Equations in Self-Similar Groups

Dissertation
zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades

"Doctor rerum naturalium"

der Georg-August-Universität Göttingen

im Promotionsprogramm Mathematical Science
der Georg-August University School of Science (GAUSS)

vorgelegt von
Thorsten Groth
aus Heide.
Göttingen, 2017

Betreuungsausschuss

Professor Dr. Laurent Bartholdi,
Mathematisches Institut,
Georg-August-Universität Göttingen

Dr. Vadim Alekseev,
Institut für Geometrie,
Technische Universität Dresden

Mitglieder der Prüfungskommission

Referent: Professor Dr. Laurent Bartholdi

Korreferent: Dr. Vadim Alekseev

Weitere Mitglieder der Prüfungskommission

Professor Dr. Max Wardetzky,
Institut für Numerische und Angewandte Mathematik,
Georg-August-Universität Göttingen

Professor Dr. Carsten Damm,
Institut für Informatik,
Georg-August-Universität Göttingen

Professorin Dr. Bettina Eick,
Institut for Computational Mathematics,
Technische Universität Braunschweig

Professor Dr. Markus Lohrey,
Department für Elektrotechnik und Informatik,
Universität Siegen

Tag der mündlichen Prüfung: 06.02.2018

Für Daniela.

Contents

CHAPTER 1

Introduction

1. Aspirations

When we want to understand algebraic objects we study which properties they satisfy and how they act on some natural structures. The objects we want to understand in this thesis are certain invertible mappings of words over a finite fixed alphabet. We consider such mappings that preserve the length of the words and have the property that the image of a prefix is again a prefix of the mapped word. These objects are called *tree automorphisms* because we can write every word on a tree starting with one-letter-words in the first level, two-letter-words in the second level and so on, connecting a word to its immediate prefix (see Figure 1). A tree automorphism permutes these words while making sure that connected words stay connected.

We shall consider in this thesis groups of tree automorphisms; namely sets of tree automorphisms that are closed under composition and inverses. The algebraic structures we consider are therefore groups given with an action on the set of words.

Which properties do such groups enjoy? For this we consider equations in these groups: Formulas with unknowns, equalities and constants from the group. For example a group G satisfies the property that every element has a square-root if "$\forall g \in G \exists x \in G : x^2 = g$" holds. Namely the equation $X^2 g^{-1} = 1$ is solvable for all g.

For a specific class C of equations and a collection of tree automorphisms G the *Diophantine problem* for C in G asks whether there exists an algorithm that decides for every equation in C if there exists a solution with elements in G.

To have an algorithm that decides if an equation is solvable we need in particular a finite description of the tree automorphism. There are uncountably many different tree automorphisms and we can hence not describe every such automorphism by finite data. Thus we focus on special tree automorphisms that have a certain *self-similarity* structure.

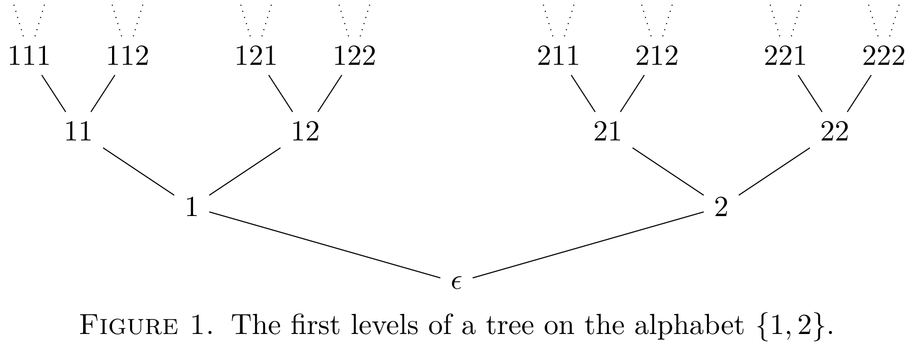


FIGURE 1. The first levels of a tree on the alphabet $\{1, 2\}$.

For some classes C of equations the Diophantine problems for C are classic decision problems in algebraic structures. For example to decide whether the equations of type $x^{-1}gx = h$ have a solution for fixed elements g and h is known as the conjugacy problem or "*Transformationsproblem*" in the original text by Max Dehn (see [**Deh11**]).

For general groups even for *finitely presented* ones this problem is known to be undecidable. For the *self-similar* groups we're considering in this text this particular decision problem is already rather well studied. It is known to be solvable in the group of so called *bounded* tree automorphisms (see [**BBSZ13**]) and for some very prominent examples of self similar groups: A. Rozhkov and Yu. Leonov proved this independently for the Grigorchuk group ([**Roz98**] and [**Leo98**]) and J. Wilson and P. Zalesskii proved this for the Gupta-Sidki group ([**WZ97**]). Moreover this problem is known to be solvable in a large class of so called *branch groups* (see [**GW00**]). On the other hand there are examples of self-similar groups that have undecidable conjugacy problem (see [**vSV12**]).

The aim of this thesis is to develop a toolkit to decide also other quadratic equations and to find conditions under which these tools work. It turns out that for explicitly given groups the question of solvability of some equations reduces to a large but finite number of calculations. In principle it would be possible but boring and easily erroneous to do the corresponding calculations by hand. We implemented some new methods for the computer algebra system GAP ([**GAP14**]) that help to do these calculations. On one hand this gives the opportunity to solve explicitly given examples and on the other to get an intuition for common patterns and to find proves that certain equations always have a solution.

We use the developed tools to compute the commutator width of some self-similar groups, i.e. the minimal number n such that every element of the derived subgroup G' is a product of n commutators. Furthermore we describe for some groups which other equations always have a solution.

2. Notations

We will denote the symmetric group on a set $\mathcal{A}$ by $\mathrm{Sym}(\mathcal{A})$ or if $\mathcal{A}$ is the set of natural numbers $\{1, \dots, n\}$ by S_n. Permutations in S_n will be denoted in cycle notation and act from the right. Thus for example $(1,2)(2,3) = (1,3,2)$. Analogously we will write $\mathrm{Alt}(\mathcal{A})$ and A_n for the alternating group.

The groups occurring in the text will often act from the right as well and we thus denote the conjugation of a group element g by an element h by $g^h := h^{-1}gh$ and denote the commutator of g and h by $[g,h] := g^{-1}h^{-1}gh$.

3. Self-similar groups

We briefly introduce self-similar groups as groups acting on a regular rooted tree by tree automorphisms. For a more detailed introduction into this topic and examples see Chapter 2. A good reference for this field of research is the book [**Nek05**] by V. Nekrashevych.

For a finite set $\mathcal{A}$ we will consider the free monoid $\mathcal{A}^*$ as an infinite regular rooted tree by defining the empty word to be the root and connect two words of length ℓ and $\ell - 1$ by an edge if the shorter word is a prefix of the longer. We then denote by $\mathrm{Aut}(\mathcal{A}^*)$ the group of all tree automorphisms of the tree $\mathcal{A}^*$. An automorphism $\alpha \in \mathrm{Aut}(\mathcal{A}^*)$ can be uniquely described by its action on the root and its action on the subtrees with roots $\mathcal{A}$ hanging from the root. In particular we have an isomorphism

$\Phi\colon \mathrm{Aut}(\mathcal{A}^*) \xrightarrow{\sim} \mathrm{Aut}(\mathcal{A}^*) \wr \mathrm{Sym}(\mathcal{A}) = \mathrm{Aut}(\mathcal{A}^*)^{|\mathcal{A}|} \rtimes \mathrm{Sym}(\mathcal{A})$. It is convenient to identify $\alpha \in \mathrm{Aut}(\mathcal{A}^*)$ with $\Phi(\alpha)$ and write $\alpha = \langle\!\langle \alpha_1, \dots, \alpha_n \rangle\!\rangle \sigma$. We will call the automorphisms α_i the *states* of α and σ the *activity* of α and write $\alpha @ i := \alpha_i$ and $\mathrm{act}(\alpha) := \sigma$.

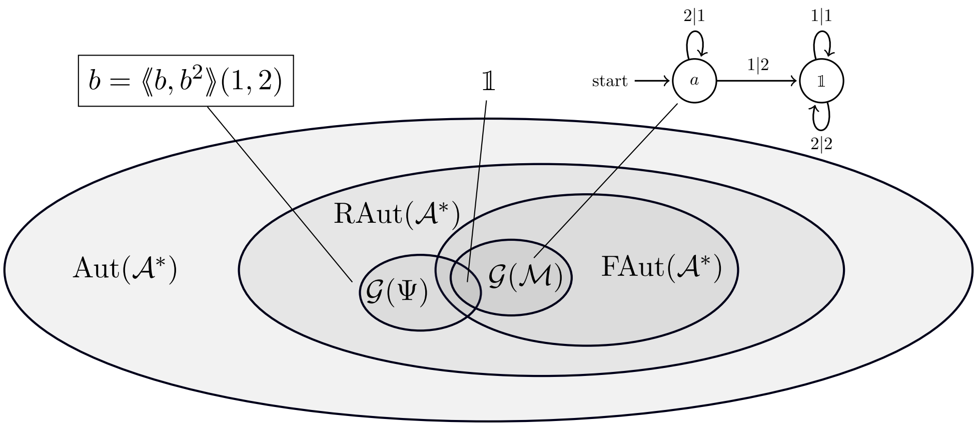
A group $G \leq \mathrm{Aut}(\mathcal{A}^*)$ is *self-similar* if Φ restricts to an embedding $\Phi\colon G \mapsto G \wr \mathrm{Sym}(\mathcal{A})$. In other words a self-similar group is a group G together with a free monoid $\mathcal{A}^*$, an action of G on $\mathcal{A}^*$ that preserves the word length and prefixes and an action @ of the monoid $\mathcal{A}^*$ on G satisfying the axiom $(gh)@x = (g@x)(h@x^g)$.

More generally if we fix a finite set S and consider the free group F_S on this generating set we can define a mapping called *self-similarity structure* $\Psi\colon S \times \mathcal{A} \to F_S \times \mathcal{A}$. Note that we only need finite data to specify such a mapping Ψ. We can extend the mapping to $F_S \times \mathcal{A}^*$ with the following recursive formulas for $\Psi = \Psi_s \times \Psi_a$:

$$\begin{aligned} \Psi_s(gh, w\ell) &= \Psi_s(g, w\ell) \cdot \Psi_s(h, \Psi_a(g, w\ell)) && \text{for } g \in F_S, h \in S \\ \Psi_a(gh, w\ell) &= \Psi_a(gh, w) \cdot \Psi_a(\Psi_s(gh, w), \ell) && \text{and } w \in \mathcal{A}^*, \ell \in \mathcal{A}. \end{aligned}$$

This way we obtain an action Ψ_a of the free group F_S on $\mathcal{A}^*$ by tree automorphisms. We define by $\mathcal{G}(\Psi)$ the subgroup of $\mathrm{Aut}(\mathcal{A}^*)$ that is the quotient of F_S by the kernel of these action. In case G is a finitely generated self-similar group with generating set S and thus a quotient of the free group F_S we obtain such a structure by setting $\Psi(g, x) = (g@x, x^g)$ for $g \in S$ and $x \in \mathcal{A}$. Then we have $\mathcal{G}(\Psi) = G$. A tree automorphism α is called *functional recursive* if there is a self-similarity structure Ψ such that $\alpha \in \mathcal{G}(\Psi)$. The set of all functional recursive tree automorphisms of a tree $\mathcal{A}^*$ form a self-similar group $\mathrm{RAut}(\mathcal{A}^*)$. A group generated by finitely many functional recursive elements will in general not be self-similar but can be embedded in a finitely generated self similar one. We will thus call subgroups of finitely generated self-similar groups *functional recursive groups*.

A tree automorphism $\alpha \in \mathrm{Aut}(\mathcal{A}^*)$ is *finite-state* if there is a finite subset $S \subset \mathrm{Aut}(\mathcal{A}^*)$ such that $\alpha \in S$ and for all $\beta \in S$ and $x \in \mathcal{A}$ we have $\beta @ x \in S$. An *automaton group* is a group that is finitely generated by finite state elements. Finite state automorphisms are in bijection to *Mealy machines* (or *Mealy transducers*) a variant of finite state automata. Given a Mealy machine $\mathcal{M}$ without the choice of an initial vertex we can define for every vertex v the machine $(\mathcal{M}, v)$ with initial state v. The group $\mathcal{G}(\mathcal{M})$ generated by this machines is then both an automaton group and a self-similar group.



$b = \langle\!\langle b, b^2 \rangle\!\rangle (1,2)$
$\mathbb{1}$
start
a
2|1
1|2
1|1
$\mathbb{1}$
2|2
$\mathrm{RAut}(\mathcal{A}^*)$
$\mathrm{Aut}(\mathcal{A}^*)$
$\mathcal{G}(\Psi)$
$\mathcal{G}(\mathcal{M})$
$\mathrm{FAut}(\mathcal{A}^*)$

FIGURE 2. Euler diagram of tree automorphisms, with some examples of self similar groups $\mathcal{G}(\Psi)$ and $\mathcal{G}(\mathcal{M})$.

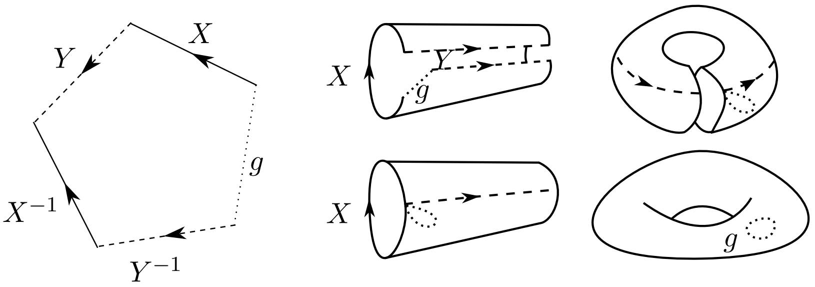


FIGURE 3. Gluing of the equation $[X,Y]g$ to a torus with a hole.

From the viewpoint of definability we can compare the full automorphism group $\operatorname{Aut}(\mathcal{A}^*)$ to the real numbers; most of them are not definable with finite data. The algebraic numbers are described finitely e.g. by the polynomial they satisfy. In this analogy the algebraic numbers correspond to the functional recursive automorphisms. In both cases this is not the maximal set of finitely definable objects. E.g. we can have the real number π or the automorphism that changes that positions of a word that are 1 in the infinite decimal representation of π. Finally the rational numbers correspond to finite state automorphisms as kind of finite objects.

We will often reduce a question about a group $G \leq \operatorname{Aut}(\mathcal{A}^*)$ of tree automorphisms to the same question about the states of the automorphisms. This is especially useful if the group G is *contracting*. That is: There is a finite set N called the *nucleus* of the group such that for every group element $g \in G$ the states $g@w$ lie in the set N for all words $w \in \mathcal{A}^*$ long enough.

4. Equations

We fix a set $\mathcal{X}$ and denote by $F_{\mathcal{X}}$ the free group with generating set $\mathcal{X}$. The elements of $\mathcal{X}$ will be called variables. For an arbitrary group G a *G-equation* is an element $\mathcal{E}$ of the free product group $F_{\mathcal{X}} * G$. A solution for a G-equation $\mathcal{E}$ is a homomorphism $\varphi\colon F_{\mathcal{X}} \to G$ with the property that $(\varphi * \mathrm{id}_G)(\mathcal{E}) = \mathbb{1}$. We can regard $\mathcal{E}$ as a reduced word over the alphabet $G \cup \mathcal{X} \cup \mathcal{X}^{-1}$. If for every variable $X \in \mathcal{X}$ that occurs in $\mathcal{E}$ also the variable X^{-1} occurs, we call $\mathcal{E}$ an oriented quadratic equation. If every variable occurs exactly twice then $\mathcal{E}$ is an unoriented quadratic equation. This terminology originates from the classification of surfaces. If we write a quadratic equation on the edges of a closed polygon and give the edge an orientation corresponding to the exponent ±1 of the generators X_i and then glue the edges of the polygon with same labels together we obtain a surface with holes. See Figure 3 for an example of this gluing. This analogy is used by M. Culler in [**Cul81**] to solve some equations in free groups.

The class of quadratic equations already covers many important cases: For example the question whether a group element g is a product of 2 commutators is equivalent to the question whether the equation $[X_1, X_2][X_3, X_4]g^{-1}$ has a solution. Or the question whether two group elements g and h are conjugate is equivalent to the question whether the equation $g^{X_1}h^{-1}$ has a solution. The theory of quadratic equations was first developed by R. Lyndon in [**Lyn59**] for equations in free groups. An equation with a linear component, i.e. it exists a variable X that occurs exactly once, is always solvable hence we do not consider equations with a linear term.

Sometimes we need a solution of an equation to fulfill a certain constraint that is a homomorphism γ from $F_{\mathcal{X}}$ to a finite quotient Q of G. Then the constrained equation $(\mathcal{E}, \gamma)$ has a solution s if s solves $\mathcal{E}$ and has the property that $s(X)$ lies in the residue class of $\gamma(X)$ for every $X \in \mathcal{X}$.

5. Main method

Let $\Phi\colon G \to G \wr \operatorname{Sym}(\mathcal{A})$ be a self-similar group and $\mathcal{E} \in F_{\mathcal{X}} * G$ be an equation in variables $\{X_1, \dots, X_n\} \subset \mathcal{X}$. Denote further by $\{Y_{i,j} \mid i = 1, \dots, n, j = 1, \dots, |\mathcal{A}|\} \subset \mathcal{X}$ some more variables. We choose a constraint $\sigma\colon F_{\mathcal{X}} \to \operatorname{Sym}(\mathcal{A})$ and define

$$\begin{aligned} \Phi_\sigma\colon F_{\mathcal{X}} * G &\to (F_{\mathcal{X}} * G) \wr S_n \\ X_i &\mapsto \langle\!\langle Y_{i,1}, \dots, Y_{i,|\mathcal{A}|} \rangle\!\rangle \sigma(X_i) \\ g &\mapsto \Phi(g). \end{aligned}$$

Then $\Phi_\sigma(\mathcal{E})$ is a system of $|\mathcal{A}|$ equations in the variables $\{Y_{i,j} \mid i, j\}$ and one additional element of $\operatorname{Sym}(\mathcal{A})$ e.g. $\Phi_\sigma(\mathcal{E}) = \langle\!\langle \mathcal{E}_1, \dots, \mathcal{E}_{|\mathcal{A}|} \rangle\!\rangle \tau_{\sigma,\mathcal{E}}$. If $\mathcal{E}$ is solvable with solution s then for $\sigma = \operatorname{act} \circ s$ we have $\tau_{\sigma,\mathcal{E}} = \mathbb{1}$ and the system $\Phi_\sigma(\mathcal{E})$ is solvable with solution $s'\colon Y_{i,j} \mapsto s(X_i)@j$.

On the other hand if there is such a homomorphism σ such that $\Phi_\sigma(\mathcal{E})$ is of trivial activity and is solvable by a solution s' then the equation $\mathcal{E}$ is solvable in $\operatorname{Aut}(\mathcal{A}^*)$ with solution $s\colon X_i \mapsto \langle\!\langle s'(X_{i,1}), \dots, s'(X_{i,|\mathcal{A}|}) \rangle\!\rangle \sigma(X_i)$.

Thus for layered self-similar groups G, i.e. a group such that the embedding Φ is an isomorphism, the solvability of $\mathcal{E}$ and the system $\Phi_\sigma(\mathcal{E})$ is the same.

For quadratic equations $\mathcal{E}$ it turns out that the system $\Phi_\sigma(\mathcal{E})$ is always equivalent to a system of pairwise independent quadratic equations.

If the group G is not layered but regular branched, that is there is a finite index normal subgroup $K \trianglelefteq G$ that fulfills $K^{|\mathcal{A}|} \leq \Phi(K)$, then we can in general not lift arbitrary solutions for $\Phi_\sigma(\mathcal{E})$ to a solution of $\mathcal{E}$.

But by choosing constraints γ not only to the symmetric group but to the quotient G/K we can find a constraint γ' for the system $\Phi_\gamma(\mathcal{E})$ such that a solution for the constrained system $(\Phi_\gamma(\mathcal{E}), \gamma')$ will admit a solution for the constrained equation $(\mathcal{E}, \gamma)$.

In Chapter 4 we will describe in detail some conditions under which this helps to solve equations.

6. Results

6.1. Layered Groups. We already remarked that if the group G is *layered* then every solution of $\Phi_\sigma(\mathcal{E})$ can be lifted back to a solution of $\mathcal{E}$. With that we can prove the following:

THEOREM 1.1. *Every element of the derived group of* $\operatorname{Aut}(\mathcal{A}^*)$ *is a commutator.*

Also more generally: We can define the distance between two tree automorphisms α and β by 2^{-n} where n is the largest level such that the action on the finite subtree up to level n of α and β is identical. With this length $\operatorname{Aut}(\mathcal{A}^*)$ becomes a profinite group as the inverse limit of the finite groups acting on the first n levels of the tree.

THEOREM 1.2. *If* $G \leq \operatorname{Aut}(\mathcal{A}^*)$ *is layered and complete and for every element g in the derived subgroup G' the permutation* $\operatorname{act}(g)$ *is a commutator then every element of the derived group G' is a commutator.*

The Neumann-Segal groups are examples of finitely generated layered groups. For $n \in \mathbb{N}$ these are groups $\mathfrak{N}_n$ acting on the n-ary tree by the following generators.

$$a_\pi = \langle\!\langle 1, \dots, 1 \rangle\!\rangle \pi, \qquad \alpha_\pi = \langle\!\langle a_\pi, \alpha_\pi, 1, \dots, 1 \rangle\!\rangle \qquad \text{for } \pi \in \mathrm{A}_n,$$
$$\mathfrak{N}_n = \langle \alpha_\pi, a_\pi \mid \pi \in \mathrm{A}_n \rangle .$$

For $n \geq 5$ the groups $\mathfrak{N}_n$ are layered and perfect. With a similar method as before we can prove:

Theorem 1.3. *For $n \geq 5$ every element in the group $\mathfrak{N}_n$ is a commutator.*

Furthermore the proof is constructive and we implemented an algorithm that, given an element $g \in \mathfrak{N}_n$ in the form of its representing Mealy machine, returns two elements of the group that satisfy the equation $[X, Y]g = \mathbb{1}$.
Moreover we prove that for every n there is a constant C_n such that every element of $\mathfrak{N}_n$ is a product of C_n conjugates of an arbitrary nontrivial permutation $\sigma \in \mathrm{A}_n$ and conclude the following theorem:

Theorem 1.4. *There is an algorithm that decides for every quadratic oriented equation in the Neumann-Segal group if a solution exists.*

A self-similar group is contracting if in the general case the states of elements are shorter than the element itself with respect to the word metric. We note that this is the key ingredient for the calculation in the Neumann-Segal group and prove more generally:

Theorem 1.5. *Every layered contracting self-similar group, where every element of the nucleus is a commutator, is of commutator width* 1.

6.2. Grigorchuk Group. The Grigorchuk group $\mathfrak{G}$ is a regular branched self-similar group acting on the binary tree generated by the four elements

$$a = \langle\!\langle \mathbb{1}, \mathbb{1} \rangle\!\rangle (1,2), \qquad b = \langle\!\langle a, c \rangle\!\rangle, \qquad c = \langle\!\langle a, d \rangle\!\rangle, \qquad d = \langle\!\langle \mathbb{1}, b \rangle\!\rangle.$$

The branching subgroup K is the normal closure of $\langle (ab)^2 \rangle$ and the quotient G/K is of order 16. Instead of pure equations in $\mathfrak{G}$ we consider equations with constraints $\gamma \colon G_{\mathcal{X}} \to \mathfrak{G}/K$. Elements of K have only trivial activity hence we can deduce the activity of an element $g \in \mathfrak{G}$ by its image in $\mathfrak{G}/K$ and thus define the homomorphism Φ_γ as before.
We use a result of I. Lysenok from [**LMU16**]) that there is a finite number of constraints Γ such that every constrained oriented quadratic equation is equivalent to a constrained equation $(\mathcal{E}, \gamma)$ with $\gamma \in \Gamma$. If we consider the equation $\mathcal{E} = [X_1, X_2] \cdots [X_{2n-1}, X_{2n}]g$ for $g \in \mathfrak{G}'$ then for most constraints γ the system $\Phi_\gamma(\mathcal{E})$ is solvable if an equation with higher genus $\mathcal{E}' = [X_1, X_2] \cdots [X_{4n-3}, X_{4n-2}]g'$ is solvable. We use the computer algebra software GAP ([**GAP14**]) to compute this set Γ explicitly and possible constraints for the equation $\mathcal{E}'$ to derive the following theorem:

Theorem 1.6. *The Grigorchuk group $\mathfrak{G}$ and its branching subgroup K has commutator width* 2.

With a similar method as before we see that for a good choice of constraints γ the equations $\Phi_\gamma(\prod_{i=1}^{6} a^{X_i} g)$ are equivalent to products of commutators and certain constraints and obtain the result that answers a question of Elisabeth Fink (see [**Fin14**]):

Corollary 1.7. *Every element of $\mathfrak{G}'$ is a product of* 6 *conjugates of the generator* a *and there are elements* $g \in \mathfrak{G}'$ *which are not products of* 4 *conjugates of* a.
Every element $g \in \mathfrak{G}$ *is a product of at most* 8 *conjugates of the generator* a.

To complete the picture of the commutator width in the group $\mathfrak{G}$ we use the description of the subgroup lattice in $\mathfrak{G}$ which can be found in [**BGŠ03**]) to obtain:

Theorem 1.8. *Every finitely generated subgroup of $\mathfrak{G}$ has finite commutator width; however, their commutator width cannot be bounded, not even among finite-index subgroups. Furthermore there is a subgroup of $\mathfrak{G}$ of infinite commutator width.*

6.3. Gupta-Sidki Group. The computations for the Grigorchuk group were quite explicit but the general method works well for other similar groups. We adapt the specific computations for the Gupta-Sidki 3 group and show:

Theorem 1.9. *The Gupta-Sidki group has commutator width at most* 2.

However in contrast to the case of the Grigorchuk group we could not prove that the commutator width of the Gupta-Sidki group is larger then one.

6.4. Order problem. S. Sidki introduced in [**Sid00**] the notion of *bounded* tree automorphisms: A tree automorphism $\alpha \in \mathrm{Aut}(\mathcal{A}^*)$ is *bounded* by a polynomial p if the number of elements in the n-th level of the tree $\mathcal{A}^*$ that are not fixed by α is bounded by $p(n)$. An automorphism that is bounded by a constant polynomial will be called *bounded* and a group $G \leq \mathrm{Aut}(\mathcal{A}^*)$ will be called bounded if it is generated by bounded elements.
It is well known that the order problem is solvable in bounded self-similar groups (see [**BBSZ13**]). On the other hand very recent results ([**Gil17, BM17**]) show that the order problem in finitely generated self-similar groups can be unsolvable.
L. Bartholdi, V. Kaimanovich and V. Nekrashevych introduced in [**BKN10**] the so called *mothergroup*. This is a finitely generated self-similar group that contains every bounded group as a subgroup. This notion was extended by G. Amir, O. Angel and B. Virág in [**GOB13**] to d-mothergroups $\mathcal{M}_{n,d}$ on an alphabet of size n. All groups bounded by a polynomial of degree d can be embedded into such a group $\mathcal{M}_{n,d}$ for some n, that is possibly larger then the alphabet of the original group. We can show:

Theorem 1.10. *The order problem in the mothergroup $\mathcal{M}_{2,1}$ is decidable.*

This is a further indication that it could be true that the order problem is decidable in the group $\mathrm{Poly}(\infty)$, the group of all tree automorphisms bounded by a polynomial of any degree.

7. GAP

Some results of this text are inspired by experiments in the computer algebra system GAP ([**GAP14**]). The GAP package *fr* by L. Bartholdi already allows computations in self-similar groups (see [**Bar16b**]). I build upon it a set of tools to manipulate equation in them. In fact the GAP package I developed is more general and can be used to study equations in all kinds of groups.
This package is freely distributed under the terms of the GNU General Public License an extension to GAP. It is stored as an repository on *git-hub* under the url `https://github.com/ThGroth/gap-equations`. All examples in this thesis that refer to a file can be found in the directory `phd/` of this repository.

With this package it is possible to define an equation $\mathcal{E} \in F_{\mathbb{N}} * G$ stored internally as word over elements of G and variables.

```
gap> G := FreeGroup("g");;
gap> EqG := EquationGroup(G);;
gap> F := VariablesOfEquationGroup(EqG);
```

```
[ FreeProductElm([ X1 ]), FreeProductElm([ X2 ]), ... ]
```

```
gap> x1:=F[1];; x2:=F[2];; x3:=F[3];; g:=G.1;;
gap> eq := Equation(x1^-1*x2*g*x1*x3^2*g*x2^-1);
```

```
Equation in [ X1, X2, X3 ]
```

```
gap> Print(eq);
```

```
FreeProductElm([ X1^-1*X2, g, X1*X3^2, g, X2^-1 ])
```

One of the main features of the package is for now its implementation of a normal form for quadratic equations.

```
gap> IsQuadraticEquation(eq);
```

```
true
```

```
gap> nf := NormalFormOfEquation(eq);
```

```
<Equation in [ X1, X2, X3 ]>
```

```
gap> Print(nf);
```

```
FreeProductElm([ X1^2*X2^2*X3^2, g^2 ])
```

```
gap> eq^NormalizingHomomorphism(nf)=nf;
```

```
true
```

For a self-similar group G, an equation $\mathcal{E} \in F_{\mathbb{N}} * G$ and a homomorphism $\sigma\colon F_{\mathbb{N}} \to \mathrm{Sym}(\mathcal{A})$ we have the system of equations $\Phi_\sigma(\mathcal{E})$. We can compute this new system of equations with the method `DecompositionEquation` and for a quadratic equation the equivalent system that consists again of quadratic equations:

```
gap> G := GrigorchukGroup;; a:=G.1;; b:=G.2;;
gap> EqG := EquationGroup(G);;
gap> eq := Equation(Comm(EqG.5,EqG.6)*b*b^a);;
gap> deq := DecompositionEquation(eq,[(),(1,2)]);
```

```
DecomposedEquation in [ Xn1, Xn2, Xn3, Xn4 ]
```

```
gap> djf := DisjointFormOfDecomposedEquation(deq);
gap> Sys := EquationComponents(djf);
```

```
[ <Equation in [ Xn2, Xn3, Xn4 ]>, <Equation in [  ]> ]
```

```
gap> deq^DisjointFormHomomorphism(djf)=djf;
```

```
true
```

With this methods we have an environment in which we can implement some algorithms to solve equations. One such algorithm is implemented in the example file examples/NeumannSegal.g:

```
gap> Read("examples/NeumannSegal.g");
gap> Nn := NeumannSegalGroup(5);
```

```
<recursive group over [ 1 .. 5 ] with 4 generators>
```

```
gap> Nn := NeumannSegalGroup(5);;
gap> eq := RandomQuadraticEquation(EquationGroup(Nn),10);
```

```
<Equation in [ X1, X3, X4, X5, X9 ]>
```

```
gap> sol := SolveEquation(eq);;
gap> List(EquationVariablesEmbedded(eq),x->x^sol);
```

```
[ <Mealy element on alphabet [ 1 .. 5 ] with 60 states>,
<Mealy element on alphabet [ 1 .. 5 ] with 2 states>,
<Mealy element on alphabet [ 1 .. 5 ] with 36 states>,
<Mealy element on alphabet [ 1 .. 5 ] with 7 states>,
<Mealy element on alphabet [ 1 .. 5 ] with 7 states> ]
```

```
gap> IsSolution(sol,eq);
```

```
true
```

For more examples for what is possible with the *equations* package see Chapter 5, Section 3 and the manual in Appendix B.

CHAPTER 2

Groups of tree automorphisms

For convenience of the later proofs we will define the general setup for self-similar group. Groups generated by automata are already studied since the 60's (see for example [**Gs61**]) and gained larger attention since R. Grigorchuk defined the now famous Grigorchuk group (see [**Gri80**]) as a group of measure preserving transformations of an interval, which soon turned out to be easy to describe in the language of automaton groups. For our definitions we follow roughly the book [**Nek05**] by V. Nekrashevych.

1. Trees

Definition 2.1. A graph will consist of a set V of *vertices* and a set $E \subset V \times V$ of edges or a set $E \subset V \times V \times L$ of labeled edges for a set L of labels.
In particular edges have a direction (and will hence sometimes called arrows) and we allow loops and multiple edges (with different labels).

Definition 2.2. A *one-rooted regular tree of degree d* is a tree where one vertex (the root) has valency d and all other vertices have valency $d+1$. The set of the vertices and edges can be described by its *levels*. The n-th *level* is the set of vertices with distance n to the root.
In this text all trees will be one-rooted and regular hence the term tree will always refer to an one-rooted regular tree.

Definition 2.3.

(1) The free monoid on a finite set $\mathcal{A}$ is notated as $\mathcal{A}^*$ and is called the set of *words* over the *alphabet* $\mathcal{A}$.
(2) The set $\mathcal{A}^\omega$ is the set of all infinite words.

Remark 2.4. $\mathcal{A}^*$ forms a tree of degree $|\mathcal{A}|$ with vertex set $\mathcal{A}^*$ and the set of edges equal to $\{(v, vx) \mid v \in \mathcal{A}^*, x \in \mathcal{A}\}$.

Definition 2.5. The length of a word $w \in \mathcal{A}^*$ is the minimal number of free generators which are necessary to form the word.

$$|w| = \min\{n \in \mathbb{N} \mid w = \prod_{i=1}^{n} x_i,\ x_i \in \mathcal{A}\}.$$

The length $|w|$ equals the level of w in the tree.

1.1. Automorphisms of rooted Trees and wreath products.

Definition 2.6. Let $T = (V, E)$ and $T' = (V', E')$ be trees. A mapping $f \colon V \to V'$ is called a tree homomorphism if for each edge $(v, w) \in E$ there is an edge $(f(v), f(w)) \in E'$.

Remark 2.7. If T and T' are two trees of the same degree, then they are isomorphic. Especially a subtree (a subset of the tree which is itself a tree) is isomorphic to the whole tree.

Definition 2.8. The group of all tree automorphisms of a tree T will be denoted by $\mathrm{Aut}(T)$. Because the automorphisms have a lot in common with permutations, we will consider their action on the tree as a right action.

Lemma 2.9. *The levels of a tree are invariant under a tree automorphism.*

Proof. Let a be a tree automorphism. The root is fixed by a because the valency is invariant under tree homomorphisms. Then by induction every level is fixed. □

The action of a tree automorphism on a finite proportion of the tree can be described by permutations. For a tree of degree d and a subgroup $G \leq \mathrm{Aut}(T)$ we denote by $\mathrm{Perm}_n(G) \hookrightarrow \mathrm{S}_{d^n}$ the subgroup of $\mathrm{Aut}(T)$ that acts on the n-th level of the tree like G does and that stabilizes all levels m for $m > n$. Further we denote by $\mathrm{Stab}_n(G)$ the subgroup of G that stabilizes the n-th level. The groups $\mathrm{Perm}_n(G)$ can be described by *wreath products*:

Definition 2.10. Let P and G be groups and $\mathcal{A}$ be a right P-set. Then the unrestricted *wreath product* is:

$$G \wr_{\mathcal{A}} P := G^{\mathcal{A}} \rtimes P \ ,$$

where the semi direct product is defined respective to the induced action of P on $G^{\mathcal{A}}$: $a \in P$ defines a map $a\colon \mathcal{A} \to \mathcal{A},\ x \mapsto x^a$. For $f\colon \mathcal{A} \to G$ define $f^a := a \circ f$.

In our purpose the set $\mathcal{A}$ will be finite and P be a subgroup of the symmetric group on $\mathcal{A}$. The set $G^{\mathcal{A}}$ is then the set of $|\mathcal{A}|$-tupels and the action of P on $G^{\mathcal{A}}$ is given by permutation of the indices. If the set $\mathcal{A}$ is clear from the context we will omit it from the notation and simply write $G \wr P$. Let us for now fix a set $\mathcal{A} = \{1, \dots, d\}$ and a group $P \leq \mathrm{S}_d$.
With $g_1, \dots, g_d \in G$ and $\sigma \in P$ we will denote an element of $G \wr P$ by $\langle\!\langle g_1, \dots, g_d \rangle\!\rangle \sigma$. Then the multiplication of two elements can be computed as following:

$$\langle\!\langle g_1, \dots, g_d \rangle\!\rangle \sigma \cdot \langle\!\langle h_1, \dots, h_d \rangle\!\rangle \tau = \langle\!\langle g_1 \cdot h_{(1)\sigma}, \cdots, g_d \cdot h_{(d)\sigma} \rangle\!\rangle \sigma \cdot \tau.$$

Let $T = \mathcal{A}^*$ be the rooted tree of degree d. For a group $G \leq \mathrm{Aut}(T)$ the wreath product $G \wr A$ acts on T by an element $g = \langle\!\langle g_1, \dots, g_d \rangle\!\rangle \sigma$ on a word $\ell_1 \dots \ell_n \in \mathcal{A}^*$ by:

$$(\ell_1 \dots \ell_n)^g = \ell_1^\sigma (\ell_2 \dots \ell_n)^{g_\ell} \text{ for all } \ell_i \in \mathcal{A},\ n \in \mathbb{N}.$$

Lemma 2.11. *Let T be the tree of degree d. We then have the following isomorphisms:*

$$\text{(1)} \qquad \mathrm{Perm}_{n+1}(\mathrm{Aut}(T)) \simeq \mathrm{Perm}_n(\mathrm{Aut}(T)) \wr \mathrm{S}_d \qquad \simeq \mathrm{S}_d \wr_{\mathcal{A}^n} \mathrm{Perm}_n(\mathrm{Aut}(T))$$

$$\text{(2)} \qquad \mathrm{Aut}(T) \simeq \varprojlim_n \mathrm{Perm}_n(\mathrm{Aut}(T)),$$

$$\text{(3)} \qquad \Phi\colon \mathrm{Aut}(T) \xrightarrow{\simeq} \mathrm{Aut}(T) \wr \mathrm{S}_d\,.$$

Proof. For $g \in \mathrm{Perm}_{n+1}(\mathrm{Aut}(T))$ there is a unique $\sigma \in \mathrm{S}_d$ such that $\ell^g = \ell^\sigma$ for all $\ell \in \mathcal{A}$ and furthermore for each $\ell \in \mathcal{A}$ there is a unique $g_\ell \in \mathrm{Perm}_n(\mathrm{Aut}(T))$ such that $(\ell v)^g = \ell^\sigma v^{g_\ell}$. Furthermore there is a unique $h \in \mathrm{Perm}_n(\mathrm{Aut}(T))$ and for $v \in \mathcal{A}^n$ there are $\sigma_v \in \mathrm{S}_d$ such that $(v\ell)^g = v^h \ell^{\sigma_v}$ for all $\ell \in \mathcal{A}$.
Hence the mappings $g \mapsto \langle\!\langle g_1, \dots, g_d \rangle\!\rangle \sigma$ and $g \mapsto \langle\!\langle \sigma_v \mid v \in \mathcal{A}^n \rangle\!\rangle h$ give rise to the claimed isomorphisms.
For the inverse limit we define the bonding maps to be the natural projections:

$$f_{n,n+1}\colon \mathrm{Perm}_{n+1}(\mathrm{Aut}(T)) \simeq \mathrm{S}_d \wr \mathrm{Perm}_n(\mathrm{Aut}(T)) \to \mathrm{Perm}_n(Aut(T)).$$

Hence $(g_i)_{i=1}^{\infty} \in \lim \operatorname{Perm}_n(\operatorname{Aut}(T)) < \prod_n \operatorname{Perm}_n$ if and only if we have $g_n = \langle\!\langle \cdots \rangle\!\rangle g_{n-1}$ for all $n \in \mathbb{N}$ and every tree automorphism can be written in that form. (3) then follows automatically. □

Remark 2.12. The inverse limit defines a topology on $\operatorname{Aut}(T)$ under that the group is complete. We can define the distance between elements $g, h \in \operatorname{Aut}(T)$ by the following construction: Let n be the maximal number m such that $gh^{-1} \in \operatorname{Stab}_m(\operatorname{Aut}(T))$ or $n = \infty$ if this holds for every m. Then $\operatorname{dist}(x, y) := 2^{-n}$.

Definition 2.13. Let $a \in \operatorname{Aut}(T)$ and Φ be as in the proof of the last lemma and $\Phi(a) = \langle\!\langle a_1, \dots, a_d \rangle\!\rangle \sigma_a$. We call the a_ℓ *states* of a and notate $a_\ell =: a@\ell$ and call σ_a the *activity* of a and write $\sigma_a =: \operatorname{act}(a)$. Furthermore for $w = \ell v \in T$ we define recursively $a@w = (a@l)@v$.

Corollary 2.14. *We have the following computation rules:*

$$(vw)^a = v^a w^{a@v}, \qquad (ab)@v = (a@v)(b@v^a), \qquad (a@v)^{-1} = a^{-1}@v^a,$$
$$(\ell v)^a = \ell^{\operatorname{act}(a)} v^{a@l}, \qquad \operatorname{act}(ab) = \operatorname{act}(a)\operatorname{act}(b), \qquad \operatorname{act}(a^{-1}) = \operatorname{act}(a)^{-1}$$

for all $\ell \in \{1, \dots, d\}$, $v, w \in T$, and $a, b \in \operatorname{Aut}(T)$.

Definition 2.15. A subgroup $G \leq \operatorname{Aut}(\mathcal{A}^*)$ is called *self-similar* (or *state closed*) if for all $g \in G$ and all $x \in \mathcal{A}$ we have $g@x \in G$. I.e all states of group members are also group members.
The embedding Φ then restricts to $\Phi\colon G \hookrightarrow G \wr \operatorname{Sym}(\mathcal{A})$.

1.2. Automata. The term *state* refers to an alternative way to define the tree automorphisms. We can define a tree automorphism by an invertible (possibly infinite) *Mealy machine*.

Definition 2.16. A Mealy machine is a certain type of a state automaton and consist of the following data:

(1) A finite set $\mathcal{A}$ called the *alphabet*.
(2) A nonempty set $\mathcal{S}$, the so called set of *states*.
(3) A *transition map* $t = t_A \times t_S\colon \mathcal{A} \times \mathcal{S} \to \mathcal{A} \times \mathcal{S}$.
(4) An element $s_0 \in \mathcal{S}$, the so called *start state*.

If the set S is finite we will call the machine *finite-state*.

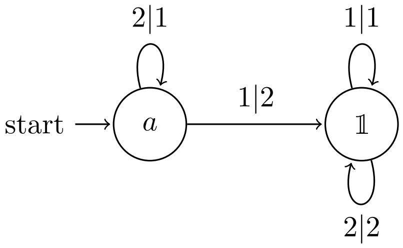
We will often depict a finite-state Mealy machine by a graph with vertex set S. Whenever $t(\ell, s) = (k, t)$ we will draw an arrow from s to t with label "$\ell|k$". The start state will be indicated by an incoming edge labeled *start*.
Let $\mathfrak{M} = (\mathcal{A}, \mathcal{S}, t_A \times t_S, s)$ be a Mealy machine. The transition map extends to $\mathcal{A}^* \times S$ by the recursive definition:

$$\begin{aligned} \tilde{t} = \tilde{t_A} \times \tilde{t_S}\colon \mathcal{A}^* \times \mathcal{S} &\to \mathcal{A}^* \times \mathcal{S} \\ (\ell v, s) &\mapsto t_A(\ell, s) t_A(v, t_S(\ell, s)) \qquad \text{for } \ell \in \mathcal{A} \text{ and } v \in \mathcal{A}^*. \end{aligned}$$

The action of $\mathfrak{M}$ on $v \in \mathcal{A}^*$ is then defined to be $\tilde{t}_A(v, s_0)$ which by definition fixes the levels of the tree.

Example 2.17. Consider the following automaton $\mathfrak{M}$ called the *adding machine*:

$$\mathfrak{M} = \left(\{1,2\}, \{a, \mathbb{1}\}, \begin{array}{ll} (1,a) & \mapsto (2,\mathbb{1}), \\ (2,a) & \mapsto (1,a), \\ (1,\mathbb{1}) & \mapsto (1,\mathbb{1}), \\ (2,\mathbb{1}) & \mapsto (2,\mathbb{1}), \end{array} , a \right),$$



The action of the automaton replaces all occurrences of twos by ones until it reads the first one. If one reads the word from right to left and replaces the alphabet by $\{0,1\}$ the automaton gives the result of a binary addition of one.

1.3. Minimal Automata. Two formally different automata can define the same action on the tree. We want to have a faithful action on the tree and hence define *equivalent automata* and the *minimal automaton*.

DEFINITION 2.18. Two Mealy machines $\mathfrak{M}_1$, $\mathfrak{M}_2$ with the same alphabet are equivalent if their corresponding actions on the tree are identical.
In an equivalence class of Mealy machines, a machine is called *minimal machine* or *minimal automaton* if its set of states has minimal size.

DEFINITION 2.19. Let $\mathfrak{M} = (\mathcal{A}, \mathcal{S}, t = t_A \times t_S, s_0)$ and $\mathfrak{M}' = (\mathcal{A}, \mathcal{S}', t' = t'_A \times t'_S, s'_0)$ be equivalent Mealy machines. A map $\varphi\colon \mathcal{S} \to \mathcal{S}'$ is called a *state homomorphism* if

(1) $\varphi(s_0) = s'_0$
(2) $\varphi(t_S(a,s)) = t'_S(a, \varphi(s))$ for all $a \in \mathcal{A}$ and $s \in \mathcal{S}$.

If there is an inverse state homomorphism, φ is called a *state isomorphism* and the two machines are isomorphic.

REMARK 2.20. If φ is bijective, the inverse is automatically a state homomorphism.

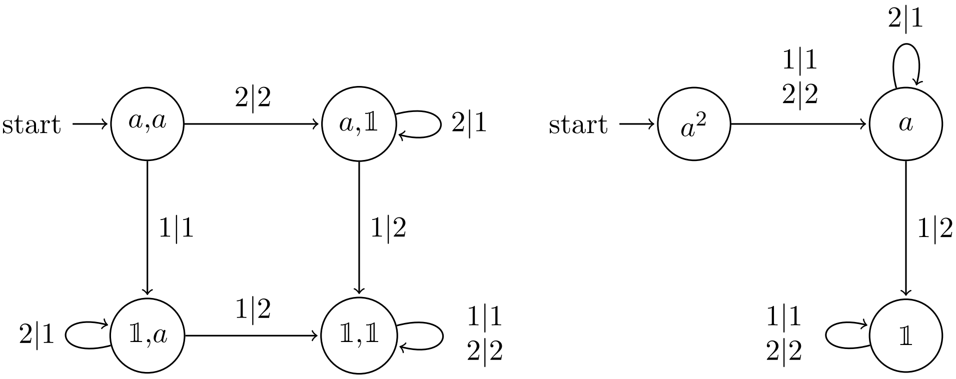
A classic result in automata theory is that there is a minimal form and there are a large variety of algorithms that minimize automata. For some examples see the standard book [**HU79**].

