Computing in finitely presented groups

Derek Holt

University of Warwick

Bonn, September 2018

LECTURE 1: Resources

Charles C. Sims. Computation with Finitely Presented Groups, CUP, 1994.

Derek Holt, Bettina Eick, and Eamonn O'Brien. Handbook of Computational Group Theory. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2005.

Alexander Hulpke. Notes on Computational Group Theory, Chs III and IV. `www.math.colostate.edu/~hulpke/CGT/cgtnotes.pdf`

Derek Holt. Lecture notes on presentations of groups. `https://warwick.ac.uk/fac/sci/maths/people/staff/fbouyer/presentation_of_group.pdf`

Max Neunhöffer's 2013 lectures on same topic: `http://www-groups.mcs.st-andrews.ac.uk/cgt2013/lectures.shtml`

Software: **GAP**, **Magma** for Computational Group Theory. KBMAG (available from **GAP**, **Magma**) for Knuth-Bendix completion and automatic groups software.'

Introduction

Groups can be represented on a computer in a number of ways:

- (finite) groups of permutations;
- groups of matrices over a finite or infinite field (or commutative ring);
- power-conjugate presentations of polycyclic groups;
- finite presentations with generators and defining relators.

For the first three of these, each group element has a unique representation, and a multitude of algorithms exist to carry out structural computations in the groups, particularly (but not entirely) for finite groups.

These three lectures will concern the fourth class, for which we do not in general have a unique or canonical representation of groups elements. Indeed the problem of deciding whether two words in the group generators represent the same group element is undecidable in general.

Finitely presented groups

From now on, we assume that the group G is defined by a presentation $G = \langle X \mid R \rangle$ with X and R finite.

The set X of **generators** is a just a finite set of symbols. We let $A := X \cup X^{-1}$, where X^{-1} is a set of formal inverses of the elements of X.

The set of **words** or **strings** over A is denoted by A^*, and we use 1 to denote the empty word.

$R \subset A^*$ is the set of **defining relators** of G, and by definition we have

$$G := F(X)/\langle\langle R \rangle\rangle,$$

where $F(X)$ is the group freely generated by X, and $\langle\langle R \rangle\rangle$ denotes the normal closure of R in $F(X)$.

Tietze transformations

In examples, we often use **relations** rather than **relators**. A relation is an equation $w_1 = w_2$ with $w_1, w_2 \in A^*$, which we can regard as being equivalent to the relator $w_1 w_2^{-1}$.

Example

$$G = \langle a, b \mid aba = bab \rangle = \langle a, b \mid abab^{-1}a^{-1}b^{-1} \rangle.$$

Presentations can be manipulated using **Tietze transformations**

In the above example, we could introduce new generators $x := ab$ and, $y := aba$, which enable us to eliminate a, b, and replace the defining relation by $x^3 = y^2$, so

$$G \cong \langle x, y \mid x^3 = y^2 \rangle.$$

The Dehn problems

These are decision problems that were formulated by Max Dehn in 1911. They are basically theoretical questions, but Dehn described implementable algorithms for their solution in some cases.

- **The Word Problem** WP(G): given $w \in A^*$, is $w =_G 1$?
- **The Conjugacy Problem**: given $v, w \in A^*$, does there exists $c \in A^*$ with $w =_G c^{-1}vc$?
- **The Isomorphism problem**: given another finitely presented group $G' = \langle X' \mid R' \rangle$, is $G \cong G'$?
  (This is the most difficult, both theoretically and practically.)

These have all been proved to be undecidable in general. For the first two problems, there are specific examples of groups G in which they are undecidable. For isomorphism, we cannot even decide whether G is isomorphic to the trivial group.

Computing the largest abelian quotient

What we can do is to compute various types of quotient groups of G.

For $G = \langle X|R\rangle$, the largest abelian quotient $G_{\mathtt{ab}} = G/[G,G]$ of G has the presentation $G = \langle X|R \cup C\rangle$, where $C = \{[x,y] : x,y \in X\}$.

This could also be regarded as the quotient $\mathrm{FA}(X)/\langle R\rangle$ of the free abelian group $\mathrm{FA}(X)$ on the set X.

It is convenient to switch to additive notation, and regard the elements of R as row vectors in $\mathbb{Z}^n$, where $|X| = n$. Then the group $G_{\mathtt{ab}}$ is represented by an $m \times n$ matrix over $\mathbb{Z}$, where $m = |R|$.

We can then calculate the abelian invariants of $G_{\mathtt{ab}}$ by unimodular row and column operations, resulting in the **Smith Normal Form** of the matrix.

Computing the largest abelian quotient: example

Example

Calculate G_{ab} with

$$G = \langle a, b, c, d \mid a(bd^{-1})^2,\ (bc)^2,\ d^2a^{-1}(b^{-1}c)^4 \rangle.$$

$G = \mathbb{Z}^4/H$, where H is the subgroup of $\mathbb{Z}^4$ spanned by the rows of

$$\begin{pmatrix} 1 & 2 & 0 & -2 \\ 0 & 2 & 2 & 0 \\ -1 & -4 & 4 & 2 \end{pmatrix}$$

which we can reduce to Smith Normal Form using row and column operations:

$$\xrightarrow{r3:=r3+r1} \begin{pmatrix} 1 & 2 & 0 & -2 \\ 0 & 2 & 2 & 0 \\ 0 & -2 & 4 & 0 \end{pmatrix} \xrightarrow{r3:=r3+r2} \begin{pmatrix} 1 & 2 & 0 & -2 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 6 & 0 \end{pmatrix}$$

$$\xrightarrow{c2:=c2-2c1} \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 6 & 0 \end{pmatrix} \xrightarrow{c3:=c3-c2} \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 6 & 0 \end{pmatrix}$$

$$\xrightarrow{c4:=c4+2c1} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 6 & 0 \end{pmatrix}$$

So

$$G_{\mathrm{ab}} \cong \mathbb{Z}/\mathbb{Z} \oplus \mathbb{Z}/(2\mathbb{Z}) \oplus \mathbb{Z}/(6\mathbb{Z}) \oplus \mathbb{Z} \cong \mathbb{Z}/(2\mathbb{Z}) \oplus \mathbb{Z}/(6\mathbb{Z}) \oplus \mathbb{Z}.$$

For larger matrices, problems arise with exponentially growing integer size. These can be ameliorated by using the LLL-algorithm.

