Rewriting Techniques in Finitely Presented Groups and Monoids
Lectures given at New Mexico State University, Las Cruces, January 3-7, 1997

Derek F. Holt

Abstract

This document contains an amplified version of the five talks given by the author at the 22nd Holiday Mathematics Symposium at the New Mexico State University, January 3-7 1997, on the topic "Rewriting Techniques and Non-commutative Gröbner Bases".

1 Introduction

These notes are predominantly about finitely presented groups, and it is assumed that the reader is familiar with basic definitions and results about this class of groups. The book [Lyndon, Shupp, 77] is perhaps the best general reference.

The most concise description of a particular group, finite or infinite, is often by means of a finite presentation (assuming there is one at all). In other words, $G = \langle X|R\rangle$, where $X = \{x_1, \ldots, x_m\}$ is a finite set of generators of G, and $R = \{r_1, \ldots r_n\}$ is a finite set of words in the x_i and their inverses, which form a set of *defining relators* for G. Indeed, many important classes of groups coming from other branches of mathematics, such as knot groups in topology, arise naturally as finite presentations.

The disadvantage of this mode of representation of a group is that it is difficult, in general, to say anything much about the size or structure of a group that is known only via a finite presentation. Indeed, the most basic problem (the word problem), of deciding whether two words in the group generators are equal in the group, is known to be undecidable in general. (This non-trivial result was proved in the mid 1950's independently by Novikov and Boone.) Other problems, such as deciding finiteness (or even triviality) of the group, or deciding conjugacy of elements are also unsolvable.

Fortunately, in many classes of groups that do arise naturally as finite presentations (including knot groups), there are theorems that tell us that the word problem at least, and sometimes some of the other problems, are solvable. During the past ten years or so, with the increasing availability of fast computers, and powerful, user-friendly computer algebra systems, there has been increasing interest in implementing algorithms to solve these problems efficiently in practice. The theorems mentioned above do not always tell us how to do this, and so new methods may need to be devised.

The older, classical algorithms for computing in finitely presented groups, include coset enumeration and related techniques, and various quotient algorithms, the best known of which is the abelian quotient algorithm for determining the structure of $G/[G,G]$, which is often taught to undergraduates. These are limited in that ultimately they only provide information about finite quotients of G, or about rather restricted classes of quotients, such as nilpotent or polycyclic quotients.

On the other hand, the newer methods based on string rewriting, which constitute the major theme of this article, are capable of making effective computation possible in the whole group and, when successful, they enable the word problem to be solved very quickly indeed. In the early research in this area, the aim was to find a finite *complete* rewriting system, which is one that enables any word in the group generators to be reduced to a normal form by repeated application of a finite set of rewrite-rules. This often proved frustrating, since although many such examples were found by diligent searching, success was largely a matter of chance, and of choosing the correct generating set and ordering of words. Apart from in a few isolated well-behaved classes of groups, such as polycyclic groups and two-dimensional surface groups, there was no coherent theory as to when one might expect to find such a finite system and when not.

The development of the theory of *automatic groups* in the mid-1980's improved this situation to some extent. In this class of groups, it is possible to solve the word problem using finite state automata. The construction of these automata uses the techniques of string rewriting systems, but does not rely on finding a finite complete system. Furthermore, there are many important classes of groups, including Coxeter groups, word-hyperbolic groups and most knot groups, that have been proved to be automatic, and so the algorithms are guaranteed (in theory) to succeed, although of course they may prove to be very expensive in terms of computing resources.

In the first lecture, we give a brief but self-contained account of the theory of string rewriting systems. In the second, we consider applications to finitely presented groups G, and introduce the interesting example of rewriting in polycyclic groups (which is traditionally known as the collection process). In the third lecture, we discuss the problem of finding polycyclic quotients of general finitely presented groups. A recent algorithm for this by Eddie Lo depends heavily on the use of Gröbner bases over the (non-commutative) group algebra of a polycyclic group. We then go on to describe how string rewriting may be used to prove that the original group G is polycyclic.

In the fourth lecture, we move on to the study of finite state automata and automatic groups mentioned above. Then, in the final lecture, we discuss the generalisation of string rewriting techniques and automatic group theory to the enumeration of cosets of a subgroup of G.

2 LECTURE 1: String Rewriting and the Knuth-Bendix Algorithm

The most obvious way to attempt to solve the word problem in a finitely presented group is to use the group relations to replace one string in the given word by another, in an attempt to reduce the word to some kind of normal form. In this section, we shall be studying such rewriting processes. There is a fairly large body of literature on this topic, which has applications to all branches of algebra. There is a good historical account, with a large bibliography in [Buchberger, 87]. One of the earliest papers describing the use of critical pairs and the completion process in a very general setting is that of Knuth and Bendix (see [Knuth, Bendix, 70]), after whom this process is usually named. Applications to group theory are described in the two papers of Gilman ([Gilman, 79] and [Gilman, 84]). In addition, there is now an extensive treatment of this topic in the book of Sims ([Sims, 94]).

Let A be a finite set, known as the alphabet, and let A^* denote the set of all strings (or words) in A, including the empty string, which will be denoted by ε. Then A^* forms a monoid under the operation of concatenation of strings.

A rewriting system on A^* is a set S of ordered pairs (w_1, w_2), with $w_1, w_2 \in A^*$. The elements of S are called rewrite-rules, and w_1 and w_2 called, respectively, the left and right hand sides of the rule. The idea is that we can replace occurrences of w_1 in a string by w_2. We shall assume throughout that no two distinct rules have the same left hand sides.

For $u, v \in A^*$, we write $u \to_S v$ if there exist strings $x, y, w_1, w_2 \in A^*$ such that $u = xw_1y, v = xw_2y$ and $(w_1, w_2) \in S$; in other words, if v is obtained from u by making a single substitution with a rewrite-rule. We shall omit the subscript S when there is no danger of ambiguity.

A string $u \in A^*$ is called *irreducible* under S if there is no string $v \in A^*$ with $u \to v$.

We denote the reflexive, transitive closure of $\to$ by $\to^*$. Hence $u \to^* v$ if and only if, for some $n \geq 0$, there exist $u = u_0, u_1, \ldots, u_n = v \in A^*$ with $u_i \to u_{i+1}$ for $0 \leq i < n$. The symmetric closure of $\to^*$ is denoted by $\leftrightarrow^*$. Thus $u \leftrightarrow^* v$ if and only if, for some $n \geq 0$, there exist $u = u_0, u_1, \ldots, u_n = v$ with

$$\begin{array}{ccccccccccc} & & u_1 & & & & u_3 & & & & u_{n-1} & & \\ & {}_*\swarrow & & \searrow_* & & {}_*\swarrow & & \searrow_* \;\cdots\; {}_*\swarrow & & & & \searrow_* & \\ u_0 & & & & u_2 & & & & & & & & u_n \end{array} \qquad (\dagger)$$

The system S is called *Noetherian* (or terminating) if there is no infinite chain of strings $u_i (i > 0)$ with $u_i \to u_{i+1}$ for all i. This implies that for any $u \in A^*$ there exists an irreducible $v \in A^*$ with $u \to^* v$.

We call S *confluent* if, whenever $u, v_1, v_2 \in A^*$ with $u \to^* v_1$ and $u \to^* v_2$, there exists $w \in A^*$ with $v_1 \to^* w$ and $v_2 \to^* w$, and we call S *complete* if it is Noetherian and confluent.

It is called *locally confluent* if whenever $u, v_1, v_2 \in A^*$ with $u \to v_1$ and $u \to v_2$, there exists $w \in A^*$ with $v_1 \to^* w$ and $v_2 \to^* w$.

The following sequence of lemmas demonstrates that completeness is a highly desirable property for rewriting-systems to possess. For $u \in A^*$, we define the set $\mathrm{desc}(u)$ of descendents of u to be $\{w \in A^* - \{u\} | u \rightarrow^* w\}$. So u is irreducible if and only if $\mathrm{desc}(u)$ is empty.

Lemma 2.1 *Suppose that S is Noetherian. Then* $\mathrm{desc}(u)$ *is finite for all* $u \in A^*$.