PROPOSITION 2.21. *For every finite state Mealy machine there is an equivalent minimal machine, which is unique up to state isomorphisms and we can construct the minimal machine by an algorithm.*

PROOF. Let $\mathfrak{M} = (\mathcal{A}, \mathcal{S}, t_A \times t_S, s_0)$ be a Mealy machine. The minimal machine can't have unreachable states. For this purpose we define the set of reachable states by following recursion.

$$\begin{aligned} &\mathcal{R}_0 = \emptyset, && \zeta_0 = \{s_0\} \\ &\mathcal{R}_i = \mathcal{R}_{i-1} \cup \zeta_{i-1}, && \zeta_i = \{t_S(x,a) \mid a \in \mathcal{A}, x \in \zeta_{i-1},\ t_S(x,a) \notin \mathcal{R}_i\}, \quad \text{for } i > 0\ . \end{aligned}$$

As there are only finitely many states and $\mathcal{R}_i$ grows as long as $\zeta_i \neq \emptyset$, there is an n such that $\zeta_n = \emptyset$. Then the set of *reachable states* is $\mathcal{R}_n$.
We can hence assume without loss of generality that $\mathfrak{M}$ has only reachable states by replacing the set of states accordingly and restricting the transition map.
The next step is to determine equivalent states and remove them. Two states $r, s \in \mathcal{S}$ are *equivalent* ($r \sim s$) if the machines $\mathfrak{M}_r = (\mathcal{A}, \mathcal{S}, t, r)$ and $\mathfrak{M}_s = (\mathcal{A}, \mathcal{S}, t, s)$ are equivalent. Hence $r \sim s$ if and only if $t_A(\ell, r) = t_A(\ell, s)$ and $t_S(\ell, r) \sim t_S(\ell, s)$ for



(A) Square of the adding machine (B) Minimal form of (A)

Figure 4. Equivalent machines

all $\ell \in \mathcal{A}$. This leads to a recursive problem:

$$\begin{aligned} &E_0 = \{(s,s) \mid s \in \mathcal{S}\}, \quad C_0 = \{(r,s)\},\\ &E_i = E_{i-1} \cup C_{i-1},\\ &C_i = \{(t_S(\ell,u), t_S(\ell,v)) \mid (u,v) \in C_{i-1},\ \ell \in \mathcal{A}\} \setminus E_i,\\ &e_i = |\{x \mid t_A(x,u) \neq t_A(x,v),\ (u,v) \in C_{i-1}\}|, \qquad \text{for } i > 0. \end{aligned}$$

As long as $e_i = 0$ for all $i \leq m$ and $(r,s) \in E_m$ the machines $\mathfrak{M}_r$ and $\mathfrak{M}_s$ have the same action on words of length at most m. Hence the states r and s are equivalent if and only if there is an $n \in \mathbb{N}$ such that $e_i = 0$ for all $i \leq n$ and $C_n = \emptyset$.
Since the sets E_i grow in every step as long as C_i is nonempty, this is a finite problem. For the minimal automaton $\mathfrak{M}'$ we choose as set of states $\mathcal{S}'$ the set of equivalence classes of $\mathcal{S}$. The start state is the equivalence class of s_0 and the transition-map is

$$\begin{aligned} t'\colon \mathcal{A} \times \mathcal{S}' &\to \mathcal{A} \times \mathcal{S}'\\ (\ell, [s]) &\mapsto ([t_S(\ell,s)], t_A(\ell,s))\ . \end{aligned}$$

This mapping is well-defined because $t_A(\ell,r) = t_A(\ell,s)$ and $t_S(\ell,r) \sim t_S(\ell,s)$ for $r \sim s$ and $\ell \in \mathcal{A}$. We have now defined an equivalent machine $\mathfrak{M}' = (\mathcal{A}, \mathcal{S}', t', s_0')$.
Claim: $\mathfrak{M}'$ is indeed minimal.
Assume there is an equivalent minimal machine $\mathfrak{M}'' = (\mathcal{A}, \mathcal{S}'', t'', s_0'')$. Since $\mathfrak{M}''$ is minimal it cannot have unreachable states. Hence for every $s'' \in S''$ there is a $w_{s''} \in \mathcal{A}^*$ such that $s'' = t''(w_{s''}, s_0'')$.
We define a map $\varphi\colon \mathcal{S}'' \to \mathcal{S}'$ by $s'' \mapsto t_S'(w_{s''}, s_0')$. This mapping is well-defined since for w, w' with $t_S''(w, s_0') = t_S''(w', s_0')$ we have

$$t_A(w,s_0)t_A(v,t_S(w,s_0)) = t_A(wv,s_0) = t_A''(wv,s_0'') = t_A''(w,s_0'')t_A''(v,t_S''(w,s_0''))$$

and hence $t_A''(u, t_S''(v,s_0'')) = t_A(u, t_S(v,s_0))$ for all $u, v \in \mathcal{A}^*$. Especially

$$t_A(v,t_S(w,s_0)) = t_A''(v,t_S''(w,s_0'')) = t_A''(v,t_S''(w',s_0'')) = t_A(v,t_S(w',s_0))$$

for all $v \in \mathcal{A}^*$. Hence $t_S(w,s_0) \sim t_S(w',s_0)$ and thus $t_S'(w,s_0') = t_S'(w',s_0')$.
This map φ is surjective because $\mathfrak{M}'$ contains only reachable states. Hence $\mathfrak{M}''$ has at least as many states as $\mathfrak{M}'$. If the number of states matches then the map φ is a state isomorphism and hence the minimal machine is unique. □

For an example of a minimal machine see Figure 4.

For two Mealy machines over the same alphabet $\mathfrak{M} = (\mathcal{A}, \mathcal{S}, t_A \times t_S, s_0)$ and $\mathfrak{M}' = (\mathcal{A}, \mathcal{S}', t'_A \times t'_S, s'_0)$ the *product* machine $\mathfrak{M}\mathfrak{M}'$ is defined as the machine with states $\mathcal{S} \times \mathcal{S}'$, start state (s_0, s'_0) and transition map

$$(\ell, (r, s)) \mapsto t'_A(t_A(\ell, r), s)\ ,\ (t_S(\ell, r), t'_S(t_A(\ell, r), s)).$$

This transition map is build in the way such that the action of $\mathfrak{M}\mathfrak{M}'$ on $v \in \mathcal{A}^*$ is the same as first act by $\mathfrak{M}$ and then by $\mathfrak{M}'$.

LEMMA 2.22. *The set of all machines with fixed alphabet $\mathcal{A}$ up to isomorphic machines forms a semigroup.*

PROOF. It is clear that the previous defined product is associative and that the the product of a machine and the trivial machine $\mathfrak{N} = (\mathcal{A}, \{\mathbb{1}\}, (\ell, \mathbb{1}) \mapsto (\ell, \mathbb{1}), \mathbb{1})$ is isomorphic to the original one. □

1.4. Corresponding tree automorphisms. Given a tree of degree d and an automorphism $a \in \mathrm{Aut}(T)$ we can define a corresponding Mealy machine by setting $\mathcal{A} = \{1, \ldots, d\}$, $S = \mathrm{Aut}(T)$, $s_0 = a$ and $t(\ell, b) = (\ell^b, b@\ell)$ for $\ell \in \mathcal{A}$ and $b \in S$. On the other hand: Not every Mealy machine defines a tree automorphism because the transition map doesn't need to be bijective. For example the machine

$$\mathfrak{M} = (\{1, 2\}, \{s_0\}, (1, s_0) \mapsto (1, s_0), (2, s_0) \mapsto (1, s_0), s_0)$$

maps every word to the word consisting only of zeros.

DEFINITION 2.23. Let $\mathfrak{M} = (\mathcal{A}, \mathcal{S}, t_A \times t_S, s_0)$ be an Mealy machine. If $t_A(\cdot, s)$ is a bijection on $\mathcal{A}$ for every $s \in \mathcal{S}$ we will call $\mathfrak{M}$ an *invertible* Mealy machine.

The justification for this naming is given by the following lemma:

LEMMA 2.24. *Let $\mathfrak{M} = (\mathcal{A}, \mathcal{S}, t_A \times t_S, s_0)$ be an invertible Mealy machine then $\mathfrak{M}$ defines a tree automorphism.*

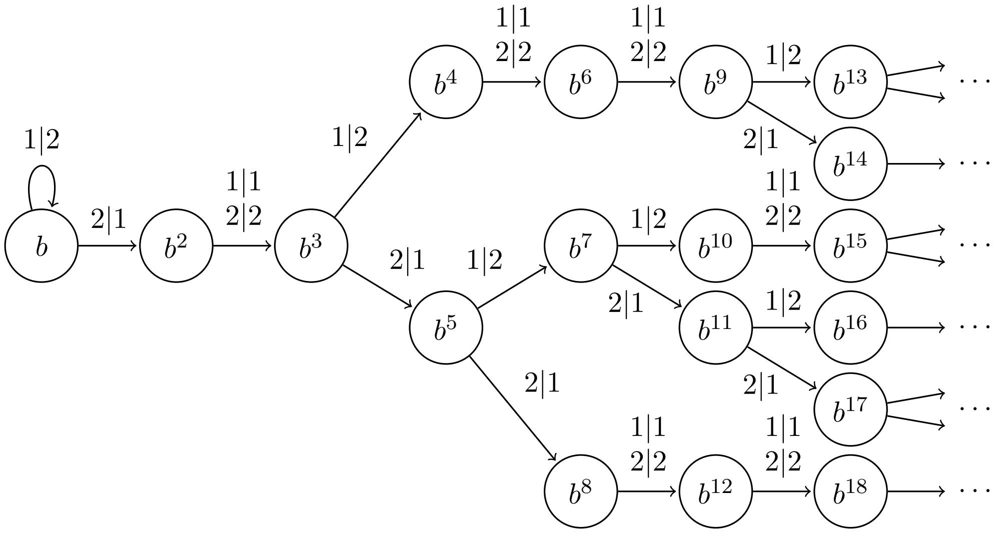
PROOF. Let $d = |\mathcal{A}|$, T be the tree of degree d and $\sigma_s \in \mathrm{S}_d$ be the permutation defined by the bijection $t_A(\cdot, s)$. We can define mappings $\varphi_n \colon S \to \mathrm{Perm}_n(\mathrm{Aut}(T))$ recursively by

$$\varphi_n \colon s \mapsto \begin{cases} \Phi^{-1}\left(\langle\!\langle \varphi_{n-1}(t_S(1, s)), \ldots, \varphi_{n-1}(t_S(d, s))\rangle\!\rangle \sigma_s\right) & \text{if } n > 0 \\ \mathbb{1}_{\mathrm{Aut}(T)} & \text{if n=0.} \end{cases}$$

The limit $\lim \varphi_n(s_0)$ then gives a tree automorphism with the same action on the tree as the Mealy machine. □

COROLLARY 2.25. *If $\mathfrak{M}$ is an invertible Mealy machine with alphabet $\mathcal{A}$, then there is a machine $\mathfrak{M}^{-1}$ such that $\mathfrak{M}\mathfrak{M}^{-1}$ is isomorphic to the trivial machine.*

For simplicity of the notation we will from now on identify tree automorphisms and their images under Φ. Furthermore we will allow us to write recursive definitions having in mind that $\mathrm{Aut}(T)$ is complete and thus to define an element uniquely by a Cauchy sequence. For example we will write $a = \langle\!\langle \mathbb{1}, a\rangle\!\rangle(1, 2)$ to describe the tree automorphism corresponding to the adding machine or even further: The automorphism $b = \langle\!\langle b, b^2\rangle\!\rangle(1, 2)$ will correspond to the following infinite automaton:



1.5. Formal self-similar groups. Since now self-similar groups where always subgroups of $\mathrm{Aut}(T)$. We can define formal self-similar groups without a tree. In Definition 2.15 we see that the defining relation of a self-similar subgroup of $\mathrm{Aut}(T)$ depends on the embedding Φ.

Definition 2.26. Let G be a group, P a finite group acting on a finite set $\mathcal{A}$, and $\Phi\colon G \hookrightarrow G\wr_{\mathcal{A}}P$ be an embedding. The pair (G, Φ) is called a self-similarity structure.

Given a self-similarity structure (G, Φ) we have an action of G on $\mathcal{A}$ via the action of P on $\mathcal{A}$. We can extend this action in the usual way recursively on $\mathcal{A}^*$ by $(wx)^g = w^g x^{g@w}$ for $w \in \mathcal{A}^*$, $x \in \mathcal{A}$ and $g \in G$ to obtain a not necessarily faithful action by tree automorphisms. Let us denote by φ this mapping $G \to \mathrm{Aut}(\mathcal{A}^*)$. We define the group $\mathcal{G}(G, \Phi)$ to be the quotient $G/\ker\varphi$.

Example 2.27. Let $F_3 = \langle f, g, h\rangle$ be the free group on three generators then we can define a self similarity structure by:

$$\Phi\colon F_3 \to F_3 \wr S_2, \quad f \mapsto \langle\!\langle g, g\rangle\!\rangle, \quad g \mapsto \langle\!\langle h, g\rangle\!\rangle(1,2), \quad h \mapsto \langle\!\langle h, h^2\rangle\!\rangle.$$

We see immediately that the generator h acts as the identity on the tree and a quick inspection shows that g acts like the adding machine and f like g^2 and hence $\mathcal{G}(F_3, \Phi) \simeq \mathbb{Z}$.

1.6. Functionally recursive groups. We note that the class of self-similar groups as defined in Definition 2.15 is not closed under subgroups. For example: Denote by $a \in \mathrm{Aut}(\{1,2\}^*)$ the adding machine automorphism (see Example 2.17). Then the group $G = \langle a^2\rangle$ is not self-similar because $a = a^2@1 \notin G$.

Definition 2.28. Let S be a *finite* subset of $\mathrm{Aut}(\mathcal{A}^*)$. The set S is called *functionally recursive* if for all $s \in S$ and $x \in \mathcal{A}$ the state $s@x$ is an element of the group generated by S.
An automorphism $a \in \mathrm{Aut}(T)$ is then called *functionally recursive* if there is a functionally recursive set S with $a \in S$.

Lemma 2.29. *The set of all functionally recursive automorphisms forms a group.*

Proof. Let s, t be functionally recursive automorphisms with the corresponding functionally recursive sets S, T. Then the states of $s^{-1}t$ are members of the group generated by the finite set $S \cup T$. □

Notation 2.30. The group $\mathrm{RAut}(T)$ is the group of all functionally recursive automorphisms of the tree T.

1.7. Finite State Automorphisms. A stronger condition on automorphisms than being functionally recursive is the notion of *finite state* automorphisms.

Definition 2.31. A finite set $F \subset \mathrm{Aut}(\mathcal{A}^*)$ is called *finite state* if for all $f \in F$ and $x \in \mathcal{A}$ holds that $f@x \in F$.
An automorphism f is then called finite state if there is a finite state set F with $f \in F$. A minimal finite state set for a finite state automorphism is then called the *state set* of f:

$$\mathrm{States}(f) = \{f@w \mid w \in \mathcal{A}^*\} \ .$$

A group is called finite state if all its elements are finite state.

Lemma 2.32. *The set of all finite state automorphisms in* $\mathrm{Aut}(T)$ *forms a group.*

Notation 2.33. The group $\mathrm{FAut}(T)$ is the group of all finite state automorphisms of the tree T.

Proof. With the computation rules for states (Corollary 2.14) it's clear that for $f, g \in \mathrm{FAut}(\mathcal{A}^*)$ we have $\mathrm{States}(f^{-1}g) \subseteq \{s^{-1}t \mid s \in \mathrm{States}(f),\ t \in \mathrm{States}(g)\}$. □

Lemma 2.34. *The group of all minimal Mealy machines on an alphabet $\mathcal{A}$ is isomorphic to* $\mathrm{FAut}(\mathcal{A}^*)$*.*

Corollary 2.35. *The word problem in* $\mathrm{FAut}(\mathcal{A}^*)$ *is solvable. That is: Given an element a in* $\mathrm{FAut}(\mathcal{A}^*)$ *by its wreath recursive definition there is an algorithm to determine in finite time if a is the trivial automorphism.*

2. Bounded groups

Definition 2.36 ([**Sid00**]). A tree automorphism a is called *bounded* if there is a constant C such that for all n holds:

$$|\{w \in \mathcal{A}^n \mid a@w \neq \mathbb{1}\}| \leq C \ .$$

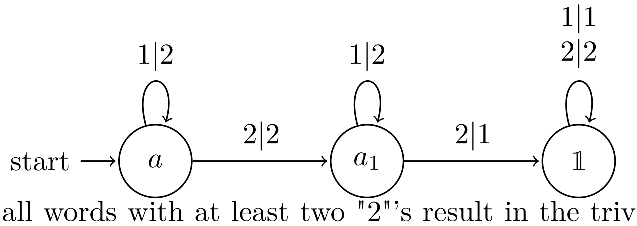
For example the binary adding machine automorphism a is bounded by $C = 1$ and a^n is bounded by $C = \lceil n/2 \rceil$ for every $n \in \mathbb{N}$.
A more general approach are polynomial automorphisms.

Definition 2.37. A tree automorphism a is polynomial bounded of degree d if there is a polynomial p of degree d such that for all n

$$|\{w \in \mathcal{A}^n \mid a@w \neq \mathbb{1}\}| \leq p(n) \ .$$

Example 2.38.

- A polynomial finite state automorphism which isn't bounded:



$(\mathcal{A}_1)$

Exactly all words with at least two "2"'s result in the trivial state. Hence the word consisting of "1"'s and the n words in $\mathcal{A}^n$ with exactly one "2" result in a nontrivial state. Therefore a is a polynomial tree automorphism with polynomial $p(n) = n + 1$. The state a_1 is bounded with bounding constant 1.

- A bounded automorphism which is not finite state:

$$c := \langle\!\langle c_1, \mathbb{1}\rangle\!\rangle,\ c_n := \langle\!\langle c_{n+1}, \mathbb{1}\rangle\!\rangle\sigma_n \text{ with } \sigma_n = \begin{cases}(0,1) & \text{if } n = 2^m \text{ for some } m\\ \mathbb{1} & \text{else}\end{cases} .$$

For every length n the only word which doesn't result in the trivial state is the word consisting only of zeros. Hence c is bounded by 1. All states are different because for every $n < m \in \mathbb{N}$ we have

$$(\underbrace{0\dots0}_{2^n+1})^{c_{2^n}} = 1\underbrace{0\dots0}_{2^n-2}1 \qquad \text{and} \qquad (\underbrace{0\dots0}_{2^n+1})^{c_{2^m}} = 1\underbrace{0\dots0}_{2^n-2}0.$$

Therefore all the states c_i are different and c can't be finite state.

Lemma 2.39. *The finite state polynomial automorphisms of degree d form a group.*

Proof. Define for tree automorphisms $a \in \mathrm{Aut}(\mathcal{A}^*)$ the set of words in each level $n \in \mathbb{N}$ which doesn't result in the trivial state:

$$W(a,n) = \{w \in \mathcal{A}^n \mid a@w \neq \mathbb{1}\}.$$

Let a, b be degree d polynomial bounded automorphisms hence there are degree d polynomials p, q with $|W(a,n)| \leq p(n)$ and $|W(b,n)| \leq q(n)$ for all $n \in \mathbb{N}$.
Define $V(n) = \{w^a \mid w \in W(a,n) \cup W(b,n)\}$. For $v \in \mathcal{A}^n \setminus V(n)$ we have $v^{a^{-1}} \notin W(a,n) \cup W(b,n)$ and therefore

$$(a^{-1}b)@v = a^{-1}@v \cdot b@v^{a^{-1}} = \left(a@v^{a^{-1}}\right)^{-1} \cdot b@v^{a^{-1}} = \mathbb{1}.$$

Consequently $|W(a^{-1}b, n)| \leq p(n) + q(n)$. □

Notation 2.40. The group of *finite state* degree d polynomial bounded automorphisms of the n-ary tree is denoted by $\mathrm{Poly}_n(d)$.

Said Sidki developed an easy to apply rule to determine if a finite state automorphism is bounded by the analysis of the cyclic structure of the graph of the corresponding automaton (see [**Sid00**]).
In short words: A finite state automorphism is bounded if and only if a corresponding Mealy machine has on each directed path from the start state to an identity state at most one circle.

Definition 2.41.

(1) A non-trivial tree automorphism a is called *circular* if there is $w \in \mathcal{A}^n$ such that $a@v = a$. A word w with this property such that no prefix of w has the same property is called a *circular word* for this circle.
(2) The *number of circles* for an automorphism a is defined by the number of distinct circular words.

$$\#_C(a) := |\{w \in \mathcal{A}^* \mid a@w = a \nexists \epsilon < v < w : a@v = a\}| \in \mathbb{N}_0 \cup \infty\ .$$

(3) We call two circular automorphisms a, b equivalent if there is a circular word w for a that has a prefix $v \leq w$ such that $a@v = b$. The equivalence class will be called a *circle* and denote the class by $[a]$.
(4) The multiplicity μ of a circle $[a]$ is 1 if for all $b \in [a]$ we have $\#_C([a]) = 1$. Otherwise it will be ∞.
(5) The number of circles of an automorphism a on a word w is the number of all distinct non-trivial circles along this path with multiplicity:

$$\#_C^w(a) = \sum_{[b]\in\{[a@v]:\ v\leq w\}\setminus[\mathbb{1}]} \mu([b])\ .$$

(6) The number of circles for a finite state automorphism a is then defined as the maximal path-circle-number:

$$\#\,\mathrm{Circles}(a) := \max\{\#^{w}_{C}(a) \mid w \in \mathcal{A}^*\} \ .$$

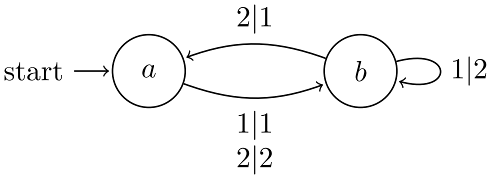
(7) A tree automorphism a is called *pre-circular* if $\#\,\mathrm{Circles}(a) > 0$ and finitary if $\#\,\mathrm{Circles}(a) = 0$.

Notation 2.42. The group of finitary automorphisms on an n-ary tree will be denoted by $\mathrm{Poly}_n(-1)$.

Example 2.43.

(1) The automaton $\mathcal{A}_1$ has two disjoint circles with multiplicity 1 which are connected with a directed path. Therefore $\#\,\mathrm{Circles}(a) = 2$.

(2) The automorphism a as given by the following automaton



and the state $b = a@2$ are both circular and belong to the same circle. The number of circles for b is three (circular words are $1, 22, 21$) and a has an infinite number of circles, because $22, 212, 2112, 21\dots 12$ are all circular words.

Lemma 2.44. *If a is a polynomial bounded automorphism then every circle in a has multiplicity* 1.

Proof. Let b be a circular automorphism with at least two distinct circular words v, w. Then we have 2^n distinct words of length at most $n \cdot \max\{|v|, |w|\}$ that result in a nontrivial state. Hence the set $W(b, n) = \{w \in \mathcal{A}^n \mid a@w \neq \mathbb{1}\}$ grows exponentially and any automorphism a that has b as state can't be polynomial bounded by any degree. □

Proposition 2.45. [**Sid00**] *For a finite state tree automorphism $b \in \mathrm{Aut}(T)$ of an n-ary tree T we have $b \in \mathrm{Poly}_n(d)$ if and only if $\#\,\mathrm{Circles}(b) \leq d + 1$.*

Proof. If a is a finitary automorphism the claim is obvious. Let $d > -1$ and a be a degree d polynomial bounded automorphism. Define the sets of first and second reachable circles:

$$C_0(n) = \{[a@w] \mid w \in \mathcal{A}^n, a@w \text{ is circular}, \nexists v < w \text{ such that } a@v \text{ is circular}\}$$
$$C_1(n) = \{[a@w] \mid w \in \mathcal{A}^n, \exists v < w\colon a@v \text{ is circular} \Rightarrow [a@v] \in C_0(n)\} \setminus C_0(n).$$

If $C_1(n)$ is empty then $\#\,\mathrm{Circles}(a) = 1$ and we have nothing to show. We note that the size of the set $W_0(n) = \{w \in \mathcal{A}^n \mid [a@w] \in C_0\}$ is bounded by a constant and the size of $W_1(n) = \{w \in \mathcal{A}^n \mid [a@w] \in C_1(n)\}$ is bounded by a linear polynomial in n. Because the size of the set $W(a, n) = \{w \in \mathcal{A}^n \mid a@w \neq \mathbb{1}\}$ is bounded by a degree d polynomial, the size of the sets $W(c, n)$ for $c \in [b] \in C_1(n)$ is bounded by a degree $d - 1$ polynomial and hence by induction $\#\,\mathrm{Circles}(b) \leq d$. By the definition of $C_1(n)$ therefore $\#\,\mathrm{Circles}(a) \leq d + 1$. □

Corollary/Definition 2.46. A tree automorphism a is *finitary* if there is an $n \in \mathbb{N}$ such that for all $w \in \mathcal{A}^m$, $m \geq n$ holds that $a@w = \mathbb{1}$. The minimal n with this property is called the *depth* of the automorphism.

3. Contracting groups

That some self-similar groups enjoy the property to be contracting is already used in the papers by R. Grigorchuk in the proof of the properties of the Grigorchuk group. The terminology we present here however follows the wording of V. Nekrashevych's book [**Nek05**].

DEFINITION 2.47. A self-similar group G is *contracting* if there is a finite self-similar set $N \subset G$ such that for all $g \in G$ there is a bound $n_g \in \mathbb{N}$ such that for all longer words $w \in \mathcal{A}^m$, $m \geq n_g$ the states $g@w$ belong to N.
The minimal such set N is called the *nucleus* of G. A finite state automorphism a is called *contracting* if the group generated by the states of a is contracting.

LEMMA 2.48. *If G is a contracting group, then every element in G is finite state.*

PROOF. Let N be the nucleus of G and $g \in G$, then $g@w \in N$ for all $w \in \mathcal{A}^{\geq n_g}$. Thus $\mathcal{S}(g)$ is contained in $\{g@w \mid w \in \mathcal{A}^{\leq n_g}\} \cup N$ which is finite. □

If one considers the word metric on a finitely generated group G, there is a different description of being contracting that explains the name in a more obvious manner.

LEMMA 2.49. *If G is finitely generated and contracting, then there is $\lambda < 1$ and $C, n \in \mathbb{N}$ such that for all $g \in G$ and $w \in \mathcal{A}^{\geq n}$ it holds:*

$$|g@w| < \lambda|g| + C \ .$$

PROOF. Let N be the nucleus of G and C be larger then the largest length of elements in N and fix an arbitrary integer $\ell > 1$. Since G is contracting, we can find for every element $h \in G$ an integer n_h such that $h@v \in N$ for every word v of length at least n_h. We take

$$n = \max\{n_h \mid |h| < \ell C\}.$$

Let us fix an element $g \in G$ with length $|g| = k\ell C + C_1$ for an integer k and some $C_1 < \ell C$. Hence we can find elements $h_0, \ldots, h_k \in G$ with $|h_i| \leq \ell C$ and $g = \prod_{i=0}^{k} h_i$. For $v \in \mathcal{A}^m$ with $m \geq n$ we thus have $g@v = \prod_{i=0}^{k} h_i@v^{t_i}$ with $t_i = \prod_{j=0}^{i} h_j$. Because for every word w of length at least n we have $h_i@w \in N$ for every i and hence $|h_i@w| \leq C - 1$. Thus

$$|g@v| \leq \sum_{i=0}^{k} |h@v^{t_i}| \leq (k+1)(C-1) \leq \frac{1}{\ell}|g| + C - 1 < \frac{1}{\ell}|g| + C.$$

□

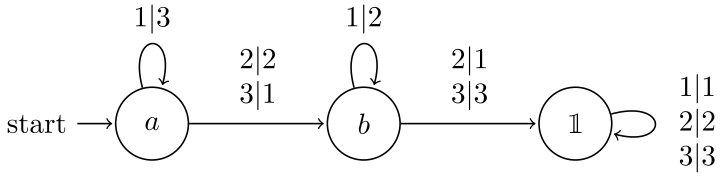
COROLLARY 2.50. *We can choose λ in Lemma 2.49 arbitrarily small with the cost of increasing the constant n.*

EXAMPLE 2.51.

(i) It is easy to see that the group generated by the adding machine a is contracting with nucleus $N = \{a, a^{-1}, \mathbb{1}\}$ and contracting constants $n_{a^m} = \lceil \log_2(m) \rceil$. More generally: All groups generated by the states of a finite state bounded automorphism are contracting (see [**BN03**, Theorem 5.3.]).

(ii) The group generated by the automaton $\mathcal{A}_1$ is not contracting because $a^n@1^m = a^n$ for every m and hence a^n would be an element of the nucleus for every n. All those elements are distinct because $a^n@2 = a_1^n$ and a_1 is the adding machine and thus of infinite order.

(iii) Finitely generated contracting groups need not to be generated by bounded elements: The group generated by the following not bounded Mealy machine is contracting with nucleus $N = \{\mathbb{1}, a, a^{-1}, b, b^{-1}, a^{-1}b, b^{-1}a\}$.



(iv) The infinitely generated group $\mathrm{Poly}(-1)$ is contracting with trivial nucleus.

Lemma 2.52. *If G is a contracting group with nucleus N such that all elements of the nucleus are in Poly_d then $G \leq \mathrm{Poly}_d$.*

Proof. For $g \in G$ denote by n the number such that $g@w \in N$ for all $w \in \mathcal{A}^{\geq n}$ and by p a degree d polynomial in m that bounds the number of words of length m that result in a nontrivial state of a nucleus element.
Then the set $W(m) = \{w \in \mathcal{A}^m \mid g@w \neq 1\}$ is bounded in size by $|\mathcal{A}|^n + p(m)$ and hence $g \in \mathrm{Poly}_d$. □

Lemma 2.53. *The nucleus of a finitely generated contracting group given by the wreath recursive definitions of its generators is computable.*

Proof. Let $S = S^{-1}$ be a finite set of generators of a contracting group G with $\mathbb{1} \in S$. We define the following ascending family of finite sets:

$$N_0 = \{s@v \mid s \in S, v \in \mathcal{A}^{\geq|\mathrm{States}\,(s)|}\},$$
$$N_m = \{(ts)@v \mid s \in S, t \in N_{m-1}, v \in \mathcal{A}^{\geq|\mathrm{States}\,(ts)|}\}.$$

Consider $x = (ts)@v \in N_m \setminus N_{m-1}$. Then the element ts is not finitary and hence there are words v_1, v_2, v_3 such that $v = v_1v_2v_3$ and $(ts)@v_1$ is circular with circular word v_2. Hence $x = (ts)@v_1v_2^\ell v_3$ for every integer ℓ and thus x is in the nucleus of G. Because the nucleus is finite the sequence of sets N_k becomes constant and the nucleus is contained in this limit. □

Definition 2.54. A self-similar group $G \leq \mathrm{Aut}(\mathcal{A}^*)$ is *recurrent* if G acts transitively on the first level of the tree $\mathcal{A}$ and $\{g@x \mid g \in \mathrm{Stab}_G(x)\} = G$ for all $x \in \mathcal{A}$.

Definition 2.55. A self-similar group $G \leq \mathrm{Aut}(\mathcal{A}^*)$ is *weakly-recurrent* if G acts transitively on the first level of the tree $\mathcal{A}$ and $\{g@x \mid g \in G,\ x \in \mathcal{A}\} = G$.

Proposition 2.56. *The isomorphism problem is solvable among finitely generated weakly-recurrent contracting groups over a fixed alphabet. That is: There exists an algorithm that, given two sets of generators with its wreath recursive definition, decides whether the groups generated by the corresponding sets are equal.*

This is already well known for some time, we follow the proof of [**Nek05**].

Proof. We prove that a finitely generated recurrent contracting self-similar group is generated by its nucleus. Then to decide whether two groups are equal we only need to compute their nuclei and compare if they are equal.
Let G be a contracting group with nucleus N and generating set S. Let further n be the minimal number such that for all $s \in S$ and words w of length at least n we have $s@w \in N$. Thus for every $g \in G$ that is a product of m generators the state $g@w$ is a product of at most m elements of N. Because G is weakly-recurrent we

have $G = \{g@w \mid w \in \mathcal{A}^n, g \in G\}$ and hence every element of g is generated by elements of the nucleus. □

It is still an open problem if there exists an algorithm that decides if a given set of finite state generators generate a contracting group.

4. Branch groups

Definition 2.57. A self-similar group $G \leq \mathrm{Aut}(\mathcal{A}^*)$ is *layered* if the embedding $\Phi\colon G \hookrightarrow G \wr_{\mathcal{A}} \mathrm{Perm}_1(G)$ is an isomorphism.

Example 2.58.

(1) The groups $\mathrm{Aut}(\mathcal{A}^*)$, $\mathrm{RAut}(\mathcal{A}^*)$, $\mathrm{FAut}(\mathcal{A}^*)$, $\mathrm{Poly}_n(\mathcal{A}^*)$ for $n \geq -1$ are all layered.
(2) The group generated by the adding machine $G = \langle a \rangle$ is not layered since $\langle\!\langle a, a^2 \rangle\!\rangle \notin G$.

We will see a finitely generated example of a layered group in Section 5.3.

Definition 2.59. A self-similar group $G \leq \mathrm{Aut}(\mathcal{A}^*)$ is *regular branched* if it has a normal subgroup K of finite index such that $K^{\mathcal{A}} \leq \Phi(K)$.
A normal subgroup K with this property is called a *branching subgroup*.

Note that to require G to have a layered normal subgroup of finite index is a stronger condition. We do not assume that the branching subgroup is self-similar. But it is functionally recursive by its definition.

Definition 2.60 ([**Bar13**]). A *branch structure* of a group $G \hookrightarrow G \wr \mathrm{S}_n$ consists of

(1) a branching subgroup $K \trianglelefteq G$;
(2) the quotient $Q = G/K$ and the factor homomorphism $\pi\colon G \to Q$;
(3) a group $Q_1 \subset Q \wr \mathrm{S}_n$ such that $\langle\!\langle q_1, \dots, q_n \rangle\!\rangle \sigma \in Q_1$ for $q_i \in Q$ and $\sigma \in \mathrm{S}_n$ if and only if $\langle\!\langle g_1, \dots, g_n \rangle\!\rangle \sigma \in G$ for one and hence all $g_i \in \pi^{-1}(q_i)$;
(4) a map $\omega\colon Q_1 \to Q$ with the following property: if $g = \langle\!\langle g_1, \dots, g_n \rangle\!\rangle \sigma \in G$ then $\omega(\langle\!\langle \pi(g_1), \dots, \pi(g_n) \rangle\!\rangle \sigma) = \pi(g)$.

Lemma 2.61. *All regular branched groups have a branch structure.*

Proof. Let G be a regular branched group with branching subgroup K and canonical factor homomorphism $\pi\colon G \to G/K$. If $g = \langle\!\langle g_1, \dots, g_n \rangle\!\rangle \sigma$ is a group element then also $\langle\!\langle k_1, \dots, k_n \rangle\!\rangle \cdot g$ is a member of G for every choice of $k_i \in K$. Hence the map $\omega\colon \langle\!\langle g_1^\pi, \dots, g_n^\pi \rangle\!\rangle \sigma \to (\langle\!\langle g_1, \dots, g_n \rangle\!\rangle \sigma)^\pi$ is well defined.

□

5. Examples

5.1. Grigorchuk group. The first Grigorchuk group defined for the first time by R. Grigorchuk in [**Gri80**] is a finitely generated self-similar group acting faithfully on the binary rooted tree with generators:

$$a = \langle\!\langle \mathbb{1}, \mathbb{1} \rangle\!\rangle (1,2), \qquad b = \langle\!\langle a, c \rangle\!\rangle, \qquad c = \langle\!\langle a, d \rangle\!\rangle, \qquad d = \langle\!\langle \mathbb{1}, b \rangle\!\rangle.$$

This group is famous to be the first example of a group of intermediate word growth and thus answers a question of J. Milnor (see [**CWM$^+$68**] and [**Gri83**]). An overview over some of its other remarkable properties can be found in the overview article [**Gri05**].

Notation 2.62. We will denote the Grigorchuk group by $\mathfrak{G}$.

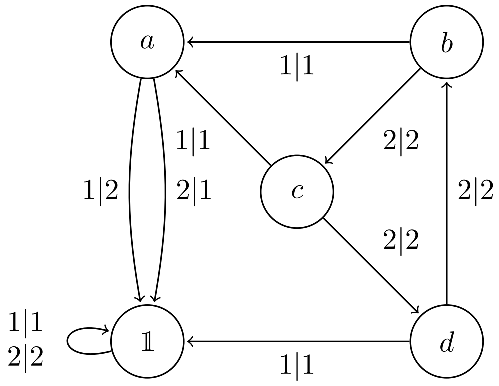


Figure 5. Generating automaton for the Grigorchuk group

Note that $\mathfrak{G}$ is already generated by a and any two of $\{b, c, d\}$ because $bc = d$. Some other useful short identities are

$$a^2 = b^2 = c^2 = d^2 = bcd = \mathbb{1},$$
$$b^a = \langle\!\langle c, a\rangle\!\rangle, c^a = \langle\!\langle d, a\rangle\!\rangle, d^a = \langle\!\langle b, \mathbb{1}\rangle\!\rangle,$$
$$(ad)^4 = (ac)^8 = (ab)^{16} = \mathbb{1}.$$

Lemma 2.63. *The stabilizer of the first level is given by* $\mathrm{Stab}_1(G) = \langle b, c, d, b^a, c^a, d^a\rangle$.

Proof. It is obvious that $\mathrm{Stab}_1(\mathfrak{G}) \supseteq \langle b, c, d, b^a, c^a, d^a\rangle$. If $g \in \mathrm{Stab}_1(\mathfrak{G})$ is written as reduced word $w(a, b, c, d)$ over the generators of G then the number of occurrences of a has to be even and two letters in b, c, d have to be separated by a because otherwise the word is not reduced. □

Lemma 2.64. *The Grigorchuk group is bounded.*

Proof. This is a direct consequence of Proposition 2.45 and the graph representation of the generating automaton (Figure 5). □

Corollary 2.65. *The group* $\mathfrak{G}$ *is contracting with nucleus* $\{b, c, d, a, \mathbb{1}\}$. □

Lemma 2.66. *The group* $\mathfrak{G}$ *is recurrent.*

Proof. From the above identities we have especially: $a = b@1$, $b = d^a@1$, $c = b^a@1$ and hence $G = \{x@1 \mid x \in \mathrm{Stab}_1(\mathfrak{G})\} = \{x^a@2 \mid x \in \mathrm{Stab}_1(\mathfrak{G})\}$. □

Lemma 2.67 ([**Roz93**]). *The Grigorchuk group is regular branched with branching subgroup*

$$K := \left\langle (ab)^2 \right\rangle^{\mathfrak{G}} = \left\langle (ab)^2, (bada)^2, (abad)^2 \right\rangle.$$

The quotient $Q := \mathfrak{G}/K$ *has order* 16. □

Lemma 2.68. *The group* $\mathfrak{G}$ *is not complete.*

Proof. We define a Cauchy sequence $(x_m)_{m\in\mathbb{N}}$ in $\mathfrak{G}$ by:

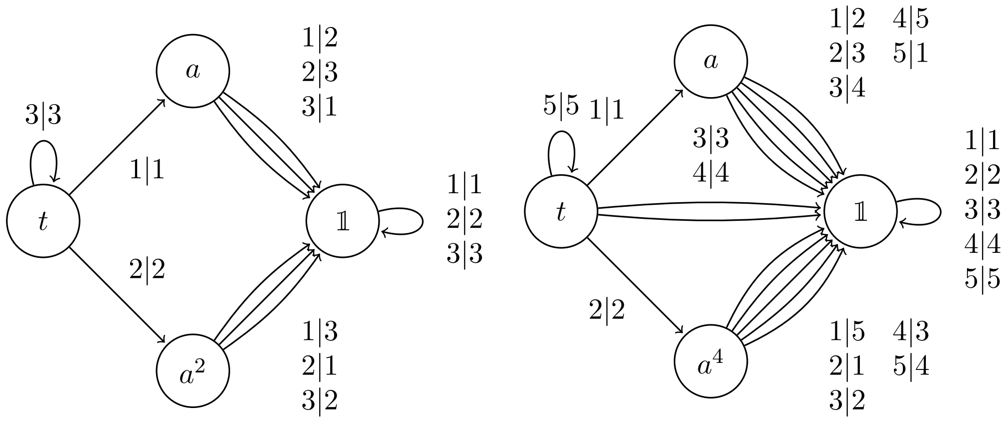
$$x_m^{(n)} = \begin{cases} \langle\!\langle x_m^{(n+1),(ab)^2,} \rangle\!\rangle & \text{if } n < m \\ \mathbb{1} & \text{otherwise} \end{cases}, \qquad x_m = x_m^{(1)}.$$

For every $m \in \mathbb{N}$ we have that $x_m \in K$ because both states are members of K. In $\mathrm{Aut}(T)$ the Cauchy sequence converges towards $x = \langle\!\langle x, (ab)^2\rangle\!\rangle$. If we assume that $x \in \mathfrak{G}$ then x has to be in the nucleus because $x@1\dots 1 = x$ for arbitrary long words. But it is immediate to check that $x \neq \mathbb{1}, a, b, c, d$.

□

5.2. Gupta-Sidki group. The Gupta-Sidki p-groups defined in [**GS83**] are for any prime p a two generated self-similar group acting faithfully on the p-nary rooted tree with generators:

$$a = \langle\!\langle \mathbb{1}, \dots, \mathbb{1} \rangle\!\rangle (1, 2, \dots, p), \qquad\qquad t = \langle\!\langle a, a^{-1}, \mathbb{1}, \dots, \mathbb{1}, t \rangle\!\rangle.$$



(A) Gupta-Sidki 3-group (B) Gupta-Sidki 5-group

FIGURE 6. Generating automata for some Gupta-Sidki p-groups

NOTATION 2.69. We will denote the Gupta-Sidki p-group by $\mathfrak{G}_p$.

LEMMA 2.70 ([**GS83**]). *The group $\mathfrak{G}_p$ is indeed a p-group.* □

LEMMA 2.71. *The Gupta-Sidki p-group is bounded for every prime p.*

PROOF. This is a direct consequence of Proposition 2.45. □

5.3. Neumann-Segal groups. We define for $n \in \mathbb{N}$ groups $\mathfrak{N}_n$ acting on the n-ary tree by the following generators.

$$\begin{aligned}
a_\pi &= \langle\!\langle 1, \dots, 1 \rangle\!\rangle \pi, \\
\alpha_\pi &= \langle\!\langle a_\pi, \alpha_\pi, 1, \dots, 1 \rangle\!\rangle \text{ for } \pi \in \mathrm{A}_n, \\
N_n &= \langle a_\pi \mid \pi \in \mathrm{A}_n \rangle, \\
M_n &= \langle \alpha_\pi \mid \pi \in \mathrm{A}_n \rangle, \\
\mathfrak{N}_n &= \langle N_n, M_n \rangle .
\end{aligned}$$

Note that $N_n \cong M_n \cong \mathrm{A}_n$. This construction was first done by D. Segal in [**Seg00**] similar to a construction of P. Neumann in [**Neu86**]. In [**BdlH10**] L. Bartholdi and P. de la Harpe proved some properties about this group.

LEMMA 2.72 ([**Bar03**]). *The groups $\mathfrak{N}_n$ are layered for $n \geq 5$.*

PROOF. It's well known that the alternating group A_n is perfect for $n \geq 5$ and that furthermore every element is a commutator (see [**Mil99**]). For fixed $\pi \in \mathrm{A}_n$ we choose $\rho, \sigma \in \mathrm{A}_n$ such that $\pi = [\rho, \sigma]$. Then $a_\pi = [a_\rho, a_\sigma]$ and $\alpha_\pi = [\alpha_\rho, \alpha_\sigma]$. With $\tau := (2,3)(4,5)$ we then have

$$\begin{aligned}
b_\pi &:= [\alpha_\rho, \alpha_\sigma^{a_\tau}] = \langle\!\langle a_\pi, 1, \dots, 1 \rangle\!\rangle \\
\beta_\pi &:= b_\pi^{-1} \cdot \alpha_\pi = \langle\!\langle 1, \alpha_\pi, 1, 1, 1 \rangle\!\rangle.
\end{aligned}$$

Now it is obvious that the set $\{b_\pi, \beta_\pi, a_\pi \mid \pi \in A_n\}$ generates $\mathfrak{N}_n \wr \mathrm{A}_n$. □

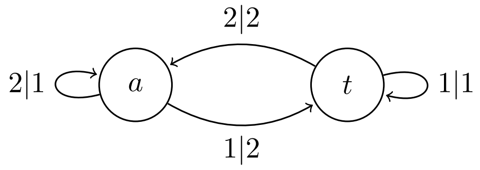


FIGURE 7. Mealy machine generating the lamplighter group.

LEMMA 2.73. *The groups $\mathfrak{N}_n$ are contracting with nucleus $\mathcal{N}(\mathfrak{N}_n) = N_n \cup M_n$.*

PROOF. It is immediate to see that $N_n \cup M_n$ is contained in the nucleus because the elements in M_n are circular. In the same manner as in the proof of Lemma 2.53 we see that products of those generators have no additional circular elements. □

5.4. Menagerie of self-similar groups.

LEMMA 2.74. *If G is a self-similar group acting on a d-ary tree and H a self-similar group acting on a e-ary tree. Then the product $G \times H$ acts self similarly on a $(d \cdot e)$-ary tree. Furthermore if G and H are both automaton groups then so is the product.*

PROOF. Let $G \leq \mathrm{Aut}(\mathcal{A}_1^*)$ and $H \leq \mathrm{Aut}(\mathcal{A}_2^*)$ be two self-similar groups and $g \in G$, $h \in H$ two elements. We define the action on and by $(x, y) \in \mathcal{A}_1 \times \mathcal{A}_2$ as following:

$$(x, y)^{(g,h)} = (x^g, y^h), \qquad (g, h)@(x, y) = (g@x, h@y)$$

for $w \in (\mathcal{A}_1 \times \mathcal{A}_2)^*$ and $x \in \mathcal{A}_1 \times \mathcal{A}_2$ we extend this action in the usual sense recursively by $(wx)^{(g,h)} = w^{(g,h)} x^{(g,h)@w}$. A short calculation shows that this action fulfills the self-similarity axiom:

$$\begin{aligned}(g_1 g_2, h_1 h_2)@(x, y) &= (g_1@x \cdot g_2@x^{g_1}, h_1@y \cdot h_2@y^{h_1})\\ &= (g_1@x, h_1@y)(g_2@x^{g_1}, h_2@y^{h_1})\\ &= (g_1, h_1)@(x, y) \cdot (g_2, h_2)@(x, y)^{(g_1,h_1)}.\end{aligned}$$

The given definition ensures directly that if g and h are finite state homomorphisms on their corresponding trees then so is (g, h). □

LEMMA 2.75. *Every finite group can be realized as a self-similar automaton group.*

PROOF. By Cayley's theorem every finite group embeds into a symmetric group S_n for some n and we can hence realize it as a automaton group generated by finitary automata of depth one. □

COROLLARY 2.76. *Every finitely generated abelian group is isomorphic to a self similar automaton group.*

PROOF. We did already see that the adding machine generates the free group of rank 1 and because every finitely generated abelian group is isomorphic to $\mathbb{Z}^n \times C_{k_1} \times \ldots \times C_{k_m}$ for some $n \in \mathbb{N}$ and cyclic groups $C_{k_1}, \ldots, C_{k_m}$ the result follows from Lemma 2.74. □

The free abelian group $\mathbb{Z}$ acts on itself by addition. The group $\mathbb{Z}/2\mathbb{Z} \wr_{\mathbb{Z}} \mathbb{Z}$ is known as the lamplighter group. It was proved by R. Grigorchuk and A. Żuk in [**GZ01**] that it is generated by the two state automaton displayed in Figure 7. For more general lamplighter groups P. Silva and B. Steinberg proved the following:

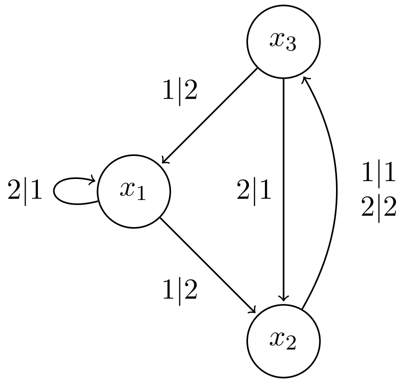


FIGURE 8. Mealy machine generating the free group $\langle x_1, x_2, x_3\rangle$.

PROPOSITION 2.77 ([**SS05**]). *Let H be a finite abelian group. Then the groups $H \wr_{\mathbb{Z}} \mathbb{Z}$ are isomorphic to some self similar automaton groups.*

LEMMA 2.78 ([**BS98**]). *The groups $\mathrm{GL}_n(\mathbb{Z})$ are isomorphic to automaton groups on an alphabet of size 2^n.*

COROLLARY 2.79 ([**BS98**]). *The free groups on arbitrary rank are isomorphic to automaton groups.*

The automaton groups given by A. Brunner and S. Sidki that generate $\mathrm{GL}_n(\mathbb{Z})$ are not generated by all states of a single automaton and hence the given examples are not self-similar. Y. Glasner and S. Mozes used in [**GM05**] the idea of bi-reversible automata, i.e. an invertible Mealy machine whose dual is also an invertible Mealy machine, to construct self-similar automaton groups that are isomorphic to free groups. An example for a Mealy machine that generates a free group of rank three was already given by S. Aleshin in [**Ale83**] depicted in Figure 8 although it was only proved two decades later by M. and V. Vorobets in [**VV07**] that this machine indeed generates a free group.

CHAPTER 3

Equations

1. Basic Definitions

We fix a set $\mathcal{X}$ and call its elements *variables*. We assume that $\mathcal{X}$ is infinite countable, is well ordered, and that its family of finite subsets is also well ordered, by size and then lexicographic order. We denote by $F_{\mathcal{X}}$ the free group on the generating set $\mathcal{X}$.

Definition 3.1. Let G be a group. A *G-group* is a group with a distinguished copy of G inside it; a typical example is $G * H$ for some group H. A *G-homomorphism* between G-groups is a homomorphism that is the identity between the marked copies of G.
A *G-equation* is an element $\mathcal{E}$ of the G-group $F_{\mathcal{X}} * G$, regarded as a reduced word in $\mathcal{X} \cup \mathcal{X}^{-1} \cup G$. For $\mathcal{E}$ a G-equation, its set of *variables* $\operatorname{Var}(\mathcal{E}) \subset \mathcal{X}$ is the set of symbols in $\mathcal{X}$ that occur in it; namely $\operatorname{Var}(\mathcal{E})$ is the minimal subset of $\mathcal{X}$ such that $\mathcal{E}$ belongs to $F_{\operatorname{Var}(\mathcal{E})} * G$.
An *evaluation* is a G-homomorphism $e\colon F_{\mathcal{X}} * G \to G$. A *solution* of an equation $\mathcal{E}$ is an evaluation s satisfying $s(\mathcal{E}) = \mathbb{1}$. If a solution exists for $\mathcal{E}$ then the equation $\mathcal{E}$ is called *solvable*. The set of elements $X \in \mathcal{X}$ with $s(X) \neq \mathbb{1}$ is called the *support* of the solution.

