The Word Problem in the Chomsky Hierarchy

Sarah Rees,
University of Newcastle

Les Diablerets, March 8th-12th 2010

Abstract

I shall talk about the word problem for groups, and how hard it is to solve it. I'll view the word problem of $G = \langle X \rangle$ as the recognition of the set $\mathtt{WP}(G, X)$ of words over X that represent the identity, and look to relate properties of G to the complexity of $\mathtt{WP}(G, X)$ as a formal language, that is to the complexity of the Turing machine (model of computation) needed to recognise that set. I'll start by introducing the basic formal language theory that we need.

An elementary result identifies a group as finite precisely if its word problem is a regular language (recognised by a finite state automaton). A well known result of Muller and Schupp proves a group to be virtually free precisely if its word problem is context-free (recognised by a pushdown automaton), while Thomas and Herbst classify virtually cyclic groups using

one counter automata. So at this bottom end of the Chomsky hierarchy we have complete classification results. I shall survey these briefly.

Many more groups are admitted if the word problem is allowed to be co-context-free, such as abelian groups, but no nilpotent groups that are not virtually abelian, various wreath product groups and the Houghton groups. I shall report on various results by myself, Holt, Röver, Thomas, Lehnert and Schweitzer, and the techniques used to achieve them.

Many groups can be shown to have word problem that is soluble in linear space, and hence context-sensitive. In a second lecture I shall look at this class of groups, and at the subclasses with word problem that is growing context-sensitive or can be solved on a real-time Turing machine. In particular I'll examine Dehn's linear time algorithm for hyperbolic groups, which is growing context-sensitive, and can actually be programmed in real time. And I'll examine Cannon's generalisation of Dehn's algorithm, which

allows the word problem for nilpotent groups to be solved in real time. I'll illustrate with examples of groups in all these classes, and results that separate the classes, referring to results of myself, Holt, Röver, Goodman, Shapiro, Kambites and Otto.

Plan for the two lectures

Lecture I

Introduction
Groups with regular word problem
Context-free word problem
Indexed word problem
Co-context-free word problem
Co-indexed word problem

Lecture II

Context-sensitive word problem
Dehn's algorithm
Real-time word problem
Generalising Dehn - Cannon's algorithm
More on real-time
Growing context-sensitive

The word problem

$$G = \langle X \mid R \rangle$$

The word problem (Dehn 1912) for G is soluble if there's a terminating algorithm that, for any input word w (i.e. any string over $X^{\pm} := X \cup X^{-1}$) can decide whether or not $w =_G 1$.

$$w =_G 1 \iff w =_{F(X)} u_1 r_1 u_1^{-1} \dots u_k r_k u_k^{-1}, \quad r_i \in R, u_i \in F(X).$$

But rhs may be arbitrarily longer than w, so recognition of

$$\mathtt{WP}(G, X) := \{w : w =_G 1\}$$

may not be easy, and is insoluble in general (Novikov, Boone, 1950's), even when R is finite (so G finitely presented).

Dehn solved the word problem for surface groups; his algorithm applies more generally (to word hyperbolic groups).

How is the word problem solved?

- How much **time** do we need to solve it?
- How much **space** do we need to solve it?
- What kind of **algorithm** do we need?
- What kind of **automaton** (Turing machine) recognises $\mathtt{WP}(G, X)$?
- What kind of **grammar** defines $\mathtt{WP}(G, X)$, ie what sort of production rules $\xi \longrightarrow \eta$ over $X^{\pm} \cup V$ construct it from a start symbol $s_0 \in V$?
- How do the answers to these questions relate to the **geometry** of the group, **combinatorics** of its presentation, etc?

NB: When the word problem is soluble, its complexity is independent of choice of generators, and solution passes to fg subgroups, supergroups of finite index ('finite extensions').

Languages, automata and grammars

A **language** over A is a set of strings over A. (A^* for the set of all strings, ϵ for the empty string.)

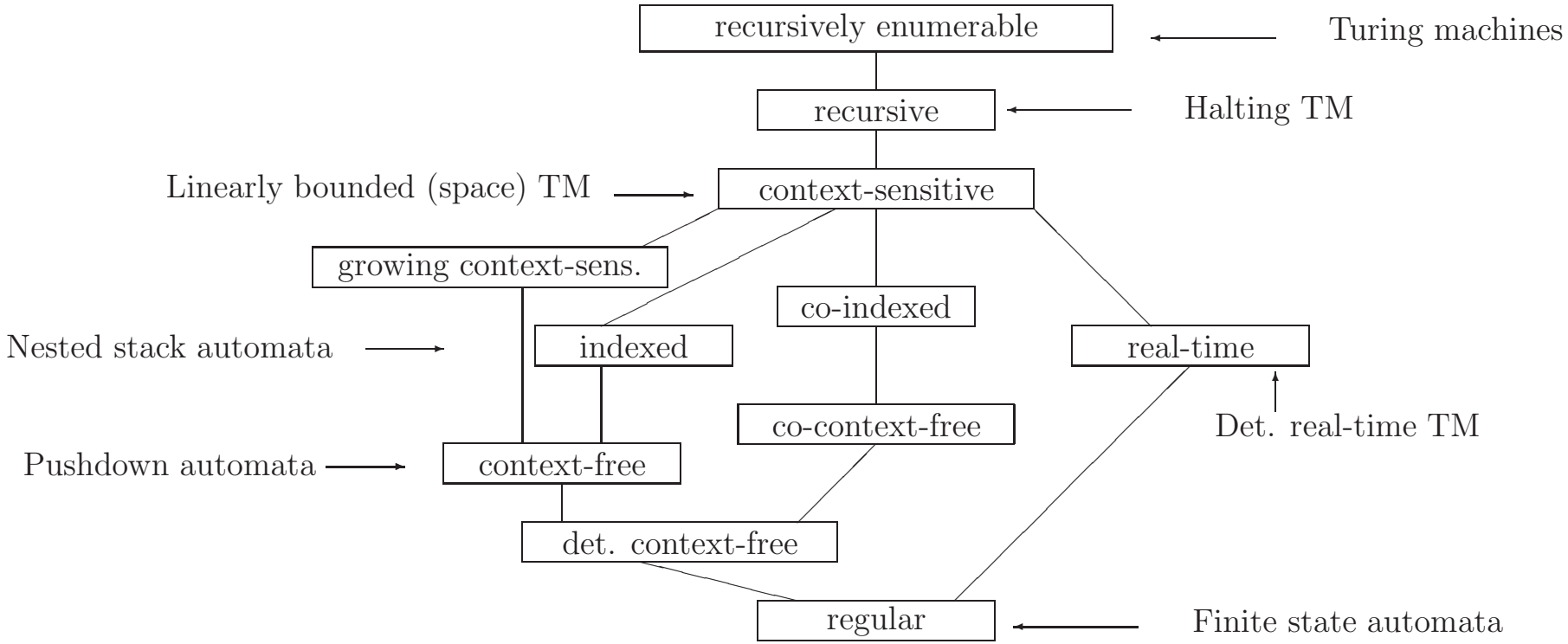
An **automaton** is a device that takes input strings from a tape, and accepts some of them; the set of strings it accepts is its **language**, and is **recognised** by the automaton.