Other quotient algorithms

Other quotient algorithms include algorithms to compute:

- finite *p*-quotients for a specified prime *p*;
- nilpotent quotients;
- polycyclic quotients;
- solvable quotients.

For the first three of these, power-conjugate presentations of the quotients are returned, which facilitate structural computations.

The *p*-quotient algorithm has been used to:

- compute restricted **Burnside groups** $\mathrm{RB}(2,5)$ and $\mathrm{RB}(2,7)$;
- enumerate all finite *p*-groups up to orders 2^{10}, p^7.

Subgroups of finite index (finite quotients)

Let $G := \langle X \mid R \rangle$. There is a bijection

$$\begin{array}{ccc}
\{H \le G \mid [G:H] < \infty\} & \xrightarrow{\cong} & \{\varphi : G \to S_n \mid n \in \mathbb{N}, \varphi \text{ a } \textbf{grp. hom.}\} \\
 & & \text{(image transitive)} \\
H & \longmapsto & \textbf{trans. action} \text{ on } \{Hg \mid g \in G\} =: H\backslash G \\
\varphi^{-1}(\mathrm{Stab}_{S_n}(1)) & \longleftarrow\!\shortmid & \varphi
\end{array}$$

which respects

$$H = K^x \text{ for some } x \in G \iff \textbf{actions} \text{ on } \{Hg \mid g \in G\} \text{ and } \{Kg \mid g \in G\} \textbf{ equivalent}$$

The conjugacy classes of finite index subgroups

are in bijection with

the equivalence classes of actions on finitely many points.

Coset tables

We can describe finite actions by **coset tables**:

Example (A coset table)

$$\begin{aligned} \text{Let } G &:= \left\langle c, d \mid c^2 = 1 = d^3 = (cd)^7 = [c,d]^4 \right\rangle \\ &\cong L_2(7) \cong \langle (1,2)(4,5), (2,3,4)(5,6,7) \rangle \le S_7. \end{aligned}$$

Coset #	c	c^{-1}	d	d^{-1}
1	2	2	1	1
2	1	1	3	4
3	3	3	4	2
4	5	5	2	3
5	4	4	6	7
6	6	6	7	5
7	7	7	5	6

Here, $H = \left\langle d, cdcd^{-1}c \right\rangle$.

Todd-Coxeter coset enumeration

Let $G := \langle X \mid R \rangle$ and $H = \langle h_1, \ldots, h_k \rangle < G$.

Idea of coset enumeration

We construct the permutation action of G on the right cosets of H.
We assign numbers to right cosets and make sure that

- multiplication by elements of R fixes all cosets, and
- multiplication of H by the generators of H fixes the first coset, H.

A "**name**" of a coset is a number and a word representing the coset.

We make up new names and draw conclusions as we go and hope that the process completes.

Coset enumeration - a worked example

Let $G := \left\langle a, b \mid a^3, b^3, abab \right\rangle$ and $H := \langle a \rangle$.

Events:

#	coset	a	a^{-1}	b	b^{-1}
1	H				

We start with an empty table like this.

Coset enumeration - a worked example

Let $G := \langle a, b \mid a^3, b^3, abab \rangle$ and $H := \langle a \rangle$.

#	coset	a	a^{-1}	b	b^{-1}
1	H	1	1		

Note $Ha = H$, since $a \in H$, this is a deduction.

Events:

Ded. $1a = 1$ from $a \in H$

Coset enumeration - a worked example

Let $G := \langle a, b \mid a^3, b^3, abab \rangle$ and $H := \langle a \rangle$.

#	coset	a	a^{-1}	b	b^{-1}
1	H	1	1	2	
2	Hb				1

We call the coset Hb number 2, a definition.
Note that $2 = 1b$ is equivalent to $2b^{-1} = 1$.

Events:

Ded. $1a = 1$ from $a \in H$
Def. $2 := 1b$.

Coset enumeration - a worked example

Let $G := \left\langle a, b \mid a^3, b^3, abab \right\rangle$ and $H := \langle a \rangle$.

#	coset	a	a^{-1}	b	b^{-1}
1	H	1	1	2	
2	Hb			3	1
3	Hbb				2

Events:

Ded. $1a = 1$ from $a \in H$
Def. $2 := 1b$
Def. $3 := 2b$

A new definition $3 = 2b = Hbb$ and equivalently $3b^{-1} = 2$.

Coset enumeration - a worked example

Let $G := \langle a, b \mid a^3, b^3, abab \rangle$ and $H := \langle a \rangle$.

$\#$	coset	a	a^{-1}	b	b^{-1}
1	H	1	1	2	3
2	Hb			3	1
3	Hbb			1	2

Events:

Ded. $1a = 1$ from $a \in H$
Def. $2 := 1b$
Def. $3 := 2b$
Ded. $1 = 3b$ from $1b^3 = 1$

Now we get a deduction $3b = 1$ from $3b = Hb^3 = H = 1$.

Coset enumeration - a worked example

Let $G := \langle a, b \mid a^3, b^3, abab \rangle$ and $H := \langle a \rangle$.

#	coset	a	a^{-1}	b	b^{-1}
1	H	1	1	2	3
2	Hb	3		3	1
3	Hbb		2	1	2

Events:

Ded. $1a = 1$ from $a \in H$
Def. $2 := 1b$
Def. $3 := 2b$
Ded. $1 = 3b$ from $1b^3 = 1$
Ded. $3 = 2a$ from $1abab = 1$

Another deduction $3 = 2a$ since $1abab = 1 \Rightarrow 2a = 1aba = 1b^{-1} = 3$.

Coset enumeration - a worked example

Let $G := \langle a, b \mid a^3, b^3, abab \rangle$ and $H := \langle a \rangle$.

$\#$	coset	a	a^{-1}	b	b^{-1}
1	H	1	1	2	3
2	Hb	3	4	3	1
3	Hbb		2	1	2
4	Hba^{-1}	2			

A new definition $4 := 2a^{-1} = Hba^{-1}$.

