of each stabilizer in the previous
- $|G| = |G : G_{B_1}||G_{B_1} : G_{B_2}| \ldots |G_{B_{r-1}} : G_{B_r}|$ (the order of $G$)
- stabilizers of all subsets
- conjugacy classes of elements
- normal subgroups, subnormal series, composition series
- transversals for arbitrary subgroups
- intersections of arbitrary subgroups
- maximal subgroups, other permutation representations.

## Demonstrations

An interactive MAGMA session will show how some of these things work, for various examples:

- the dihedral groups $D_5$ and $D_6$
- the symmetric group $S_5$
- the Mathieu groups $M_{12}$ and $M_{24}$
- the group PSL$(2, 7)$, as a permutation group of degree 8
- constructing new actions (e.g. action on cosets)
- selections from the transitive groups database
- selections from the primitive groups database

## Defining/generating matrix groups

A matrix group is any subgroup of the general linear group $\mathrm{GL}_n(R)$ for some $n$ and some ring $R$ (usually a field).

Again, this may be a specific known example (such as the special linear group $\mathrm{SL}_n(R)$ or the orthogonal group $O_n(R)$), or just a subgroup of $\mathrm{GL}_n(R)$ generated by given elements.

Algorithms for investigating matrix groups are somewhat less well-developed than those for permutation groups, but are the subject of a worldwide 'matrix group recognition' project which has made great advances recently.

MAGMA now has sophisticated methods for recognising particular matrix groups or their subgroups or representations.

## Standard families of matrix groups

MAGMA has in-built definitions of members of several well-known families of matrix groups, such as the following:

- `GeneralLinearGroup(n,q)`
- `GeneralUnitaryGroup(n,q)`
- `GeneralOrthogonalGroup(n,q)`
- `SOPlus(n,q)`
- `Omega(n,q)`
- `ReeGroup(q)`
- `SpecialLinearGroup(n,q)`
- `SpecialUnitaryGroup(n,q)`
- `SpecialOrthogonalGroup(n,q)`
- `SOMinus(n,q)`
- `SymplecticGroup(n,q)`
- `SuzukiGroup(q)`

... plus others and many variants of both these and their projective images (e.g. `ProjectiveGammaLinearGroup(n,q)`).

## Methods for dealing with matrix groups

The simplest (and until now the standard) way to deal with a matrix group is to convert it into a permutation group, e.g. using the action of $\mathrm{GL}_n(F)$ on the vector space $F^n$.

But unless $n$ and $F$ are small, this can become impractical (e.g. for subgroups of $\mathrm{GL}_5(81)$ the degree is 3486784401).

Modern methods are based on other approaches, including

- random sampling of elements, for probabilistic recognition
- explicit recognition of particular subgroups (e.g. $\mathrm{SL}_2(q)$)
- Aschbacher's classification of maximal subgroups of $\mathrm{GL}_n(q)$.

There are also various clever methods (based on little more than undergraduate algebra) for finding things like the order of a given element in a matrix group over a finite field.

## What can MAGMA tell you about a matrix group?

Aside from the usual functions available for finite groups, MAGMA has a number of commands that can help with the study of a given matrix group $G$ or an element $g \in G$, e.g.

- `Orbits(G)`
- `Stabilizer(G,v)`
- `IsIrreducible(G)`
- `Order(g)`
- `Trace(g)`
- `MinimalPolynomial(g)`
- `LineOrbits(G)`
- `OrbitClosure(G,S)`
- `MinimalField(G)`
- `ProjectiveOrder(g)`
- `Determinant(g)`
- `CharacteristicPolynomial(g)`

where $v$ is a point and $S$ is a subset of the vector space.

## Demonstrations

An interactive MAGMA session will show how some of these things work, for various examples:

- the group $\mathrm{GL}_2(5)$ and its subgroup $\mathrm{SL}_2(5)$
- subgroups of $\mathrm{GL}_2(27)$
- the group $\mathrm{SO}_5(3)$
- the Suzuki group Sz(32).