PROOF: Suppose that $\mathrm{desc}(u)$ is infinite. Since we are assuming that no two rules in S have the same left hand side, there are only finitely many w with $u \rightarrow w$. Thus there exists $u_2 \in A^*$ with $u \rightarrow u_2$ and $\mathrm{desc}(u_2)$ infinite. Similarly, there exists u_3 with $u_2 \rightarrow u_3$ and $\mathrm{desc}(u_3)$ infinite, and we get an infinite chain $\{u_i\}$ with $u_i \rightarrow u_{i+1}$, contradicting the assumption that S is Noetherian. □

Lemma 2.2 *Suppose that S is Noetherian and locally confluent. Then S is complete.*

PROOF: We shall prove that, under the given hypotheses, for all $u \in A^*$ there is a unique irreducible w with $u \rightarrow^* w$. The result clearly follows from this. The proof is by induction on $|\mathrm{desc}(u)|$, the result being clear for $|\mathrm{desc}(u)| = 0$. Suppose that $u = u_0 \rightarrow u_1 \rightarrow \ldots \rightarrow u_m$ and $u = u_0 \rightarrow u'_1 \rightarrow \ldots \rightarrow u'_n$, with u_m and u'_n irreducible. By local confluence, there exists v with $u_1 \rightarrow^* v$ and $u'_1 \rightarrow^* v$. Let $v \rightarrow^* w$ with w irreducible. Since $|\mathrm{desc}(u_1)| < |\mathrm{desc}(u)|$ and $|\mathrm{desc}(u'_1)| < |\mathrm{desc}(u)|$, it follows by inductive hypothesis that $u_m = w$ and $u'_n = w$, so $u_m = u'_n$, which completes the proof. □

Lemma 2.3 *If S is Noetherian and locally confluent, then each equivalence class under $\leftrightarrow^*$ contains a unique irreducible element.*

PROOF: Let u and v be irreducible strings with $u \leftrightarrow^* v$. Then there exist $u = u_0, u_1, \ldots, u_n = v \in A^*$ satisfying the equation (†) above. We have to prove that $u = v$. This is clear if $n = 0$, so we use induction on n. By Lemma 2.2 S is confluent, so there exists $w \in A^*$ with $u \rightarrow^* w$ and $u_2 \rightarrow^* w$. Since u is irreducible, we must have $u = w$, and so we can replace u_2 by u in the diagram, and the result follows by induction. □

The systems that we shall be considering will all be Noetherian, but they will not always be confluent. The following lemma provides a method of checking confluence (and hence completeness) constructively, at least when S is finite.

Lemma 2.4 *The system S is locally confluent if and only if the following conditions are satisfied for all pairs of rules $(u_1, t_1), (u_2, t_2) \in S$.*

(i) *If $u_1 = rs$ and $u_2 = st$ with $r, s, t \in A^*$ and $s \neq \varepsilon$, then there exists $w \in A^*$ with $t_1 t \rightarrow^* w$ and $r t_2 \rightarrow^* w$.*

(ii) *If $u_1 = rst$ and $u_2 = s$ with $r, s, t \in A^*$ and $s \neq \varepsilon$, then there exists $w \in A^*$ with $t_1 \rightarrow^* w$ and $r t_2 t \rightarrow^* w$.*

PROOF: First assume that S is locally confluent. Then in both cases, the existence of w follows by applying the definition of local confluence to the string rst. Conversely, assume that the conditions (i) and (ii) are valid, and let u be a string with $u \to v_1$ and $u \to v_2$. Then u must have two substrings u_1 and u_2 such that there exist rules $(u_1, t_1), (u_2, t_2) \in S$, which are used in the reductions to v_1 and v_2. If u_1 and u_2 do not overlap in u, then we have $u = ru_1su_2t$ for some strings r, s, t and $v_1 = rt_1su_2t, v_2 = ru_1st_2t$. Thus $v_1 \to w$ and $v_2 \to w$, with $w = rt_1st_2t$. Otherwise, u_1 and u_2 overlap in S, and (after interchanging the rules if necessary) we must have one of the two situations postulated in conditions (i) and (ii) of the lemma. It then follows by assumption that there exists $w \in A^*$ with $v_1 \to^* w$ and $v_2 \to^* w$, so S is locally confluent. $\square$.

A pair of rules satisfying either of the two conditions of the lemma is called a *critical pair*. If one of these conditions fail (so S is not confluent), we end up with two distinct irreducible strings w_1 and w_2 which are irreducible and equivalent under $\leftrightarrow^*$. We can resolve this instance of incompleteness by adding either (w_1, w_2) or (w_2, w_1) to S as a new rule. We can then continue with the check for local confluence.

This procedure of examining critical pairs and adding new rules to S if necessary is known as the *Knuth-Bendix completion process*. It may happen that this process eventually terminates with a finite complete set S, which is what we would like. In many cases, however, the process does not complete, but it does, in principal, generate an infinite complete set of rules. In a few examples, as we shall see, such an infinite set can have a very transparent structure, which can make it almost as useful as a finite complete set.

In general, it can be difficult to decide which of (w_1, w_2) or (w_2, w_1) to add to S in order to resolve a critical pair. In the applications that we shall be considering, we do this by imposing a well-ordering $\leq$ on A^* that must have the property that $u \leq v$ implies $uw \leq vw$ and $wu \leq wv$ for all $w \in A^*$. Then we always make the larger of the pair the left hand side of the equation.

The assumption on the ordering ensures that $u \to^* v$ implies $u \geq v$ and, since it is a well-ordering, S will necessarily be Noetherian. Thus, if S is complete, then the irreducible elements correspond to the least representatives in the $\leftrightarrow^*$ equivalence classes.

One of the most common examples of such an ordering is a *shortlex* ordering. We first choose a total ordering of A (which we are assuming to be finite), and then, for $u, v \in A^*$, we define $u < v$ if either u is shorter than v, or if u and v have the same length, and u comes before v in the lexicographic (dictionary) ordering relative to the ordering of A.

For example, if $A = \{a, b, c\}$ and (just to be awkward) we decide to order A by $b < c < a$, then the first few strings in the shortlex order will be

$$\varepsilon, b, c, a, bb, bc, ba, cb, cc, ca, ab, ac, aa, bbb, bbc, bba, \ldots$$

¿From now on, we shall assume that the rules are ordered according to some such well-ordering, although not necessarily a shortlex ordering. (A completely different well-ordering, appropriate for polycyclic groups, will be considered in Section 4.)

A rule (w_1, w_2) is called *minimal*, if w_2 and all proper substrings of w_1 are irreducible in S. If all rules are minimal, then S is called *reduced*.

If a rule is not minimal, then we can use the other rules in S to simplify it, without changing the relation $\rightarrow^*$. The rule may either turn out to be redundant, or we may replace it by a minimal rule (w_1', w_2') in which $w_1' < w_1$ or $w_2' < w_2$. (Of course, (w_1', w_2') must then be regarded as a new rule of S.)

The new rules that we add to S during the Knuth-Bendix completion process will always be minimal, but they will often destroy the minimality of the existing rules. It is usually efficient to reduce S by replacing all rules by minimal rules at regular intervals.

The minimal rules in the completion of S consist of those for which w_2 and all proper substrings of w_1 are the least elements in their $\leftrightarrow^*$ classes. They are therefore uniquely determined by S. We will call such rules *essential*.

The set of essential rules is itself reduced and complete. There may be infinitely many of them but, since the collection of all rules produced by the completion process is complete, any particular essential rule will eventually be found. Thus the algorithm will terminate if and only if there are finitely many essential rules.

3 LECTURE 2: Applications to Group Theory

We turn now to the applications to group theory. Much of what follows applies equally well to monoids, where we do not necessarily have inverses for each generator, and there are interesting and important applications of the Knuth-Bendix algorithm to monoids, but we shall concentrate on the case of groups in this article.

Let $A/\leftrightarrow^*$ denote the set of $\leftrightarrow^*$-equivalence classes of A. Then the monoid operation of concatenation on A^* induces a well-defined operation on $A/\leftrightarrow^*$. The resulting quotient structure, which we shall denote by A_S, is precisely the monoid defined by the presentation

$$\langle\, A \mid \{w_1 = w_2 | (w_1, w_2) \in S\}\,\rangle$$

We shall assume from now on that, for each $a \in A$, there is a string $\bar{a}$ in A^*, for which $a\bar{a} \rightarrow^* \varepsilon$ and $\bar{a}a \rightarrow^* \varepsilon$ in S. Then A_S is a group.

Conversely, given any finitely presented group $G = \langle X|R \rangle$ (where R is a set of relations), we can represent G as a quotient structure for a rewriting system as follows.

Let X^{-1} be a set disjoint from X and in one-one correspondence with X, where for $x \in X$ we denote the corresponding element of X^{-1} by $\bar{x}$. Let $A = X \cup X^{-1}$, and let

$$S = \{(x\bar{x}, \varepsilon)|x \in X\} \ \cup\ \{(\bar{x}x, \varepsilon)|x \in X\} \ \cup\ \{(w_1, w_2)|w_1 = w_2 \in R\}.$$

It is straightforward to show that A_S is isomorphic to G, by observing that A_S has the universal property of the group presentation $\langle X|R \rangle$.