Events:

Ded. $1a = 1$ from $a \in H$
Def. $2 := 1b$
Def. $3 := 2b$
Ded. $1 = 3b$ from $1b^3 = 1$
Ded. $3 = 2a$ from $1abab = 1$
Def. $4 := 2a^{-1}$

Coset enumeration - a worked example

Let $G := \langle a, b \mid a^3, b^3, abab \rangle$ and $H := \langle a \rangle$.

$\#$	coset	a	a^{-1}	b	b^{-1}
1	H	1	1	2	3
2	Hb	3	4	3	1
3	Hbb	4	2	1	2
4	Hba^{-1}	2	3		

Events:

Ded. $1a = 1$ from $a \in H$
Def. $2 := 1b$
Def. $3 := 2b$
Ded. $1 = 3b$ from $1b^3 = 1$
Ded. $3 = 2a$ from $1abab = 1$
Def. $4 := 2a^{-1}$
Ded. $4 = 3a$ from $2a^3 = 2$

Deduction $4 = 3a$ since $2a^3 = 2 \Rightarrow 4 = 2a^2 = 2a^{-1} = 3$.

Coset enumeration - a worked example

Let $G := \langle a, b \mid a^3, b^3, abab \rangle$ and $H := \langle a \rangle$.

$\#$	coset	a	a^{-1}	b	b^{-1}
1	H	1	1	2	3
2	Hb	3	4	3	1
3	Hbb	4	2	1	2
4	Hba^{-1}	2	3	4	4

Events:

Ded. $1a = 1$ from $a \in H$
Def. $2 := 1b$
Def. $3 := 2b$
Ded. $1 = 3b$ from $1b^3 = 1$
Ded. $3 = 2a$ from $1abab = 1$
Def. $4 := 2a^{-1}$
Ded. $4 = 3a$ from $2a^3 = 2$
Ded. $4 = 4b$ from $3abab = 3$
Table closed.

Deduction $4 = 4b$ since $3abab = 3 \Rightarrow 4b = 3ab = 3b^{-1}a^{-1} = 4$.
Table is closed.

Indeed, we have found a permutation representation on 4 points. The subgroup H fixes the first point.

Since we have checked all relations we have found a group homomorphism from G to S_4.

Coset enumeration - coincidences

Sometimes we get a deduction $k = ia$ for some cosets numbered k, i and $a \in A$ when we already know that $j = ia$ for some $j \neq k$.

This is called a **coincidence**. It means that the cosets numbered j and k are equal.

To deal with this, if say $k > j$, then we replace every k in the coset table by j, and we also have to merge the rows of the table that are labelled j and k.

This merging process typically results in further coincidences, and then we have to form a **coincidence queue** of coincidences to be processed.

This is tricky to program efficiently, and there are a number of technical pitfalls.

Coset enumeration - strategies

When we need to make a new definition, we have a lot of choice as to what we define next.

An algorithm for deciding what to define next is called a **strategy**.

A good strategy will run as fast as possible but, for large enumerations, in which space might be an issue, it should try to minimize coincidences. These two aims may conflict.

Many strategies have been implemented. The two most basic are:

1. **HLT** (Haselgrove-Leech-Trotter): define cosets while scanning relators. This is often fast on easy examples but does not attempt to avoid coincidences.
2. **Felsch**: define cosets by filling rows of the coset table and then making all possible deductions. This usually results in fewer coincidences, but is sometimes slower.

Coset enumeration - strategies and variants

The implementation ACE (Havas & Ramsay) allows combinations of the two basic strategies in user-specified proportions, together with some other more complicated strategies.

It has been used in successful coset enumerations with $|G : H| \approx 10^{10}$.

However:

No strategy is optimal on all examples.

Runtime and memory usage can vary enormously with the strategy.

Algorithms related to coset enumeration

Coset enumeration can also be used to enumerate representatives of the conjugacy classes of all subgroups of G of index up to a specified positive integer n. This is known as the **low index subgroups** algorithm.

Unlike coset enumeration itself, it is guaranteed to complete, although its complexity appears to be worse than exponential in n.

Another application, originally due to John Cannon is to find a presentation of a given finite group of known order, such as a subgroup of $\mathrm{Sym}(n)$.

The idea is to start with some easy relations satisfied by the group generators, such as x^3, $(xy)^2$, etc, and use coset enumeration to try and prove that we have a complete presentation.
If it fails to complete quickly, then we interrupt the enumeration, add a new relator, and resume.

The first implementations used coset enumeration over the trivial subgroup, but later ones were able to handle larger groups by using nontrivial subgroups.

LECTURE 2: Presentations of subgroups: theory

Recall: $G = \langle X \mid R \rangle$, and $G \cong F/N$ where $F = F(X)$ and $N = \langle\langle R \rangle\rangle$.

Let $H = E/N < G$ and let $\tilde{T} \subseteq F$ be a right transversal of E in F that maps onto the right transversal T of H in G:

$$F = \dot{\bigcup_{\tilde{t} \in T}} E\tilde{t} \quad \text{and thus} \quad G = \dot{\bigcup_{t \in T}} Ht$$

We assume $1_F \in \tilde{T}$. For $w \in F$, define $\overline{w} := t \in T$ with $w \in Et$.

Theorem (Nielsen-Schreier)

$E = \langle Y \rangle$ and hence $H = \langle yN : y \in Y \rangle$, where

$$Y := \left\{ tx(\overline{tx})^{-1} \mid t \in T, x \in X, tx \neq \overline{tx} \right\} \subseteq F \setminus \{1_F\}$$

If T is prefix-closed, then E is freely generated by Y.

The Reidemeister-Schreier Algorithm

Theorem (Reidemeister-Schreier)

Suppose that G, F, H, E, T are as above with T prefix closed. Then

$$H \cong \langle Y \mid \rho(twt^{-1}) \text{ for all } t \in T, w \in R \rangle,$$

where $\rho : E \to F(Y)$ is the isomorphism mentioned above.

Thus, if $|G : H|$ is finite and we have a coset table for H in G, then we can compute the Schreier generators and write down this presentation for H explicitly. This is the **Reidemeister-Schreier Algorithm**.