The support of a solution for an equation $\mathcal{E}$ may be assumed to be a subset of $F_{\operatorname{Var}(\mathcal{E})}$ and hence the data of a solution is equivalent to a map $\operatorname{Var}(\mathcal{E}) \to G$. The question whether an equation $\mathcal{E}$ is solvable will be referred to as the *Diophantine problem* of $\mathcal{E}$.
For every homomorphism $\varphi\colon G \to H$ there is a unique extension to an $F_{\mathcal{X}}$-homomorphism $\varphi_*\colon F_{\mathcal{X}} * G \to F_{\mathcal{X}} * H$. In this manner every G-equation $\mathcal{E}$ gives rise to an H-equation $\varphi_*(\mathcal{E})$, which is solvable whenever $\mathcal{E}$ is solvable.

Definition 3.2. Let $\mathcal{E}, \mathcal{F} \in F_{\mathcal{X}} * G$ be two G-equations. We say that $\mathcal{E}$ and $\mathcal{F}$ are *equivalent* if there is a G-automorphism φ of $F_{\mathcal{X}} * G$ that maps $\mathcal{E}$ to $\mathcal{F}$. We denote by $\operatorname{Stab}(\mathcal{E})$ the group of G-automorphisms of $\mathcal{E}$.

Lemma 3.3. *Let $\mathcal{E}$ be an equation and let φ be a G-endomorphism of $F_{\mathcal{X}} * G$. If $\varphi(\mathcal{E})$ is solvable then so is $\mathcal{E}$. In particular the Diophantine problem is the same for equivalent equations.*

Proof. If s is a solution for $\varphi(\mathcal{E})$, then $s \circ \varphi$ is a solution for $\mathcal{E}$. □

1.1. Quadratic equations. A G-equation $\mathcal{E}$ is called *quadratic* if for each variable $X \in \operatorname{Var}(\mathcal{E})$ exactly two letters of $\mathcal{E}$ are X or X^{-1}, when $\mathcal{E}$ is regarded as a reduced word.
A G-equation $\mathcal{E}$ is called *oriented* if for each variable $X \in \operatorname{Var}(\mathcal{E})$ the number of occurrences with positive and with negative sign coincide, namely if $\mathcal{E}$ maps to the identity under the natural map $F_{\mathcal{X}} * G \to F_{\mathcal{X}}/[F_{\mathcal{X}}, F_{\mathcal{X}}] * 1$. Otherwise $\mathcal{E}$ is called *unoriented*.

Lemma 3.4. *Being oriented or not is preserved under equivalence of equations.*

Proof. $\mathcal{E}$ is oriented if and only if it belongs to the normal closure of $[F_{\mathcal{X}}, F_{\mathcal{X}}] * G$; this subgroup is preserved by all G-endomorphisms of $F_{\mathcal{X}} * G$. □

2. Normal form of quadratic equations

Definition 3.5. For $m, n \geq 0$, $X_i, Y_i, Z_i \in \mathcal{X}$ and $c_i \in G$ the following two kinds of equations are called in *normal form*:

$$\mathcal{O}_{n,m}: \quad [X_1, Y_1][X_2, Y_2] \cdots [X_n, Y_n] c_1^{Z_1} \cdots c_{m-1}^{Z_{m-1}} c_m \tag{1}$$

$$\mathcal{U}_{n,m}: \quad X_1^2 X_2^2 \cdots X_n^2 c_1^{Z_1} \cdots c_{m-1}^{Z_{m-1}} c_m \ . \tag{2}$$

The form $\mathcal{O}_{n,m}$ is called the oriented case and $\mathcal{U}_{n,m}$ for $n > 0$ the unoriented case. The parameter n is referred to as *genus* of the normal form of an equation. The pair (n, m) will be called the *signature* of the quadratic equation.

We recall the following result, and give the details of the proof in an algorithmic manner, because we will need them in practice:

Theorem 3.6 ([**CE81, GK92**]). *Every quadratic equation $\mathcal{E} \in F_{\mathcal{X}} * G$ is equivalent to an equation in normal form, and the isomorphism can be effectively computed.*

Proof. The proof proceeds by induction on the number of variables. Starting with the oriented case: if the reduced equation $\mathcal{E}$ has no variables then it is already in normal form $\mathcal{O}_{0,1}$. If there is a variable $X \in \mathcal{X}$ occurring in $\mathcal{E}$ then X^{-1} also appears. Therefore the equation has the form $\mathcal{E} = uX^{-1}vXw$ or can be brought to this form by applying the automorphism $X \mapsto X^{-1}$. Choose $X \in \mathcal{X}$ in such a way that $\operatorname{Var}(v)$ is minimal.

We distinguish between multiple cases:

Case 1.0: $v \in G$. The word uw has fewer variables than $\mathcal{E}$ and can thus be brought into normal form $r \in \mathcal{O}_{n,m}$ by a G-isomorphism φ. If r ends with a variable, we use the G-isomorphism $\varphi \circ (X \mapsto Xw^{-1})$ to map $\mathcal{E}$ to the equation $rv^X \in \mathcal{O}_{n,m+1}$. If r ends with a group constant b, say $r = sb$, we use the isomorphism $\varphi \circ (X \mapsto Xbw^{-1})$ to map $\mathcal{E}$ to the equation $sv^X b \in \mathcal{O}_{n,m+1}$.

Case 1.1: $v \in \mathcal{X} \cup X^{-1}$. For simplicity let us assume $v \in \mathcal{X}$; in the other case we can apply the G-homomorphism $v \mapsto v^{-1}$. Now there are two possibilities: either v^{-1} occurs in u or v^{-1} occurs in w. In the first case $\mathcal{E} = u_1 v^{-1} u_2 X^{-1} v X w$, and then the G-isomorphism $X \mapsto X^{u_1} u_2$, $v \mapsto v^{u_1}$ yields the equation $[v, X] u_1 u_2 w$. In the second case $\mathcal{E} = uX^{-1}vXw_1v^{-1}w_2$ is transformed to $[X, v] u w_1 w_2$ by the G-isomorphism $X \mapsto X^{uw_1} w_1^{-1}$, $v \mapsto v^{-uw_1}$. In both cases u_1u_2w, respectively uw_1w_2 have fewer variables and so composition with the corresponding G-isomorphism results in a normal form.

Case 2: $\operatorname{Length}(v) > 1$. In this case v is a word consisting of elements $X \cup X^{-1}$ with each symbol occurring at most once as v was chosen with minimal variable set, and some elements of G. If v starts with a constant $b \in G$ we use the G-homomorphism $X \mapsto bX$ to achieve that v starts with a variable $Y \in \mathcal{X}$, possibly by using the G-homomorphism $Y \mapsto Y^{-1}$. As in Case 1.1 there are two possibilities: Y^{-1} is either part of u or part of w. In the first case $\mathcal{E} = u_1 Y^{-1} u_2 X^{-1} Y v_1 X w$ we can use the G-isomorphism $X \mapsto X^{u_1 v_1} u_2$, $Y \mapsto Y^{u_1 v_1} v_1^{-1}$ to obtain $[Y, X] u_1 v_1 u_2 w$. In the second we use the G-isomorphism $X \mapsto X^{uw_1v_1} v_1^{-1} w_1^{-1}$, $Y \mapsto Y^{-uw_1v_1} v_1^{-1}$ to obtain

$[X,Y]uw_1v_1w_2$. In both cases the second subword has again fewer variables and can be brought into normal form by induction.

Therefore each oriented equation can be brought to normal form by G-isomorphisms. In the unoriented case there is a variable $X \in \mathcal{X}$ such that $\mathcal{E} = uXvXw$. Choose v to have a minimal number of variables. By induction, the shorter word $uv^{-1}w$ is equivalent by φ to a normal form r.
The G-isomorphism $\varphi \circ (X \mapsto X^u v^{-1})$ maps $\mathcal{E}$ to $X^2 r$. If $r \in \mathcal{U}_{n,m}$ for some n, m, there remains nothing to do. Otherwise $r = [Y,Z]s$, and then the G-homomorphism

$$X \mapsto XYZ, \qquad Y \mapsto Z^{-1}Y^{-1}X^{-1}YZXYZ, \qquad Z \mapsto Z^{-1}Y^{-1}X^{-1}Z$$

maps X^2r to $X^2Y^2Z^2s$. This homomorphism is indeed an isomorphism, with inverse

$$X \mapsto X^2Y^{-1}X^{-1}, \qquad Y \mapsto XYX^{-1}Z^{-1}X^{-1}, \qquad Z \mapsto XZ.$$

Note that $s \in \mathcal{O}_{n,m}$. If $n \geq 1$ then this procedure can be repeated with Z, in place of X, r. □

For a quadratic equation $\mathcal{E}$ we denote by $\mathfrak{nf}(\mathcal{E}) := \mathfrak{nf}_{\mathcal{E}}(\mathcal{E})$ the image of $\mathcal{E}$ under the G-isomorphism $\mathfrak{nf}_{\mathcal{E}}$ constructed in the proof.

2.1. Constrained equations.

Definition 3.7 ([**LMU16**]). Given an equation $\mathcal{E} \in F_{\mathcal{X}} * G$, a group H, a homomorphism $\pi\colon G \to H$ and a homomorphism $\gamma\colon F_{\mathcal{X}} \to H$, the pair $(\mathcal{E}, \gamma)$ is called a *constrained* equation and γ is called a *constraint* for the equation $\mathcal{E}$ on H.
A *solution* for $(\mathcal{E}, \gamma)$ is a solution s for $\mathcal{E}$ with the additional property that $\pi \circ s = \gamma$.

We note that the constraint γ needs only be specified on $\mathrm{Var}(\mathcal{E})$.

2.2. Systems of equations.

Definition 3.8. A *system of equations* is a tuple $S \in (F * G)^n$. A *solution* for such a system is a G-homomorphism $s\colon F * G \to G$ such that for the projections π_i on the i-th coordinate the maps $s \circ \pi_i$ are solutions for the equations $\pi_i(S)$.
A *constraint* for a system S is a homomorphism $\gamma\colon F \to H$ such that $\gamma \circ \pi_i$ are constraints for the equations $\pi_i(S)$.

Definition 3.9. If for all $i \neq j$ the intersections of $\mathrm{Var}\pi_i(S) \cap \mathrm{Var}(\pi_j(S))$ are empty the system S is called an *independent system*.

For a G-homomorphism φ and a system of equations $S \in (F * G)^m$ by applying φ to S we mean applying the homomorphism to every component and hence identify φ with the m-fold direct product of φ.

CHAPTER 4

Decidable Equations

Definition 4.1. Let $G \leq H$ be two arbitrary groups. Two elements $g, h \in G$ are said to be *conjugate* in H if the equation $g^X h$ is solvable in H. We then write $g \sim_H h^{-1}$.
The element $f \in H$ such that $g^f = h$ is referred to as the *conjugator* of g and h.
If there exists an algorithm that decides for every pair of elements $g, h \in G$ if they are conjugate in G then we say that the group G has solvable *conjugacy problem*.

1. Commutator width

Let G be a group. It is well-known that usually elements of G' are not commutators—for example, $[X_1, X_2] \cdots [X_{2n-1}, X_{2n}]$ is not a commutator in the free group F_{2n} when $n > 1$.

Definition 4.2. The *commutator width* of a group G is the minimal $n \in \mathbb{N} \cup \infty$ such that every element of the commutator subgroup G' is a product of at most n commutators.

1.1. Commutator width of $\mathbf{Aut(T_2)}$. To give a general idea of a method to solve equations in self-similar groups we consider as an example the group of the binary tree T_2 and the group $\operatorname{Aut}(T_2)$. This group is layered and hence for two elements $g, h \in \operatorname{Aut}(T_n)$ the element $\langle\!\langle g, h \rangle\!\rangle$ is also a member of the group. This will allow us to solve an equation by solving the equation for the states and then lift the solution.

Proposition 4.3. *The commutator width of* $\operatorname{Aut}(T_2)$ *is* 1.

For the proof we need a small observation:

Lemma 4.4. *Let G be a self-similar group acting on a binary tree. If $g \in H'$ then $g@2 \cdot g@1 \in H'$.*

Proof. It suffices to consider a commutator $g = [g_1, g_2]$ in G'. Then $g@2 \cdot g@1$ is the product, in some order, of the eight terms $(g_i@j)^\varepsilon$ for all $i, j \in \{1, 2\}$ and $\varepsilon \in \{\pm 1\}$ each occurring exactly once. □

Proof of Proposition 4.3. Given any element $g \in \operatorname{Aut}(T_2)'$ we consider the equation $[X, Y]g$. If we replace in the equation the variable X by $\langle\!\langle X_1, X_2 \rangle\!\rangle$ and Y by $\langle\!\langle Y_1, Y_2 \rangle\!\rangle(1, 2)$ we obtain $\langle\!\langle X_1^{-1} Y_2^{-1} X_2 Y_2 g@1, X_2^{-1} Y_1^{-1} X_1 Y_1 g@2 \rangle\!\rangle$. Therefore, $[X, Y]g$ is solvable if the system of equations $\{X_2^{-1} Y_2^{-1} X_2 Y_2 g@1,\ X_1^{-1} Y_1^{-1} X_1 Y_1 g@2\}$ is solvable. We apply the $\operatorname{Aut}(T_2)$-homomorphism $X_1 \mapsto X_1, X_2 \mapsto Y_1^{-1} X_1 Y_1 g@2, Y_i \mapsto Y_i$ to eliminate one equation and one variable.
Therefore the solvability of the constrained equation $([X, Y]g, (X \mapsto \mathbb{1}, Y \mapsto (1, 2)))$ follows from the solvability of $X_1^{-1} Y_2^{-1} Y_1^{-1} X_1 Y_1 (g@2) Y_2 g@1$ which is under the normal form $\operatorname{Aut}(T_2)$-isomorphism $Y_1 \mapsto Y_1 Y_2^{-1}$ equivalent to the solvability of

$[X_1, Y_1](g@2)^{Y_2} g@1$. After choosing $Y_2 = \mathbb{1}$ we are again in the original situation because $g@2g@1 \in H'$.
This allows us to recursively define a solution s for the equation $[X, Y]g$ as follows:

$$s(X) = \langle\!\langle a_1, b_1^{-1} a_1 b_1 g@2 \rangle\!\rangle, \qquad s(Y) = \langle\!\langle b_1, \mathbb{1} \rangle\!\rangle(1, 2), \qquad c_1 = g@2 \cdot g@1,$$

and for all $i \geq 1$

$$a_i = \langle\!\langle a_{i+1}, b_{i+1}^{-1} a_{i+1} b_{i+1} c_i@2 \rangle\!\rangle, \qquad b_i = \langle\!\langle b_{i+1}, \mathbb{1} \rangle\!\rangle(1, 2), \quad c_{i+1} = c_i@2 \cdot c_i@1.$$

Note that the elements $a_i, b_i \in \mathrm{Aut}(T_2)$ are well-defined, although they are constructed recursively out of the a_j, b_j for *larger* j. Indeed, if one considers the recursions above for $i \in \{1, \dots, n\}$ and sets $a_{n+1} = b_{n+1} = \mathbb{1}$, one defines in this manner elements $a_1^{(n)}, b_1^{(n)} \in \mathrm{Aut}(T_2)$ which form Cauchy sequences, and therefore have well-defined limits $a_1 = \lim a_1^{(n)}$ and $b_1 = \lim b_1^{(n)}$. □

2. The branching homomorphism

The goal of this section is to give a map that reduces the question of decidability of an equation to the decidability of an equation in the states of the former one.

For this purpose we fix a self-similar group $G \overset{\Phi}{\hookrightarrow} G \wr \mathrm{S}_n$ and a group Q with a projection $\pi\colon G \twoheadrightarrow Q$.
We notate the following two free groups with its free generators:

$$F_{\mathbb{N}} = \langle X_\ell \mid \ell \in \mathbb{N} \rangle, \qquad F_{\mathbb{N} \times n} = \langle X_{\ell,k} \mid \ell \in \mathbb{N}, 1 \leq k \leq n \rangle.$$

Furthermore we enrich a constraint $\gamma_{\mathrm{con}}\colon F_{\mathbb{N}} \to Q$ with an activity and write an enriched constraint as a pair: $\gamma = (\gamma_{\mathrm{con}}, \gamma_{\mathrm{act}})\colon F_{\mathbb{N}} \to Q \times \mathrm{S}_n$ such that $\gamma_{\mathrm{act}}(x_\ell) \in \mathrm{act}(\pi^{-1}(\gamma_{\mathrm{con}}(x_\ell))$ for all $\ell \in \mathbb{N}$. Note that in our application Q will often be a quotient group of G and act will be well defined on Q and therefore γ_{act} will be uniquely defined by γ_{con}. In that cases we will omit γ_{act} from the notation.
We define the following homomorphism which we will call the *branching homomorphism*:

$$\begin{aligned} \Phi_\gamma\colon F_{\mathbb{N}} * G &\to (F_{\mathbb{N} \times n} * G) \wr \mathrm{S}_n \\ X_\ell \quad &\mapsto \langle\!\langle X_{\ell,1}, X_{\ell,2}, \dots, X_{\ell,n} \rangle\!\rangle \gamma_{\mathrm{act}}(X_i) \\ g &\mapsto \Phi(g). \end{aligned}$$

Consider a quadratic equation $\mathcal{E} \in F_{\mathbb{N}} * G$ together with an enriched constraint $\gamma = (\gamma_{\mathrm{con}}, \gamma_{\mathrm{act}})\colon F_{\mathbb{N}} \to H \times \mathrm{S}_n$ and denote by $\mathcal{E}_1, \dots, \mathcal{E}_n$ equations such that $\Phi_\gamma(\mathcal{E}) = \langle\!\langle \mathcal{E}_1, \dots, \mathcal{E}_n \rangle\!\rangle \tau$.
Note that if $\tau \neq \mathbb{1}$ then the pair (E, γ) is unsolvable and on the other hand if (E, γ) is solvable then so is the system of equations $(\mathcal{E}_i)_{i=1\dots n}$.
The system $(\mathcal{E}_i)_i$ now seems to be more complicated then the single equation $\mathcal{E}$, but with a few changes of variables we can obtain an independent system.
If G is a layered group, we can lift solutions from the new system to the old equation and don't need any constraints. In this case we will take the trivial group for H. An enriched constraint then consists only of the homomorphism $\sigma\colon F_{\mathbb{N}} \to \mathrm{S}_n$.

PROPOSITION 4.5. *Let G be a layered self-similar group $G \simeq G \wr P$ with $P \leq \mathrm{S}_n$ and $\mathcal{E} \in F_{\mathbb{N}} * G$ be a quadratic equation.*
Then $\mathcal{E}$ is solvable if and only if there is a $\sigma\colon F_{\mathbb{N}} \to P$ such that $\mathrm{act}(\Phi_\sigma(E)) = \mathbb{1}$ and the system $\Phi_\sigma(\mathcal{E})$ is solvable.

Proof. If s is a solution for $\mathcal{E}$, define $\sigma\colon X_\ell \mapsto \operatorname{act}(s(X_\ell))$ then $t\colon X_{\ell,k} \mapsto s(X_\ell)@k$ is a solution for $\Phi_\sigma(\mathcal{E})$.
On the other hand if there is a $\sigma\colon F_\mathbb{N} \to \mathrm{S}_n$ such that $\operatorname{act}(\Phi_\sigma(\mathcal{E})) = \mathbb{1}$ and t is a solution for $\Phi_\sigma(\mathcal{E})$, then $s\colon X_\ell \mapsto \langle\!\langle t(X_{\ell,1}), \dots, t(X_{\ell,n})\rangle\!\rangle \sigma(X_\ell)$ defines a solution for $\mathcal{E}$. □

The idea now is to generalize this proposition for regular branch groups G. Fix a regular branched group $G \hookrightarrow G \wr \mathrm{S}_n$, a branch structure (K, Q, π, Q_1, ω) for G and an equation $\mathcal{E} \in F_\mathbb{N} * G$ in m variables. We denote by $F_\mathcal{E}$ the rank m free subgroup of $F_\mathbb{N}$ generated by $\operatorname{Var}(\mathcal{E})$. Similarly we denote by $F_{\mathcal{E}\times n}$ the free subgroup of $F_{\mathbb{N}\times n}$ with generating set $\{X_{\ell,k} \mid X_\ell \in \operatorname{Var}(\mathcal{E}), 1 \leq k \leq n\}$.
Let us assume that K has the property that $\operatorname{act}(k) = \mathbb{1}$ for all $k \in K$. Then any constraint $\gamma\colon F_E \to Q$ yields a map $\gamma_{\text{act}}\colon F_E \to \mathrm{S}_n$.
Let us for now fix a constraint $\gamma\colon F_E \to Q$ such that $\operatorname{act}(\Phi_\gamma(\mathcal{E})) = \mathbb{1}$.
If we have a solution s for the system of equations $\Phi_\gamma(\mathcal{E})$ with the property that

$$(1)\quad g_\ell := \langle\!\langle s(X_{\ell,1}), \dots, s(X_{\ell,n})\rangle\!\rangle \gamma_{\text{act}}(X_\ell) \in G \text{ and } \pi\left(g_\ell)\right) = \gamma(X_\ell) \text{ for all } 1 \leq \ell \leq m$$

we can lift the solution s to a solution $t\colon X_\ell \mapsto g_\ell$ for $(\mathcal{E}, \gamma)$.
This property can be rewritten in terms of constraints of s. Denote by $\gamma'\colon F_{\mathcal{E}\times n} \to Q$ a constraint such that

$$\langle\!\langle \gamma'(X_{\ell,1}), \dots, \gamma'(X_{\ell,n})\rangle\!\rangle \gamma_{\text{act}}(X_\ell) \in \omega^{-1}(\gamma(X_\ell)).$$

A solution s for the constrained equation $(\Phi_\gamma(\mathcal{E}), \gamma')$ fulfills property (1). Since Q_1 is finite, there are only finitely many such constraints γ'. We will denote by $\Gamma_1(\gamma)$ the set of all those constraints γ'. Furthermore for the system $(\Phi_\gamma(\mathcal{E}), \gamma')$ to be solvable it is necessary that the equation is solvable modulo K. Therefore we loose no solutions if we restrict the set of constraints to

$$\Gamma_1^\mathcal{E}(\gamma) = \{\gamma' \in \Gamma_1 \mid (\gamma' * \pi)(\mathcal{E}_k) = \mathbb{1}\ \forall k \in \mathbb{N}\}.$$

With this notation an analog for Proposition 4.5 for branch groups is:

Proposition 4.6. *Let G be a regular branched group with fixed branch structure (K, Q, π, Q_1, ω) such that $\operatorname{act}(K) = \mathbb{1}$ and let $(\mathcal{E}, \gamma)$ be a quadratic equation with constraint $\gamma\colon F_\mathbb{N} \to Q$.*
Then $(\mathcal{E}, \gamma)$ is solvable if and only if $\operatorname{act}(\Phi_\gamma(\mathcal{E})) = \mathbb{1}$ and one of the constrained equation systems $(\Phi_\gamma(\mathcal{E}), \gamma')$ for $\gamma' \in \Gamma_1^\mathcal{E}(\gamma)$ is solvable.

The next step is to reduce the constrained system of equations to an independent system without affecting the solvability.
Let G be a regular branch group with fixed branch structure as in the last proposition, let $\mathcal{E}$ be a quadratic equation and $\Phi_\gamma(\mathcal{E}) = \langle\!\langle \mathcal{E}_1, \dots, \mathcal{E}_n\rangle\!\rangle$ for a constraint γ.
If for $i \neq j$ there is $X \in \operatorname{Var}(\mathcal{E}_i) \cap \operatorname{Var}(\mathcal{E}_j)$ then $X \notin \operatorname{Var}(\mathcal{E}_k)$ for all $k \neq i, j$ since $\mathcal{E}$ is quadratic. We can assume without loss of generality that X occurs in $\mathcal{E}_i$ (otherwise apply the G-isomorphism $X \mapsto X^{-1}$) and then $\mathcal{E}_i = V_1 X V_2$ and $\mathcal{E}_j = W_1 Y W_2$ with $Y \in \{X^{-1}, X\}$ depending whether $\mathcal{E}$ is oriented or not.
Applying the G-homomorphism $\varphi\colon X \mapsto V_1^{-1} V_2^{-1}$ to the system $(\mathcal{E}_i)_{i=1\dots n}$ trivializes one equation. The solvability however stays the same.

Lemma 4.7. *For a constraint $\gamma' \in \Gamma_1^\mathcal{E}(\gamma)$ the system $((\mathcal{E}_1, \dots, \mathcal{E}_n), \gamma')$ is solvable if and only if the system $((\varphi(\mathcal{E}_1), \dots, \varphi(\mathcal{E}_n)), \gamma')$ is solvable.*

Proof. If s is a solution for $((\varphi(\mathcal{E}_i))_{i=1\dots n}, \gamma')$ we can define the G-evaluation

$$t\colon X_i \mapsto \begin{cases} s(X_i) & \text{if } X_i \neq X \\ s(V_1^{-1}V_2^{-1}) & \text{if } X_i = X \end{cases}.$$

The homomorphism t is a solution for $(\mathcal{E}_i)_{i=1\dots n}$ and because $(\gamma' * \pi)(V_1^{-1}V_2^{-1}) = \gamma'(X)$ it solves $((\mathcal{E}_i)_{i=1\dots n}, \gamma')$. On the other hand if t is a solution for $(\mathcal{E}_i)$ then $t(X)$ is forced to be identical to $t(V_1^{-1}V_2)$. □

This process can be repeated until there are no equations with a common set of variables left.
The appearance of the resulting system of equations can depend on the order in which the common variable are removed; the solvability however is equivalent under all such choices, as seen earlier. With a fixed enumeration on the generators of $F_{\mathbb{N}}$ we can choose to always eliminate the smallest common variable and in that way determine a well defined homomorphism φ that replaces the system $S = \Phi_\gamma(\mathcal{E})$ by an equivalent independent system.

Notation 4.8. We denote by $\Phi_\gamma^\varphi(\mathcal{E})$ the reduced system $(\varphi(\mathcal{E}_1), \dots, \varphi(\mathcal{E}_n))$.

Lemma 4.9. *Let $\mathcal{E} \in F_{\mathbb{N}} * G$ be a quadratic equation with a constraint γ and $\gamma' \in \Gamma_1^{\mathcal{E}}(\gamma)$ be a derived constraint then the solvability of $\Phi_\gamma(\mathcal{E})$ is equivalent to the solvability of $\Phi_\gamma^\varphi(\mathcal{E})$.*

Proof. This is just an iteration of Lemma 4.7. □

Lemma 4.10. *Let $\mathcal{E} \in F_{\mathbb{N}} * G$ be a quadratic equation with a constraint γ and $\Phi_\gamma^\varphi(\mathcal{E}) = (\mathcal{E}_1, \dots, \mathcal{E}_n)$ then $\mathcal{E}_i$ is again a quadratic equation for every $i = 1, \dots, n$. If $\mathcal{E}$ is oriented then also all $\mathcal{E}_i$ are oriented.*

Proof. Let $X_\ell \in \operatorname{Var}(\mathcal{E})$ be a variable then every variable $X_{\ell,k}$ for $k = 1, \dots, n$ will occur in the system $\Phi_\gamma(X_\ell)$ exactly once and hence every variable $X_{\ell,k}$ for $\ell \in \mathbb{N}$ will occur either twice or not at all in the system $\Phi_\gamma(\mathcal{E})$ and since $\Phi_\gamma^\varphi(\mathcal{E})$ is an independent system every equation in the system is quadratic.
If $\mathcal{E}$ is oriented then $X_{\ell,k}^{-1}$ for $k = 1, \dots, n$ will occur in the system $\Phi_\gamma(X_\ell^{-1})$ exactly once. The result follows. □

Example 4.11. It is not true that all nontrivial components of $\Phi_\gamma(\mathcal{E})$ are unoriented if $\mathcal{E}$ is unoriented: Consider $\gamma\colon X, Y \mapsto (), Z \mapsto (1, 2)$ and $\mathcal{E} = [X, Y]Z^2 \sim X^2Y^2Z^2$ then $\Phi_\gamma^\varphi(\mathcal{E})$ has a nontrivial oriented component.

Let us again write $\Phi_\gamma^\varphi(\mathcal{E}) = (\mathcal{E}_1, \dots, \mathcal{E}_n)$. According to the last lemma we can find a G-isomorphism $\psi_{\mathcal{E},\gamma}$ that maps each $\mathcal{E}_i$ to a normal form in $F_{\mathbb{N}\times n} * G \simeq F_{\mathbb{N}} * G$.

Notation 4.12. We denote by $\Phi_\gamma^{\mathrm{nf}}(\mathcal{E})$ the system $(\psi_{E,\gamma}(\mathcal{E}_1), \dots, \psi_{E,\gamma}(\mathcal{E}_n))$.

The system $\Phi_\gamma^{\mathrm{nf}}(\mathcal{E})$ is an independent system of quadratic equations and we have the following result:

Proposition 4.13. *Let G be a regular branched group with fixed branch structure (K, Q, π, Q_1, ω) such that $\operatorname{act}(K) = \mathbb{1}$ and let $\mathcal{E}$ be a quadratic equation with constraint $\gamma\colon F_{\mathbb{N}} \to Q$.*
The constrained equation $(\mathcal{E}, \gamma)$ is solvable if and only if there is

$$\gamma'' \in \{(\gamma' * \pi) \circ \psi_{\mathcal{E},\gamma}\colon F_{\mathbb{N}} \to Q \mid \gamma' \in \Gamma_1^{\mathcal{E}}(\gamma)\}$$

such that the system $(\Phi_\gamma^{\mathrm{nf}}(\mathcal{E}), \gamma'')$ is solvable.

For layered groups this proposition becomes much simpler since property (1) is always satisfied.

Corollary 4.14. *Let $G \leq \mathrm{Aut}(\mathcal{A}^*)$ be a layered group and $\mathcal{E} \in F_{\mathbb{N}} * G$ be an equation. Then $\mathcal{E}$ is solvable if and only if there is an enriched constraint $\sigma \colon F_{\mathbb{N}} \to \mathrm{Sym}(\mathcal{A})$ with $(\sigma * \mathrm{act})(\mathcal{E}) = \mathbb{1}$ such that the system $\Phi_\sigma^{\mathrm{nf}}(\mathcal{E})$ is solvable.*

This proposition is in applications useful if the equations in $\Phi_\sigma^{\mathrm{nf}}(\mathcal{E})$ are in a certain sense easier then the equation $\mathcal{E}$.
To demonstrate the use of this notion even without this premise we prove a more general version of Proposition 4.3:

Proposition 4.15. *The group $\mathrm{Aut}(T)$ has commutator width 1 for every tree T.*

We have the same lemma as before:

Lemma 4.16. *If $G \leq \mathrm{Aut}(\mathcal{A}^*)$ is a self-similar group and $g \in G'$ then $\prod_{x \in \mathcal{A}} g@x \in G'$.*

Proof. We only need to show this for a commutator $g = [f, h]$.

$$\prod_{x \in \mathcal{A}} g@x = \prod_{x \in \mathcal{A}} (h@x^{h^{-1}})^{-1} (h@x^{h^{-1}f^{-1}})^{-1} (h@x^{h^{-1}f^{-1}})(f@x^{h^{-1}f^{-1}h}).$$

We can obtain the identity by reordering the factors of this product and hence $\prod_{x \in \mathcal{A}} g@x \in G'$. □

Proof of 4.15. We have noted already that the group $\mathrm{Aut}(\mathcal{A}^*)$ is layered. Let us denote by n the cardinality of $\mathcal{A}$. For every element $g \in \mathrm{Aut}(\mathcal{A}^*)$ we want to solve the equation $[X, Y]g$ and for that purpose consider the enriched constraint $\sigma \colon X \mapsto (), Y \mapsto (1, 2, \ldots, n)$. Then $\mathcal{E}$ is solvable if the system $\Phi_\sigma(\mathcal{E}) = \langle\!\langle \mathcal{E}_1, \ldots, \mathcal{E}_n \rangle\!\rangle$ is solvable. We have common variables in $\mathcal{E}_i$ and $\mathcal{E}_{i+1}$ for all i modulo n and hence without doing any further calculations we know that the system is equivalent to a single quadratic equation involving all states of g.
By eventually sending some variables to the identity we see that the solvability of the equation is implied by the solvability of $[X_1, Y_1] \prod_{x \in \mathcal{A}} (g@x)$ which is of the same form as our initial equation. Now we can iterate this procedure and with the same argument as in the proof of Lemma 4.3 the limit of the corresponding Cauchy sequence is a solution to the initial equation. □

Looking again at the proof we obtain the following corollary:

Corollary 4.17. *Let $G \leq \mathrm{Aut}(\mathcal{A}^*)$ be a layered, self-similar group and $\widetilde{G}$ be its completion with respect to the topology of $\mathrm{Aut}(\mathcal{A}^*)$. If the group $\mathrm{Perm}(G)$ has commutator width 1 then for every element $g \in G'$ the equation $[X, Y]g$ has a solution in $\widetilde{G}$.*

3. Alternating group

We will use the branching homomorphism Φ from the last section to show that every element in the Neumann-Segal group is a commutator. For this we also need the corresponding computations in the alternating groups A_n.

Theorem 4.18 ([**Mil99**]). *For $n \geq 5$ each group A_n is perfect and of commutator width 1.*

This theorem is a special case of Ore's theorem, conjectured by Ore in [**Ore51**] and finally proved in [**LOST10**]. Here the proof by Miller is sketched again to give the explicit formulas for the commutating elements. Some special cases are altered a bit to be of use in Proposition 4.20.

Proof. We have the identity $[x,y]^a = [x^a, y^a]$ for every element x, y, a in a group G and hence it suffices to show that for every conjugacy class of S_n that is in A_n there is a representative which is a commutator. We distinguish 10 cases:

(1) $z = (1, 2, \dots, 2r+1)$ with $r > 1$ even,
(2) $z = (1, 2, \dots, 2r+1)$ with $r > 1$ odd,
(3) $z = (1, 2, \dots, 2s)(2s+1, 2s+2, \dots, 2r)$ with $0 < s \leq \frac{r}{2}$, r even,
(4) $z = (1, 2, \dots, 2s)(2s+1, 2s+2, \dots, 2r)$ with $0 < s \leq \frac{r-1}{2}$, r odd,
(5) $z = (1, \dots 2r+1)(2r+2, 2r+3, 2r+4)$ with $r > 1$ even if $n > 5$,
(6) $z = (1, \dots 2r+1)(2r+2, 2r+3, 2r+4)$ with $r > 1$ odd if $n > 5$,
(7) $z = (1, 2, \dots, 2s)(2s+1, 2s+2, \dots, 2r)(2r+1, 2r+2, 2r+3)$ with $0 < s \leq \frac{r}{2}$, r even if $n > 5$,
(8) $z = (1, 2, 3)(4, \dots, 2r+3)(2r+4, \dots, 2r+2s+3)$ with $0 < s \leq \frac{r-1}{2}$, r odd if $n > 5$,
(9) $z = (1, 2, 3)(4, 5, 6)$ if $n > 5$,
(10) $z = (1, 2, 3)$.

The idea in general is as follows: Write z depending on r even or odd as product of two cycles of the same (odd) length with either one or three intersections. Then one cycle is conjugate by an element of A_n to the inverse of the other and thus the product is a commutator.

Case 1. As $z = (1, \dots, r+1) \cdot (1, r+2, \dots, 2r+1)$ and $(1, \dots, r+1)^{-1} = (1, r+2, 2r+1)^c$ with $c = (2, 2r+1)(3, 2r) \dots (r+1, r+2)$ it holds that

$$z = [\ (2, 2r+1)(3, 2r) \dots (r+1, r+2)\ ,\ (1, r+2, r+3, \dots, 2r+1)\ ]\ .$$

Case 2. As $z = (1, r+2, 2, 3, \dots, r+1) \cdot (1, r+2, 2, r+3, r+4, \dots, 2r+1)$ and $(1, r+2, 2, 3, \dots, r+1)^{-1} = (1, r+2, 2, r+3, r+4, \dots, 2r+1)^c$ with $c = (1, 2)(2r+1, 3, 2r, 4)(5, 2r-1)(6, 2r-2) \dots (r+1, r+3)$ it holds that

$$z = [(1, 2)(2r+1, 3, 2r, 4)(5, 2r-1) \dots (r+1, r+3)\ ,\ (1, r+2, 2, r+3, \dots, 2r+1)].$$

Case 3. As $z = (1, 2, 3, \dots, r+1) \cdot (1, r+2, r+3, \dots, 2r-1, 2r, 2s+1)$ and $(1, 2, 3, \dots, r+1)^{-1} = (1, r+2, r+3, \dots, 2r-1, 2r, 2s+1)^c$ with $c = (2s+1, 2)(2r, 3)(2r-1, 4) \dots (r+2, r+1)$ it holds that

$$z = [(2s+1, 2)(2r, 3)(2r-1, 4) \dots (r+2, r+1)\ ,\ (1, r+2, r+3, \dots, 2r, 2s+1)].$$

Case 4. As $z = (1, r+2, 2, 3, \dots, r+1) \cdot (1, r+2, 2, r+3, r+4, \dots, 2r, 2s+1)$ and $(1, 2, 3, \dots, r+1)^{-1} = (1, r+2, 2, r+3, r+4, \dots, 2r-1, 2r, 2s+1)^c$ with $c = (1, 2)(2s+1, 3, 2r, 4)(2r-1, 5) \dots (r+3, r+1)$ if $s > 1$ and $c = (1, 2)(2r, 4)(2r-1, 5) \dots (r+3, r+1)$ respectively if $s = 1$ it holds that

$$\begin{aligned} z = [\ &(1, 2)(2s+1, 3, 2r, 4)(2r-1, 5) \dots (r+3, r+1)\ , \\ &(1, r+2, 2, r+3, r+4, \dots, 2r, 2s+1)\ ] \quad \text{if } s > 1, \\ z = [\ &(1, 2)(2r, 4)(2r-1, 5) \dots (r+3, r+1)\ , \\ &(1, r+2, 2, r+3, r+4, \dots, 2r, 2s+1)\ ] \quad \text{if } s = 1. \end{aligned}$$

Case 5. As we have the same decomposition like in case 1 but with c not in A_n but $c = (2r+1, 2, 2r, 3)(4, 2r-1)\dots(r+1, r+2)$ we have

$$\begin{aligned} z = [\ &(2r+1, 2, 2r, 3)(4, 2r-1)\dots(r+1, r+2)(2r+2, 2r+4)\ ,\\ &\qquad (1, r+2, r+3, \dots, 2r+1)(2r+2, 2r+4, 2r+3)\ ]\ . \end{aligned}$$

Case 6. As we have the same decomposition like in case 2 but with c not in A_n but $c = (1, 2)(3, 2r+1)(4, 2r)(5, 2r-1)\dots(r+1, r+3)$ we have

$$\begin{aligned} z = [\ &(1, 2)(3, 2r+1)(4, 2r)\dots(r+1, r+3)(2r+2, 2r+4)\ ,\\ &\qquad (1, r+2, r+3, \dots, 2r+1)(2r+2, 2r+4, 2r+3)\ ]\ . \end{aligned}$$

Case 7. In this case some sub-cases need to be considered. We take the same decomposition as in case 3 but a different c:

$$\begin{aligned} s > 1 :&\quad c = (2s+1, 2, 2r, 3)(2r-1, 4)(2r-1, 5)\dots(r+2, r+1),\\ s = 1, r > 2 :&\quad c = (3, 2)(2r, 3, 2r-1, 4)(2r-2, 5)\dots(r+2, r+1). \end{aligned}$$

Then

$$z = [\ c\cdot(2r+1, 2r+3)\ ,\ (1, r+2, r+3, \dots, 2r, 2s+1)\cdot(2r+1, 2r+3, 2r+2)\ ]$$

For $s = 1$ and $r = 2$ choose

$$z = [\ (1, 2)(2r+1, 2r+3)\ ,\ (1, 3)(2, 4)(2r+1, 2r+3, 2r+2)\ ].$$

Case 8. Like in case 7 some sub-cases need to be considered. We take the same decomposition as in case 4 but a different c:

$$\begin{aligned} s > 1 :&\quad c = (1, 2)(2s+1, 3)(2r, 4)(2r-1, 5)\dots(r+3, r+1),\\ s = 1, r > 3 :&\quad c = (1, 2)(2r, 4, 2r-1, 5)(2r-2, 6)\dots(n+3, n+1). \end{aligned}$$

Then

$$z = [\ c\cdot(2r+1, 2r+3)\ ,\ (1, r+2, r+3, \dots, 2r, 2s+1)\cdot(2r+1, 2r+3, 2r+2)\ ].$$

For $s = 1$ and $r = 3$ choose

$$z = [\ (1, 4, 3, 6)(2r+1, 2r+3)\ ,\ (1, 2, 3, 5, 4)(2r+1, 2r+3, 2r+2)\ ].$$

Case 9.

$$z = [\ (1, 3)(5, 6)\ ,\ (1, 3, 2, 6, 4)\ ].$$

Case 10.

$$z = [\ (1, 3, 5)\ ,\ (1, 3, 5, 2, 4)\ ].$$

All elements of A_n are products of conjugates of the Cases 1-9 or conjugate to Case 10 and with pairwise disjoint support. Therefore for $n \geq 5$ every element of A_n is a commutator of two elements of A_n. □

For the use in the following denote by $C\colon A_n \to A_n \times A_n$ the map which sends an element σ to (π, τ) where π and τ are chosen as in the previous proof such that $[\pi, \tau] = \sigma$.

Corollary 4.19. *If σ is not a 3-cycle then*

$$\operatorname{supp}(C(\sigma)) \subset \operatorname{supp}(\sigma) \times \operatorname{supp}(\sigma).$$

We have implemented this function C in GAP in the file `examples/Ore.g`.

```
gap> GetCommutators((1,2,3)(4,5)(7,8,9,10));
```

```
[ (1,3)(4,8,7,10), (1,3,2)(4,5,7,9,8) ]
```

4. Neumann-Segal group

The calculations done in the previous section are of use to compute the commutator width in some self-similar groups which are generated by copies of A_n's. For this the following proposition is of importance:

PROPOSITION 4.20. *Let $n \geq 5$, $\mathcal{A}$ of size n and $G < \mathrm{Aut}(\mathcal{A}^*) \wr A_n$ and $g \in G$. Then there is a homomorphism $\sigma\colon F_{\mathbb{N}} \to A_n$ such that $\Phi_\sigma^{\mathrm{nf}}([X,Y]g) = \langle\!\langle \mathcal{E}_1, \dots, \mathcal{E}_n \rangle\!\rangle()$ where each $\mathcal{E}_i$ has either genus greater 1, is trivial or of the form $[X_i, Y_i]g@x_i$ for some $x_i \in \mathcal{A}$.*
The map σ can be chosen such that $(\sigma(X), \sigma(Y)) = C(\mathrm{act}(g)^{-1})$ with the map C from the proof of Theorem 4.18.

PROOF. Let $g \in G$ be as given in the proposition and let the homomorphism σ be such that $(\sigma(X), \sigma(Y)) = C(\mathrm{act}(g)^{-1})$. If $j \notin \mathrm{supp}(\mathrm{act}(g))$ then by Corollary 4.19 it is $\Phi_\sigma([X,Y]g)@j = [X_j, Y_j]g@j$.
We denote for an easier notation $g = \langle\!\langle g_1, \dots, g_n \rangle\!\rangle\tau$ and $\Phi_\sigma(X) = \langle\!\langle X_1, \dots, X_n \rangle\!\rangle\sigma(X)$ and $\Phi_\sigma(Y) = \langle\!\langle Y_1, \dots, Y_n \rangle\!\rangle\sigma(Y)$. Now the proof goes along the same 10 cases for τ as the proof of Theorem 4.18.
In the first case it is $\tau = (1, 2, \dots, 2r+1)^{-1}$ with $1 < r \leq n$ even. Then $\sigma(x) = (2, 2r+1)(3, 2r)\dots(r+1, r+2)$ and $\sigma(y) = (1, r+2, r+3, \dots, 2r+1)$. Thus

$$\begin{aligned}\Phi_\sigma([X,Y]g) = \langle\!\langle & X_1^{-1}Y_{2r+1}^{-1}X_{2r+1}Y_2g_2, \\ & X_{2r+1}^{-1}Y_{2r}^{-1}X_{2r}Y_3g_3, \dots, X_{r+3}^{-1}Y_{r+2}^{-1}X_{r+2}Y_{r+1}g_{r+1}, \\ & X_{r+2}^{-1}Y_1^{-1}X_1Y_1g_{r+2}, \\ & X_{r+1}^{-1}Y_{r+1}^{-1}X_{r+1}Y_{r+2}g_{r+3}, \dots X_3^{-1}Y_3^{-1}X_3Y_{2r}g_{2r+1}, \\ & X_2^{-1}Y_2^{-1}X_2Y_{2r+1}g_1, \\ & [X_{2r+2}, Y_{2r+2}]g_{2r+2}, \dots, [X_n, Y_n]g_n\rangle\!\rangle.\end{aligned}$$

The G-homomorphism

$$\begin{array}{lrlrl}\varphi\colon & X_{2r+1} & \mapsto Y_{2r}^{-1}X_{2r}Y_3g_3, & Y_{r+1} & \mapsto X_{r+1}Y_{r+2}g_{r+3}X_{r+1}^{-1}, \\ & & \vdots & & \vdots \\ & X_{r+3} & \mapsto Y_{r+2}^{-1}X_{r+2}Y_{r+1}g_{r+1}, & Y_3 & \mapsto X_3Y_{2r}g_{2r+1}X_3^{-1}, \\ & X_{r+2} & \mapsto Y_1^{-1}X_1Y_1g_{r+2}, & Y_2 & \mapsto X_2Y_{2r+1}g_1X_2^{-1},\end{array}$$

maps the system $\Phi_\sigma([X,Y]g)$ to

$$\begin{aligned}\Phi_\sigma^\varphi([X,Y]g) = \langle\!\langle X_1^{-1} & \left(\prod_{j=0}^{r-1} Y_{2r+1-j}^{-1}\right) Y_1^{-1}X_1Y_1g_{r+2} \\ & \cdot \left(\prod_{j=0}^{r-2} X_{r+1-j}Y_{r+2+j}g_{r+3+j}X_{r+1-j}^{-1}g_{r+1-j}\right) X_2Y_{2r+1}g_1X_2^{-1}g_2, \\ & 1, \dots, 1, [X_{2r+2}, Y_{2r+2}]g_{2r+2}, \dots, [X_n, Y_n]g_n\rangle\!\rangle.\end{aligned}$$

So all but the first state are already in the requested situation. After normalizing the first state we obtain:

$$\Phi_\sigma^{\mathrm{nf}}([X,Y]g)@1 = [X_1, Y_1][X_{r+1}, Y_{r+2}][X_r, Y_{r+3}]\cdots[X_2, Y_{2r+1}]g_1g_{2r+1}g_{2r}\cdots g_3g_2.$$

This has genus $r+1>2$.

$$\operatorname{genus}(\Phi_\sigma([X,Y]g)@i)=\begin{cases} r+1 & \text{for } i=1\\ 0 & \text{for } 1<i\le 2r+1\\ 1 & \text{for } 2r+1<i\le n.\end{cases}$$

Similar computations lead to the genera in the other cases:
If $\tau^{-1}=(1,2,\dots,2r+1)$ with r odd then we have:

$$\operatorname{genus}(\Phi_\sigma([X,Y]g)@i)=\begin{cases} r+1 & \text{for } i=1\\ 0 & \text{for } 1<i\le 2r+1\\ 1 & \text{for } 2r+1<i\le n.\end{cases}$$

If $\tau^{-1}=(1,2,\dots,2s)(2s+1,2s+2,\dots,2r)$ with $0<s\le\frac{r}{2}$, r even or with $0<s\le\frac{r-1}{2}$, r odd we have:

$$\operatorname{genus}(\Phi_\sigma([X,Y]g)@i)=\begin{cases} r & \text{for } i=1\\ 0 & \text{for } 1<i\le 2r\\ 1 & \text{for } 2r<i\le n.\end{cases}$$

If $\tau^{-1}=(1,\dots 2r+1)(2r+2,2r+3,2r+4)$ with $r>1$ even or odd we have:

$$\operatorname{genus}(\Phi_\sigma([X,Y]g)@i)=\begin{cases} r+1 & \text{for } i=1\\ 0 & \text{for } 1<i\le 2r+1\\ 2 & \text{for } i=2r+2\\ 0 & \text{for } 2r+2<i\le 2r+4\\ 1 & \text{for } 2r+4\le n.\end{cases}$$

If $\tau^{-1}=(1,2,\dots,2s)(2s+1,2s+2,\dots,2r)(2r+1,2r+2,2r+3)$ with $0<s\le\frac{r}{2}$, r even or with $0<s\le\frac{r-1}{2}$, r odd:

$$\operatorname{genus}(\Phi_\sigma([X,Y]g)@i)=\begin{cases} r+1 & \text{for } i=1\\ 0 & \text{for } 1<i\le 2r\\ 2 & \text{for } i=2r+1\\ 0 & \text{for } 2r+1<i\le 2r+3\\ 1 & \text{for } 2r+3<i\le n.\end{cases}$$

If $\tau^{-1}=(1,2,3)(4,5,6)$ we have:

$$\operatorname{genus}(\Phi_\sigma([X,Y]g)@i)=\begin{cases} 3 & \text{for } i=1\\ 0 & \text{for } 1<i\le 6\\ 1 & \text{for } 6<i\le n.\end{cases}$$

If $\tau^{-1}=(1,2,3)$ we have:

$$\operatorname{genus}(\Phi_\sigma([X,Y]g)@i)=\begin{cases} 2 & \text{for } i=1\\ 0 & \text{for } 1<i\le 5\\ 1 & \text{for } 5<i\le n.\end{cases}$$

□

Remark 4.21. In the proof the calculations are done for a very specific σ; in fact many choices for σ with $[\sigma(X), \sigma(Y)]\tau = 1$ lead to the same result. With $\sigma(Y)$ being a large enough cycle all involved pairs of states share a common symbol and thus they can be brought to a form with just one nontrivial state. To obtain a maximal genus it is sufficient that $\sigma(X)$ consists of disjoint transpositions.

