Self-similar groups and their applications
ICMU Minicourse Lecture Notes

Dmytro Savchuk
University of South Florida

June 19, 2026

Contents

Introduction

These notes serve as a basic introduction to the area of self-similar groups (or groups generated by automata) that lies at the intersection of the geometric and combinatorial group theory. These groups have been studied extensively since 1980's, when it was realized that some of them, while easily definable, possess very unusual properties like being infinite finitely generated torsion groups, or being of intermediate growth. As of today, this is a reach theory with connections to topology, dynamics, combinatorics, analysis, cryptography, and computer science.

1 Trees

We will study groups that (in most cases) act on regular rooted trees.

Definition 1.1. A connected graph with no cycles is called a *tree*. A tree T is called *rooted* if one its vertices, called the *root* is selected.

As all graphs, trees are endowed with a combinatorial metric in which the distance between two vertices is equal to the number of edges in the shortest path connecting them.

Definition 1.2. The *n-th level* of the tree T is the set of vertices whose distance from the root is n.

Since the tree has no cycles, for each vertex v of level n there exists exactly one path connecting v to the root. The vertex v' in this path that lies in level $n-1$ is called the *parent* of v. The vertex v in this case is called a *child* of v'. Thus every vertex except the root has exactly one parent and may have several children.

Definition 1.3. A rooted tree T is called *regular* (or d-regular, or d-ary) if there is $d \in \mathbb{Z}_+$ such that each vertex of T has exactly d children. In case when $d = 2$ such a tree is called *binary*.

Historically, the rooted trees in the context of groups acting on them grow from top to bottom, so that the root is the highest vertex and the children of each vertex v are located right under v. The example of a binary tree is shown in Figure 1. The standard notation for the d-ary tree that we will use throughout the text is T_d.

Remark 1.4.

- *A regular rooted tree is not a regular graph as the degree of each vertex except the root is $d+1$, while the degree of the root is d.*
- *A regular rooted tree is always infinite and has infinite number of levels.*

We will be interested in groups that act on regular rooted trees by graph automorphisms, i.e. by maps from the set of vertices to itself preserving the adjacency relation. Since the degree of the root is different from the degrees of all other vertices, every such automorphisms must map the root to itself. More generally, the levels of the tree must be invariant under the action of each such automorphism because it preserves the distance from the root.

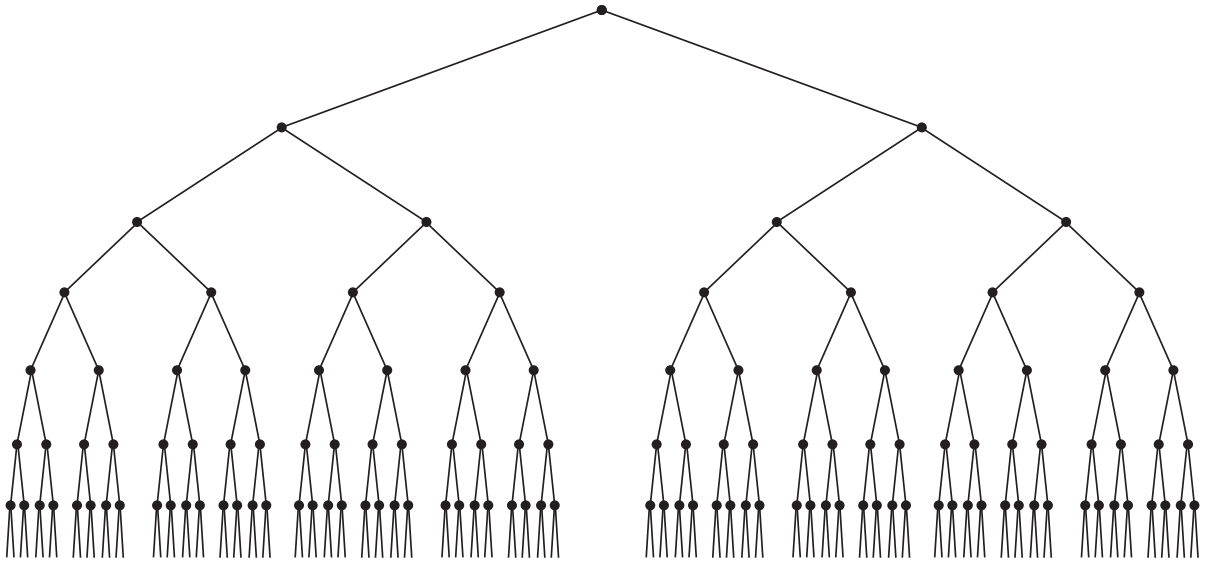


Figure 1: Binary tree

Definition 1.5. The group of all graph automorphisms of the regular rooted tree T_d with the operation of composition is called the *full group of automorphisms of* T_d and is denoted $\operatorname{Aut} T_d$.

The first trivial observation about $\operatorname{Aut} T_d$ is:

Exercise 1.6. *Prove that* $\operatorname{Aut} T_d$ *is uncountable for* $d \geq 2$.

But in most cases we will be interested not in the full group of automorphisms, but in its finitely generated subgroups. In order to describe some of them, we have to come up with some "coordinate system" on T_d. Namely, we will identify vertices of T_d with finite words over the alphabet consisting of d letters.

Let X be a finite set of cardinality d and let X^* denote the set of all finite words over X (that can be though as the free monoid generated by X). This set can be naturally endowed with a structure of a rooted d-ary tree by declaring that v is adjacent to vx for any $v \in X^*$ and $x \in X$. The empty word corresponds to the root of the tree T_d and X^n corresponds to the n-th level of the tree. With this interpretation in mind we will often write $\operatorname{Aut} X^*$ for $\operatorname{Aut} T_d$, meaning that we consider automorphisms of X^* as a tree, and not as a free monoid on X.

One of the convenient languages to describe automorphisms of $\operatorname{Aut} X^*$ is the language of Mealy automata introduced in the next section. Later on we will also talk about the language of portraits and the language of wreath recursion.

2 Mealy automata

We start from the informal description of what automaton is and how it acts on finite words over X. There are many different types of automata in mathematics and in computer science:

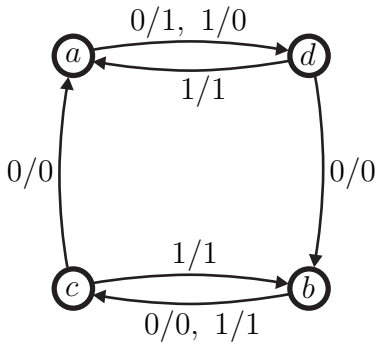


Figure 2: Example of a Mealy automaton

automata acceptors, cellular automata, Moore machines, Turing machines, etc. We will word with the class of deterministic automata transducers (or Mealy automata).

Informally, each *noninitial Mealy automaton* (or simply *noninitial automaton*) is defined by a diagram (called *Moore diagram*) with some nodes (called the *states*) and the arrows connecting the states that are labelled by the pairs of elements of X (conventionally, a pair $(x, y) \in X^2$ is written as $x|y$). The states are often labelled by their names. An example of such diagram is shown in Figure 2.

A noninitial automaton $\mathcal{A}$ does not act on finite words over X, but as soon as we select a state q of $\mathcal{A}$, we get the *initial Mealy automaton* (or simply *initial automaton*) $\mathcal{A}_q$. This is the object that acts on the set of finite words in the following way.

Given a finite word $w = x_1 x_2 \dots x_n$ over X, the automaton $\mathcal{A}_q$ performs the following operations:

1. Reads the first letter x_1 of w.
2. Finds the arrow from the initial state q with label $x_1|y_1$ for some $y_1 \in X$ to some (possibly the same) state q'.
3. Outputs y_1 as the first letter of the output word.
4. Changes its current state q' to the terminal state of the edge E.
5. Handles the rest of the word in the same fashion by the initial automaton $\mathcal{A}_{q'}$.
6. When all the letters of w are read, declares $y_1 y_2 \dots y_n$ to be the output word.

Remark 2.1. *In this way each state (under the extra condition below) defines a transformation* $\mathcal{A}_q \colon X^* \to X^*$.

- *In order for this transformation to be well-defined, for each state q of A and each input letter $x \in X$ there must be exactly one arrow initiating from q and whose label is of the form $x|*$. Otherwise the automaton will not know what to output or what state to go to after reading an input letter. Such Mealy automata are called* deterministic.
- *It is not always the case that this transformation is invertible.*

For example, the following table shows how the state a of an automaton shown in Figure 2 acts on an input word 110010100. The table has 3 rows: the first one contains the input word, the second one shows the states in which the automaton is in while reading corresponding letter of the input word, and the third row shows the output word.

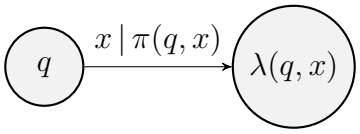
$$\begin{array}{lccccccccc} \text{Input:} & & 1 & 1 & 0 & 0 & 1 & 0 & 1 & \\ \text{States:} & a & \to & d & \to & a & \to & d & \to & b & \to & c & \to & a & \to & d \\ \text{Output:} & & 0 & 1 & 1 & 0 & 1 & 0 & 0 & \end{array} \tag{1}$$

Exercise 2.2. *Compute the image of a word* 001100110 *under the action of* $\mathcal{A}_b$ *from Figure 2.*

With this intuitive understanding in mind, we can now formulate the rigorous (and somewhat technical at the first glance) definition of a Mealy automaton.

Definition 2.3. A *Mealy automaton* (or simply *automaton*) is a tuple (Q, X, π, λ), where Q is a set (a set of *states*), X is a finite alphabet, $\pi\colon Q \times X \to Q$ is the *transition function* and $\lambda\colon Q \times X \to X$ is the *output function*. If the set of states Q is finite the automaton is called *finite*. If for every state $q \in Q$ the output function $\lambda(q, x)$ induces a permutation of X, the automaton $\mathcal{A}$ is called *invertible*. Selecting a state $q \in Q$ produces an *initial automaton* $\mathcal{A}_q$.

As mentioned earlier, automata are often represented by the *Moore diagrams*. The Moore diagram of an automaton $\mathcal{A} = (Q, X, \pi, \lambda)$ is a directed graph in which the vertices are the states from Q and the edges have form



for $q \in Q$ and $x \in X$. If the automaton is invertible, then it is common to label vertices of the Moore diagram by the permutation $\lambda(q, \cdot)$ and leave just first components from the labels of the edges.

The action of an initial automaton $\mathcal{A}_q$ on X^* is formally defines as follows. Given a word $v = x_1x_2x_3 \dots x_n \in X^*$ it scans its first letter x_1 and outputs $\lambda(q, x_1)$. The rest of the word is handled in a similar fashion by the initial automaton $\mathcal{A}_{\pi(q,x_1)}$. Formally speaking, the functions π and λ can be extended to $\pi\colon Q \times X^* \to Q$ and $\lambda\colon Q \times X^* \to X^*$ via

$$\begin{aligned} \pi(q, x_1x_2 \dots x_n) &= \pi(\pi(q, x_1), x_2x_3 \dots x_n), \\ \lambda(q, x_1x_2 \dots x_n) &= \lambda(q, x_1)\lambda(\pi(q, x_1), x_2x_3 \dots x_n). \end{aligned}$$

Exercise 2.4. *Prove that an initial automaton acts on* X^* *as a (graph) homomorphism preserving the root and every invertible initial automaton acts on* X^* *as a (graph) automorphism.*

Now, we see that (invertible) initial automata define homomorphisms (automorphisms) of X^*. However, the situation is much better than this since this link can be reversed and, as we will see now, automata (not necessarily finite, though) give us a way to describe every homomorphism of X^*.