It is convenient to do this following a coset enumeration of H in G.

The following example illustrated how we can define T and Y and calculate the relators $\rho(twt^{-1})$ in practice.

Example (The symmetric group S_4)

Let $G := \langle x, y \mid x^3, y^4, (xy)^2 \rangle$, and $H := \langle x,\ yx^{-1}y^{-2} \rangle$. First, do coset enumeration. Now the coset table is

	x	y	x^{-1}	y^{-1}
1	1	$\underline{2}$	1	3
2	$\underline{3}$	$\underline{4}$	4	1
3	4	1	2	4
4	2	3	3	2

We have underlined the first occurrence of each coset number $i > 1$ in the images under x, y. We can think of these as definitions, and they can be used to define the elements of T corresponding to the four cosets, where coset 1 is always assigned the identity element 1.

Then the elements of $\tilde{T}$ representing the cosets 1, 2, 3, and 4 are respectively 1_F, y, yx, and y^2.

Example (The symmetric group S_4 (ctd))

Now, we can define the elements of the free generating set Y of E by inserting new symbols into the remaining entries on the first two columns of the coset table.

	x	y	x^{-1}	y^{-1}
1	$a1$	$\underline{2}$	$a^{-1}1$	$c^{-1}3$
2	$\underline{3}$	$\underline{4}$	$d^{-1}4$	1
3	$b4$	$c1$	2	$e^{-1}4$
4	$d2$	$e3$	$b^{-1}3$	2

So now $Y = \{a, b, c, d, e\}$ with

$$a = x, \quad b = yx^2y^{-2}, \quad c = yxy, \quad d = y^2xy^{-1}, \quad e = y^3x^{-1}y^{-1}.$$

Example (The symmetric group S_4 (ctd))

We calculate the relators $\rho(twt^{-1})$ of H by scanning the cosets of H in G under the relators of G using the modified coset table.

	x		x		x	
1		$a1$		$a^2 1$		$a^3 1$
2		3		$b4$		$bd2$

	y		y		y		y	
1		2		4		$e3$		$ec1$

	x		y		x		y	
1		$a1$		$a2$		$a3$		$ac1$
3		$b4$		$be5$		$beb5$		$(be)^2 2$
4		$d2$		$d4$		$d^2 2$		$d^2 4$

Example (The symmetric group S_4 (ctd))

So

$$H \cong \langle a, b, c, d, e \mid a^3, bd, ec, ac, (be)^2, d^2 \rangle$$

We can now perform Tietze transformations on this presentation and use the relators bd, ec, ac to eliminate $d = b^{-1}$, $e = c^{-1}$, $c = a^{-1}$, to give the presentation

$$H \cong \langle a, b \mid a^3, (ba)^2, b^2, \rangle$$

which is a standard presentation of the dihedral group of order 6 (and also of the symmetric group S_3).

Finding presentations on the user supplied generators is also possible, but we have to do the calculation of the subgroup presentation while we are doing the coset enumeration. When the index is large, this can result in very long relators in the subgroup presentation.

Application to proving groups infinite

Problem

Let $G := \langle X \mid R \rangle$. How could we prove that $|G| = \infty$?

First idea: Find abelian invariants of G, but what if they are all finite?

Second idea:

- Compute some low index subgroups of G
- Use Reidemeister-Schreier to find presentations for them.
- Compute the abelian invariants of these presentations.
- If we find an infinite factor, then the group G is infinite as well.

Example

$$G := \langle x, y \mid x^3, y^3, (xy)^3 \rangle; \quad H := \langle x^{-1}y, yx^{-1} \rangle.$$

Show that $|G : H| = 3$, calculate a presentation of H and show that H has abelian invariants $[0, 0]$. (In fact $H \cong \mathbb{Z}^2$.)

Computing in the whole group

Everything that we have discussed far concerns computing in various types of quotients of a finitely presented group $G = \langle X \mid R \rangle$.

We know from the undecidability of the Dehn problems (the word, conjugacy and isomorphism problems) that the most natural questions about the group G itself are undecidable in general.

The rest of this short course will be about attempts to answer such questions in specific classes of groups.

Small cancellation theory

Let $G := \langle X \mid R \rangle$, and assume that all $r \in R$ are cyclically reduced. Let $\hat{R}$ be the set of all rotations and inversions of the elements of R.

For example, if $R = \{a^3, b^3, aba^{-1}b^{-1}\}$, then (with $\bar{a} := a^{-1}$, $\bar{b} = b^{-1}$):

$$\hat{R} = \{a^3, \bar{a}^3, b^3, \bar{b}^3, ab\bar{a}\bar{b}, b\bar{a}\bar{b}a, \bar{a}\bar{b}ab, \bar{b}ab\bar{a}, ba\bar{b}\bar{a}, a\bar{b}\bar{a}b, \bar{b}\bar{a}ba, \bar{a}ba\bar{b}\}.$$

Definition (Piece)

A **piece** of the presentation is a nonempty word p that is a prefix of two different elements of $\hat{R}$; i.e.: $pu, pv \in \hat{R}$ for $u, v \in A^*$ with $u \neq v$.

Definition (Condition $C'(\lambda)$ for $1/\lambda \in \mathbb{N}$)

We say that the presentation $\langle X \mid R \rangle$ is $C'(\lambda)$ if, whenever p is a piece of $r \in \hat{R}$, then $|p| < |\lambda| \cdot |r|$.

Small cancellation theory (ctd)

Example

$$G = \left\langle a, b, c, d \mid a^7, b^7, c^7, d^7, a^{-1}b^{-1}abc^{-1}d^{-1}cd \right\rangle .$$

The pieces are $a^{\pm 1}, b^{\pm 1}, c^{\pm 1}, d^{\pm 1}$, and G satisfies $C'(1/6)$.

Definition (Condition $T(q)$)

We say that $\langle X \mid R\rangle$ is $T(q)$, if the following holds:

- Let $3 \le h < q$ and $(r_1, r_2, \ldots, r_h) \in \hat{R}^h$ with no successive elements $r_i,\ r_{i+1}$ or r_h, r_1 an inverse pair. Then at least one of the products $r_1r_2, r_2r_3, \ldots, r_hr_1$ is reduced without cancellation.