In particular, if R is empty, then A_S is a free group on X, and it follows immediately from Lemma 2.4 that S is confluent in this case, so the reduced words are precisely those which contain no adjacent inverse generators. This is in agreement with the usual definition of the free group on X.

By default, we use a shortlex ordering on A^* to decide on the left and right hand sides of the rules but, as we shall see in the next section, in particular situations other orderings may be more effective.

If, by using the Knuth-Bendix completion procedure, we can extend S to a finite complete set, then we can solve the word problem effectively in G by reducing a given word in A^* to its unique irreducible representative using substitutions with the rewrite-rules in S.

It is not difficult to show that, if G is finite, then the procedure will always complete with a finite set S. (This is because, in the completion of S, there are only finitely many irreducible words, and hence only finitely many essential rules (w_1, w_2). Once these have been generated, S will be complete.)

As an easy example, let $G = \langle\, x, y \mid x^3 = 1, y^2 = 1, (xy)^2 = 1 \,\rangle$ be the dihedral group of order 6. Then $A = \{x, \bar{x}, y, \bar{y}\}$, and (using the order $x < \bar{x} < y < \bar{y}$ on A) we can start with

$$S = \{(x\bar{x}, \varepsilon), (\bar{x}x, \varepsilon), (y\bar{y}, \varepsilon), (\bar{y}y, \varepsilon), (xx, \bar{x}), (\bar{y}, y), (y\bar{x}, xy)\}.$$

(When each generator has an inverse generator and we are using a shortlex ordering, then we can always rewrite our rules so that the length of the left hand side is greater than the right hand side by at most 2, and it is generally efficient to do this.)

One critical pair, for example, is $(\bar{x}x, \varepsilon), (xx, \bar{x})$ with the overlap x. This leads to the word $\bar{x}xx$ reducing in two different ways to x and $\bar{x}\bar{x}$, and to resolve this we add the rule $(\bar{x}\bar{x}, x)$ to S.

It is extremely tedious to examine all such critical pairs by hand, but easy for a computer, and we end up with the complete set of rules

$$S = \{(x\bar{x}, \varepsilon), (\bar{x}x, \varepsilon), (y\bar{y}, \varepsilon), (\bar{y}y, \varepsilon), (xx, \bar{x}), (\bar{x}\bar{x}, x), (\bar{y}, y), (yy, \varepsilon), (y\bar{x}, xy), (yx, \bar{x}y)\}.$$

We now observe that the rules $(y\bar{y}, \varepsilon)$ and $(\bar{y}y, \varepsilon)$ are not minimal, since the word $\bar{y}$ is reducible. In fact they are both redundant, and if we remove them, we are left with the set of all essential rules.

Most of the time, if G is an infinite group, then the Knuth-Bendix procedure will not terminate, but there are some examples where it does. There does not seem to be any satisfactory theory which will predict when this will happen; in many examples, termination depends on the correct choice of ordering of A^*, and even of A when a shortlex ordering is used.

For example, let $G = \langle\, x, y \mid yx = xy \,\rangle$ be the free abelian group on two generators. If we use the ordering $x < \bar{x} < y < \bar{y}$ of A, then we end up with the finite complete set

$$S = \{(x\bar{x}, \varepsilon), (\bar{x}x, \varepsilon), (y\bar{y}, \varepsilon), (\bar{y}y, \varepsilon), (yx, xy), (y\bar{x}, \bar{x}y), (\bar{y}x, x\bar{y}), (\bar{y}\bar{x}, \bar{x}\bar{y})\}.$$

If we use the ordering $x < y < \bar{x} < \bar{y}$, however, then we generate the infinite complete set

$$S = \{(x\bar{x},\varepsilon),(\bar{x}x,\varepsilon),(y\bar{y},\varepsilon),(\bar{y}y,\varepsilon),(yx,xy),(\bar{x}y,y\bar{x}),(\bar{y}x,x\bar{y}),(\bar{y}\bar{x},\bar{x}\bar{y}),\\ (xy^n\bar{x},y^n),(y\bar{x}^n\bar{y},\bar{x}^n) \mid \text{for all } n>0\}.$$

This final example is one in which the infinite complete set is regular (in the sense of regular languages), and it can still be used effectively to reduce words to normal form.

The known infinite examples for which finite complete rewriting systems have been found using some choice of generators and ordering include polycyclic groups, the two-dimensional surface groups, the von Dyck (triangle) groups $G = \langle\, x,y \mid x^p = y^q = (xy)^r = 1 \,\rangle$ (by I. Wanless), many Coxeter groups [Hermiller, 94] and a number of isolated examples, such as $\langle\, x,y \mid xy = yx^2 \,\rangle$. Some of these systems are catalogued in [Le Chenadec, 86].

Finally, it is worth noting that there are certain similarities between the Knuth-Bendix procedure and Todd-Coxeter coset enumeration over the trivial subgroup. Both are in some sense looking systematically for consequences of the defining relations of the group. Of course the Todd-Coxeter algorithm can only complete for finite groups, but it appears to be more efficient than Knuth-Bendix for most straightforward examples.

There are exceptions, however. Knuth-Bendix works very well for certain Coxeter groups, such as the symmetric groups S_n, which have a large order and relatively small complete rewriting systems (about $(n-1)^2$ equations).

For certain pathological examples, such as the presentation

$$\begin{aligned} G = \langle\, r,s,t \;\mid\; & ssr^{-1}s^{-1}rs^{-1}tst^{-1}t^{-1}r^{-1}srs^{-1}s^{-1}tts^{-1}t^{-1}stts^{-1}t^{-1}s \\ &= tts^{-1}t^{-1}st^{-1}rtr^{-1}r^{-1}s^{-1}tst^{-1}t^{-1}rrt^{-1}r^{-1}trrt^{-1}r^{-1}t \\ &= rrt^{-1}r^{-1}tr^{-1}srs^{-1}s^{-1}t^{-1}rtr^{-1}r^{-1}ssr^{-1}s^{-1}rssr^{-1}s^{-1}r \;=\; 1 \,\rangle \end{aligned}$$

of the trivial group, due to B.H. Neumann, Knuth-Bendix succeeds eventually, whereas no implementation of Todd-Coxeter has got anywhere with this to date.

It is also possible to find a string-rewriting analogue of coset enumeration over a nontrivial subgroup H of G. This topic is discussed in Section 8.

Efficient computer implementations of the Knuth-Bendix algorithm for groups and monoids are available in the Rutgers Knuth-Bendix Package (rkbp) written by Charles Sims and available from him, and the author's own KBMAG package. Both offer a wide choice of controlling parameters and word orderings.

4 LECTURE 2 (cont'd): Polycyclic groups

In this section, we shall give a brief exposition of the theory of polycyclic groups in the context of string rewriting and Knuth-Bendix completion.

A group G is said to be polycyclic if it has a descending chain of subgroups $G = G_1 \supset G_2 \supset \ldots \supset G_{n+1} = 1$ in which each G_{i+1} is a normal subgroup of G_i, and the quotient group G_i/G_{i+1} is cyclic. (This is equivalent to solubility of G in the finite case.)

Any such group has a generating set $X = \{x_1, \ldots, x_n\}$, where $x_i \in G_i$ and $x_i G_{i+1}$ generates G_i/G_{i+1}.

Let I be the subset of $\{1, 2, \ldots, n\}$ consisting of those i for which G_i/G_{i+1} is finite, and let $m_i = |G_i/G_{i+1}|$ for $i \in I$.

As in Section 2, we define our alphabet to be the set $A = \{x_i, x_i^{-1} | 1 \leq i \leq n\}$, which generates G as a monoid. (Note that we are using x_i^{-1} rather than $\overline{x_i}$ in this section to denote the monoid generator mapping on to the inverse of x_i.)

We may, if we wish, use the smaller set $A' = \{x_i | 1 \leq i \leq n\} \cup \{x_i^{-1} | i \notin I\}$. It is easy to show (by induction on $n - i$) that the elements x_i^{-1} for $i \in I$ can be expressed, in G, as products of the generators in A', and so A' generates G as a monoid. In particular, if G is finite, then we may dispose of the inverse generators altogether.

In fact, it is easy to see that every element $g \in G$ has a unique representative word in $(A')^*$ of the form $x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$, where each $\alpha_i \in \mathbb{Z}$ and $0 \leq \alpha_i < m_i$ if $i \in I$. This word will be called the normal form for g.