Indeed, any homomorphism of X^* can be encoded by the action of an initial automaton. In order to show this we need a notion of a *section* of a homomorphism at a vertex of the tree. Let g be a homomorphism of the tree X^* and $x \in X$. Then for any $v \in X^*$ we have

$$g(xv) = g(x)v'$$

for some $v' \in X^*$. Then the map $g|_x \colon X^* \to X^*$ given by

$$g|_x(v) = v'$$

defines a homomorphism of X^* and is called the *section* of g at vertex x. Furthermore, for any $x_1x_2 \dots x_n \in X^*$ we define

$$g|_{x_1x_2\dots x_n} = g|_{x_1}|_{x_2} \dots |_{x_n}.$$

Given a homomorphism g of X^* we construct an initial automaton $\mathcal{A}_g$ whose action on X^* coincides with that of g as follows. The set of states of $\mathcal{A}_g$ is the set $\{g|_v \colon v \in X^*\}$ of different sections of g at the vertices of the tree. The transition and output functions are defined by

$$\begin{aligned} \pi(g|_v, x) &= g|_{vx}, \\ \lambda(g|_v, x) &= g|_v(x), \end{aligned} \tag{2}$$

and the initial state of $\mathcal{A}_g$ is g. Note, that $\mathcal{A}_g$ need not be finite.

Exercise 2.5. *Prove that for each homomorphisms g of X^* the initial automaton $\mathcal{A}_g$ defined by transition and output functions* (2) *induces g.*

Finally, we note that any homomorphism of X^* induces an action on the set X^∞ (often denoted as X^ω or $X^{\mathbb{N}}$ in the literature) of all infinite words over X that can be viewed as a boundary of the tree X^*. Topologically, X^∞ is homeomorphic to the Cantor set, and homomorphisms and automorphisms of X^* induce continuous transfomrations and homeomorphisms of X^∞, respectively.

3 Groups and semigroups generated by Mealy automata

After we defined the action of initial automata on finite strings, we can proceed with a very natural definition of a (semi)group generated by noninitial automaton $\mathcal{A}$.

Definition 3.1. The semigroup (group) generated by all states of an automaton $\mathcal{A}$ is called an *automaton semigroup* (*automaton group*) and denoted by $\mathbb{S}(\mathcal{A})$ (respectively $\mathbb{G}(\mathcal{A})$).

Another popular name for automaton groups and semigroups is self-similar groups and semigroups (see [Nek05]). The reason for such name will be clear from the discussion below.

Throughout the paper we will use the following convention. If g and h are the elements of some (semi)group acting on set A and $a \in A$, then

$$gh(a) = h(g(a)). \tag{3}$$

Note that this is not the only convention that is used in the literature. However, this notation agrees with the order of multiplication of permutations in `GAP` system [GAP24], which we will use quite substantially later.

Taking into account convention (3) one can compute sections of any element of an automaton semigroup as follows. If $g = g_1 g_2 \cdots g_n$ and $v \in X^*$, then

$$g|_v = g_1|_v \cdot g_2|_{g_1(v)} \cdots g_n|_{g_1 g_2 \cdots g_{n-1}(v)}. \tag{4}$$

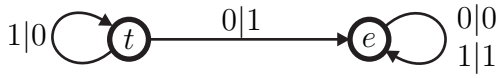
Example 3.2 (Trivial example)**.** Consider the following automaton $\mathcal{I}$:

This is the automaton consisting only of one state, and this state defines the trivial permutation of the alphabet X. Moreover, the transformation of X^* defined by this state is also trivial, since this state after reading each input letter x simply outputs x and stays in the same state. Thus, this automaton generates the trivial group.

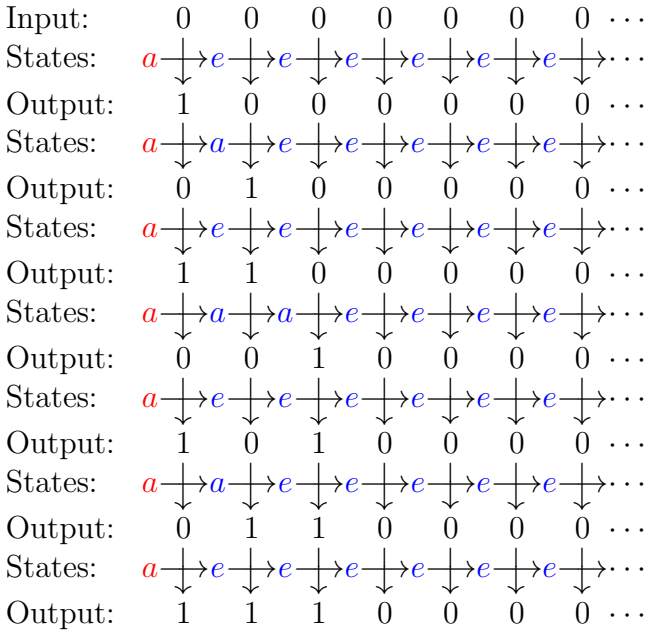
Example 3.3 (Baby example)**.** If we modify the output function in the previous example we obtain the following automaton $\mathcal{A}$:

Similarly, this automaton consists only of one state s, so the group $\mathbb{G}(\mathcal{A})$ is cyclic. Let us look at the transformation of X^* defined by $\mathcal{A}_s$. It changes the first letter of an input word and handles the rest of the word in the same way. Therefore, it will flip all letters of the input word and, hence, this automaton generates the group $\mathbb{Z}_2$ of order 2.

Example 3.4 (Toddler example)**.** Consider the following automaton $\mathcal{B}$:



This example requires slightly more sophisticated analysis. First of all, note that there are no arrows from the state e to a. Therefore, the transformation of X^* defined by $\mathcal{B}_e$ is the same as in Example 3.2, i.e. the trivial transformation. Hence, the automaton group $\mathbb{G}(\mathcal{B})$ is cyclic and generated by the transformation $\mathcal{B}_t$. The only question remaining is whether $\mathcal{B}_t$ has finite or infinite order. We will prove that it has infinite order by looking at the action of powers of $\mathcal{B}_t$ on the sequence 0^∞ (recall, that each initial automaton naturally acts also on the set of infinite words over X). Let us construct a table similar to (1).



(5)

From this table one can clearly see the pattern of $t^n(0^\infty)$. Namely, each infinite string over $X = \{0, 1\}$ defines a 2-adic integer. Then each application of t to a 2-adic integer w simply adds $1 (= 10^\infty)$ to w in the ring of all 2-adic integers (usually also denoted by $\mathbb{Z}_2$ in the literature, but in these notes we generally will denote by $\mathbb{Z}_2$ the group of size 2). Now it is easy to explain this phenomenon. Indeed, the addition of one in $\mathbb{Z}_2$ (2-adic integers) is performed as follows. If the first input letter is 0, then we flip it to 1 and do not change the rest of the input number. On the other hand, if the input letter is 1, then we output 0, but then we have to carry over 1 to the next position, i.e. we will add one to the "rest" of the 2-adic number. This procedure is clearly encoded in the state t of automaton $\mathcal{B}$.

Automaton $\mathcal{B}$ plays an important role and we will see it often in future. It is called the *adding machine* (or *odometer*). Since the set of 2-adic integers is infinite, the addition of 1 clearly has an infinite order. Therefore, the group $\mathbb{G}(\mathcal{B})$ is isomorphic to the infinite cyclic group $\mathbb{Z}$.

Note, that Example 3.4 used the fact that one can endow the boundary of the tree X^∞ with certain algebraic structure ($\mathbb{Z}_2$ in this case). Similar ideas will be used later, but with different algebraic structures. For example, for an alphabet X of cardinality d one can identify X^∞ with the ring $\mathbb{Z}_d[[t]]$ of formal power series over $\mathbb{Z}_d$ (see [GNS00, SS05], or with the infinite dimensional $\mathbb{Z}_d$-module $\mathbb{Z}_d^\infty$ (see [SS16]).

The three examples above, (un)fortunately (you choose), conclude the list of trivial examples where the structure of the group is immediately evident. In most cases the groups generated by automata are quite weird and untame, with many essentially new examples that simply do not have simpler description than the one given by automaton. As mentioned in Introduction, many of them have extraordinary properties, like being torsion infinite finitely generated, or being of intermediate growth. It is probably a right time now to introduce the first (and most famous) example of an automaton group up to date, constructed by Grigorchuk in 1980 [Gri80].

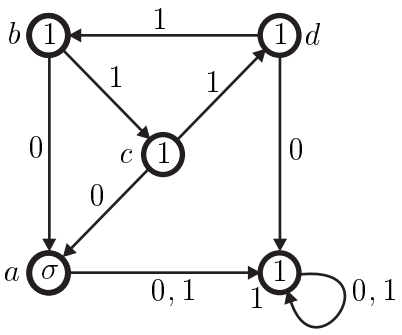


Figure 3: The automaton generating the Grigorchuk group

Example 3.5 (Adult example)**.** Consider the 5-state automaton $\mathcal{G}$ depicted in Figure 3. Note, that the Moore diagram on the figure is a bit different from those shown in Figure 2. In the literature both types of the diagrams are used, so even though it might look confusing initially, it is important to know how to interpret both of these types. In this diagram only the first components of arrows' labels are preserved (defining the transition function), but the states are labelled by the permutations of X (where 1 and σ denote the trivial and nontrivial permutations in $\mathrm{Sym}(X)$, respectively). These labels define the output function of the automaton. Namely, if the state q is labelled by the permutation $\pi \in \mathrm{Sym}(X)$ and the automaton reads $x \in X$ while being in state q, then the output letter will by $\pi(x)$. So, these labels effectively record the information contained in the second components of arrows' labels in the other type of Moore diagrams. This type of diagrams is usually more compact and clear, but a bit less convenient while trying to find the images of input words and when working with non-invertible automata or automata over large alphabets.

The next theorem is the congregation of results of several papers describing some of the most striking features of the group generated by $\mathcal{G}$.

Theorem 3.6 ([Gri80, Gri84, Lys85])**.** *The group $\mathbb{G}(\mathcal{G})$ is an infinitely presented, amenable but not elementary amenable 3-generated infinite 2-group of intermediate growth.*

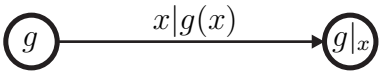
In these notes we will prove only that $\mathbb{G}(\mathcal{G})$ is infinite 2-group.

3.1 Language of self-similar groups

When we talk about sections of automorphisms (versus the states of corresponding automata), often the term "*self-similar*" is used instead of "automaton". Using both of these essentially equivalent terms quite often is helpful to emphasise the context.

Definition 3.7. A subset S of $\mathrm{Aut}\, X^*$ is called *self-similar* if all sections of each of the element in S are in S again. A group $G < \mathrm{Aut}\, X^*$ is called *self-similar* if it is self-similar as a set.

Clearly, a self-similar set S has a structure of an automaton with transitions defined by:



for each $g \in S$ and $x \in X$.

An automaton group can be naturally defined as a group generated by a self-similar set. On the other hand, the whole self-similar group G defines an automaton $\mathcal{A}(G)$ whose states are elements of the group.

The following trivial observation relates self-similar and automaton groups.

Proposition 3.8. *A group G is self-similar if and only if it is generated by (possibly infinite) automaton.*

Proof. If G is self-similar, then the it is generated by $\mathcal{A}(G)$. On the other hand, equations (8) and (9) that state that the sections of the products and inverses are products and inverses of the sections, guarantee that the section of any element of G generated by an automaton at any vertex will again belong to G. □

4 Wreath products and wreath recursion

The language of automata and their Moore diagrams is a very convenient tool to visualize the object defining a group. It also allows us to define several important classes of automaton groups. However, in order to have better computational handle, the language of, so-called, *wreath recursion* is used. This is a way to encode the same data as the automaton does, but it provides several crucial computational tools and techniques.