$T(q)$ says that all vertices in any reduced van Kampen diagram for the presentation have valency at least q.

Dehn's algorithm

Theorem (Lyndon, Schupp)

Let $G = \langle X \mid R \rangle$. where all $r \in R$ are cyclically reduced. If $\langle X \mid R \rangle$ fulfills at least one of:

- *$C'(1/6)$, or*
- *$C'(1/4)$ and $T(4)$, or*
- *$C'(1/3)$ and $T(6)$,*

then **Dehn's algorithm** *solves the word problem for $G = \langle X \mid R \rangle$.*

But what is Dehn's algorithm?

Rewrite systems

Rewrite systems arise in a wide variety of situations, such as text editing and formal language theory as well as group theory!

Definition (Rewrite system)

Let A be a finite set, known as the **alphabet**. A **rewrite system (RWS)** $\mathcal{R}$ over A is a set of **rules** $u \to v$ with $u, v \in A^*$.

We apply a rule $u \to v$ in $\mathcal{R}$ to a word $w \in A^*$ by replacing an occurrence of u as a subword of w by v. We then write $w \to_{\mathcal{R}} w'$ (or usually just $w \to w'$) where w' is the result of the replacement.

In other words, we write $svt \to swt$ for all $s, t \in A^*$.

We write $w \Rightarrow_{\mathcal{R}} w'$ if either $w = w'$, or w' results from w by a sequence of applications of rewrite rules from $\mathcal{R}$. That is: $w = w'$ or there is a finite tuple $(w_1, w_2, \ldots, w_k)$ of words with

$$w \to w_1 \to w_2 \to \cdots \to w_k \to w'$$

Dehn's algorithm

Definition (Dehn RWS)

For a given finite presentation $G = \langle X \mid R \rangle$, write all $r \in \hat{R}$ as $r = ab$ with $|a| > |b| \geq |a| - 2$ and define a rewrite rule $a \to b^{-1}$ over $A = X \cup X^{-1}$. We define the **Dehn** RWS $\mathcal{R}$ of the presentation to be the set of these rules together with the rules $aa^{-1} \to 1$ for all $a \in A$.

We say that Dehn's algorithm solves the word problem for $G = \langle X \mid R \rangle$ if, for all $w \in A^*$,

$$w =_G 1 \iff w \Rightarrow_{\mathcal{R}} 1$$

More precisely, we require that, whenever $w =_G 1$, we have $w \Rightarrow_{\mathcal{R}} 1$ irrespective of the order in which the rewrite rules are applied to w.

Since the rewrite rules are all length reducing, we can test whether $w \Rightarrow_{\mathcal{R}} 1$ in linear time.

So we can solve the word problem in groups satisfying various small cancellation hypotheses in linear time.

Dehn's algorithm: example

Consider the word $w = a^4b^6abc^{-1}d^{-1}cd^{-6}a^4$ in the example above. Then

$$a^{-1}b^{-1}abc^{-1}d^{-1}cd \in R \Rightarrow abc^{-1}d^{-1}cda^{-1}b^{-1} \in \hat{R}$$

so there is a rewrite rule $abc^{-1}d^{-1}c \to ba^{-1}d^{-1}$.

Applying this to w gives

$$w = a^4b^6\underline{abc^{-1}d^{-1}c}d^{-6}a^4 \to a^4b^6\underline{ba^{-1}d^{-1}}d^{-6}a^4 = a^3b^7a^{-1}d^{-7}a^4.$$

Now $b^7, d^{-7}, a^7 \in \hat{R}$ yield rewrite rules $b^4 \to b^{-3}$, $d^{-4} \to d^3$, $a^4 \to a^{-3}$, and applying these to w, followed by applications of $b^{-1}b \to 1$, etc, gives:

$$a^4b^7d^{-7}a^4 \to a^4b^{-3}b^3a^{-1}d^3d^{-3}a^4 \to a^7 \to a^{-3}a^3 \to 1$$

so $w =_G 1$.

Hyperbolic groups

It turns out that there is a whole collection of properties of groups G that are equivalent to the existence of a finite presentation of G for which Dehn's algorithm solving the word problem.

Groups with these properties were first studied by Gromov and are generally known as **hyperbolic groups** or sometimes **word hyperbolic** or **Gromov hyperbolic** groups.

Theorem

The following conditions are equivalent for groups $G = \langle X \rangle$ with X finite.

- *There is a set of relators $R \subset A^*$ such that $G = \langle X \mid R \rangle$ and the Dehn algorithm of this presentation solves the word problem for G;*
- *The* **Dehn function** *of G (w.r.t any set of defining relators) is linear;*
- *Geodesic triangles in the Cayley graph $\Gamma(G, A)$ of G are uniformly slim (or uniformly thin);*
- *various other conditions.*

Software for hyperbolic groups and Dehn's algorithm

There is a fast procedure, available in **Magma** and as a **GAP** package, due to Richard Parker, Colva Roney-Dougal, Max Neunhöffer, Markus Pfeiffer and others that attempts to prove that a given finitely presented group is hyperbolic and, if so, to calculate a set of rewrite rules for a Dehn algorithm.

It works on group presentations that satisfy the above-mentioned small cancellation conditions, and on many other, but not all hyperbolic groups.

We shall be discussing **automatic groups** later. KBMAG can compute these and then attempt to verify that the group is hyperbolic. This is much slower than the other procedure, but succeeds on more examples.

Also, KBMAG does not immediately compute a Dehn presentation for the group (which would solve the word problem in linear time), although it does compute data that can solve this problem in quadratic time.

LECTURE 3: Rewrite systems

Definition (Rewrite system)

Let A be a finite set, known as the **alphabet**. A **rewrite system (**RWS**)** $\mathcal{R}$ over A is a set of rules $u \to v$ with $u, v \in A^*$.

We write $svt \to swt$ for all $s, t \in A^*$.
We write $w \Rightarrow w'$ if either $w = w'$, or there is a finite tuple $(w_1, w_2, \ldots, w_k)$ of words with

$$w \to w_1 \to w_2 \to \cdots \to w_k \to w'$$

A word $v \in A^*$ is called **irreducible**, if there is no $w \in A^*$ with $v \to w$.