We should like to find an ordering of the words in A^* for which the normal form for each group element g is the least word representing g. Since the normal form may be far from being the shortest word for g, the orderings based on length are not suitable. In fact, the appropriate ordering is an instance of a *wreath-product ordering*, as defined on page 48 of [Sims, 94].

In general, a wreath product ordering on B^*, for an alphabet $B = \{z_1, \ldots z_m\}$ is defined as follows.

First we assign levels to each of the letters z_i, which should be positive integers. The idea is that a single occurrence of a generator at a higher level renders a word larger than any number of generators at lower levels. For words involving generators of the same level only, we use some other standard ordering, which may, for example, be a shortlex ordering.

Suppose, inductively, that for some positive integer r, we know how to order words involving only generators of level less than r. Then distinct words v, w involving generators of level r are ordered as follows.

First remove all generators of level less than r from v and w, and compare the two resulting words v', w' in the level r generators. If they are distinct and, say, $v' > w'$, then we define $v > w$.

On the other hand, if $v' = w' = z_{i_1} z_{i_2} \ldots z_{i_l}$, say, then we have $v = t_1 z_{i_1} t_2 z_{i_2} \ldots t_l z_{i_l} t_{l+1}$ and $w = u_1 z_{i_1} u_2 z_{i_2} \ldots u_l z_{i_l} u_{l+1}$, where, for $1 \leq k \leq l+1$, t_k and u_k are words involving only generators of level less than r. We define $w < x$ if $t_1 < u_1$ or, for some $k \leq l$, $t_i = u_i$ for $1 \leq i \leq k$, but $t_{k+1} < u_{k+1}$.

It is routine to check that this is a well-ordering which satisfies $u \leq v$ implies $uw \leq vw$ and $wu \leq wv$ for all $u, v, w \in A^*$, which are the required conditions specified in Section 2.

In the application to polycyclic groups, we give each of the monoid generators distinct levels, with x_1^{-1} having the highest level, followed by x_1, then x_2^{-1} then x_2, and so on, with x_n having the lowest level. (These orderings are also referred to as *recursive path orderings* in the literature.)

Now G has a presentation (as a monoid) in which the relations have the following form

$$
\begin{array}{rcll}
x_i x_i^{-1} = & x_i^{-1} x_i = \varepsilon & (i \notin I), \\
x_i^{-1} = & w_i' & (i \in I), \\
x_i^{m_i} = & w_i & (i \in I), \\
x_j x_i = & x_i w_{ij} & (1 \leq i < j \leq n), \\
x_j x_i^{-1} = & x_i^{-1} w_{ij}' & (1 \leq i < j \leq n, i \notin I), \\
x_j^{-1} x_i = & x_i \overline{w_{ij}} & (1 \leq i < j \leq n), \\
x_j^{-1} x_i^{-1} = & x_i^{-1} \overline{w_{ij}}' & (1 \leq i < j \leq n, i \notin I)
\end{array}
$$

where $w_i, w_{ij}, \overline{w_{ij}}, w_{ij}'$ and $\overline{w_{ij}}'$ are normal form words in $(A' \cap \{x_k, x_k^{-1} | k > i\})^*$, and w_i' is in normal form and in $(A' \cap \{x_i, x_k, x_k^{-1} | k > i\})^*$. (Note that $\overline{w_{ij}}$ is equal to the inverse of w_{ij}.)

We call such a presentation a *polycyclic presentation* of G.(Such presentations are also called *AG-presentations* and, for nilpotent groups, *power-commutator presentations* in the literature.)

Notice that in each of these equations, the left hand side is greater than the right hand side in the wreath product ordering just defined, and so we can define a corresponding rewriting system S in which the left and right hand sides of the rules are given by the left and right hand sides of the relations in the presentation.

It can be seen that the words in normal form are precisely those that do not have the left hand side of any of these equations as a subword; that is they are the irreducible words with respect to the ordering. Since distinct irreducible words represent distinct elements of G, they cannot be equivalent in $\leftrightarrow^*$, and so S is complete, and it is clear that all of its rules are minimal and hence essential.

Conversely, suppose that we are given an arbitrary monoid presentation of the above form, over the alphabet A. Then we can still define the rewriting system S as in the preceding paragraph.

We should start by checking that the words $x_i w_i'$ and $w_i' x_i$ reduce to the identity in S for all $i \in I$. If this is the case, then the presentation defines a group G, in which the inverse generators are what they seem.

It is not difficult to see that, if we define $G_i = \langle x_j | j \geq i \rangle$ (as a group), for $1 \leq i \leq n+1$, then G_{i+1} is still normal in G_i with G_i/G_{i+1} cyclic for all i, so G is certainly polycyclic. Furthermore, the rewriting system S is still Noetherian, and the set of reduced words is the same as before.

However, the order m_i' of G_i/G_{i+1} may be less than the number m_i given in the presentation when $i \in I$, or m_i' may be finite when $i \notin I$. In fact, this occurs if and only if some group element has more than one irreducible representative word; that is, if and only if S is not confluent.

Traditionally, a presentation for which G_i/G_{i+1} has order m_i when $i \in I$ and is infinite when $i \notin I$ is called *consistent*, and so the concepts of confluence, completeness and consistency are equivalent in this setting.

Lemma 2.4 provides us with a method for testing for confluence. This method was first used and justified by Wamsley in the nilpotent case in [Wamsley, 70] (although not in the context of critical pair completion). The equations arising from the critical pairs that have to be checked include the following:

$$\begin{array}{rcll}
x_i^{-1}w'_{ij}x_i & = & x_j & (j > i, i \notin I), \\
x_i & = & x_j^{-1}x_iw_{ij} & ((j > i, j \notin I), \\
x_i^{-1} & = & x_j^{-1}x_i^{-1}w'_{ij} & ((j > i, j \notin I, i \notin I), \\
x_iw_i & = & w_ix_i & (i \in I), \\
x_j^{m_i-1}x_iw_{ij} & = & w_jx_i & (j \in I, j > i), \\
x_iw_{ij}x_i^{m_i-1} & = & x_jw_i & (i \in I, j > i), \\
x_jw_{jk}x_i & = & x_kx_iw_{ij} & (k > j > i)
\end{array}$$

(This is only a proper subset of all such equations, but it is proved in Proposition 8.3, Chapter 9 of [Sims, 94] that it suffices to check the listed equations only in order to check consistency.)

For large n, the set of equations in the final line of the list is the largest, and contains up to $n(n-1)(n-2)/6$ equations. In the nilpotent case, many more of these checks can be shown to be redundant by assigning weights to the generators according to how far down the lower central series of G they lie. In the case of finite p-groups, it is proved in [Vaughan-Lee, 70] that it is sufficient to test approximately $n^2d/2$ equations, where d is the minimal number of generators of the group.

5 LECTURE 3: Quotient Algorithms

In this section, we look briefly at algorithms for computing various types of quotients of finitely presented groups, without going into technical details. In particular, a recent polycyclic quotient algorithm developed by Eddie Lo [Lo, 96] relies heavily on the computaion of Gröbner bases in non-commutative integral group rings.

These algorithms provide information on quotients of the group only, and cannot tell us whether or not the group itself is polycyclic. A possible method for proving that a group is nilpotent or polycyclic is described in the next section.

Let $G = \langle Y|S\rangle$ be a finitely presented group. Let $G = \Gamma^1(G) \supseteq \Gamma^2(G) \supseteq \ldots$ be the lower central series of G. The *nilpotent quotient algorithm* (NQA) attempts to calculate polycyclic presentations of the quotients $G/\Gamma^i(G)$, successively, for $i = 2, 3, \ldots$. It terminates if and only if the lower central series becomes constant.

An implementation for the special case of quotients that are finite p-groups for a fixed prime p was first undertaken by I.D. MacDonald in about 1973. Other successful and efficient implementations of this case have been done by Wamsley, Havas, and O'Brien (see [Havas, Newman, 80] or [O'Brien, 90] for further details and references). There have recently been two good implementations of a general nilpotent quotient algorithm, by Sims and Nickel. These are described in [Nickel, 93] and Chapter 11 of [Sims, 94].

These algorithms all proceed by first finding a polycyclic presentation that is not usually confluent, and then using the equations described in the preceding section to enforce confluence.

A general polycyclic quotient algorithm (PQA) was first described by Baumslag, Cannonito and Miller in [Baumslag, et. al., 81a] and [Baumslag, et. al., 81b]. It was not in a form suitable for immediate implementation, but the ideas have recently been developed into a practical algorithm and implemented by Lo [Lo, 96] a doctoral student of Sims.