An automaton might be a finite state automaton (FSA), pushdown automaton (PDA), or some other type of Turing machine(TM).

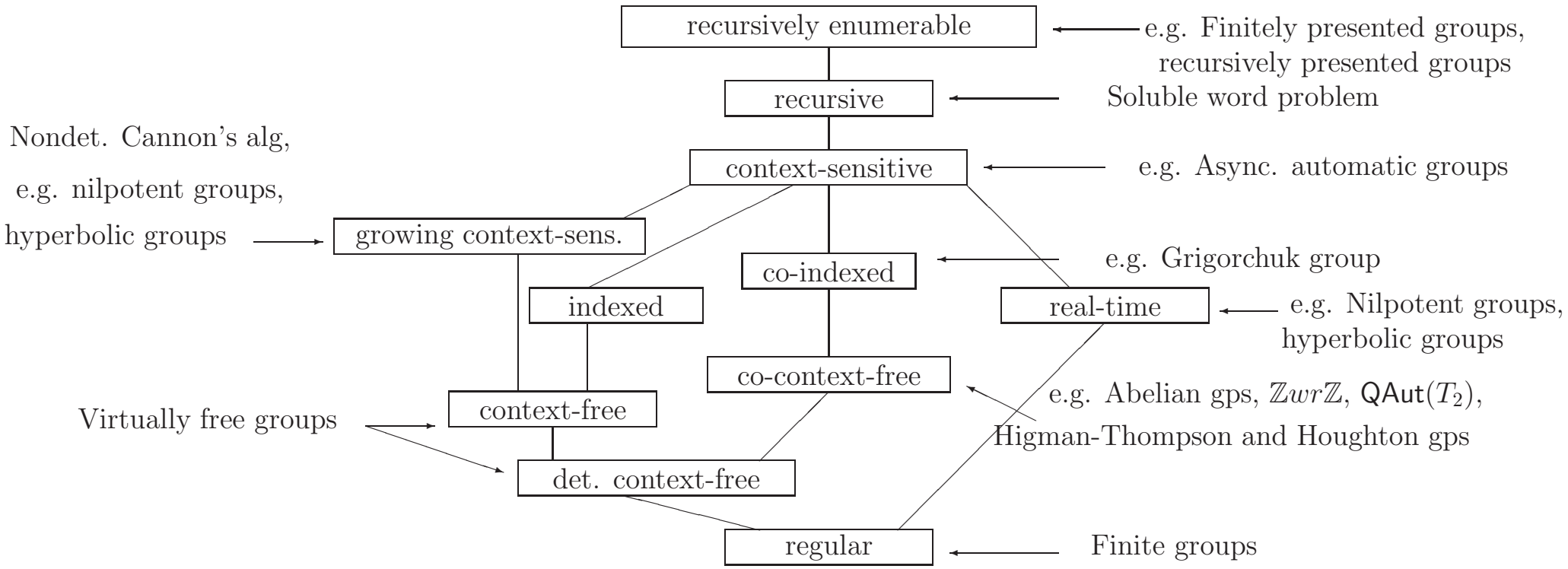
A **grammar** over A uses a set P of productions $\xi \longrightarrow \eta$, for ξ, η strings over $A \cup V$ to construct its **language**, the set of all strings over A derivable using P from a start symbol $s_0 \in V$.

Correspondences link the types of languages, automata and grammars.

Hierarchy of formal languages



Fitting $\mathtt{WP}(G, X)$ into the formal language hierarchy



FSA, regular languages and grammars

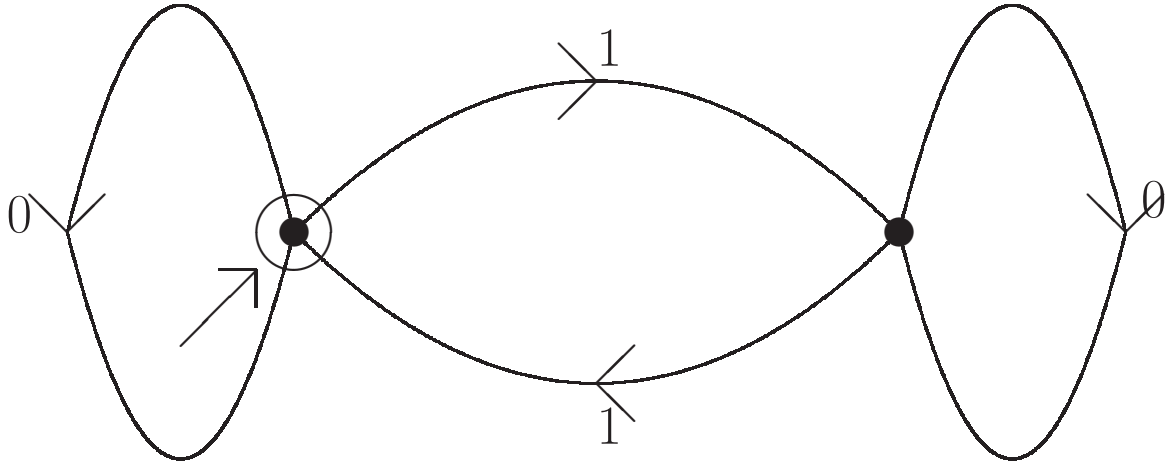
A language is regular iff accepted by a finite state automaton (FSA) iff generated by a regular grammar.

An FSA over A is a finite, directed graph, edges labelled by elements of A, with a **start** vertex and some **accepting** vertices. A word over A is accepted if it labels a path from the start vertex to an accepting vertex.

A grammar over A is (right)-regular if all of its productions have the form $x \longrightarrow \alpha y$ or $x \longrightarrow \alpha$, where $x, y \in V, \alpha \in A^*$.

Example:

The set EvenWt of all binary strings containing an even number of 1's is regular. It's recognised by an automaton with 2 states. And it's



generated by a grammar with variables s_0, s_1 and production rules:

$$s_0 \to \epsilon,\ s_0 \to 0s_0,\ s_0 \to 1s_1$$
$$s_1 \to 0s_1,\ s_1 \to 1s_0$$

Regular word problem

$\mathtt{WP}(G, X)$ is regular $\iff$ G is finite (Anisimov).

When G is finite, the Cayley graph $\Gamma(G, X)$ is an FSA over $X^{\pm}$ accepting $\mathtt{WP}(G, X)$, with 1 as the start and sole accepting vertex.

