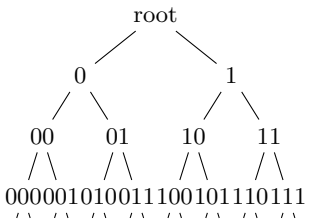
A short introduction to self-similar groups

Murray Elder*

Abstract

Self-similar groups are a fascinating area of current research. Here we give a short, and hopefully accessible, introduction to them.

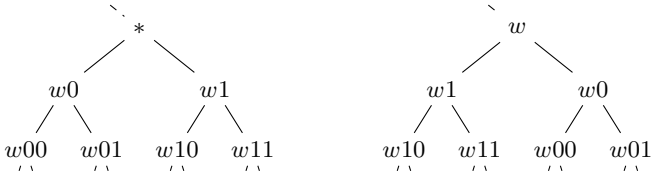
1. Introduction



The figure above shows (part of) the *infinite rooted binary tree*, T. The *root* is the node (or vertex) at the top, and each node has exactly two nodes below it joined by an edge. It goes on forever down the page. We have labelled each node with a binary number in a systematic way—if w is a binary string labelling a node, then the two nodes below it joined by an edge are labelled $w0$ and $w1$.

An *automorphism* of T is a bijective map from the nodes to the nodes which preserves adjacencies, meaning if two nodes are joined by an edge, then the nodes they map to are again joined by an edge. This definition works for any graph, but we'll stick with T for now.

Before we give an example, here is a convention which will help describe automorphisms of T. Drawing $*$ at node labelled by w (some binary string) means exchange the subtree with root $w0$ and the subtree with root $w1$, as indicated here:



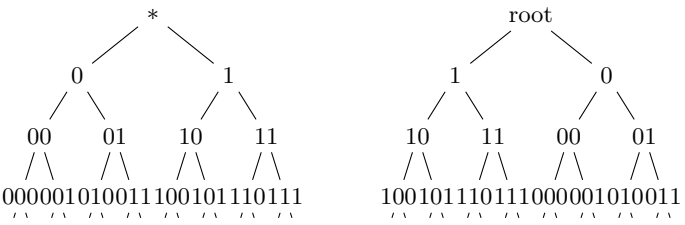
*School of Mathematical and Physical Sciences, The University of Newcastle, Callaghan, NSW 2308. Email: Murray.Elder@newcastle.edu.au

Invited technical paper, communicated by Jon Borwein.

The author's research is supported by the Australian Research Council.

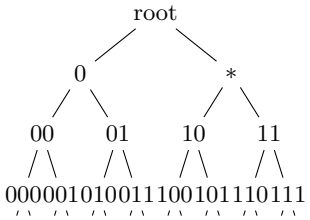
Once the move is performed we remove the $*$, and we can verify that with this definition, if an automorphism has several $*$s then it doesn't matter in which order they are performed.

Now for an example. Define a to be the automorphism of T which fixes the root, sends nodes labelled $0w$ to $1w$, and nodes labelled $1w$ to $0w$, where w is any binary string (possibly empty). The tree on the right shows what T looks like after a is applied.



Applying a twice puts T back as it started, so we say that aa is the same as the map that does nothing to the tree. We call the map which leaves the tree unchanged the *identity*, and denote it by the letter e. Note that if a and b are automorphisms of T, the notation ab means apply a first then b.

For example, if b is the automorphism given by



then the reader can check that ab sends the node labelled 00 to position 11 while ba sends it to 10.

The representation of an automorphism of T by T decorated by $*$s is called a *portrait* of the automorphism. Note that every automorphism of T can be expressed using this notation (possibly with infinitely many $*$s).

The *inverse* of an automorphism x is an automorphism y such that $xy = e(= yx)$. Since automorphisms are bijective maps, they have inverses.

If G is a set of automorphisms of T and their inverses, such that for each $x, y \in G$ the products xy and yx are also in G, then the algebraic object we obtain is called a *group*[1].

[1]Groups are not just sets of automorphisms of T — they can be the configurations of a Rubik's cube, automorphisms of graphs other than T, *braids*, and maps of the real line to itself. A *group* is just a set with a *multiplication* defined on it, so that products of things in the set are also in the set, such that the multiplication is associative, it has some *identity* (like e) and each element has an *inverse* (each x has a y so that $xy = e$).

