

Non-automaticity:

A Look at automatic groups
using Cayley graphs and the KBMAG program

John Baker

Stacey Barbosa

Professor Bryan Mosher, Advisor

INTRO

The purpose of this paper is to investigate two issues in the theory of automatic groups. First, given an automatic group, we investigated how the size of its presentation relates to the size of the word acceptor for the automatic structure on that group. Second, we examined how geometric methods can be used to show that a group is not automatic. The methods used included: using previously discovered automatic and non-automatic groups, i.e. finite groups and Baumslag-Solitar groups respectively; using KBMAG, a program which uses the Knuth-Bendix method to attempt to create an automatic structure for a group; and using Cayley graphs to determine if the group fails the fellow-traveler property and is therefore non-automatic.

BACKGROUND

An integral part of understanding automatic groups is to understand the concept of a finite state automaton.

Definition: a finite state automaton “or simply automaton” is a quintuple (S, A, μ, Y, s_0) where S is a finite set, called the state set, A is a finite set, called the alphabet, $\mu: S \times A \rightarrow S$ is a function, called the transition function, Y is a (possibly empty) subset of S called the subset of accept states, and $s_0 \in S$ is called the start state or the initial state. (Epstein 199, p. 7).

In other words, you could think of an automaton as a spell checker: where A is the generating set of the 26 letters in the English language, S is the set generated by the concatenation of all these letters with a finite length, say limited to the longest word in the language, μ is the spell checker, Y is a set of states that indicates if the word is spelled correctly, and s_0 is the symbol that means, “start of a word.” It is important to note here that throughout this paper when the expressions word and language are used, they refer

not to spoken words or languages but combinations of generating elements (words) and the collection of these combinations (languages).

Finite state automata are an important part of the following definition of Automatic Groups:

Definition: Let G be a group. An automatic structure on G consists of a set A of semigroup generators of G , a finite state automaton W over A , and finite state automata M_x over (A, A) , for $x \in A \cup \{e\}$, satisfying the following conditions:

1. The map $\Pi: L(W) \rightarrow G$ is surjective.
2. For $x \in A \cup \{e\}$, we have $(w_1, w_2) \in L(M_x)$ if and only if $w_1x = w_2$ and both w_1 and w_2 are elements of $L(W)$.

We call W the word acceptor, M_e the equality recognizer and each M_x , for $x \in A$, a multiplier automaton for the automatic structure. An automatic group is one that admits an automatic structure. (Epstein 1992 p. 45).

In this definition $L(W)$ is the language accepted by W , M_e reads in two words and determines if they are equal or not and w is defined to be the group element represented by the word w (For example $xxXy$ reduces to xy).

Automatic groups are a recent development spurred by James Cannon's work on recursively defined Cayley graphs in 1984. Their importance lies in the fact that an automatic structure of a group yields the solution to "the word problem". This solution is an algorithm that takes as input a word over A and tells if the word is related to the identity.

The last piece of background information needed to understand our research is the concept of a Cayley graph. The Cayley graph is a physical representation of a group such that the vertex set consists of elements of G , with a directed edge labeled s going from g to $g * s$ for each $g \in G, s \in A$.

KBMAG PROGRAM

KBMAG stands for Knuth-Bendix on monoids and automatic groups. This program has tremendous capabilities, yet our familiarity with it was fairly limited. For a more detailed explanation of the functions of this program, see Holt (1998). For the purposes of our research, we input the generators and inverses of a two-generator, one-relator group, namely x, x^{-1}, y, y^{-1} , in short-lex order. This means that $v < w$ if and only if v is shorter than w , or they have the same length and v comes before w in lexicographical order. (Epstein 1992 p.56). (Hereafter x^{-1} and y^{-1} will be written as X and Y respectively). In addition, the relator word was entered. Once this information was entered, we ran a program called “autgroup”. This program attempts to create an automatic structure for the group using the Knuth-Bendix method.

This method creates a finite set of equations, E , which is pairs of words that are equivalent. If we input $xxYxy$ as the identity word, the program would create such equations as $xxY=Yx$, $xYx=XY$, and so on, until all of the possible related words are accounted for. The program then takes a word and looks it up in this set to see if it can be written as a shorter word. This is how the program generates the word acceptor. It determines which words are in shortest form, and which ones can be reduced.

Provided the program terminated successfully, we were then able to view the word acceptor for the automatic group. The word acceptor for the relator $xxyy$ was given as:

x, X, y, Y
[[2,3,4,5],
[6,0,4,5],
[0,3,7,0],
[8,9,0,0],
[10,0,0,0],

[6,0,0,5],
[0,9,0,0],
[0,0,4,5],
[0,0,7,0],
[0,0,0,5]].

The first row of the matrix is the start state, row i of the matrix represents the i 'th state of the word acceptor, and the j 'th entry per row indicates what row to go to next. For example, if we have the word xyx , the you start at row one, go to the x entry which is 2, go to the second row, go to the y entry which is 4, go to the fourth row and go to the x entry which is 8. This means that the word xyx ends at state 8, and will therefore act like all other words at state 8 by multiplication on the right. Once we obtained word acceptors for numerous identity words of lengths varying from two to nine, we were able to compare word acceptors and to analyze the different structure of each word acceptor. Some initial results follow.