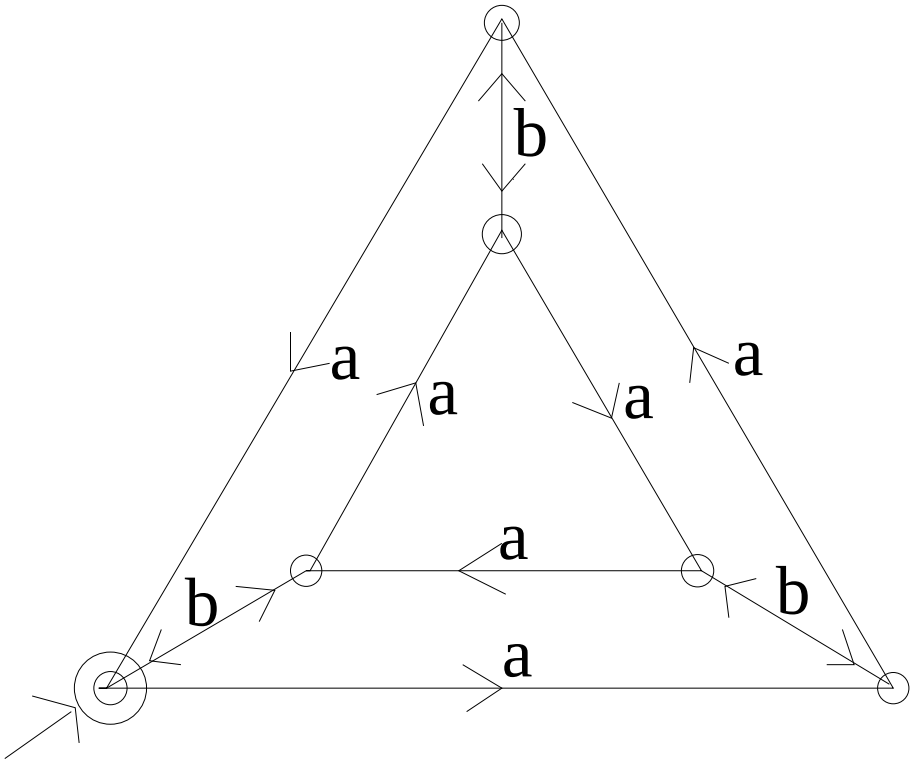
The corresponding grammar over $X^{\pm}$ has $V = G$, $s_0 = 1$, and productions

$$\overline{x} \to x, \quad g \to x\overline{(x^{-1}g)}$$

for each $x \in X^{\pm}$, and each $g \in G, g \neq \overline{x}$,
where $\overline{w}$ denotes the element of G represented by w.

Example:

$$S_3 = \langle a, b \mid a^3 = b^2 = 1, ba = a^3 b \rangle$$



PDA, context-free languages and grammars

A language is context-free (cf) iff accepted by a pushdown automaton (PDA) iff generated by a context-free grammar.

A PDA over A consists of a FSA with attached stack as additional memory. A move is determined by symbols read from both input string and top of stack. Then a string of any length may be added at the top of the stack.

Example:

The set BALANCED of all binary strings with equal numbers of 0's and 1's is context-free.

It's recognised by a PDA with 3 states that record whether more 0's, more 1's or equal numbers of 0's and 1's have been read. The stack counts the difference.

A grammar over A is context-free if all of its productions have the form $x \to \xi$, $x \in V, \xi \in (A \cup V)^*$.

BALANCED is defined by a grammar with one variable s_0, and production rules

$$s_0 \to \epsilon \quad s_0 \to s_0 01 \quad s_0 \to s_0 10$$
$$s_0 \to 0 s_0 1 \quad s_0 \to 1 s_0 0$$
$$s_0 \to 01 s_0 \quad s_0 \to 10 s_0$$

A context-free grammar is in Chomsky form if all productions have the form $x \to yz$ or $x \to a$, where $x, y, z \in V$, $a \in A$. Additionally $s_0 \to \epsilon$ is allowed if s_0 is never in a rhs. Every cf language can be generated (inefficiently) by a grammar in Chomsky form.

Solving WP(G) on a PDA for G virtually free

Free reduction of a word over free generators in F_n can be done on a PDA. For virtually free groups, we need states too.

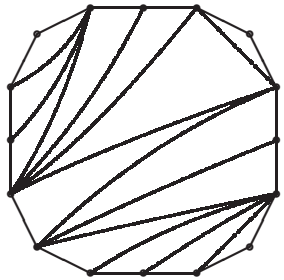
Where $G = \langle X \rangle$ contains $\langle Y \rangle$, free on Y, with T a finite transversal for that containing 1, the Reidemeister Schreier process provides rewrite rules

$$tx^{\delta} \to ut', \quad x \in X,\, \delta = \pm 1,\, t, t' \in T, u \in (Y^{\pm})^*$$

Using these rules any input word w over X can be rewritten from the left to the form vt, v a word over Y, $t \in T$.

During the rewrite process, we can use the states of a PDA to keep track of the transversal element, a stack to ensure that the word over Y is freely reduced. In that case

$$w =_G 1 \iff (v = \epsilon \quad \text{and} \quad t = 1)$$



Theorem (Muller,Schupp,1983)

$\mathtt{WP}(G, X)$ is context-free $\iff$ G is virtually free.

Proof:

(1) $\mathtt{WP}(G, X)$ is generated by grammar in Chomsky form, where each rule has the form $\alpha \to \beta\gamma$, $\alpha \to a$, or $s_0 \to \epsilon$. So any loop in the Cayley graph has a diagonal triangulation whose chords have bounded length. (In particular this implies G is finitely presented.)

(2) If infinite, the Cayley graph is disconnected by deletion of a ball of bounded radius, and so has more than one end.

(3) Stallings' ends theorem allows decomposition of G.

If G is torsion-free, then
either $G \cong \mathbb{Z}$
or $G \cong H * K$, H, K f.g., and of lower rank (Gruschko). In this case $H, K \in \mathtt{CF}$, so we finish by induction.

If G has torsion, then
$G \cong H *_A K$ or $H *_{A,\phi}$, with finite subgroup A; then we can finish proof by induction, using accessibility of fp groups (Dunwoody, 1985).

Adding non-determinisism to a PDA

Allowing non-determinism (a choice of moves for some input strings) increases the power of a PDA, i.e. dcf $\subsetneq$ cf.

Example: the set of all palindromes can only be accepted by a non-deterministic PDA.

We'll call the classes of groups with context-free and deterministic context-free word problems $\mathtt{CF}$ and $\mathtt{DCF}$.

Since the PDA described above to solve $\mathtt{WP}(G)$ for G virtually free is deterministic, Muller and Schupp's result proves that

$$\mathtt{CF} = \mathtt{DCF}\,,$$

i.e. for PDA recognising the word problem, non-determinism doesn't add any power.

One-counter languages and automata

A PDA is a one-counter automaton if the stack alphabet contains just one element. The language it accepts is called a one-counter language.

Theorem (Thomas, Herbst)

$\mathtt{WP}(\mathsf{G})$ is a one-counter language $\iff$ G is virtually cyclic.

We see that the classification of the word problem for regular, context-free and one-counter languages is complete.

Beyond context-free languages we know much less!

Indexed languages and nested stack automata

A language is **indexed** iff accepted by a nested stack automaton iff generated by an indexed grammar.

A nested stack automaton is an FSA with attached nested stack.

- Reading is allowed throughout a stack, writing only at the top.
- A new stack may be created, and entered, anywhere, in any stack. But once a stack is exited to move back to its parent, its contents are lost.

The set $\{a^n b^n c^n : n \in \mathbb{N}\}$ is indexed. So is the set of paths in $\mathbb{Z}^2$ closest to the straight line in $\mathbb{R}^2$ from $\mathbf{0}$ to $\mathbf{x} \in \mathbb{Z}^2$ (Bridson,Gilman,1996).

It is conjectured that there are no groups in $\mathtt{Ind} \setminus \mathtt{CF}$.
Evidence: $\mathtt{WP}(G)$ accepted by a deterministic nested stack automaton with limited erasing + G accessible $\Rightarrow G \in \mathtt{CF}$ (Gilman,Shapiro,preprint).