4.1 Basics on wreath products

Suppose there are two groups G and H acting on the sets X and Y, respectively. We start by introducing the construction of the permutational wreath product of these two groups. The formal definition involves the notion of the semidirect product, but we will explain below how it actually works.

Definition 4.1. The group $G \wr H = G^Y \rtimes H$, in which H acts on G^Y by permuting the indices in the tuples according to its action on the set Y, is called the *permutational wreath product* of G and H.

To simplify the notation, we will temporarily assume that $Y = \{y_1, y_2, \dots, y_d\}$. Then by definition, the elements of $G \wr H$ are tuples $\big((g_1, g_2, \dots, g_d), h\big)$ for some $g_i \in G$ and $h \in H$. We will usually write this element simply in the form $(g_1, g_2, \dots, g_d)h$ if it does not cause any confusion. This element acts on $X \times Y$ in the following picturesque way. Represent $X \times Y$ as a "wreath" with the main circle corresponding to Y, and $|Y|$ small circles that correspond to X (one circle for each element of Y). See Figure 4 for illustration. Then $(g_1, g_2, \dots, g_d)h$ first permutes elements of the small circles corresponding to y_i by g_i, and then permutes the small circles along the big circle according to the action of $h \in H$.

The product of elements $g = (g_1, g_2, \dots, g_d)h$ and $g' = (g'_1, g'_2, \dots, g'_d)h'$ corresponds to the composition of this actions and thus, is defined by the following formula:

$$gg' = (g_1 g'_{h(1)}, g_2 g'_{h(2)}, \dots, g_d g'_{h(d)})hh'. \tag{6}$$

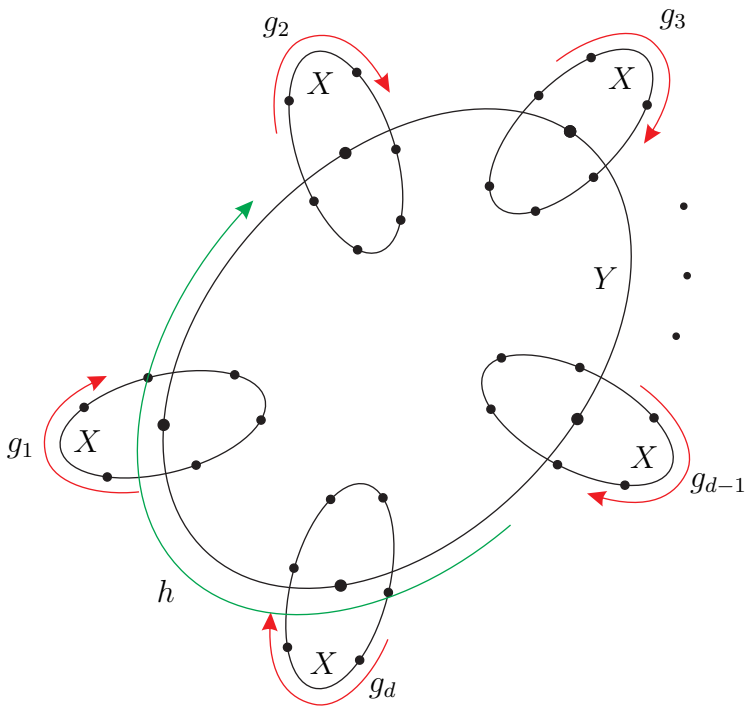


Figure 4: The action of an element $(g_1, g_2, \ldots, g_d)h \in G \wr H$ on $X \times Y$

Example 4.2. The permutational wreath product of two symmetric groups $S_d \wr S_d$ is naturally isomorphic to the automorphisms group of the finite d-ary tree of depth 2.

Example 4.3. The full automorphism group of the d-ary tree $\operatorname{Aut} T_d$ satisfies:

$$\operatorname{Aut} T_d \cong \operatorname{Aut} T_d \wr S_d.$$

4.2 Wreath recursion

Similarly to Example 4.3, for any automaton group G there is a natural embedding

$$\psi \colon G \hookrightarrow G \wr \operatorname{Sym}(X)$$

defined by

$$G \ni g \mapsto (g_1, g_2, \ldots, g_d)\sigma_g \in G \wr \operatorname{Sym}(X), \tag{7}$$

where $g_1, g_2, \ldots, g_d$ are the sections of g at the vertices of the first level of X^*, and σ_g is a permutation of X induced by the action of g on the first level of the tree. In this case, with a slight abuse of notation, we will write

$$g = (g_1, g_2, \ldots, g_d)\sigma_g.$$

If σ_g is the trivial permutation, it is customary to omit it.

The above embedding is convenient in computations involving the sections of automorphisms, as well as for defining automaton groups. Sometimes it is called the *wreath recursion*

defining the group. The main benefit of the wreath recursion language is that it is easy to multiply automorphisms and find their inverses in the same form. Indeed, if $g \mapsto (g_1, g_2, \ldots, g_d)\sigma_g$ and $h \mapsto (h_1, h_2, \ldots, h_d)\sigma_h$ are two elements of $\operatorname{Aut} X^*$, then equation (6) yields:

$$gh = (g_1 h_{\sigma_g(1)}, g_2 h_{\sigma_g(2)}, \ldots, g_d h_{\sigma_g(d)})\sigma_g\sigma_h \tag{8}$$

and

$$g^{-1} = (g^{-1}_{\sigma_g^{-1}(1)}, g^{-1}_{\sigma_g^{-1}(2)}, \ldots, g^{-1}_{\sigma_g^{-1}(d)})\sigma_g^{-1}. \tag{9}$$

Remark 4.4. *A particular important case of the above formulas is when g and h act trivially on the first level, i.e. both σ_g and σ_h are trivial permutations of X. In this case σ_g does not cause any shuffling of the sections, so the multiplication is performed simply as in the direct product.*

For example, the adding machine automaton from Example 3.4 corresponds to the wreath recursion:

$$\begin{aligned} 1 &= (1,1), \\ t &= (1,t)\sigma, \end{aligned}$$

and the Grigorchuk group introduced in Example 3.5 can be defined as:

$$\begin{aligned} 1 &= (1,1), \\ a &= (1,1)\sigma, \\ b &= (a,c), \\ c &= (a,d), \\ d &= (1,b). \end{aligned}$$

In most cases the line corresponding to the trivial generator of the group is omitted as well.

We will first show the power of the language of wreath recursion by showing a couple of examples of calculations.

Example 4.5 (Example 3.3 revised)**.** Recall that the group $\mathbb{G}(\mathcal{A})$ from Example 3.3 is generated by an element a of order 2 and is isomorphic to $\mathbb{Z}_2$. Let us see from the wreath recursion that a^2 represents the trivial automorphism. Indeed, by (8) we get:

$$a^2 = (a,a)\sigma \cdot (a,a)\sigma = (a \cdot a, a \cdot a)\sigma^2 = (a^2, a^2).$$

The last equality should be understood as follows. The automorphism a^2 does not swap the vertices of the first level, and then acts on both left and right subtrees of X^* again as itself. So, it will fix the vertices of the second level and will act on the subtrees hanging down from the second level again as itself. Propagating this argument down the tree shows that a^2 will not move any vertex of X^*, so $a^2 = 1$.

Now let us go back to Example 3.4 of the adding machine.

Example 4.6 (Example 3.4 revised)**.** Recall that we proved that the generator t has an infinite order by looking at the orbit of 0^∞. While that method works for the adding machine, its power is quite limited and it does not work in many situations. Wreath recursion gives another simple way to see it.

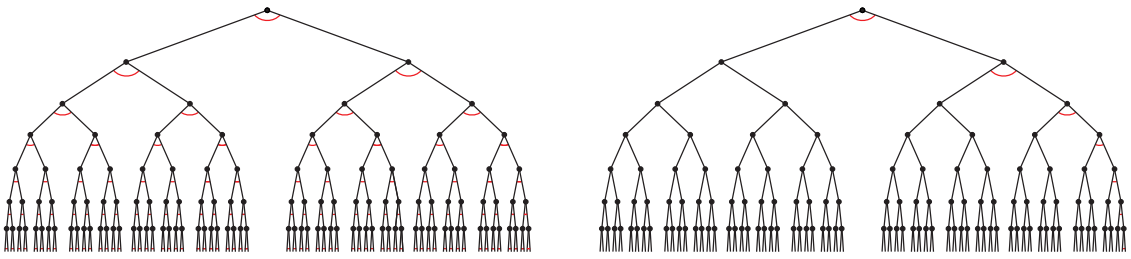


Figure 5: Full portraits of elements (a) $a = (a, a)\sigma$ and (b) the adding machine $t = (1, t)\sigma$

Suppose t has a finite order n. Since t swaps the vertices of the first level, any of its odd powers will also swap these vertices, hence it will represent a nontrivial element of $\mathbb{G}(\mathcal{B})$. Therefore the order n can only be even, say, $n = 2m$. But then applying (8) to $t = (1, t)\sigma$ we get:

$$t^2 = (1, t)\sigma \cdot (1, t)\sigma = (1 \cdot t, t \cdot 1)\sigma^2 = (t, t)$$

Taking into account Remark 4.4, we then immediately obtain that:

$$t^{2m} = (t^m, t^m).$$

But since the order of t is $2m$, t^{2m} is the trivial automorphism of X^*. Hence, its sections at the vertices of the first level also must be trivial. So we obtain that $t^m = 1$, which is in clear contradiction with the fact that the order of t is $2m$.

5 Portraits of automorphisms

There is another way to look at automorphisms of X^*. Namely, every automorphism is completely defined by its *portrait*. There are at least four variations of this notion, so we will start from the one that can be applied to any automorphism.

Definition 5.1. The *full portrait* of an automorphism $g \in \mathrm{Aut}(X^*)$ is a labelled rooted tree X^* in which each vertex v is labelled by the permutation of X induced by the section $g|_v$.

Clearly, each such full portrait is an infinite graph. In all the figures, labels of vertices will be drawn right under the vertex. In the case of the binary tree (when $|X| = 2$), since there is only one nontrivial permutation of X, it is customary to denote this nontrivial permutation by a little arc (called the *switch*) connecting the edges sticking out from the vertex. The labels corresponding to the trivial permutation of X are also suppressed in this case. The label of the vertex tells how the immediate children of this vertex are permuted under the action of an automorphism. The examples of the full portraits of the element $a = (a, a)\sigma$ and the adding machine $t = (1, t)\sigma$ from Examples 3.3 and 3.4 are shown in Figure 5.

As you can see from Figure 5 drawing the whole portrait in the case of the adding machine might actually hide the essential structure. Often, to see what is going one, it is better not to draw subtrees where the action of an automorphism is trivial. I.e. after reaching a vertex v such that $g|_v$ is trivial, we stop drawing anything along this branch.