A good way to ensure the property that products of G stay in G is to take a set of automorphisms, say a and b, their inverses (which in this case are the same), and let G be the set of all finite products of these automorphisms. In this case we say G is *generated* by the set $\{a, b\}$, and a and b are the *generators*. Whenever a group is generated by a *finite* set of elements (automorphisms), we call it a *finitely generated group*.

Here is another way to define automorphisms. Let w be a binary string. The map a sends the node labelled $0w$ to position $1w$, and the node labelled $1w$ to $0w$, so we can describe it using the following rules:

$$a(0w) = 1.e(w), \qquad a(1w) = 0.e(w),$$

where $e(w)$ means apply the identity map (do nothing) to the suffix w. More interesting are the rules describing b:

$$b(0w) = 0.e(w), \qquad b(1w) = 0.a(w).$$

The first rule just says that nodes on the left subtree of the root are not changed, but the second rule says if a node label starts with 1, apply a to the suffix of the label. Note that the rules for b uses a and e, and the rules for a only uses e, so the set $\{a, b, e\}$ of automorphisms can be described by a *self-referencing* or *self-similar* set of rules. Products of a and b can also be expressed with rules of this form, for example $ab(0w) = b(1.e(w)) = 1.(ea(w))$ (i.e. apply e first to w then a), and $ba(0w) = a(0.e(w)) = 1.ee(w)$.

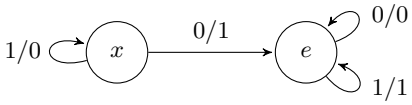
Definition 1.1. Let G be a group of automorphisms of T. Then G is a *self-similar group* if for each $g \in G$, each $x \in \{0, 1\}$, and each binary string w, there is a $y \in \{0, 1\}$ and a $h \in G$ such that

$$g(xw) = y.h(w).$$

See [6] for more details. Note that the definition easily extends to groups of automorphisms of rooted n-ary trees, but again we will stick with binary trees for this paper.

2. Automata

Another way to describe automorphisms of T is by an *automaton*. Here is an example:



This automaton has two *states* labelled e and x. If we start at the state x and read the string 010, we follow the edge labelled 0/1, replacing the first letter of the string by 1, then from the state e we follow the edge 1/1, replacing the second letter 1 by 1, then (since we are still in the state e) we follow 0/0 and keep the third letter as 0. So the edge label tells us how to rewrite the next letter of the

string, and the state tells us what to do with the suffix of the string. If we start at the state e and read a string, the string stays the same.

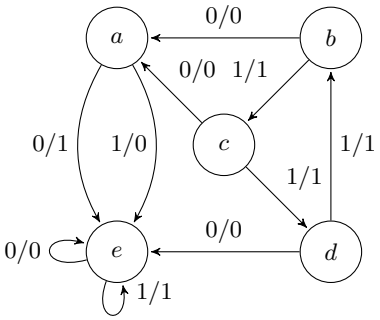
Exercise 2.1. Draw the portrait of the automorphism x in this example[2].

Exercise 2.2. Draw the automaton encoding the rules for a and b in the previous section.

Exercise 2.3. If you know some basic group theory, do you recognise the group generated by $\{x\}$[3] and the group generated by $\{a, b\}$?

3. Grigorchuk's group

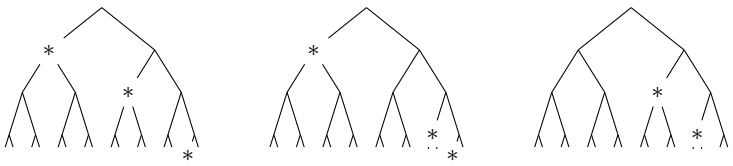
Here is an example which really kicked off the theory of self-similar (or automaton) groups. We start with the automaton describing the self-similar rules for five actions.



If we start at state a, we switch the first letter of the string, then move to state e for the rest of the string. So a is the same action as described at the start (while b isn't, since it sends 00 to 01).

Exercise 3.1. Write the self-similar rules for a, b, c, d by reading off the automaton.

Here are portraits of b, c and d (the period 3 pattern keeps going all the way down).



Exercise 3.2. Using the portraits, or otherwise, work out what bb, cc, dd and bc do. Which automorphisms are they the same as?

[2]Solutions for all the exercises can be found at www.austms.org.au/gazette.
[3]Hint: think of x^n acting on w as doing something to a binary number (written backwards).