Complementing context-free languages

The set of deterministic context-free languages is closed under complementation (dcf=co-dcf). But the set of context-free languages is not.

We'll call a group co-context-free (coCF) if the complement of its word problem (its coword problem, coWP) is context-free.

Qn: Which groups are in coCF?

What follows is joint work of myself with Holt, Röver and Thomas (2005).

Elementary results for coCF groups

dcf=co-dcf $\Rightarrow$ DCF $=$ coDCF . Combining this with DCF $=$ CF gives

$$\mathtt{CF} \subseteq \mathtt{coCF}$$

NB: If $G \in \mathtt{coCF} \setminus \mathtt{CF}$ then coWP(G) must be non-deterministically cf.

The following shows that coCF $\setminus$ CF $\neq \emptyset$.

Example:

$$\mathbb{Z}^2 = \langle a, b \mid ab = ba \rangle \in \mathtt{coCF} \setminus \mathtt{CF}\,.$$

$w = w(a, b)$ is non-trivial $\iff$ either (or both) of its projections onto $\langle a \rangle$ or $\langle b \rangle$ is non-trivial.

Hence coWP$(\mathbb{Z}^2)$ is solved by a non-deterministic automaton that first chooses which of a and b to project onto and then decides using a PDA whether or not that projection is non-trivial.

The same argument as for $\mathbb{Z}^2$ shows that

- a direct product of any finite number of coCF groups is coCF .

Using standard arguments we show that

- membership of a group in coCF is independent of the choice of generating set,
- coCF is closed under passage to finitely generated subgroups, finite extension.

In particular we notice that finitely generated subgroups of direct products of free and virtually free groups are in coCF .

We conjecture that coCF is not closed under free products, and that $\mathbb{Z}^2 * \mathbb{Z}$ is outside coCF .

Decision problems in coCF

We can solve the word problem in cubic time in any coCF group (standard result for membership of context-free languages).

By Miller, $\exists$ f.g. $H \subseteq F_2 \times F_2$ with insoluble conjugacy problem and insoluble generalised word problem.

We know that $H \in$ coCF .

Hence we see that the conjugacy problem and generalised word problem are not in general soluble within coCF .

Another operation within coCF

Theorem (Röver,2005) For $G \in$ coCF , $H \in$ CF the standard restricted wreath product $G \wr H$ is in coCF .

Proof: Let $w = v_1u_1v_2u_2 \ldots u_nv_{n+1}$ be input, where $G = \langle X \rangle$, $H = \langle Y \rangle$, u_i, v_i words over X, Y.
$W = G \wr H$ can be expressed as a semidirect product

$$(\prod_{h \in H} G^h) \rtimes H$$

So any element has a unique representation as a product

$$\prod_{j=1}^{m} (g_{h_j}^{h_j}) h, \quad \text{each} \quad g_j \in G,\ h_j, h \in H,$$

We test $w \neq_W 1$ by identifying and checking its components in this rep.

We test non-deterministically.

Either (1) we check if $h \neq_H 1$,
or (2) we randomly select $v \in (Y^{\pm})^*$, and check if $g_v \neq_G 1$.

Within the free group $F(X \cup Y)$, w can be collected into the form

$$u_1^{z_1} \dots u_n^{z_n} v', \quad \text{where} \quad z_1^{-1} =_F v_1, z_2^{-1} =_F v_1 v_2, \dots, z_n^{-1} =_F v_1 \dots v_n$$

and we can identify h as the element of H represented by v', and for each $v \in H$, g_v^v as the product of those conjugates $u_j^{z_j}$ for which $z_j =_H v$.

We use a PDA M that functions variously as the PDA for $\mathtt{WP}(H)$, $\mathtt{coWP}(H)$ and $\mathtt{coWP}(G)$.

M combines the states and transitions functions of 3 PDA.

To test (1) that $h \neq_H 1$,

- M reads w ignoring symbols from X, and passes the symbols it reads from Y to $\mathsf{M}_{\mathtt{coWP}(\mathsf{H})}$.

To test (2) that $g_v \neq 1$, for randomly selected v:

- Initially M passes randomly selected v to $\mathsf{M}_{\mathtt{WP}(\mathsf{H})}$.
- Symbols from Y are input to M acting as $\mathsf{M}_{\mathtt{WP}(\mathsf{H})}$ as they are read.
- Symbols from X are input to M operating as $\mathsf{M}_{\mathtt{coWP}(\mathsf{G})}$ provided that last report from $\mathsf{M}_{\mathtt{WP}(\mathsf{H})}$ was 'identity input', but are otherwise ignored.

The X-word that is processed is the product of those u_i with $z_i =_F v$.

coCF contains groups outside fp

Röver's result shows that coCF contains $\mathbb{Z} \wr \mathbb{Z}$, which is not in fp.

For a while we conjectured that coCF consisted only of finite generated subgroups of groups virtually built using direct products, restricted wreath products.

But we were wrong ....

Some new examples

Theorem (Lehnert, Schweitzer, 2007) The Higman-Thompson groups $G_{n,r}$ are in coCF.

Theorem (Lehnert, Schweitzer, 2007) The Houghton groups H_n are in coCF for $n \geq 2$.

Theorem (Lehnert, 2008 thesis) The group $\mathrm{QAut}(T_2)$ of quasi-automorphisms of the rooted edge-labelled binary tree is in coCF.

These groups are outside the class of groups built from free groups by the operations we have so far shown keep us in coCF. But the set of fg subgroups of $\mathrm{QAut}(T_2)$ **is** closed under all these operations.

Conjecture (Lehnert, Schweitzer) coCF is the set of all fg subgroups of $\mathrm{QAut}(T_2)$.

More on these new examples

A quasi-automorphism of a graph $\Gamma = (V, E)$ is a permutation of V that fails to preserve at most finitely many adjacencies of Γ.

Higman-Thompson groups and Houghton groups embed as fg subgroups of QAut(T_2) (Lehnert, Röver). So the third theorem $\Rightarrow$ the first two.

Where $\mathbb{N}_0$ is the graph $(\mathbb{N} \cup \{0\}, \{\{i, i+1\}\})$, and $*_0^n$ is the star formed from n copies of $\mathbb{N}_0$ by identifying the vertices labelled 0, H_n can be defined as the subgroup of QAut$(*_0^n)$ that induces the identity on the set of n semi-infinite rays of $*_0^n$, has finite index in QAut$(*_0^n)$.

Further interest:

$G_{n,r}$ and H_n were shown in 2006 to be in `coInd` (Holt,Röver), had been conjectured to separate `coInd` from `coCF` .

Proof that $\mathsf{QAut}(T_2) \in \mathtt{coCF}$

The basic components.

- For any cf language L, the language L^o of all cyclic permutations of words in L is also cf (Maslo, 1973).
- for some generating set X of $\mathsf{QAut}(T_2)$ a word w over X is non-trivial iff a cyclic permutation of w moves some vertex in the subtree of depth 4 of T_2.
- $\mathtt{WP}(\mathsf{QAut}(T_2), X) = L^o$, where L is the language of words over X that move some vertex in the subtree of depth at most 4.
- L is context-free.

Which groups cannot be in coCF ?

How is a set proved not-context-free?