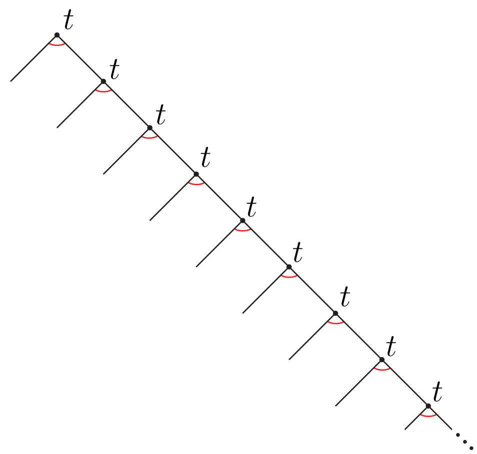


Figure 6: Simplified portrait of the generator t of the adding machine

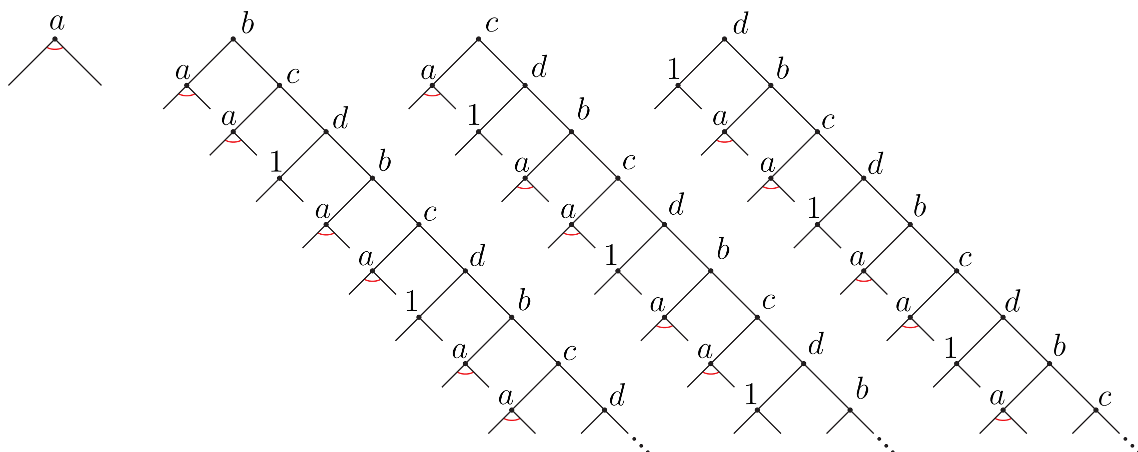


Figure 7: Simplified portraits of the generators of the Grigorchuk group

Definition 5.2. The *simplified portrait* of an automorphism $g \in \mathrm{Aut}(X^*)$ is obtained from its full portrait by removing all subtrees hanging down from vertices v such that $g|_v$ is the trivial automorphism.

Often for convenience the vertices of a portrait of g are additionally labelled by the names of the sections $g|_v$ acting on corresponding subtrees. These labels are usually located above the vertices.

For example, the simplified portrait of the generator t of the adding machine is shown in Figure 6, and those of the generators of the Grigorchuk group are depicted in Figure 7.

Finally, if an automorphism is defined by finite initial automaton (or, equivalently, has finite number of sections), it is often convenient to stop drawing the branch in its portrait after reaching a vertex v such that the section $g|_v$ has already appeared earlier in the portrait.

Definition 5.3. The *reduced portrait* of an automorphism $g \in \mathrm{Aut}(X^*)$ with finite number of sections is obtained from its full portrait in the following way. Order the vertices of X^*

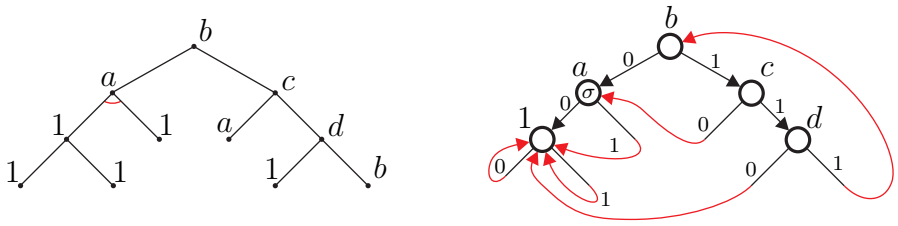


Figure 8: Reduced portrait of the generator b of the Grigorchuk group and the initial automaton built from this portrait

lexicographically and loop over all the vertices remembering the sections that have appeared, at the same time including corresponding vertices and their labels in the full portrait of g (including the names of the sections) into the reduced portrait. In this process, when a section at vertex v have appeared earlier on the list, include v into the reduced portrait, stop moving along the branch containing v and move to the next vertex on the list that does not have v as a prefix.

If an automorphism g has only finitely many sections, the procedure in the above definition will clearly terminate and produce a finite tree whose vertices are labelled by permutations of X and by the names of the sections of g. An example of the reduced portrait of the generator b of the Grigorchuk group is shown in the left side of Figure 8.

Note, that the reduced portrait allows us to recover the initial automaton defining g as shown in the right side of Figure 8. Simply keep in mind that the sections of g correspond to the states of this automaton, transitions are coming from the edges in the tree, and the output function is defined by the permutational labels of the vertices in the portrait. The last observation also allows us to effectively construct reduced portraits of automorphisms defined by initial automata or by wreath recursion. Observe that the automaton in the right side of Figure 8 is, as expected, isomorphic to the automaton in Figure 3.

6 Algorithmic aspects

6.1 Word Problem

Even though most of the algorithmic questions in the class of (semi)groups generated by automata are either not known to be solvable, or are known to be unsolvable, one of the most important ones – the word problem – has quite an easy general solution.

The word problem is one of the three classical algorithmic problems in group theory formulated by Max Dehn in 1911 [Deh11]. It is stated as follows:
The word problem for a finitely generated group G is an algorithmic problem of deciding whether two given words over a symmetric generating set for G represent the same element of G.

The other two problems formulated by Dehn are the *conjugacy problem* (asking whether two given elements of a group are conjugate in it) and the *isomorphism problem* (asking

whether there is an algorithm testing the isomorphism of groups from a given class). Conjugacy problem was quite recently shown to be undecidable in groups generated by automata [ŠV12]. The question whether the isomorphism problem is decidable in this class is still open.

Perhaps, among these three problems the positive solution of the word problem in G has the strongest impact on the study of G. Indeed, if we know how to distinguish elements of G, we can:

- test whether the word represents the trivial element of G or not;
- search for relations in G;
- enumerate effectively all the elements of G;
- draw finite balls in the Cayley graphs of G.

Not all groups admit an algorithm deciding the word problem. The first example of a finitely presented group with undecidable word problem was constructed by Novikov in 1955. Therefore, it is certainly very good that this problem has an easy solution in the class of groups generated by automata.

The procedure itself is recursive and quite simple. We have actually seen an instance of it in Example 4.5. Let G be an automaton group generated by a finite automaton $\mathcal{A}$. Then the set S of states of $\mathcal{A}$ is the generating set for G. Suppose we need to check whether two words w and w' over $S \cup S^{-1}$ correspond to the same element of G. Since the equality $w = w'$ is equivalent to $w^{-1}w' = 1$, we can without loss of generality assume that w' is the trivial word. So we only need to check whether w corresponds to a nontrivial element of G or not.

In order to do this, we go over all possible sections of w and look either for nontrivial action on the first level, or for repetitions of sections. Whenever we see a repetition, we stop going along this branch of the tree. Note, that the set of sections of w is finite, since every section is represented by a word of length not more than the length of w.

- If at some point we find a section that acts nontrivially on the first level, then $w \neq 1$.
- If, on the contrary, we searched over all the sections (since w has only finite number of sections, the process will eventually terminate) and did not find a section acting nontrivially on the first level, we declare $w = 1$.

This is not an extremely efficient algorithm and it requires an exponential (in terms of the length of the input word) running time and space in the worst case. Philosophically, this is because the sections of w are parameterized by the words of length at most length of w, but the size of this set clearly grows exponentially with the length of w.

However, there are some classed of automaton groups that admit polynomial time word problem algorithms. The most important of those is the class of *contracting groups* to be discussed in Section 8.

6.2 Introduction to *AutomGrp* package

As you can see from Subsection 6.1 the calculations related to groups generated by automata are often conceptually simple, but technically difficult and time-consuming. This is precisely the situation where computer help is needed. As of now, there are two packages that deal with this class: `AutomGrp` by Yevgen Muntyan and the author [MS25] and `FR` by Laurent Bartholdi [Bar24], both written for the `GAP` system [GAP24]. In the rest of the text we will use mostly `AutomGrp` package and will try to hide as many technical details into computer calculations as the proofs will allow us. Often, this will make the important arguments much simpler.

The `GAP` system is a free multi-platform powerful computer algebra system specializing in algebra with emphasis in group theory. At the present time it is one of the most influential system in the group-theoretic community. It is available for download at

`http://www.gap-system.org`

This web-page also contains the information for the installation and various manuals on how to use the system. The package `AutomGrp` is a part of the standard installation and can be loaded by issuing:

```
gap> LoadPackage("automgrp");
-----------------------------------------------------------------
     ^                                        ___
    / \                                      /   \
   /   \           _______  ___            ||         ___   __
  /_____\   ||  ||    |    /   \  |\ /| ||   __ |/   | |  \
 /       \  ||  ||    |   ||   || | V | ||   || |      |__/
/         \  \__/     |    \___/  |   |  \___/  |      |

Loading  AutomGrp 1.3 (Automata Groups and Semigroups)
by Yevgen Muntyan (muntyan@fastmail.fm)
   Dmytro Savchuk (http://savchuk.myweb.usf.edu/)
Homepage: http://finautom.sourceforge.net/
-----------------------------------------------------------------
true
```

in `GAP` terminal.

The package has quite a reach toolbox consisting of various functions. The complete manual for the package is available at its website. Here we will be using some of its functionality at appropriate places.

The primary way to define automaton groups is using its wreath recursion and `AutomatonGroup` command. The alphabet in `AutomGrp` will always be $X = \{1, 2, \dots, d\}$ for some $d \geq 2$, and the permutations of X are written in the standard cyclic form (e.g., $(1, 3, 4)(2, 5)$ is a permutation of $\{1, 2, 3, 4, 5\}$). For example, here is the line that creates the Grigorchuk group G:

```
gap> G := AutomatonGroup( "a = (1,1)(1,2), b = (a,c), c = (a,d), d = (1,b)");
< a, b, c, d >
```

Now one can ask many things about G: whether it is finite, does it have elements of infinite order, find short relations, etc. But for the next section we will mostly interested in a fast and automatic way to solve the word problem and to find the order of an element. The standard algorithm solving the word problem in automaton groups is implemented in `AutomGrp` and can be called by the command `IsOne` applied to a group element. For example, to verify that the generator b of G is an involution, one can check:

```
gap> IsOne(b^2);
true
```

Alternatively, one could try to issue the `Order` command:

```
gap> Order(b);
2
```

However, there is no known algorithm that always finds the order of an element of an automaton group. Therefore, this command in some cases can run until it uses all of the available resources and might not produce a definite answer. For Grigorchuk group (and more generally for groups generated by, so-called, bounded automata) this never happens.

Similarly, we can compare two words simply by using `=` operation:

```
gap> b*c = d;
true
```

Finally, one of the most useful functions is `Decompose`, which allows us to quickly compute the wreath recursion of a given automorphism:

```
gap> Decompose(a*b*a*c*a);
(c*a, a*d)(1,2)
```

We will gradually introduce more functions along the exposition below.

6.3 Order problem

Even though the word problem in automaton groups is decidable, the good algorithmic news unfortunately end at this point. On the other hand, the lack of general solutions for other algorithmic problems makes it interesting to study the class of automaton groups as one has to try to apply various tools to get a result. In this subsections we turn our attention to the, so-called, order problem. This partial solutions of this problem will be used on multiple occasions throughout the text.

The order problem for a finitely generated group G is an algorithmic problem of deciding whether a given element of G has finite order or not.

The first thing to point out is that the word problem gives half of the solution to the order problem. Indeed, if an element g has finite order, this can be verified in finite time simply by checking whether g^n is trivial for all $n \geq 1$. Therefore, we will mostly be interested in finding the ways that show that an element has infinite order. There are three main ways that sometimes allow us to make such conclusion.