The Grigorchuk group is the self-similar group generated by a, b, c, d. If you did the exercises right, you would have found some *relations* between letters. Each generator done twice is e, and bc is the same as d. It follows from these facts that any product can be written more *efficiently* by never writing aa and never putting two letters b, c, d next to each other (since $bc = d$, $bd = c$, $cd = b$). That is, every product in the group can be reduced to something of the form $ax_1ax_2\ldots$ or $x_1ax_2a\ldots$ where $x_i = b, c$ or d.

Exercise 3.3. Show that $adad$ is the same automorphism as $dada$.

A finitely generated group G is said to be *finitely presented* if a finite number of relations, say $u_1 = v_1, \ldots, u_n = v_n$ where u_i and v_i are finite products of generators (or inverses of generators), suffice to show equality between arbitrary products of generators. Even though we have found a few relations that the generators a, b, c, d for Grigorchuk's group satisfy, like $aa = bb = cc = dd = e, bc = d, adad = dada$, it is known that Grigorchuck's group is not finitely presented.

4. Word problem

The *word problem* for a finitely generated group is the following question: given a word (or finite product) of generators, is the product equal to the identity element or not? In the case of groups of automorphisms of T, this is the same as asking if a product of generators puts T back in its original configuration. For example, $adadadadad$ does nothing to the tree, so the answer on this input is *yes*[4].

Here is an algorithm (given by Grigorchuk) to solve the word problem for his group. Write the input word in the form $ax_1ax_2\ldots$ or $x_1ax_2a\ldots$ where $x_i \in \{b, c, d\}$. Count the number of a letters. If it is odd, we know that the automorphism it represents switches the nodes 0 and 1, so this word is not the identity.

So suppose the number of a letters is even. If the word starts with a, write it as $(ax_1a)x_2(ax_3a)x_4\ldots$, and otherwise write it as $x_1(ax_2a)x_3(ax_4a)\ldots$. Now we know the word does not switch the two nodes at level 1. What does it do to the subtree hanging from the node 0? The subword (aba) has the effect of doing what c does to the subtree (check this — a moves the subtree over to the right side, then b acts by switching down the right side of the subtree, then a puts it back). Similarly we can work out that (aca) acts like d on the subtree, and (ada) acts like b. In a very similar way, we can work out what b, c and d do to the subtree — b and c flip it (so act like a) and d does nothing to it.

So to work out what the input word does to the subtree hanging from 0, we *rewrite* $(ax_1a)x_2(ax_3a)x_4\ldots$ or $x_1(ax_2a)x_3(ax_4a)\ldots$ by replacing b and c by a, and d by e, and aba, aca, ada by c, d, b respectively. It's a similar story for the subtree hanging from node 1.

Exercise 4.1. Work out the replacement rules for b, c, d, aba, aca, ada for the right subtree.

[4]This follows from Exercise 3.3 — asking if $u = v$ in a group is the same as asking if $uv^{-1} = e$.

We should now be able to see how this will turn into a recursive algorithm — given a subtree and a word in a, b, c, d acting on it, count the number of as, and if it is even, see what happens to the two subtrees (under two rewritten (and shorter) words).

A good exercise is to figure is out the worst-case time (and space) complexity of the algorithm. A variation of this algorithm works for a large class of self-similar groups; see [6].

In general, the word problem for an arbitrary finitely generated group is an *undecidable* problem — there are even finitely *presented* groups for which, if we could decide if a given word is the same as the identity, then we could solve the *Halting problem* for Turing machines, which is unsolvable (see for example [5] for more details).

5. Growth

Grigorchuk's group is famous because it was the first example of a group having *intermediate growth*. For a group G generated by a finite set of elements, define the *growth function* $f\colon \mathbb{N} \to \mathbb{N}$ by $f(n)$ where $f(n)$ is the number of elements of G that are equal to some product of generators of length at most n; the maximum value this function could attain is exponential in the number of generators, since there are k^n strings of k letters of length n. Milnor asked if a group could have a growth function that is superpolynomial, but subexponential (like $f(n) = e^{\sqrt{n}}$), which is called *intermediate*, and Grigorchuk showed that his group has an intermediate growth function. Two excellent sources in which to read accounts of this are [3] and [4].

Open question 5.1. Is there a *finitely presented* group of intermediate growth?