## Groups of small order

MAGMA has a database of all groups of order up to 2000 (except 1024). Accessing the list of all groups of order $n$ is easy, using the functions

- `NumberOfSmallGroups(n)`
- `SmallGroup(n,j)` ... which gives the $j$th group of order $n$

The groups are stored in different ways: some as permutation groups, and some as 'PC-groups' (solvable groups defined by power-commutator presentations). But it is easy to convert from one form to another — and whichever form is used, it is possible to find out information about the group using the wide array of procedures and functions available.

# Lecture 3: **Finitely-presented groups**

This lecture deals with ways of investigating groups defined by generators and relations, such as the triangle groups

$$\Delta(2,p,q) = \langle\, x,y,z \,|\, x^2 = y^p = z^q = xyz = 1 \,\rangle$$

associated with regular maps (of type $\{p,q\}$ in this case).

Given a group $G$ with finite presentation $G = \langle\, X \,|\, R \,\rangle$, there are methods for

- finding the order of $G$ (when this is finite)
- enumerating cosets of a finitely-generated subgroup of $G$
- obtaining a presentation for a finitely-generated subgroup
- finding all subgroups of up to a given index in $G$
- finding all quotients of $G$ of up to a given order
  and all nilpotent quotients of $G$ of up to a given class.

## Summary of important functions for f.p. groups

- `ToddCoxeter(G,H)` ... gives a coset table for $H$ in $G$
- `Order(G)` ... attempts to find the order of $G$
- `Rewrite(G,H)` ... finds a presentation for the subgroup $H$
- `LowIndexSubgroups(G,n)` ... finds subgroups of index $\leq n$
- `LowIndexNormalSubgroups(G,n)` ... finds normal subgroups of index $\leq n$
- `AbelianQuotient(G)` ... finds the abelianization $G/G'$
- `pQuotient(G,p,c)` ... finds $p$-quotients of $G$ of class $\leq c$
- `NilpotentQuotient(G,c)` ... finds nilpotent quotients of $G$ of class $\leq c$

# Coset enumeration

Let $G = \langle X \,|\, R \rangle$, and let $H$ be the subgroup generated by some finite set $Y$ of words on the alphabet $X = \{x_1, \ldots, x_m\}$.

Methods exist for systematically enumerating the cosets $Hg$ for $g \in G$. It is helpful to store these in a coset table, which shows the effect of multiplying each (numbered) coset $Hg$ by a generator $x_i$ or its inverse ${x_i}^{-1}$:

| | $x_1$ | $x_2$ | $\ldots$ | $x_1^{-1}$ | $x_2^{-1}$ | $\ldots$ |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | | 4 | | |
| 2 | | | | 1 | | |
| 3 | | | | | 1 | |
| 4 | 1 | | | | | |
| $\vdots$ | | | | | | |

Each relation from the defining presentation $\langle X \,|\, R \rangle$ for $G$ forces pairs of cosets to be equal: $Hgr = Hg$ for all $g \in G$.

The same thing happens on application of each generator $y \in Y$ to the trivial coset $H$: $Hy = H$.

New cosets are defined (if needed), and all such coincidences are processed, until the coset table either 'closes' or has too many rows.

If the coset table closes with $n$ cosets, then $|G : H| = n$. Moreover, the coset table gives us the natural permutation representation of $G$ on the right coset space $(G:H)$.

If it does not close, then the index $|G:H|$ could be infinite, or just too large to be found (or it might even be small but the computation was not given enough resources).

## Schreier coset graphs

Suppose the group $G$ is generated by $X = \{x_1, x_2, \ldots, x_m\}$.