Let $G = G^{(1)} \supseteq G^{(2)} \supseteq \ldots$ be the derived series of G. The algorithm successively tests whether $G/G^{(i)}$ is polycyclic, for $i = 2, 3, \ldots$ and, if so, it finds a confluent polycyclic presentation of it.

The main step in this process involves Gröbner bases. Suppose, more generally, than N is a normal subgroup of G with polycyclic quotient group $Q = G/N$, and that we have a polycyclic presentation of Q, together with the images of the generators of G as words in the polycyclic generators of Q. The problem is to determine whether $N/[N,N]$ is polycyclic (where $[N,N]$ is the commutator subgroup of N) and, if so, to find a polycyclic presentation of $G/[N,N]$.

We can regard $N/[N,N]$ as a $\mathbb{Z}Q$-module M, which is generated (as a module) by a finite set R consisting essentially of the relators of the polycyclic presentation of Q pulled back to G and evaluated as elements of $N/[N,N]$. The group $N/[N,N]$ (and hence also $G/[N,N]$) is polycyclic if and only if M is finitely generated as an abelian group.

The idea is to start with a free $\mathbb{Z}Q$-module L on R, to write down a rewriting system for a group E that looks like an extension of L by Q, but is in fact a presentation of $G/[N,N]$.

Each critical pair in this system gives rise to an element $m \in L$ that maps to zero in E, and it is shown that only finite many such elements m arise, and that $M = L/L'$, where L' is the finitely generated submodule of L generated by these zero elements m.

The problem of finding a normal form for $M = L/L'$, and hence determining whether it has finite rank as an abelian group can now be solved by using Gröbner bases.

The module L is a direct sum of copies of $\mathbb{Z}Q$, and $\mathbb{Z}Q$ has the free basis $\{g \in Q\}$ as an abelian group. But, as described in the preceding section, each $g \in Q$ has a unique expression of the form $x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$ (with some restrictions on some of the α_i), and the polycyclic generators x_i of Q can be regarded as non-commuting variables of a polynomial ring over $\mathbb{Z}$.

In [Lo, 96], it is proved (in fact, under the slightly more general assumption that Q is polycyclic-by-finite) that the Gröbner basis calculation for the submodule L' of

L always terminates, and so gives rise to a practical algorithm to solve the problem.

6 LECTURE 3 (cont'd): Verifying polycyclicity

It is of course undecidable whether or not a given presentation defines a polycyclic group but, if it does, then it is possible to verify this fact. The method of doing this to be described here, which makes use of a computer implementation of the Knuth-Bendix completion algorithm, was first proposed by Sims, and is described in Chapter 11 of [Sims, 94]. It requires the use of a polycyclic quotient algorithm. (Of course, if we are trying to verify nilpotency, then the NQA suffices.)

The first step is to use the PQA (or NQA) to compute the largest polycyclic quotient G/K of the given finitely presented group G. If G really is polycyclic, then K will be trivial, but we need to verify this.

Let the given presentation be $G = \langle\, Y \mid R \,\rangle$, where $Y = \{y_1, \ldots, y_m\}$.

The output of the PQA consists of a consistent polycyclic presentation of G/K on generating set $X = \{x_1, \ldots, x_n\}$ (and inverses), in the form described in the Section 4, together with the images of the y_j in G/K, given as words u_j in normal form in the generators x_i and x_i^{-1}.

It is then possible to use these relations to express each x_i as a word v_i in $Y \cup \{x_j | j < i\}$ and inverses. (In the existing implementations of the NQA, at least, the output is in such a form as to make this process straightforward.)

Let D be the set of relators $v_i x_i^{-1}$. Then D is effectively a set of definitions of the x_i in terms of the y_i, and so the presentation $\hat{G} = \langle\, Y \cup X \mid R \cup D \,\rangle$ defines a group isomorphic to G.

Now let $A = \{y_i, y_i^{-1}, x_j, x_j^{-1} | 1 \leq i \leq m, 1 \leq j \leq n\}$ be the complete set of monoid generators of $\hat{G}$, and define a wreath product ordering on A^* (as defined in Section 4), in which the list of generators, in decreasing order of levels, is

$$y_1^{-1}, y_1, y_2^{-1}, \ldots, y_m, x_1^{-1}, x_1, \ldots, x_n^{-1}, x_n.$$

We can now run the Knuth-Bendix completion algorithm on the presentation $\hat{G}$ using this ordering.

If G/K and G really are the same, then the equalities $y_j = u_j, y_j^{-1} = u_j^{-1}$ $(1 \leq j \leq m)$, together with each of the relations in the x_i and their inverses which constitute the consistent polycyclic presentation of G/K will all hold in $\hat{G}$, and it is easily seen that they form a complete rewriting system for $\hat{G}$ with respect to the specified ordering of A^*.

These relations are all essential (at least after writing u_j^{-1} in normal form), and will therefore inevitably be generated by the Knuth-Bendix process (although it could take a long time). On the other hand, if they are generated by Knuth-Bendix, then it follows immediately that G really is polycyclic, and so we can verify polycyclicity in this manner.

Let us now work through a fairly straightforward example, to illustrate this process. For group elements g, h, the commutator $[g, h]$ is defined to be $g^{-1}h^{-1}gh$, and we define $[g, h, k] = [[g, h], k]$, etc. The following group was proved nilpotent theoretically by Heineken in [Heineken, 61].

$$G = \langle\ a, b \mid [b, a, a, a], [b^{-1}, a, a, a], [a, b, b, b], [a^{-1}, b, b, b], \\ [a, ab, ab, ab], [a^{-1}, ab, ab, ab]\ \rangle.$$

Running the NQA (using the implementation by Nickel described in [Nickel, 93]) reveals that G has a largest nilpotent quotient of class 4, with 6 polycyclic generators $x_1, \ldots, x_6$, $I = \{6\}$ and $m_6 = 2$.

Since a maps to x_1 and b to x_2 we have $u_1 = x_1, u_2 = x_2$, and so we can take $v_1 = a, v_2 = b$. In fact, in a case like this, we can simplify things by identifying a and b with x_1 and x_2, and we will rename the generators $x_1, \ldots, x_6$ as a, b, c, d, e, f, respectively.

The relations in the full polycyclic presentation are

$$f^2 = 1,\ ba = abc,\ ba^{-1} = a^{-1}bc^{-1}d,\ ca = acd,\ ca^{-1} = a^{-1}cd^{-1},\ cb = bce, \\ cb^{-1} = b^{-1}ce^{-1},\ db = bdf,\ db^{-1} = b^{-1}df,\ ea = aef,\ ea^{-1} = a^{-1}ef,$$

where we have omitted relations between pairs of commuting generators.

Of these, we can take $d = [c, a], e = [c, b]$ and $f = [d, b]$ as definitions of d, e and f, respectively, and so we let

$$\hat{G} = \langle\ a, b, c, d, e, f \mid [b, a, a, a], [b^{-1}, a, a, a], [a, b, b, b], [a^{-1}, b, b, b], \\ [a, ab, ab, ab], [a^{-1}, ab, ab, ab], d = [c, a], e = [c, b], f = [d, b]\ \rangle,$$

and use this as input to the Knuth-Bendix algorithm, with the wreath-product ordering as above. In this example, the relations in the polycyclic presentation were produced very rapidly (in less than a second), thereby verifying nilpotency.

Although this particular example was very easy, in general the Knuth-Bendix process can struggle and get itself tied up with very long words if the implementation is too simplistic. There is a lengthy discussion of possible techniques for improving the performance in Chapters 2 and 3 of [Sims, 94], together with implementation details.

For example, it is often a good idea only to store relatively short reduction equations in the early stages of the process. Although this means that the resulting set of stored equations may not be complete at the end, it is relatively easy to verify completeness later by restarting on the enlarged set of rules, once the polycyclic relations have all been deduced.

A slightly more difficult example is

$$G = \langle\ a, b \mid [b, a, b], [b, a, a, a, a], [b, a, a, a, b, a, a]\ \rangle,$$

which turned out to be nilpotent of class 7, with 8 polycyclic generators. An interesting open question is whether or not the group is still nilpotent if we omit the third relator. The maximal nilpotent quotient has class 10.

7 LECTURE 4: Automatic groups

In this section, we present an overview of the theory of automatic groups. They provide the setting for a situation in which the Knuth-Bendix procedure can be useful even when it does not complete. In fact it constitutes one component of a more complicated algorithm which, when successful, yields a normal form for group elements and an efficient algorihtm for reducing words to normal form.

These methods, together with their underlying theory, make essential use of finite state automata (which we shall abbreviate to fsa). For background information on the theory of fsa's, and the related theory of regular languages, see any textbook on formal language theory, such as [Aho et al., 74]. The definition that we give here is of a complete deterministic fsa.