- The first method is based on the analysis of the sections of an element and is a generalization of the method used in Example 4.6.
- The second way is a generalization of Example 3.4 and is based on the (partial) understanding the orbits of infinite words over X under the action of powers of g. The second way is less automatic at this moment than the first one, but it seems to be applicable in many situations where the first method fails.
- The third method is based on the algorithm verifying whether an element acts transitively on all levels of the tree. This is quite a restrictive method as most of automorphisms do not possess this property, but it is simple to check on the binary tree and in some situations on larger trees.

There are also other methods (see, for example, [KPS16]), but their application range is quite limited at the moment.

We will start by describing the first method. Below, for any element g of any group we will denote by $\mathcal{O}(g)$ its order. Further, for each $m \geq 0$ each element $g \in \operatorname{Aut} X^*$ induces the permutation of X^m that we will denote by $\pi_g(m) \in \operatorname{Sym}(X^m)$.

Theorem 6.1. *If for an element h of a self-similar group G there exists a vertex $v \in X^l$ for some $l > 0$ satisfies:*

1. *there is $n > 1$ such that $\gcd(n, \mathcal{O}(\pi_h(m))) > 1$ for some level $m > 1$ and $h^n|_v = h^{\pm 1}$,*
2. *$h^n(v) = v$,*

then $\mathcal{O}(h) = \infty$.

Proof. Suppose h has finite order r. Then

$$1 = h^{\operatorname{lcm}(n,r)} = h^{\frac{nr}{\gcd(n,r)}} = (h^n)^{\frac{r}{\gcd(n,r)}}.$$

In particular all of the sections of $(h^n)^{\frac{r}{\gcd(n,r)}}$ are trivial. But since $h^n(v) = v$, we get

$$1 = (h^n)^{\frac{r}{\gcd(n,r)}}|_v = (h^n|_v)^{\frac{r}{\gcd(n,r)}} = (h^{\pm 1})^{\frac{r}{\gcd(n,r)}}.$$

To finish the proof it is enough to mention that $\mathcal{O}(\pi_h(m)) \mid r$ since $\pi_h(m)$ is a homomorphic image of h under the canonical epimorphism $\operatorname{Aut} X^* \to \operatorname{Sym}(X^m)$, and so $\gcd(n, r) > 1$ and $\frac{r}{\gcd(n,r)} < r$. The last inequality is in clear contradiction with the minimality of r. □

Often it is important to obtain at least one element of infinite order since it enables us to construct many other such elements by the following proposition.

Proposition 6.2. *If for an element g of a self-similar group G there exists a vertex $v \in X^l$ such that:*

1. $g(v) = v$;
2. $\mathcal{O}(g|_v) = \infty$,

then $\mathcal{O}(g) = \infty$.

Example 6.3. Consider the group G defined by the wreath recursion below:

$$\begin{aligned} a &= (c, a)\sigma, \\ b &= (a, b), \\ c &= (b, a). \end{aligned}$$

We will show that an element $h = ab^{-1}$ has an infinite order. Indeed, for a vertex $v = 00$ and $n = 2$ we have:

- $h^2|_{00} = (ab^{-1})^2|_{00} = ba^{-1} = h^{-1}$,
- $h^2(00) = 00$.

Also, since h acts nontrivially on the first level of the binary tree, $\pi_h(1)$ is a nontrivial transposition of order 2. Hence, we have $\gcd(n, \mathcal{O}(\pi_h(1))) = \gcd(2, 2) > 1$ and all the conditions of Theorem 6.1 are satisfied. Thus, $h = ab^{-1}$ has infinite order.

In `AutomGrp` package the algorithm described in Theorem 6.1 and Proposition 6.2 is implemented in functions `OrderUsingSections` and `FindElementOfInfiniteOrder`. In many cases, to get that the order of an element is infinite, we show that some of its powers have a section at some fixed vertex that satisfies conditions of Theorem 6.1. This is shown in example below.

Example 6.4. Consider a group G acting on a ternary tree with the wreath recursion below:

$$\begin{aligned} a &= (1, b, a)(012), \\ b &= (a, a, b)(01). \end{aligned}$$

The group itself was chosen essentially at random to demonstrate the power of the method. We will use `SetInfoLevel` command to make `AutomGrp` produce the proof for us.

```
gap> G:=AutomatonGroup("a=(1,b,a)(1,2,3),b=(a,a,b)(1,2)");
< a, b >
gap> SetInfoLevel(InfoAutomGrp,3);
gap> FindElementOfInfiniteOrder(G,10,10);
#I  (b*a^2)^2*a^2 is obtained from (a)^18
    by taking sections and cyclic reductions at vertex [ 1, 1, 1, 2 ]
#I  (b*a^2)^2*a^2 is obtained from ((b*a^2)^2*a^2)^3
    by taking sections and cyclic reductions at vertex [ 1, 2, 1, 2 ]
a
```

Now we can check that for $h = (ba^2)^2a^2$ there is $n = 3$ and a vertex $v = 0101$ such that:

- $h^3|_{0101} = h$,
- $h^3(0101) = 0101$,
- $\mathcal{O}(\pi_h(1)) = \mathcal{O}((021)) = 3$, so $\gcd(3, 3) = 3 > 1$.

Hence, by Theorem 6.1 the element h has infinite order.

At the second stage, we check that

- $a^{18}(0001) = 0001$,
- $a^{18}|_{0001} = h$.

Therefore, by Proposition 6.2 the elements a^{18}, and consequently a as well, have infinite order.

7 Motivating example: Grigorchuk group

In this section we will use the developed language and techniques to obtain one of the first striking results in the area.

We first familiarize ourselves with G a little better. Recall, that in the previous section we observed that $b^2 = 1$ in G. More generally, using the algorithm solving the word problem in automaton groups described in Subsection 6.1, we obtain:

Proposition 7.1. *In the Grigorchuk group G the following relations hold:*

$$a^2 = b^2 = c^2 = d^2 = bcd = (ab)^{16} = (ac)^8 = (ad)^4 = 1.$$

In particular, it follows that there is an epimorphism from a finitely presented group $\Gamma = \langle a, b, c, d \mid a^2 = b^2 = c^2 = d^2 = bcd = 1\rangle$ to G. This group Γ is isomorphic to the free product $\mathbb{Z}_2 * K_4$ of the group $\langle a\rangle$ of order 2 and the 4-element Klein group $\langle b, c, d\rangle$. Therefore, each element g of G can be represented by a word over generators of the following form:

$$g = (a) * a * a * a \cdots * a * (a), \tag{10}$$

where each $*$ is one of b, c, or d, and the first and/or the last a may be missing. This representation is not unique. For example, by Proposition 7.1 $(ad)^4 = 1$, which implies that $adad = dada$.

As often happens in group theory, to understand better the structure of a group, one studies its subgroups. Any group G acting on a rooted tree X^* has a natural descending sequence of normal subgroups $\mathrm{Stab}_G(n), n \geq 1$ consisting of elements of G that stabilize levels X^n of the tree. I.e.,

$$\mathrm{Stab}_G(n) = \{g \in G \mid v^g = v \text{ for all } v \in X^n\}.$$

These subgroups are called the *stabilizers of levels* of X^* and play a crucial role in many algorithmic questions.

Exercise 7.2. *Prove that* $\mathrm{Stab}_G(n)$ *is a normal subgroup of* G *for any* n*.*

Since the intersection of all the stabilizers is trivial (as any element of the intersection must stabilize all the levels of X^*), we immediately obtain the following corollary.

Corollary 7.3. *Any group acting faithfully on the rooted tree is residually finite.*

The quotients $G/\mathrm{Stab}_G(n)$ are finite groups, whose size usually grows rapidly with n. They naturally consist of permutations induced by the elements of G on the n-th level. This factor-groups in many cases give a descent finite approximation of the whole group. A lot of questions are often answered simply by looking at these finite quotients.

Let us go back to the Grigorchuk group. We denote by H the stabilizer $\mathrm{Stab}_G(1)$ of the first level in G. I.e. H consists of those elements of G that act trivially on the first level. Since the only generator of G that swaps the vertices of the first level is a, the group H consists of exactly those elements of G that can be represented by words of the form (10) with even number of a's. This immediately gives that $[G : H] = 2$, since each element of $G - H$ can be presented as ha for some $h \in H$. Hence, we obtain that H is normal in G as a subgroup of index 2. Moreover, each word of the form (10) that has even number of a's (and thus representing some element of H) can be broken down into the blocks from $\{b, c, d, aba, aca, ada\} = \{b, c, d, b^a, c^a, d^a\}$ as shown in the example below:

$$abacacadabacada = aba \cdot c \cdot aca \cdot d \cdot aba \cdot c \cdot ada.$$

We summarize these relatively trivial observations in the following proposition.

Proposition 7.4. *The subgroup* $H = \mathrm{Stab}_G(1)$ *of* G *is normal in* G*, has index* 2 *and is generated as*

$$H = \langle b, c, d, b^a, c^a, d^a \rangle.$$

7.1 Infiniteness

The goal in this subsection is to prove the following proposition.

Proposition 7.5. *The group* G *is infinite.*

Proof. To see that G is infinite, it is enough to construct an epimorphism from H onto G (since H has index 2 in G it is a proper subgroup).

Since H stabilizes the first level, the restriction of the embedding $\psi\colon G \to G \wr \mathrm{Sym}(X)$ defined by 7 to H is

$$\psi|_H\colon H \to G \times G.$$

With a slight abuse of notation we will denote this restriction also by ψ.

This embedding is defined on the generators of H from Propositoin 7.4 as

$$\psi\colon \begin{cases} b \mapsto (a, c), \\ c \mapsto (a, d), \\ d \mapsto (1, b), \\ b^a \mapsto (c, a), \\ c^a \mapsto (d, a), \\ d^a \mapsto (b, 1). \end{cases}$$

The compositions of ψ with the projections onto each of the coordinates of $G \times G$ are onto (since all generators of G are present in both first and second coordinate of the images of the generators of H). This observation completes the proof of infiniteness of G. $\square$

7.2 Periodicity

In this section we prove periodicity of the Grigorchuk group G. The argument is essentially the original argument by Grigorchuk from 1980 [Gri80]. More precisely, the following theorem holds.

Theorem 7.6. *The Grigorchuk group is a 2-group.*

Proof. We need to show that every element g of G has order 2^k for some $k \geq 0$. We proceed by induction on the length of a word in $\{a, b, c, d\}$ of the form (10) representing g. For words of length up to 2 the statement is trivially verifiable using the word problem algorithm. In `AutomGrp` package this can be checked as follows.

```
gap> G := AutomatonGroup("a = (1,1)(1,2), b = (a,c), c = (a,d), d = (1,b)");
< a, b, c, d >
gap> els := ListOfElements(G, 2);
[ 1, a, b, c, d, a*b, a*c, a*d, b*a, c*a, d*a ]
gap> List(els, x -> Order(x));
[ 1, 2, 2, 2, 2, 16, 8, 4, 16, 8, 4 ]
```

Now assume that for some $n > 2$ each word of length less than n represents an element of G whose order is some power of 2. Let g be an arbitrary element represented by a word of the form (10) of length n. We will consider several cases.
Case I. $g = a * a * \cdots a * a$. In this case $g^a = *a * \cdots a*$ has length at most $n - 2$, so by induction assumption it has a finite order that is a power of 2. But the order of g is always equal to the one of g^a.
Case II. $g = *_1 a * \cdots a *_2$. In this case $g^{*_1} = a * \cdots a(*_2 *_1)$ has length at most $n - 1$ since $*_2 *_1$ is either trivial if $*_1 = *_2$, or equals to the third element among $\{b, c, d\}$ different from $*_1$ and $*_2$. Similarly to Case I we deduce that the order of g is a power of 2 by induction assumption.
Case III. $g = *a * a \cdots a * a$ or $g = a * a * a \cdots a*$. Clearly as before, conjugation by a reduces the second option to the first one. Therefore we assume that $g = *a * a \cdots a * a$ and look into the following subcases.

