Fast Searching for Andrews–Curtis Trivializations

R. Sean Bowman and Stephen B. McCaul

CONTENTS

2000 AMS Subject Classification: 20-04, 20F05

Keywords: Andrews–Curtis conjecture, computer search

A high-performance computer program for searching a tree of words in the group generated by Andrews–Curtis transformations is presented. The program enumerates words in this group using a disk-based hash table to store words already seen. We discuss several issues that affect the performance of the program and examine the growth of the size of the Cayley graph being searched.

1. INTRODUCTION

The Andrews–Curtis conjecture states that every balanced presentation of the trivial group can be transformed to the presentation $\langle x_1, \dots, x_n \,|\, x_1, \dots, x_n \rangle$ by a sequence of certain moves. We present a program that searches for sequences of these moves and applies them to some presentations that are particularly tough to reduce. Our program makes effective use of disk-based memory in order to scale to larger searches and relies on a concept of "canonical form" in order to reduce the size of the search space.

Let $P = \langle x_1, \dots, x_i \,|\, r_1, \dots, r_j \rangle$ be a group presentation. We say that P is *balanced* if $i = j$. The *trivial presentation* of the trivial group is the balanced presentation $\langle x_1, \dots, x_n \,|\, x_1, \dots, x_n \rangle$.

The group of Andrews–Curtis transformations on balanced presentations with n generators, AC_n, is the group generated by the following transformations:

1. replace r_i with $r_i r_j$ for some $j \neq i$,
2. replace r_i with r_i^{-1}, and
3. replace r_i with $g r_i g^{-1}$, where g is a generator or its inverse.

There are $n^2 - n$ moves of the first kind, n of the second kind, and $2n^2$ of the third kind. The group AC_n acts on the set of balanced presentations with n generators in the obvious way. Two balanced presentations on n generators P_1 and P_2 are *AC-equivalent* if there is an element $\sigma \in \mathrm{AC}_n$ such that $\sigma P_1 = P_2$. Andrews and

Curtis [Andrews and Curtis 65] conjectured that every balanced presentation of the trivial group on n generators is AC-equivalent to the trivial presentation. Another way of saying this is that the orbit of the trivial presentation of the trivial group under AC_n includes every balanced presentation of the trivial group with n generators.

In this paper, we present software for finding Andrews–Curtis trivializations of balanced presentations of the trivial group. The software is available at http://www.math.utexas.edu/users/sbowman/ac-bfs.tar.gz and is licensed under the GNU GPL.

Using breadth-first search to search for Andrews–Curtis trivializations is a technique first investigated by Havas and Ramsay [Havas and Ramsay 03]. They were able to verify that the Akbulut–Kirby presentation of the trivial group for $n = 2$ is AC trivial. Miasnikov [Miasnikov 99] used a genetic programming approach to eliminate potential counterexamples to the Andrews–Curtis conjecture.

Casson [Casson 03] described a set of programs to examine equivalence classes of group presentations modulo Andrews–Curtis transformations. We adapted his ideas on using a "canonical form" for presentations in order to make it easier to determine which have been seen before.

2. BREADTH-FIRST SEARCH

One way to show that a balanced presentation on n generators P is AC-equivalent to the trivial presentation is to enumerate words in AC_n, apply them to P, and see whether the result contains the trivial presentation. In order to enumerate words of AC_n, we use a breadth-first search. Breadth-first search is a technique for searching a graph for some distinguished vertex. In this case, the graph is a tree with the identity transformation at its root and all words of length m at the mth level. Since some sequences of transformations yield the identity transformation (for example, inverting r_1 twice), the tree contains redundant words.

Two presentations may differ by a cyclic permutation of one or more relators. In order to know when two presentations are the same, equivalence classes of presentations under cyclic permutation of relators are represented by a presentation in canonical form. Generators are ordered and assigned a numerical value. A relator can then be represented as a tuple of numbers. For example, if $a = 1$ and $b = 2$, the word $abab^{-1}a^{-1}b^{-1}$ is represented by the tuple $(1, 2, 1, -2, -1, -2)$. We examine the relator's cyclic permutations and select the lexicographically largest one as the canonical representative. To get the canonical presentation, we sort the relators lexicographically, padding with zeros when the tuples are of different sizes. For example, the canonical representative of

$$\langle a, b \mid abab^{-1}a^{-1}b^{-1}, a^2b^{-3}\rangle$$

is

$$\langle a, b \mid a^2b^{-3}, bab^{-1}a^{-1}b^{-1}a\rangle.$$

The algorithm begins by inserting the presentation to be checked for reducibility into a queue. This presentation, which represents the identity transformation, is the root of the tree that will be constructed. The algorithm loops by taking a presentation off the queue, applying each of the $3n^2$ generators of AC_n to it, and putting the presentations that haven't been seen before back on the queue. To tell which presentations have been seen before, we use a hash table.

