An Alternative Proof That the Fibonacci Group F(2, 9) is Infinite

Derek F. Holt

This note contains a report of a proof by computer that the Fibonacci group $F(2,9)$ is automatic. The automatic structure can be used to solve the word problem in the group. Furthermore, it can be seen directly from the word-acceptor that the group generators have infinite order, which of course implies that the group itself is infinite.

For $n \geq 2$ an integer, the Fibonacci groups $F(2,n)$ is defined by the presentation

$$\langle a_1, \ldots, a_n \mid a_1 a_2 = a_3,\ a_2 a_3 = a_4,\ \ldots,\ a_{n-1} a_n = a_1,\ a_n a_1 = a_2 \rangle.$$

These groups have been favourite test examples in combinatorial group theory for many years. By 1974, it had been determined that they were finite for $n = 2, 3, 4, 5$ and 7 and infinite for all other values of n, except possibly 9. The finiteness proofs were either by hand or by computer, using coset enumeration, whereas the infiniteness proofs either constructed explicit epimorphisms onto infinite groups or (for $n > 10$) used small cancellation theory (for further details and references see [Johnson 1980, Sections 9 and 26], for example).

The remaining case $F(2,9)$ remained open until 1990, when it was finally proved infinite [Newman 1990]. In the meantime, it had been shown that this group had a finite quotient of order $152 \cdot 5^{741}$ [Havas et al. 1979], and Newman was able to use the structure of this quotient, together with some theoretical results, to show that it in fact had finite quotients of order $152 \cdot 5^t$, for arbitrarily large values of t. This of course proved that the group was infinite, but it provided no information about the

1058-6458/96 $0.50 per page

group itself, as distinct from its finite quotients. In particular, it remained an open problem whether the generators a_i had finite or infinite order. (In fact, it still appears to be unknown whether there exists any finitely presented infinite torsion group.)

In [Helling et al. 1994], it is shown that certain Fibonacci groups are fundamental groups of hyperbolic three-manifolds, which implies immediately that they are automatic, and in fact short-lex automatic. (The notion of an automatic group is briefly defined below; for the general theory and details, see [Epstein et al. 1992].) However, these proofs do not apply to $F(2,n)$ for odd values of n.

The purpose of this note is to report a successful computer proof that $F(2,9)$ is automatic, and that its generators have infinite order. The programs used in the proof implement algorithms described in [Epstein et al. 1991] and, more recently, in [Holt 1994]. It was much easier to prove the automaticity of $F(2,6)$, $F(2,8)$ and $F(2,10)$ computationally, since the associated automatic structures are much smaller. The author had been attempting the calculation on $F(2,9)$ for several years, but it was only recently that a computer with enough memory became available.

Of course, with an enormous machine calculation of this nature, one inevitably asks how far the result can be trusted, and whether it is likely that a small logical or other error in the code (which cannot realistically be ruled out as impossible) could have resulted in an incorrect final result. I am of the opinion that this is extremely unlikely in this particular case, for the following reasons. The correct automatic structure is the result of a sequence of constructions of approximations to this structure, and the eventual correct structure is considerably smaller than the incorrect approximations. Also, there is a final stage to the computation, in which the structure is checked for correctness by another program; this so-called "axiom-checking" program constructs a series of much larger structures, in pairs, and the components of each pair have to be identical for the verification process to succeed. In addition, all of the calculations have been carried out successfully by two radically different versions of the complete package, and yielded the same results.

Roughly speaking, a group G, together with a finite set A that generates G as a monoid, is automatic if there exist two finite-state automata W and M, the *word-acceptor* and the *multiplier*, with the following properties. The word-acceptor W has input language A, and must accept a unique word in A^* for each group element. The multiplier M reads pairs of words (w_1, w_2), with $w_1, w_2 \in A^*$, and accepts such a pair if and only if w_1 and w_2 are both accepted by W and $w_1^{-1}w_2$ is equal in G to one of the generators a_i. (See the references above for a more detailed definition, and other equivalent definitions.) It turns out that automaticity is a property of the group, and is independent of the generating set, although the automata W and M will of course depend on A. The Warwick programs used in the calculation for $F(2,9)$ are only capable of calculating short-lex automatic structures, that is, those in which W accepts the lexicographically least amongst the shortest words that map onto a particular group element. (This assumes that an ordering has been specified for A.) The automatic structure can be used to solve the word-problem in G efficiently, by reducing words in A^* to their representatives in the language of W.