A RWS is called **terminating**, if there is no infinite chain of words $w_1 \to w_2 \to w_3 \to \cdots$.

Definition (Confluence and completeness)

A RWS $\mathcal{R}$ is called **confluent**, if for all $q, r, s, t \in A^*$ with $q \Rightarrow r$ and $q \Rightarrow s$ there is a $t \in A^*$ with $r \Rightarrow t$ and $s \Rightarrow t$.

$\mathcal{R}$ is **complete** if it is terminating and confluent.

$\mathcal{R}$ is called **locally confluent** if, for all $q, r, s, t \in A^*$ with $q \to r$ and $q \to s$ there is a $t \in A^*$ with $r \Rightarrow t$ and $s \Rightarrow t$.

Lemma

A terminating and locally confluent RWS *is complete.*

In a complete RWS, *every $w \in A^*$ can be rewritten to a unique irreducible word.*

Dehn algorithms are terminating, but not usually complete, because words w with $w \neq_G 1$ can sometimes be rewritten to more than one irreducible word.

Question

How could it happen, that $q \to r$ and $q \to s$, but that r and s cannot be rewritten to any common word t?

If this can happen, then there must be a word q to which we can apply two distinct rewrite rules $v_1 \to w_1$ and $v_2 \to w_2$.

Suppose first that the subwords v_1 and v_2 of q do not overlap. Then $q = xv_1yv_2z$ for some $x, y, z \in A^*$, and

$$\begin{tikzcd}
 & xw_1yv_2z \arrow[dr] & \\
xv_1yv_2z \arrow[ur] \arrow[dr] & & xw_1yw_2z \\
 & xv_1yw_2z \arrow[ur] &
\end{tikzcd}$$

So the problem can only arise if the left hand sides of the two rules overlap.

Critical pairs

Definition (Critical pair)

A pair of rules $v_1 \to w_1$ and $v_2 \to w_2$ is called a **critical pair**, if:

- $v_1 = rs$ and $v_2 = st$ for some $r, s, t \in A^*$, or
- $v_1 = rst$ and $v_2 = s$ for some $r, s, t \in A^*$,

with $s \neq 1$ in both cases.

In the first case we have $rst \to w_1 t$, $rst \to rw_2$, and in the second case $rst \to w$, $rst \to rw_2 t$.

Lemma

A RWS is locally confluent if and only if the following conditions are fulfilled for all critical pairs $v_1 \to w_1$ and $v_2 \to w_2$:

- *If $v_1 = rs$ and $v_2 = st$, then $\exists w \in A^*$ with $w_1 t \Rightarrow w$ and $rw_2 \Rightarrow w$.*
- *If $v_1 = rst$ and $v_2 = s$ then $\exists w \in A^*$ with $rw_2 t \Rightarrow w$ and $w_1 \Rightarrow w$.*

Reduction orderings

So we can check local confluence, and hence **completeness**, of a finite, terminating RWS algorithmically.

Definition (Reduction ordering)

A well-ordering on A^* is called a **reduction ordering**, if $u \leq v$ implies $uw \leq vw$ and $wu \leq wv$ for all $u, v, w \in A^*$.

Example

shortlex orderings: choose a total ordering on A, and order A^* first by length and then lexicographically.
For example, if $A = \{a, b, c\}$ with ordering $b < c < a$, then
$1 < b < c < a < bb < bc < ba < cb < cc < ca < ab < ac < aa < bbb < \cdot\cdot$

Let $<$ be a reduction ordering, and let $\mathcal{R}$ be a RWS in which $v > w$ for all rules $v \to w$. Then $w_1 > w_2$ whenever $w_1 \to w_2$, and so $\mathcal{R}$ is terminating.

The Knuth-Bendix completion procedure

Start with a finite RWS and choose a reduction ordering such that $v > w$ for all rules $v \to w$.

Consider all possible critical pairs $rs \to w_1$ and $st \to w_2$, and:

- rewrite $w_1 t \Rightarrow w_1'$ and $rw_2 \Rightarrow w_2'$ with w_1' and w_2' irreducible,
- if $w_1' \neq w_2'$, then
  - add to $\mathcal{R}$ the rule $w_1' \to w_2'$ if $w_1' > w_2'$,
  - or add to $\mathcal{R}$ the rule $w_2' \to w_1'$ if $w_2' > w_1'$.

Similarly for critical pairs $rst \to w_1$ and $s \to w_2$.

This procedure is not guaranteed to terminate: it may continue to add new rules indefinitely.

But if it terminates, then the resulting RWS is locally confluent, and hence complete.

Knuth-Bendix and groups

Let $G = \langle X \mid R \rangle$ and $A = X \cup X^{-1}$. We can present G as a monoid by adding relations $aa^{-1} = 1$ for all $a \in A$.

Choose a reduction order on A^* and define an RWS to have one rule $v \to w$ with $v > w$ for each relation $v = w$ in the monoid presentation. Then apply the Knuth-Bendix completion procedure.

If it terminates with a complete RWS, then group elements are uniquely represented by irreducible words, and we can solve the word problem in G by reducing words to their unique irreducible representatives.

The procedure is guaranteed to complete with a finite set of rules if G is finite. Occasionally this happens when G is infinite, but this occurs relatively rarely, and may depend on the right choice of reduction ordering.

For example, for a polycyclic group G, there is a type of ordering known as a **recursive path** ordering, with which the procedure completes, and the resulting rewrite system consists of the relations of a **power-conjugate presentation** for G.

As a simple example, consider the nonabelian group of order 6.