Our hash table uses a multiplicative hash scheme [Knuth 73] based on canonicalized presentations. This hash scheme produces a number based on the tuple representation of a group presentation P as discussed above. The hash value is used to index an array of indices to lists of presentations on disk. There is one list for every hash value so that all presentations with identical hash values can be quickly compared with P. If no presentation matches P then P is not in the hash table, and P can be added to the hash table by appending it to the list of other presentations with the same hash value.

In order to search large trees, the hash table is kept on disk. Since disk IO is much slower than memory, some performance aspects of the hash table merit attention. The index of presentation lists is in memory and can determine without any disk access whether P is not a member when no presentation with the same hash value of P has been added to the hash table. When a presentation is added it is cached in RAM and written to disk in large groups to reduce the number of separate disk accesses. This cache also functions as a read cache when a presentation list has a member that has not yet been written to disk. When a search starts running, the majority of membership tests take no disk access, since most presentation lists are empty. During this phase the main use of time is in computing the hash values. As the search progresses more time is spent reading in presentation lists from disk. This is the dominant time use for the majority of a large search.

The algorithm emits a proof when it has reduced all but one relator to a single letter. This means that our algorithm doesn't actually find a word $\sigma \in \mathrm{AC}_n$ such that σP is the trivial presentation. A word that reduces all but one relator to one letter can be easily extended to a word that *does* trivialize P by repeated conjugation and multiplication by the single-letter relators. This optimization greatly reduces the size of the search space (especially for the case $n = 2$). Note that this optimization also makes the forward search for the word in AC_n that trivializes a presentation much more practical than the backward search for the inverse of this word.

3. EXPERIMENTS AND DISCUSSION

The presentation $\langle a, b \mid abab^{-1}a^{-1}b^{-1}, a^n b^{-(n+1)}\rangle$ for $n \in \mathbb{N}$ is a balanced presentation of the trivial group due to Akbulut and Kirby. This presentation for $n = 3$ is the smallest potential counterexample to the Andrews–Curtis conjecture [Havas and Ramsay 03]. We ran a series of experiments in which we searched for trivializations of the Akbulut–Kirby presentations for $n = 2$, 3, 4, and 5, varying the maximum relator length. We searched the tree for $n = 2$ from maximum relator length 10 to 25, the $n = 3$ tree from 10 to 17, $n = 4$ from 10 to 16, and $n = 5$ from 11 to 18. In each case the program was to stop if a trivialization was found, but trivializations were found only for the $n = 2$ case.

The computer used for exploring the Akbulut–Kirby presentations is an IBM z800 mainframe with a Sharc disk array. The IBM mainframe is a class of computers well known for its high-performance disk. This is due to the use of dedicated processors for performing disk operations, high-speed communications between CPU and disk, and the extensive use of cache. The Sharc array used during our research had 8 GB of cache and the operating system had access to 1 GB of system RAM. The software was run under Linux for z/Series.

Even when duplicate presentations are eliminated, the number of presentations in the tree of AC moves generated by breadth-first search grows exponentially at first, as shown in Figure 1. As the search continues, many of the presentations are found in the hash table, and so the number of presentations eventually reaches a plateau.

With two relators constrained to a length of fewer than 17 generators, a typical modern computer with 1 GB of RAM can hold a hash table with about 32 million entries. To compute the Akbulut–Kirby tree for $n = 3$ with relators constrained to a length of fewer than 17 generators, 85 million presentations must be searched (see Figure 2). The large size of the tree necessitates the use of disk-based storage.

The $n = 3$ run with maximum relator length 17 was the longest run, taking 93 hours. Approximately 41 GB of disk space was used. Figure 3 shows that the search space grows exponentially with increasing maximum relator length.

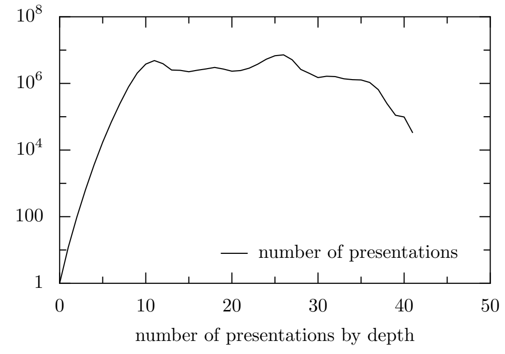

FIGURE 1. Number of presentations by depth for AK $n = 3$ with relator length less than 18.