An 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* of M.

Some of the states are designated as *accept states*. If M is in an accept 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.

Let S be a finite rewriting system on A^*. Then it is possible to construct an fsa $M = M(S)$ for which $L(M)$ is equal to the set of irreducible strings in A^*; that is, the set of strings which do not contain the left hand side of any element of S as a substring.

For example, consider the complete rewriting system for the dihedral group of order 6 with $A = \{x, \bar{x}, y, \bar{y}\}$ that we considered in Section 2. The following fsa M has the required property. Let M have 5 states {0,1,2,3,4}, where 1 is the start state, and all states except 0 are accept states. The transition function τ is defined by the following table.

	0	1	2	3	4
x	0	2	0	0	0
$\bar{x}$	0	3	0	0	0
y	0	4	4	4	0
$\bar{y}$	0	0	0	0	0

The language accepted is $\{\varepsilon, x, \bar{x}, y, xy, \bar{x}y\}$.

The construction of $M(S)$ from S is reasonably easy to do computationally. It is also possible to construct such an $M(S)$ when S is infinite, but regular. In the author's implementation of the Knuth-Bendix completion process in KBMAG, the failure states of $M(S)$ are in one-one correspondence with the rules in S, and the particular failure state indicates the rule of which the left hand side has been read. This is used to assist in the reduction process.

If $G = \langle X|R \rangle$ is a finitely presented group, and $A = X \cup X^{-1}$, then a finite state automataton W 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.

If S is constructed from the group relations as before, then any two words equivalent under $\leftrightarrow^*$ are equal in the group, and each word has at least one equivalent irreducible word. This means that $M(S)$ is always a word-acceptor for G, and if S is complete, then $M(S)$ is a unique word-acceptor.

One of the standard operations that one can perform on an fsa M is to enumerate its language $L(M)$. (If we are only interested in the words up to a given length, then a depth-first search is more efficient than a breadth-first search.) This means that we can enumerate the elements of a finitely presented group.

If our system S is complete, then we get each element exactly once. If not, we get some repetition, but if we have run the Knuth-Bendix process for some time, then there may well be no repetition on short words.

Some of the most important applications that this subject has found have been outside of group theory, and in fields such as hyperbolic geometry and ergodic theory. In these applications, all that is required is to enumerate the group elements with as little repetition as possible.

A further application worth mentioning is to the calculation of growth-rates. For an fsa M, let $\alpha(n)$ denote the number of accepted words of M of length n. Then the function $f(t) = \sum_{i=0}^{\infty} \alpha(n)t^n$ is called the *growth function* of M, and it turns out that this is a rational function of t, which can be calculated from M.

If W is a unique word acceptor for a group, and all accepted words are shortest possible for the group element that they represent, then the growth function for W is the same as that for G. This can be calculated if we can find a finite complete set S using a shortlex ordering on A^*. More generally, such a word-acceptor can be found for shortlex automatic groups, which we are about to define.

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 ([Cannon, 84]). Bill Thurston noticed that some of the properties that were proved about these groups could be reformulated in terms of finite state automata.

Following the definition, several people were involved in the rush to develop the theory of automatic groups; the more straightforward properties were proved quickly, and some less easy results have been proved since then. There remain some irritating open questions, however.

The most comprehensive reference is the six-author book [Epstein et al., 92], and we refer the reader to this book for further information and references. In this section, we shall content ourselves with stating the definition and some of the most important properties that have been proved, and explaining briefly the application of string-rewriting methods to computation in automatic groups.

As before, G will be a group with finite generating set X, and we put $A = X \cup X^{-1}$.

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 (which we assume is not already in A) and put $A' = A \cup \{\$\}$. Let w and x be words in A^*. Then we adjoin a number of \$'s to the end of the shorter of the two words, to make their lengths equal.

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

For a word $w \in (A')^*$, we will denote the element of G that w maps onto by $\overline{w}$, where we regard \$ as mapping onto the identity element of G.

Definition 7.1 *Let G be a group with finite generating set X, and let $A = X \cup X^{-1}$. Then G is said to be* automatic *(with respect to X), if there exist* fsa*'s W, and M_a for each $a \in A$, 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 the padded pair (w^+, x^+) for $w, x \in A^*$ if and only if $w, x \in L(W)$ and $\overline{wa} = \bar{x}$ in G.

The idea is that the fsa M_a recognises multiplication on the right by a in $L(W)$. There are a number of equivalent definitions to this, and some other related definitions such as *asynchronously automatic groups*, in which the two words in M_a can be read at different speeds, and *biautomatic groups* in which multiplication by a generator on the left in $L(W)$ can also be recognised by an fsa. See [Epstein et al., 92] for details.

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 will of course depend on the chosen ordering for A).

It is shown in [Epstein et al., 92] that the property of being automatic depends only on the group G, and not on the generating set X, but the property of being shortlex automatic does depend on X, and it can even depend on the chosen ordering of A.

For example, the group $\langle x, y, a | a^2 = 1, xy = yx, a^{-1}xa = y \rangle$ is shortlex automatic using the ordering $x < x^{-1} < y < y^{-1} < a$, but not using the ordering $a < x < x^{-1} < y < y^{-1}$. All automatic groups are finitely presented.

The fsa W can be used to enumerate the elements of an automatic group, as described in the preceding section.

Furthermore, the automata M_a can be used to solve the word-problem for an automatic group in time that is quadratic in the length of the words being tested for equality, by replacing these words by the unique words in $L(W)$ that represent the same group elements.

It is not known whether the conjugacy problem is solvable in automatic groups in general, but it is known to be solvable in biautomatic groups.

The hyperbolic groups considered in [Cannon, 84] are automatic; in fact, all *word-hyperbolic* groups, as defined by Gromov, are shortlex automatic, with any choice and any ordering of the generating set. This class includes free groups, and various small-cancellation groups.

Euclidean groups (that is, extensions of torsion-free finitely generated abelian groups by finite groups, are automatic), but nonabelian torsion-free nilpotent groups are not automatic. It seems probable that nonabelian torsion-free polycyclic groups are not automatic, and this has been proved in some cases, but not in general.

More recently, it has been proved that all braid groups, Artin groups of finite type, and Coxeter groups are automatic. In fact, Coxeter groups are shortlex automatic with the natural generating set (see [Brink, Howlett, 93]).

The class of automatic groups is closed under direct products, free products with finite amalgamated subgroup, and HNN-extensions with finite conjugated subgroup.

If H has finite index in G, then G is automatic if and only if H is.

Free factors of automatic groups are automatic, but it has not been proved that direct factors are.

The author's KBMAG package (available via ftp) mentioned earlier has the capability of computing and proving the correctness of the finite state automata for shortlex automatic groups. An earlier version of this package is described in [Epstein et al., 91] and in [Holt, 96].

We shall give a brief description of the algorithm here. In fact, if we apply it to an arbitrary presentation, then it will terminate if and only if this presentation does define a shortlex automatic group, but of course we cannot bound the time it will take in advance.

Thus, it can be used to prove automaticity in some cases, but never to disprove it. In fact, particularly when the relators are long, it tends not to be successful for space reasons, even when the group is shortlex automatic.

Assume that $G = \langle X|R \rangle$ is shortlex automatic, using some fixed ordering of A. For a word $w \in A^*$ and an integer t with $0 \leq t \leq \text{length}(w)$, we define $w(t)$ to be the prefix of w of length t.

For a padded pair $\pi = (w^+, x^+) \in (A' \times A')^*$, we define the *word-difference set* $D(\pi)$ to be the set of elements in G of the form $\overline{w(t)}^{-1}\overline{u(t)}$ for some t. If Π is a set of padded pairs of this form, then $D(\Pi)$ is defined to be the union of the sets $D(\pi)$ for $\pi \in \Pi$. It is not difficult to show that $D(L(M_a))$ must be finite for all $a \in A$. This is the key to the algorithm.

Suppose that we run the Knuth-Bendix completion algorithm on the presentation as described above, and at regular intervals we stop and reduce the system as described in Section 2.

Let (w, x) be an essential rule, and let $w = w'a$ with $a \in A$. Then w' and x must both be in $L(W)$ (for otherwise, the rule would eventually become non-minimal), and so $(w', x) \in L(M_a)$.

It follows that, if S_∞ is the set of all essential rules, then $D(S_\infty)$ is finite (even though S_∞ itself will usually be infinite). This means that after running the Knuth-Bendix process for a finite amount of time, $D(S)$ will eventually contain the whole of $D(S_\infty)$. As usual, we have no idea exactly how long we will have to run it for. In practice, we run it until the set $D(S)$ has apparently become constant.