4.1. An example.

We will now apply Proposition 4.20 to the Neumann-Segal groups defined in Chapter 2, Section 5.3 to compute its commutator width. To avoid too many indices we will in this section fix an n and write $\mathfrak{N} := \mathfrak{N}_n$ with the generating subgroups $N := N_n$, $M := M_n$.

Definition 4.22. Let G be a contracting self-similar group over an alphabet $\mathcal{A}$. Denote by $d_{\mathcal{N}}(g)$ the distance from an element $g \in G$ to the nucleus:

$$d_{\mathcal{N}}(g) := \min\left\{n \in \mathbb{N} \mid g@w \in \mathcal{N}(G)\ \forall w \in \mathcal{A}^n\right\}.$$

Lemma 4.23. *The distance $d_{\mathcal{N}}$ can be computed recursively:*

$$d_{\mathcal{N}}(g) = \begin{cases} 0 & \text{if } g \in \mathcal{N}(G), \\ \max\{d_{\mathcal{N}}(g@x) \mid x \in \mathcal{A}\} + 1 & \text{else.} \end{cases}$$

□

Since $\mathfrak{N}$ is generated by the two finite subgroups N and M, every element $g \in \mathfrak{N}$ can be written in the form

$$(*) \qquad g = \left(\prod_{i=1}^{s} n_i m_i\right) n_{s+1} \quad \text{with } n_i \in N, m_i \in M.$$

Lemma 4.24. $d_{\mathcal{N}}(n_1 g n_2) \leq \max\{d_{\mathcal{N}}(g), 1\}$ *for all $g \in \mathfrak{N}$ and $n_i \in N$.*

Proof. If $d_{\mathcal{N}}(n_1 g n_2) \leq 1$ nothing is to show. Otherwise $n_1 g n_2 \notin \mathcal{N}(\mathfrak{N})$ and so $d_{\mathcal{N}}(n_1 g n_2) = \max\{d_{\mathcal{N}}(g@x) + 1 \mid x \in \mathcal{A}\}$ but the sets of states of g and $n_1 g n_2$ are identical, therefore the inequality follows. □

Definition 4.25. We call the minimal s such that g can be written in the form $(*)$ the M-length of g and notate:

$$\#_M(g) := \min\left\{s \in \mathbb{N} \,\middle|\, g = \left(\prod_{i=1}^{s} n_i m_i\right) n_{s+1} \text{ with } n_i \in N, m_i \in M\right\}.$$

Lemma 4.26.

(1) $\#_M(g@x) \leq \left\lceil \frac{\#_M(g)}{2} \right\rceil$ *for all x in $\mathcal{A}$ and all $g \in \mathfrak{N}$.*

(2) $\#_M\left(\prod_{x\in\mathcal{A}} g@\pi(x)\right) \leq \#_M(g)$ *for all $\pi \in \mathrm{S}_n$ and all $g \in \mathfrak{N}$.*

Proof. Every element from M has at most two non trivial states. One belongs to M and another belongs to N. Elements from N have only trivial states thus the states of a group element of M-length r have in sum at most again r members of M. This proves (2).
The product of states has maximal M-length when the elements from M and N alternate thus the maximal M-length of a single state is bounded by $\lceil \#_M g/2 \rceil$. □

Proposition 4.27. $d_{\mathcal{N}}(g) \leq \max\{0, \lceil \log_2(\#_M g) \rceil + 1\}$.

Proof. We do an Induction on $\#_M g$. States from elements in M are already in $\mathcal{N}(\mathfrak{N})$ hence the proposition is clear for all g with $\#_M g \leq 1$. If $\#_M g = s \geq 2$ then by Lemma 4.26 the inequality $\#_M(g@x) < \#_M(g)$ holds and so with the induction hypothesis and the recursive definition of $d_{\mathcal{N}}$ we obtain:

$$d_{\mathcal{N}}(g) = d_{\mathcal{N}}(g@x) + 1 \leq \left\lceil \log_2 \left\lceil \frac{s}{2} \right\rceil \right\rceil + 2 \quad \text{for a } x \in \mathcal{A} \; .$$

With the identity $\lceil \log_2(s+1) \rceil = \lceil \log_2(s) \rceil$ for odd s then the claim follows. □

Proposition 4.28. *The group $\mathfrak{N}$ has commutator width* 1.

Proof. The proof is by induction on $\#_M g$. If $\#_M g = 0$ then we have $g \in \mathcal{N}(\mathfrak{N})$ and thus it is a commutator because every element of A_n has commutator width 1. For $g \in \mathfrak{N}$ we consider the equation $[X,Y]g$ and take σ as in Proposition 4.20. If $\mathcal{E}_i := \Phi^{\mathrm{nf}}_{\sigma}([X,Y]g)@i$ has genus 1 then we have $\mathcal{E}_i = [X_i, Y_i]g@i$. By Lemma 4.26 we have $\#_M(g@i) < \#_M(g)$ so this equation $\mathcal{E}_i$ has a solution φ_i by induction.
If the genus of $\mathcal{E}_i$ is greater than 1 then $\mathcal{E}_i = [X_1, Y_1] \cdots [X_r, Y_r](g@\ell_1)^{Z_1} \cdots (g@\ell_s)^{Z_s}$ with ℓ_j non trivial and pairwise distinct and $s \leq n$. If $s = 1$ this equation $\mathcal{E}_i$ has again a solution φ_i by induction. If $s > 1$ there is a solution φ_i' for the equation $[X_r, Y_r]g@\ell_1$ and a solution φ_i'' for $[X_1, Y_1] \prod_{j=2}^{s} X@\ell_j$ because again by Lemma 4.26 we have $\#_M(g@\ell_1), \#_M(\prod_{j=2}^{s} g@\ell_j) < \#_M(g)$. The solutions φ_i' and φ_i'' can be chosen with disjoint support and thus there is a solution φ_i solving both equations. Now all φ_i for nontrivial $\mathcal{E}_i$ can be chosen with minimal and thus disjoint support. Thus the join $\circ_{i=1}^{n} \varphi_i$ is a solution for all $\mathcal{E}_i$. □

This proposition can be extended to similar groups. The same proof gives the following proposition:

Proposition 4.29. *Let G be a contracting layered group with $G \simeq G \wr A_n$ for $n \geq 5$, where there is a sub-multiplicative function $d \colon G \to \mathbb{N}_0$ such that $d(\prod_{i=1}^{n} g@i^{\sigma}) \leq d(g)$ for all $\sigma \in \mathrm{S}_n$ and $d(g@k) < d(g)$ for all $g \notin \mathcal{N}(G)$.*
If additionally each element of the nucleus is a commutator, then G has commutator width 1 .

4.2. Conjugacy problem. In this section, we show that the conjugacy problem of the groups $\mathfrak{N}_n$ is solvable. For this purpose we repeat some more general results.

Lemma 4.30 ([**BBSZ13**]). *Let $G < \mathrm{Aut}(\mathcal{A}^*)$ be a self-similar group and $g, h \in G$ be conjugate by a conjugator $f \in G$ i.e. $g^f = h$. Then we have for every $x \in \mathcal{A}$ with $m_x = |\{x^{g^i} \mid i \geq 1\}|$ that $g^{m_x}@x$ is conjugate to $h^{m_x}@x^f$ with conjugator $f@x$.*

Proof. We fix $x \in \mathcal{A}$ and first note that if g and h are conjugate then so are their activities and thus the size of the forward orbit of x under $\mathrm{act}(g)$ and $\mathrm{act}(h)$ are identical and equal $m := m_x$. If f is a conjugator we furthermore have $h@x^{fh^i} = (g^f)@x^{fh^i} = f^{-1}@x^{g^i f} \cdot g@x^{g^i} \cdot f@x^{g^{i+1}}$. Then the following chain of equations proves the claim:

$$h^m@x^f = \prod_{i=0}^{m-1} h@x^{fh^i} = f^{-1}@x^f \cdot \prod_{i=0}^{m-1} g@x^{g^i} \cdot f@x^{g^m} = (f@x)^{-1} \cdot g^m@x^f \cdot f@x.$$

□

Lemma 4.31 ([**BBSZ13**]). *Let $G \simeq G \wr P < \mathrm{Aut}(\mathcal{A}^*)$ be a layered self-similar group and $g, h \in G$ and $\sigma \in P$ be such that $\mathrm{act}(g)^{\sigma} = \mathrm{act}(h)$ and for all $x \in \mathcal{A}$ with $m_x = |\{x^{g^i} \mid i \geq 1\}|$ we have that $g^{m_x}@x$ and $h^{m_x}@x^{\sigma}$ are conjugate in G.*
Then g and h are also conjugate in G.

Proof. Assume the activities of g and h are conjugate by a conjugator σ and assume $g^{m_x}@x \sim h^{m_x}@x^\sigma$ by a conjugator $\tilde{f}_x$ for every x. Let us further denote by α the activity of g and decompose $\mathcal{A}$ into the orbits under α and choose a transversal T of these orbits. We define for $t \in T$ and $0 \leq i < m_t$: $f_{t^{\alpha^i}} := (g^i@t)^{-1}\tilde{f}_t h^i@t^\sigma)$ and $f = \langle\!\langle f_x \mid x \in \mathcal{A} \rangle\!\rangle \sigma$. Then for every $t \in T$ and $0 \leq i < m_t$ we have:

$$\begin{aligned} g^f@t^{\alpha^i\sigma} &= (f@t^{\alpha^i})^{-1} \cdot g@t^{\alpha^i} \cdot f@t^{\alpha^{i+1}} \\ &= (h^i@t^\sigma)^{-1}(f@t)^{-1}g^i@t \cdot g@t^{\alpha^i} \cdot (g^{i+1}@t)^{-1}f@t \cdot h^{i+1}@t^\sigma \\ &= h^{-i}@t^{\alpha^i\sigma}h^{i+1}@t^\sigma = h@t^{\alpha^i\sigma}. \end{aligned}$$

Thus $g^f@x = h@x$ for all $x \in \mathcal{A}$ and hence $g^f = h$. □

In the proof we see in the level of states we only need for every forward orbit of g a representative x such that $g^{m_x}@x \sim h^{m_x}@x^\sigma$ and thus we have the following slightly stronger corollary:

Corollary 4.32. *Let $G \simeq G \wr P < \mathrm{Aut}(\mathcal{A}^*)$ be a layered self-similar group and $g, h \in G$ and $\sigma \in P$ be such that $\mathrm{act}(g)^\sigma = \mathrm{act}(h)$. Let T be a transversal of the forward orbits $\mathcal{A}/\langle g \rangle$. If for every $x \in T$ with $m_x = |\{x^{g^i} \mid i \geq 1\}|$ we have that $g^{m_x}@x$ and $h^{m_x}@x^\sigma$ are conjugate in G then g and h are also conjugate in G.*

Remark 4.33. If we formulate the question whether two elements g, h are conjugate into the introduced language of equations and consider the equation $\mathcal{E} = g^X h$ then the independent system $\Phi^\varphi_\sigma()$ consists of the equation $X^{-1}g^{m_x}@xX$.

In [**BBSZ13**] the authors show among other results that the conjugacy problem is decidable for the group $\mathrm{Poly}(0)$. For that purpose they construct a finite graph. We use a similar approach to prove the following:

Proposition 4.34. *The conjugacy problem in $\mathfrak{N}_n$ is solvable for every $n \geq 5$.*

Proof. As before we fix some $n \in \mathbb{N}$ and denote $\mathfrak{N} = \mathfrak{N}_n$, $N = N_n$ and $M = M_n$. For two elements $g, h \in \mathfrak{N}$ we do an induction on $\#_M(g,h) := \#_M(g) + \#_M(h)$. First let $g, h \in \mathcal{N}(G) = N \cup M$. If not both of g, h are in N or not both are in M then they are obviously not conjugate to each other. If both are in N they are conjugate to each other if and only if their activities are conjugate to each other in A_n.
If both g and h are in M say $g = \alpha_\tau$, $h = \alpha_\pi$ then they have trivial activity and so by Lemma 4.30 they can only by conjugate if their states are. This can only be the case if $a_\tau \sim a_\pi$ and thus if and only if $\tau \sim \pi$.
We assume now that $\#_M(g,h) = C > 2$ and that there is an algorithm to decide whether $g' \sim h'$ for all g' and h' with smaller value of $\#_M(g', h')$. We define a graph with the finite set $\mathcal{P} = \{(g', h') \in \mathfrak{N}^2 \mid \#_M(g', h') \leq C\}$ as vertices.
For all the pairs (g, h) with $\#_M(g,h) < C$ we can mark the corresponding vertex by induction with the color *green* if $g \sim h$ and with the color *red* if $g \not\sim h$.
For fixed $(g,h) \in \mathcal{P}$ with $\#_M(g,h) = C$ we write the activity of g in terms of cycles i.e. as $(1, \ldots, \ell_1)(\ell_1 + 1, \ldots, \ell_2) \ldots (\ldots \ell_k)$ and choose as transversal the set $T = \{\ell_i \mid i = 1, \ldots, k\}$. The size of the orbit containing ℓ_i is then $m_i := m_{\ell_i} = \ell_i - \ell_{i-1}$ (with $\ell_0 = 0$). We have always $\#_M(h^{m_i}@\ell_i) \leq \#_M(h)$ and $\#_M(g^{m_i}@\ell_i) \leq \#_M(g)$. For every σ that is a conjugator for the activities of g and h and every $\ell_i \in T$ we draw an edge from (g,h) to $(g^{m_i}@\ell_i, h^{m_i}@\ell_i^\sigma)$ with label $\ell_i|\ell_i^\sigma$. Note that because of Lemma 4.26 for every σ there is at most one edges that goes to a not already colored vertex.

We will repeat this process for every other pair $(g, h) \in \mathcal{P}$ with $\#_M(g, h) = C$. The general situation is visualized as an example in Figure 9.

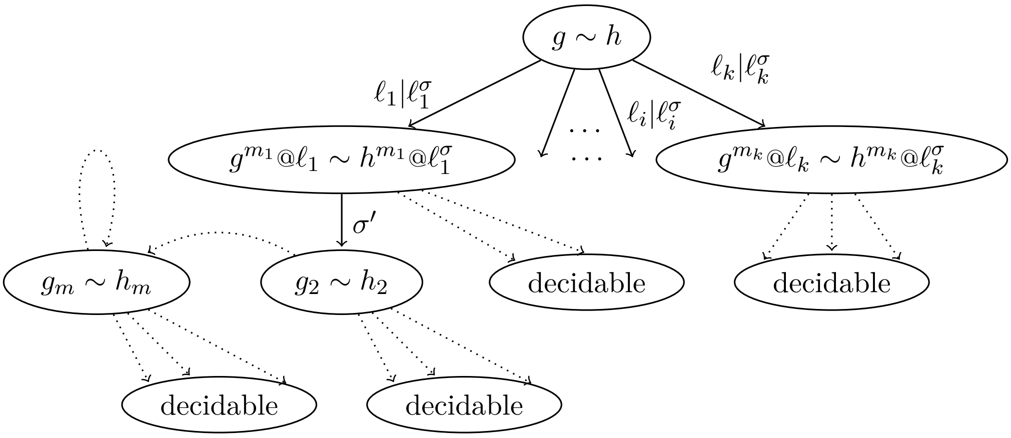


FIGURE 9. The decision graph for the problem $g \sim h$.

We now again fix an uncolored vertex (g, h) and consider all σ that conjugate the activities of g and h. If for such σ there is an outgoing edge labeled by σ leading to a *red* vertex then we remove all edges from this vertex labeled with σ. If there are no outgoing edges left we mark (g, h) as *red*. If otherwise for one such σ all edges labeled with σ lead to green vertices then we mark (g, h) green. Corollary 4.32 and Lemma 4.30 imply that *green* vertices are indeed conjugacy pairs and *red* ones are not conjugate.

If there are uncolored vertices left then there is a circle in the graph. If (g, h) is a vertex on that circle and $f \in \operatorname{Poly}(0)$ a conjugator for g and h then f is itself circular i.e. it exists $v \in \mathcal{A}^*$ such that $f@w = f$. Thus $f \in \mathfrak{N}$ only if $f \in \mathcal{N}(\mathfrak{N})$. Because the nucleus is finite we can simply test if this is the case and color the vertex accordingly. After coloring all states on circles in the graph we can again use the previous step to color all remaining vertices. □

4.3. Product of Conjugates. We will show the decidability of oriented equations in $\mathfrak{N}_n$ of genus 0. As an in-between step we will prove that every element of $\mathfrak{N}_n$ is a product of boundedly many conjugates of every fixed element of N_n.

PROPOSITION 4.35. *There is a constant $C_n \in \mathbb{N}$ depending only on n such that for every $\sigma \in N_n$, $\sigma \neq \mathbb{1}$ and $g \in \mathfrak{N}_n$ the equation $\prod_{i=1}^{C_n} \sigma_i^Z g$ has a solution. I.e every element $g \in \mathfrak{N}_n$ is a product of at most C_n conjugates of any fixed nontrivial element of N_n.*

PROOF FOR THE CASE $n = 5$. In the group A_5 we can verify with a simple calculation:

Claim 1: For every two elements $\sigma, \tau \in \mathrm{A}_5$ with $\sigma \neq \mathbb{1}$ there are elements $x, y, z \in \mathrm{A}_5$ such that $\sigma^x \sigma^y \sigma^z = \tau$.

Moreover we can again by direct calculations verify:

Claim 2: For every $\sigma \in N$, $\sigma \neq \mathbb{1}$ and equation $\sigma^X \sigma^Y \sigma^Z g$ there is a constraint $\gamma\colon \langle X, Y, Z\rangle \to \mathrm{A}_5$ such that $\Phi_\gamma^{\mathrm{nf}}(\sigma^X \sigma^Y \sigma^Z g)$ consists of exactly one nontrivial equation and that is of positive genus.

Let $\sigma \in N$ be nontrivial and γ a constraint like in the second claim. Then, because every element of $\mathfrak{N}_5$ is a commutator, the system $\Phi_\gamma(\sigma^X \sigma^Y \sigma^Z g) = \langle\!\langle \mathcal{E}_1, \dots, \mathcal{E}_5 \rangle\!\rangle \tau$ is solvable and hence there is a solution for the equation $\sigma^X\sigma^Y\sigma^Z\tau^{-1}g$. Thus claim 1 gives us a solution for the equation $\mathcal{E} = \prod_{i=1}^6 \sigma^{Z_i}g$. □

For the proof of the general case we will need some properties of the alternating group.

LEMMA 4.36. *For every $\sigma \in \mathrm{A}_n$ the identity can be written as a product of either* 2 *or of* 4 *conjugates of σ.*

PROOF. It is clear that it is sufficient to prove the result only for one representative of each conjugacy class of σ.
Let $\sigma = (1, 2, \dots, m)$ by a cycle of odd length m. There is always an element $\pi \in \mathrm{S}_m$ such that $\sigma^\pi = \sigma^{-1}$. E.g. we can take $\pi = (2, m)(3, m-1)\cdots((m+1)/2, (m+3)/2)$. If $m \equiv 1 \pmod 4$ then $\pi \in \mathrm{A}_m$. Moreover for

$$\pi' = (2, m)(3, m-1)\cdots\left(\frac{m-1}{2}, \frac{m+5}{2}\right)\left(1, \frac{m+1}{2}, \frac{m+3}{2}\right)$$

we have $\sigma\sigma^{\pi'} = (1, \frac{m+1}{2})(\frac{n-1}{2}, n)$ and hence $\sigma\sigma^{\pi'}\sigma\sigma^{\pi'} = \mathbb{1}$. If $m \equiv 3 \pmod 4$ then $\pi \in \mathrm{A}_m$.
For the case that m is even we can similarly set $\pi = (2, m)(3, m-1)\cdots(m/2, m/2+2)$ to obtain $\sigma\sigma^\pi = \mathbb{1}$ and $\pi' = (2, m)(3, m-1)\cdots(m/2-1, m/2+3)\,(1, m/2, m/2+2)$ to get $\sigma\sigma^{\pi'} = (1, m/2)(m/2-1, m)$ with $\pi \in \mathrm{A}_m$ if $m \equiv 2 \pmod 4$ and $\pi' \in \mathrm{A}_m$ if $m \equiv 0 \pmod 4$.
This gives the following result: Let $\sigma = \prod_{i=1}^{s+r} \sigma_i$ be the decomposition of σ into disjoint cycles with σ_i of odd length ℓ_i for $i = 1, \dots, s$ and of even length ℓ_i for $i = s+1, \dots, s+r$. If either $|\operatorname{supp}\sigma| \leq n-2$ or the sum

$$\ell = \sum_{i=1}^{s} \frac{\ell_i - 1}{2} + \sum_{i=s+1}^{s+r} \frac{\ell_i^e}{2}$$

is even then there is $\pi \in \mathrm{A}_n$ such that $\sigma\sigma^\pi = 1$. Otherwise there is $\pi' \in \mathrm{A}_n$ such that $\sigma\sigma^\pi\sigma\sigma^\pi = 1$. □

LEMMA 4.37. *For every $\sigma \in \mathrm{A}_n$ the identity is a product of* 3 *conjugates of σ.*

PROOF. If σ is a simple cycle of odd length, say $\sigma = (1, 2, \dots, n)$ then we take:

$$x = \begin{cases} (1, 2, n)(3, 4, n-2)\cdots\left(\frac{n-1}{2}, \frac{n+1}{2}, \frac{n+3}{2}\right) & \text{if } n \equiv 3 \pmod 4 \\ (1, 2, n)(3, 4, n-2)\cdots\left(\frac{n-3}{2}, \frac{n-1}{2}, \frac{n+5}{2}\right) & \text{if } n \equiv 1 \pmod 4, \end{cases}$$

and obtain $\sigma \cdot \sigma^x \cdot \sigma^{x^{-1}} = \mathbb{1}$. For a product of two cycles of even length say $\sigma = (1, \dots, 2m)(2m+1, \dots, 2n)$ we take if m and n are odd:

$$\begin{aligned} x =& (1, 2m, 2m+1)(3, 2nm-2, 2m-1)\cdots(m, m+1, m+2)\cdot \\ & (2m+2, 2m+3, 2n)(2m+4, 2m+5, 2n-2)\cdots(n+m, n+m+1, n+m+2). \end{aligned}$$

If m is odd and and n is even:

$$\begin{aligned} x =& (1, 2m, 2m+1)(3, 2nm-2, 2m-1)\cdots(m, m+1, m+2)\cdot \\ & (2m+2, 2m+3, 2n)(2m+4, 2m+5, 2n-2)\cdots(n+m-1, n+m, n+m+3). \end{aligned}$$

If m is even and and n is odd:

$$\begin{aligned} x =& (1, 2m, 2m+1)(3, 2nm-2, 2m-1)\cdots(m-1, m, m+3)\cdot \\ & (2m+2, 2m+3, 2n)(2m+4, 2m+5, 2n-2)\cdots(n+m, n+m+1, n+m+2). \end{aligned}$$

And if m and n are both even:

$$\begin{aligned} x =&(1, 2m, 2m+1)(3, 2nm-2, 2m-1)\cdots(m-1, m, m+3)\cdot \\ &(2m+2, 2m+3, 2n)(2m+4, 2m+5, 2n-2)\cdots(n+m-1, n+m, n+m+3). \end{aligned}$$

With this choice we get $\sigma \cdot \sigma^x \cdot \sigma^{x^{-1}} = \mathbb{1}$. □

Proof of Proposition 4.35 (general case). We will similar to the $n = 5$ case prove two claims, that together imply the result.

Claim 1: There is a constant $C \in \mathbb{N}$ such that for every $\sigma \in \mathrm{A}_n \setminus \{\mathbb{1}\}$ and $\tau \in \mathrm{A}_n$ there are elements $x_i \in \mathrm{A}_n$ such that $\prod_{i=1}^{C} \sigma^{x_i} = \tau$.

First note that the group generated by $\{\sigma^x \mid x \in \mathrm{A}_n\}$ is normal in A_n and because A_n is simple the set generates the whole group. Hence every element is a product of boundedly many conjugates of σ. For every $\tau \in \mathrm{A}_n$ denote by $C_{\sigma,\tau} \leq \operatorname{diam}(\mathrm{A}_n)$ the integer such that τ is a product of $C_{\sigma,\tau}$ conjugates of σ. We need that there is a constant C such that every τ is a product of exactly C conjugates of σ.

If all the values of $C_{\sigma,\tau}$ for different τ coincide, then nothing else is to do. According to Lemma 4.36 and Lemma 4.37 also every τ is either a product of $C_{\sigma,\tau} + 2\ell + 3k$ or $C_{\sigma,\tau} + 3\ell + 4k$ conjugates of σ for every $\ell, k \in \mathbb{N}$. Because $\gcd(2,3) = \gcd(3,4) = 1$ in both cases we can find a common bound $C \in \mathbb{N}$ such that every τ is a product of exactly C conjugates of σ.

Claim 2: For every $\sigma \in N_n$, $\sigma \neq \mathbb{1}$ and an equation $\mathcal{E} = \prod_{i=1}^{3r} \sigma^{X_i} g$ with $r = \lceil \frac{n}{5} \rceil$ there is a constraint $\gamma\colon \langle X_1, \dots, X_{3r} \rangle \to \mathrm{A}_n$ such that $\Phi_\gamma^{\mathrm{nf}}(\mathcal{E})$ consists only of equations of positive genus and trivial equations.

It is obviously sufficient to prove the second claim only for the conjugacy classes of σ. We will prove it first for those σ that are longest cycles in A_n. For the longest cycles there are two different conjugacy classes. We have thus a few cases to consider: If n is odd there could be $\sigma = (1, \dots, n)$ and $\sigma = (1, \dots, n-2, n, n-1)$ and if n is even there can be $\sigma = (1, \dots, n-1)$ and $\sigma = (1, \dots, n-2, n)$. For the first case $\sigma = (1, 2, \dots, n)$ we choose $\gamma\colon X_i \mapsto ()$ for $i = 1, 2, 3$ and obtain:

$$\Phi_\gamma\left(\prod_{i=1}^{3} \sigma^{X_i} g\right) = \langle\!\langle X_{1,i}^{-1} X_{1,i+1} X_{2,i+1}^{-1} X_{2,i+2} X_{3,i+2}^{-1} X_{3,i+3} g@i+3 \mid \begin{matrix} i \equiv 1, \dots, n \\ \bmod n \end{matrix} \rangle\!\rangle_{\tau_{g,\sigma}}.$$

Then the $\mathfrak{N}_n$-homomorphism

$$\varphi\colon X_{1,i} \mapsto X_{1,i+1} X_{2,i+1}^{-1} X_{2,i+2} X_{3,i+2}^{-1} X_{3,i+3} g@i+3 \text{ for } i \neq 1, i \pmod{n}$$

trivializes all but one equation: $\Phi_\gamma^\varphi(\sigma^{X_1}\sigma^{X_2}\sigma^{X_3} g)@1 = \langle\!\langle \mathcal{E}', \mathbb{1}, \dots, \mathbb{1} \rangle\!\rangle$ which is of positive genus:

$$\begin{tikzcd}
\mathcal{E}' = X_{2,1}^{-1} X_{2,2} X_{3,2}^{-1} X_{3,3} (g@3) X_{2,n}^{-1} X_{2,1} X_{3,1}^{-1} X_{3,2} (g@2) \cdots X_{2,2}^{-1} X_{2,3} X_{3,3}^{-1} X_{3,4} (g@4) \arrow[d, Rightarrow, "X_{2,2} \mapsto X_{2,2} X_{2,n} (g@3)^{-1} X_{3,3}^{-1} X_{3,2}"] \\
X_{2,1}^{-1} X_{2,2} X_{2,1} X_{3,1}^{-1} X_{3,2} (g@2) \cdots X_{3,2}^{-1} X_{3,3} (g@3) X_{2,n}^{-1} X_{2,2}^{-1} X_{2,3} X_{3,3}^{-1} X_{3,4} (g@4) \arrow[d, Rightarrow, "X_{2,1} \mapsto X_{2,1} X_{2,n} (g@3)^{-1} X_{3,3}^{-1} \cdots (g@2)^{-1} X_{3,2}^{-1} X_{3,1}"] \\
X_{3,1}^{-1} X_{3,2} (g@2) \cdots X_{3,2}^{-1} X_{3,3} (g@3) X_{2,n}^{-1} [X_{2,1}, X_{2,2}^{-1}] X_{2,3} X_{3,3}^{-1} X_{3,4} (g@4).
\end{tikzcd}$$

n	genus($\mathcal{E}'$)	n	genus($\mathcal{E}'$)
5	4	13	12
6	4	14	13
7	6	15	13
8	7	16	15
9	7	17	16
10	9	18	16
11	10	19	18
12	10	20	19

TABLE 1. Genus of the derived equations for small values of n.

The exact value of the genus depends on n; in Table 1 we give the genera for small values of n. For the other cases very similar calculations yield the same result. So in the cases of this choices of σ the value $r = 1$ already suffices. For the equation $\mathcal{E}$ we can iterate the choice of γ for the next variables and thus choose $\gamma\colon X_i \mapsto ()$ for all i. Then the resulting equation is $\lceil n/5 \rceil$ times the genus of the the equation $\mathcal{E}'$.
If σ is a cycle of shorter length i.e. $\sigma \in \mathrm{A}_m$ for $m < n$ then the same choice of γ as before results in

$$\Phi_\gamma\left(\prod_{i=1}^{3} \sigma^{X_i} g\right) = \langle\!\langle \mathcal{E}', \mathbb{1}, \dots, \mathbb{1}, g@m+1, g@m+2, \dots, g@n \rangle\!\rangle \tau_{g,\sigma}.$$

There is $\pi \in \mathrm{Alt}(n)$ such that

$$\Phi_{\gamma\circ\pi}\left(\prod_{i=1}^{3} \sigma^{X_i} g\right) = \langle\!\langle g@1, \dots, g@m, \mathcal{E}', \mathbb{1}, \dots, \mathbb{1}, g@2m+1, g@2m+2, \dots, g@n \rangle\!\rangle \tau_{g,\sigma}.$$

The product of at most $\lceil \frac{n}{m} \rceil$ such shifted version has the requested property. Because of the earlier proof for the case $n = 5$ this provides the result.
If $\sigma = \sigma_1 \cdots \sigma_\ell$ is the product of disjoint cycles we have seen before that the support of the constraints γ_i that give the previous result for σ_i are pairwise disjoint. Hence for $\gamma = \circ_{i=1}^{\ell} \gamma_i$ we obtain $\Phi_\gamma(\mathcal{E}) = \langle\!\langle \mathcal{E}_1', \mathbb{1}, \dots, \mathbb{1}, \mathcal{E}_2', \mathbb{1}, \dots, \mathbb{1}, \mathcal{E}_\ell', \mathbb{1}, \dots \rangle\!\rangle \tau_{g,\sigma}$. Where each equation $\mathcal{E}_i'$ for $i = 1 \dots, \ell$ is of positive genus. □

The normal form of an equation of genus 0 in $F_\mathbb{N} * \mathfrak{N}_n$ is $\mathcal{E} = g_1^{Z_1} \cdots g_r^{Z_r} g_{r+1}$. We generalized the previously defined M-length of an element g to such kind of equations. The generalized M-length for an equation $\mathcal{E} = g_1^{Z_1} \cdots g_r^{Z_r} g_{r+1}$ is defined by

$$\#_M(\mathcal{E}) = \sum_{i=1}^{r+1} \#_M(g_i).$$

We start with an observation:

LEMMA 4.38. *If $\mathcal{E} \in F_\mathbb{N} * \mathfrak{N}$ is an oriented equation of genus 0 with $\#_M(\mathcal{E}) = 0$ then $\mathcal{E}$ is solvable if and only if $(\mathbb{1} * \mathrm{act})(\mathcal{E})$ is solvable in $F_\mathbb{N} * \mathrm{A}_n$.*

PROOF. We only need to prove the "if" part. Let $\sigma\colon F_\mathbb{N} \to \mathrm{A}_n$ be a solution for $(\mathbb{1} * \mathrm{act})(\mathcal{E})$. If $\#_M\mathcal{E} = 0$ then $\mathcal{E} = \prod_{i=1}^{s} n_i^{Z_i} n_{s+1}$ with $n_i \in N$ and thus all states of $\Phi_\sigma(\mathcal{E})$ are in $F_\mathbb{N}$ and hence solvable by the trivial homomorphism. Therefore the map $Z_i \mapsto \langle\!\langle 1, \dots, 1 \rangle\!\rangle \sigma(Z_i)$ is a solution for $\mathcal{E}$. □

PROPOSITION 4.39. *The question whether an oriented equation $\mathcal{E} \in F_\mathbb{N} * \mathfrak{N}$ of genus 0 has a solution is decidable.*

Proof. The proof is again an induction on $\#_M\mathcal{E}$. The case $\#_M\mathcal{E}=0$ is already done in Lemma 4.38. We assume now that $\#_M(\mathcal{E})>0$. Similar to the proof of Proposition 4.34 we note that if $\Phi^{\mathrm{nf}}_\gamma(\mathcal{E})=\langle\!\langle\mathcal{E}_1,\dots,\mathcal{E}_n\rangle\!\rangle$ then $\#_M(\mathcal{E}_i)\leq\#_M(\mathcal{E})$ for every choice of an enriched constraint γ. Whenever the $\#_M$ value gets strictly smaller we can decide by induction if the equation is solvable. Also if $\#_M(\mathcal{E}_i)=\#_M(\mathcal{E})$ then $\#_M(\mathcal{E}_j)=0$ for all $j\neq i$. Thus in the worst case we get a sequence of equations $\mathcal{E}^{(i)}$ all with $\#_M\mathcal{E}^{(i)}=\#_M(\mathcal{E})$. In the proof of Proposition 4.34 we could use that the set of those derived equations $\mathcal{E}^{(i)}$ was finite. If this is the case we can proceed similarly. But it can be that this sequence is infinite.
In the case that $\mathcal{E}=\prod_{i=1}^{s}n_i^{Z_i}n_{s+1}$ with $s>1$ it can occur that $\mathcal{E}_i$ has positive genus. In that case we already know that then the equation $\mathcal{E}_i$ is solvable. But it can also occur that $\mathcal{E}_i$ has signature $(0,s')$ with $s'>s$. Hence the set of equations $\{\mathcal{E}^{(i)}\mid i\in\mathbb{N}\}$ is possibly infinite. Let us assume we have such an infinite sequence of equations $\mathcal{E}^{(i)}$ of signature $(0,s_i)$.
In particular there is an $i\in\mathbb{N}$ such that $s_i>C_n\cdot|N_n|+\#_M(\mathcal{E})$, with C_n the constant from Proposition 4.35, and because we have $\#_M(\mathcal{E}^{(i)})\leq\#_M(\mathcal{E})$ the equation $\mathcal{E}^{(i)}$ is thus equivalent to

$$\prod_{j=1}^{s_i-C_n-1}g_j^{Z_j}\prod_{j=1}^{C_n}\sigma^{Z'_j}g_{s_i-C_n}\qquad\text{for some } g_k\in\mathfrak{N}_n \text{ and } \sigma\in N_n.$$

According to Proposition 4.35 this is always solvable. □

Corollary 4.40. *All quadratic oriented equations over G are decidable.*

Proof. If an equation has genus greater 0 then by Proposition 4.28 it is solvable. □

5. Grigorchuk group

For the Grigorchuk group $\mathfrak{G}$ it is already shown by I. Lysenok, A. Miasnikov and A. Ushakov in [**LMU16**] that there exists an algorithm to decide for an arbitrary quadratic equations $\mathcal{E}\in F_{\mathbb{N}}*\mathfrak{G}$ if it is solvable. The algorithm however is not made explicit it depends on initial choices on a set far too large to compute. In the same paper it is shown that the commutator width of the group $\mathfrak{G}$ is finite but no bound could be given with that method.
In joint work with Laurent Bartholdi and Igor Lysenok we proved that the commutator width of the Grigorchuk group is two in the paper [**BGL17**] that is also in the Appendix A. Furthermore in that paper we prove that every element in the derived subgroup is a product of at most 6 conjugates of the generator a.
Here we mention some ideas that are not contained in the paper. We remark that for example the unoriented equation $\mathcal{E}=X_1^2X_2^2X_3^2X_4^2X_5^2g$ is equivalent to the equation $\mathcal{E}'=X_1^2[X_2,X_3][X_4,X_5]g$ under the $\mathfrak{G}$-isomorphism

$$\begin{aligned}
X_1&\mapsto X_1^2X_2^{-1}X_1^{-1}\\
X_2&\mapsto X_1X_2X_1^{-1}X_3^{-1}X_1^{-1}\\
X_3&\mapsto X_1X_3X_1X_3X_4^{-1}X_3^{-1}X_1^{-1}\\
X_4&\mapsto X_1X_3X_4X_3^{-1}X_1^{-1}X_5^{-1}X_3^{-1}X_1^{-1}\\
X_5&\mapsto X_1X_3X_5.
\end{aligned}$$

Hence if g is in the derived subgroup $\mathfrak{G}'$ then the equation $\mathcal{E}$ has always a solution and if $g \notin \mathfrak{G}'$ then it has no solution because every generator of $\mathfrak{G}$ is of order two and thus the square of any element is in the derived subgroup.
The equations $\mathcal{E} = g_1^{X_1} g_2$ for $g_1, g_2 \in \mathfrak{G}$ are solvable if and only if g_1 is conjugate to g_2. This problem is already known to be decidable and there is an implementation of an algorithm in the GAP package *fr*.
Given an arbitrary equation $\mathcal{E}$ that is not already known to be solvable by the previously mentioned results we can try to solve the equations that occur if we apply the homomorphism Φ_γ for some enriched constraints γ. If we want to lift the solution back we need to consider constraints $\gamma\colon F_{\mathbb{N}} \to \mathfrak{G}/K$ like we did for the computation of the commutator width. But for an overview how the signature of the equation changes under Φ_γ it is sufficient to consider only the activity part γ_{act} of the enriched constraint. In figure 10 we give an overview how the signature (r, s) of an oriented equation $\mathcal{E} = \prod_{i=1}^{r}[X_{2i-1}, X_{2i}] \prod_{i=1}^{s-1} g_i^{X_{2r+i}} g_s$ behaves after applying the Φ_γ in correspondence of the activities $\operatorname{act}(g_i)$ and γ_{act}.

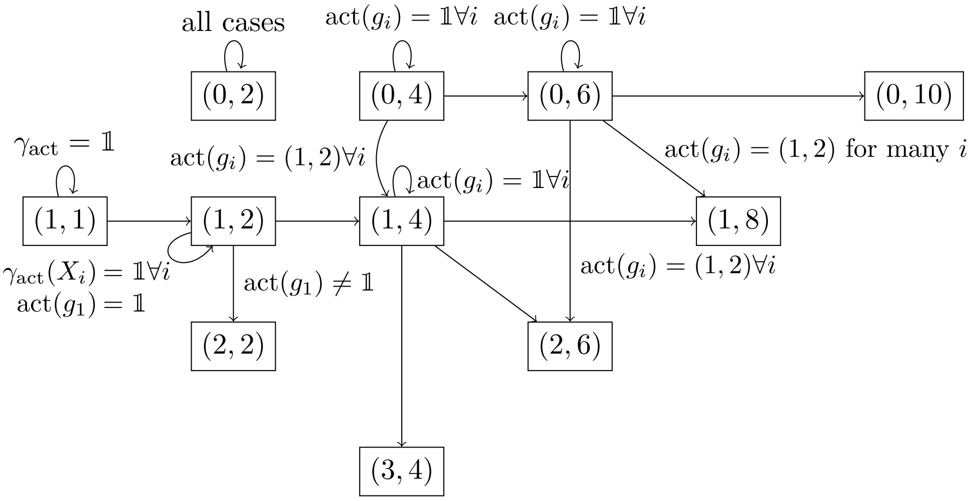

all cases
$\operatorname{act}(g_i) = \mathbb{1} \forall i$
$\operatorname{act}(g_i) = \mathbb{1} \forall i$
$(0,2)$
$(0,4)$
$(0,6)$
$(0,10)$
$\gamma_{\text{act}} = \mathbb{1}$
$\operatorname{act}(g_i) = (1,2) \forall i$
$\operatorname{act}(g_i) = \mathbb{1} \forall i$
$\operatorname{act}(g_i) = (1,2)$ for many i
$(1,1)$
$(1,2)$
$(1,4)$
$(1,8)$
$\gamma_{\text{act}}(X_i) = \mathbb{1} \forall i$
$\operatorname{act}(g_1) = \mathbb{1}$
$\operatorname{act}(g_1) \neq \mathbb{1}$
$\operatorname{act}(g_i) = (1,2) \forall i$
$(2,2)$
$(2,6)$
$(3,4)$

FIGURE 10. Diagram that describes how the signature of an oriented quadratic equation over a group $G \leq \operatorname{Aut}(\{1,2\}^*)$ changes under application of Φ_γ.

With the *equation* package this sort of computations are easy to make. In the file `examples/GrigorchukGenus.g` there are two methods defined that both require two arguments r and s. The function `PossibleDerivedSignaturesOriented` prints the occurring signatures of equations in $\Phi_\gamma(\prod_{i=1}^{r}[X_{2i-1}, X_{2i}] \prod_{i=1}^{s-1} g_i^{X_{2r+i}} g_s)$ depending on the different cases for the activities of g_i and of γ_{act}. For the unoriented equations the function `PossibleDerivedSignaturesUnoriented` prints the signatures of the equations in $\Phi_\gamma(\prod_{i=1}^{r} X_i^2 \prod_{i=1}^{s-1} g_i^{X_{r+i}} g_s)$ given additionally an indication if the derived equation is oriented or not.
The limited number of steps that are needed to obtain an equation of genus at least two indicates that an algorithm using this and constraints like in the proof of the commutator width will very likely be able to decide the solvability of an equation in the Grigorchuk group.

6. Gupta-Sidki group

In a very analogous way to the computation in the Grigorchuk group $\mathfrak{G}$ we give an upper bound for the commutator width of the Gupta-Sidki Group $\mathfrak{S}_3$.

LEMMA 4.41 ([**Sid87**]). *The group $\mathfrak{S}_p$ is regular branched with the derived subgroup ${\mathfrak{S}_p}'$ as branching subgroup.*

LEMMA 4.42 ([**Sid87**]). *Some normal subgroups of $\mathfrak{S}_3$ with their corresponding indices are:*

$$\begin{aligned}
\Gamma &= \overline{\langle t\rangle}^{\mathfrak{S}_3} = \mathrm{Stab}_1(\mathfrak{S}_3) = \langle\langle\!\langle t, t^a, t^{a^2}\rangle\!\rangle\rangle,\\
{\mathfrak{S}_3}' &= \langle [a,t], [a,t]^a, [a,t]^t, [a,t]^{at}\rangle,\\
\Gamma' &= {\mathfrak{S}_3}' \times {\mathfrak{S}_3}' \times {\mathfrak{S}_3}',\\
\gamma_3(\mathfrak{S}_3) &= [\mathfrak{S}_3, {\mathfrak{S}_3}'] = \mathrm{Stab}_2(\mathfrak{S}_3) = \Gamma'\langle\langle\!\langle t,t,t\rangle\!\rangle\rangle,\\
\mathfrak{S}_3/\Gamma &\simeq C_3,\\
Q := \mathfrak{S}_3/{\mathfrak{S}_3}' &\simeq C_3 \times C_3,\\
{\mathfrak{S}_3}'/\gamma_3(\mathfrak{S}_3) &\simeq C_3,\\
{\mathfrak{S}_3}'/\Gamma' &\simeq C_3 \wr C_3,\\
{\mathfrak{S}_3}'/{\mathfrak{S}_3}'' &\simeq C_3 \times C_3 \times C_3 \times C_3,
\end{aligned}$$

$$\begin{tikzcd}
\mathfrak{S}_3 \arrow[dr, dash, "3"] \arrow[dd, dash, "9"] & \\
& \Gamma = \mathrm{Stab}_1(\mathfrak{S}_3) \arrow[dl, dash, "3"] \\
{\mathfrak{S}_3}' \arrow[dr, dash, "3"] \arrow[dd, dash, "9"] & \\
& \gamma_3(\mathfrak{S}_3) = \mathrm{Stab}_2(\mathfrak{S}_3) \arrow[dl, dash, "3"] \\
\Gamma' \arrow[d, dash, "9"] & \\
{\mathfrak{S}_3}'' & .
\end{tikzcd}$$

We fix a finite subset $\{X_1, \dots, X_{2n}\}$ of generators of $F_{\mathbb{N}}$ say $\langle\{X_1, \dots, X_{2n}\}\rangle =: F_{\mathcal{X}}$ and denote by U_n the group of G-automorphisms $F_{\mathcal{X}} * G \to F_{\mathcal{X}} * G$ that is generated by the following automorphisms of $F_{\mathcal{X}}$:

$$\begin{aligned}
&\varphi_i\colon && X_i \mapsto X_{i-1}X_i, \text{ others fixed} && \text{for } i = 2, 4, \dots, 2n,\\
&\varphi_i\colon && X_i \mapsto X_{i+1}X_i, \text{ others fixed} && \text{for } i = 1, 3, \dots, 2n-1,\\
& && X_i \mapsto X_{i+1}X_{i+2}^{-1}X_i, && \\
&\psi_i\colon && X_{i+1} \mapsto X_{i+1}X_{i+2}^{-1}X_{i+1}X_{i+2}X_{i+1}^{-1}, && \text{for } i = 1, 3, \dots, 2n-3\\
& && X_{i+2} \mapsto X_{i+1}X_{i+2}^{-1}X_{i+2}X_{i+2}X_{i+1}^{-1}, && \\
& && X_{i+3} \mapsto X_{i+1}X_{i+2}^{-1}X_{i+3}, \text{ others fixed} && \\
& && X_i \mapsto X_{i+2}, && \\
&\xi_i\colon && X_{i+1} \mapsto X_{i+3}, && \text{for } i = 1, 3, \dots, 2n-3\\
& && X_{i+2} \mapsto [X_{i+3}, X_{i+2}]\ X_i\ [X_{i+2}, X_{i+3}], && \\
& && X_{i+3} \mapsto [X_{i+3}, X_{i+2}]\ X_{i+1}\ [X_{i+2}, X_{i+3}], && \\
& && \quad\text{others fixed.} &&
\end{aligned}$$

Note that U_n is a subgroup of $\mathrm{Stab}(R_n)$ in particular if we consider its elements as $\mathfrak{S}_3$-isomorphisms it fixes for all $g \in \mathfrak{S}$ the equation $\mathcal{E} = \prod_{i=1}^{n}[X_{2i-1}, X_{2i}]g$.

LEMMA 4.43. *If we identify the set of homomorphisms $\{\gamma\colon F_{\mathbb{N}} \to Q \mid \mathrm{supp}(\gamma) \subset \langle X_1, \dots, X_n\rangle\}$ with Q^n. Then*

$$\left|Q^{2n}/U_n\right| = 8 \textit{ for all } n \geq 2.$$

Furthermore for every such coset in Q^n/U_n there is a representative that has non-trivial entries only in the first 4 coordinates.