Example

$$G = \langle a, b \mid a^2, b^3, (ab)^2 \rangle,\ A = \{a, b, \bar{b}\},\ \bar{b} := b^{-1};$$
$$\text{shortlex ordering: } a < b < \bar{b}.$$

Initial rules from presentation:

$$a^2 \to 1,\ b\bar{b} \to 1,\ \bar{b}b \to 1,\ b^2 \to \bar{b},\ ba \to a\bar{b}$$

From overlap of left hand sides b^2 and $b\bar{b}$, we get:

$$b^2\bar{b} \to b,\ b^2\bar{b} \to \bar{b}^2 \quad \textbf{new rule} : \bar{b}^2 \to b$$

From overlap of left hand sides b^2 and ba, we get:

$$b^2a \to \bar{b}a,\ b^2a \to ba\bar{b} \to a\bar{b}^2 \to ab \quad \textbf{new rule} : \bar{b}a \to ab$$

The rewrite system of 7 rules is now complete:

$$a^2 \to 1,\ b\bar{b} \to 1,\ \bar{b}b \to 1,\ b^2 \to \bar{b},\ \bar{b}^2 \to b,\ ba \to a\bar{b},\ \bar{b}a \to ab$$

The Knuth-Bendix completion procedure is a useful tool for computing with groups.

Unfortunately it does not usually terminate on infinite groups. But it can help solve positive instances of the word problem in the following sense.

We can run Knuth-Bendix as described above on $G = \langle X \mid R \rangle$, and stop at some stage with an enlarged RWS $\mathcal{R}$. Then:

(i) If $w \Rightarrow_{\mathcal{R}} 1$ then $w =_G 1$.

(ii) Conversely, if $w =_G 1$ then, if we run Knuth-Bendix for long enough, we will eventually have $w \Rightarrow_{\mathcal{R}} 1$.

We turn now to another approach to solving the word problem, which works in many classes of infinite groups.

Finite state automata

A **finite state automaton (fsa)** is a simple computing machine M which takes as input strings in a fixed finite alphabet A.

The machine is always in one of a finite set Σ of states, one of which is designated as the start state.

The letters in the string are read successively by M. If it is in state s and reads the letter a, then it changes into the state $\tau(s, a)$, where $\tau : \Sigma \times A \to \Sigma$ is a function known as the **transition function** for M.

Some of the states are designated as **accepting** (or **final**). If M is in an accepting state after reading the string w, then it outputs the answer **yes** and is said to accept w. Otherwise it outputs the answer **no**, and rejects w. These are the only two possible outputs.

The subset of A^* consisting of the accepted strings is called the **language** $L(M)$ of M.

Finite state automata: example

Example

In this example $A = \{a, a^{-1}, b, b^{-1}\}$, $\Sigma = \{1, 2, 3, 4, 5, 0\}$, the start state is 1, the accepting states are $\{1, 2, 3, 4, 5\}$, and the transition function τ is given by the table below.

	a	a^{-1}	b	b^{-1}
1	2	3	4	5
2	2	0	4	5
3	0	3	4	5
4	2	3	4	0
5	2	3	0	5
0	0	0	0	0

The accepted language $L(M)$ is the set of reduced words in A^*.

We can also represent M as a labelled directed graph. We have omitted the failure state 0 and its associated transitions, and written $\bar{a}$ and $\bar{b}$ for a^{-1} and b^{-1}.

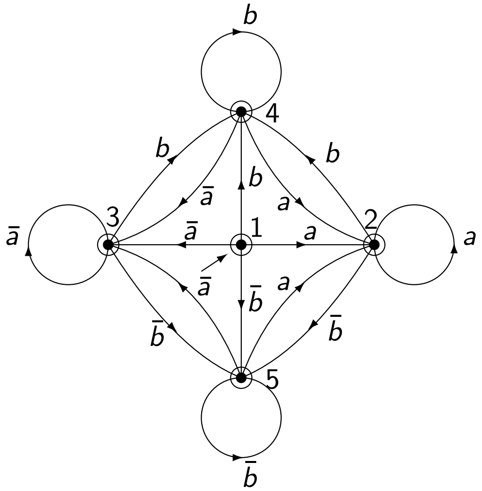


Figure: fsa recognising the reduced words in $F_2 = \langle a, b \mid \rangle$

Word acceptors

If $G = \langle X|R \rangle$ is a finitely presented group, and $A = X \cup X^{-1}$, then a fsa with input alphabet A is called a **word-acceptor** for G if it accepts at least one word in A^* for each group element.

It is called a **unique word-acceptor** for G if it accepts a unique word for each group element.

The above example is a unique word acceptor for the free group $F(\{a, b\})$. It is not difficult to construct a unique word acceptor for a group for which we have a finite complete rewriting system.

One of the standard operations that one can perform on an fsa M is to enumerate its language $L(M)$ (usually up to a given word-length). In particular, we can enumerate the elements of a finitely presented group have a unique word-acceptor.

It is also straightforward to determine $|L(M)|$ (which may be infinite) and hence, if we have a unique word acceptor for G, then we can calculate $|G|$.

Automatic groups

The definition of the class of **automatic groups** evolved in the mid-1980's following the appearance of a paper by Jim Cannon on hyperbolic groups. Bill Thurston noticed that some of the properties that were proved could be reformulated in terms of finite state automata.

The following definition involves fsa's that have to read two words in A^* at a time at the same speed. We can almost achieve this by making the input alphabet $A \times A$, but there is a problem if the two words do not have the same length. To solve this, we adjoin a symbol \$ to A, and put $A' := A \cup \{\$\}$. For $u, v \in A^*$, we append \$'s to the shorter of the two words, to make their lengths equal.

Denote the new words by u^+ and v^+ (where at most one of these is different from u and v). Then we call the pair $(u, v)^+ := (u^+, v^+)$ a **padded pair**. Since its left and right hand sides have the same length, it can be regarded as a word in $(A' \times A')^*$.

Example

$u = aba^{-1}$, $v = bab^{-1}aa$, $(u, v)^+ = (a, b)(b, a)(a^{-1}, b^{-1})(\$, a)(\$, a)$.

Definition

The group G is said to be **automatic** (with respect to X), if there exist fsa's W, and M_a for each $a \in A \cup \{1\}$, such that:

(i) W has input alphabet A, and is a (unique) word acceptor for G;

(ii) each M_a has input alphabet $A' \times A'$, it accepts only padded pairs, and it accepts $(w, v)^+$ for $w, v \in A^*$ if and only if $w, v \in L(W)$ and $wa =_G v$.

The idea is that the fsa M_a recognises multiplication on the right by the generator a within the language $L(W)$.