RESULTS

We first studied the word acceptor for a four generator, one relator group: the fundamental group of a genus-two topological surface. We looked at what would cause a word to end in a fail state (when the stage number is 0). We also looked what the state numbers meant, and how different words of very different length could end in the same state. From here we tried to find all of the automatic structures for two generator groups with one relator, relator length from one to four. When the relator is of length one, then the group is isomorphic to the group of all integers under addition. This is because if x is equal to the identity, then so is its inverse, X . Then there is really only one generator, y , which would act as the generator for the integer group, one.

Finding the word acceptor automaton for all of the groups with relators of length one through four was a daunting task. So first we reduced the number of relator words by removing the words that were not already reduced, since once they were to be reduced the word would be of shorter length. With those out of the way, we also found eight simple isomorphisms that greatly condensed the number of reduced words to find. These are:

0. $x \rightarrow x, X \rightarrow X, y \rightarrow y, Y \rightarrow Y$
1. $x \rightarrow x, X \rightarrow X, y \rightarrow Y, Y \rightarrow y$
2. $x \rightarrow X, X \rightarrow x, y \rightarrow y, Y \rightarrow Y$
3. $x \rightarrow X, X \rightarrow x, y \rightarrow Y, Y \rightarrow y$
4. $x \rightarrow y, X \rightarrow Y, y \rightarrow x, Y \rightarrow X$
5. $x \rightarrow y, X \rightarrow Y, y \rightarrow X, Y \rightarrow x$
6. $x \rightarrow Y, X \rightarrow y, y \rightarrow x, Y \rightarrow X$
7. $x \rightarrow Y, X \rightarrow y, y \rightarrow X, Y \rightarrow x$

Here is an example to illustrate how we used these isomorphisms:

$$xyXY \xleftarrow{1} xYXy \xleftarrow{2} XyXy \xleftarrow{3} XYxy \xleftarrow{4} yxYX \xleftarrow{5} yXYx \xleftarrow{6} YxyX \xleftarrow{7} Yxyx$$

Based on this now reduced set of relator words we found the automatic structure for every group of relator length one to four. This data obtained is as follows:

Relator	Number of States	Number of Zeros in word acceptor	Average Number of Zeros per state
x	3	8	2.667
xx	4	7	1.750
xy	3	8	2.667
xxx	4	4	1.000
xyy	3	6	2.000

xyx	3	6	2.000
xyy	3	6	2.000
xyX	3	8	2.667
xxxx	5	5	1.000
xxxxy	4	7	1.750
xxyx	4	7	1.750
xyxx	4	7	1.750
xyyy	4	7	1.750
xxyy	10	21	2.100
xyyx	10	21	2.100
xyxy	5	6	1.200
xxyX	3	8	2.667
xyXX	3	8	2.667
xyxY	5	8	1.600
xyXy	5	8	1.600
xyXy	5	8	1.600
xyyX	4	7	1.750

From this data, we discovered another isomorphism, that of cyclic permutation. For example the group with relator xxy is isomorphic to the group with xyx and the group with yxx. It is important to note, though, that isomorphic groups have similar, but not always the same, automatic structure. An automatic structure depends on the choice of presentation and even the order of the generators. It is also true that groups with the same automatic structure might not be isomorphic.

Once we compiled this initial data, we started looking for ways of classifying these groups. We ended up looking at the number of fail states, or zeros, in the automatic structure. We compiled the total number of zeros in the structure and then divided by the number of states created by the word acceptor. It seemed that the groups with high zero to row ratios were less complex. In other words, the more fail states that appear in the group, the smaller the number of pathways there are to create longer and longer words. It was from looking at patterns like these in the smaller relator groups that we started to look at groups with longer relator words. Because of the sheer number of relators

yielding non-isomorphic groups, we looked only at specific words to try to find patterns.

We did research on groups with length five, six, seven, and eight relators. Something

interesting came out of this data:

Relator	Number of States	Total Number of Zeros in Word Acceptor	Average Number of Zeros per State
xxxxx	6	6	1.000
xxxxy	5	8	1.600
xxxyy	20	40	2.000
xyxy	6	12	2.000
xyXYx	N/A	N/A	
xxxxxx	7	7	1.000
xxxxxy	6	9	1.500
xxxxyy	25	45	1.800
xxxyyy	26	45	1.731
xyxyxy	10	13	1.300
xyxyy	N/A	N/A	
xyXYxx	N/A	N/A	
xyXYxy	N/A	N/A	
yxyXYx	N/A	N/A	
xxxxxxx	8	8	1.000
xxxyyyy	46	76	1.652
xyxyxyx	10	20	2.000
xxxxyyx	48	77	1.604
xxxxyxx	32	56	1.750
xxxxxy	7	10	1.429
xYXxyXYx	N/A	N/A	
xyxyXYX	N/A	N/A	
xyXYxyX	N/A	N/A	
xyxyXXY	N/A	N/A	
xxxxxxxx	9	9	1.000
xyyxyxy	25	32	1.280
xxxxyyyy	56	84	1.500
xyxyXXYY	39	50	1.282
xxxxyyxx	58	86	1.483
xxxxxyX	6	9	1.500
xxxxxyy	39	68	1.744
xyxyxyx	37	77	2.081
xyXYxyXY	13	16	1.231
xyxyxyxy	9	10	1.111
XYXYxyyx	5	8	1.600
xyyXXYx	77	104	1.351