For the calculation in $F(2,9)$, we used the ordered monoid generating set

$$\{a_1, a_1^{-1}, a_2, a_2^{-1}, \ldots, a_9, a_9^{-1}\}.$$

We have not experimented much with other generating sets or orderings, but the ordering does not seem to have much influence on the difficulty of the computation, whereas other generating sets (such as one with only two group generators) seem to make things much more difficult. Our general experience in this area suggests that the "natural" generating set is the best to use, whenever this makes sense. Of course, the large number of generators does increase the space requirements in some places. The final correct automata W and M

have 3251 and 25741 states. Some of the intermediate automata constructed were considerably larger than this; further technical details follow below.

By looking at the transitions of W, one can observe immediately that when reading the word a_1^n for $n > 0$, the automaton starts in state 1, moves to state 2, then to state 20, then to state 172 and then to state 686, where it remains. Since all positive states of W are accepting, this shows that a^n is an accepted word for all n, which means that a^n cannot be equal to the identity for any $n > 0$. Thus a has infinite order.

We conclude with some technical details of the computation. It was done on a SPARCstation 20 with 256 Mb (megabytes) of core memory. As mentioned above, it was successfully completed twice, the first time using older versions of the software, which took several weeks of continuous CPU time. The second time, we were able to use a new, completely rewritten, version of the code. The run then took about 12.5 hours of CPU time, and used a maximum of just over 100 Mb of addressable memory, in addition to about 140 Mb of disk space for temporary files. The files for the final correct automatic structure have total size about 5 Mb.

The following description assumes some familiarity with the algorithm. The first step is to run the Knuth–Bendix process on the presentation until the number of word-differences arising from reduction equations appears to have become constant. This is probably the component of the algorithm that has most scope for improvement, since the last few word-differences seem to be very difficult to obtain. We stopped with 539 word-differences (or 629 when closed under inversion), at which point there were about 250,000 reduction equations, and the process was occupying about 100 Mb. In fact, we did not have the full set of word-differences at this point, which rendered the next few steps in the calculation more difficult. The word-acceptor W (calculated using all 629 word-differences) then had 8538 states. Using this and the word-differences to calculate the multiplier M resulted in M having 1,980,342 states initially, which minimized to 42808 states. It was this calculation that required the large temporary filespace for storing the original unminimized transition table for M (this table does not need to be held in core memory, but can be read in state by state during minimization).

The next step is a partial correctness test on M (we test whether, for each word u accepted by W and each generator a_i, there is a word v accepted by W such that (u, v) is accepted by M, where ua_i is equal to v in G). This test failed, and increased the size of the (inverse-closed) word-difference set to 653. We proceeded to recalculate W and A, which then had 8547 and 31021 states, respectively. The correctness test failed again, at which point we had 661 word-differences. This time, however, W had only 3251 states, and it turned out that W was correct at this stage. The number of states of M was then 863 871 before minimization and 25729 after minimization; the reduced sizes were due to the reduced size of W. The correctness test failed twice more, but W remained unchanged. The number of word-differences increased to 671, and the number of states of M to 25741. At this stage, the partial correctness test succeeded, and we could proceed to the full axiom-checking. This process took about 4.7 hours of CPU time and required about 105 Mb of core memory (the largest amount of memory used at any stage). The fact that all of the relations are short, having length two or three, rendered it more straightforward than usual, however. From the correct automatic structure, it was then possible to construct an automaton that accepts the minimal complete set of reduction rules, and this in turn could be used to find the correct minimal set of word-differences, of which there were 563. This is useful, since it can be used to make the word-reduction process in the group more efficient.

REFERENCES

[Epstein et al. 1991] D. F. Holt, D. B. A. Epstein and S. Rees, "The use of Knuth–Bendix methods to solve the word problem in automatic groups", *J. Symb.*

Comp. **12** (1991), 397–414.

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

[Havas et al. 1979] G. Havas, J. S. Richardson and L. S. Sterling, "The last of the Fibonacci groups", *Proc. Roy. Soc. Edinburgh* A**83** (1979), 199–203.

[Helling et al. 1994] H. Helling, A. C. Kim and J. L. Mennicke, "On Fibonacci groups", preprint, University of Bielefeld, Germany.

[Holt 1994] Derek F. Holt, "The Warwick Automatic Groups Software", Submitted to *Proceedings of DIMACS Conference on Computational Group Theory, Rutgers, March 1994.*

[Johnson 1980] D. L. Johnson, "*Topics in the Theory of Group Presentations*", London Math. Soc. Lecture Note Series **42**, Cambridge University Press, Cambridge, 1980.

[Newman 1990] M. F. Newman, "Proving a group infinite", *Arch. Math.* **54** (1990), 209–211.

Derek F. Holt, Mathematics Institute, University of Warwick, Coventry CV4 7AL, Great Britain (dfh@maths.warwick.ac.uk)

Received January 31, 1995; accepted May 8