Pumping Lemma for CF languages

If L is context-free, then $\exists n$ such that, for any $z \in L$ with $|z| > n$,

$\exists u, v, w, x, y, \quad z = uvwxy, |uvxy| \leq n, |vx| \geq 1,$ and $\forall i \geq 0, uv^i wx^i y \in L.$

Parikh's lemma

If $L' \subset w_1^* w_2^* \dots w_k^*$ is context-free then the set L, defined by

$$L := \{(n_1, \dots n_k) \in \mathbb{N}_0^k : w_1^{n_1} w_2^{n_2} \dots w_k^{n_k} \in L\}, \quad \mathbb{N}_0 = \mathbb{N} \cup \{0\}$$

is a finite union of sets L_i,

where $L_i = c_i + \langle p_{i1}, p_{i2}, \dots p_{ij_i} \rangle_{\mathbb{N}_0} = c_i + \langle P_i \rangle_{\mathbb{N}_0}, \quad c_i, p_{ij} \in \mathbb{N}_0^k.$

Some negative results

Theorem (HRRT, 2005)

A finitely generated nilpotent group has context-free co-word problem iff it is virtually abelian.

Theorem (HRRT,2005)

A Baumslag-Solitar group $G = \langle x, y \mid y^{-1}x^p y = x^q \rangle$ has context-free co-word-problem iff it is virtually abelian (i.e. if $p = \pm q$).

Theorem (HRRT,2005)

A polycyclic group has context-free co-word problem iff it is virtually abelian.

Groups with co-indexed word problem

Theorem (Holt,Röver, 2006)

- For all $r \geq 1, n \geq 2$ the Higman-Thompson groups $G_{n,r}$ are in `coInd`; in fact they are 'stack groups' (use of stack is restricted); the first fact is a consequence of Lehnert and Schweitzer's 2007 result, but the second is not.
- The Grigorchuk group is in `coInd`, as is every fg bounded automata group.
- `coInd` is closed under finite direct products, fg subgroups, finite extensions, restricted wreath product with top group in `CF`.
- a free product of stack groups is a stack group.

Open problems

- Are `CF` and `Ind` equal?
- Is `coCF` equal to the set of fg subgroups of QAut(T_2) (Lehnert, Schweitzer)?
- Is `coCF` closed under free products? We think not, suspect that $\mathbb{Z}^2 * \mathbb{Z}$ is not in `coCF` .
- Are `coCF` and `coInd` equal?
- Is the Grigorchuk group in `coCF` ?
- Is every group in `coInd` actually a stack group?

The word problem in the Chomsky hierarchy, II

Context-sensitive word problem

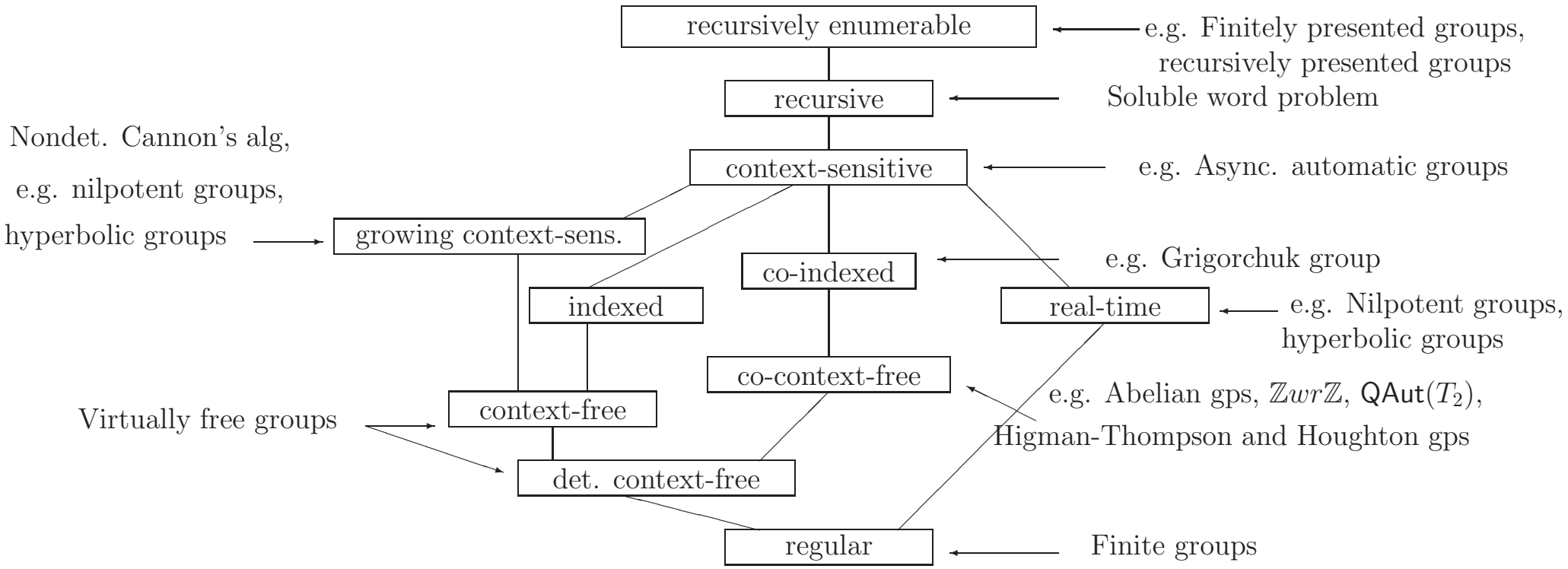
Hopcroft and Ullman claim that

> 'Almost any language one can think of is context-sensitive.'

This is certainly a large class, and for instance it contains linear time.

It contains all the classes we've looked at so far....

Recap: Fitting $\mathtt{WP}(G, X)$ into the formal language hierarchy



Plan for today's lecture

Context-sensitive word problem

Dehn's algorithm

Real-time word problem

Generalising Dehn - Cannon's algorithm

More on real-time

Growing context-sensitive

Linearly bounded TM, context-sensitive languages and grammars

A language is context-sensitive (cs) iff accepted by a non-det TM with linearly bounded memory, iff generated by a context-sensitive grammar.

(Technically speaking a cs language can't contain ϵ, but the language of a linearly bounded TM can.)

A grammar over A is cs if all of its productions have the form $\xi \longrightarrow \eta$, $\xi, \eta \in (A \cup V)^*$, $|\eta| \geq |\xi|$, $\xi \notin A^*$.

Membership of a CS language is PSPACE complete, and not in general soluble in polynomial time.

The complement of a context-sensitive language is also context-sensitive. It is open whether CS =DCS .

Groups with context-sensitive word problem

If G is asynchronously automatic, or even if G has a cs asynchronous combing with length and departure function, then $G \in \mathtt{DCS}$ (Shapiro,1994).

Hence DCS contains Coxeter groups, Baumslag-Solitar groups, $F_n \rtimes F_m$, Artin groups of spherical, large, or right-angled type, $\pi_1(M)$, M a compact 3-manifold.

For G asynchronously combable (Gersten,1992), almost convex (Riley,2002), or fg nilpotent (Holt,Riley),

$$\forall w \in WP(G), \exists \text{chain} \quad w \to w_1 \to \cdots \to w_n \to \epsilon, \quad |w_i| < c|w|, \forall i,$$

(i.e. the groups have linear filling length). So all these groups are in CS.