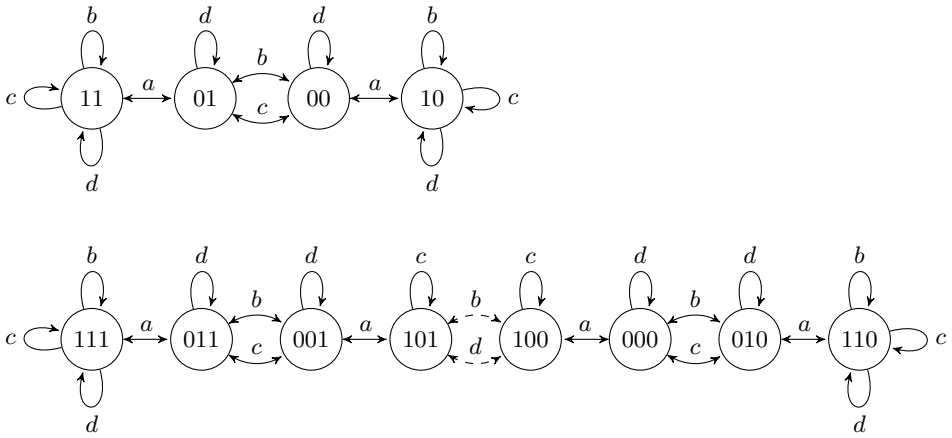
6. Schreier graphs

Whenever we have a group G generated by a finite set $\mathcal{G}$, acting on a set X, and M is some subset of X, there is a useful device called the *Schreier graph*, which is defined as follows. A good example to keep in mind while reading this definition is to take X as the nodes of T, M as the set of nodes at some fixed level k of T, and G as a self-similar group acting on T.

For each element of M draw a node labelled by this element. Connect nodes m_i, m_j by a directed edge labelled $s \in \mathcal{G}$ whenever $m_j = sm_i$. Note that this graph is *connected* if for any two points in M there is always some group element (which can be expressed as a finite product of generators from $\mathcal{G}$) which takes the point m_i to m_j. If this is satisfied, we say that G acts *transitively* on M.

For example, if G is Grigorchuk's group, $X = T$, and M is the set of nodes at level k, then the action of G is transitive on M since we can find combinations of a, b, c, d which move any point to any other in this level.

Here are the Schreier graphs for Grigorchuk's group acting on levels 2 and 3:



To get the graph for the next level, we make two copies of the graph for the previous level, append 1 to the nodes in one copy and 0 to the other, flip the 0 copy then glue them together. The dashed lines indicate where gluing occurs to get level 3. In this way we see some more *self-similarity*.

Exercise 6.1. Draw the Schreier graph for level 4.

More generally, given any group G with generating set $\mathcal{G}$, and any subgroup H, the set of left cosets G/H is a set on which G acts transitively, so we can form Schreier graphs for G acting on G/H. If H is the trivial subgroup, then the Schreier graph coincides with another standard construction in group theory: the *Cayley graph*. If G is a self-similar group acting transitively on each level of T, let H be the subgroup of G containing all elements which fix a node at level k. Then the Schreier graph for G acting on G/H is the same as the graph for G acting on T when M is the set of nodes at level k.

7. Geodesics

Each element of a finitely generated group is the product of some number of generators. A product of generators is a *geodesic* if there is no shorter product that equals the same element. For example bcd is not a geodesic in the Grigorchuk group, but $adad$ is[5].

If G is a group with finite generating set $\mathcal{G}$, the *geodesic growth function* for G with respect to $\mathcal{G}$ counts the number of *geodesics* of length (at most) n. It is clear that this function is bounded below by the usual growth rate (since each element has at least one geodesic representing it), and bounded above by an exponential function (since there are k^n strings of k letters of length n).

[5]Don't believe me? Run the word problem algorithm on $adadu^{-1}$ for all words u of length at most three.

In [1], groups with polynomial geodesic growth functions are considered, and a natural extension to Milnor's question arises:

Open question 7.1. Is there a group of intermediate *geodesic* growth?

The first example to try is Grigorchuk's group, since the number of geodesics of length n is at least the number of elements, so its geodesic growth is superpolynomial.