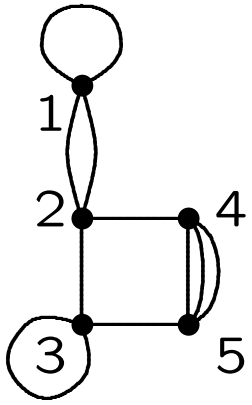
Given any transitive permutation representation of $G$ on a set $\Omega$ of size $n$, we may form a graph with vertex-set $\Omega$, and with edges of the form $\alpha$ — $\alpha x_i$ for $1 \leq i \leq m$.

Similarly, if $H$ is a subgroup of index $n$ in $G$, we may form a graph whose vertices are the right cosets of $H$ and whose edges are of the form $Hg$ — $Hgx_i$ for $1 \leq i \leq m$.

These two graphs are the same when $\Omega$ is the right coset space $(G : H)$, and $H$ is the stabilizer of a point of $\Omega$. It is called the Schreier coset graph $\Sigma(G, X, H)$.

## **Schreier coset graphs** (cont.)

The Schreier coset graph $\Sigma(G, X, H)$ gives a diagrammatic representation of the natural action of $G$ on cosets of $H$, and hence is equivalent to the coset table, e.g. as follows:



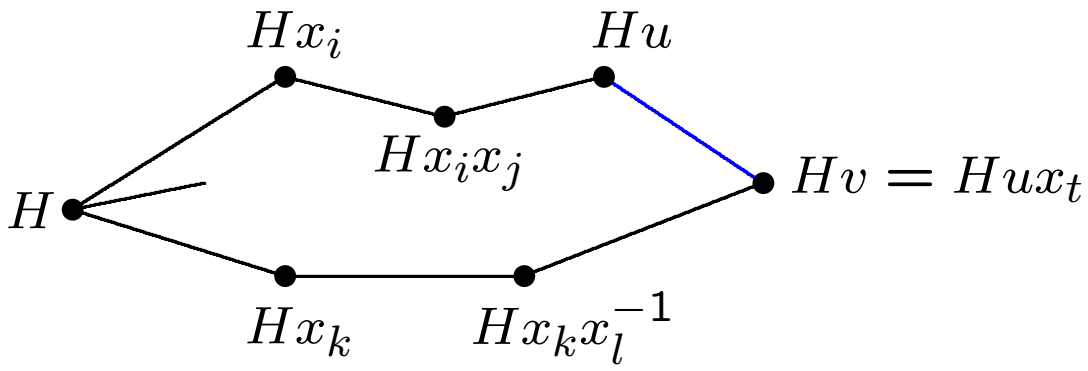

| | $x$ | $y$ | $x^{-1}$ | $y^{-1}$ |
|---|---|---|---|---|
| 1 | 2 | 1 | 2 | 1 |
| 2 | 1 | 3 | 1 | 4 |
| 3 | 3 | 5 | 3 | 2 |
| 4 | 5 | 2 | 5 | 5 |
| 5 | 4 | 4 | 4 | 3 |

when $x \mapsto (1,2)(4,5)$ and $y \mapsto (2,3,5,4)$

## Some observations

1) A spanning tree for the coset graph $\Sigma$ gives a Schreier transversal $T$ for $H$ in $G$

2) Edges of the coset graph not used in the spanning tree give a Schreier generating-set for $H$ in $G$ :




The edge $Hu$ — $Hv$ given by multiplication by $x_i$ gives the Schreier generator $ux_iv^{-1} = ux_i(\overline{ux_i})^{-1}$ for $H$.