A. There is an even number of a's in the word representing g. Then $g \in H$ and g can be written as a product of at most $\frac{n+1}{2}$ blocks from $\{b, c, d, aba, aca, ada\}$. In this case $g = (g_0, g_1)$, where g_0 and g_1 are the products of sections of blocks forming g at vertices 0 and 1, respectively. Since each such section is in $\{1, a, b, c, d\}$ and has length at most 1, the length of each of g_0 and g_1 is bounded above by the number of blocks, i.e. $\frac{n+1}{2} < n$ for $n > 2$. Hence, by induction assumption, the orders of g_0 and g_1 are 2^l and 2^m for some $l \geq 0$ and $m \geq 0$. In this case the order of $g = (g_0, g_1)$ is $2^{\max\{l,m\}}$.

B. There is an odd number of a's in the word representing g. This is the most important case hiding the heart of the whole argument. In this case $g = ha = (h_0, h_1)a$ for some $h = (h_0, h_1) \in H$ of length $n - 1$. Since

$$g^2 = (h_0, h_1)a \cdot (h_0, h_1)a = (h_0h_1, h_1h_0),$$

it will be enough to show that the order of h_0h_1 is a power of 2 (note: the order of h_1h_0 is the same as it is conjugate to h_0h_1. However, by the argument in Subcase A the length of h_0h_1 can be at most $2 \cdot \frac{(n-1)+1}{2} = n$. Hence we cannot immediately apply the induction assumption. But in the case when $l(h_0h_1) < n$, induction assumption does apply.

To finish the proof we will use the structure of G. Since $d = (1, b)$ and $ada = (b, 1)$ (so these blocks introduce 1 into either h_0 or h_1), the only situation in which the length of h_0h_1 is equal to the length of g is when h does not have blocks involving d. Additionally, if h contains a block involving c, then since $c = (a, d)$ and $aca = (d, a)$ the word h_0h_1 will either have length less than n (so that the induction assumption applies), or will contain a block involving d, thus hitting the situation just described. Finally, if all blocks in h involve no c's or d's, we must have $g = (ba)^{\frac{n}{2}}$ that clearly has an order of the form 2^i for $i \in \{0, 1, 2, 3, 4\}$ since the order of ba is 16.

□

7.3 Intermediate growth

Let $G = \langle S \rangle$ be a finitely generated group, where S is a finite generating set. The *word length* of an element $g \in G$ with respect to S is

$$|g|_S = \min\{n \geq 0 : g = s_1 s_2 \cdots s_n,\ s_i \in S \cup S^{-1}\}.$$

The *ball of radius* n centered at the identity is

$$B_S(n) = \{g \in G : |g|_S \leq n\}.$$

The *growth function* of G with respect to the generating set S is the function

$$\gamma_{G,S}(n) = |B_S(n)|,$$

that is,

$$\gamma_{G,S}(n) = \big|\{g \in G : |g|_S \leq n\}\big|.$$

Although the function $\gamma_{G,S}(n)$ depends on the choice of the finite generating set S, its asymptotic growth type does not. Therefore, one often writes $\gamma_G(n)$ when the generating set is understood or when discussing the growth type of the group.

A growth function of a finitely generated group can be at most exponential with respect to n. For a long time only groups of polynomial or exponential growth were known, and the question of the existence of a group whose growth function grows faster than any polynomial function and slower than an exponential function was formally asked by Milnor in 1968 [Mil68]. Such groups are called groups of *intermediate growth*.

A celebrated theorem of Gromov relating a geometric and algebraic properties of a group is widely considered as one of the cornerstones in the geometric group theory.

Theorem 7.7 (Gromov [Gro81]). *A finitely generated group has polynomial growth if and only if it is virtually nilpotent.*

Gromov proved the more complicated "only if" direction establishing that every group of polynomial growth is virtually nilpotent. The "if" direction was known earlier due to works of Wolf (1968), Guivarc'h, and Bass (1972).

Grigorchuk in [Gri83] proved that his group has intermediate growth, thus solving positively Milnor's problem on growth of groups. Up to now all known examples of such groups can essentially be traced back to this construction (in a broad sense). Introductory paper by Grigorchuk and Pak [GP08] (available at `http://arxiv.org/abs/math/0607384`) provides a simple proof of intermediate growth of Grigorchuk group. Several more accurate estimates were obtained over the years by Grigorchuk, Bartholdi, Šunić, with the best known obtained by Erschler and Zheng in [EZ20]. They proved that if $\gamma_G(n)$ denotes the growth function of the first Grigorchuk group G, then

$$\lim_{n\to\infty} \frac{\log\log v_G(n)}{\log n} = \alpha_0,$$

where

$$\alpha_0 = \frac{\log 2}{\log \lambda_0} \approx 0.767429,$$

and λ_0 is the unique positive root of

$$x^3 - x^2 - 2x - 4 = 0.$$

Equivalently,

$$\gamma_G(n) = \exp\left(n^{\alpha_0 + o(1)}\right).$$

8 Contracting Groups

In the previous section we have observed the phenomenon that for Grigorchuk group the length of sections $g|_v$ as words in the generators is always shorter than the length of g for all v from the third level of X^*. This contraction played a crucial role in both periodicity and intermediate growth of G. This concept is a basis for a wide class of automaton groups called *contracting groups* that is the subject of this section. We will see that the groups from this class possess some nice algorithmic properties and allow us to formulate some general result of algebraic nature.

8.1 Definition(s) of contracting groups

There are two equivalent and equally important properties of contracting group that can be taken as a definition. We will start from one of them and state the other one in Proposition 8.2, but one should keep in mind that they are equally important.

Definition 8.1. A self-similar group $G < \operatorname{Aut} X^*$ is called *contracting* if there exists a finite set $\mathcal{N} \subset G$ such that for each $g \in G$ there is level $l \geq 0$ so that for all $k \geq l$ we have $g|_v \in \mathcal{N}$ for all $v \in X^k$. The minimal such set $\mathcal{N}$ is called the *nucleus* of G. Any finite subset of G containing the nucleus is called *quasinucleus*.

The second "definition" deals with the shortening of the lengths of sections of elements while going deeper in the tree. If a group is generated by a set S, for each $g \in G$ let $l(g)$ denote the length of the shortest word in $S \cup S^{-1}$ representing g. If a group $G < \operatorname{Aut} X^*$ is generated by a self-similar set (i.e., automaton), then the sections of each element $g \in G$ are again elements of G, so we can compute the length of these sections and compare it to the length of g.

Proposition 8.2 (Can be used as the second definition)**.** *A self-similar group G generated by a self-similar generating set S is contracting if and only if there exist $C > 0$ and level $N \geq 1$ such that for any element g of G*

$$l(g|_v) < \frac{1}{2} l(g) + C$$

for each vertex $v \in X^N$.

Proof. ($\Leftarrow$) First of all note, that since the generating set S is self-similar, the length of the sections is always not larger than the length of element according to equations (8) and (9).

Suppose $g \in G$ is an arbitrary element represented by a word of length n. Then the length of the sections of g at vertices of level N will be at most $\frac{1}{2}n + C < n$ for $n > 2C$. Thus, by going down the tree by N levels each time we can get to some level kN at which the sections of g at all vertices will have length at most $2C$. Since there is only finite number of such elements, G must be contracting.

($\Rightarrow$) Suppose that G is contracting with the nucleus $\mathcal{N}$. It is clear that for each individual element $g \in G$ the lengths of sections will eventually go down to the length of the elements of $\mathcal{N}$. However, we need to prove that the length is decreased uniformly for all elements.

Let M denote the maximal length of elements in $\mathcal{N}$. By contraction for each pair element $g \in G$ of length up to $2M$ there exists some level N_g such that the sections of g at the vertices of this level belong to the nucleus. Let

$$N = \max_{l(g) \leq 2M} \{N_g\}$$

be the level at which all the elements in G of length up to $2M$ contract to $\mathcal{N}$.

Let now g be an arbitrary element of G of length n. We can write G as $g = g_1 g_2 g_3 \cdots g_k g_{k+1}$ for $k = \left\lfloor \frac{n}{2M} \right\rfloor$ and some $g_i \in G$ such that $l(g_i) = 2M$ for $1 \leq i \leq k$ and $l(g_{k+1}) \leq 2M$. Each of g_i's will contract to the nucleus at level N, so any section of g at this level can be written as $g|_v = h_1 h_2 \cdots h_{k+1}$ for some $h_i \in \mathcal{N}$, and thus its length can be estimated as:

$$l(g|_v) \leq (k+1)M \leq \left(\frac{n}{2M} + 1\right) M = \frac{n}{2} + M = \frac{l(g)}{2} + M.$$

□

8.2 Nucleus of a contracting group

The nucleus of a contracting group G can be easily described in terms of the group automaton $\mathcal{A}(G)$. But first we introduce another important notion.

Definition 8.3. Let $S \subset \operatorname{Aut} X^*$ be a set of automorphisms of X^*. The *self-similar closure* of S is the set containing elements of S together with all the sections of all elements of S.

Proposition 8.4. *Let G be a contracting group acting on X^*. Then its nucleus is the self-similar closure of the union of all cycles in $\mathcal{A}(G)$.*

Proof. Let $\mathcal{N}$ denote the nucleus of G. If an element $g \in G$ belongs to some cycle in $\mathcal{A}(G)$, then it will be the section of itself at some level $N > 0$. But in this case it will also be a section of itself at each level kN for $k \geq 1$. Therefore each such element must be in the nucleus since all elements, including g, must contract to $\mathcal{N}$. Together with g by the same reason all of its sections must be in the nucleus. So $\mathcal{N}$ must contain the self-similar closure of all the cycles in $\mathcal{A}(G)$.

Conversely, if an element of G is not a section of itself at some vertex, then by minimality of $\mathcal{N}$ we cannot have $g \in \mathcal{N}$ (otherwise $\mathcal{N} - \{g\}$ would be a smaller set satisfying the definition of the nucleus as everything that contracts to g will also contract to its sections). □

8.3 Solving the word problem in polynomial time

One of the main algorithmic motivations to consider the class of contracting groups is the following theorem.

Theorem 8.5. *Let G be a contracting group. Then there exists a polynomial (in the length of the input word) time algorithm solving the word problem in G.*

Proof. We will hide technical details under the rug here to underline the idea. The details can be written out as an exercise. We will use the idea from the proof of Proposition 8.2.

Let G be a contracting group generated by a finite set S with the nucleus $\mathcal{N}$, and suppose that M is the largest length of elements of $\mathcal{N}$ with respect to S. Consider the following algorithm solving the word problem. Suppose we have a word w of length n in $S \cup S^{-1}$ and we want to check whether $w = 1$ in G or not.