We can use the Schreier graphs of the Grigorchuk group to find geodesics as follows. A word of the form $u = ax_1ax_2 \dots ax_{n/2}$ labelling a path starting from the node labelled $11\dots1$ and moving right (ending at a node we will call k) encodes an automorphism that sends $11\dots1$ to k in T. Suppose v is a product of a, b, c, d that also sends $11\dots1$ to k. Then v labels a path in the Schreier graph, starting at $11\dots1$ and ending at k. Since the Schreier graph describes all possible ways of moving between nodes at a fixed level of the tree, if v does not travel in a straight line in this graph (like u does) it has no hope of sending $11\dots1$ to k. In other words, no word shorter than u can be the same group element as u. For each x_i we have two choices for b, c, d, so there are $2^{n/2} = (\sqrt{2})^n$ different words of length n like this, so the geodesic growth function is exponential. More details can be found in [2].

8. Example: the basilica group

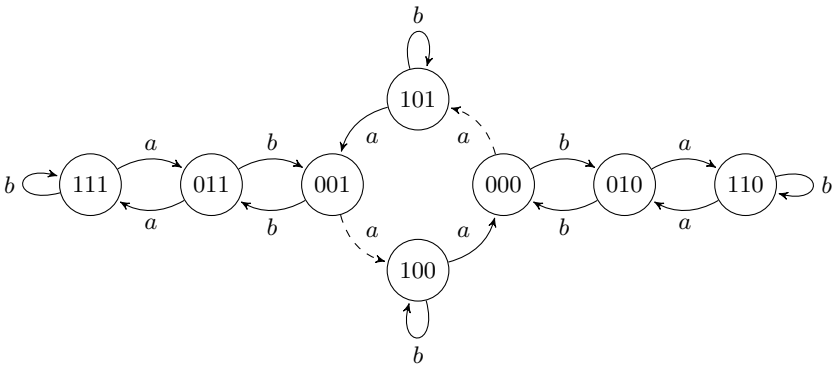
Here is one more self-similar group. Let B be the group acting on T, generated by two automorphisms a, b described by these self-similar rules:

$$a(0w) = 1b(w), \qquad a(1w) = 0e(w), \qquad b(0w) = 0a(w), \qquad b(1w) = 1e(w).$$

Exercise 8.1. Draw an automaton (with states labelled a, b, e) which encodes these rules.

Exercise 8.2. Draw the Schreier graph of the action of B on level 2 of T.

Here is the Schreier graph for level 3, which should look like two copies of the Schreier graph for level 2, stuck together by breaking some edges and reconnecting.



Exercise 8.3. Draw the Schreier graph of the action of B for levels $4, 5, \ldots$ of T. What do you see?

I hope this short introduction might inspire some readers to explore the topic further. Some excellent starting points are [3], [4] and [6].

Acknowledgements

I am extremely grateful to Nick Davis, Slava Grigorchuk, Dima Kravchek, Zoran Šunić and Mike Whittaker who have taught me much about the subject.

References

[1] Bridson, M., Burillo, J., Elder, M. and Šunić, Z. (2012). On groups whose geodesic growth is polynomial. *Internat. J. Algebra Comput.* **22** (to appear).
[2] Elder, M., Gutiérrez, M. and Šunić, Z. Geodesics in the first Grigorchuk group. In preparation.
[3] Grigorchuk, R. and Pak, I. (2008). Groups of intermediate growth: an introduction. *Enseign. Math.* **54,** 251–272.
[4] Mann, A. (2012). *How Groups Grow* (London Mathematical Society Lecture Note Series **395**). Cambridge University Press.
[5] Miller III, C.F. (1992). Decision problems for groups—survey and reflections. In *Algorithms and Classification in Combinatorial Group Theory*, eds Baumslag, G. and Miller III, C.F. (Math. Sci. Res. Inst. Publ. **23**), Springer, New York, pp. 1–59.
[6] Nekrashevych, V. (2005). *Self-Similar Groups* (Mathematical Surveys and Monographs **117**). American Mathematical Society, Providence, RI.

Murray Elder is an ARC Future Fellow at The University of Newcastle. He did a PhD at The University of Melbourne working with Walter Neumann on automatic groups and geometric group theory, then took several postdocs and lecturing positions in the US and Scotland, before returning to Australia in 2008 when he secured a temporary position at The University of Queensland. His research interests include formal languages and automata, geometric group theory, computation and experimental mathematics, complexity and computability, and enumerative combinatorics.