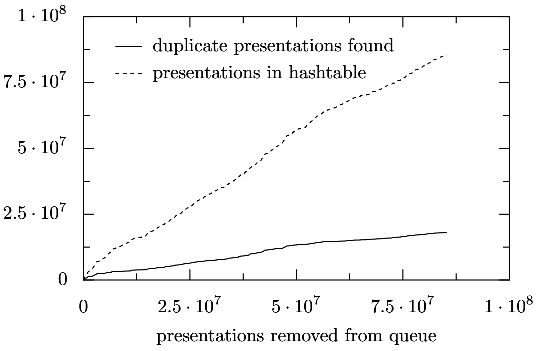

FIGURE 2. Queue statistics for AK $n = 3$, maximum relator length 17.

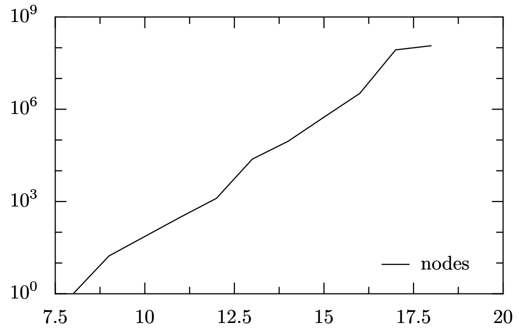

FIGURE 3. Number of presentations for AK $n = 3$ by relator lengths.

Although we couldn't crack the Akbulut–Kirby presentation, our software is able to find relatively lengthy trivializations of a sequence of presentations due to Gordon; see [Brown 84]. These presentations have the form $\langle a, b \mid a = [a^m, b^n], b = [a^p, b^q]\rangle$ for $m, n, p, q \in \mathbb{N}$.

Our software showed that all ten presentations in this sequence whose total relator length is 14 are AC-trivializable, and most of these have relatively short proofs. However, the presentation $\langle a, b \mid aba^{-2}b^{-1}, ab^3a^{-1}b^{-4}\rangle$ shows the advantages of disk-based storage. Its trivialization consists of 20 moves and was found using the heuristic of expanding the presentation with smallest total relator length after 6 moves.

Beginning with the presentation

$$\langle a, b \mid aba^{-2}b^{-1}, ab^3a^{-1}b^{-4}\rangle$$

we obtain

$\langle a, b \mid ba^2b^{-1}a^{-1}, ab^3a^{-1}b^{-4}\rangle$ (invert r_0)
$\langle a, b \mid b^2a^2b^{-1}a^{-1}b^{-1}, ab^3a^{-1}b^{-4}\rangle$ (conjugate r_0 by b)
$\langle a, b \mid ba^2b^{-1}a^{-1}, ab^3a^{-1}b^{-2}a^2b^{-1}a^{-1}b^{-1}\rangle$
(multiply $r_1 = r_1r_0$)
$\langle a, b \mid ba^2b^{-1}a^{-1}, b^2a^{-1}b^{-2}a^2b^{-1}a\rangle$ (multiply $r_1 = r_1r_0$)
$\langle a, b \mid ba^2b^{-1}a^{-1}, ab^2a^{-1}b^{-2}a^2b^{-1}\rangle$ (conjugate r_1 by a)
$\langle a, b \mid ba^2b^{-1}a^{-1}, ba^{-1}b^{-2}a^4\rangle$ (multiply $r_1 = r_1r_0$)
$\langle a, b \mid ba^2b^{-1}a^{-1}, aba^{-1}b^{-2}a^3\rangle$ (conjugate r_1 by a)
$\langle a, b \mid ba^2b^{-1}a^{-1}, b^{-2}a^3ba\rangle$ (multiply $r_1 = r_1r_0$)
$\langle a, b \mid ba^2b^{-1}a^{-1}, a^3bab^{-2}\rangle$ (conjugate r_1 by b^2)
$\langle a, b \mid ba^2b^{-1}a^{-1}, a^2bab^{-1}a^2b^{-1}\rangle$ (multiply $r_1 = r_1r_0$)
$\langle a, b \mid ba^2b^{-1}a^{-1}, abab^{-1}a^4b^{-1}\rangle$ (multiply $r_1 = r_1r_0$)
$\langle a, b \mid ba^2b^{-1}a^{-1}, ab^{-1}a^6\rangle$ (multiply $r_1 = r_1r_0$)
$\langle a, b \mid ba^2b^{-1}a^{-1}, a^7b^{-1}\rangle$ (conjugate r_1 by ba^{-1})
$\langle a, b \mid a^2b^{-1}a^6, a^7b^{-1}\rangle$ (multiply $r_0 = r_0r_1$)
$\langle a, b \mid a^{-6}ba^{-2}, a^7b^{-1}\rangle$ (invert r_0)
$\langle a, b \mid a^{-6}ba^{-2}, ab^{-1}a^6\rangle$ (conjugate r_1 by ab^{-1})
$\langle a, b \mid a^{-6}ba^{-2}, a^{-1}\rangle$ (multiply $r_1 = r_1r_0$)

The importance of maximum depth and maximum relator length to computer optimization should be obvious. However, these factors play a crucial role in the viability of the conjecture itself, as embodied in Lemma 3.3 below.