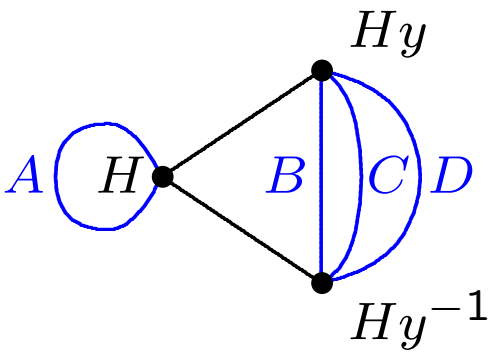
## The Reidemeister-Schreier process

Given a finitely-presented group $G = \langle X \mid R \rangle$ and a subgroup $H$ of finite index in $G$, Reidemeister-Schreier theory provides a method for obtaining a presentation for $H$ (in terms of generators and relations):

1) Construct the coset graph — using the coset table
2) Take a spanning tree in the coset graph — this gives a Schreier transversal for $H$ in $G$
3) Label the unused edges with Schreier generators
4) Apply each of the relators from $R$ to each of the cosets in turn, to obtain the defining relations for $H$.

## Example

Let $G = \langle\, x, y \,|\, x^2, y^3 \,\rangle$, and let $H$ be the stabilizer of 1 in the permutation representation $x \mapsto (2,3)$, $y \mapsto (1,2,3)$:




Schreier generators

$A = x$

$B = y^3 \ (= 1)$

$C = yxy$

$D = y^{-1}xy^{-1}$

Relation $x^2 = 1$ gives new relations $A^2 = 1$ and $CD = 1$

Relation $y^3 = 1$ gives new relation $B = 1$

Thus $H$ has presentation $\langle\, A, C \,|\, A^2 \,\rangle$ via $A = x$ and $C = yxy$.

## Proving finitely-presented groups are infinite

There are several ways of proving a finitely-presented group $G = \langle X \mid R \rangle$ is infinite, with the help of MAGMA:

- Show the abelianisation $G/G'$ is infinite
- Check to see if $G$ has more generators than relations
- Find a subgroup of $G$ with infinite abelianisation
- Construct an epimorphism onto a known infinite group

Note: the third of these depends on having a collection of subgroups to check—such as all subgroups of small index.

# Low index subgroup methods

Again, let $G = \langle X \,|\, R \rangle$ be a finitely-presented group, and suppose we want to find all subgroups of small index in $G$.

Subgroups of index $\leq n$ can be found (up to conjugacy) by a systematic enumeration of coset tables with $\leq n$ rows.

The 'low index subgroups' algorithm starts with the identity subgroup and attempts to enumerate its right cosets. Then (or at any later stage) if more than $n$ cosets are defined, all possible concidences between two cosets are considered.

This sets up a branching process for a backtrack search, which is guaranteed to complete (given sufficient time and memory), by Schreier's subgroup lemma!

## Low index normal subgroups

Small homomorphic images of a finitely-presented group $G$ can be found as the groups of permutations induced by $G$ on cosets of subgroups of small index. This gives $G/K$ where $K$ is the core of $H$, but produces only images that have small degree faithful permutation representations.

Alternatively, the (standard) low index subgroups method can be adapted to produce only normal subgroups.

A new method was developed recently by Derek Holt and his student, which systematically enumerates the possibilities for the composition series of the factor group $G/K$, for any normal subgroup $K$ of small index in $G$.

# Some references

W. Bosma, J. Cannon & C. Playoust, The Magma Algebra System I: the user language, *J. Symbolic Comput.* 24 (1997), 235–265.

M.D.E. Conder, Combinatorial and computational group-theoretic methods in the study of graphs, maps and polytopes with maximal symmetry, in: *Applications of Group Theory to Combinatorics* (ed. J. Koolen, J.H. Kwak & M.Y. Xu), Taylor & Francis, London, 2008, pp. 1–11.

D.F. Holt, B. Eick & E.A. O'Brien, *Handbook of Computational Group Theory*, CRC Press, 2005.

J. Neubüser, *An elementary introduction to coset-table methods in computational group theory*, Groups – St. Andrews 1981, London Math. Soc. Lecture Note Series, vol. 71, 1982, pp. 1–45.

C.C. Sims, *Computation with finitely presented groups,* Encyclopedia of Mathematics & its Applications, vol. 48. Cambridge Univ. Press, 1994.