Proof. For $n = 2$ this is a direct computation with GAP with the method `OrbitDomain`. See the file `examples/GupstaSidkiCW.g` for details. If we write $\bar{t}$ and $\bar{a}$ for the images of a and b correspondingly under the natural homomorphism $\pi\colon \mathfrak{S}_3 \to Q$ then a system of representatives of Q^4/U_2 is:

$$\begin{aligned} \gamma_1 &= (\mathbb{1},\mathbb{1},\mathbb{1},\mathbb{1}), & \gamma_2 &= (\bar{t},\mathbb{1},\mathbb{1},\mathbb{1}),\\ \gamma_3 &= (\bar{a},\mathbb{1},\mathbb{1},\mathbb{1}), & \gamma_4 &= (\bar{t}\bar{a},\mathbb{1},\mathbb{1},\mathbb{1}),\\ \gamma_5 &= (\bar{t}^2\bar{a},\mathbb{1},\mathbb{1},\mathbb{1}), & \gamma_6 &= (\bar{t},\bar{a},\mathbb{1},\mathbb{1}),\\ \gamma_7 &= (\bar{t},\bar{a}^2,\mathbb{1},\mathbb{1}), & \gamma_8 &= (\bar{t},\mathbb{1},\bar{a},\mathbb{1}). \end{aligned}$$

For the general case we proceed similar to [**LMU16**, Lemma 6.4]. The group Q is polycyclic with polycyclic sequence $1 < \langle\bar{a}\rangle < \langle\bar{a},\bar{t}\rangle = Q$. The group Q is also abelian hence the induced action of U_n becomes:

$$\begin{aligned} \varphi_i(q_1,q_2,\dots,q_{2n}) &= \begin{cases} (q_1,\dots,q_{i-1},q_{i-1}q_i,q_{i+1},\dots,q_{2n}) & \text{if } i \text{ is even}\\ (q_1,\dots,q_{i-1},q_{i+1}q_i,q_{i+1},\dots,q_{2n}) & \text{if } i \text{ is odd}\end{cases}\\ \psi_i(q_1,q_2,\dots,q_{2n}) &= (q_1,\dots,q_{i-1},q_{i+1}q_{i+2}^{-1}q_i,q_{i+1},q_{i+2},q_{i+1}q_{i+2}^{-1}q_{i+3},q_{i+4},\dots,q_{2n})\\ \xi_i(q_1,q_2,\dots,q_{2n}) &= (q_1,\dots,q_{i-1},q_{i+2},q_{i+3},q_i,q_{i+1}). \end{aligned}$$

We partition the group Q into three sets $S_1 = \{\mathbb{1},\bar{a},\bar{a}^2\}$, $S_2 = \{\bar{t},\bar{t}\bar{a},\bar{t}\bar{a}^2\}$ and $S_3 = \{\bar{t}^2,\bar{t}^2\bar{a},\bar{t}^2\bar{a}^2\}$ and look at a block $(q_1,q_2) \in Q^2$. We want to transform this block by elements of U_1 to an element in $Q \times \langle\bar{a}\rangle$. If we have $q_2 \in S_1$ nothing is to do. If it is $q_2 \in S_2$ and $q_1 \in S_1$ then after applying φ_1 we obtain $(q_2q_1,q_2) \in S_2 \times S_2$. In that case we proceed with applying φ_2 two times to obtain $(q_2q_1,q_2q_1q_2q_1q_2) \in S_2 \times S_1$. If it is $q_2 \in S_1$ and $q_1 \in S_3$ we need only apply φ_2 once to obtain $(q_1,q_1q_2) \in S_3 \times S_1$. Similarly we proceed in the case that $q_2 \in S_3$. See Table 2 for the needed homomorphism in that case. Hence a constraint $\gamma \in Q^{2n}$ is equivalent to one of the form $(q_1,p_1,q_2,p_2,\dots,q_n,p_n)$ with $p_i \in \langle\bar{a}\rangle$.

$q_1\backslash q_2$	S_1	S_2	S_3
S_1	1	$\varphi_2\varphi_2\varphi_1$	$\varphi_2\varphi_2\varphi_1$
S_2	1	$\varphi_2\varphi_2$	φ_2
S_3	1	φ_2	$\varphi_2\varphi_2$

Table 2. The elements of U_1 that are needed to send an element (q_1,q_2) to one in $Q \times \langle a\rangle$.

Now we take a block $B = (q_1,p_1,q_2,p_2) \in Q \times \langle\bar{a}\rangle \times Q \times \langle\bar{a}\rangle$. This block is equivalent to one of the form $(q_2,\tilde{p}_2,\tilde{p}_1,p_1) \in Q \times S_1^3$: If $q_2 \in S_1$ nothing is to do. If $q_1 \in S_2$ then $\xi_1(B)$ has the required form. If both q_1 and q_2 are in the same set S_2 or S_3 then $\xi_1\varphi_4\psi_1(B) = (q_2,p_1p_2,p_1q_2^{-1}q_1) \in Q \times S_1^3$. If q_1 and q_2 are in different sets S_2 and S_3 then $\xi_1\varphi_4^2\psi_1^2(B) = (q_2,p_1^2p_2,p_1^2q_2^{-2}q_1,p_1) \in Q \times S_1^3$.
Hence γ is equivalent to $(q_1,p_2,\dots,p_{2n})$ with $p_i \in \langle\bar{a}\rangle$. Now we continue analogously with a next step. A block (p_1,p_2) is equivalent to one of the form $(\tilde{p}_1,\mathbb{1})$ and a block $(p_1,\mathbb{1},p_2,\mathbb{1})$ to one of the form $(\tilde{p}_1,\mathbb{1},\mathbb{1},\mathbb{1})$:

$$(\mathbb{1}, \bar{a}) \xmapsto{\varphi_1} (\bar{a}, \bar{a}) \xmapsto{\varphi_2} (\bar{a}, \bar{a}^2) \xmapsto{\varphi_2} (\bar{a}, \mathbb{1})$$

$$(\mathbb{1}, \bar{a}^2) \xmapsto{\varphi_1} (\bar{a}^2, \bar{a}^2) \xmapsto{\varphi_2} (\bar{a}^2, \bar{a}) \xmapsto{\varphi_2} (\bar{a}^2, \mathbb{1})$$

$$\begin{tikzcd}
(\bar{a}, \mathbb{1}, \bar{a}^2, \mathbb{1}) \arrow[r, mapsto, "\psi_1"] & (\bar{a}^2, \mathbb{1}, \bar{a}^2, \bar{a}) \arrow[d, mapsto, "\varphi_4"] & (\mathbb{1}, \mathbb{1}, \bar{a}^2, \mathbb{1}) \arrow[r, mapsto, "\xi_1"] & (\bar{a}^2, \mathbb{1}, \mathbb{1}, \mathbb{1}) \\
 & (\bar{a}^2, \mathbb{1}, \bar{a}^2), \mathbb{1} \arrow[r, mapsto, "\psi_1"] & (\mathbb{1}, \mathbb{1}, \bar{a}^2, \bar{a}) \arrow[u, mapsto, "\varphi_4"'] &
\end{tikzcd}$$

$$\begin{tikzcd}
(\bar{a}^2, \mathbb{1}, \bar{a}, \mathbb{1}) \arrow[r, mapsto, "\psi_1"] & (\bar{a}, \mathbb{1}, \bar{a}, \bar{a}^2) \arrow[d, mapsto, "\varphi_4"] & (\mathbb{1}, \mathbb{1}, \bar{a}, \mathbb{1}) \arrow[r, mapsto, "\xi_1"] & (\bar{a}, \mathbb{1}, \mathbb{1}, \mathbb{1}) \\
 & (\bar{a}, \mathbb{1}, \bar{a}), \mathbb{1} \arrow[r, mapsto, "\psi_1"] & (\mathbb{1}, \mathbb{1}, \bar{a}, \bar{a}^2) \arrow[u, mapsto, "\varphi_4"'] & .
\end{tikzcd}$$

Therefore γ is equivalent to an element $(q_1, p_1, p_2, \mathbb{1}, \dots, \mathbb{1}) \in Q \times \langle \bar{a} \rangle \times \langle \bar{a} \rangle \times 1^{2n-3}$ and with the computed result for the first four coordinates we obtain the result. □

We denote the set of the 8 constraints $\gamma_1, \dots, \gamma_n \colon F_{\mathbb{N}} \to Q$ that correspond to representatives of those 8 orbits by $\mathfrak{R}$.
We want to solve the equations $\mathcal{E} = [X_1, X_2][X_3, X_4]g$ for every $g \in \mathfrak{S}_3{}'$. Applying the branching homomorphism will give us equations of higher genus. We hence abbreviate the oriented quadratic equation $[X_1, X_2] \cdots [X_{2n-1}, X_{2n}]$ by $R_n(X_*)$.

Definition 4.44. Given $g \in \mathfrak{S}_3{}'$ and $\gamma \in \mathfrak{R}$, the tuple (g, γ) is called a *good pair* if $(R_n g, \gamma)$ is solvable for some $n \in \mathbb{N}$.

Definition 4.45. We denote by

$$\tau \colon \mathfrak{S}_3 \to \mathfrak{S}_3/\mathfrak{S}_3{}'' \quad \text{and} \quad \rho \colon \mathfrak{S}_3/\mathfrak{S}_3{}'' \to (\mathfrak{S}_3/\mathfrak{S}_3{}'')/(\mathfrak{S}_3{}'/\mathfrak{S}_3{}'') \simeq \mathfrak{S}_3/\mathfrak{S}_3{}'$$

the natural projections.

Lemma 4.46. *The pair (g, γ) is a good pair if and only if there is a solution $s \colon F_{\mathbb{N}} \to \mathfrak{S}_3/\mathfrak{S}_3{}''$ for $R_2\tau(g)$ with $s(X_i) \in \rho^{-1}(\gamma(X_i))$.*

Proof. If (g, γ) is a good pair and s a solution for $(R_n g, \gamma)$ then $s(X_i) \in \mathfrak{S}_3{}'$ for $i \geq 4$, so $s(R_n) = s(R_2) \cdot h'$ for some $h' \in \mathfrak{S}_3{}''$. Therefore there is a solution $\tau \circ s$ for $R_2\tau(g)$ with $s(X_i) = \gamma(X_i)$.
If there is a solution $s \colon F_{\mathbb{N}} \to \mathfrak{S}_3/\mathfrak{S}_3{}''$ for $R_2\tau(g)$ such that each $s(X_i)$ is contained in $\rho^{-1}(\gamma(X_i))$ then for $g_i \in \tau^{-1}(s(X_i))$ there is some $h' \in \mathfrak{S}_3{}''$ such that $R_2(g_1, \dots, g_4)h'g = \mathbb{1}$ and so (g, γ) is a good pair. □

Corollary/Definition 4.47. For $q \in \mathfrak{S}_3/\mathfrak{S}_3{}''$ and $\gamma \in \mathfrak{R}$ the pair (q, γ) is called a good pair if (g, γ) is a good pair for an arbitrary (and hence every) preimage of $g \in \tau^{-1}(q)$.

Lemma 4.48. *Let $x, y \in \mathfrak{S}_3$ be arbitrary elements and $\sigma \in \mathrm{S}_3$. Consider the map*

$$h_{x,y,\sigma} \colon \mathfrak{S}_3 \to \mathfrak{S}_3,\ g \mapsto g@\sigma(1)^x \cdot g@\sigma(2)^y \cdot g@\sigma(3).$$

If $g \in \mathfrak{S}_3{}''$ then $h_{x,y,\sigma}(g) \in \mathrm{Stab}(2)$ for all choices of x, y and σ. Furthermore the map

$$\bar{h}_{x,y,\sigma} \colon \mathfrak{S}_3{}'/\mathfrak{S}_3{}'' \to \mathfrak{S}_3{}'/\operatorname{Stab}(2),\ g\mathfrak{S}_3{}'' \mapsto (g@\sigma(1)^x \cdot g@\sigma(2)^y \cdot g@\sigma(3)) \operatorname{Stab}(2)$$

is well defined.

Proof. We can directly verify that all generators of $\mathfrak{S}_3''$ enjoy the claimed property. Because all elements of $\mathfrak{S}_3''$ have no activity the maps $h_{x,y,\sigma}$ are then homomorphisms. □

Lemma 4.49. *The map* $\mathrm{act}\colon \mathfrak{S}_3/\mathfrak{S}_3'' \to \mathrm{S}_3$, $g\mathfrak{S}_3'' \mapsto \mathrm{act}(g)$ *is well defined and for* $i = 1, 2, 3$ *the map* $@i\colon \mathfrak{S}_3'/\mathfrak{S}_3''$, $g\mathfrak{S}_3'' \mapsto (g@i)G'$ *is well defined.*

Proof. The first statement is clear because every element $g \in \mathfrak{S}_3''$ has trivial activity.
Also every element $g \in \mathfrak{S}_3'$ has trivial activity. Hence the map $@i\colon \mathfrak{S}_3' \to \mathfrak{S}_3$, $g \mapsto g@i$ is a homomorphism for every $i = 1, 2, 3$.
It is easy to check that for every generator $g \in \mathfrak{S}_3''$ we have that $g@i \in \mathfrak{S}_3'$ and thus we obtain the result. □

The activity of every element $g \in \mathfrak{S}_3'$ is trivial and hence every constraint $\gamma\colon F_{\mathbb{N}} \to Q$ admits a unique enriched constraint $\gamma = (\gamma_{\mathrm{con}}, \gamma_{\mathrm{act}})$.
We fix the set $\{\mathbb{1}, a, a^2, t, at, a^2t, t^2, at^2, a^2t^2\}$ as transversal for $\mathfrak{S}_3/\mathfrak{S}_3'$ and denote by rep: $\mathfrak{S}_3/\mathfrak{S}_3'$ the corresponding left inverse of π.
We fix an element $q \in \mathfrak{S}_3'/\mathfrak{S}_3''$ and an active constraint $\gamma \in \mathfrak{R}$ such that (q, γ) forms a good pair. We consider all $g \in \mathfrak{S}_3'$ such that $g^\tau = q$.
Let $\mathcal{E} = \prod_{i=1}^n [X_{2i-1}, X_{2i}]g$ be an equation and $\Phi_\gamma(\mathcal{E}) = \langle\!\langle \mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3 \rangle\!\rangle$ then the sets $\mathrm{Var}(\mathcal{E}_1) \cap \mathrm{Var}(\mathcal{E}_2)$ and $\mathrm{Var}(\mathcal{E}_2) \cap \mathrm{Var}(\mathcal{E}_3)$ are not empty because γ has nontrivial activity in some component. Hence $\Phi_\gamma^\varphi(\mathcal{E})$ contains exactly one nontrivial equation.
Let $\psi_{\gamma,n}$ be the normalization homomorphism that maps $\Phi_\gamma^\varphi(\mathcal{E})$ to its normal form.
Then a direct computation gives us that

$$\psi_{\gamma,n}(\mathcal{E}) = \Phi_\gamma^{\mathrm{nf}}(\mathcal{E}) = R_{3n-2}(X_*)(g@\mu(3))^{X_{6n-3}} \cdot (g@\mu(2))^{X_{6n-2}} \cdot g@\mu(1)$$

holds for some $\mu \in \mathrm{S}_3$. Note that according to Lemma 4.49 the value of $(g@i)^\pi$ depends only on q.
We need a small variation of the definition of $\Gamma_1^{\mathcal{E}}$ than we defined it in Section 2:

$$\Gamma_2^{q,n}(\gamma) = \{(\gamma' * \pi) \circ \psi_{\gamma,n}^{-1} \mid \gamma' \in \Gamma_1(\gamma), \gamma'(X_i) = \mathbb{1} \text{ for } i > 12, (\gamma' * act)(R_2 q) = \mathbb{1}\}.$$

To show that we can take $\Gamma_2^{q,n}(\gamma)$ independent of n we need to go back to the normalization process of Φ_γ:
Let $g = \langle\!\langle g_1, g_2, g_3 \rangle\!\rangle \in \mathfrak{S}_3'$ be an arbitrary element and $\gamma \in \mathfrak{R}$ be a reduced constraint, $n \geq 2$ and $\mathcal{E}^{(n)} = \prod_{i=1}^n [X_{2i-1}, X_{2i}]g$ be an equation. Let further be $\Phi_\gamma(\mathcal{E}^{(2)}) = \langle\!\langle \mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3 \rangle\!\rangle \langle\!\langle g_1, g_2, g_3 \rangle\!\rangle$ with decomposition $\mathcal{E}_1 = U_1 X^{-1} U_2$, $\mathcal{E}_2 = V_1 X V_2 Y V_3$ and $\mathcal{E}_3 = W_1 Y^{-1} W_2$ for some choice of variables X and Y. (This decomposition does always exist with with eventually changing the signs of X or Y).
Consider the following sequences of $\mathfrak{S}_3$-homomorphisms to obtain the form $\Phi_\gamma^{\mathrm{nf}}(\mathcal{E}^{(n)})$:

$$\begin{array}{l}
\Phi_\gamma(\mathcal{E}^{(n)}) = \langle\!\langle \mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3 \rangle\!\rangle \langle\!\langle \prod_{i=3}^n [X_{2i-1,k}, X_{2i,k}] \mid k = 1,2,3 \rangle\!\rangle \langle\!\langle g_1, g_2, g_3 \rangle\!\rangle \\
\qquad \Big\Vert \\
\qquad Y \mapsto W_2 \prod_{i=3}^n [X_{2i-1,3}, X_{2i,3}] g_3 W_1 \\
\qquad X \mapsto V_1^{-1} g_2^{-1} \prod_{i=3}^n [X_{2i,2}, X_{2i-1,2}] V_3^{-1} W_1^{-1} g_3^{-1} \prod_{i=3}^n [X_{2i,3}, X_{2i-1,3}] W_2^{-1} V_2^{-1} \\
\qquad \Big\Downarrow \\
U_1 V_2 W_2 \prod_{i=3}^n [X_{2i-1,3}, X_{2i,3}] g_3 W_1 V_3 \prod_{i=3}^n [X_{2i-1,2}, X_{2i,2}] g_2 V_1 \prod_{i=3}^n [X_{2i-1,1}, X_{2i,1}] g_1 \\
\qquad \Big\Vert \\
\qquad X_{k,1} \mapsto X_{k,1}^{g_1^{-1}} \qquad\qquad\qquad \forall k > 4 \\
\qquad X_{k,2} \mapsto X_{k,1}^{g_1^{-1} V_1^{-1} g_2^{-1}} \qquad\qquad \forall k > 4 \\
\qquad X_{k,3} \mapsto X_{k,1}^{g_1^{-1} V_1^{-1} g_2^{-1} V_3^{-1} W_1^{-1} g_3^{-1}} \quad \forall k > 4 \\
\qquad \Big\Downarrow \\
U_1 V_2 W_2 g_3 W_1 V_3 g_2 V_1 g_1 \prod_{i=3}^n [X_{2i-1,3}, X_{2i,3}] \prod_{i=3}^n [X_{2i-1,2}, X_{2i,2}] \prod_{i=3}^n [X_{2i-1,1}, X_{2i,1}] \\
\qquad \Big\Downarrow \ \mathfrak{nf}_{U_1 V_2 W_2 g_3 W_1 V_3 g_2 V_1 g_1} \\
R_4(X_{*,*}) g_{\mu(3)}^{X_{3,3}} g_{\mu(2)}^{X_{4,1}} g_{\mu(1)} \prod_{i=3}^n [X_{2i-1,3}, X_{2i,3}] \prod_{i=3}^n [X_{2i-1,2}, X_{2i,2}] \prod_{i=3}^n [X_{2i-1,1}, X_{2i,1}] \\
\qquad \Big\Vert \\
\qquad X_{k,i} \mapsto X_{k,i}^{g_3^{X_{3,3}} g_2^{X_{4,1}} g_1} \quad \forall k > 4, i = 1,2,3 \\
\quad R_{3(n-2)+4} g_{\mu(3)}^{X_{3,3}} g_{\mu(2)}^{X_{4,1}} g_{\mu(1)} = \Phi_\gamma^{\mathrm{nf}}(\mathcal{E}^{(n)}).
\end{array}$$

We denote by $\psi_{n,\gamma}$ the composition of this $\mathfrak{S}_3$-homomorphisms and remark that $\psi_{\gamma,n}^{-1}(X_{3,3}) = \psi_{\gamma,2}^{-1}(X_{3,3})$, $\psi_{\gamma,n}^{-1}(X_{4,1}) = \psi_{\gamma,2}^{-1}(X_{4,1})$ and if $\gamma' \colon F_{\mathbb{N}\times 3} \to Q$ is such that $\gamma'(X_{i,k}) = \mathbb{1}$ for all $i > 2$ and $k = 1,2,3$ then $(\gamma' * \pi) \circ \psi_{n,\gamma} = (\gamma' * \pi) \circ \psi_{2,\gamma}$. Hence $\Gamma_2^{n,q}(\gamma) = \Gamma_2^{2,q}(\gamma) =: \Gamma_2^q(\gamma)$.
We fix now further $\gamma' \in \Gamma_2^q(\gamma)$ and denote

$$x := \mathrm{rep}(\gamma'(X_{3,3})), \qquad\qquad y := \mathrm{rep}(\gamma'(X_{4,1})).$$

Because x and y are specified by the choice of γ' in $\Gamma_2^q(\gamma)$ we denote further by $h_{q,\gamma'} = h_{x,y,\mu}$ the map as defined in Lemma 4.48 and consider the equation $\mathcal{E}' = R_{3n-2} h_{q,\gamma'}(g)$. There exists an element $p \in \mathrm{Stab}_2 / \mathfrak{S}_3''$ such that $h_{q,\gamma'}(g)^\tau = \bar{h}_{x,y,\mu}(q) \cdot p$. Hence if all the constrained equations $R_{3n-2} f$ with $f^\tau \in \bar{h}_{x,y,\mu}(q) \cdot \mathrm{Stab}_2 / \mathfrak{S}_3''$ are solvable, then the constrained equations $(R_n g, \gamma)$ are solvable for all $g^\tau = q$.
In the case that $(\gamma', \bar{h}_{x,y,\mu}(q) \cdot p)$ is a good pair for all $p \in \mathrm{Stab}_2(\mathfrak{S}_3)/\mathfrak{S}_3''$ we call γ' a good succeeding constraint for the pair (q, γ). We will use this set later again and denote by

$$\mathrm{Desc}(\gamma', q) = \{\bar{h}_{x,y,\mu}(q) \cdot p \mid p \in \mathrm{Stab}_2(\mathfrak{S}_3)/\mathfrak{S}_3''\} \subset \mathfrak{S}_3'/\mathfrak{S}_3''$$

the set of possible descendants of q under the choice of $\gamma' \in \Gamma_2^q(\gamma)$.

Proposition 4.50. *There is a subset $\mathfrak{R}_1 \subset \mathfrak{R}$ with the following properties:*

(1) All $\gamma \in \mathfrak{R}_1$ have nontrivial activity in some component.
(2) For all $q \in \mathfrak{S}_3'/\mathfrak{S}_3''$ there is $\gamma \in \mathfrak{R}_1$ such that (q, γ) is a good pair.
(3) For every good pair (q, γ) with $q \in \mathfrak{S}_3'/\mathfrak{S}_3''$ and $\gamma \in \mathfrak{R}_1$ there is a good succeeding constraint γ' that is under an automorphism $\varphi \in U_n$ equivalent to an element of $\mathfrak{R}_1$.

Proof. We can compute the sets $\Gamma_2^q(\gamma)$ explicitly for every $q \in \mathfrak{S}_3{}'/\mathfrak{S}_3{}''$ and every reduced constraint γ. If we take for $\mathfrak{R}_1$ the set of constraints $\{\gamma_4, \dots, \gamma 5\}$ we can verify that all properties are satisfied by a direct computation.
We implemented this computation in GAP. See the file `examples/GuptaSidkiGW.g` and Chapter 5, Section 3.3 for details.

□

The previous proposition already shows that every element $g \in \mathfrak{S}_3{}'$ is a product of at most two commutators of elements in $\widetilde{\mathfrak{S}_3}$ the completion of $\mathfrak{S}_3$.
We fix now for every good pair (γ, q) with $\gamma \in \mathfrak{R}_1$ a good succeeding constraint $\operatorname{suc}(\gamma, q) \in \mathfrak{R}_1$. Furthermore for $g \in \mathfrak{S}_3{}'$ and an initial constraint $\gamma \in \mathfrak{R}_1$ such that (g, γ) is a good pair we define the *succeeding sequence* of (g, γ) by setting $(g_0, \gamma_0) = (g, \gamma)$, $\gamma_{k+1} = \operatorname{suc}(\gamma_k, g_k^\tau)$ and $g_{k+1} = h_{x,y,\mu}(g_k)$ where the x, y and μ are determined by γ_{k+1}.
So far the procedure is quite analogue to the proof of the commutator width of the Grigorchuk group. The analog next step would be to use the following lemma.

Lemma 4.51. *Let (g_k, γ_k) be the succeeding sequence of a good pair (g, γ). If $(g_i, \gamma_i) = (g_j, \gamma_j)$ for some distinct i, j then the equation $(R_n g, \gamma)$ is solvable for all $n \geq 2$.*

However we could not prove that for a succeeding sequence $(g_i, \gamma_i)_{i \in \mathbb{N}}$ the set $\{g_i \mid i \in \mathbb{N}\}$ is always finite. We will in the following define a length on $\mathfrak{S}_3$ that grows at most linearly when passing to the next element of the succeeding sequence.
Then a refinement of the term *good succeeding constraint* will prove the final step.

Definition 4.52. For $g \in \mathfrak{S}_3$ let $\ell(g)$ be the minimal number m such that g is a product of m elements t's and t^2's and an arbitrary number of a's. I.e

$$\ell(g) = \min\{m \in \mathbb{N} \mid g = h_0 \prod_{i=1}^{m} T_i^{h_i},\ T_i = t, t^2,\ h_i = \mathbb{1}, a, a^2\}.$$

Lemma 4.53. *Every element $g \in \mathfrak{S}_3{}'$ is a product of at most $\ell(g) + 1$ commutators.*

Proof. We note first that for $g \in \mathfrak{S}_3{}'$ there is a representation of g of the form $g = h_0 \prod_{i=1}^m T_i^{h_i}$ with $T_i \in \{t, t^2\}$ and $h_i \in \{\mathbb{1}, a, a^2\}$ with $h_0 = \mathbb{1}$ because otherwise the element g would be of nontrivial activity. Now we can multiply g with a product of at most m commutators $C_1, \dots, C_m$ to obtain the form $g = T_1' \cdot T_2'^a \cdot T_3'^{a^2} \cdot C_1 \cdots C_m$ with $T_i' \in \{\mathbb{1}, t, t^2\}$. There are only finitely many such elements $T_1' \cdot T_2'^a \cdot T_3'^{a^2} \in \mathfrak{S}_3{}'$ hence there is a constant N such that every such element is a product of at most N commutators and hence every element $g \in \mathfrak{S}_3{}'$ is a product of at most $\ell(g) + N$ commutators. In fact N can chosen to be 1 because every such element $T_1' \cdot T_2'^a \cdot T_3'^{a^2} \in \mathfrak{S}_3{}'$ is a commutator.

$$\begin{aligned} \mathbb{1} &= [\mathbb{1}, \mathbb{1}], & t^2 t^{a^2} &= [t, a^2], & t(t^2)^{a^2} &= [t^2, a^2], \\ t^2 t^a &= [t, a], & t t^a t^{a^2} &= [a^2 t^2, t^2 a^2], & t^a (t^2)^{a^2} &= [t^2 a, a], \\ t(t^2)^a &= [t^2, a], & (t^2)^a t^{a^2} &= [ta, a], & t^2 (t^2)^a (t^2)^{a^2} &= [a^2 t, t a^2]. \end{aligned}$$

□

Corollary 4.54. *For every $g \in \mathfrak{S}_3{}'$ there is a reduced constraint $\gamma \in \mathfrak{R}$ such that the equation $(R_{\ell(g)+1} g, \gamma)$ is solvable.*

Lemma 4.55. *Let $x, y \in \{\mathbb{1}, a, a^2, t, at, a^2t, t^2, at^2, a^2t^2\}$ be elements of the previously chosen transversal of $\mathfrak{S}_3/\mathfrak{S}_3{}'$ and $\sigma \in \mathrm{S}_3$ be an arbitrary permutation. Then we have for every $g \in \mathfrak{S}_3$ that $\ell(h_{x,y,\sigma}(g)) \leq \ell(g) + 4$.*

Proof. Let $\ell(g) = n$ and g and $h_{x,y,\sigma}(g)$ be written as a word over the letters a, a^2, t, t^2 in the minimal form according to the definition of ℓ.
The number of occurrences of the elements t and t^2 in the states $g@1$, $g@2$ and $g@3$ depends only on the the number of occurrences of this letters in the word representing g and is bounded in sum by $\ell(g)$ hence $\ell(h_{x,y,\sigma}(g)) \leq \ell(g) + 2(\ell(x) + \ell(y)) \leq \ell(g) + 4$. □

We denote four sets:

$$\begin{aligned}
\mathrm{GP} &= \{(q,\gamma) \in \mathfrak{S}_3{}'/\mathfrak{S}_3{}'' \times \mathfrak{R}_1 \mid (q,\gamma) \text{ is a good pair }\},\\
S_0 &= \{q \in \mathfrak{S}_3{}'/\mathfrak{S}_3{}'' \mid \exists! \gamma \in \mathfrak{R}\colon (q,\gamma) \text{ is a good pair }\},\\
S_1 &= \{(q,\gamma) \in \mathrm{GP} \mid \forall \gamma' \in \Gamma_2^q(\gamma) \exists q' \in \mathrm{Desc}\colon q' \notin S_0\},\\
S_2 &= \{(q,\gamma) \in \mathrm{GP} \mid \exists \gamma' \in \Gamma_2^q(\gamma) \forall q' \in \mathrm{Desc}\colon q' \notin S_1\}.
\end{aligned}$$

Lemma 4.56. *$S_1 = S_2$.*

Proof. This is again a finite calculation. The classification in which of these sets a good pair (q,γ) lies can be done during the verification of Proposition 4.50. It is implemented in GAP. For details see the file `examples/GuptaSidkiCW.g` and Chapter 5, Section 3.3. □

Proposition 4.57. *For every $g \in \mathfrak{S}_3{}'$ the equation $R_2 g$ has a solution.*

Proof. Let $g \in \mathfrak{S}_3{}'$ be an arbitrary element and $(p,\gamma) \in \mathrm{GP}$ be a good pair such that $g^\tau = p$. The existence of such a good pair is granted by Proposition 4.50. Let (g_i, γ_i) be the corresponding succeeding sequence. There is $n \in \mathbb{N}$ such that $\ell(g) + 4n + 1 \leq 3^n + 1$ and thus the equation $(R_2 g, \gamma)$ is solvable if the constrained equation $(R_{\ell(g_n)+1} g_n, \gamma_n)$ is solvable.
Let $q = g_n^\tau$. According to Proposition 4.50 we do already know that $(q,\gamma_n) \in \mathrm{GP}$. If $q \in S_0$ then there is only one reduced constraint that forms a good pair with q and hence because of Lemma 4.53 the constrained equation $(R_{\ell(g_n)+1} g_n, \gamma_n)$ is solvable.
If $(q,\gamma_n) \notin S_1$ then there exists $\gamma' \in \Gamma_2^q(\gamma_n)$ such that $(h_{q,\gamma'}(g_n)^\tau, \gamma') \in S_0$. Thus the equation $(R_{\ell(h_{q,\gamma'}(g))+1} h_{q,\gamma'}(g), \gamma_n)$ is solvable and hence also $(R_{\ell(g_n)+1} g_n, \gamma_n)$.
If we have that $(q,\gamma_n) \in S_1$ then, according to Lemma 4.56, it exists $\gamma' \in \Gamma_2^q(\gamma_n)$ such that $(h_{q,\gamma'}(g_n)^\tau, \gamma') \notin S_1$. Let us denote by $g' = h_{q,\gamma'}(g_n)$ and by $q' = g'^\tau$. Then it exists $\gamma'' \in \Gamma_2^{q'}(\gamma')$ such that $(h_{q',\gamma''}(g'))^\tau \in S_0$. Hence as seen before all the constrained equations $(R_{\ell(h_{q',\gamma''}(g'))+1} h_{q',\gamma''}(g'), \gamma'')$, $(R_{\ell(g')+1} g', \gamma')$, $(R_{\ell(g_n)+1} g_n, \gamma_n)$ and $(R_2 g, \gamma)$ are solvable.

□

Remark 4.58. We note that the question whether $\mathfrak{S}_3$ has commutator width 1 is still open.
In the Grigorchuk group we were able to find a finite quotient that has commutator width 2. But in the Gupta-Sidki group the size of the groups $\mathrm{Perm}_n(\mathfrak{S}_3)$ grows to fast. For $n = 2, 3, 4, 5, 6$ the size of $\mathrm{Perm}_n(\mathfrak{S}_3)$ is 3^3, 3^7, 3^{19}, 3^{55} and 3^{163}. For the cases $n \leq 4$ these groups have commutator width one.
If the commutator width of $\mathfrak{S}_3$ is 1 then one could try to start with the equation $[X_1, X_1]g$ and after the choice of a good active constraint γ we would need to solve

$$\begin{tikzcd}
R_2 g, \gamma & & & \\
R_4 g_1, \gamma_1 \arrow[u] & & & \\
& \text{Solvable if } q \in S_0. & & \\
R_{3^n+1} g_n, \gamma_n \arrow[uu, dashed] & \boxed{R_{\ell(g_n)+1} g_n, \gamma_n} \arrow[l] & & \\
& & \text{Solvable if } (q,\gamma) \notin S_1. & \\
& R_{3\ell(g_n)+1} h_{q,\gamma'}(g_n), \gamma' \arrow[uu] & \boxed{R_{\ell(h_{q,\gamma'}(g_n))+1} h_{q,\gamma'}(g_n), \gamma'} \arrow[l] & \\
& & R_{3\ell(g')+1} h_{q',\gamma''}(g'), \gamma'' \arrow[u] & \\
& & & \text{Solvable if } (q,\gamma) \in S_1. \\
& & & \boxed{R_{\ell(h_{q',\gamma''}(g'))+1} h_{q',\gamma''}(g'), \gamma''} \arrow[uul]
\end{tikzcd}$$

FIGURE 11. Diagram that shows how the solution of the equation $R_2 g$ is derived. The arrows indicate that a solution can be lifted.

$[X_1, X_2] g_2^{X_3} g_1$ for some $\gamma' \in \Gamma_1^g(\gamma)$. Note that we cannot reduce γ' by U_2 any further because U_2 only fixes the commutator part. Then in a second step we would need to solve the constrained equations in $\Phi_{\gamma'}([X_1, X_2] g_2^{X_3} g_1)$ for a constraint depending on γ'. If g_2 is active then this is equivalent to an equation of genus 2. If there is always a choice such that we would obtain an equation with constraint $\gamma \in \mathfrak{R}_1$ then this would show that the commutator width of $\mathfrak{S}_3$ is one. Unfortunately there are too many choices to test them all.

7. Mothergroups

The easiest examples of bounded groups and polynomial bounded groups have in common that their generating automata have only circles of length one and no activity in these circles. This is not a strong restriction: We can define groups, so called *mothergroups* where every element has this property and embed arbitrary bounded elements in this group. This groups where introduced in [**BKN10**] to prove the amenability of bounded self-similar groups. The notion was then extended in [**GOB13**] to obtain mothergroups of higher degree.

DEFINITION 4.59 ([**GOB13**]). For $n \geq 2$ and $d \geq -1$ the *(n, d)-mothergroup $\mathcal{M}_{n,d}$* is defined as follows:

$$\begin{aligned}
a_{-1,\pi} &= \langle\!\langle 1, \dots, 1 \rangle\!\rangle \pi \text{ for } \pi \in \mathrm{S}_n, \\
a_{i,\pi} &= \langle\!\langle a_{i,\pi}, a_{i-1,\pi}, 1, \dots, 1 \rangle\!\rangle \quad \text{for } \pi \in \mathrm{S}_n, 0 \leq i \leq d, \\
b_{0,\pi} &= \langle\!\langle b_{0,\pi}, 1, \dots, 1 \rangle\!\rangle \pi \quad \text{for } \pi \in \mathrm{Stab}_{\mathrm{S}_n}(1), \\
b_{i,\pi} &= \langle\!\langle b_{i,\pi}, b_{i-1,\pi}, 1, \dots, 1 \rangle\!\rangle \quad \text{for } \pi \in \mathrm{Stab}_{\mathrm{S}_n}(1), 1 \leq i \leq d, \\
A_i &= \langle a_{i,\pi} \mid \pi \in \mathrm{S}_n \rangle \quad \text{for } -1 \leq i \leq d, \\
B_i &= \langle b_{i,\pi} \mid \pi \in \mathrm{Stab}_{\mathrm{S}_n}(1) \rangle \text{ for } 0 \leq i \leq d, \\
\mathcal{M}_{n,d} &= \langle A_j, B_k \mid -1 \leq j \leq d,\ 0 \leq k \leq d \rangle .
\end{aligned}$$

Note that $A_i \cong \mathrm{S}_n$ and $B_i \cong \mathrm{Stab}_{\mathrm{S}_n}(1) \cong \mathrm{S}_{n-1}$ for every i.

Definition 4.60 ([**BKN10**]). The *n-mothergroup* $\mathcal{M}_n$ is defined as follows:

$$\begin{aligned} a_\pi &= \langle\!\langle 1,\dots,1\rangle\!\rangle\pi \text{ for } \pi \in \mathrm{S}_n, \\ b_{\pi,\pi_2,\dots,\pi_n} &= \langle\!\langle b_{\pi,\pi_2,\dots,\pi_n}, a_{\pi_2},\dots,a_{\pi_n}\rangle\!\rangle\pi \quad \text{for } \pi \in \mathrm{Stab}_{\mathrm{S}_n}(1), \pi_i \in \mathrm{S}_n, \\ \mathcal{M}_n &= \langle a_{\pi_1}, b_{\pi,\pi_2,\dots,\pi_n} \mid \pi \in \mathrm{Stab}_{\mathrm{S}_n}(1), \pi_i \in \mathrm{S}_n\rangle\,. \end{aligned}$$

Lemma 4.61. *The mothergroups $\mathcal{M}_{n,d}$ of degree $d \geq 1$ are not contracting.*

Proof. It is enough to prove this for $\mathcal{M}_{n,1}$ as $\mathcal{M}_{n,1} < \mathcal{M}_{n,d}$ for all $d \geq 1$. Choose $\pi \in \mathrm{S}_n$ with $\pi(1) = 2$, $\pi(2) = 1$. The group $\langle a_{1,\pi}, a_{0,\pi}, a_{-1,\pi}\rangle$ is a subgroup of $\mathcal{M}_{n,1}$. If it would be contracting, then $(a_1 \cdot a_0)^\ell$ is in the nucleus for all ℓ since

$$(a_1 \cdot a_0)^\ell @ \underbrace{1\dots1}_{m} = (a_1 \cdot a_0)^\ell \text{ for all } \ell, m.$$

But $a_1 \cdot a_0$ is of infinite order and thus there cannot be a finite nucleus. □

Lemma 4.62. $\mathcal{M}_{n,0} = \mathcal{M}_n$

Proof. It is

$$\begin{aligned} a_{-1,\pi} &= a_\pi, \\ a_{0,\pi} &= b_{1,\pi,1,\dots,1}, \\ b_{0,\pi} &= b_{\pi,1,\dots,1}. \end{aligned}$$

Therefore $G_{n,0} < \mathcal{M}_n$. For the other direction it is now enough to see that:

$$b_{\pi,\pi_2,\dots,\pi_n} = a_{0,\pi_2} \cdot a_{0,\pi_3}^{b_{0,(2,3)}} \cdot a_{0,\pi_4}^{b_{0,(2,4)}} \cdots a_{0,\pi_n}^{b_{0,(2,n)}} \cdot b_{0,\pi}.$$

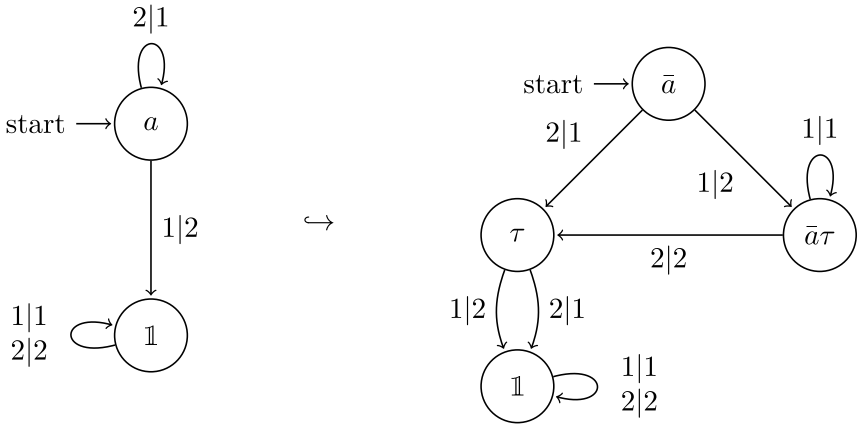
Hence $G_{n,0} = \mathcal{M}_n$. □

7.1. Properties of mothergroup elements. The following theorem motivates the consideration of the order problems in the mothergroups.

Theorem 4.63 ([**GOB13**],[**BKN10**]). *Every finitely generated subgroup of* $\mathrm{Poly}_n(d)$ *can be embedded in a mothergroup $\mathcal{M}_{m,d}$ for some $m \in \mathbb{N}$ possibly larger than n and this embedding can be effectively constructed.*

Corollary 4.64. *The order problem of the groups* $\mathrm{Poly}_n(d)$ *reduces to the order problem in the groups $\mathcal{M}_{n,d}$.*

Example 4.65. The adding machine can be embedded into the mothergroup $\mathcal{M}_{2,0}$.



7.2. Order problem in the group $\mathcal{M}_{n,d}$. In [**BBSZ13**] it is shown that the order of bounded group elements can be computed.

Definition 4.66 ([**BBSZ13**]). Let $g \in \mathrm{Aut}(\mathcal{A}^*)$ be a tree automorphism. Then the *orbit signalizer* is defined as the set

$$\mathrm{OS}(g) = \{(g^m)@w \mid w \in \mathcal{A}^*, m = |\mathrm{Orb}_g(w)|\}.$$

We want to build a graph from the orbit signalizer and therefore make the following definition.

Definition 4.67. Let $g \in \mathrm{Aut}(\mathcal{A}^*)$ and $w \in \mathcal{A}^*$, then we define the orbit signalizer successor of g under the word w by

$$\mathrm{oss}_w(g) = g^{|\mathrm{Orb}_g(w)|}@w.$$

So the *orbit signalizer* is the set $\mathrm{OS}(g) = \{\mathrm{oss}_w(g) \mid w \in \mathcal{A}^*\}$.

Lemma 4.68. *The successor can be computed iteratively by* $\mathrm{oss}_{vx}(g) = \mathrm{oss}_x(\mathrm{oss}_v(g))$ *for* $x \in A$ *and* $v \in A^*$*.*

Proof. Denote $m = |\mathrm{Orb}_g(vx)|$ and $n = |\mathrm{Orb}_g(v)|$ then we have $n \mid m$ and:

$$\begin{aligned}
\mathrm{oss}_{vx}(g) = g^m@vx &= g@vx \cdot g@(vx)^g \cdot \ldots \cdot g@(vx)^{g^{m-1}} \\
&= g@vx \cdot g@v^a x^{g@v} \cdot \ldots \cdot g@v^{g^{m-1}} x^{g^{m-1}@v} \\
&= g@vx \cdot \ldots \cdot g@v^{g^{n-1}} x^{g^{n-1}@v} \cdot g@vx^{g^n@v} \cdot \ldots \cdot g@v^{g^{m-1}} x^{g^{m-1}@v} \\
&= \left(g@v \cdot \ldots \cdot g@v^{g^{n-1}}\right) @x \cdot \ldots \cdot \left(g@v \cdot \ldots \cdot g@v^{g^{n-1}}\right) @x^{g^{m-n}@v} \\
&= \left(g^{|\mathrm{Orb}_g(v)|}@v\right)^{|\mathrm{Orb}_{g^n@v}(x)|} @x = \mathrm{oss}_x(\mathrm{oss}_v(g)) \ .
\end{aligned}$$

□

Definition 4.69. We define the graph of the orbit signalizer of a group element $g \in \mathrm{Aut}(\mathcal{A}^*)$ by the graph with vertex set $\mathrm{OS}(g)$ and draw an edge $(h, \mathrm{oss}_x(h))$ with label $|\mathrm{Orb}_h(x)|$ for every $h \in \mathrm{OS}(g)$ and $x \in \mathcal{A}$.

Proposition 4.70 ([**BBSZ13**]). *Let* $g \in \mathrm{Aut}(\mathcal{A}^*)$ *be a tree automorphism and denote by* $|g|$ *the order of the element. Then:*

1. $|g| = \mathrm{lcm}\{|\mathrm{oss}_x(g)| \mid x \in \mathcal{A}\} \cdot |\mathrm{act}(g)|$.
2. *If the orbit signalizer* $\mathrm{OS}(g)$ *is finite for all* g *in a collection* $G \subseteq \mathrm{Aut}(\mathcal{A}^*)$ *then the order problem for* G *is decidable.*
3. *If* $g \in G$ *is finite state and* $\mathrm{OS}(g)$ *is infinite, then* g *is of infinite order.*

Lemma 4.71 ([**BBSZ13**]). *All elements* $g \in \mathrm{Poly}(0)$ *have finite orbit signalizer.*

Corollary 4.72 ([**BBSZ13**]). *The order problem of the groups* $\mathcal{M}_{n,0}$ *is decidable.*

We present a different proof for the case $\mathcal{M}_{2,0}$:

Lemma 4.73. *Every nontrivial element of* $\mathcal{M}_{2,0}$ *is either of order* 2 *or of infinite order.*

Proof. Let us denote the two generators of $\mathcal{M}_{2,0}$ by $a = (1,2)$ and $x = \langle\!\langle x, a \rangle\!\rangle$. Then every element of $\mathcal{M}_{2,0}$ is conjugate to either $(ax)^n$ for some n or to $(ax)^n a$ for some $n \in \mathbb{N}$. Since ax is of infinite order all elements of the first kind are of infinite order as well. Moreover $(ax)^n a (ax)^n a = (ax)^{n-n} a^2 = \mathbb{1}$ and hence all elements of the second kind are of order 2. □

We will now consider the group $\mathcal{M}_{2,1}$ in more details. It is by definition generated by a_{-1}, a_0 and a_1 all the b-generators are trivial. For simplicity let us denote $a = a_{-1}, x = a_0, y = a_1$. All those generators are of order two.
In this group the orbit signalizer of elements is is not necessarily finite.

Example 4.74. Consider $g = yxya$ or more generally $g_n = yxy(xa)^n a$. Then the decomposition of g_{2n} is $\langle\!\langle yxy(xa)^n, xax(ax)^n \rangle\!\rangle(1,2)$ and hence the orbit signalizer of g is the infinite graph with vertices $\{g_{2n} \mid n \in \mathbb{N}\}$. Note that all elements in this set are distinct because xa is of infinite order.

The order problem reduces to the question whether the orbit signalizer of an element is finite or infinite.

Remark 4.75. There are not many short nontrivial relators in $\mathcal{M}_{2,1}$. One nontrivial group relation is $yxayayaxy(ax)^2 a = 1$.

Lemma 4.76. *The order problem in $\mathcal{M}_{2,1}$ is decidable.*

Proof. Every element $g \in \mathcal{M}_{2,1}$ can be written (not necessarily unique) in the form $g = (xa)^{n_0} a^{\varepsilon_0} \prod_{i=1}^m y(xa)^{n_i} a^{\varepsilon_0}$ for $n_i \in \mathbb{Z}$ and $\varepsilon_i \in \{0,1\}$. We choose a representation in this form with minimal m and denote by $\#_y(g)$ this minimal m and prove by induction on $\#_y(g)$ that we can compute the order of g.
If $\#_y(g) = 0$ then $g \in \mathcal{M}_{2,0}$ and hence we can determine its order. Assume now that $\#_y(g) > 0$ then g is conjugate to an element of the form $\prod_{i=1}^m y(xa)^{n_i} a^{\varepsilon_0}$. We associate to each such element a triple (m, n, ℓ) with $n = \sum |n_i|$ aka "number of x's" and $\ell = \sum n_i + \varepsilon_i$ aka "number of a's". The element g has nontrivial activity if and only if ℓ is odd for any such representation.
If g is inactive then $|g| = \operatorname{lcm}\{|g@1|, |g@2|\}$ and at least one of the two has smaller m-length and we thus know its order by induction. The set

$$\{\operatorname{oss}_w(g) \mid w \in \mathcal{A}^*, \#_y(\operatorname{oss}_w(g)) = \#_y(g), \operatorname{act}(\operatorname{oss}_v(g)) = \mathbb{1} \forall v \leq w\}$$

is finite (because g is finite state) and hence we can either determine the order of g or we need to determine the order of an active element with same m-length.
If g is active we distinct between the possible associated triples (m, n, ℓ).

(e,o,o): If m is even and n is odd. Then $|g| = 2|g@1 \cdot g@2|$ and $g@1 \cdot g@2$ is of the form (m', n', ℓ') with again m' even, and n', ℓ' odd. An element of this form is always active and hence never trivial. Thus in this case $|g| = \infty$.

(e,e,o): If m and n are even the orbit successor $h = \operatorname{oss}_1(g)$ is of type (e,e,e) and hence inactive. The state $h@2$ has always a smaller value of $\#_y$ and we hence know its order. If $h@1$ has also a smaller $\#_y$ value we know the order of h. Otherwise if we write $h = \prod_{i=1}^m y(xa)^{n_i} a^{\varepsilon_i}$ we have that $n_i + \varepsilon_i$ is even for all i and hence

$$h@1 = \prod_{i=1}^m y(xa)^{\lceil \frac{n_i}{2} \rceil} a^{n_i \pmod 2}. \tag{1}$$

Moreover if we denote by n_o the number of odd n_i's we have $\sum_{i=1}^m \lceil \frac{n_i}{2} \rceil = \frac{1}{2} \sum n_i n_o$ and because n_o is an even number therefore the type of $h@1$ is either (e,o,o) or (e,e,e). If it's the first one we already know how to compute the order and if it's the second one we are in the same case again but with an eventually shorter element.

(o,o,o): If g is of this type then its order is twice the order of the successor $\operatorname{oss}_1(g)$ that is of type (o,e,o) which we will consider next.

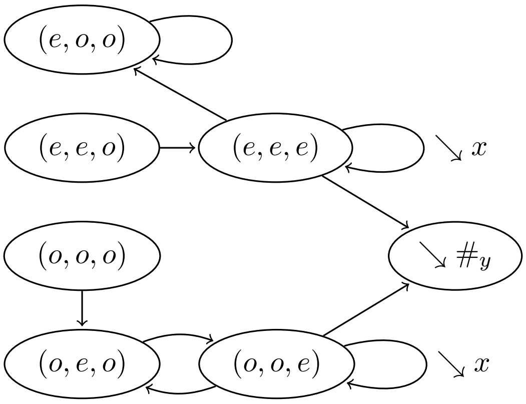


FIGURE 12. Diagram describing the different cases.