CS is closed under free and direct products and finite extension (Lakin, thesis 2001).

Dehn's algorithm to solve the word problem

But in particular, Dehn's algorithm to solve the word problem in surface groups runs in linear time, and so is in DCS .

Input word is reduced by string substitution using finitely many length reducing rules

$$u_1 \to v_1, u_2 \to v_2, \ldots, u_n \to v_n$$

over $X^{\pm}$, reduces to ϵ iff trivial. We can find rules like this if G has a Dehn presentation, that is, a presentation

$$\langle X \mid R \rangle.$$

for which any $w \in W(G, X)$ either freely reduces or contains more than half of an element of R as a subword. We get rules $xx^{-1} \to \epsilon$ for free reduction, and rules $u \to v$ for each relator uv^{-1} (or its conjugate or inverse).

Example:

$$G = \langle a, b, c, d \mid ABabCDcd \rangle$$

The standard presentation of a surface group is a Dehn presentation.

We reduce $DCddcBAbaDBAba$ as follows

$$\begin{aligned} DCd\underline{dcBAb}aDBAba &\to DCd\underline{cdA}aDBAba \to \overline{DCdcB}Aba \\ &\to \overline{ABa}Aba \to \epsilon. \end{aligned}$$

Notice that we need to backtrack, i.e. that the third reduction is further left in the word than the first two.

We can program Dehn's algorithm on a fairly straightforward machine with two stacks (one for input, one for output) and a 'read window'.

$N :=$ max length of a lhs of a rule.

Put w on input stack,

Read letters one by one from input to output until last N symbols of output stack contain lhs of a rule $u \longrightarrow v$.

Delete u from output, add v to input, and continue reading from input stack.

If both stacks are empty, w represents the identity.

Total time depends on number of reductions ($\leq n$), and number of symbols read from input stack, so is linear in n.

Some words provoke a lot of backtracking

$|$ $DCdcaDCddcBAAbaDCdcBbaDCcBAbBAba$

$DCdcaDCddcBA\,AbaDC$ $dcBbaDCcBAbBAba$

$AbaDC \to bCD$

$DCdcaDC\,ddcBA$ $bCDdcBbaDCcBAbBAba$

$DCdcaDCd\,dcBAb$ $CDdcBbaDCcBAbBAba$

$dcBAb \to cdA$

$DCd\,caDCd$ $cdACDdcBbaDCcBAbBAba$

$DCdc\,aDCdc$ $dACDdcBbaDCcBAbBAba$

$aDCdc \to Bab$

$DCdc$ $BabdACDdcBbaDCcBAbBAba$

$DCdcB$ $abdACDdcBbaDCcBAbBAba$

$DCdcB \to ABa$

$|$ $ABaabdACDdcBbaDCcBAbBAba$

A sequence of free reductions finishes the reduction.

$\|$ $|$

But in fact backtracking is under control ...

Theorem (Holt,2000) Dehn's algorithm for a hyperbolic group can be run on a Turing machine that operates deterministically in real-time.

Equivalently the word problem for a hyperbolic group is a real-time language.

Intro. to real-time

We define a **real-time Turing machine** ($\mathbb{R}\text{TM}$) to be a deterministic TM with finitely many doubly-infinite tapes, one containing the input as a string of consecutive symbols, where

- input is read once from left to right,
- processing of input is completed in the move in which last symbol is read,
- in a single move, one symbol is read from the input, and there may be one operation on each other tape (a symbol may be read, a new symbol written, the tape head shifted at most one position).

Clearly real-time languages are recognised in linear time, and so are context-sensitive.

Example: $\{a^{n!} : n \in \mathbb{N}\}$ is recognised on a $\mathbb{R}\text{TM}$ with 3 work tapes. (This isn't indexed.)

We recognise $a^{n!}$ recursively.

Suppose we can recognise $a^{k!}$ and then have $a^{(k-1)!}$ on T1, T2, and a^k on T3. Now we want to recognise the remainder of $a^{(k+1)!}$ and update tapes T1-T3 to analogous configurations.

Now $(k+1)! = k! + k! + (k-1)k!$.

We recognise a second $a^{k!}$ by traversing T1 k times (using T3 to count to k). On the first pass we move right on T2, and on further passes write to T2, as we read, so that ultimately this contains $a^{k!}$.

We recognise the remaining $(k-1)k!$ by traversing T3 $k-1$ times (using T3 to count to $k-1$). As we do this we copy T2 to T1 so that ultimately this contains $a^{k!}$.

Non-example: $\{0^{k_1}10^{k_2}1\ldots0^{k_r}12^s0^k_{r-s} : k_i \geq 1, s \leq r\}$ is **not** in $\mathbb{R}\mathtt{T}$, but **is** in dcf.

For any language in $\mathbb{R}\mathtt{T}$ we must have at most c^k equivalence classes for the relation E_k defined by

$$xE_ky \iff (|z| \leq k \Rightarrow (xz \in L \iff yz \in L)),$$

For this example, the number of equivalence classes $> c^k$ for all c.

Unusual feature of $\mathbb{R}\mathtt{T}$: It's closed under Boolean operations.

Hyperbolic groups have real-time word problem

Crucial observation: Dehn's algorithm reduces any input word w to a k-local geodesic, where k is max length of a lhs. In a hyperbolic group, a k-local geodesic is quasigeodesic, so at most K times longer than a geodesic.

The machinery

We run Dehn's algorithm on a $\mathbb{R}\mathrm{TM}$ with 4 work tapes T1–T4, two movable RW heads H1, H2, which delete after reading. It's convenient to allow $M > 1$ moves per work tape between inputs.

- T1 stores reduced output. The suffix of length k is visible.
- T2 holds the right hand sides of rules after reductions.
- T3, T4 are used alternately to store input symbols read while processing.
- H1 reads symbols at a slow, constant rate from the input tape to the right hand end of one of T1,T3,T4.
- H2 reads symbols rapidly from the left hand end of one of T2,T3,T4 to the right hand end of T1, unless all three are empty. When H2 is operational, it reads many (cK) symbols between 2 readings of H1.

The algorithm

Initially H1 is set to write to T1,and T2,T3,T4 are empty.

After a symbol is read to T1:

if T1 contains the lhs of a rule $u \longrightarrow v$ at its right hand end,
u is deleted from T1, v is written onto the left hand end of T2,
H1 is set to write to T3 if that is non-empty, otherwise to T4,
H2 is set to read from T2.

if T2 is empty,
if T3 is non-empty & H1 is set to write to T4, H2 set to read from T3,
else if T4 is non-empty & H1 is set to write to T3, H2 set to read from T4,
else H2 is set to read from input.

Groups with real-time word problem

In the algorithm above, the $\mathbb{R}\text{TM}$ is 'tidy' after accepting an input, so in fact hyperbolic groups have 'tidy real-time' ($\texttt{t}\mathbb{R}\texttt{T}$) word problem.