As you can see we moved from simple two generator relators to using ones that included the relators' inverses. As the generators used more of a variety of letters— x , y , X , and Y , instead of just x and y —the automatic structure of the groups became larger. This is true of all, except those groups which are isomorphic to groups which smaller relators (i.e., by cyclic permutation, $XYXYxyyx$ is isomorphic to $XYxy$).

An interesting thing to notice is where N/A is indicated in the number of states column. This is where the program came up with no automatic structure. There were four ways the program would end, and only one produced a proper automatic structure. All the groups listed above with the N/A ended in two ways. Either the program would stall on a certain screen and would not move forward, or it would print out a message that said that the set E of equations mentioned earlier was too large and based on this the program could not compile an automatic structure. The fourth end state of the program was that it would find a word acceptor for the group, but the word acceptor could not be accepted by the axioms program. This is the program that checks if the automatic structure created by the computer is really automatic (by using theorem 4.1 from the 1991 paper by Epstein). The program would instead continue running the axioms program forever. Some examples of group relators that produced this result, not listed above, are $xyyyxyyx$ and $xYXyxyXY$.

The main point here is that the program does not always find an automatic structure. We can conclude that if the program produces an automatic structure that is accepted by the axioms program, then the group is automatic. But if an automatic structure is not found, that says nothing of the automaticity of the groupⁱ. With this in mind, we then shifted our project towards looking at groups that might not be automatic.

NON-AUTOMATIC GROUPS

Based on the groups we already researched, we began to look at groups similar to these. One class of relators that was very similar to the Baumslag-Solitar relator (see below) was of the form wx^pWx^q where $p \neq \pm q$ and w can be a word made up of an infinite amount of different generators as long as W is its inverse. There was also the class of relators that were simply: $xyy(xy)^n$, $n \geq 1$. The program could not find an automatic structure.

Based on this information, we decided to look at other ways determining whether a group is non-automatic. we first looked at groups that were proven non-automatic. It was at this time we came upon the work of Baumslag and Solitar. They used a geometric method of classifying groups using a Cayley graph. They worked with a group already proven in a more direct way non-automatic by the work of Thurston. Still, they revealed the non-automaticity of the Baumslag-Solitar group by proving that the Cayley graph of this group, $\{x, y \mid yx^pYx^q\}$ where $p \neq \pm q$, fails the fellow traveler or Lipschitz property. The simple definition of this property is that if two different words are equivalent in a group, then their paths in the Cayley graph to that final destination cannot get too far apart. If a metric is used in the Cayley graph, say one equals the distance each generator spans in the graph, then the property states that they should get no farther than $2c - 1$ lengths apart, where c is a constant which stands for maximum number of states over all M_x . A final way of describing this property is if the two words move linearly from the beginning of the word, where they start together, to the end, where they meet again, then the distance between the words should also change linearly. In the case of the Baumslag-Solitar group, it changed exponentially.

This relates to our research because we tried to look at the Cayley graphs of groups that failed the KBMAG program. The group that was similar to the Baumslag-Solitar group had a similar proof of non-automaticity. Unfortunately, when the relator words become longer, it becomes more difficult to sketch the Cayley graph. And although we may be able to prove certain smaller groups are not automatic with simple graphs, the algorithm for solving the automatic problem remains. Our method was to use the KBMAG program to see if an automatic structure emerges, and if one does not then employ a geometric method.

CONCLUSION

Automatic groups are important in part because they give a solution to the word problem. The data found in our research shows a relationship, though not patterned, between the length of the relator word and the size of the word acceptor. We did find a way to organize the “complexity” of an automatic structure by determining the number of fail states present in the word acceptor. We also compiled various methods of determining whether a group is automatic or not. This method was simply using the KBMAG program to initially determine if a group is automatic, and if the program failed, trying to disprove the fellow traveler property with the Cayley graph of the group. But even this combination of methods has still left a large number of groups for which the automaticity is unknown.

Bibliography

Epstien, D. B. A., Cannon, J. W., et al., Word Processing in Groups. Jones and Bartlett Pub: Boston 1992.

Epstien, D. B. A., Holt, D. F., and Rees, S.E., “The Use of Knuth-Bendix Methods to Solve the Word Problem in Automatic Groups.” *J. Symbolic Computation* vol. 12 (1991): 397-411.

Farb, Benson, “Automatic Groups: A guided Tour.” *L’Enseignement Mathematiques* vol. 38 (1992): 291 – 313.

¹ It is entirely possible that the fourth exit from the program means that the group is not automatic, but we were unable to find the information to back that up