Algorithm 8.6. *Solving the word problem in contracting groups in polynomial time.*

Step 1. *Precompute the level N at which all elements of G up to length $2M$ contract to $\mathcal{N}$.*

Step 2. *If g acts nontrivially on the N-th level, then return $g \neq 1$.*

Step 3. *Otherwise, write g as a product $g = g_1 g_2 \cdots g_k g_{k+1}$ of elements of length up to $2M$, where $k = \left\lfloor \frac{n}{2M} \right\rfloor$.*

Step 4. *According to the proof of Proposition 8.2, the sections of g at the vertices of level n will have the length at most $\frac{n}{2} + M$, so we can repeat again from step 2 until we get that all sections have length at most $2M$.*

Step 5. *If all the obtained sections are trivial, then return $g = 1$.*

The number of sections that one gets in step 5 of the algorithm will be $|X|^{Nk}$, where $k \approx \log_2 n$ is the number of times the length was shortened until it reached $2M$. Therefore, the complexity can be roughly estimated as

$$|X|^{Nk} = |X|^{N \log_2 n} = |X|^{\log_2 n^N} = |X|^{\frac{\log_{|X|} n^N}{\log_{|X|} 2}} = (n^N)^{\log_2 |X|} = n^{N \log_2 |X|}.$$

□

8.4 Contracting or noncontracting?

Now as we have good reasons to work with contracting groups, it is natural to ask whether we can actually detect the contracting property. Unfortunately, as happens in most of the cases in automaton groups, there is no known algorithm of verifying that a group is contracting or noncontracting that always works (and no proof that no such algorithm exists). However, if G is contracting, then there is a procedure that finds the nucleus of G thus proving the claim that G is contracting. This procedure will never stop if G happens to be noncontracting, unfortunately. On the other hand, there is quite an efficient procedure allowing us to prove that the group is non-contracting. Similarly, this procedure will never stop if the group is actually contracting. Unfortunately, it might not stop sometimes even if G is noncontracting. Both of these procedures are implemented in `AutomGrp` package in functions `FindNucleus` and `IsNoncontracting`, respectively.

To construct the procedure of finding the nucleus of contracting group we will use the next proposition similar in spirit to Proposition 8.2. For a set of automorphisms $A \subset \operatorname{Aut} X^*$ and a set V of vertices of X^* we denote by $A|_V$ the set of all sections $\{a|_v \colon a \in A, v \in V\}$.

Proposition 8.7 (Lemma 2.11.2 in [Nek05])**.** *A self-similar group G with a generating set $S = S^{-1}, 1 \in S$ is contracting if and only if there exists a finite set $\mathcal{N}$ and a number $k \geq 0$ such that*

$$\big((S \cup \mathcal{N})^2\big)|_{X^k} \subset \mathcal{N}. \tag{11}$$

Proof. If G is contracting, then all elements of G, including elements from $(S \cup \mathcal{N})^2$ must contract to $\mathcal{N}$. Conversely, if some finite set $\mathcal{N}$ satisfies the inclusion (11), then every element $g \in G$ can be written in the form $g = g_1 g_2 \cdot g_3 g_4 \cdots g_{2n-1} g_{2n}$ for some $g_i \in S \cup \mathcal{N}$ (i.e., $g \in (S \cup \mathcal{N})^{2n}$), which implies that the sections of g at the vertices of X^k will be in $(\mathcal{N})^n$. Applying induction argument we deduce that there must be some $m \geq 0$ such that $\{g\}|_{x^{km}} \subset \mathcal{N}$, which implies that G is contracting with the nucleus containing in $\mathcal{N}$. □

The above proposition can now be used to find the nucleus of a finitely generated contracting self-similar group G. Indeed, we can apply the following procedure (we do not call this procedure an algorithm since it will not stop in the case when G is non-contracting).

Procedure 8.8. *Finding the nucleus of a contracting group.*

Step 1. *Start with a finite self-similar generating set $S = S^{-1}, 1 \in S$ of G.*

Step 2. *For each element $st \in S^2$ check if there exists k such that $(st)|_{X^k} \subset S$. Since the length of all sections of each element $st \in S^2$ is bounded by 2, along each branch we will either obtain an element of S, or no later than at level $|S|^2 + 1$ get into a cycle obtaining another elements $s't' \in S^2$ such that $s't'$ is a section of itself at some (nonroot) vertex.*

Step 3. *If we have that for any $st \in S^2$ there is k such that $(st)|_{X^k} \subset S$, then by Proposition 8.7 G is contracting with the nucleus containing in S (namely, the nucleus is the self-similar closure of the union of all cycles of the automaton defining S by Proposition 8.4).*

Step 4. *Otherwise we will find an element $s't'$ that is a section of itself, so it (together with its self-similar closure) belongs to a cycle in $\mathcal{A}(G)$ and must be in the nucleus by Proposition 8.4. Therefore, we add the self-similar closure of $s't'$ to S and start over from Step 1.*

In the case when G is contracting, this procedure will terminate since the nucleus is finite and at each iteration of the procedure we find at least one element of it. If G is noncontracting, the procedure will never stop and will try bigger and bigger generating sets.

Thus, if a self-similar group G happens to be contracting, we will be able to find its nucleus and thus prove that it is indeed contracting. But how do we now check that G is noncontracting? Unfortunately, there is no general answer to this question and most known examples of non-contracting groups are shown to be noncontracting by the following method.

Proposition 8.9. *Suppose a self-similar group G contains an element g for which there is a vertex $v \in X^*$ such that*

1. *$g(v) = v$,*
2. *$g|_v = g$,*
3. *g has infinite order.*

Then G is noncontracting.

Proof. The first two conditions imply that $(g^n)|_v = g^n$, so g^n belongs to a cycle in the full automaton $\mathcal{A}(G)$ of G. Since g has infinite order, the union of all cycles of $\mathcal{A}(G)$ is infinite and by Proposition 8.4 the group G is noncontracting. □

Finding such an element g and a vertex v is often a quite hard task, especially because there is no general algorithm that decides whether a given element of a group has finite or infinite order. The partial procedure for search of such elements is implemented in `IsNoncontracting` function in `AutomGrp` package.

Another class of groups which are known to be non-contracting are infinite groups generated by reversible automata (see, for example, [SV11] for definitions). These are the automata in which every letter of the alphabet defines a permutation on the set of states (i.e., the functions $\pi_x \colon Q \to Q$, $\pi_x(q) = \pi(q, x)$ are bijections). However, it is often hard to algorithmically determine whether a group from this class is infinite.

In the next subsection we will discuss several examples of contracting and noncontracting groups.

	1	a	b	c	d
1	$(1,1)$	$(1,1)\sigma$	(a,c)	(a,d)	$(1,b)$
a	$(1,1)\sigma$	$(1,1)$	$(c,a)\sigma$	$(d,a)\sigma$	$(b,1)\sigma$
b	(a,c)	$(a,c)\sigma$	$(1,1)$	$(1,b)$	(a,d)
c	(a,d)	$(a,d)\sigma$	$(1,b)$	$(1,1)$	(a,c)
d	$(1,b)$	$(1,b)\sigma$	(a,d)	(a,c)	$(1,1)$

Table 1: Contracting table for the Grigorchuk group

8.5 Examples

Let us start from a trivial observation that all finite self-similar groups are contracting simply by definition. Therefore, we will talk only about infinite groups below.

Adding Machine. Recall from example 3.4 that the adding machine is a group $A \cong \mathbb{Z}$ generated by the automaton with the wreath recursion:

$$\begin{aligned} 1 &= (1,1), \\ t &= (1,t)\sigma, \end{aligned}$$

To check that it is contracting we start from the symmetric generating set $S = \{1, t, t^{-1}\}$ containing 1 and verify that the elements of S^2 contract to S. Indeed, since $S^2 = \{1, t, t^{-1}, t^2, t^{-2}\}$ and the first three elements of S^2 are already in S, we only need to check this for t^2 and t^{-2}. The calculation

$$\begin{aligned} t^2 &= (t,t), \\ t^{-2} &= (t^{-1}, t^{-1}), \end{aligned}$$

shows that $(S^2)|_X \subset S$ and, hence, A is contracting with the nucleus contained in S by Proposition 8.7. Since each element of S is contained is a section of itself, S coincides with the nucleus of A.

Grigorchuk group. Similarly to the adding machine example, we will see that the Grigorchuk group G is contracting with the nucleus coinciding with the automaton generators. Since the generators a, b, c, d are involutions, the set $S = \{1, a, b, c, d\}$ is a symmetric generating set of G containing 1. The contraction of all elements of S^2 can be conveniently recorded in the, so-called, *contracting table* shown in Table 1. This table serves as the proof that Grigorchuk group is contracting with the nucleus $S = \{1, a, b, c, d\}$.

Basilica group. Another example of a contracting group is, so-called, Basilica group $\mathcal{B}$. This group is generated by a 3-state automaton shown in Figure **??** with the following wreath recursion:

$$\begin{aligned} a &= (b,1)\sigma, \\ b &= (a,1), \end{aligned}$$

We will study this group later in the text in connection to iterated monodromy groups. Namely, it is isomorphic to the iterated monodromy group of a complex map $z \mapsto z^2 - 1$, whose Julia set resembles the St. Mark's Basilica in Venice (this explains the name of the group). Additionally, basilica group was the first example of an amenable, but not subexponentually amenable group.

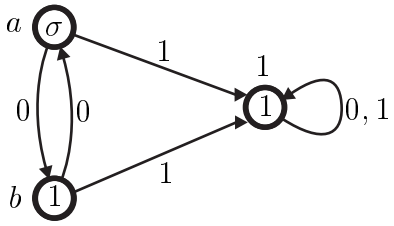


Figure 9: Automaton generating the basilica group

To find the nucleus we, as before, start from a symmetric generating set $S = \{1, a, b, a^{-1}, b^{-1}\}$ that contains the identity, and check whether S^2 contracts to S. Direct calculations gathered in table below describing the sections of the elements from S^2 on the second level show that all the pairs contract to S except $a^{-1}b$ and $b^{-1}a$.

	1	a	b	a^{-1}	b^{-1}
1	$[1,1,1,1]$	$[a,1,1,1]$	$[b,1,1,1]$	$[1,1,a^{-1},1]$	$[1,b^{-1},1,1]$
a	$[a,1,1,1]$	$[a,1,a,1]$	$[a,1,b,1]$	$[1,1,1,1]$	$[a,1,1,b^{-1}]$
b	$[b,1,1,1]$	$[b,a,1,1]$	$[b,b,1,1]$	$[b,1,a^{-1},1]$	$[1,1,1,1]$
a^{-1}	$[1,1,a^{-1},1]$	$[1,1,1,1]$	$[1,1,a^{-1}b,1]$	$[a^{-1},1,a^{-1},1]$	$[1,1,a^{-1},b^{-1}]$
b^{-1}	$[1,b^{-1},1,1]$	$[1,b^{-1}a,1,1]$	$[1,1,1,1]$	$[1,b^{-1},a^{-1},1]$	$[b^{-1},b^{-1},1,1]$

But this table also shows that $a^{-1}b$ has itself as a section at vertex 10 and $b^{-1}a = (a^{-1}b)^{-1}$ has itself as a section at vertex 01. Therefore, if $\mathcal{B}$ is contracting, then both of these elements must be in the nuclues by Proposition 8.4. According to Procedure 8.8 we add them to the current set S and start the same process with the new set

$$S_1 := \{1, a, b, a^{-1}, b^{-1}, a^{-1}b, b^{-1}a\}.$$

This time we obtain the following table showing that all elements from S_1^2 contract to S_1 on the second level, thus proving that $\mathcal{B}$ is contracting and the S_1 contains the nucleus of $\mathcal{B}$.