There are a number of problems with this. The worst is that, in some examples, a small number of elements of $D(S_\infty)$ take a very long time to appear. This situation needs further investigation, since it can sometimes be improved by processing the existing rules in a different order.

Another potential problem is that sometimes additional word-differences keep appearing that are coming from rules that later turn out not to be minimal. This is a nuisance, since we have no way of telling whether a particular rule will eventually turn out to be essential, but it does not appear to be a serious problem in most examples, since a few superfluous word-differences do not matter.

Once we have found the complete set $D(S_\infty)$, it can be used to construct the word-acceptor W. This is done by using the fact that, if a word w does not lie in $L(W)$, then it must contain the left-hand side of a rule in S_∞ as a subword. The finiteness of $D(S_\infty)$ implies that this property can be determined by an fsa (see [Epstein et al., 91] for details).

Although the sets $D(M_a)$ may contain elements that are not in $D(S_\infty)$, they themselves can be found by a similar process, and then the fsa's M_a can be constructed.

Of course, the correctness of all of this still depends on our unproved supposition that we have indeed found the complete set $D(S_\infty)$. However, the algorithm concludes with the so-called *axiom-checking* process, in which other (unfortunately much larger) fsa's are constructed which effectively prove the correctness of the whole calculation.

So, in some sense, fsa's are being used to prove results about other fsa's are doing. This is generally not theoretically possible for more powerful types of computing devices, such as Turing machines.

These programs run fairly easily on some straightforward examples, such as the Von Dyck triangle groups and various simple knot-groups.

Some more difficult examples in which they have been successful include the Fibonacci group $F(2,8)$, and the Picard group $SL(2, \mathbf{Z}[i])$.

A much more difficult example which we recently computed successfully with KBMAG is the Fibonacci group $F(2,9)$. Here, the word acceptor has 3251 states, the multipliers about 25000 states, and the whole calculation takes approximately 12 hours cpu-time and needs about 130 Megabytes of RAM (plus a similar amount of swap-space) on a Sun SparcStation20 machine.

These programs are effectively extending the range of the string-rewriting techniques for solving the word problem to certain groups in which the Knuth-Bendix completion process does not terminate with a finite complete set.

8 LECTURE 5: Subgroups and Cosets

In this section, we let H be a finitely generated subgroup of a finitely presented group $G = \langle X|R\rangle$, where H is defined by a finite generating set Y of words in A^* and, as usual, $A = X \cup X^{-1}$.

The Knuth-Bendix process can be used to derive rewrite-rules between the right cosets Hg of H in G. This application is described in Section 2.8 of [Sims, 94].

A new symbol, not already in the set A, which we shall also denote by H (since it represents the subgroup H), is appended to the alphabet.

For the initial rewrite-rules, we use the group relators as before, together with the rules $\{(Hy, H) \mid y \in Y\}$. Notice that H does not have an inverse, and so it does not cancel on the left.

When we run the Knuth-Bendix completion process, the rules that arise are either of the form (w_1, w_2) (the group rules), or (Hw_1, Hw_2) (the coset rules), where $w_1, w_2 \in A^*$ in both cases. In fact the group rules are exactly the same as they would be in the absence of the subgroup H.

Overlaps can occur between two group rules, or between a coset rule on the left and a group rule on the right.

The irreducible words in the completion of this system include the minimal representatives $w \in A^*$ for each group element, and the words Hw for the minimal representatives $w \in A^*$ of each right coset. (There are other irreducible words containing more than one occurrence of H, but they do not concern us.)

We are now mainly interested in the coset rules and the irreducible cosets, but unfortunately we cannot derive these without processing the group rules at the same time.

It can happen that the full completion of the system is infinite, but contains only finitely many coset rules. In that case, we would ideally like to stop as soon as we have found all of the coset rules. In the case where the index $|G : H|$ is finite, a method for checking when this has happened, is discussed in Section 3.10 of [Sims, 94].

These ideas can be extended to derive rewrite-rules between the generators y of the subgroup H, as described in [Hurt, 96]. We first need to introduce new and distinct symbols $\sigma(y), \sigma(y^{-1})$ for these generators and their inverses. We then replace the initial coset rules by $\{(Hy, \sigma(y)H) \mid y \in Y\}$ and introduce the rules $\{(\sigma(y)\sigma(y^{-1}), \varepsilon), (\sigma(y^{-1})\sigma(y), \varepsilon) \mid y \in Y\}$. We also allow the symbol H to cancel on the right in rewrite-rules, but not on the left.

It is convenient to use a wreath product ordering on words, in which the levels of the subgroup generators $\sigma(y), \sigma(y^{-1})$ are all less than those of the group generators in A.

We now get three types of rewrite-rules, the group rules as before, the coset rules (Hw_1, vHw_2), and the subgroup rules (v_1, v_2), where v, v_1 and v_2 are words in the subgroup generators $\sigma(y), \sigma(y^{-1})$.

It is proved in [Hurt, 96] that, if the Knuth-Bendix process completes, or if we halt the process because we know, from the criteria of Sims mentioned in the preceding paragraph, that we have found all coset rules, then the subgroup rules form a complete rewriting system for H. In particular, they define a finite presentation of H.

As a simple example, let G be the symmetric group S_4 and H a subgroup of index 4, isomorphic to S_3. The generators of G will be $\{a, b, \overline{b}\}$ and those of H $\{x, y, \overline{x}, \overline{y}\}$, where we have reverted to using bars to denote inverses. We use shortlex orderings for words in the generators of G or H alone.

For the initial group rules, we take

$$\{(a^2, \varepsilon), (b\bar{b}, \varepsilon), (\bar{b}b, \varepsilon), (b^2, \bar{b}), (\bar{b}a\bar{b}a, abab)\},$$

and for the initial coset and subgroup rules

$$\{(Hb, xH), (Ha\bar{b}aba, yH), (x\bar{x}, \varepsilon), (\bar{x}x, \varepsilon), (y\bar{y}, \varepsilon), (\bar{y}y, \varepsilon)\}.$$

The complete set of 25 rules contans 11 group rules (which we shall not list), the four coset rules

$$\{(Hb, xH), (Ha\bar{b}a, yHa\bar{b}), (H\bar{b}, \bar{x}H), (Haba, yxHab)\}$$

and the nine subgroup rules

$$\{(x\bar{x}, \varepsilon), (\bar{x}x, \varepsilon), (y^2, \varepsilon), (\bar{y}, y), (x^2, \bar{x}), (yxy, \bar{x}), (\bar{x}y, yx), (y\bar{x}, xy), (xyx, y)\}.$$

The theory of automatic groups can also be generalised to deal with cosets of subgroups rather than with group elements. This was first attempted in [Redfern, 93] and developed further in [Hurt, 96]. (See also [Holt, Hurt, 96].) Definition 7.1 is modified as follows.

Definition 8.1 *Let G be a group with finite generating set X, and let $A = X \cup X^{-1}$, and H a subgroup of G. Then G is said to be* coset automatic*, if there exist* fsa*'s W, and M_a for each $a \in A$, such that:*
(i) W has input alphabet A, and accepts a unique word in each right coset of H in G;
(ii) Each M_a has input alphabet $A' \times A'$, it accepts only padded pairs, and it accepts the padded pair (w^+, x^+) for $w, x \in A^*$ if and only if $w, x \in L(W)$ and $H\overline{wa} = H\bar{x}$.

The fsa's W and M_a are called the *coset word acceptor* and the *coset multipliers*, respectively, and the complete collection of fsa's is called an *automatic coset system* for H in G. As in the automatic group case, it can be proved that the existence of such a system is independent of the choice of generating set X of G.

However, the author's programs that compute these structures work only for *shortlex* automatic coset systems, for which the accepted word for a coset Hg is the least word in that coset under a shortlex ordering.

Considerably fewer theoretical results have been proved about coset automatic groups than about plain automatic groups. However, it was shown in [Redfern, 93] that quasiconvex subgroups H of word-hyperbolic groups G always have shortlex coset automatic systems, for any choice of X.

(The subgroup H is said to be quasiconvex in G if there exists a constant κ such that, for any word $w \in A^*$ with $\overline{w} \in H$, and any t with $0 \leq t \leq \text{length}(w)$, there is a word $v \in A^*$ of length at most κ such that $w(t)v \in H$. See [Gersten, Short, 94] for the basic theory and properties of such subgroups.)

However, successful calculations of automatic structures using the author's programs suggest that the existence of automatic coset systems is quite common, and is certainly not restricted to quasiconvex subgroups of word-hyperbolic groups.