Proposition (Holt,Rees) If $\texttt{WP}(G)$, $\texttt{WP}(H)$ are in $\mathbb{R}\texttt{T}$ ($\texttt{t}\mathbb{R}\texttt{T}$) then so are

- fin. gen. subgroups of G,
- groups in which G has fin. index,
- quotients of G by finite normal subgroups,
- $G \times H$,

If $\texttt{WP}(G)$, $\texttt{WP}(H)$ are in $\texttt{t}\mathbb{R}\texttt{T}$ then so are $G * H$ and, for finite K, $G *_K H$ and $G*_K$.

Generalising Dehn

Work due to Goodman and Shapiro, which develops an idea due to Cannon, generalises Dehn's algorithm to give a deterministic linear time solution to the word problem for many non-hyperbolic groups, including all virtually nilpotent groups.

To see how it works, we look at why Dehn's algorithm fails for $\mathbb{Z}^2$.

Dehn's algorithm fails for $\mathbb{Z}^2$

We usually solve the word problem in

$$\mathbb{Z}^2 = \langle a, b \mid ba = ab \rangle$$

by reducing an input word using free reduction together with the rules

$$ba \to ab,\ b^{-1}a \to ab^{-1},\ ba^{-1} \to a^{-1}b,\ b^{-1}a^{-1} \to a^{-1}b^{-1}.$$

This isn't Dehn's algorithm because the rules don't reduce length. The algorithm is quadratic, not linear.

Although rules such as $bab^{-1} \to a$ are length reducing, we cannot find finitely many of those to handle all non-reduced words in $\mathbb{Z}^2$.

We need to be able to reduce words of the form

$$b^n a^n b^{-n} a^{-n} \quad \text{for all} \quad n$$

.

Fixing Dehn's algorithm for $\mathbb{Z}^2$

Cannon's generalisation of Dehn's algorithm can solve the word problem for $\mathbb{Z}^2$ in linear time using the injective homomorphism

$$\phi : \mathbb{Z}^2 \to \mathbb{Z}^2, \quad \text{defined by} \quad \phi(a) = a^4, \phi(b) = b^4.$$

We denote $\phi(a)$ by tat^{-1}, $\phi(b)$ by tbt^{-1}, i.e. embed $\mathbb{Z}^2$ in the HNN extension $\mathbb{Z}^2 *_{\mathbb{Z}^2} \phi, t$

Then we can solve the word problem in $\mathbb{Z}^2$, rewriting words of $\mathbb{Z}^2$ within the larger groups, and using the rules like

$$a^4 \to tat^{-1},\ b^4 \to tbt^{-1},\ a^{-4} \to ta^{-1}t^{-1},\ b^{-4} \to tb^{-1}t^{-1}$$

as well as $ba^ib^{-1} \to a^i,\ b^{-1}a^ib \to a^i,\ ab^ia^{-1} \to b^i,\ a^{-1}b^ia \to b^i,$

for $|i| \leq 3$, and free reduction.

Example: Reduction of $b^{12}a^{12}b^{-12}a^{-12}$

- 3 applications of each of the first 4 rules ($a^4 \longrightarrow tat^{-1}$ etc) reduce input word to

$$tb^3t^{-1}ta^3t^{-1}tb^{-3}t^{-1}ta^{-3}t^{-1},$$

- then free reduction reduces that to

$$tb^3a^3b^{-3}a^{-3}t^{-1},$$

- then 3 applications of $ba^3b^{-1} \longrightarrow a^3$ reduces that to

$$ta^3a^{-3}t^{-1},$$

- and free reduction reduces that to ϵ.

Cannon's algorithm:

- Length reducing rules $u \longrightarrow v$ may contain symbols outside $X \cup X^{-1}$.
- A rule might be anchored (apply only at beginning/end of word).
- Where there is a choice of rules to apply, the rule is used which finishes earliest and then is as long as poss. So the algorithm is deterministic and $R(uv) = R(R(u)v)$.

Theorem (Goodman, Shapiro,2008) $\mathtt{WP}(G)$ can be solved deterministically in linear time for G nilpotent, $G = \pi_1(M)$ for M a geom. finite hyperbolic manifold, or G hyperbolic relative to nice subgroups.

Expanding homomorphisms allow construction of appropriate rules in nilpotent groups, and combining this with use of negative curvature allows construction of rules in relatively hyperbolic groups.

More groups with real-time word problem

Theorem (Holt,Rees) If G is hyperbolic, virtually nilpotent or geometrically finite hyperbolic, it has tidy real-time word problem.

Proof:

We use the algorithm we described for hyperbolic groups. It works for these groups because the Dehn algorithm R admits a non-zero function $f : \mathbb{N} \to \mathbb{N}$ bounding below the geodesic length of any word $w \in X^*$ for which $R(w)$ has length n.

Groups not admitting Cannon's algorithm

If G satisfies certain conditions on the growth of a pair of commuting subsets then it cannot admit Cannon's algorithm.

Theorem (Goodman, Shapiro,2008)

Suppose that, for each $n \geq 0$, G contains sets $S_1(n), S_2(n)$ of elements, where

(1) each elt of $S_i(n)$ can be represented by a word of length n,

(2) each element of $S_1(n)$ commutes with each element of $S_2(n)$,

(3) **either** for infinitely many n, each i, $|S_i(n)| \geq \alpha_0 \alpha_1^n$,
or for all but finitely many n $|S_1(n)| \geq \alpha_0 \alpha_1^n$ while $|S_2(n)| \geq \alpha_2 n$,

then G can't admit Cannon's algorithm.

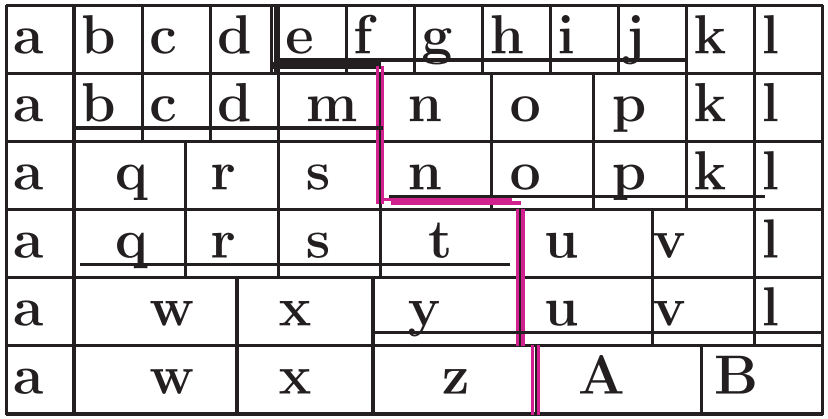
In particular, the theorem covers

- $F_2 \times \mathbb{Z}$,
- braid groups B_n for $n \geq 3$,
- Thompson's group F,
- Baumslag-Solitar groups $B_{p,q}$, $p \neq \pm q$,
- π_1 of various closed 3-manifolds.