(o, e, o): Assume g is of type (m, n, ℓ) with n even and m and ℓ odd. Then its order is twice that of $h = \mathrm{oss}_1(g)$ and this is of type (o, o, e). As before the order of h is the least common multiple of the order of its states. If no state has the same $\#_y$ value as h we can compute the order by induction. Otherwise we write $h = \prod_{i=1}^{m} y(xa)^{n_i} a^{\varepsilon_i}$ and have again the equality (1). This time however n_o is odd and hence the type of $h@1$ is either (o, o, e) or (o, e, o). If we stay in the case (o, o, e) the number of x's gets shorter but in the other case it can get longer again. The type of h is (in absolute values) at most $(m, m+n, n)$ and hence the number of $x's$ in $h@1$ is at most $\frac{m+n+n_o}{2}$. Note that n_o is bounded by m. If we assume the worst case that m is larger then n and we iterate between this cases $(o, o, e) \to (o, e, o) \to (o, o, e)$ we end up by a maximal number of x's of at most $\sum_{k=0}^{\infty} 2^{-k} m = 2m$ and hence we can compute the order in this case.

□

CHAPTER 5

Computer algebra

The results in the previous chapter often rely on a large number of computations some of them can be done easily by hand but with a larger number of computations the number of potential errors increases. Many of the calculations follow the same pattern. Very often the normal form of an equation has to be determined and similarly often we compute the image of a constrained equation $(\mathcal{E}, \gamma)$ under Φ_γ.
It is convenient to at least check the results on this computations by a computer. To be able to do so we need to implement some layers of abstraction to use a similar language in the source code of the algorithms.
Most of this abstraction layers are already implemented in the GAP system.

1. The system GAP

The computer algebra system GAP ([**GAP14**]) is an open source computer algebra system originally focused for computations in groups.
Beside a core system that provides basic functionality the program is structured into packages that provide specialized methods. For more information about the system see the documentation [**GAP17**].

2. Implementation of equations

We introduce a new package for GAP with the name *equations*. It aims to allow definition of equations as defined in Chapter 3 with basic functionality like computing the normal form of quadratic equations or determining the genus of an equation. However in the current state the package becomes much more useful in combination with the package *fr* ([**Bar16b**]) that allows computations in functionally recursive groups.

2.1. Free products. In the definition of an equation $\mathcal{E} \in F * G$ we need the free product of F and G. GAP already supports free products of finitely presented groups but we need a more basic approach. For this purpose we declared a new filter to the system with the name `IsGeneralFreeProduct` that is implemented by simply storing the two given groups and embeddings from the free factors.
We use this filter as collection for `FreeProductElements`. Those elements consist of a mixed type list where the elements are in one of the free factors and a second list which stores the correspondence of elements to its free factors. During its creation the list will be automatically reduced such that two consecutive list entries never lie in the same free factor.

```
gap> F := FreeGroup(2);; S3 := SymmetricGroup(3);;
gap> G := FreeProduct(F,S3);;
gap> Print(FreeProductElm(G,[F.1,F.2,(1,2),F.1],[1,1,2,1]));
```

```
FreeProductElm([ f1*f2, (1,2), f1 ])
```

```
gap> G := FreeProduct(S3,S3);;
gap> Print(FreeProductElm(G,[(1,2,3),(1,2),(1,2)],[1,1,2]));
```

```
FreeProductElm([ f1*f2, (1,2), f1 ])
```

There is a second representation of this elements that does not reduces the given word.

```
gap> G := FreeProduct(F,S3);;
gap> Print(FreeProductElmLetterRep(G,[F.1,F.2,(1,2),F.1],[1,1,2,1]));
```

```
FreeProductElm([ f1, f2, (1,2), f1 ])
```

2.2. Equations. We assign to the object `GeneralFreeProduct` the attribute `IsEquationGroup` if the second free factor is a free group. Internally we store the equation group in an `ComponentElementRep` with two components `group` and `free`. We can construct an equation group by the command `EquationGroup` that takes the two free factors as arguments. We can omit the second factor. It will then be the countable generated free group automatically.

To define an equation we can either specify the equation group and a list that represents the equation as word over the free factors of the equation group.

```
gap> F := FreeGroup(2);; S3 := SymmetricGroup(3);;
gap> EqG := EquationGroup(S3,F);;
gap> eq := Equation(EqG,[(1,2),F.1^2,(1,2,3),F.2^2,(2,3)]);
```

```
Equation in [ f1, f2 ]
```

```
gap> Print(eq);
```

```
FreeProductElm([ f1^2, (1,2,3), f2^2, (1,2,3) ])
```

Alternatively we can first get the possible variables of the equation group by the command `VariablesOfEquationGroup`. These variables are already members of the equation group and we can hence form an equation simply by multiplying this elements to others. A new method for `\*` for equations and general group elements is implemented to make this possible.

The equation is stored as a cyclical reduced `FreeProductElm`.

```
gap> EqG := EquationGroup(SymmetricGroup(3));;
gap> vars := VariablesOfEquationGroup(EqG);
gap> eq := Equation(vars[1]*(1,2,3)*vars[2]^-1*vars[1]*vars[2]);
```

```
Equation in [ X1, X2 ]
```

For a more detailed overview of all methods of the *equations* package see the manual, which is in the Appendix B.

3. Examples of Computations

The following examples are distributed with the *equation* package. The files are in the `phd/` directory of the git-hub repository of the package available under the url `https://github.com/ThGroth/gap-equations`.

3.1. Neumann-Segal groups. In Proposition 4.28 we proved that the Neumann-Segal groups $\mathfrak{N}_n$ have commutator width 1 for $n \geq 5$.
The calculation rely partly on the computation of the commutator width of A_n. In the file `examples/Ore.g` we implemented a function `GetCommutators` that given an element $\sigma \in \mathrm{A}_n$ computes two elements $\pi_1, \pi_2 \in \mathrm{A}_n$ such that $[\pi_1, \pi_2] = \sigma$. The implementation follows precisely the constructions in the proof of Proposition 4.18.

```
gap> Read("examples/Ore.g");
gap> sigma := (1,2,3)(4,5)(7,8,9,10);;
gap> pi := GetCommutators(sigma);
```

```
[ (1,3)(4,8,7,10), (1,3,2)(4,5,7,9,8) ]
```

```
gap> Comm(pi)=sigma;
```

```
true
```

In the file `examples/NeumannSegel.g` we implemented the methods used in the proof of Proposition 4.28. We define a function `CommutatorEquationSolver` that computes for every $g \in \mathfrak{N}_n$ a solution for the equation $[X, Y]g$.

```
gap> Read("examples/NeumannSegal.g");
gap> Nn := NeumannSegalGroup(7);
```

```
<recursive group over [ 1 .. 7 ] with 4 generators>
```

```
gap> RandNn := m ->Product([1..m],i->Random(GeneratorsOfGroup(Nn)));;
gap> g := RandNn(10);
```

```
<Mealy element on alphabet [ 1 .. 7 ] with 8 states>
```

```
gap> res := CommutatorEquationSolver(g);
```

```
[ <Mealy element on alphabet [ 1 .. 7 ] with 62 states>,
<Mealy element on alphabet [ 1 .. 7 ] with 52 states> ]
```

```
gap> IsOne(Comm(res)*g);
```

```
true
```

For large n quite some time of the computation is used to compute the sets N_n and M_n that form the nucleus of the group $\mathfrak{N}_n$. If many computations for the same n should be done, it is more convenient to pass these sets to the function `CommutatorEquationSolver`.

```
NeumannSegalRandomTest := function(deg,len,num)
   local Nn,RandomNn,N,M,i,g;
   Nn := NeumannSegalGroup(deg);
   RandNn := m ->Product([1..m],i->Random(GeneratorsOfGroup(Nn)));
   N := List(AlternatingGroup(deg),s->api(s,deg));;
   M := List(AlternatingGroup(deg),s->alphapi(s,deg));;
   Print("Nucleus␣computed\n");
   for i in [1..num] do
     g := RandNn(Random([1..len]));
     if not IsOne(Comm(CommutatorEquationSolver(g,N,M))*g) then
```

```
      Error("There␣is␣a␣problem␣with␣g",g);
    fi;
    Print(Int(i*100/num),"%␣done.\r");
  od;
  Print("100%␣done.␣Everything␣great\n");
end;
gap> NeumannSegalRandomTest(9,10,50);
```

```
Nucleus computed
100% done. Everything great
```

The rest of the file examples/NeumannSegal.g contains the computations that are needed for the proof of Proposition 4.35. In particular it contains the computations that show that every element of $\mathfrak{A}_5$ is a product of 6 conjugates of an arbitrary nontrivial permutation and the computations of the genera portrayed in Table 1.

3.2. Grigorchuk group. All the files and examples that are mentioned in the paper [**BGL17**] can be found in the directory GrigorchukCommutatorWidth/ of the git-hub repository. Additionally we provide two functions to compute all possible signatures of the equations in $\Phi_\gamma(\mathcal{E})$ given only the signature of the equation $\mathcal{E} \in F_\mathbb{N} * \mathfrak{G}$ in the file examples/GrigorchukGenus.g; one for the oriented case and one for the unoriented.

Given a signature (n, ℓ) of an oriented or unoriented equation $\mathcal{E}$ we know that $\mathcal{E}$ is equivalent to an equation of the form $\mathcal{E} = V \prod_{i=1}^{\ell-1} g_i^{X_i} g_\ell$ with $V \in F_\mathbb{N}$ and $g_i \in \mathfrak{G}$. The states of $\Phi_\gamma(\mathcal{E})$ do not depend on the activity of g_ℓ. To obtain all possible signatures of the equations in $\Phi_\gamma^{\mathrm{nf}}(\mathcal{E})$ it is therefore sufficient to take into account all possibilities of $\mathrm{act}(g_i)$ with $1 \le i < \ell$ and γ_{act}. The function PossibleDerivedSignaturesOriented lists all possibilities of $\mathrm{act}(g_i)$ and collects the outcome under the different values of γ_{act}. For example we consider equations with signature $(1, 2)$:

```
gap> Read("examples/GrigorchukGenus.g");
gap> PossibleDerivedSignaturesOriented(1,2);
```

```
FreeProductElm([ X1^-1*X2^-1*X1*X2*X3^-1, g1, X3, g2 ])
Signatures for act(g_i)=[(),()]: [ [ [1,2], 2 ], [ [1,4], 6 ] ]
Signatures for act(g_i)=[(1,2),()]: [ [ [ 2, 2 ], 8 ] ]
```

This prints first the initial equation $\mathcal{E}$. Then the output tells us that in the case of $\mathrm{act}(g_1) = ()$ there are two constraints γ such that the signature of the first state of $\Phi_\gamma(\mathcal{E})$ is $(1, 2)$ and 6 constraints γ such that the signature is $(1, 4)$. In the case that $\mathrm{act}(g_2) = (1, 2)$ the signature of the descendant equation is independent of the constraint always $(2, 2)$.

Similarly the function PossibleDerivedSignaturesOriented computes the derived signatures of unoriented equations. In contrast to the unoriented case it needs not be the case that all the equations in $\Phi_\gamma(\mathcal{E})$ are unoriented if $\mathcal{E}$ was. Thus the output of the function contains an additional letter to indicate if the corresponding equation is oriented or unoriented.

```
gap> PossibleDerivedSignaturesUnoriented(2,1);
```

```
FreeProductElm([ X1^2*X2^2, g1 ])
Signatures for act(g_i)=[ () ]:
O:[ 1, 2 ], 1
U:[ 2, 1 ], 1
U:[ 2, 2 ], 2
```

Thus for two choices of constraints γ the equations in $\Phi_\gamma(X_1^2 g_1^{X_2})$ are oriented of signature $(1,2)$ and for all other cases the derived equations are unoriented and either of signature $(2,1)$ or $(2,2)$.

3.3. Gupta-Sidki group. In Chapter 4, Section 6 we need multiple results that are calculated in GAP. The code that implements these computations can be found in the file `examples/GuptaSidkiCW.g`.
Execution of the file will take about 3 hours of computation on an ordinary machine. There is some status indication what is computed and longer calculations provide a progress bar. Note that it will need more then 6GB of RAM thus it should be executed using the `-o` parameter of GAP to provide more RAM. For example use `gap -o 7G examples/GuptaSidkiCW.g`.
The first computation proves Lemma 4.43 and computes the orbits of Q^4/U_2 and a representative system of these orbits. It uses the branch structure of $\mathfrak{S}_3$ and therefore needs the *fr* package.
The next step is the computation of the groups $\mathfrak{S}_3/\mathfrak{S}_3''$ and $\mathfrak{S}_3/\operatorname{Stab}(2)$ and the corresponding quotient homomorphisms. This is faster using the L-presentations of the groups and therefore we use the package *lpres* (see [**BH16**]).
After this the good pairs are computed as described in Lemma 4.46. The computation of the set $\Gamma_2^q(\gamma)$ for every $q \in \mathfrak{S}_3'/\mathfrak{S}_3''$ and reduced constraint γ is the most time consuming. The function `GetSuccessor` serves multiple purposes it does not only compute the set $\Gamma_2^q(\gamma)$ but also verifies that Proposition 4.50 is true for the set

$$\mathfrak{R}_1 = \{(\bar{t}\bar{a}, \mathbb{1}, \mathbb{1}, \mathbb{1}), (\bar{t}^2\bar{a}, \mathbb{1}, \mathbb{1}, \mathbb{1}), (\bar{t}, \bar{a}, \mathbb{1}, \mathbb{1}), (\bar{t}, \bar{a}^2, \mathbb{1}, \mathbb{1}), (\bar{t}, \mathbb{1}, \bar{a}, \mathbb{1})\}.$$

During the computation the good pairs in $(q,\gamma) \in \mathfrak{S}_3'/\mathfrak{S}_3''$ are also tested against the properties of Lemma 4.56.

CHAPTER 6

Conclusions

We have put some pieces to the mosaic of decision problems in self-similar groups. The inspection of decomposed equations with the help of computer algebra turns out to be very useful in particular also for those proofs that do not depend on the explicit calculation but are inspired by computer experiments. Our results leave space for enhancement:

1. Future work

For the Gupta-Sidki group it would be nice to pin down to the exact value of the commutator width. Moreover there are open questions for equations in tree automorphisms: Can we decide for all finitely generated bounded tree automorphism groups which quadratic equations hold? How about polynomially bounded automorphisms? Can this be generalized to systems of equations and equations of higher degree?

For the question of the order problem it would be great to have a result also for higher degree mothergroups thus we could conclude that the order problem is decidable for all polynomial bounded tree automorphisms.

In the same directions the *equation* package can develop. An integration of equation systems, and constrained equations as new objects can further enhance computer experiments. Moreover the functionality could be extended into the direction of equations over free groups. This theory is already well studied and algorithms for deciding equations already exists but are not yet implemented in GAP.

Appendices

CHAPTER A

Commutator Width In The First Grigorchuk group

The following paper is joint work with Laurent Bartholdi and Igor Lysenok. The core ideas were developed in June 2016 during a joint visit in the École normale supérieure de Paris.

I had studied a preprint of the article [**LMU16**] that already contained the idea of constrained equations and that it suffices for oriented equations of signature $(n, 1)$ over the Grigorchuk group $\mathfrak{G}$ to consider a finite number (2^{20}) of constraints. The authors conclude that the commutator width of the $\mathfrak{G}$ is finite.

I wrote some computer code to show that indeed 90 constraints already suffice. Together with the branching process of equations and some other properties of quotients of the Grigorchuk group I proved this reduced the question whether every element of $\mathfrak{G}'$ is a product of at most two commutators to a finite set of conditions. I implemented the branching process in the computer algebra system GAP to show that all these conditions are satisfied.

Later we extended the results to subgroups of $\mathfrak{G}$ and I found an element in $\mathfrak{G}'$ that is not a commutator.

Furthermore I found out that the same method could be used to prove that every element of $\mathfrak{G}'$ is a product of at most 6 conjugates of the generator a.

Commutator width in the first Grigorchuk group

Thorsten Groth

Laurent Bartholdi

Thorsten Groth

Igor Lysenok

Abstract. Let G be the first Grigorchuk group. We show that the commutator width of G is 2: every element $g \in [G,G]$ is a product of two commutators, and also of six conjugates of a. Furthermore, we show that every finitely generated subgroup $H \leq G$ has finite commutator width, which however can be arbitrarily large, and that G contains a subgroup of infinite commutator width. The proofs were assisted by the computer algebra system GAP.

1. Introduction

Let Γ be a group and let $\Gamma' = [\Gamma, \Gamma]$ denote its derived subgroup. The *commutator width* of Γ is the least $n \in \mathbb{N} \cup \infty$ such that every element of Γ' is a product of n commutators.

We compute, in this article, the commutator width of the *first Grigorchuk group* G, see §1.2 for a brief introduction. This is a prominent example from the class of *branched groups*, and as such is a good testing ground for decision and algebraic problems in group theory. We prove:

Theorem A. *The first Grigorchuk group and its branching subgroup have commutator width* 2*.*

It was already proven in [**LMU16**] that the commutator width of G is finite, without providing an explicit bound. Our result also answers a question of Elisabeth Fink [**Fin14**, Question 3]:

Corollary B. *Every element of G' is a product of* 6 *conjugates of the generator a and there are elements $g \in G'$ which are not products of* 4 *conjugates of a.*

There are examples of groups of finite commutator width with subgroups of infinite commutator width; and even finitely presented, perfect examples in which the subgroup has finite index, see Example 1.1. However, we can prove:

Theorem C. *Every finitely generated subgroup of G has finite commutator width; however, their commutator width cannot be bounded, even among finite-index subgroups. Furthermore, there is a subgroup of G of infinite commutator width.*

1.1. Commutator width.

Let Γ be a group. It is well-known that usually elements of Γ' are not commutators—for example, $[X_1, X_2] \cdots [X_{2n-1}, X_{2n}]$ is not a commutator in the free group F_{2n} when $n > 1$. In fact, every non-abelian free group has infinite commutator width, see [**Rhe68**].

On the other hand, some classes of groups have finite commutator width: finitely generated virtually abelian-by-nilpotent groups [**Seg09**], and finitely generated solvable groups of class 3, see [**Rhe69**].

Finite groups are trivial examples of groups of finite commutator width. There are finite groups in which some elements are not commutators, the smallest having order 96, see [**Gur80**]. On the other hand, non-abelian finite simple groups have commutator width 1, as was conjectured by Ore in 1951, see [**Ore51**], and proven in 2010, see [**LOST10**]. The commutator width cannot be bounded among finite groups; for example, $\Gamma_n = \langle x_1, \dots, x_{2n} \mid x_1^p, \dots, x_{2n}^p, \gamma_3(\langle x_1, \dots, x_{2n} \rangle) \rangle$ is a finite class-2 nilpotent group in which Γ_n' has order $p^{\binom{2n}{2}}$ but at most $\binom{p^{2n}}{2}$ elements are commutators, so Γ_n's commutator width is at least $n/2$.

Commutator width of groups, and of elements, has proven to be an important group property, in particular via its connections with “stable commutator length” and bounded cohomology [**Cal09**]. It is also related to solvability of quadratic equations in groups: a group Γ has commutator width $\leq n$ if and only if the equation $[X_1, X_2] \cdots [X_{2n-1}, X_{2n}]g = \mathbb{1}$ is solvable for all $g \in \Gamma'$. Needless to say, there are groups in which solvability of equations is algorithmically undecidable. It was proven in [**LMU16**] that there exists an algorithm to check solvability of quadratic equations in the first Grigorchuk group.

We note that if the character table of a group Γ is computable, then it may be used to compute the commutator width: Burnside shows (or, rather, hints) in [**Bur55**, §238, Ex. 7] that an element $g \in \Gamma$ may be expressed as a product of r commutators if and only if

$$\sum_{\chi \in \operatorname{Irr}(\Gamma)} \frac{\chi(g)}{\chi(1)^{2r-1}} > 0.$$

This may yield another proof of Theorem A, using the quite explicit description of $\operatorname{Irr}(G)$ given in [**Bar13**].

Consider a group Γ and a subgroup Δ. There is in general little connection between the commutator width of Γ and that of Δ. If Δ has finite commutator width and $[\Gamma : \Delta]$ is finite, then obviously Γ also has finite commutator width—for example, because $\Gamma / \operatorname{core}(\Delta)'$ is virtually abelian, and every commutator in Γ can be written as a product of a commutator in Δ with the lift of one in $\Gamma / \operatorname{core}(\Delta)'$, but that seems to be all that can be said. Danny Calegari pointed to us the following example:

Example 1.1. Consider the group Δ of orientation-preserving self-homeomorphisms of $\mathbb{R}$ that commute with integer translations, and let Γ be the extension of Δ by the involution $x \mapsto -x$. Then, by [**EHN81**, Theorems 2.3 and 2.4], every element of $\Gamma' = \Delta$ is a commutator in Γ, while the commutator width of Δ is infinite.

Both Γ and Δ can be made perfect by replacing them respectively with $(\Gamma \wr A_5)'$ and $\Delta \wr A_5$; and can be made finitely presented by restricting to those self-homeomorphisms that are piecewise-affine with dyadic slopes and breakpoints.

1.2. Branched groups. We briefly introduce the first Grigorchuk group [**Gri80**] and some of its properties. For a more detailed introduction into the topic of self-similar groups we refer to [**BGŠ03, Nek05**] and to Section 3.

A *self-similar group* is a group Γ endowed with an injective homomorphism $\Psi \colon \Gamma \to \Gamma \wr S_n$ for some symmetric group S_n. It is *regular branched* if there exists a finite-index subgroup $K \leq \Gamma$ such that $\Psi(K) \geq K^n$. It is convenient to write $\langle\!\langle g_1, \ldots, g_n \rangle\!\rangle \pi$ for an element $g \in \Gamma \wr S_n$. We call g_i the *states* of g and π its *activity*. It is also convenient to identify, in a self-similar group, elements with their image under Ψ.

A self-similar group may be specified by giving a set S of generators, some relations that they satisfy, and defining Ψ on S. There is then a maximal quotient Γ of the free group F_S on which Ψ induces an injective homomorphism to $\Gamma \wr S_n$.

The first Grigorchuk group G may be defined in this manner. It is the group generated by $S = \{a, b, c, d\}$, with $a^2 = b^2 = c^2 = d^2 = bcd = \mathbb{1}$, and with

$$a = \langle\!\langle \mathbb{1}, \mathbb{1} \rangle\!\rangle (1,2), \quad b = \langle\!\langle a, c \rangle\!\rangle, \quad c = \langle\!\langle a, d \rangle\!\rangle, \quad d = \langle\!\langle \mathbb{1}, b \rangle\!\rangle.$$

Here are some remarkable properties of G: it is an infinite torsion group, and more precisely for every $g \in G$ we have $g^{2^n} = \mathbb{1}$ for some $n \in \mathbb{N}$. On the other hand, it is not an Engel group, namely it is not true that for every $g, h \in G$ we have $[g, h, \ldots, h] = \mathbb{1}$ for a long-enough iterated commutator [**Bar16a**]. It is a group of intermediate word growth [**Gri83**], and answered in this manner a celebrated question of Milnor.

We have decided to concentrate on the first Grigorchuk group in the computational aspects of this text; though our code would function just as well for other examples of self-similar branched groups, such as the Gupta-Sidki groups [**GS83**].

1.3. Sketch of proofs. The general idea for the proof of Theorem A is the decomposition of group elements into states via Ψ. We show that each element $g \in G'$ is a product of two commutators by solving the equation $\mathcal{E} = [X_1, X_2] \cdots [X_{2n-1}, X_{2n}]g$ for all $n \geq 2$.

If there is a solution then the values of the variables X_i have some activities σ_i. If we fix a possible activity of the variables of $\mathcal{E}$ then by passing to the states of the X_i we are led to two new equations which (under mild assumptions and after some normalization process) yields a single equation of the same form but of higher genus. Not all solutions for the new equations lead back to solutions of the original equation. Thus instead of pure equations we consider *constrained* equations: we require the variables to lie in specified cosets of the finite-index subgroup K. The pair composed of a constraint and an element $g \in G$ will be a *good pair* if there is some n such that the constrained equation $[X_1, X_2] \cdots [X_{2n-1}, X_{2n}]g$ is solvable. It turns out that this only depends on the image of g in the finite quotient G/K'.

Then by direct computation we show that every good pair leads to another good pair in which the genus of the equation increases. We build a graph of good pairs which turns out to be finite since the constants of the new equation are states of the old equation and we can use the strong contracting property of G.

The computations could in principle be done by hand, but one of our motivations was precisely to see to which point they could be automated. We implemented them in the computer algebra system GAP [**GAP14**]. The source code for these computations is distributed with this document as ancillary material. It can be validated using precomputed data on a GAP standard installation by running the command `gap verify.g` in its main directory.

To perform more advanced experimentation with the code and to recreate the precomputed data, the required version of GAP must be at least 4.7.6 and the packages FR [**Bar16b**] and LPRES [**BH16**] must be installed.

2. Equations

We fix a set $\mathcal{X}$ and call its elements *variables*. We assume that $\mathcal{X}$ is infinite countable, is well ordered, and that its family of finite subsets is also well ordered, by size and then lexicographic order. We denote by $F_{\mathcal{X}}$ the free group on the generating set $\mathcal{X}$. We use $\mathbb{1}$ for the identity element of groups, and for the identity maps, to distinguish it from the numerical 1.

Definition 2.1 (G-group, G-homomorphism). Let G be a group. A *G-group* is a group with a distinguished copy of G inside it; a typical example is $H * G$ for some group H. A *G-homomorphism* between G-groups is a homomorphism that is the identity between the marked copies of G.

A *G-equation* is an element $\mathcal{E}$ of the G-group $F_{\mathcal{X}} * G$, regarded as a reduced word in $\mathcal{X} \cup \mathcal{X}^{-1} \cup G$. For $\mathcal{E}$ a G-equation, its set of *variables* $\operatorname{Var}(\mathcal{E}) \subset \mathcal{X}$ is the set of symbols in $\mathcal{X}$ that occur in it; namely, $\operatorname{Var}(\mathcal{E})$ is the minimal subset of $\mathcal{X}$ such that $\mathcal{E}$ belongs to $F_{\operatorname{Var}(\mathcal{E})} * G$.

An *evaluation* is a G-homomorphism $e\colon F_{\mathcal{X}} * G \to G$. A *solution* of an equation $\mathcal{E}$ is an evaluation s satisfying $s(\mathcal{E}) = \mathbb{1}$. If a solution exists for $\mathcal{E}$ then the equation $\mathcal{E}$

is called *solvable*. The set of elements $X \in \mathcal{X}$ with $s(X) \neq \mathbb{1}$ is called the *support* of the solution.

The support of a solution for an equation $\mathcal{E}$ may be assumed to be a subset of $F_{\mathrm{Var}(\mathcal{E})}$ and hence the data of a solution is equivalent to a map $\mathrm{Var}(\mathcal{E}) \to G$. The question of whether an equation $\mathcal{E}$ is solvable will be referred to as the *Diophantine problem* of $\mathcal{E}$.
Every homomorphism $\varphi\colon G \to H$ extends uniquely to an $F_{\mathcal{X}}$-homomorphism $\varphi_*\colon F_{\mathcal{X}} * G \to F_{\mathcal{X}} * H$. In this manner, every G-equation $\mathcal{E}$ gives rise to an H-equation $\varphi_*(\mathcal{E})$, which is solvable whenever $\mathcal{E}$ is solvable.

Definition 2.2 (Equivalence of equations). Let $\mathcal{E}, \mathcal{F} \in F_{\mathcal{X}} * G$ be two G-equations. We say that $\mathcal{E}$ and $\mathcal{F}$ are *equivalent* if there is a G-automorphism φ of $F_{\mathcal{X}} * G$ that maps $\mathcal{E}$ to $\mathcal{F}$. We denote by $\mathrm{Stab}(\mathcal{E})$ the group of G-automorphisms of $\mathcal{E}$.

Lemma 2.3. *Let $\mathcal{E}$ be an equation and let φ be a G-endomorphism of $F_{\mathcal{X}} * G$. If $\varphi(\mathcal{E})$ is solvable then so is $\mathcal{E}$. In particular, the Diophantine problem is the same for equivalent equations.*

Proof. If s is a solution for $\varphi(\mathcal{E})$, then $s \circ \varphi$ is a solution for $\mathcal{E}$. □

2.1. Quadratic equations.

A G-equation $\mathcal{E}$ is called *quadratic* if for each variable $X \in \mathrm{Var}(\mathcal{E})$ exactly two letters of $\mathcal{E}$ are X or X^{-1}, when $\mathcal{E}$ is regarded as a reduced word.
A G-equation $\mathcal{E}$ is is called *oriented* if for each variable $X \in \mathrm{Var}(\mathcal{E})$ the number of occurrences with positive and with negative sign coincide, namely if $\mathcal{E}$ maps to the identity under the natural map $F_{\mathcal{X}} * G \to F_{\mathcal{X}}/[F_{\mathcal{X}}, F_{\mathcal{X}}] * \mathbb{1}$. Otherwise $\mathcal{E}$ is called *unoriented*.

Lemma 2.4. *Being oriented or not is preserved under equivalence of equations.*

Proof. $\mathcal{E}$ is oriented if and only if it belongs to the normal closure of $[F_{\mathcal{X}}, F_{\mathcal{X}}] * G$; this subgroup is preserved by all G-endomorphisms of $F_{\mathcal{X}} * G$. □

2.2. Normal form of quadratic equations.

Definition 2.5 ($\mathcal{O}_{n,m}$,$\mathcal{U}_{n,m}$). For $m, n \geq 0$, $X_i, Y_i, Z_i \in \mathcal{X}$ and $c_i \in G$ the following two kinds of equations are called in *normal form*:

$$\mathcal{O}_{n,m}: \qquad [X_1, Y_1][X_2, Y_2] \cdots [X_n, Y_n] c_1^{Z_1} \cdots c_{m-1}^{Z_{m-1}} c_m \tag{1}$$

$$\mathcal{U}_{n,m}: \qquad X_1^2 X_2^2 \cdots X_n^2 c_1^{Z_1} \cdots c_{m-1}^{Z_{m-1}} c_m\ . \tag{2}$$

The form $\mathcal{O}_{n,m}$ is called the oriented case and $\mathcal{U}_{n,m}$ for $n > 0$ the unoriented case. The parameter n is referred to as the *genus* of the normal form of an equation.

We recall the following result, and give the details of the proof in an algorithmic manner, because we will need them in practice:

Theorem 2.6 ([**CE81**]). *Every quadratic equation $\mathcal{E} \in F_{\mathcal{X}} * G$ is equivalent to an equation in normal form, and the G-isomorphism can be effectively computed.*

Proof. The proof proceeds by induction on the number of variables. Starting with the oriented case: if the reduced equation $\mathcal{E}$ has no variables then it is already in normal form $\mathcal{O}_{0,1}$. If there is a variable $X \in \mathcal{X}$ occurring in $\mathcal{E}$ then X^{-1} also appears. Therefore the equation has the form $\mathcal{E} = uX^{-1}vXw$ or can be brought to this form by applying the automorphism $X \mapsto X^{-1}$. Choose $X \in \mathcal{X}$ in such a way that $\mathrm{Var}(v)$ is minimal.
We distinguish between multiple cases:

Case 1.0: $v \in G$. The word uw has fewer variables than $\mathcal{E}$ and can thus be brought into normal form $r \in \mathcal{O}_{n,m}$ by a G-isomorphism φ. If r ends with a variable, we use the G-isomorphism $\varphi \circ (X \mapsto Xw^{-1})$ to map $\mathcal{E}$ to the equation $rv^X \in \mathcal{O}_{n,m+1}$. If r ends with a group constant b, say $r = sb$, we use the isomorphism $\varphi \circ (X \mapsto Xbw^{-1})$ to map $\mathcal{E}$ to the equation $sv^X b \in \mathcal{O}_{n,m+1}$.

Case 1.1: $v \in \mathcal{X} \cup X^{-1}$. For simplicity let us assume $v \in \mathcal{X}$; in the other case we can apply the G-homomorphism $v \mapsto v^{-1}$. Now there are two possibilities: either v^{-1} occurs in u or v^{-1} occurs in w. In the first case $\mathcal{E} = u_1 v^{-1} u_2 X^{-1} v X w$, and then the G-isomorphism $X \mapsto X^{u_1} u_2$, $v \mapsto v^{u_1}$ yields the equation $[v, X] u_1 u_2 w$. In the second case $\mathcal{E} = uX^{-1} vXw_1 v^{-1} w_2$ is transformed to $[X, v] uw_1 w_2$ by the G-isomorphism $X \mapsto X^{uw_1} w_1^{-1}$, $v \mapsto v^{-uw_1}$. In both cases $u_1 u_2 w$, respectively $uw_1 w_2$ have fewer variables and so composition with the corresponding G-isomorphism results in a normal form.

Case 2: Length$(v) > 1$. In this case v is a word consisting of elements $X \cup X^{-1}$ with each symbol occurring at most once as v was chosen with minimal variable set, and some elements of G. If v starts with a constant $b \in G$ we use the G-homomorphism $X \mapsto bX$ to achieve that v starts with a variable $Y \in \mathcal{X}$, possibly by using the G-homomorphism $Y \mapsto Y^{-1}$. As in Case 1.1 there are two possibilities: Y^{-1} is either part of u or part of w. In the first case $\mathcal{E} = u_1 Y^{-1} u_2 X^{-1} Y v_1 X w$ we can use the G-isomorphism $X \mapsto X^{u_1 v_1} u_2$, $Y \mapsto Y^{u_1 v_1} v_1^{-1}$ to obtain $[Y, X] u_1 v_1 u_2 w$. In the second we use the G-isomorphism $X \mapsto X^{uw_1 v_1} v_1^{-1} w_1^{-1}$, $Y \mapsto Y^{-uw_1 v_1} v_1^{-1}$ to obtain $[X, Y] uw_1 v_1 w_2$. In both cases the second subword has again fewer variables and can be brought into normal form by induction.

Therefore each oriented equation can be brought to normal form by G-isomorphisms. In the unoriented case there is a variable $X \in \mathcal{X}$ such that $\mathcal{E} = uXvXw$. Choose v to have a minimal number of variables. By induction, the shorter word $uv^{-1}w$ is equivalent by φ to a normal form r.

The G-isomorphism $\varphi \circ (X \mapsto X^u v^{-1})$ maps $\mathcal{E}$ to $X^2 r$. If $r \in \mathcal{U}_{n,m}$ for some n, m, there remains nothing to do. Otherwise $r = [Y, Z]s$, and then the G-homomorphism

$$X \mapsto XYZ, \qquad Y \mapsto Z^{-1}Y^{-1}X^{-1}YZXYZ, \qquad Z \mapsto Z^{-1}Y^{-1}X^{-1}Z$$

maps $X^2 r$ to $X^2 Y^2 Z^2 s$. This homomorphism is indeed an isomorphism, with inverse

$$X \mapsto X^2 Y^{-1} X^{-1}, \qquad Y \mapsto XYX^{-1}Z^{-1}X^{-1}, \qquad Z \mapsto XZ.$$

Note that $s \in \mathcal{O}_{n,m}$. If $n \geq 1$ then this procedure can be repeated with Z, in place of X, r. □

For a quadratic equation $\mathcal{E}$ we denote by $\mathfrak{nf}(\mathcal{E}) := \mathfrak{nf}_{\mathcal{E}}(\mathcal{E})$ the image of $\mathcal{E}$ under the G-isomorphism $\mathfrak{nf}_{\mathcal{E}}$ constructed in the proof.

From now on we will consider oriented equations $\mathcal{O}_{n,1}$. For this we will use the abbreviation

$$R_n(X_1, \ldots, X_{2n}) = \prod_{i=1}^{n} [X_{2i-1}, X_{2i}]$$

and often write $R_n = R_n(X_1, \ldots, X_{2n})$ if the X_i are the first generators of $F_{\mathcal{X}}$.

2.3. Constrained equations.

Definition 2.7 (Constrained equations [**LMU16**]). Given an equation $\mathcal{E} \in F_{\mathcal{X}} * G$, a group H, a homomorphism $\pi\colon G \to H$ and a homomorphism $\gamma\colon F_{\mathcal{X}} \to H$, the pair $(\mathcal{E}, \gamma)$ is called a *constrained* equation and γ is called a *constraint* for the equation $\mathcal{E}$ on H.
A *solution* for $(\mathcal{E}, \gamma)$ is a solution s for $\mathcal{E}$ with the additional property that $\pi \circ s = \gamma$.

We note that the constraint γ needs only to be specified on $\operatorname{Var}(\mathcal{E})$.

3. Self-similar groups

Let T_n be the regular rooted n-ary tree and let S_n be the symmetric group on n symbols. The group $\operatorname{Aut}(T_n)$ consists of all root-preserving graph automorphisms of the tree T_n.
Let $T_{1,n}, \dots, T_{n,n}$ be the subtrees hanging from neighbors of the root. Every $g \in \operatorname{Aut}(T_n)$ permutes the $T_{i,n}$ by a permutation σ and simultaneously acts on each of them by isomorphisms $g_i\colon T_{i,n} \to T_{i^\sigma,n}$.
Note that for all i the tree T_n is isomorphic to $T_{i,n}$; identifying each $T_{i,n}$ with T_n, we identify each g_i with an element of $\operatorname{Aut}(T_n)$, and obtain in this manner an isomorphism

$$\Psi\colon \begin{array}{c} \operatorname{Aut}(T_n) \xrightarrow{\sim} \operatorname{Aut}(T_n) \wr S_n \\ g \mapsto \langle\!\langle g_1, \dots, g_n \rangle\!\rangle \sigma. \end{array}$$

A *self-similar group* is a subgroup G of $\operatorname{Aut}(T_n)$ satisfying $G \leq \Psi(G)$. For the sake of notation we will identify elements with their image under this embedding and will write $g = \langle\!\langle g_1, \dots, g_n \rangle\!\rangle \sigma$ for elements $g \in G$. Furthermore we will call $g_i \in G$ the *states* of the element g, will write $g@i := g_i$ to address the states, will call $\sigma \in S_n$ the *activity* of the element g, and will write $\operatorname{act}(g) := \sigma$.

3.1. Commutator width of $\mathbf{Aut(T_2)}$.

To give an idea of how the commutator width of Grigorchuk's group is computed, we consider as an easier example the group $\operatorname{Aut}(T_2)$. In this group we have the following useful property: for every two elements $g, h \in \operatorname{Aut}(T_n)$ the element $\langle\!\langle g, h \rangle\!\rangle$ is also a member of the group. This is only true up to finite index in the Grigorchuk group and will produce extra complications there.

Proposition 3.1. *The commutator width of* $\operatorname{Aut}(T_2)$ *is* 1.

For the proof we need a small observation:

Lemma 3.2. *Let H be a self-similar group acting on a binary tree. If $g \in H'$ then $g@2 \cdot g@1 \in H'$.*

Proof. It suffices to consider a commutator $g = [g_1, g_2]$ in H'. Then $g@2 \cdot g@1$ is the product, in some order, of all eight terms $(g_i@j)^\epsilon$ for all $i, j \in \{1, 2\}$ and $\epsilon \in \{\pm 1\}$. □

Proof of Proposition 3.1. Given any element $g \in \operatorname{Aut}(T_2)'$ we consider the equation $[X, Y]g$. If in it we replace the variable X by $\langle\!\langle X_1, X_2 \rangle\!\rangle$ and Y by $\langle\!\langle Y_1, Y_2 \rangle\!\rangle(1,2)$ we obtain $\langle\!\langle X_1^{-1} Y_2^{-1} X_2 Y_2 g@1, X_2^{-1} Y_1^{-1} X_1 Y_1 g@2 \rangle\!\rangle$. Therefore, $[X, Y]g$ is solvable if the system of equations $\{X_2^{-1} Y_2^{-1} X_2 Y_2 g@1,\ X_1^{-1} Y_1^{-1} X_1 Y_1 g@2\}$ is solvable. We apply the $\operatorname{Aut}(T_2)$-homomorphism $X_1 \mapsto X_1, X_2 \mapsto Y_1^{-1} X_1 Y_1 g@2, Y_i \mapsto Y_i$ to eliminate one equation and one variable.
Thus the solvability of the constrained equation $([X, Y]g, (X \mapsto \mathbb{1}, Y \mapsto (1,2)))$ follows from the solvability of $X_1^{-1} Y_2^{-1} Y_1^{-1} X_1 Y_1 (g@2) Y_2 (g@1)$ which is under the normal form $\operatorname{Aut}(T_2)$-isomorphism $Y_1 \mapsto Y_1 Y_2^{-1}$ equivalent to the solvability of the

equation $[X_1, Y_1](g@2)^{Y_2} g@1$. After choosing $Y_2 = \mathbb{1}$ we are again in the original situation since $g@2g@1 \in H'$.

This allows us to recursively define a solution s for the equation $[X, Y]g$ as follows:

$$s(X) = \langle\!\langle a_1, b_1^{-1} a_1 b_1 g@2 \rangle\!\rangle, \qquad s(Y) = \langle\!\langle b_1, \mathbb{1} \rangle\!\rangle(1,2), \qquad c_1 = g@2 \cdot g@1,$$

and for all $i \geq 1$

$$a_i = \langle\!\langle a_{i+1}, b_{i+1}^{-1} a_{i+1} b_{i+1} c_i@2 \rangle\!\rangle, \qquad b_i = \langle\!\langle b_{i+1}, \mathbb{1} \rangle\!\rangle(1,2), \quad c_{i+1} = c_i@2 \cdot c_i@1.$$

Note that the elements $a_i, b_i \in \mathrm{Aut}(T_2)$ are well-defined, although they are constructed recursively out of the a_j, b_j for *larger* j. Indeed, if one considers the recursions above for $i \in \{1, \dots, n\}$ and sets $a_{n+1} = b_{n+1} = \mathbb{1}$, one defines in this manner elements $a_1^{(n)}, b_1^{(n)} \in \mathrm{Aut}(T_2)$ which form Cauchy sequences, and therefore have well-defined limits $a_1 = \lim a_1^{(n)}$ and $b_1 = \lim b_1^{(n)}$. □

4. The first Grigorchuk Group

The first Grigorchuk group [**Gri80**] is a finitely generated self-similar group acting faithfully on the binary rooted tree, with generators

$$a = \langle\!\langle \mathbb{1}, \mathbb{1} \rangle\!\rangle(1,2), \quad b = \langle\!\langle a, c \rangle\!\rangle, \quad c = \langle\!\langle a, d \rangle\!\rangle, \quad d = \langle\!\langle \mathbb{1}, b \rangle\!\rangle.$$

Some useful identities are

$$a^2 = b^2 = c^2 = d^2 = bcd = \mathbb{1},$$
$$b^a = \langle\!\langle c, a \rangle\!\rangle, c^a = \langle\!\langle d, a \rangle\!\rangle, d^a = \langle\!\langle b, \mathbb{1} \rangle\!\rangle,$$
$$(ad)^4 = (ac)^8 = (ab)^{16} = \mathbb{1}.$$

Definition 4.1 (Regular branched group). A self-similar group Γ is called *regular branched* if it has a finite-index subgroup $K \leq \Gamma$ such that $K^{\times n} \leq \Psi(K)$.

Lemma 4.2 ([**Roz93**]). *The Grigorchuk group is regular branched with branching subgroup*

$$K := \left\langle (ab)^2 \right\rangle^G = \left\langle (ab)^2, (bada)^2, (abad)^2 \right\rangle.$$

The quotient $Q := G/K$ has order 16. □

For an equation $\mathcal{E} \in F_{\mathcal{X}} * G$, recall that $\mathrm{Stab}(\mathcal{E})$ denotes the group of G-automorphisms of $\mathcal{E}$.

Denote by U_n the subgroup of $\mathrm{Stab}(R_n)$ generated by the following automorphisms of F_{2n}:

$$\begin{array}{lll}
\varphi_i\colon & X_i \mapsto X_{i-1}X_i, \text{ others fixed} & \text{for } i = 2, 4, \dots, 2n, \\
\varphi_i\colon & X_i \mapsto X_{i+1}X_i, \text{ others fixed} & \text{for } i = 1, 3, \dots, 2n-1, \\
 & X_i \mapsto X_{i+1}X_{i+2}^{-1}X_i, & \\
\psi_i\colon & X_{i+1} \mapsto X_{i+1}X_{i+2}^{-1}X_{i+1}X_{i+2}X_{i+1}^{-1}, & \text{for } i = 1, 3, \dots, 2n-3 \\
 & X_{i+2} \mapsto X_{i+1}X_{i+2}^{-1}X_{i+2}X_{i+2}X_{i+1}^{-1}, & \\
 & X_{i+3} \mapsto X_{i+1}X_{i+2}^{-1}X_{i+3}, \text{ others fixed} &
\end{array}$$

Remark 4.3. In fact, we have $U_n = \mathrm{Stab}(R_n)$ though formally we do not need the equality. Due to classical results of Dehn–Nielsen, $\mathrm{Stab}(R_n)$ is isomorphic to the mapping class groups $M(n, 0)$ of the closed orientable surface of genus n. It can be checked that the automorphisms φ_i and ψ_i represent the Humphries generators of $M(n,0)$. For details on mapping class groups, see for example [**FM11**].

LEMMA 4.4 ([**LMU16**]). *Given $n \in \mathbb{N}$ and a homomorphism $\gamma\colon F_{\mathcal{X}} \to Q$ with* $\operatorname{supp}(\gamma) \subset \langle X_1, \dots, X_{2n}\rangle$ *there is an element $\varphi \in U_n < \operatorname{Aut}(F_{\mathcal{X}})$ such that* $\operatorname{supp}(\gamma \circ \varphi) \in \langle X_1, \dots, X_5\rangle$. □

LEMMA 4.5. *Identify the set $\{\gamma\colon F_{\mathcal{X}} \to Q \mid \operatorname{supp}(\gamma) \subset \langle X_1, \dots, X_n\rangle\}$ with Q^n. Then*

$$\left|Q^{2n}/U_n\right| \leq 90 \textit{ for all } n.$$

PROOF. Note that according to our identification we have $Q^m \subset Q^n$ for $m < n$. By Lemma 4.4 every orbit Q^{2n}/U_n has a representative in Q^5. Let $\mathfrak{R}_n$ denote a set of representatives of Q^{2n}/U_n in Q^5. Since $U_n \subset U_{n+1}$ we can assume that $\mathfrak{R}_{n+1} \subset \mathfrak{R}_n$ for $n \geq 3$.
Direct computation shows that $|\mathfrak{R}_3| = 90$, see Section 6.3. □

REMARK 4.6. In fact we have $\left|Q^{2n}/U_n\right| = 90$ for all $n \geq 3$. To prove this one can show by direct computation that $\mathfrak{R}_3 = \mathfrak{R}_4 = \mathfrak{R}_5$ and then show for all $\theta \in U_n$, $n \geq 6$ and $\gamma, \gamma' \in \mathfrak{R}_3, \gamma \neq \gamma'$ that $\gamma \circ \theta \neq \gamma'$.

NOTATION 4.7 ($\mathfrak{R}$, reduced constraint). Lemmas 4.4 and 4.5 imply that there is a set of 90 homomorphisms $\gamma\colon F_{\mathcal{X}} \to Q$ with $\operatorname{supp}(\gamma) \subset \langle X_1, \dots, X_5\rangle$ that is a representative system of the orbits Q^{2n}/U_n for each $n \geq 3$. Fix such a set $\mathfrak{R}$ and for $\gamma\colon F_{\mathcal{X}} \to Q$ with finite support (say $X_1, \dots, X_{2n}$) denote by φ_γ the G-homomorphism in U_n such that $\gamma \circ \varphi_\gamma \in \mathfrak{R}$.
The element $\gamma \circ \varphi_\gamma$ will be called a *reduced constraint*.

LEMMA 4.8. *The solvability of a constrained equation $(R_n g, \gamma)$ is equivalent to the solvability of $(R_n g, \gamma \circ \varphi_\gamma)$.*

PROOF. If s is a solution for $(R_n g, \gamma)$ then $s \circ \varphi_\gamma$ is a solution for $(R_n g, \gamma \circ \varphi_\gamma)$. □

DEFINITION 4.9 (Branch structure [**Bar13**]). A *branch structure* for a group $G \hookrightarrow G \wr S_n$ consists of

(1) a branching subgroup $K \trianglelefteq G$ of finite index;
(2) the corresponding quotient $Q = G/K$ and the factor homomorphism $\pi\colon G \to Q$;
(3) a group $Q_1 \subset Q \wr S_n$ such that $\langle\!\langle q_1, \dots, q_n\rangle\!\rangle\sigma \in Q_1$ if and only if $\langle\!\langle g_1, \dots, g_n\rangle\!\rangle\sigma \in G$ for all $g_i \in \pi^{-1}(q_i)$;
(4) a map $\omega\colon Q_1 \to Q$ with the following property: if $g = \langle\!\langle g_1, \dots, g_n\rangle\!\rangle\sigma \in G$ then $\omega(\langle\!\langle \pi(g_1), \dots, \pi(g_n)\rangle\!\rangle\sigma) = \pi(g)$.

All regular branched groups have a branch structure (see [**Bar13**, Remark after Definition 5.1]). We will from now on fix such a structure for G and take the group K defined in Lemma 4.2 as branching subgroup and denote by Q the factor group with natural homomorphism $\pi\colon G \to G/K = Q$.