Algorithms for calculating shortlex automatic coset systems were first implemented by Redfern, and are described in [Redfern, 93]. Their ability to enumerate words for the coset representatives of subgroups led to an interesting application to the drawing of limit sets of Kleinian groups, which is described in [McShane et. al., 1994].

These implementations had the disadvantage that it was not possible to verify the correctness of the computed systems by an axiom checking process. The more recent implementations by the author do not suffer from these restrictions, and they follow a similar general structure to the algorithms for automatic groups described in Section 7.

A new feature arises from the fact that the two-variable machines need to accept word-pairs (w_1, w_2) arising from equalities in the group of the form $w_1 =_G vw_2$, where $w_1, w_2 \in A^*$ and $v \in H$.

The solution adopted is to construct fsa's that have more than one initial state (and so they are non-deterministic in this respect only), where the initial states are labeled by the words v that arise in these equations. This means that they will only work when there are only finitely many such words v altogether, but it can be shown that this is indeed the case for quasiconvex subgroups of word-hyperbolic groups.

When the programs complete successfully, they can also compute a presentation of the subgroup H if required. (This is currently output on a set of generators of H chosen by the program; in future, it may be possible to calculate a presentation on the given generators Y of H.) The theory of these calculations are described in [Hurt, 96] and [Holt, Hurt, 96].

We will not into any further details here, but will describe one example of intermediate difficulty. We take

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

This was proposed by Heineken several years ago as a possible candidate for a finite group with a balanced cyclic presentation.

However, we were able to show, using KBMAG, that it is shortlex automatic (in fact, it is even word-hyperbolic), and its generators have infinite order, so it cannot be finite.

We used this group as input to our coset programs, with the subgroup $H = \langle [x, y], [y, z], [z, x] \rangle$.

The coset word acceptor has 58 states, and the multipliers 2536 states. The computation of the automatic coset system and verification of its correctness took a total of about 600 seconds of cpu-time and used about 7 Megabytes of RAM on a Sun SparcStation 20 machine running under Solaris. Computing a subgroup presentation took about 2800 seconds, and used up to 27 Megabytes.

The presentation has 10 monoid generators (that is, 5 group generators), which simplifies to a free group of rank 3, and so H must in fact be a free group on its defining generators.

As a final related application, KBMAG can sometimes calculate an fsa of which the language consists of the elements of the subgroup H.

More precisely, it accepts the shortlex least word in A^* for each element of H. It is proved in [Gersten, Short, 94] that this is a regular language for a quasiconvex subgroup of a word-hyperbolic group, and the computation will (at least theoretically) succeed in this case.

It is also possible to use such an fsa to find generators of the subgroup that it accepts. Thus, one can, for example, start with two subgroups H and K of G, find word acceptors W_H and W_K for both of them, then compute an automaton $W_{H\cap K}$ whose language is the intersection of those of W_H and W_K, and therefore is a word-acceptor for $H \cap K$, and finally use this to compute generators of $H \cap K$.

Similar calculations can be done by using Gröbner bases in the group algebra KG of G over a field K. These work, for example, for all subgroups H of a free group G, as described in [Rosenmann, 93], for example.

References

[Aho et al., 74] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms*, (Addison-Wesley, Reading, MA, 1974).

[Baumslag, et. al., 81a] G. Baumslag, F.B. Cannonito & C.F. Miller, 'Some recognizable properties of solvable groups', *Math. Zeit.* 178, 289–295 (1981).

[Baumslag, et. al., 81b] G. Baumslag, F.B. Cannonito & C.F. Miller, 'Computable algebra and group embeddings', *J. Alg.* 69, 186–212 (1981).

[Brink, Howlett, 93] B. Brink and R.B. Howlett, 'A finiteness property and an automatic structure for Coxeter groups', *Math. Ann.* 296, 179–190 (1993).

[Buchberger, 87] B. Buchberger, 'The History and Basic Features of the Critical-Pair/Completion Procedure', *J. of Symbolic Computation* 3, 3–38 (1987).

[Cannon, 84] J.W. Cannon, 'The combinatorial structure of cocompact discrete hyperbolic groups', *Geom. Dedicata* 16, 123–148 (1984).

[Epstein et al., 92] D.B.A. Epstein, J.W. Cannon, D.F. Holt, S. Levy, M.S. Paterson, W.P. Thurston, *Word Processing and Group Theory*, (Jones and Bartlett, 1992).

[Epstein et al., 91] D.B.A. Epstein, D.F. Holt and S.E. Rees, 'The use of Knuth-Bendix methods to solve the word problem in automatic groups', *J. of Symbolic Computation* 12, 397–414 (1991).

[Gersten, Short, 94] S.M. Gersten and H.B. Short, 'Rational Subgroups Of Biautomatic Groups', *Ann. Math.* 134, 125–158 (1991).

[Gilman, 79] R.H. Gilman, 'Presentations of groups and monoids', *J. of Algebra* 57, 544–554 (1979).

[Gilman, 84] R.H. Gilman, 'Enumerating infinitely many cosets', in *Computational Group Theory (Durham 1982)*, ed. M. Atkinson, Academic Press, 51–55 (1984).

[Havas, Newman, 80] G. Havas and M.F. Newman, 'Applications of computers to questions like those of Burnside', in: *Burnside Groups*, Lecture Notes in Math., vol. 806, ed. J.L. Mennicke, Springer-Verlag, Berlin and New York, 211–30 (1980).

[Heineken, 61] H. Heineken, 'Engelsche Elemente der Länge drei', *Illinois J. Math.* 24, 681–707 (1961).

[Hermiller, 94] Susan M. Hermiller, 'Rewriting systems for Coxeter groups', *J. of Pure and Applied Algebra* 92, 137–148 (1994).

[Holt, 96] D.F. Holt, 'The Warwick automatic groups software', in *Geometrical and Computational Perspectives on Infinite Groups*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 25, ed. Gilbert Baumslag et. al., 69–82 (1996).

[Holt, Hurt, 96] D.F. Holt and D.F. Hurt, 'Computing automatic coset systems and subgroup presentations', submitted to *J. of Symbolic Computation.*

[Hurt, 96] Darren F. Hurt, *The Use of Knuth-Bendix Methods and Automatic Coset Systems for Solving the Generalized Word Problem and Finding Subgroup Presentations*, (PhD Thesis, University of Warwick, 1996).

[Knuth, Bendix, 70] D.E. Knuth and P.B. Bendix, 'Simple word problems in universal algebra', in *Computational Problems in Abstract Algebra*, ed. J. Leech, Pergamon Press, 263–297 (1970).

[Le Chenadec, 86] P. Le Chenadec, 'A catalogue of complete presentations', *J. of Symbolic Computation* 2, 363–381 (1986).

[Lo, 96] Eddie H. Lo, *A Polycyclic Quotient Algorithm*, (Ph. D. Dissertation, Rutgers University, New Jersey, 1996).

[Lyndon, Shupp, 77] R.C. Lyndon and P.E. Shupp, *Combinatorial Group Theory*, (Springer-Verlag, Berlin-Heidelberg-New York, 1977).

[McShane et. al., 1994] G. McShane, J. Parker and I. Redfern, "Drawing Limit Sets of Kleinian Groups Using Finite State Automata", Experimental Mathematics, **3** (1994), 153–172.

[Nickel, 93] WERNER NICKEL, *Central Extensions of Polycyclic Groups*, (Ph. D. Thesis, ANU, Canberra, 1993).

[O'Brien, 90] E.A. O'BRIEN, 'The p-group generation algorithm', *J. of Symbolic Computation* 9, 677–698 (1990).

[Redfern, 93] IAN REDFERN, *Regular Systems of Reductions in Groups* (Ph. D. Thesis, University of Warwick, 1993).

[Rosenmann, 93] A. ROSENMANN, 'An algorithm for constructing Gröbner and free Schreier bases in free group algebras', *J. of Symbolic Computation* 16, 523–549 (1993).

[Sims, 94] C.C. SIMS, *Computation with Finitely Presented Groups*, (Cambridge University Press, 1994).

[Vaughan-Lee, 70] M.R. VAUGHAN-LEE, 'An aspect of the nilpotent quotient algorithm' in *Computational Group Theory (Durham 1982)*, ed. M. Atkinson, Academic Press, 75–83 (1984).

[Wamsley, 70] J.W. WAMSLEY, 'The deficiency of metacyclic groups', *Proc. Amer. Math. Soc.* 24, 724–726 (1970).

Derek F. Holt
Mathematics Institute
University of Warwick
Coventry CV4 7AL
Great Britain