	1	a	b	a^{-1}	b^{-1}	$a^{-1}b$	$b^{-1}a$
1	$[1,1,1,1]$	$[a,1,1,1]$	$[b,1,1,1]$	$[1,1,a^{-1},1]$	$[1,b^{-1},1,1]$	$[1,1,a^{-1}b,1]$	$[1,b^{-1}a,1,1]$
a	$[a,1,1,1]$	$[a,1,a,1]$	$[a,1,b,1]$	$[1,1,1,1]$	$[a,1,1,b^{-1}]$	$[b,1,1,1]$	$[a,1,1,b^{-1}a]$
b	$[b,1,1,1]$	$[b,a,1,1]$	$[b,b,1,1]$	$[b,1,a^{-1},1]$	$[1,1,1,1]$	$[b,1,a^{-1}b,1]$	$[a,1,1,1]$
a^{-1}	$[1,1,a^{-1},1]$	$[1,1,1,1]$	$[1,1,a^{-1}b,1]$	$[a^{-1},1,a^{-1},1]$	$[1,1,a^{-1},b^{-1}]$	$[a^{-1}b,1,a^{-1},1]$	$[1,1,a^{-1},b^{-1}a]$
b^{-1}	$[1,b^{-1},1,1]$	$[1,b^{-1}a,1,1]$	$[1,1,1,1]$	$[1,b^{-1},a^{-1},1]$	$[b^{-1},b^{-1},1,1]$	$[1,b^{-1},a^{-1}b,1]$	$[b^{-1}a,b^{-1},1,1]$
$a^{-1}b$	$[1,1,a^{-1}b,1]$	$[1,1,a^{-1}b,a]$	$[1,1,a^{-1}b,b]$	$[a^{-1},1,a^{-1}b,1]$	$[1,1,a^{-1},1]$	$[a^{-1}b,1,a^{-1}b,1]$	$[1,1,1,1]$
$b^{-1}a$	$[1,b^{-1}a,1,1]$	$[1,b^{-1}a,a,1]$	$[1,b^{-1}a,b,1]$	$[1,b^{-1},1,1]$	$[1,b^{-1}a,1,b^{-1}]$	$[1,1,1,1]$	$[1,b^{-1}a,1,b^{-1}a]$

Virtually $\mathbb{Z}^3$. In the next example we show using computer that some contracting groups defined by automata with small number of states may have quite large nuclei. Consider a group defined by the following wreath recursion:

$$\begin{aligned} a &= (b,b)\sigma, \\ b &= (c,a), \\ c &= (a,a). \end{aligned}$$

In the classification of groups generated by 3-state automata over 2-letter alphabet [BGK+08] it is shown that it contains a subgroup of finite index isomorphic to $\mathbb{Z}^3$. Since in the mentioned classification it has number 752, we will denote it here by G_{752}. The following `AutomGrp` calculations show that G_{752} is contracting with the nucleus consisting of 41 elements. To illustrate the work of the Procedure 8.8 we use `SetInfoLevel` command in `GAP` to produce more output describing intermediate calculations.

```
gap> G:=AutomatonGroup("a = (b,b)(1,2), b = (c,a), c=(a,a)");
< a, b, c >
gap> SetInfoLevel(InfoAutomGrp,3);
gap> GroupNucleus(G);
#I  Trying generating set with 4 elements
#I  Elements added:[ b*a, c*b, a*c, a*b, b*c, c*a ]
#I  Trying generating set with 10 elements
#I  Elements added:[ c*b*a, a*c*b, b*a*c, a*b*a, b*a*b, a*b*c, b*c*a, c*a*b,
  b*c*b, c*a*c ]
#I  Trying generating set with 20 elements
#I  Elements added:[ a*c*b*a, b*a*c*b, a*b*a*c, (b*a)^2, c*b*a*b, a*b*c*b,
  b*c*a*c, c*a*b*a, (a*b)^2, b*a*b*c, a*b*c*a, b*c*a*b,
  b*c*b*a, c*a*c*b ]
#I  Trying generating set with 34 elements
#I  Elements added:[ a*b*a*c*b, (b*a)^2*c, c*(b*a)^2, a*c*b*a*b, b*a*c*b*a,
  a*b*c*b*a, b*c*a*c*b ]
#I  Trying generating set with 41 elements
[ 1, a, b, c, b*a, c*b, a*c, a*b, b*c, c*a, c*b*a, a*c*b,
  b*a*c, a*b*a, b*a*b, a*b*c, b*c*a, c*a*b, b*c*b, c*a*c,
  a*c*b*a, b*a*c*b, a*b*a*c, (b*a)^2, c*b*a*b, c*a*b*a,
  (a*b)^2, b*a*b*c, a*b*c*a, b*c*a*b, a*b*c*b, b*c*a*c,
  b*c*b*a, c*a*c*b, a*b*a*c*b, (b*a)^2*c, c*(b*a)^2,
  a*c*b*a*b, b*a*c*b*a, a*b*c*b*a, b*c*a*c*b ]
```

The lines starting with `Elements added` contain the elements that belong to cycles in the automaton of the group and their sections. One such line is printed for each iteration of Procedure 8.8 (more precisely, every time an element that is in the cycle is found, the procedure in `AutomGrp` adds it and its sections to the candidate for the nucleus and starts the process over).

Noncontracting groups. We finish this subsection with two examples of noncontracting groups. In both cases our main tool will be Proposition 8.9. In the first example we will take a contracting Adding machine and will add one extra state to ruin the contraction. Consider the extension of the lamplighter with the following wreath recursion:

$$\begin{aligned} 1 &= (1,1), \\ t &= (1,t)\sigma, \\ s &= (s,t). \end{aligned}$$

We already know from Example 3.4 that the generator t of the adding machine has infinite order. Since $s^n = (s^n, t^n) \neq 1$ for $n \neq 0$ we immediately obtain that s also has infinite order. Finally, since $s(0) = 0$ and $s|_0 = s$, all three conditions of Proposition 8.9 are satisfied and we deduce that this group is noncontracting.

For some noncontracing groups, however, it seems imperative to use a computer to obtain the proof of noncontractness. For example, the group G_{744} from the (3,2)-classification [BGK+08] has the following wreath recursion:

$$\begin{aligned} a &= (c, b)\sigma, \\ b &= (b, a), \\ c &= (a, a). \end{aligned}$$

The proof that this group is noncontracing can be generated by the `Isnoncontracting` command in `AutomGrp`. As before, we use `SetInfoLevel` command to actually generate the proof.

```
gap> G:=AutomatonGroup("a = (c,a)(1,2), b = (a,b), c=(b,a)");
< a, b, c >
gap> SetInfoLevel(InfoAutomGrp,3);
gap> IsNoncontracting(G);
#I  There are 37 elements of length up to 2
#I  There are 187 elements of length up to 3
#I  There are 937 elements of length up to 4
#I  There are 4687 elements of length up to 5
#I  b*c^-1 is obtained from (a*b*c*b^-1*a^-1*b^-1)^2
    by taking sections and cyclic reductions at vertex [ 1, 1, 1, 1, 1, 1, 1, 1 ]
#I  c*b^-1 is obtained from (b*c^-1)^2
    by taking sections and cyclic reductions at vertex [ 1, 1 ]
#I  a*b*c*b^-1*a^-1*b^-1 has a*b*c*b^-1*a^-1*b^-1 as a section at vertex [ 1, 1, 1, 2 ]
true
```

The output above should be read as follows. The program is searching for an element g satisfying the conditions of Proposition 8.9. It enumerates all elements of the group by length, and for each element g searches (for a bounded depth) for a vertex v satisfying conditions 1 and 2. For each such occurrence it then tries to verify that g has infinite order. In the example above the element $g = abcb^{-1}a^{-1}b^{-1}$ satisfies $g(0001) = 0001$ and $g|_{0001} = g$. The last condition that g has infinite order is verified by the "Order by sections" algorithm from Theorem 6.1.

Unfortunately, Proposition 8.9 has some limitations since the group must at least contain elements of infinite order. Let us formulate the following open question.

Problem 8.10. *Is it true that every periodic subgroup of* $\operatorname{Aut} X^*$ *is contracting?*

More generally, the answer to the following question would certainly be influential.

Problem 8.11. *Find an algorithm deciding whether a given automaton group is contracting.*

8.6 Contracting groups have no free nonabelian subgroups

We would like to state a beautiful corollary from a result obtained by Nekrashevych in [Nek10] regarding the free subgroups in groups acting faithfully on locally finite rooted trees.

Theorem 8.12 ([Nek10])**.** *Contracting groups do not contain free nonabelian subgroups.*

References

[Bar24] L. Bartholdi. FR, computations with functionally recursive groups, Version 2.4.13. `https://gap-packages.github.io/fr`, Jan 2024. GAP package.

[BGK+08] Ievgen Bondarenko, Rostislav Grigorchuk, Rostyslav Kravchenko, Yevgen Muntyan, Volodymyr Nekrashevych, Dmytro Savchuk, and Zoran Šunić. On classification of groups generated by 3-state automata over a 2-letter alphabet. *Algebra Discrete Math.*, (1):1–163, 2008. (available at *http://arxiv.org/abs/0803.3555*).

[Deh11] M. Dehn. Über unendliche diskontinuierliche Gruppen. *Math. Ann.*, 71(1):116–144, 1911.

[EZ20] Anna Erschler and Tianyi Zheng. Growth of periodic grigorchuk groups. *Inventiones Mathematicae*, 219(3):1069–1155, 2020.

[GAP24] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.14.0*, 2024.

[GNS00] R. I. Grigorchuk, V. V. Nekrashevich, and V. I. Sushchanskiĭ. Automata, dynamical systems, and groups. *Tr. Mat. Inst. Steklova*, 231(Din. Sist., Avtom. i Beskon. Gruppy):134–214, 2000.

[GP08] Rostislav Grigorchuk and Igor Pak. Groups of intermediate growth: an introduction. *Enseign. Math. (2)*, 54(3-4):251–272, 2008.

[Gri80] R. I. Grigorchuk. On Burnside's problem on periodic groups. *Funktsional. Anal. i Prilozhen.*, 14(1):53–54, 1980.

[Gri83] R. I. Grigorchuk. On the Milnor problem of group growth. *Dokl. Akad. Nauk SSSR*, 271(1):30–33, 1983.

[Gri84] R. I. Grigorchuk. Degrees of growth of finitely generated groups and the theory of invariant means. *Izv. Akad. Nauk SSSR Ser. Mat.*, 48(5):939–985, 1984.

[Gro81] Mikhail Gromov. Groups of polynomial growth and expanding maps. *Publications Mathématiques de l'IHÉS*, 53:53–78, 1981.

[KPS16] Ines Klimann, Matthieu Picantin, and Dmytro Savchuk. Orbit automata as a new tool to attack the order problem in automaton groups. *J. Algebra*, 445:433–457, 2016.

[Lys85] I. G. Lysënok. A set of defining relations for the Grigorchuk group. *Mat. Zametki*, 38(4):503–516, 634, 1985.

[Mil68] J. Milnor. Problem 5603. *Amer. Math. Monthly*, 75:685–686, 1968.

[MS25] Y. Muntyan and D. Savchuk. *AutomGrp – `GAP` package for computations in self-similar groups and semigroups, Version 1.3.3*, 2025. Accepted `GAP` package (available at *http://www.gap-system.org/Packages/automgrp.html*).

[Nek05] Volodymyr Nekrashevych. *Self-similar groups*, volume 117 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2005.

[Nek10] Volodymyr Nekrashevych. Free subgroups in groups acting on rooted trees. *Groups Geom. Dyn.*, 4(4):847–862, 2010.

[SS05] P. V. Silva and B. Steinberg. On a class of automata groups generalizing lamplighter groups. *Internat. J. Algebra Comput.*, 15(5-6):1213–1234, 2005.

[SS16] Dmytro M. Savchuk and Said N. Sidki. Affine automorphisms of rooted trees. *Geom. Dedicata*, 183:195–213, 2016.

[SV11] Dmytro Savchuk and Yaroslav Vorobets. Automata generating free products of groups of order 2. *J. Algebra*, 336(1):53–66, 2011.

[ŠV12] Zoran Šunić and Enric Ventura. The conjugacy problem in automaton groups is not solvable. *J. Algebra*, 364:148–154, 2012.