REMARK 4.10. The branch structure of G is included in the FR package and can be computed by the method `BranchStructure(GrigorchukGroup)`.

4.1. Good Pairs. It is not true that for every $g \in G'$ and every constraint γ there is an $n \in \mathbb{N}$ such that the constrained equation $(R_n g, \gamma)$ is solvable. For example

$$\Big(R_n(ab)^2, (\gamma\colon X_i \mapsto \mathbb{1}\ \forall i)\Big)$$

is not solvable for any n because $(ab)^2 \notin K'$. This motivates the following definition.

Definition 4.11 (Good pair). Given $g \in G'$ and $\gamma \in \mathfrak{R}$, the tuple (g, γ) is called a *good pair* if $(R_n g, \gamma)$ is solvable for some $n \in \mathbb{N}$.

Lemma 4.12. *Denote by*

$$\tau\colon G \to G/K' \quad \text{and} \quad \rho\colon G/K' \to (G/K')/(K/K') \simeq G/K$$

the natural projections.
The pair (g, γ) is a good pair if and only if there is a solution $s\colon F_{\mathcal{X}} \to G/K'$ for $R_3\tau(g)$ with $s(X_i) \in \rho^{-1}(\gamma(X_i))$.

Proof. If (g, γ) is a good pair and s a solution for $(R_n g, \gamma)$ then $s(X_i) \in K$ for $i \geq 6$, so $s(R_n) = s(R_3) \cdot k'$ for some $k' \in K'$. Therefore there is a solution $\tau \circ s$ for $R_3\tau(g)$ with $s(X_i) = \gamma(X_i)$.
On the other hand if there is a solution $s\colon F_{\mathcal{X}} \to G/K'$ for $R_3\tau(g)$ with for each $s(X_i) \in \rho^{-1}(\gamma(X_i))$ then for $g_i \in \tau^{-1}(s(X_i))$ there is some $k' \in K'$ such that $R_3(g_1, \dots, g_6)k'g = \mathbb{1}$ and so (g, γ) is a good pair. □

The previous lemma shows that the question whether (g, γ) is a good pair depends only on the image of g in G/K'. For $q \in Q$, we call (q, γ) a *good pair* if (g, γ) is a good pair for one (and hence all) preimages of q under τ.

Corollary 4.13. *The following are equivalent:*

(a) K has finite commutator width;
(b) there is an $n \in \mathbb{N}$ such that $(R_n g, \gamma)$ is solvable for all good pairs (g, γ) with $g \in G'$ and $\gamma \in \mathfrak{R}$.

Proof. (b)⇒(a): if $k \in K'$ then $(k, \mathbb{1})$ is a good pair, so $(R_n k, \mathbb{1})$ is solvable in G; and the constraints ensures that it is solvable in K. Therefore the commutator width of K is at most n.
(a)⇒(b): if (g, γ) is a good pair there is an $m' \in \mathbb{N}$ and a solution s for $(R_{m'} g, \gamma)$. As $\pi(s(X_i)) = \mathbb{1}$ for all $i \geq 6$ there is $k \in K'$ such that s is a solution for $(R_3 k g, \gamma)$. By (a) there is an m such that all k can be written as product of m commutators of elements of K and therefore there is a solution for $(R_{m+3} g, \gamma)$. We may take $n = m + 3$. □

We study now more carefully the quotients G/K, G/K' and $G/(K \times K)$.

Lemma 4.14. *Let us write $k_1 := (ab)^2, k_2 := \langle\!\langle \mathbb{1}, k_1 \rangle\!\rangle = (abad)^2$ and $k_3 := \langle\!\langle k_1, \mathbb{1} \rangle\!\rangle = (bada)^2$. Then*

$$\begin{aligned}
G' &= \left\langle k_1, k_2, k_3, (ad)^2 \right\rangle, \\
K &= \langle k_1, k_2, k_3 \rangle, \\
K \times K &= \{\langle\!\langle k, k' \rangle\!\rangle \mid k, k' \in K\} \\
&= \left\langle k_2, k_3, [k_1, k_2], [k_1, k_3], [k_1^{-1}, k_2], [k_1^{-1}, k_3] \right\rangle, \\
K' &= \langle [k_1, k_2] \rangle^G \\
&= \left\langle [k_2, k_1], [k_1, k_2^{-1}], [k_2, k_1]^{k_2}, [k_1^{-1}, k_2], [k_2, k_1]^{k_1}, [k_2^{-1}, k_1^{-1}] \right\rangle^{\{\mathbb{1}, a\}}
\end{aligned}$$

Furthermore these groups form a tower with indices

$$[G : G'] = 8, \qquad [G' : K] = 2, \qquad [K : K \times K] = 4, \qquad [K \times K : K'] = 16.$$

PROOF. The chain of indices is shown for example in [**BGŠ03**] and the generating sets can be verified using the GAP standard methods `NormalClosure` and `Index`. □

4.2. Succeeding pairs.

DEFINITION 4.15 ($\mathfrak{R}_{\mathrm{act}}$, active constraints). We define the activity $\mathrm{act}(q)$ of an element $q \in Q$ as the activity of an arbitrary element of $\pi^{-1}(q)$. This is well defined since all elements of K have trivial activity.
Consider a constraint $\gamma\colon F_{\mathcal{X}} \to Q$. Define $\mathrm{act}(\gamma)\colon F_{\mathcal{X}} \to C_2$ by $X \mapsto \mathrm{act}(\gamma(X))$. Denote by $\mathfrak{R}_{\mathrm{act}}$ the reduced constraints in $\mathfrak{R}$ that have a nontrivial activity.

LEMMA 4.16. *For each $q \in G'/K'$ there is $\gamma \in \mathfrak{R}_{\mathrm{act}}$ such that (q, γ) is a good pair.*

PROOF. This is a finite problem which can be checked in GAP with the function `verifyLemmaExistGoodConstraints`. For more details see Section 6.1. □

We will now give a procedure to start with a constrained equation say of class $\mathcal{O}_{n,1}$ and result with an equations of class $\mathcal{O}_{2n-1,1}$ and a a set of constraints such that the solvability of any of the later constrained equations implies the solvability of the original one.
Instead of an infinitely generated free group $F_{\mathcal{X}}$ we can restrict ourselves to a finite set $\mathcal{X}$ of order $2n$ for the variables of the original equation and another set $\mathcal{Y}$ for the variables of the resulting equation. For fixed n we notate the free groups $F_{\mathcal{X}}, F_{\mathcal{Y}}$ and $F_{\mathcal{Y}'}$ on the following generating sets:

$$\mathcal{X} = \{X_\ell \mid 1 \leq \ell \leq 2n\}, \quad \mathcal{Y} = \{Y_{\ell,i} \mid 1 \leq \ell \leq 2n,\ i = 1,2\}, \quad \mathcal{Y}' = \mathcal{Y} \setminus \{Y_{6,1}, Y_{6,2}\}.$$

Denote by S the set $\{\mathbb{1}, a, b, c, d, ab, ad, ba\} \subset G$. We will define for all $q \in G'/K'$ a map Γ^q which for any $n \geq 3$ maps a reduced constraint $\gamma \in \mathfrak{R}_{\mathrm{act}}$, say $\gamma\colon F_{\mathcal{X}} \to Q$, to a set of constraints $\gamma'\colon F_{\mathcal{Y}} \to Q$ with the following property:

(*) There is $x \in S$ with $\gamma'(Y_{6,1}) = \pi(x)$, such that for all $g \in G'$ with $\tau(g) = q$ the solvability of the constrained equation $(R_{2n-1}(g@2)^x \cdot g@1, \gamma'|_{F_{\mathcal{Y}'}})$ implies the solvability of $(R_n g, \gamma)$.

We will define this map in several steps and afterwards show that for all good pairs (q, γ) and all g such that $\tau(g) = q$ there is some constraint $\gamma' \in \Gamma^q(\gamma)$ such that $((g@2)^x \cdot g@1, \gamma'|_{F_{\mathcal{Y}'}})$ is a good pair.
For the first step we take the branching structure (K, Q, π, Q_1, ω) of the Grigorchuk group as before and complete the set S to a transversal S' of G/K. Denote by $\mathrm{rep}\colon Q \to S'$ the map such that $\pi(\mathrm{rep}(q)) = q$.

$$\Gamma_1^n(\gamma) = \left\{ \gamma'\colon F_{\mathcal{Y}} \to Q \;\middle|\; \begin{array}{l} \langle\!\langle \gamma'(Y_{\ell,1}, \gamma'(Y_{\ell,2}) \rangle\!\rangle \in \omega^{-1}(\gamma(X_\ell)), \\ \gamma'(Y_{k,i}) = \mathbb{1},\ 1 \leq \ell \leq 6,\ k > 6,\ i = 1,2 \end{array} \right\}.$$

For some formal equalities for equations in G we will need two auxiliary free groups $F_{\mathcal{G}} = \langle \mathfrak{g} \rangle$, $F_{\mathcal{H}} = \langle \mathfrak{g}_1, \mathfrak{g}_2 \rangle$, and define homomorphisms

$$\Phi_\gamma\colon \begin{array}{rl} F_{\mathcal{X}} * F_{\mathcal{G}} & \to (F_{\mathcal{Y}} * F_{\mathcal{H}}) \wr C_2, \\ \mathfrak{g} & \mapsto \langle\!\langle \mathfrak{g}_1, \mathfrak{g}_2 \rangle\!\rangle, \\ X_i & \mapsto \langle\!\langle Y_{i,1}, Y_{i,2} \rangle\!\rangle \,\mathrm{act}(\gamma(X_i)), \end{array} \qquad \tilde{\Phi}_\gamma\colon \begin{array}{rl} F_{\mathcal{X}} * G & \to (F_{\mathcal{Y}} * G) \wr C_2, \\ g & \mapsto \Psi(g), \\ X_i & \mapsto \langle\!\langle Y_{i,1}, Y_{i,2} \rangle\!\rangle \,\mathrm{act}(\gamma(X_i)). \end{array}$$

LEMMA 4.17. *If γ is a constraint with nontrivial activity, and $\Phi_\gamma(R_n\mathfrak{g}) = \langle\!\langle w_1, w_2 \rangle\!\rangle$ then* $\mathrm{Var}(w_1) \cap \mathrm{Var}(w_2) \neq \emptyset$.

Proof. Let $\ell \in 1 \ldots 2n$ be such that $\gamma(X_\ell)$ has nontrivial activity. Then R_n contains either a factor $[X_\ell, X_k]$ or $[X_k, X_\ell]$ for another generator $X_k \neq X_\ell$. Assume without loss of generality the first case. Let σ be the activity of $\gamma(X_k)$ then $\Phi_\gamma(R_n\mathfrak{g})$ contains a factor

$$[\langle\!\langle Y_{\ell,1}, Y_{\ell,2}\rangle\!\rangle(1,2), \langle\!\langle Y_{k,1}, Y_{k,2}\rangle\!\rangle\sigma] = \begin{cases} \langle\!\langle Y_{\ell,2}^{-1}Y_{k,2}^{-1}Y_{\ell,2}Y_{k,1}, Y_{\ell,1}^{-1}Y_{k,1}^{-1}Y_{\ell,1}Y_{k,2}\rangle\!\rangle & \text{if } \sigma = \mathbb{1} \\ \langle\!\langle Y_{\ell,2}^{-1}Y_{k,1}^{-1}Y_{\ell,1}Y_{k,2}, Y_{\ell,1}^{-1}Y_{k,2}^{-1}Y_{\ell,2}Y_{k,1}\rangle\!\rangle & \text{if } \sigma = (1,2). \end{cases}$$

In both cases $Y_{k,1}, Y_{k,2} \in \operatorname{Var}(w_1) \cap \operatorname{Var}(w_2)$. □

For $q_1, q_2 \in Q$ and $n \geq 3 \in \mathbb{N}$ define

$$\Gamma_2^{q_1,q_2,n}(\gamma) = \left\{\gamma' \in \Gamma_1^n(\gamma) \;\middle|\; \varpi\colon \substack{F_{\mathcal{H}} \to Q \\ \mathfrak{g}_1 \mapsto q_1 \\ \mathfrak{g}_2 \mapsto q_2} \text{ satisfies } (\gamma' * \varpi)^2(\Phi_\gamma(R_n\mathfrak{g})) = \langle\!\langle \mathbb{1}, \mathbb{1}\rangle\!\rangle\right\}.$$

For $\gamma \in \mathfrak{R}_{\text{act}}$ denote by v and w the elements of $F_{\{Y_{1,1},\ldots,Y_{6,2}\}}$ such that $\Phi_\gamma(R_3\mathfrak{g}) = \langle\!\langle v, w\rangle\!\rangle\langle\!\langle \mathfrak{g}_1, \mathfrak{g}_2\rangle\!\rangle$. Then

$$\Phi_\gamma(R_n(X_*)\mathfrak{g}) = \langle\!\langle v, w\rangle\!\rangle\langle\!\langle R_{n-3}(Y_{7,1}, \ldots, Y_{2n,1})\mathfrak{g}_1, R_{n-3}(Y_{7,2}, \ldots, Y_{2n,2})\mathfrak{g}_2\rangle\!\rangle.$$

By Lemma 4.17 there is $Y_0 \in \mathcal{Y} \cup \mathcal{Y}^{-1}$ such that $v = v_1Y_0v_2$ and $w = w_1Y_0^{-1}w_2$. Then the $F_{\mathcal{H}}$-homomorphism

$$\ell_{Y_0,n}\colon \begin{array}{l} F_{\mathcal{Y}} * F_{\mathcal{H}} \to F_{\mathcal{Y}} * F_{\mathcal{H}}, \\ \quad Y \mapsto \begin{cases} Y & \text{if } Y \neq Y_0 \\ w_2R_{n-3}(Y_{7,2}, \ldots, Y_{2n,2})\mathfrak{g}_2w_1 & \text{if } Y = Y_0 \end{cases} \end{array}$$

maps the second coordinate of $\Phi_\gamma(R_n(X_*)\mathfrak{g})$ to $\mathbb{1}$ and the first coordinate to an equation

$$\mathcal{E} = v_1w_2R_{n-3}(Y_{7,2}, \ldots, Y_{2n,2})\mathfrak{g}_2w_1v_2R_{n-3}(Y_{7,1}, \ldots, Y_{2n,1})\mathfrak{g}_1.$$

For $\gamma' \in \Gamma_2^{q_1,q_2,n}(\gamma)$ we have $\gamma'(Y_0) = (\gamma' * \varpi)(w_2\mathfrak{g}_2w_1) = (\gamma' * \varpi)(\ell_{Y_0,n}(Y_0))$ and hence

$$\gamma' = (\gamma' * \varpi) \circ \ell_{Y_0,n} \text{ for all } \gamma' \in \Gamma_2^{q_1,q_2,n}(\gamma), \varpi\colon \mathfrak{g}_i \mapsto q_i, Y_0. \tag{1}$$

Consider the automorphisms

$$\psi_1\colon \begin{array}{l} F_{\mathcal{Y}} * F_{\mathcal{H}} \to F_{\mathcal{Y}} * F_{\mathcal{H}} \\ \quad Y_{k,1} \mapsto Y_{k,1}^{\mathfrak{g}_1^{-1}} \\ \quad Y_{k,2} \mapsto Y_{k,2}^{(\mathfrak{g}_2w_1v_1\mathfrak{g}_1)^{-1}} \\ \quad Y_{k,j} \mapsto Y_{k,j}, \end{array} \qquad \psi_2\colon \begin{array}{ll} F_{\mathcal{Y}} * F_{\mathcal{H}} \to F_{\mathcal{Y}} * F_{\mathcal{H}} & \\ \quad Y_{k,1} \mapsto Y_{k,1}^{\mathfrak{g}_2^{Y_{6,1}}\mathfrak{g}_1} & \text{for } k > 6 \\ \quad Y_{k,2} \mapsto Y_{k,2}^{\mathfrak{g}_2^{Y_{6,1}}\mathfrak{g}_1} & \text{for } k > 6 \\ \quad Y_{k,j} \mapsto Y_{k,j} & \text{for } k \leq 6,\ j = 1,2 \end{array}$$

$$\psi_3\colon \begin{array}{rll} F_{\mathcal{Y}} * F_{\mathcal{H}} \to & F_{\mathcal{Y}} * F_{\mathcal{H}} & \\ Y_{2k,1} \mapsto & Y_{n+k,2} & \text{for } k > 3 \\ Y_{2k-1,1} \mapsto & Y_{n+k,1} & \text{for } k > 3 \\ Y_{2k,2} \mapsto & Y_{3+k,2} & \text{for } k > 3 \\ Y_{2k-1,2} \mapsto & Y_{3+k,1} & \text{for } k > 3 \\ Y_{k,\ell} \mapsto & Y_{k,\ell} & \text{for } k \leq 6,\ \ell = 1,2 \end{array}$$

and note that for $\mathfrak{nf}_{\gamma,n,Y_0} := \psi_3 \circ \psi_2 \circ \mathfrak{nf}_{v_1w_2\mathfrak{g}_2w_1v_2\mathfrak{g}_1} \circ \psi_1$ we have

$$\mathfrak{nf}_{\gamma,n,Y_0}(\mathcal{E}) = R_{2n-1}(Y_{1,1}, Y_{1,2}, \ldots, \cancel{Y_{6,1}}, \cancel{Y_{6,2}}, \ldots, Y_{2n,2})\mathfrak{g}_2^{Y_{6,1}}\mathfrak{g}_1.$$

This leads to the following definition.

$$\Gamma_3^{q_1,q_2,n,Y_0}(\gamma) = \left\{\gamma' \circ \mathfrak{nf}_{\gamma,n,Y_0}^{-1}\colon F_{\mathcal{Y}} \to Q \;\middle|\; \gamma' \in \Gamma_2^{q_1,q_2,n}\right\}.$$

Note that $\mathfrak{nf}_{\gamma,n,Y_0}$ fixes the sets $\{Y_{k,\ell} \mid k > 6, \ell = 1,2\}$ and $\{Y_{k,\ell} \mid k \leq 6, \ell = 1,2\}$ and hence for $k > 6$ we have $\gamma''(Y_{k,\ell}) = \mathbb{1}$ for all $\gamma'' \in \Gamma_3^{q_1,q_2,n,Y_0}(\gamma)$ independently of q_i, n, Y_0 and γ and therefore we can naturally identify the mappings $\Gamma_3^{q_1,q_2,n,Y_0}$ and $\Gamma_3^{q_1,q_2,3,Y_0}$ for every $n \geq 3$. Note further that $\mathfrak{nf}_{\gamma,n,Y_0}(Y_0) = Y_{6,2}$.

Given $g \in G'$, $g_i = g@i$ for $i = 1,2$, an active constraint $\gamma \in \mathfrak{N}_{\mathrm{act}}$ and $\gamma'' \in \Gamma_3^{\pi(g_1),\pi(g_1),n,Y_0}(\gamma)$ then a solution for the the constrained equation

$$\mathcal{E}' = (R_{2n-1}(Y_{*,*})g_2^{\mathrm{rep}(\gamma''(Y_{6,1}))}g_1, \gamma'')$$

can be extended by the map $Y_{6,1} \mapsto \mathrm{rep}(\gamma''(Y_{6,1}))$ to a solution s' of the equation $(R_{2n-1}(Y_{*,*})g_2^{Y_{6,1}}g_1, \gamma'')$. Notate the epimorphism $i_{\mathcal{H}}\colon F_{\mathcal{H}} \to G, \mathfrak{g}_k \mapsto g_k$ and note that since $\mathfrak{nf}_{\gamma,n,Y_0}$ is an $F_{\mathcal{H}}$-homomorphism $\mathfrak{n} := (\mathbb{1} * i_{\mathcal{H}}) \circ \mathfrak{nf}_{\gamma,n,Y_0}$ maps $\mathcal{E}$ to $R_{2n-1}(Y_{*,*})g_2^{Y_{6,1}}g_1$. Moreover by (1) we have that $\gamma' := \gamma'' \circ \mathfrak{n} \in \Gamma_2^{q_1,q_2,n}(\gamma)$.

Hence the map

$$s\colon Y_{i,j} \mapsto \begin{cases} w_2 g_2 w_1 & \text{if } i,j = 6,2 \\ s' \circ \mathfrak{n}(Y_{i,j}) & \text{otherwise} \end{cases}$$

is a solution for $\big((\mathbb{1} * i_{\mathcal{H}}) \circ \Phi_\gamma(R_n\mathfrak{g}), \gamma'\big)$ and thus also for $\big(\tilde{\Phi}_\gamma(R_n g), \gamma'\big)$. By the definition of ω the element $t_i := \langle\!\langle s(Y_{i,1}), s(Y_{i,2})\rangle\!\rangle\,\mathrm{act}(\gamma(X_i))$ belongs to G for all i. Moreover since $\gamma' \in \Gamma_1^n(\gamma)$ we have $\pi(t_i) = \gamma(X_i)$. Thus the mapping $X_i \mapsto t_i$ is a solution for $(R_n g, \gamma)$.

The map $\Gamma_3^{q_1,q_2,n,Y_0}$ does depend on the choice of the variable Y_0. To remove this dependency we observe that the set of all variables $Y_0 \in \mathrm{Var}(v) \cap \mathrm{Var}(w)$ does not depend on n and define

$$\Gamma_4^{q_1,q_2}(\gamma) = \bigcup_{Y_0 \in \mathrm{Var}(v) \cap \mathrm{Var}(w)} \Gamma_3^{q_1,q_2,3,Y_0}(\gamma).$$

Note that $q_1, q_2 \in Q$ are determined by $q \in G'/K'$ in the sense that there is a map $\bar{@}i\colon G'/K' \to Q$ such that if $\tau(g) = q$ and $g_i = g@i$ then $q_i = q\bar{@}i$. This map $\bar{@}i$ is well defined since $k'@i \in K$ for all $k' \in K'$. Thus we can write $\Gamma_4^{q_1,q_2}(\gamma)$ as $\Gamma_4^q(\gamma)$, and filter out those constraints that do not fulfill the requested properties; we finally define

$$\Gamma^q(\gamma) := \left\{\gamma' \in \Gamma_4^q(\gamma) \;\middle|\mathrm{act}(\gamma') \neq \mathbb{1}, \gamma'(Y_{6,1}) \in \pi(S)\right\}. \tag{2}$$

Note that (*) holds automatically by construction.

Proposition 4.18. *For each good pair (q,γ) with $q \in G'/K'$ and $\gamma \in \mathfrak{N}_{\mathrm{act}}$ the set $\Gamma^q(\gamma)$ contains some constraint γ' such that for all g with $\tau(g) = q$ the pair $\Big((g@2)^{\mathrm{rep}(\gamma'(Y_{6,1}))} \cdot g@1, \gamma'|_{F_{\mathcal{Y}'}}\Big)$ is a good pair.*

For the proof of this proposition we need an auxiliary lemma:

Lemma 4.19. *The map*

$$\bar{p}_h\colon \begin{array}{ll} G'/K' & \to G'/K \times K \\ gK' & \mapsto \Big((g@2)^h \cdot g@1\Big)\, K \times K \end{array}$$

is well defined.

Proof. We need to show that $k@i \in K \times K$ for $i = 1,2$ and $k \in K'$. Remember the generators $k_1 = (ab)^2$, $k_2 = (abad)^2$. Then

$$[k_1, k_2] = bb^a(db^a)^2b^ab(b^ad)^2 = \langle\!\langle \mathbb{1}, cabab\rangle\!\rangle = \langle\!\langle \mathbb{1}, \langle\!\langle \mathbb{1}, dabac\rangle\!\rangle\rangle\!\rangle = \langle\!\langle \mathbb{1}, \langle\!\langle \mathbb{1}, k_2^{-1}k_1\rangle\!\rangle\rangle\!\rangle.$$

So both states of $[k_1,k_2]$ are in $K\times K$. Now take an arbitrary element $k\in K'$. There is $n\in\mathbb{N}$, $\varepsilon\in\{1,-1\}$ and $g_i\in G$ such that $k=\prod_{j=1}^n[k_1,k_2]^{\varepsilon g_j}$ and therefore

$$k@i=\prod_{j=1}^{n}\left(\left([k_1,k_2]^{\varepsilon g_j}\right)@i\right)=\prod_{j=1}^{n}\left(([k_1,k_2])@i^{g_j^{-1}}\right)^{\varepsilon g_j@i^{g_j^{-1}}}\quad\in K\times K$$

Then for $k\in K'$ we have

$$p_h(gk)=((gk)@2)^h\cdot(gk)@1=(g@2)^h\cdot(k@2)^h\cdot g@1\cdot k@1\in((g@2)^h\cdot g@1)K\times K.$$

□

Proof of Proposition 4.18. In the construction above it is clear that the sets Γ_3^{q,Y_0} and hence Γ_4^{q,Y_0} are nonempty. For the finitely many $\gamma\in\mathfrak{R}_{\text{act}}$ checking whether some of the finitely many $\gamma'\in\Gamma_4^q(\gamma)$ fulfill $\gamma'(Y_{6,1})\in\pi(S)$ and $\operatorname{act}(\gamma')\neq\mathbb{1}$ (i.e. $\gamma'\in\Gamma^q(\gamma)$) is implemented in the procedure below.

Define for $h\in G$ maps $p_h\colon G\to G$ by $g\mapsto(g@2)^h\cdot g@1$. These maps are in general not homomorphisms but by Lemma 3.2 for $g\in G'$ we have $p_h(g)\in G'$ for all $h\in G$. By Lemma 4.19 we can define the map $\bar{p}_h\colon G'/K'\to G'/(K\times K)$ and the natural homomorphism

$$\rho'\colon G'/K'\to\left(G'/K'\right)/\left(K\times K/K'\right)\simeq G'/(K\times K)$$

and now we only need to show that there is a $\gamma'\in\Gamma^q(\gamma)$ such that all preimages of $\bar{p}_{\operatorname{rep}(\gamma'(Y_{6,1}))}(q)$ under ρ' form good pairs with $\gamma'|_{F_{\mathcal{Y}'}}$. In formulas with $\mathcal{P}$ the predicate of being a good pair what needs to be checked is:

$$\forall q\in G'/K'\ \forall\gamma\in\mathfrak{R}_{\text{act}}\ \exists\gamma'\in\Gamma^q(\gamma)\ \forall r\in\rho'^{-1}(\bar{p}_{\operatorname{rep}(\gamma'(Y_{6,1}))}(q))\colon\mathcal{P}(q,\gamma)\Rightarrow\mathcal{P}(r,\gamma'|_{F_{\mathcal{Y}'}}).$$

This last formula quantifies only over finite sets, and could be implemented. It can be checked in GAP with the function `verifyPropExistsSuccessor`. □

Definition 4.20 (Succeding pair). For each $q\in G'/K'$ and $\gamma\in\mathfrak{R}_{\text{act}}$ such that (q,γ) is a good pair fix a constraint $\gamma'\in\Gamma^q(\gamma)$ and an element $x=\operatorname{rep}(\gamma'(Y_{6,1}))\in S$ with the property of Proposition 4.18.

By Lemma 4.8 we can replace $\gamma'|_{F_{\mathcal{Y}'}}$ by a reduced constraint γ'_r. For a good pair $(g,\gamma)\in G'\times\mathfrak{R}_{\text{act}}$ the *succeeding pair* is defined as $((g@2)^x g@1,\gamma'_r)$. Moreover by applying this iteratively we get the *succeeding sequence* (g_k,γ_k) of (g,γ): $(g_0,\gamma_0)=(g,\gamma)$ and (g_{k+1},γ_{k+1}) is the succeding pair of (g_k,γ_k).

The following lemma illustrates the use of the construction.

Lemma 4.21. *Let (g_k,γ_k) be the succeeding sequence of a good pair (g,γ). If $(g_i,\gamma_i)=(g_j,\gamma_j)$ for some distinct i,j then the equation $(R_n g,\gamma)$ is solvable for all $n\geq 3$.*

Proof. By (*) for any i,j with $i<j$ and any $n\geq 3$ there exists $n'>n$ such that solvability of $(R_{n'}g_j,\gamma_j)$ implies solvability of $(R_n g_i,\gamma_i)$. If $(g_i,\gamma_i)=(g_j,\gamma_j)$ then n' can be taken arbitrarily large. If (g,γ) is a good pair then (g_i,γ_i) is also a good pair by construction. We deduce the solvability of $(R_n g_i,\gamma_i)$ and hence the solvability of $(R_n g,\gamma)$. □

4.3. Product of 3 commutators. We will prove that every element $g\in G'$ is a product of three commutators by proving that all succeeding sequences (g_k,γ_k) as defined after Proposition 4.18 loop after finitely many steps. For this purpose remember the map $p_x\colon g\mapsto(g@2)^x g@1$ from the proof of Proposition 4.18. We will show that for each $g\in G'$ the sequence of sets

$$\operatorname{Suc}_1^g=\{g\},\ \operatorname{Suc}_n^g=\{p_x(h)\mid h\in\operatorname{Suc}_{n-1}^g,x\in S\}$$

stabilizes in a finite set.
In [**Bar98**] there is a choice of weights on generators which result in a length on G with good properties.

Lemma 4.22 ([**Bar98**]). *Let $\eta \approx 0.811$ be the real root of x^3+x^2+x-2 and set the weights*

$$\omega(a) = 1-\eta^3 \qquad \omega(c) = 1-\eta^2$$
$$\omega(b) = \eta^3 \qquad \omega(d) = 1-\eta$$

then

$$\begin{aligned}\eta(\omega(b)+\omega(a)) &= \omega(c)+\omega(a)\\ \eta(\omega(c)+\omega(a)) &= \omega(d)+\omega(a)\\ \eta(\omega(d)+\omega(a)) &= \omega(b).\end{aligned}$$ □

The next lemma is a small variation of a lemma in [**Bar98**].

Lemma 4.23. *Denote by ∂_ω the length on G induced by the weight ω. Then there are constants $C \in \mathbb{N}$, $\delta < 1$ such that for all $x \in S$, $g \in G$ with $\partial_\omega(g) > C$ it holds $\partial_\omega(p_x(g)) \le \delta\partial_\omega(g)$.*

Corollary 4.24. *The sequences of sets*

$$\mathrm{Suc}_1^g = \{g\},\ \mathrm{Suc}_n^g = \{p_x(h) \mid h \in \mathrm{Suc}_{n-1}^g, x \in S\}$$

stabilizes at a finite step for all $g \in G$.

Proof of Lemma (see [**Bar98**, Proposition 5]). Each element $g \in G$ can be written in a word of minimal length of the form $g = a^\varepsilon x_1 a x_2 a \dots x_n a^\zeta$ where $x_i \in \{b,c,d\}$ and $\varepsilon, \zeta \in \{0,1\}$. Denote by n_b, n_c, n_d the number of occurrences of b, c, d accordingly. Then

$$\begin{aligned}\partial_\omega(g) &= (n-1+\varepsilon+\zeta)\omega(a) + n_b\omega(b) + n_c\omega(c) + n_d\omega(d)\\ \partial_\omega(p_x(g)) &\le (n_b+n_c)\omega(a) + n_b\omega(c) + n_c\omega(d) + n_d\omega(b) + 2\partial_\omega(x)\\ &= \eta\left((n_b+n_c+n_d)\omega(a) + n_b\omega(b) + n_c\omega(c) + n_d\omega(d)\right) + 2\partial_\omega(x)\\ &= \eta(\partial_\omega(g) + (1-\varepsilon-\zeta)\omega(a)) + 2\partial_\omega(x)\\ &\le \eta(\partial_\omega(g)+\omega(a)) + 2(\omega(a)+\omega(b))\\ &= \eta(\partial_\omega(g)+\omega(a)) + 2.\end{aligned}$$

Thus the length of $p_x(g)$ growths with a linear factor smaller than 1 in terms of the length of g. Therefore the claim holds. For instance one could take $\delta = 0.86$ and $C = 50$ or $\delta = 0.96$ and $C = 16$. □

This completes the proof of the following proposition:

Proposition 4.25. *If $n \ge 3$ and (g,γ) is a good pair with active constraint γ with $\mathrm{supp}(\gamma) \subset \{X_1, \dots, X_{2n}\}$ then the constrained equation $(R_n(X_1,\dots,X_{2n})g, \gamma)$ is solvable.* □

Corollary 4.26. *The Grigorchuk group G has commutator width at most 3.*

Proof. This is a direct consequence of the proposition and Lemma 4.16. □

4.4. Product of 2 commutators. The case of products of two commutators can be reduced to the case of three commutators by using the same method as before.
We can compute the orbits of Q^4/U_2 and take a representative system denoted by $\mathfrak{R}^4$. It turns out that there are 86 orbits and we can check that there are again enough active constraints:

LEMMA 4.27. *For each $q \in G'/K'$ there is $\gamma \in \mathfrak{R}^4_{\text{act}}$ such that (q, γ) is a good pair.*

PROOF. This can be checked in GAP with the function `verifyLemmaExistGoodGammasForRed4`. □

To formulate an analog of Proposition 4.18 we literally transfer the definition of the function Γ^q to the case $n = 2$. Denote the new function $\Gamma^{q,2}$. For a constraint $\gamma\colon F_{\{X_1,\ldots,X_4\}} \to Q$ with nontrivial activity it produces a set $\Gamma^{q,2}(\gamma)$ of constraints $\gamma'\colon F_{\{Y_{1,1},\ldots,Y_{4,2}\}} \to Q$.

PROPOSITION 4.28. *For each good pair (q, γ) with $q \in G'/K'$ and $\gamma \in \mathfrak{R}^4_{\text{act}}$ the set $\Gamma^{q,2}(\gamma)$ contains some active constraint γ' such that for all g with $\tau(g) = q$ the pair $\Big((g@2)^{\text{rep}(\gamma'(Y_{4,1}))} \cdot g@1, \gamma'|_{\mathcal{F}_{\{Y_{1,1},\ldots Y_{3,2}\}}}\Big)$ is a good pair.*

PROOF. The proof is the same as for Proposition 4.18. The corresponding formula which needs to be checked is

$$\forall q \in G'/K' \; \forall \gamma \in \mathfrak{R}^4_{\text{act}} \; \exists \gamma' \in \Gamma^q(\gamma) \; \forall r \in \rho'^{-1}(\bar{p}_{\text{rep}(\gamma'(Y_{4,1}))}(q))\colon \mathcal{P}(q, \gamma) \Rightarrow \mathcal{P}(r, \gamma').$$

This can be checked in GAP with the function `verifyPropExistsSuccessor`. □

The resulting succeeding pairs are now equations of genus 3 with an active constraint. Those are already shown to be solvable by Proposition 4.25. Hence we have the following corollary which improves Proposition 4.25:

COROLLARY 4.29. *If $n \geq 2$ and (g, γ) is a good pair with active constraint γ with $\operatorname{supp}(\gamma) \subset \{X_1, \ldots, X_n\}$ then the constrained equation $(R_n(X_1, \ldots, X_{2n})g, \gamma)$ is solvable.*

Together with Lemma 4.27 this proves the first part of Theorem A.

COROLLARY 4.30. *K has commutator width at most 2.*

PROOF. To show that K has commutator width at most 2 it is sufficient to show that the constrained equations $(R_2 g, \mathbb{1})$ have solutions for all $g \in K'$. Since $\mathbb{1}$ has trivial activity one cannot directly apply Proposition 4.25. However one can check that all pairs $(h, \gamma_1), (f, \gamma_2)$ such that $g = \langle\!\langle h, f \rangle\!\rangle$ and $\gamma_1 = (\mathbb{1}, \mathbb{1}, \pi(bad), \mathbb{1})$, $\gamma_2 = (\mathbb{1}, \mathbb{1}, \mathbb{1}, \pi(ca))$ are good pairs with active constraints and hence admit solutions $s_1, s_2\colon F_4 \to G$.
We can then define the map $s\colon F_4 \to G, X_i \mapsto \langle\!\langle s_1(X_i), s_2(X_i) \rangle\!\rangle$; it is a solution for $R_2 g$ and $s(X_i) \in K$ for all $i = 1, \ldots, 4$. Therefore the commutator width of K is at most 2.
This can be checked in GAP with the function `verifyCorollaryFiniteCWK`. □

4.5. Not every element is a commutator. The procedure used to prove that every element is a product of two commutators can not be used to prove that every element is a commutator since for equations of genus 1 the genus does not increase by passing to a succeeding pair.

In fact not every element $g \in G'$ is a commutator. This can be seen by considering finite quotients. A commutator in the group would be also a commutator in the quotient group.
We will define an epimorphism to a finite group with commutator width 2.
Analogously to the construction of $\Psi\colon \operatorname{Aut}(T_n) \to \operatorname{Aut}(T_n) \wr S_n$ we can define a homomorphism $\Psi_n\colon G \to G \wr_{2^n} (G/\operatorname{Stab}_G(n))$ by mapping an element g to its actions on the subtrees with root in level n and the activity on th n-th level of the tree.
Consider the following epimorphism:

$$\text{germ}\colon \quad \begin{aligned} G &\to \langle b, c, d\rangle \simeq C_2 \times C_2, \\ a &\mapsto \mathbb{1}, \\ b, c, d &\mapsto b, c, d. \end{aligned}$$

It extends to an epimorphism $\text{germ}_n\colon G \wr_{2^n} G/\operatorname{Stab}_G(n) \to \text{germ}(G) \wr_{2^n} G/\operatorname{Stab}_G(n)$.
We will call the image $\text{germ}(G) =: G_0$ the 0-th *germgroup* and furthermore $G_n := \text{germ}_n \circ \Psi_n(G)$ the n-th *germgroup*.
The 4-th germgroup of the Grigorchuk group has order 2^{26} and has commutator width 2. If the FR package is present this group can be constructed in GAP with the following command.

```
gap> Range(EpimorphismGermGroup(GrigorchukGroup,4))
```

There is an element in the commutator subgroup of this germgroup which is not a commutator. This element is part of the precomputed data and can be accessed in GAP as `PCD.nonCommutatorGermGroup4`. For the computation of this element we used the character table of G_4. For more details see Section 6.2.
A corresponding preimage in G with a minimal number of states is the automaton shown in Figure 13. The construction of the element can be found in the file `gap/precomputeNonCommutator.g`. With the representation in standard generators it is easy to show using the homomorphism π on the generators that this element is even a member of K. This finishes the proof of Theorem A.

4.6. Bounded conjugacy width.

In [**Fin14**] it is proven that G has finite bounded conjugacy width. Here we give an explicit bound on this width.

PROPOSITION 4.31. *Let g be in G'. Then the equation*

$$a^{X_1}a^{X_2}a^{X_3}a^{X_4}a^{X_5}ag = \mathbb{1}$$

is solvable in G.

PROOF. We need to solve the constrained equation $(\mathcal{E} = a^{X_1}a^{X_2}a^{X_3}a^{X_4}a^{X_5}ag, \gamma)$ for some constraint γ. Independently of the chosen constraint, replacement of the variable X_i by $\langle\!\langle Y_i, Z_i\rangle\!\rangle \operatorname{act}(\gamma(X_i))$ leads after normalization to an equivalent equation $R_2(g@2)(g@1)$. Similarly to the construction of Γ^q in the previous section, one can find for each $q \in G'/K'$ a constraint γ such that $\gamma(\mathcal{E}^{\mathbb{1}*\pi}) = \mathbb{1}$ and $\gamma' \in \Gamma_1(\gamma)$ such that for all $g \in \pi^{-1}(q)$ the pairs $(g@2g@1, \gamma')$ are good pairs and γ' is an active constraint. Therefore the constrained equation $(R_2(g@2)(g@1), \gamma')$ is solvable by Corollary 4.29 for each $g \in G'$ and hence the equation $a^{X_1}a^{X_2}a^{X_3}a^{X_4}a^{X_5}ag$. This can be checked in GAP with the function `verifyExistGoodConjugacyConstraints`. □

LEMMA 4.32. *There exits an element $g \in G'$ such that the equation*

$$a^{X_1}a^{X_2}a^{X_3}ag = \mathbb{1}$$

is not solvable.

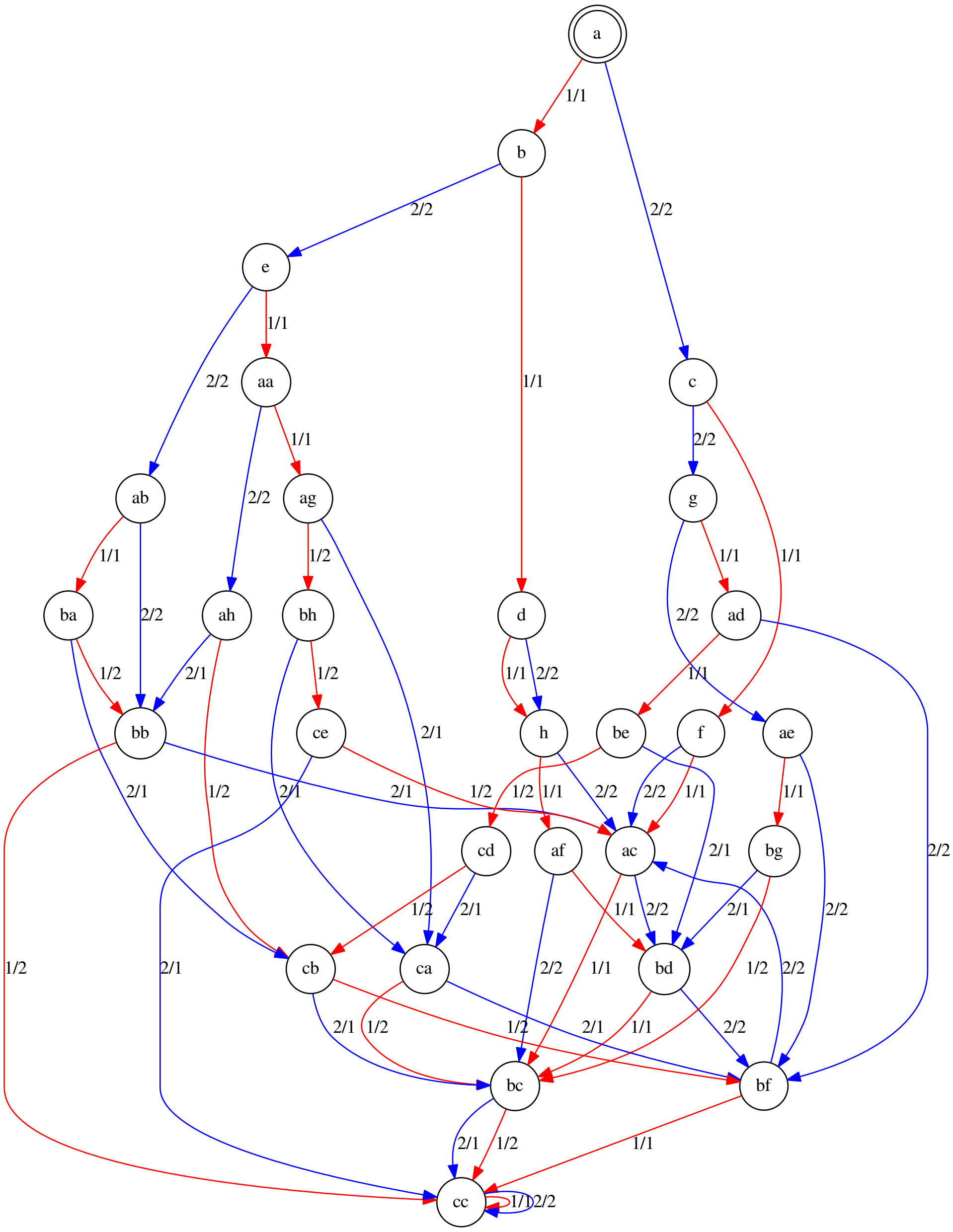


FIGURE 13. Element of the derived subgroup of the Grigorchuk group which is not a commutator. In standard generators:
$(acabacad)^3acab(ac)^2(acabacad)^2(acab)^3acadacab(ac)^2$
$(acabacad)^2(acabacadacab(ac)^3abacad(acab)^2)^5acabacadacab(ac)^2$
$(acabacad)^2(acabacadac)^2(abac)^3adacab(ac)^2(acabacad)^3$
$acab(ac)^2(acab(ac)^3abacad)^2acabacad((acabacadacab(ac)^2)^2$
$acabacad(acab)^3acadacab(ac)^2)^2((acabacad)^3acab)^2$
$acab(acabacad)^2acab(ac)^2(acabacad)^3acab(ac)^3aba$

Proof. As before independently of the activities of a possible constraint γ and of the element $g \in G'$ the normalform of $\tilde{\Phi}_\gamma(a^{X_1}a^{X_2}a^{X_3}ag)$ turns out to be $R_1(g@2)g@1$. So all there is to prove is that there is an element $h \in K$ where the products of states $h@2 \cdot h@1$ is not a commutator.
The element g displayed in Figure 13 provides such an element. It can easily be verified that $\langle\!\langle \pi(cag), \pi(ac) \rangle\!\rangle \in Q_1$ and $\omega(\langle\!\langle \pi(cag), \pi(ac) \rangle\!\rangle) = \mathbb{1}$. Thus by the properties of the branch structure we have $\langle\!\langle \pi(cag), \pi(ac) \rangle\!\rangle \in K < G'$. □

This finishes the proof of Corollary B.

Definition 4.33 (Conjugacy width [**Fin14**]). The *conjugacy width* of a group G with respect to a generating set S is the smallest number $N \in \mathbb{N}$ such that every element $g \in G$ is a product of at most N conjugates of generators $s \in S$.

Corollary 4.34. *The Grigorchuk group G with generating set $\{a, b, c, d\}$ has conjugacy width at most* 8.

Proof. The following set T is a transversal of G/G':

$$T = \{\mathbb{1}, a, d^a a, d^a, b, ab^a, ca^d, bd^a\}.$$

Therefore, every element $g \in G$ can be written as $g = th$ with $t \in T$ and $h \in G'$. As every element of G' is a product of at most 6 conjugates of a this proves the claim. □

5. Proof of Theorem C

We will prove the statement first for finite-index subgroups.

Proposition 5.1. *All finite-index subgroups $H \leq G$ have finite commutator width.*

Proof. Note that from Corollary 4.30 it follows that $K \times K$ and furthermore $K^{\times n}$ have commutator width 2.
Let H be a subgroup of finite index. Since G has the congruence subgroup property ([**BG02**]) we can find a nontrivial normal subgroup $N = \mathrm{Stab}_G(m) < H$ for some $m \in \mathbb{N}$. Furthermore since K is inactive we have

$$K < \mathrm{Stab}_G(1), \qquad K \times K < \mathrm{Stab}_G(2), \qquad K^{\times 4} < \mathrm{Stab}_G(3).$$

Furthermore we have $\mathrm{Stab}_G(n) = \mathrm{Stab}_G(3)^{\times 2^{n-3}}$ for $n \geq 4$ and hence for every subgroup H of finite index there is an n such that $K^{\times 2^n} \leq H$.
Since K' has finite index in K by Lemma 4.14, the index in $[H, H]$ of $[K^{\times 2^n}, K^{\times 2^n}]$ is finite. Taking a transversal T of $[H, H]/[K^{\times 2^n}, K^{\times 2^n}]$ we can find $m \in \mathbb{N}$ such that every element in T is a product of at most m commutators in H. We can thus write each element $h \in [H, H]$ as product kt with $k \in K^{\times 2^n}$, $t \in T$ and thus as a product of at most $2 + m$ commutators. □

Proposition 5.2. *All finitely generated subgroups $H \leq G$ are of finite commutator width.*

Proof. Every infinite finitely generated subgroup of G is abstractly commensurable to G, see [**GW03**, Theorem 1].
This, by definition, means that every infinite finitely generated subgroup $H \leq G$ contains a finite-index subgroup which is isomorphic to a finite-index subgroup of G and hence by Proposition 5.1 has finite commutator width. □

To show that there cannot be a bound on the commutator width of subgroups we need some auxiliary results. They are well-known, but since we could not find an original reference we will sketch their proofs here.

Proposition 5.3.

(1) For all $n \in \mathbb{N}$ there is a finite 2-group of commutator width at least n.
(2) K contains every finite 2-group as a subgroup.
(3) Every finite 2-group is a quotient of two finite-index subgroups of G.

Proof.

(1) Consider the groups $\Gamma_n = F_n/\langle \gamma_3(F_n), x_1^2, \dots, x_n^2\rangle$. These are extensions of C_2^n by $C_2^{\binom{n}{2}}$ and are class 2-nilpotent 2-groups. The derived subgroup is hence of order $2^{\binom{n}{2}}$. Let T be a transversal of Γ_n/Γ_n'. Thus T is of order 2^n and for $x, y \in \Gamma_n$ there are $t, s \in T$ and $x', y' \in \Gamma'$ such that every commutator $[x, y] = [tx', sy'] = [t, s]$. Therefore there are at most $\binom{2^n}{2}$ commutators.

This means there are at most $\binom{2^n}{2}^m \leq 2^{(2n-1)m}$ products of m commutators but the size of Γ_n' is $2^{\binom{n}{2}} \geq 2^{\frac{n^2}{4}}$ and hence the commutator width of Γ_{8m} is at least m.

(2) K contains for each n the n-fold iterated wreath product $W_n(C_2) = C_2 \wr \cdots \wr C_2$. This can be shown by finding finitely many vertices of the tree T_2 which define a (spaced out) copy of the finite binary rooted tree with n levels T_2^n, and finding elements $k_i \in K$ such that $\langle k_i\rangle$ acts on T_2^n like the full group of automorphisms $\mathrm{Aut}(T_2^n) \simeq W_n(C_2)$.