Definition 3.1. Let $\mathcal{P}$ consist of all balanced presentations of the trivial group on n generators. Let

$$c(P) = \max_i \{|r_i| \colon r_i \text{ is a relator of } P\}$$

denote the complexity of a balanced presentation on n generators P. Define $\mathcal{P}_i = \{P \in \mathcal{P} \colon c(P) \leq i\}$, the set of balanced presentations on n generators with complexity at most i. Let $\delta, \gamma : \mathbb{N} \to \mathbb{N}$ be defined by

$$\delta(m) = \max\{|\sigma| \colon P \in \mathcal{P}_m, \sigma \in \mathrm{AC}_n, \\ \sigma \text{ is the shortest word trivializing } P\}$$

and

$$\gamma(m) = \max\{|r_i| \colon P \in \mathcal{P}_m, t = \tau_1\tau_2\dots\tau_j \in \mathrm{AC}_n, \\ t \text{ is the shortest word trivializing } P \text{ and} \\ r_i \text{ is a relator of one of} \\ P, \tau_jP, \tau_{j-1}\tau_jP, \dots, tP.\}$$

Given a presentation $P \in \mathcal{P}$, $\delta(c(P))$ is an upper bound on the length of a sequence of Andrews–Curtis moves needed to trivialize P if it is trivializable. Correspondingly, $\gamma(c(P))$ is the maximum amount by which any relator grows when such a P is trivialized. If P is not AC-trivializable, we can discover this by noting that no word $\sigma \in \mathrm{AC}_n$ where $|\sigma| \leq \delta(c(P))$ trivializes P. In other words, δ represents searching the tree of AC moves to a certain depth, and γ represents searching this tree subject to the constraint that all searched nodes have a certain maximum complexity.

Definition 3.2. The Andrews–Curtis decision problem asks whether there is an AC-trivialization of a given presentation $P \in \mathcal{P}$.

Note that the Andrews–Curtis conjecture implies that this decision problem is decidable, and the answer is always yes.

Lemma 3.3. *If either δ or γ is computable, then the Andrews–Curtis decision problem is decidable.*

Proof: Given $P \in \mathcal{P}$, there is a finite number of presentations with complexity at most $\delta(c(P))$ or $\gamma(c(P))$ reachable from P by Andrews–Curtis moves. If any of these presentations is the trivial presentation, then P is AC-trivializable. Otherwise, it is not. □

4. CONCLUSION

We have presented a program that enumerates words in AC_n. The program makes effective use of disk-based storage to ensure scalability, but the longer access times for disk-based storage mean that performance must receive careful consideration. In order to reduce the size of the search space (and thus increase performance), a notion of "canonical presentation" was introduced. This concept simplifies the hash table used to keep track of which presentations have been visited.

We applied the program to the search for AC trivializations of the smallest potential counterexample of the AC conjecture, the Akbulut–Kirby presentation for $n = 3$. We showed that some presentations in a sequence due to Gordon are AC-trivializable and demonstrated that disk-based storage is useful for finding long trivializations that would not fit in the memory of a typical computer. Finally, we discussed some connections between computability and the Andrews–Curtis conjecture.

ACKNOWLEDGMENTS

Time on the mainframe was graciously donated by Rocket Software, Inc. Thanks to Yo'av Rieck and Chaim Goodman-Strauss for many valuable discussions on this topic.

REFERENCES

[Andrews and Curtis 65] J. Andrews and M. Curtis. "Free Groups and Handlebodies." *Proceedings of the American Mathematical Society* 16 (1965).

[Brown 84] R. Brown. "Coproducts of Crossed P-Modules: Applications to Second Homotopy Groups and to the Homology of Groups." *Topology* 23:3 (1984), 337–345.

[Casson 03] A. Casson. "The Poincaré Conjecture and the Andrews–Curtis Conjecture." Lectures given at the Spring Lecture Series, University of Arkansas, 2003.

[Havas and Ramsay 03] G. Havas and C. Ramsay. "Breadth-First Search and the Andrews–Curtis Conjecture." *International Journal of Algebra and Computation* 13:1 (2003).

[Knuth 73] D. E. Knuth. *The Art of Computer Programming.* Vol. 3: *Sorting and Searching.* Reading, MA: Addison-Wesley, 1973.

[Miasnikov 99] A. D. Miasnikov. "Genetic Algorithms and the Andrews–Curtis Conjecture." *Internat. J. Algebra Comput.* 9:6 (1999), 671–686.

R. Sean Bowman, Department of Mathematical Sciences, University of Arkansas, Fayetteville, AR 72701 (sean@rootnode.com)

Stephen B. McCaul, Rocket Software, Inc., Bentonville, AR 72712 (stephen@gizmoworks.com)

Received November 7, 2005; accepted December 5, 2005.