The set of automata $\{W, M_a\}$ is called an **automatic structure** for G.

Groups in the following classes are automatic:

- Hyperbolic groups, including free groups
- Virtually abelian groups
- Geometrically finite hyperbolic groups [Neumann & Shapiro, 95]
- Right-angled Artin groups, Artin groups of large type
- Coxeter groups [Brink & Howlett, 93]
- Garside groups, including braid groups [Charney & Meier, 04]
- Direct and free products of automatic groups
- Subgroups and supergroups of finite index in automatic groups
- **NOT** polycyclic groups that are not virtually abelian.

The group is said to be **shortlex automatic** with respect to X, if the fsa W accepts precisely the minimal words for the group elements under some shortlex ordering (which depends on the chosen ordering for A).

It turns out that automaticity of G does not depend of the choice of the generating set X, but shortlex automaticity does.

Automatic groups - an equivalent definition

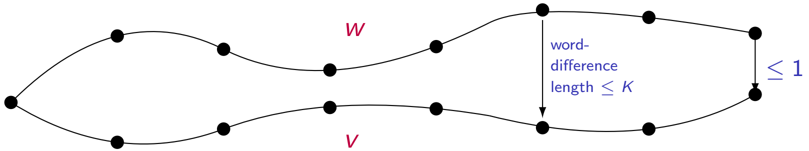
Definition

G is **automatic** if there exists an fsa W and a constant K such that:

(i) W has input alphabet A, and is a (unique) word acceptor for G;

(ii) if $w, v \in L(W)$ with $w^{-1}v \in A \cup \{1\}$, then w and v (synchronously) K-fellow travel in the Cayley graph $\Gamma := \Gamma(G, A)$ of G.
That is, $d_\Gamma(w(i), v(i)) \leq K$ for all $i \geq 0$.

The finite collection of group elements $w(i)^{-1}v(i)$ arising from relations $wa =_G v$ with $a \in A \cup \{1\}$ are called the **word-differences** of these relations.



A procedure to compute shortlex automatic structures

Input: A finite presentation of a group G;
Output: The automata in a shortlex automatic structure for G or **fail**.

Step 1 Run the Knuth-Bendix procedure (with shortlex ordering) on G until the set D of all word-differences arising from all rewrite rules $v \to w$ appears to have stabilized.

Then construct fsa W_D over $A' \times A'$ with state set D, start and accept state 1, and transitions $u \xrightarrow{(a,b)} v$ with $u, v \in D$ and $a, b \in A'$ with $a^{-1}ub =_G v$.

Step 2 Construct candidates for W and M_a as the fsa with

$$L(W) = \{w \in A^* : \nexists v \in A^* \text{ with } (w,v)^+ \in L(W_D) \text{ and } v < w\}.$$

$$L(M_a) = \{(w,v)^+,\ w, v \in A^* : w, v \in L(W),\ (w,v)^+ \in L(W_D),\ wa =_G v\}.$$

Step 3 Check the condition

$$\{\forall w \in L(W) : \exists v \in L(W) \text{ with } (w, v) \in L(M_a)\} = L(W)$$

for all $a \in A \cup \{1\}$. If it fails then enlarge the set D and repeat Step 2.

Step 4 Verify the correctness of the computed automatic structure by checking that the M_a satisfy the group relations.

Applications of automatic groups software

Once the shortlex automatic structure has been computed we can use it to:

- determine whether the group is finite or infinite;
- reduce words quickly (quadratic time) to a word in $L(W)$, and hence solve the word problem in quadratic time;
- enumerate normal form words up to a specified length.
- calculate the growth series of $G = \langle X \rangle$ as a rational function: $\sum_{n \geq 0} c_n x^n$, where c_n is number of group elements with shortest representatives in A^* of length (at most) n.

We have had significant success in resolving some difficult finiteness problems. For example the **Heineken group**

$$\langle x, y, z \mid [x,[x,y]] = z,\ [y,[y,z]] = x,\ [z,[z,x]] = y \rangle.$$

is infinite (and even hyperbolic),

and also for several difficult cases of groups in the Coxeter family

$$(\ell, m, n; p) = \langle x, y \mid x^\ell,\ y^m,\ (xy)^n,\ [x, y]^p \rangle,$$

such as $(\ell, m, n; p) = (3, 4, 13; 2)$ and $(3, 5, 7; 2)$.

Enumerating words has had applications to software for drawing pictures.

For example, the vertices in the picture below correspond to elements from the hyperbolic group

$$\begin{aligned}\langle a, b, c, d, e, f \mid\ & a^4 = b^4 = c^4 = d^4 = e^4 = f^4 = \\ & aba^{-1}e = bcb^{-1}f = cdc^{-1}a = ded^{-1}b = efe^{-1}c = faf^{-1}d = 1\rangle,\end{aligned}$$

which is the symmetry group of the tesselation in the picture.

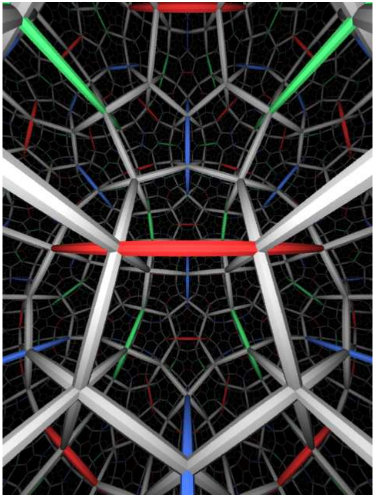


Figure: A tessellation of hyperbolic 3-space by dodecahedra

Other recent work

A variety of theoretical complexity results on decision problems in finitely presented groups have been proved, but many of these do not lead to immediately to effective algorithms, typically because they involve enormous constants.

For example, the conjugacy problem in hyperbolic groups is known to be solvable in linear time in hyperbolc groups. A version of this has been implemented (by Joe Marshall) that works reasonably quickly for elements of infinite order.

The **generalized word problem** (GWP) is to test membership of a given element $g \in G$ in a specified subgroup $\langle Y \rangle \leq G$ (with Y finite).

A generalization of the theory of automatic groups to the cosets of a subgroup enables practical algorthms to solve this in some cases.

The End