Idea of proof

In such a group admitting Cannon's algorithm, there must be a pair of commutators w_0, w_0', where

$$\begin{aligned} w_0 = u_0 v_0 x_0 y_0 = u_0 v_0 u_0^{-1} v_0^{-1} &\to^* w_t = u_t v_t x_t y_t, \\ w_0' = u_0' v_0' x_0' y_0' = u_0' v_0' u_0'^{-1} v_0'^{-1} &\to^* w_t' = u_t' v_t' x_t' y_t', \\ \text{with} \quad v_0 = v_0', y_0 = y_0' v_t x_t y_t = v_t' x_t' y_t', &\qquad \text{but} \quad u_0 \neq u_0' \end{aligned}$$



for which the sequences of rewrites share enough features within sections of each word that the LH of w_0 and the RH of w_0' can be 'spliced', giving a rewrite to ϵ of the non-identity word

$$w_0'' = u_0 v_0^- v_0'^+ x_0' y_0'.$$

The essential features of a sequence of rewrites are picked out of diagrams like the one shown. We can splice two diagrams that contain equivalent 'splitting paths'.

Growing context-sensitive languages and grammars

A context-sensitive grammar is **growing context sensitive** (gcs) if all of its rules are strictly length increasing, and s_0 is never found in a rhs. A language is gcs if generated by a gcs grammar.

Membership of **any** gcs language can be solved in deterministic poly time (Dahlhaus and Warmuth). Hence, certainly

$$G \in \texttt{GCS} \Rightarrow \texttt{WP}(G) \in \text{Det. Poly Time}$$

Any word hyperbolic group is in GCS ; we have seen that we can construct the grammar out of Dehn's algorithm.

Some questions:

- Which groups have gcs word problem?
- Which groups have word problem that is context-sensitive but not gcs?

Some quick answers:

- Groups with Cannon's algorithm are in GCS
- (Kambites,Otto,2007): $F_m \times F_n$,with $m, n > 1$, is in CS $\setminus$ GCS .
  But how about $F_2 \times \mathbb{Z}$? Of interest since, if not in gcs, its word problem would separate gcs from Ł(OW–auxPDA$(poly, log)$).

Equivalence of growing context-sensitive and non-det. Cannon's algorithm

We define a **non-deterministic Cannon's algorithm** for $\mathtt{WP}(G, X)$ be a strictly length reducing rewrite system over an alphabet containing $X^{\pm}$. Rules may be applied in any order,and the algorithm accepts a word if some (but not necessarily every) order of application of rules reduces it to the empty word.

Theorem (Holt, Rees, Shapiro,2008) A group admits a non-deterministic Cannon's algorithm iff it has growing context sensitive word problem.

Theorem (Holt, Rees, Shapiro,2008) A group containing commuting subsets of elements as in Goodman&Shapiro's theorem cannot admit a non-deterministic Cannon's algorithm.

Hence all of

- Braid groups B_n, $n \geq 3$
- $F_2 \times \mathbb{Z}$,
- Baumslag-Solitar groups $B_{p,q}$, $p \neq \pm q$,
- π_1 of various closed 3-manifolds

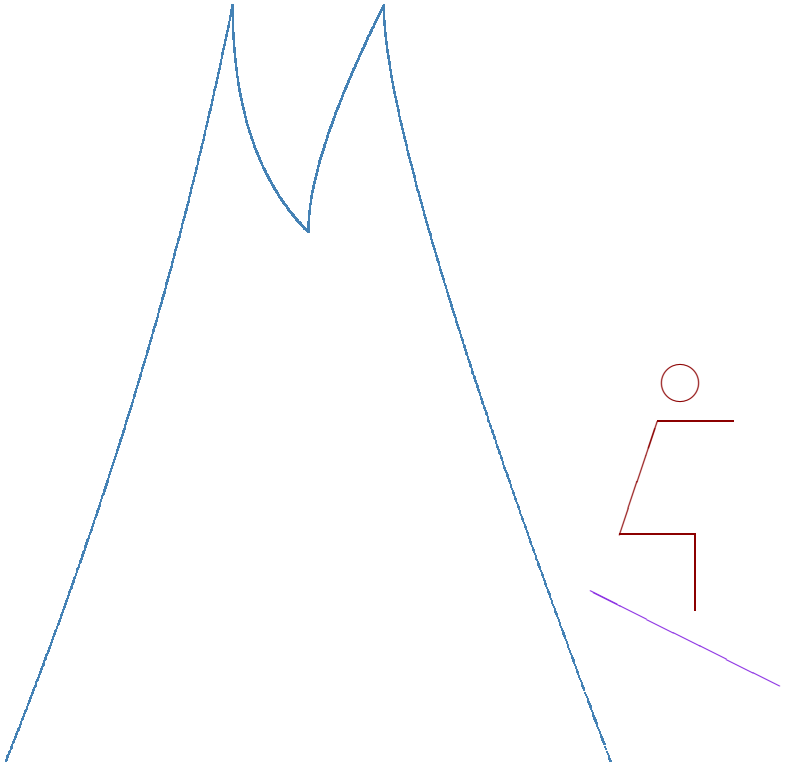
have context-sensitive but not growing context-sensitive word problem.

In essence,

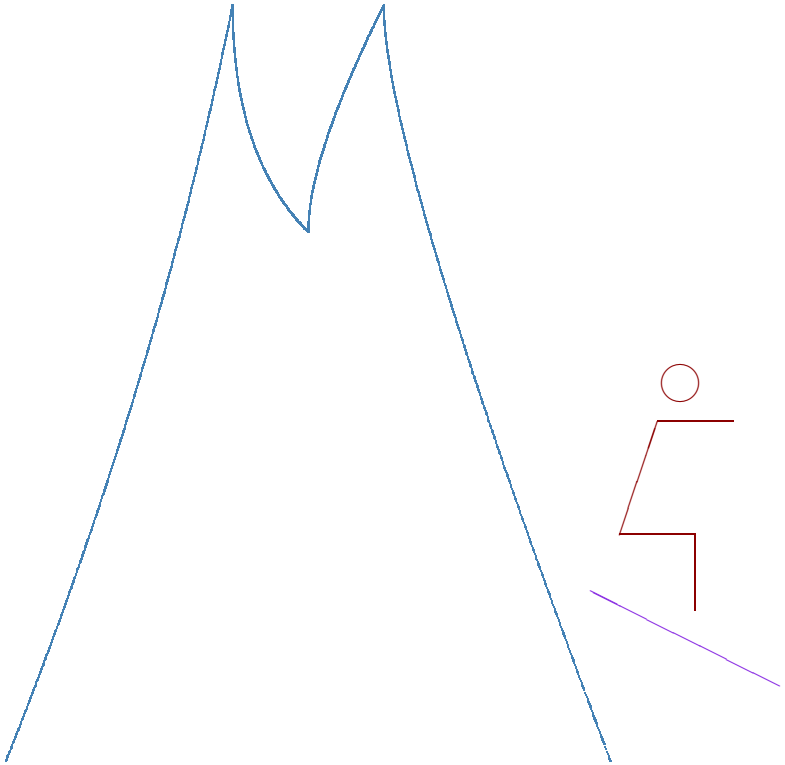
- we construct the appropriate grammar by reversing the direction of rewrites rules to get production rules; we eradicate any anchored production rules by duplicating symbols,
- the proof that non-det. Cannon's algorithm cannot apply given certain conditions follows the route of Goodman&Shapiro, with some modifications to technical definitions and counting arguments to deal with effects of non-determinism.

More open problems

- What is the relationship between $\mathtt{GCS}$ and $\mathbb{RT}$?
- Where do hyperbolic groups really belong?
- Are all the examples in $\mathtt{CS}$ actually in $\mathtt{DCS}$?



E R C I !



E R C I !

Au revoir