Then since $W_n(C_2)$ is a Sylow 2-subgroup of S_{2^n} every finite 2-group is a subgroup of $W_n(C_2)$ for some n, and hence a subgroup of K.

(3) Consider again some the vertices of T_2 which define a copy of the finite tree T_2^n on which a subgroup of K acts like $W_n(C_2)$. If we take m large enough such that all these vertices are above the m-th level we can find a copy of $W_n(C_2)$ inside $G/\operatorname{Stab}_G(m)$. □

In the following theorem we summarize our results for the commutator width of the Grigorchuk group.

Theorem 5.4.

(1) G and its branching subgroup K have commutator width 2.
(2) All finitely generated subgroups $H \leq G$ have finite commutator width.
(3) The commutator width of subgroups is unbounded even among finite-index subgroups.
(4) There is a subgroup of G with infinite commutator width.

Proof. Statements (1) and (2) are proven in Theorem A and Proposition 5.2. For every $n \in \mathbb{N}$ we can find two groups H_1, H_2 of finite index in G such that H_1/H_2 has commutator width at least n. Then H_1 has commutator width at least n as well and thus the commutator width of finite-index subgroups can not be bounded.

For the last claim, consider a sequence (H_i) of subgroups of K such that H_i has commutator width at least i. Let $\psi_0\colon K \to K \times K \leq K$ be the map $k \mapsto \langle\!\langle k, \mathbb{1}\rangle\!\rangle$ and for $i \geq 1$ let $\psi_i\colon K \to K \times K \leq K$ be the map $k \mapsto \langle\!\langle \mathbb{1}, \psi_{i-1}(k)\rangle\!\rangle$. Then $H := \langle \psi_i(H_i) : i \in \mathbb{N}\rangle$ is a subgroup of K and hence of G and is isomorphic to the restricted direct product of the H_i, so it has infinite width. □

6. Implementation in GAP

6.1. Usage of the attached files. Typing the command `gap verify.g` in the main directory of the archive will produce as output a list of functions with their return value. All these functions should return **`true`**.
This approach uses precomputed data which are also in the archive, and is very fast. Furthermore, these data can be recomputed if a sufficiently new version of GAP and some packages are present. For details see Section 6.2.
This is what the functions check:

`verifyLemma90orbits`: This function verifies that there are indeed 90 orbits of U_3 on Q^6 as claimed in Lemma 4.5.

`verifyLemma86orbits`: Analogously to the previous function this one verifies that there are 86 orbits of U_2 on Q^4.

`verifyLemmaExistGoodConstraints`: This verifies that for each $q \in G'/K'$ there is some $\gamma \in \mathfrak{R}_{\text{act}}$ such that (q, γ) forms a good pair. This is claimed in Lemma 4.16.

`verifyLemmaExistGoodConstraints4`: This is a sharper version of the previous function. It checks that the above statement is already true if one replaces $\mathfrak{R}_{\text{act}}$ by $\mathfrak{R}^4_{\text{act}}$ as claimed in Lemma 4.27.

`verifyPropExistsSuccessor`: This verifies that for each good pair $(q, \gamma) \in G'/K' \times (\mathfrak{R}_{\text{act}} \cup \mathfrak{R}^4_{\text{act}})$ there exists a $\gamma' \in \Gamma^q(\gamma)$ such that all preimages of $\bar{p}_{\text{rep}(Y_{6,1})}(q)$ under the map ρ' form good pairs with the constraint γ'. This is needed in the proof of Proposition 4.18 and Proposition 4.28.

`verifyCorollaryFiniteCWK`: Corollary 4.30 needs the existence of succeeding good pairs of the pair $(\mathbb{1}, \mathbb{1}) \in K'/K' \times \mathfrak{R}^4$. This function verifies this existence.

`verifyExistGoodConjugacyConstraints`: This verifies that for the equation $a^{X_1}a^{X_2}a^{X_3}a^{X_4}a^{X_5}a$ there are constraints γ that admit good succeeding pairs. This is needed in the proof of Proposition 4.31.

`verifyGermGroup4hasCW`: This function verifies the existence of an element in the derived subgroup of the 4-th level germgroup that is not a commutator.

6.2. Precomputed data. In the interactive gap shell started by `gap verify.g` the precomputed data is read from some files in `gap/PCD/` and stored in a record `PCD`.
One can use the function `RedoPrecomputation` with one argument. In each case the result is written to one ore multiple files and will override the original precomputed data. The argument is a string and can be one of the following:

`"orbits"`: This will compute the 90 orbits of $\text{Aut}(F_6)/U_3$ and the 86 orbits of $\text{Aut}(F_4)/U_2$. This computation will take about 12 hours on an ordinary machine and has no progress bar.

`"goodpairs"`: First this will compute for each constraint $\gamma \in \mathfrak{R} \cup \mathfrak{R}^4$ the set of all $q \in G'/K'$ such that (q, γ) is a good pair.

Then it computes for each good pair (q, γ) one $\gamma' \in \Gamma_q(\gamma)$ with decorated $X = Y_{6,1}$ or $X = Y_{4,1} \in S$ as defined in equation (2) which fulfills depending whether $\gamma \in \mathfrak{R}^4_{\text{act}}$ or $\gamma \in \mathfrak{R}_{\text{act}}$ either Proposition 4.18 or Proposition 4.28. This computation will take about half an hour on ordinary machines and is equipped with a progress bar.

Afterwards the succeeding pairs of $(\mathbb{1}, \mathbb{1})$ which are needed for Corollary 4.30 are computed.

"conjugacywidth": Denote by $\mathcal{E}_g$ the equation $a^{X_1}a^{X_2}a^{X_3}a^{X_4}a^{X_5}ag$. For each $\tau(g) = q \in G'/K'$ this will compute a constraint $\gamma\colon F_5 \to Q$ for the equations $\mathcal{E}_g$ and a constraint $\gamma'\colon F_4 \to Q$ such that $(\gamma * \pi)(\mathcal{E}_g) = \mathbb{1}$,

$$\mathcal{E}'_g := \mathfrak{nf}(\tilde{\Phi}_\gamma(\mathcal{E}_g)) = [X_1, X_2][X_3, X_4](g@2)(g@1),$$

and $(\mathcal{E}'_g, \gamma')$ is a good pair for all g with $\tau(g) = q$.

The computation will take about one hour and is equipped with a progress bar.

"charactertable": This will compute the character table of the 4-th level germgroup and the set of irreducible characters. As the germgroup is quite large, this will take about 3 hours. There is no kind of progress bar.

"noncommutator": Inside the 4-th level germgroup there is an element which is not a commutator but in the commutator subgroup. Since this group is finite we could in principle search by brute force for a commutator. Luckily there are only 3106 irreducible characters in this group and therefore we can use Burnside's formula (1.1). The search will almost immediately give a result. Most of the computation time is used to assert that the found element is indeed not a commutator.

The element is then lifted to its preimage in G with a minimal number of states.

Checking the assertion will take approximately 3 hours and is equipped with a progress bar.

"all": This will do all of the above one after another.

To recompute the orbits or the charactertable GAP should be started with the `-o` flag to provide enough memory for the computation. For example start GAP by `gap -o 8G verify.g`

6.3. Implementation details.

6.3.1. *Reduced Constraints.* The proof of Lemma 4.4 in [**LMU16**] provides a constructive method to reduce any constraint to one with support only in the first five variables. We have implemented this in the function `ReducedConstraint` in the file `gap/functionsFR.g`.

It uses that the quotient $Q = G/K$ is a polycyclic group with

$$C_0 = Q = \langle \pi(a), \pi(b), \pi(d) \rangle \,, \qquad C_1 = \langle \pi(a), \pi(d) \rangle \,, \qquad C_2 = \langle \pi(ad) \rangle \,.$$

We take the generators of U_n as given in the proof of Lemma 4.5 plus additional ones which switch two neighboring pairs:

$$s_i\colon \begin{array}{rl} X_i \mapsto & X_{i+2} \\ X_{i+1} \mapsto & X_{i+3} \\ X_{i+2} \mapsto & X_i^{[X_{i+2},X_{i+3}]} \\ X_{i+3} \mapsto & X_{i+1}^{[X_{i+2},X_{i+3}]} \end{array} \quad \text{for } i = 1, 3, \ldots, 2n-3.$$

It can easily be checked, that these are also contained in U_n. These elements are used to reduce a given constraint in a form of a list with entries in Q to a list where all entries with index larger then 5 are trivial. This constraint can then be further reduced by a lookup table for the orbits of $\mathrm{Aut}(F_6)/U_3$.

If the file `verify.g` is loaded in a GAP environment with the FR package available the function `ReducedConstraint` can be used as an alias to get reduced constraints. For example:

```
gap> f1 := Q.3;
gap> gamma:= [f1,f1,f1,f1,f1,f1];
gap> constr := ReducedConstraint(gamma);;
gap> Print(constr.constraint);
```

```
[ <id>, <id>, <id>, <id> , f1, <id>]
```

6.3.2. *Good pairs.* For $g \in G$ and a constraint γ the question whether (g, γ) is a good pair depends only on the image of g in G/K' and the representative of $\gamma \in \mathfrak{R}$. (See Section 4.1.) So this is already a finite problem.
Given a given constraint γ, to obtain all q which form a good pair we can enumerate all possible commutators $[r_1, r_2][r_3, r_4][r_5, r_6]$ with $r_i \in \rho^{-1}(\gamma(X_i))$. Since $|K/K'| = 64$, it would take too much time to consider all combinations at once; thus the possible values for $[r_1, r_2]$ are computed and in a second step triple products of those elements are enumerated. This is implemented in the function `goodPairs` in the file `gap/functions.g`.

6.3.3. *Successors.* The key ingredient for the proof of Theorem A is Proposition 4.18. The main computational effort there is to compute the sets $\Gamma_q(\gamma)$ and find good pairs inside them.
This is implemented exactly as explained in the construction of the map Γ_q in the function `GetSuccessor` in the file `gap/precomputeGoodPairs.g`. Given an element $q \in G'/K'$ and an active constraint γ this function returns a tuple (γ', X) with $\gamma \in \mathfrak{R}$ and X the decorated element $Y_{6,1}$ or $> Y_{4,1}$ depending if $\gamma \in \mathfrak{R}^4$ or $\gamma \in \mathfrak{R}$.
Given an inactive constraint γ it returns a pair of constraints γ_1, γ_2 such that both have nontrivial activity and with ω the map from the branch structure it holds: $\omega(\langle\!\langle \gamma_1(X_i), \gamma_2(X_i) \rangle\!\rangle) = \gamma(X_i)$.
If the FR package is available the function `GetSuccessorLookup` can be used to explore the successors of elements. It returns the succeeding pair. For example

```
gap> f4 := Q.1;
gap> gamma:= [f4,f4,f4,f4,f4,f4];;
gap> g := (a*b)^8;;
gap> IsGoodPair(g,gamma);
```

```
true
```

```
gap> suc := GetSuccessorLookup(g,gamma);;
gap> suc[1];
```

```
<Trivial Mealy element on alphabet [ 1 .. 2 ]>
```

```
gap> suc[2].constraint;
```

```
[ <id>, <id>, <id>, <id>, f1*f3, <id> ]
```

CHAPTER B

Documentation of the equation package

Equations

Version 0.2.0

20 Dec 2017

Thorsten Groth

Thorsten Groth Email: thorsten.groth@mathematik.uni-goettingen.de

Copyright

Contents

Chapter 1

Installation

The package is installed by unpacking the archive in the `pkg/` directory of your GAP installation.

Example

```
gap> LoadPackage("equations");
true
```

Chapter 2

Example Session

We show some examples for using this package. The used methods are described in the latter chapter.

2.1 Normal form of equations

Let us consider some equations over the alternating group A_5. We start with defining the group in which our equations live in:

Example

```
gap> LoadPackage("equations");
true
gap> A5 := AlternatingGroup(5);
Alt( [ 1 .. 5 ] )
gap> EqG := EquationGroup(A5);
<free product group>
```

Now we enter the equation $E = X_2(1,2,3)X_1^{-1}X_2^{-1}(1,3)(4,5)X_3X_1X_3^{-1}$.

Example

```
gap> g := (1,2,3);;h := (1,3)(4,5);;
gap> vars := VariablesOfEquationGroup(EqG);
[ FreeProductElm([ X1 ]), FreeProductElm([ X2 ]), ... ]
gap> x1 := vars[1];; x2 := vars[2];; x3 := vars[3];;
gap> eq := Equation(x2*g*x1^-1*x2^-1*h*x3*x1*x3^-1);
Equation in [ X1, X2, X3 ]
gap> Print(eq);
FreeProductElm([ X2, (1,2,3), X1^-1*X2^-1, (1,3)(4,5), X3*X1*X3^-1 ])
```

Let us see what the normal form of this equation is:

Example

```
gap> Genus(eq);
1
gap> nf := NormalFormOfEquation(eq);
Equation in [ X1, X2, X3 ]
gap> Print(nf);
FreeProductElm([ X1^-1*X2^-1*X1*X2*X3^-1, (1,2,3), X3, (1,3)(4,5) ])
```

We know a solution for this normal form: $s{:}X_1 \mapsto (1,2,4)$, $X_2 \mapsto (1,2,5)$, $X_3 \mapsto ()$.

Example

```
gap> s:=EquationEvaluation(EqG,EquationVariables(eq),[(1,2,4),(1,2,5),()]);
[ X1, X2, X3 ]"->"[ (1,2,4), (1,2,5), () ]
gap> IsSolution(s,nf);
true
gap> nf^s;
()
gap> IsSolution(s,eq);
false
gap> eq^s;
(1,2,4,3,5)
```

Let us compute the solution for E.

Example

```
gap> sE:= NormalizingHomomorphism(nf)*s;
CompositionMapping( [ X1, X2, X3 ]->[ (1,2,4), (1,2,5), () ],
 CompositionMapping( [ X3, X2 ]->[ X2, X3 ], CompositionMapping(
CompositionMapping( CompositionMapping( [ X2 ]->[ (X2) ], [ X2 ]->[ X2^-1 ] ),
[ X3, X1 ]->[ <Free product element of length 5>,
  <Free product element of length 3> ] ), CompositionMapping( [ X1 ]->[ X1 ],
[ X3 ]->[ X3^-1 ] ) ) ) )
gap> IsSolution(sE,eq);
true;
gap> List(EquationVariables(eq),x->ImageElm(sE,x));
[ (2,3,4), (), (1,2,5,4,3) ]
```

Thus $s_E{:}X_1 \mapsto (2,3,4)$, $X_2 \mapsto ()$, $X_3 \mapsto (1,2,5,4,3)$ is a solution for the equaition E

2.2 Decomposition

Let us now study equations over groups acting on a rooted tree without having any explicitly given group in mind. Say $G \leq \mathrm{Aut}(\{1,2\}^*)$ and $g,h \in G$ and assume we want to see how the decomposition Φ_γ of the equation $E = [X,Y]g^Z h$ looks like. This decomposition will depend on the activity of g and on γ_{act}.

Example

```
gap> F := FreeGroup("X","Y","Z");; x:=F.1; y:=F.2; z:=F.3;
X
Y
Z
gap> G := FreeGroup("g","h");; g:=G.1; h := G.2;
g
h
gap> S2 := [(),(1,2)];
gap> EqG := EquationGroup(G,F);;
gap> eq := Equation(EqG,[Comm(x,y)*z^-1,g,z,h]);
Equation in [ X, Y, Z ]
gap> PhiE := [];
[ ]
```

```
gap> for actg in S2 do
>       DeqG := DecompositionEquationGroup(EqG,2,[actg,()]);;
>       for gamma_act in Cartesian([S2,S2,S2]) do
>         Add(PhiE,DecompositionEquation(DeqG,eq,gamma_act));
>       od;
>    od;
gap> Print(PhiE[1]);
Equation([ FreeProductElm([ X1^-1*Y1^-1*X1*Y1*Z1^-1, g1, Z1, h1 ]),
      FreeProductElm([ X2^-1*Y2^-1*X2*Y2*Z2^-1, g2, Z2, h2 ]) ])
gap> Print(PhiE[16]);
Equation([ FreeProductElm([ X2^-1*Y1^-1*X1*Y2*Z2^-1, g2, Z1, h2 ]),
      FreeProductElm([ X1^-1*Y2^-1*X2*Y1*Z1^-1, g1, Z2, h1 ]) ])
```

We see that for some (indeed for all but the first two cases) the states of the decomposition do not form independent systems. Let us see how an equivalent independent system looks like and find out which genus the corresponding equations have:

Example

```
gap> Apply(PhiE,E->DisjointFormOfDecomposedEquation(E));
gap> Print(PhiE[16]);
Equation([ FreeProductElm([ X2^-1*Y1^-1*Y2^-1*X2*Y1*Z1^-1, g1, Z2, h1, Y2*Z2^-1,
 g2, Z1, h2 ]), FreeProductElm([  ]) ])
gap> Genus(EquationComponent(PhiE[16],1));
2
gap> List(PhiE,E->Genus(EquationComponent(E,1)));
[ 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2 ]
```

2.3 Using the fr package

Finally let us do some computations in the Grigorchuk group. For example let us compute a solution for the equation $E = [X,Y]dacab$.

Example

```
gap> LoadPackage("fr");;
gap> G := GrigorchukGroup;;
gap> a:= G.1;; b:=G.2;; c:=G.3;; d:= G.4;;
gap> EqG := EquationGroup(G);;
gap> x:=EqG.5;y:=EqG.6;
(X1)
(X2)
gap> eq := Equation(Comm(x,y)*d*a*c*a*b);
<Equation in [ X1, X2 ]>
gap> gamma_a := GroupHomomorphismByImages(
>       Group(EquationVariables(eq)),SymmetricGroup(2),[(),(1,2)]);
[ X1, X2 ] -> [ (), (1,2) ]
gap> neq := DecompositionEquation(eq,gamma_a);
DecomposedEquation in [ Xn1, Xn2, Xn3, Xn4 ]
gap> deq := DisjointFormOfDecomposedEquation(neq);
DecomposedEquation in [ Xn2, Xn3, Xn4 ]
gap> nf := NormalFormOfEquation(EquationComponent(deq,1));
<Equation in [ Xn1, Xn2, Xn3 ]>
```

```
gap> s := EquationEvaluation(DecomposedEquationGroup(EqG),
>             EquationVariables(nf),[d,b,b]);
[ Xn1, Xn2, Xn3 ]->[ d, b, b ]
gap> IsSolution(s,nf);
true
gap> IsSolution(NormalizingHomomorphism(nf)*s,EquationComponent(deq,1));
true
gap> sol := DisjointFormHomomorphism(deq)*NormalizingHomomorphism(nf)*s;;
gap> ForAll(EquationComponents(neq),E->IsSolution(sol,E));
true;
gap> imgs := List(EquationVariables(neq),x->ImageElm(sol,x));
[ <Mealy element on alphabet [ 1 .. 2 ] with 6 states>,
  <Mealy element on alphabet [ 1 .. 2 ] with 7 states>, b^-1,
  <Mealy element on alphabet [ 1 .. 2 ] with 9 states> ]
gap> soleq := EquationEvaluation(EqG,EquationVariables(eq),
                [ComposeElement([imgs[1],imgs[2]],()),
                 ComposeElement([imgs[3],imgs[4]],(1,2))] );
[ X1, X2 ]->[ <Mealy element on alphabet [ 1 .. 2 ] with 9 states>,
  <Mealy element on alphabet [ 1 .. 2 ] with 10 states> ]
gap> IsSolution(soleq,eq);
true;
```

Chapter 3

FreeProducts

3.1 Construction

This package installs some new method for the command `FreeProduct`. Before it was only possible to construct free products of finitely presented groups.

3.1.1 FreeProductOp

▷ `FreeProductOp(`*`list, f.g., free, group`*`)` (operation)

Returns: The free product of all groups in *`list`*.

This is the method of choice if *`list`* contains at least one finetely generated free group but not only free groups.

3.1.2 FreeProductOp

▷ `FreeProductOp(`*`list, inf.g., free, group`*`)` (operation)

Returns: The free product of all groups in *`list`*.

We choose this is the method if *`list`* contains at least one infinetely generated free group but not only free groups.

3.1.3 FreeProductOp

▷ `FreeProductOp(`*`list, group`*`)` (operation)

Returns: The free product of all groups in *`list`*.

This method does always work. We refer to a more specific method if all of the groups are finitely presented. I.e. they are in the filter `IsFpGroup` and finitely generated.

If the resulting group was constructed by one of the new methods they will be in the following filter: `IsGeneralFreeProduct`

3.2 Filters

3.2.1 IsGeneralFreeProduct

▷ `IsGeneralFreeProduct(`*`obj`*`)` (filter)

Returns: `true` if *`obj`* is a general free product.

This filter can be used to check whether a given group was created as general free product.

3.2.2 IsFreeProductElm

▷ IsFreeProductElm(*obj*) (filter)

3.2.3 IsFreeProductHomomorphism

▷ IsFreeProductHomomorphism(*obj*) (filter)

3.3 Construction

3.3.1 GeneralFreeProduct (group)

▷ GeneralFreeProduct(*group*) (operation)

Returns: A a new general free product isomorphic to *group*.

Takes a group which has free product information stored and returns a new group which lies in the filter IsGeneralFreeProduct. The returned groups represents the free product of the groups in FreeProductInfo.groups.

Example

```
gap> S2 := SymmetricGroup(2);; SetName(S2,"S2");
gap> S3 := SymmetricGroup(3);; SetName(F2,"F2");
gap> G := FreeProduct(S2,S3);
<fp group on the generators [ f1, f2, f3 ]>
gap> G := GeneralFreeProduct(G);
S2*S3
```

3.3.2 GeneratorsOfGroup (group)

▷ GeneratorsOfGroup(*group*) (operation)

Returns: The generators of *group*.

3.3.3 \= (G,H)

▷ \=(*group*, *group*) (operation)

Returns: True if the free factors of the groups *G* and *H* are equal.

3.4 Elements

3.4.1 FreeProductElm (group,list,list)

▷ FreeProductElm(*group*, *word*, *factors*) (operation)

▷ FreeProductElmLetterRep(*group*, *word*, *factors*) (operation)

Returns: A new element in the group *group*.

This function constructs a new free product element, belonging to the group *group*.

words is a dense list of elements of any of the factors of *group*.

factors is a list of integers. *word*[i] must lie in the factor *factors*[*i*] of *group*. If this is not the case an error is thrown.

FreeProductElmLetterRep does not simplify the word by multipliying neighbored equal factor elements but stores the letters as given.

Example

```
gap> F2 := FreeGroup(2);; SetName(F2,"F2");
gap> S4 := SymmetricGroup(4);; SetName(S4,"S4");
gap> G := FreeProduct(F2,S4);
F2*S4
gap> e := FreeProductElm(G,[F2.1,F2.2,(1,2),F2.1],[1,1,2,1]);
FreeProductElm of length 3
gap> Print(e^2);
FreeProductElm([ f1*f2, (1,2), f1^2*f2, (1,2), f1 ])
gap> Print(FreeProductElmLetterRep(G,[F2.1,F2.2,(1,2),F2.1],[1,1,2,1]));
FreeProductElm([ f1, f2, (1,2), f1 ])
```

There are two representations for this kind of elements.

3.4.2 IsFreeProductElmRep

▷ IsFreeProductElmRep(*obj*) (filter)

▷ IsFreeProductElmLetterRep(*obj*) (filter)

Returns: true if *obj* is a general free product element in standard/letter storing representation.

3.5 Basic operations

3.5.1 * (freeproductelm,freeproductelm)

▷ *(*e1*, *e2*) (operation)

Returns: The product of the two elements.

3.5.2 * (freeproductelm,group elm)

▷ *(*e1*, *e2*) (operation)

Returns: The product of *e1* and the image of *e2* under the embedding into the free product group.

Only works if *e2* lies in one of the free factors of the free product group.

3.5.3 InverseOp (freeproductelm)

▷ InverseOp(*elm*) (operation)

Returns: The inverse element

3.5.4 OneOp (freeproductelm)

▷ OneOp(*elm*) (operation)

Returns: The identity element

3.5.5 \= (freeproductelm,freeproductelm)

▷ `\=(e1, ee2)` (operation)

Returns: True if the two elements are equal.
#

3.5.6 Length (freeproductelm)

▷ `Length(e1)` (operation)

Returns: The length of the list that stores the elements of the free factors

3.5.7 \[\] (freeproductelm,integer)

▷ `\[\](e1, i)` (operation)

Returns: The free product element consisting only of the `i`-th entry of the underlying list of elements.

3.5.8 Position (freeproductelm)

▷ `Position(e1)` (operation)

Returns: The position of the element `e1` in the underlyig list.

Example

```
gap> F2 := FreeGroup(2);; SetName(F2,"F2");
gap> S4 := SymmetricGroup(4);; SetName(S4,"S4");
gap> G := FreeProduct(F2,S4);
F2*S4
gap> e := FreeProductElm(G,[F2.1,F2.2,(1,2),F2.1],[1,1,2,1]);;Print(e);
FreeProductElm([ f1*f2, (1,2), f1 ])
gap> Length(e);
3
gap> Position(e,(1,2));
2
gap> Print(e[1]);
FreeProductElm([ f1*f2 ])
```

3.6 Homomorphisms

3.6.1 FreeProductHomomorphism (group,group,list)

▷ `FreeProductHomomorphism(source, target, homs)` (operation)

Returns: A new group homomorphism from `source` to `target`.

This function constructs a new group homomorphism from the general free product group `source` to the general free product group `target` by mapping the factor `i` by the group homomorphism `homs[i]` to the ith factor of `target`.

`homs` is a dense list of group homomorphisms where the source of `homs[i]` must be the `i`th factor of `source` and the range of `homs[i]` must be the `i`th factor of `target`.

Example

```
gap> F2 := FreeGroup(2);; SetName(F2,"F2");
gap> S4 := SymmetricGroup(4);; SetName(S4,"S4");
gap> A4 := AlternatingGroup(4);; SetName(A4,"A4");
gap> G := FreeProduct(F2,S4); H := FreeProduct(F2,A4);
F2*S4
F2*A4
gap> hf := GroupHomomorphismByImages(F2,F2,[F2.2,F2.1]);;
gap> hg := GroupHomomorphismByFunction(S4,A4,s->Comm(s,S4.2));;
gap> h := FreeProductHomomorphism(G,H,[hf,hg]);
<mapping: F2*S4 -> F2*A4 >
gap> e := FreeProductElm(G,[F2.1,F2.2,(1,2),F2.1],[1,1,2,1]);
FreeProductElm of length 3
gap> Print(e^h);
FreeProductElm([ f2*f1*f2 ])
```

3.6.2 IsGeneralFreeProductRep

▷ IsGeneralFreeProductRep(*obj*) (filter)

Returns: true if *obj* is a general free product element in standard/letter storing representation.

3.7 Other operations

3.7.1 Abs (assocword)

▷ Abs(*obj*) (operation)

Returns: An assocword without inverses of generators

In the word *obj* all occurencies of inverse generators are replaced by the coresponding generators.

Example

```
gap> F2 := FreeGroup(2);; SetName(F2,"F2");
gap> w := F2.1^-1*F2.2*F2.1*F2.2^-1;
f1^-1*f2*f1*f2^-1
gap> Abs(w);
(f1*f2)^2
```

3.7.2 \in (elm,list)

▷ \in(*elm*, *list*) (operation)

Returns: true if elm is in the infinite list *list*

Chapter 4

Equations

We fix a set $\mathscr{X}$ and call its elements *variables*. We assume that $\mathscr{X}$ is infinite countable, is well ordered, and its family of finite subsets is also well ordered, by size and then lexicographic order. We denote by $F_{\mathscr{X}}$ the free group on the generating set $\mathscr{X}$.

Let G be a group. The *equation group* will be the free product $G * F_{\mathscr{X}}$ and the elements belonging to G will be called *constants*.

A G-equation is an element E of the group $F_{\mathscr{X}} * G$ regarded as a reduced word. For E a G-equation, its set of *variables* $\mathrm{Var}(E) \subset \mathscr{X}$ is the set of symbols in $\mathscr{X}$ that occur in it; namely, $\mathrm{Var}(E)$ is the minimal subset of $\mathscr{X}$ such that E belongs to $F_{\mathrm{Var}(E)} * G$.

A *quadratic* equation is an equation in which each variable $X \in \mathrm{Var}(E)$ occurs exactly twice. A quadratic equation is called *oriented* if for each variable $X \in \mathrm{Var}(\mathscr{X})$ both letters X and X^{-1} occure in the reduced word E.

4.1 Construction

4.1.1 IsEquationGroup

▷ `IsEquationGroup(`*obj*`)` (filter)

Returns: `true` if *obj* is a general free product over to groups `G,F` where `F` is a free group.

The free factor `F` represents the group of variables for the equations.

4.1.2 EquationGroup (group,group)

▷ `EquationGroup(`*G, F*`)` (operation)

Returns: A a new *G*-group for equations over *G*.

Uses the `FreeProduct` method to create the free product object. The second argument *F* must be a free group.

Example

```
gap> S2 := SymmetricGroup(2);; SetName(S2,"S2");
gap> F := FreeGroup(infinity,"xn",["x1","x2"]);;SetName(F,"F");
gap> EqG := EquationGroup(S2,F);
S2*F
```

4.1.3 EquationGroup (group)

▷ `EquationGroup(G)` (operation)

Returns: A a new `G`-group for equations over `G`.

Uses the `FreeProduct` method to create the free product object of the given group and the free group on infinitely many generators

Example
```
gap> S2 := SymmetricGroup(2);; SetName(S2,"S2");
gap> EqG := EquationGroup(S2);
S2*Free(oo)
```

4.1.4 VariablesOfEquationGroup (group)

▷ `VariablesOfEquationGroup(G)` (attribute)

Returns: A list of the embedded free generators of the free facotor

If the equation group `G` was constructed with an infinitely generated free group as the group of variables, this returns an infinite list of generators.

4.1.5 ConstantsOfEquationGroup (group)

▷ `ConstantsOfEquationGroup(G)` (attribute)

Returns: The image of the embedding of the group of constants in `G`

4.1.6 Equation (group,list)

▷ `Equation(G, L)` (operation)

Returns: A a new element of the equation group `G`

Creates a `FreeProductElm` from the list `L`. By default this elements will be cyclicaly reduced.

Example
```
gap> F2 := FreeGroup(2);; SetName(F2,"F2");
gap> S4 := SymmetricGroup(4);; SetName(S4,"S4");
gap> G := EquationGroup(S4,F2);
S4*F2
gap> e := Equation(G,[F2.1,F2.2,(1,2),F2.1]);
Equation in [ f1, f2 ]
gap> Print(e);
FreeProductElm([ f1*f2, (1,2), f1 ])
```

4.1.7 Equation (free product elm)

▷ `Equation(elm)` (operation)

Returns: A a new element of the equation group `G`

Creates a `FreeProductElm` from the `FreeProductElm` `elm`. By default this elements will be cyclicaly reduced.

Example
```
gap> G := EquationGroup(SymmetricGroup(4));
<free product group>
gap> e := Equation(G.4*G.1*G.2*G.3);
<Equation in [ X1, X2 ]>
```

```
gap> Print(e);
FreeProductElm([ X2, (2,3,4), X1 ])
```

4.1.8 EquationVariables (groupelement)

▷ `EquationVariables(E)` (attribute)

Returns: A list of all variables occuring in E.

The elements of the result are elements of the group of variables in the `EquationGroup`. See in contrast the attribute `EquationVariablesEmbedded`.

4.1.9 EquationVariablesEmbedded (groupelement)

▷ `EquationVariablesEmbedded(E)` (attribute)

Returns: A list of all variables occuring in E.

The elements of the result are elements of the `EquationGroup`. and thus `FreeProductElm`s of length 1. See in contrast the attribute`EquationVariables`.

4.1.10 EquationLetterRep (equation)

▷ `EquationLetterRep(E)` (attribute)

Returns: A a new element of the equation group G in letter representation which is equal to E

In the standard representation of an equation the elements of the free group that are not devided by a constant are collected. In the letter representation they are seperate letters.

Example

```
gap> F2 := FreeGroup(2);; SetName(F2,"F2");
gap> S4 := SymmetricGroup(4);; SetName(S4,"S4");
gap> G := EquationGroup(S4,F2);
S4*F2
gap> e := Equation(G,[F2.1,F2.2,(1,2),F2.1]);
Equation in [ f1, f2 ]
gap> Print(e);
FreeProductElm([ f1*f2, (1,2), f1 ])
gap> Print(EquationLetterRep(e));
FreeProductElm([ f1, f2, (1,2), f1 ])
```

4.1.11 EquationLetterRep (group,list)

▷ `EquationLetterRep(G, L)` (attribute)

Returns: Creates a new equation in letter representation

4.1.12 IsQuadraticEquation (equation)

▷ `IsQuadraticEquation(E)` (property)

Returns: `true` if E is an quadratic equation.

4.1.13 IsOrientedEquation (equation)

▷ `IsOrientedEquation(E)` (property)

Returns: `true` if E is an oriented quadratic equation.

4.2 Homomorphisms

An *evaluation* is a G-homomorphism $e\colon F_{\mathscr{X}} * G \to G$. A *solution* of an equation E is an evaluation s satisfying $s(E) = 1$. If a solution exists for E then the equation E is called *solvable*. The set of elements $X \in \mathscr{X}$ with $s(X) \neq 1$ is called the *support* of the solution.

4.2.1 EquationHomomorphism (group,list,list)

▷ EquationHomomorphism(*G*, *vars*, *imgs*) (operation)

Returns: A a new homomorphism from `G` to `G`

If `G` is the group $H * F_X$ the result of this command is a H-homomorphism that maps the i-th variable of the list *vars* to the i-th member of *imgs*. Therefore *vars* can be a list without duplicates of variables. The list *imgs* can contain elements of the following type:

- Element of the group F_X
- Elements of the group H
- Lists of elements from the groups F_X and H. The list is then regarded as the corresponding word in G
- Elements of the group G

Example

```
gap> F3 := FreeGroup(3);; SetName(F3,"F3");
gap> S4 := SymmetricGroup(4);; SetName(S4,"S4");
gap> G := EquationGroup(S4,F3);
S4*F3
gap> e := Equation(G,[Comm(F3.2,F3.1)*F3.3^2,(1,2)]);
Equation in [ f1, f2, f3 ]
gap>  h := EquationHomomorphism(G,[F3.1,F3.2,F3.3],
> [F3.1*F3.2*F3.3,(F3.2*F3.3)^(F3.1*F3.2*F3.3),(F3.2^-1*F3.1^-1)^F3.3]);
[ f1, f2, f3 ]"->"[ f1*f2*f3, f3^-1*f2^-1*f1^-1*f2*f3*f1*f2*f3, f3^-1*f2^-1*f1^-1*f3 ]
gap> Print(e^h);
FreeProductElm([ f1^2*f2^2*f3^2, (1,2) ])
```

4.2.2 EquationEvaluation (group,list,list)

▷ EquationEvaluation(*G*, *vars*, *imgs*) (operation)

Returns: A a new evaluation from `G`

Works the same as *EquationHomomorphism* but the target of the homomorphism is the group of constants and all variables which are not specified in in *vars* are maped to the identity. Hence the only allowed input for *imgs* are elements of the group of constants.

Example

```
gap> F3 := FreeGroup(3);; SetName(F3,"F3");
gap> S4 := SymmetricGroup(4);; SetName(S4,"S4");
gap> G := EquationGroup(S4,F3);
S4*F3
gap> e := Equation(G,[Comm(F3.2,F3.1)*F3.3^2,(1,2,3)]);
Equation in [ f1, f2, f3 ]
gap>  h := EquationHomomorphism(G,[F3.1,F3.2,F3.3],[(),(),(1,2,3)]);
[ f1, f2, f3 ]"->"[ (), (), (1,3,2) ]
```

```
gap>  he := EquationEvaluation(G,[F3.1,F3.2,F3.3],[(),(),(1,2,3)]);
MappingByFunction( S4*F3, S4, function( q ) ... end )
gap> e^he;
()
gap> IsSolution(he,e);
true
```

4.3 Normal Form

For $m,n \geq 0$, $X_i, Y_i, Z_i \in \mathscr{X}$ and $c_i \in G$ the following two kinds of equations are called in *normal form*:

$$\begin{aligned} O_{n,m}: &\quad [X_1,Y_1][X_2,Y_2]\cdots[X_n,Y_n]c_1^{Z_1}\cdots c_{m-1}^{Z_{m-1}}c_m \\ U_{n,m}: &\quad X_1^2X_2^2\cdots X_n^2c_1^{Z_1}\cdots c_{m-1}^{Z_{m-1}}c_m\,. \end{aligned}$$

The form $O_{n,m}$ is called the oriented case and $U_{n,m}$ for $n>0$ the unoriented case. The parameter n is referred to as *genus* of the normal form of an equation. The pair (n,m) will be called the *signature* of the quadratic equation. It was proven by Commerford and Edmunds ([CJE81]) that every quadratic equation is isomorphic to one of the form $O_{n,m}$ or $U_{n,m}$ by an G-isomorphism.

4.3.1 NormalFormOfEquation (equation)

▷ NormalFormOfEquation(*E*) (attribute)

Returns: The normal form of the equation *E*

The argument *E* needs to be a quadratic equation. For each such equation there exists an equivalent equation in normal form.

The result is an equation in one of the forms $O_{n,m}, U_{n,m}$ equivalent to the equation E. The resulting equation has the attributes *NormalizingHomomorphism* and *NormalizingInverseHomomorphism* storing in the first case the homomorphism that maps E to the result and in the second case the inverse of this homomorphism.

4.3.2 NormalizingHomomorphism (equation)

▷ NormalizingHomomorphism(*E*) (attribute)

Returns: The EquationHomomorphism that maps to *E*

Only available if *E* was obtained via NormalFormOfEquation.

4.3.3 NormalizingInverseHomomorphism (equation)

▷ NormalizingInverseHomomorphism(*E*) (attribute)

Returns: The EquationHomomorphism that maps from *E*

Only available if *E* was obtained via NormalFormOfEquation. This is the inverse homomorphism to NormalizingInverseHomomorphism(E) #

Example

```
gap> F3 := FreeGroup("x","y","z");; SetName(F3,"F3");
gap> S4 := SymmetricGroup(4);; SetName(S4,"S4");
gap> G := EquationGroup(S4,F3);
S4*F3
gap> e := Equation(G,[Comm(F3.2,F3.1)*F3.3^2,(1,2)]);
Equation in [ x, y, z ]
```

```
gap>  nf := NormalFormOfEquation(e);;
gap> Print(nf);
FreeProductElm([ x^2*y^2*z^2, (1,2) ])
gap> e^NormalizingHomomorphism(nf)=nf;
true
gap> nf^NormalizingInverseHomomorphism(nf)=e;
true
```

4.3.4 Genus (equation)

▷ `Genus(`*E*`)` (operation)

Returns: The integer that is the genus of the equation

4.3.5 EquationSignature (equation)

▷ `EquationSignature(`*E*`)` (operation)

Returns: The list *[n,m]* of integers that is the signature of the equation

Chapter 5

FR-Equations

5.1 Decomposable equations

For self-similar groups one strategy to solve equations is to consider the inherit equations by passing to states. To use this methods the package FR ([Bar16]) from Laurent Bartholdi is needed.

Let G be a group which lies in the filter *IsFRGroup* and which admitts an embedding $\psi\colon G \to \tilde{G} \wr S_n$ where $\tilde{G}$ is the group generated by the states of the group G. Note that if G is a *self-similar* group then $G \simeq \tilde{G}$. Further let F_X be the free group on the generating set X. Given an equation group $G * F_X$ we will the fix n natural embeddings $\varphi_i\colon F \to F_{X^n}$ and call the group $(\tilde{G} * F_{X^n}) \wr S_n)$ the *decomposition equation group* of $G * F_X$. The decomposition of an equation e with variables $x_1, \dots, x_k$ with respect to a choice of activities $\sigma(x_i) \in S_n$ for each variable x_i is the image of e under the homomorphism

$$\begin{aligned} \Phi_\sigma\colon G * F_X &\to (\tilde{G} * (F_{X^n}) \wr S_n \\ x_i &\mapsto \varphi_i(x_i) \cdot \sigma(x_i) \\ g &\mapsto \psi_i(x_i) \end{aligned}$$

5.1.1 DecomposedEquationGroup (group)

▷ `DecomposedEquationGroup(G)` (attribute)

Returns: A new `EquationGroup`.

This method needs `G` to be an equation group where the group of constants is an fr-group. For `G` a group with free constant group see `DecomposedEquationGroup` (5.1.1). If F is the free group on the generating set X then the free group on the gerating set X^n is isomorphic to F^{*n} the n-fold free product of F .

This method returns the `EquationGroup` $G * F^{*n}$.

▷ `DecomposedEquationGroup(G, deg, acts)` (operation)

Returns: A new `EquationGroup`.

This method needs `G` to be an equation group where the group of constants is a free group on $n < \infty$ generators. The integer `deg` is the number of states each element will have. The list `acts` should be of length n and all elements should be permutation of `deg` elements. These will represent the activity of the generators of the free group.

5.1.2 DecompositionEquation (equation,group homorphism)

▷ `DecompositionEquation(E, sigma)` (operation)

Returns: A new equation in `G` which is the decomposed of the equation `E`.

The equation `E` needs to be a member of a EquationGroup $H = K * F$ where K is an `FRGroup`.

The argument *`sigma`* needs to be a group homomorphism $\sigma\colon F \to S_n$. Alternatively it can be a list of elements of S_n it is then regarded as the group homomorphism that maps the i-th variable of `eq` to the i-th element of the list.

The representation of the returned equation stores a list of words such that the i-th word represents an element in $G * \phi_i(F)$.

Example

```
gap> F := FreeGroup(1);; SetName(F,"F");
gap> G := EquationGroup(GrigorchukGroup,F);
GrigorchukGroup*F
gap>  sigma := GroupHomomorphismByImages(F,SymmetricGroup(2),[(1,2)]);
[ f1 ] -> [ (1,2) ]
gap>  e := Equation(G,[F.1^2,GrigorchukGroup.2]);
Equation in [ f1 ]
gap>  de := DecompositionEquation(e,sigma);
DecomposedEquation in [ f11, f12 ]
gap> Print(de);
Equation([ FreeProductElm([ f11*f12,a ]), FreeProductElm([ f12*f11,c ]) ])
```

5.1.3 DecompositionEquation (EquationGroup,equation,group homomorphism)

▷ `DecompositionEquation(`*`G, E, sigma`*`)` (operation)

Returns: A new equation in `G` which is the decomposed of the equation `E`.

The group `G` needs to be a *`DecompositionEqationGroup(H)`*, the equation `E` needs to be a member of the EquationGroup $H = K * F$.

The argument *`sigma`* needs to be a group homomorphism $\sigma\colon F \to S_n$. Alternatively it can be a list of elements of S_n it is then regarded as the group homomorphism that maps the i-th variable of `eq` to the i-th element of the list.

The representation of the returned equation stores a list of words such that the i-th word represents an element in $G * \phi_i(F)$.

Example

```
gap> F := FreeGroup("x1","x2");; SetName(F,"F");
gap> G := FreeGroup("g");; SetName(G,"G");
gap> eG := EquationGroup(G,F);
G*F
gap> DeG := DecompositionEquationGroup(eG,2,[(1,2)]);
G*G*F*F
gap>  e := Equation(eG,[Comm(F.1,F.2),G.1^2]);
Equation in [ x1, x2 ]
gap>  Print(DecompositionEquation(DeG,e,[(),()]));
Equation([ FreeProductElm([ x11^-1*x21^-1*x11*x21, g1*g2 ]),
 FreeProductElm([ x12^-1*x22^-1*x12*x22, g2*g1 ]) ])
```

5.1.4 EquationComponent (equation,int)

▷ `EquationComponent(`*`E, i`*`)` (operation)

Returns: The `i`-th component of the decomposed equation `E`.

Denote by p_i the natural projection $(G * F_{X^n})^n \rtimes S_n \to G * F_{X^n}$ to the i-th factor of the product. Given a decomposed Equation `E` and an integer $0 < i \leq n$ this method returns $p_i(E)$.

▷ `EquationComponents(E)` (operation)

Returns: The list of all components of the decomposed equation `E`.

Denote by p_i the natural projection $p_i\colon (G * F_{X^n})^n \rtimes S_n \to G * F_{X^n}$ to the i-th factor of the product. Given a decomposed Equation `E` this method returns the list $[p_1(E), p_2(E), \ldots, p_n(E)]$.

▷ `EquationActivity(E)` (operation)

Returns: The activity of the decomposed equation `E`.

Denote by *act* the natural projection $(G * F_{X^n}) \wr S_n \to S_n$. Given a decomposed Equation `E` this method returns $act(E)$.

5.1.5 DecomposedEquationDisjointForm (equation)

▷ `DecomposedEquationDisjointForm(E)` (attribute)

Returns: A decomposed equation that is a disjoint system.

If `E` is a decomposed equation there may be an overlap of the set of variables of some components. If `E` is a quadratic equation there is an equation homomorphism φ that maps each component to a new quadratic equation. Hence all maped components have pairwise disjoint sets of variables. This method computes such an homomorphism φ such that the solvability of the system of components remains unchanged. If s is a solution for the new system of components, then $s \circ \varphi$ is a solution for the old system.

5.1.6 DisjointFormOfDecomposedEquation (equation)

▷ `DisjointFormOfDecomposedEquation(E)` (attribute)

Returns: A decomposed Equation with disjoint components.

If `E` is a decomposed equation there may be an overlap of the set of variables of some components. If `E` is a quadratic equation there is an equation homomorphism φ that maps each component to a new quadratic equation. Hence all maped components have pairwise disjoint sets of variables. This method computes such an homomorphism φ such that the solvability of the system of components remains unchanged. If s is a solution for the new system of components, then $s \circ \varphi$ is a solution for the old system.

The method returns a the neq decomposed equation, that has the attribute `DisjointFormHomomorphism` that is the the homomorphism φ.

5.1.7 DisjointFormHomomorphism (equation)

▷ `DisjointFormHomomorphism(E)` (attribute)

Returns: The homomorphism that maps to `E`.

Only available if `E` was obtained via the method `DisjointFormOfDecomposedEquation`.

5.1.8 LiftSolution (equation,equation,equationhom,equationhom)

▷ `LiftSolution(DE, E, sigma, sol)` (operation)

Returns: An evaluation for E `eq`.

Given an equation `E` and a solution `sol` for its decomposed equation `DE` under the decomposition with activity `sigma` this method computes a solution for the equation `E`.

Note that the solution not neccecarily maps to the group of constants of `E` but can map to the group where all elements of the group of constants can appear as states. If the group of constants is layered, this two groups will coincide.

Example

```
gap> F := FreeGroup(2);; SetName(F,"F");
gap> Gr := GrigorchukGroup;; a:=Gr.1;; d:=Gr.4;;
gap> G := EquationGroup(Gr,F);;
gap> DG := DecompositionEquationGroup(G);;
gap>  sigma := GroupHomomorphismByImages(F,SymmetricGroup(2),[(1,2),()]);
[ f1, f2 ] -> [ (1,2), () ]
gap>  e := Equation(G,[Comm(F.1,F.2),Comm(d,a)]);
Equation in [ f1, f2 ]
gap>  de := DecompositionEquation(DG,e,sigma);
DecomposedEquation in [ f11, f21, f12, f22 ]
gap> dedj := DecomposedEquationDisjointForm(de);
rec( eq := DecomposedEquation in [ f11, f12, f22 ],
  hom := [ f21 ]->[ FreeProductElm of length 3 ] )
gap> EquationComponents(dedj.eq);
[ Equation in [ f11, f12, f22 ], Equation in [  ] ]
gap> s := EquationEvaluation(DG,EquationVariables(dedj.eq),[One(Gr),One(Gr),Gr.2]);
MappingByFunction( GrigorchukGroup*F*F, GrigorchukGroup, function( q ) ... end )
gap> IsSolution(s,EquationComponent(dedj.eq,1));
true
gap> ns := dedj.hom*s;; IsEvaluation(ns);
true
gap> ForAll(EquationComponents(de),F->IsSolution(ns,F));
true
gap> ls := LiftSolution(de,e,sigma,ns);;
gap> IsSolution(ls,e);
true
gap> ForAll(EquationVariables(e),x->Equation(G,[x])^ls in Gr); # only good luck
true
```

References

[Bar16] L. Bartholdi. FR, computations with functionally recursive groups, Version 2.3.6. \verb+https://www.gap-system.org/Packages/fr.html+, Apr 2016. GAP package. 126

[CJE81] L. P. Comerford Jr. and C. C. Edmunds. Quadratic equations over free groups and free products. *J. Algebra*, 68(2):276–297, 1981. 124

Index

List of Figures

List of Tables

Bibliography

[Ale83] S. V. Aleshin, *A free group of finite automata*, Vestnik Moskov. Univ. Ser. I Mat. Mekh. (1983), no. 4, 12–14. MR 713968

[Bar98] Laurent Bartholdi, *The growth of Grigorchuk's torsion group*, Internat. Math. Res. Notices (1998), no. 20, 1049–1054. MR 1656258

[Bar03] ———, *Endomorphic presentations of branch groups*, Journal of Algebra **268** (2003), no. 2, 419 – 443.

[Bar13] ———, *Representation zeta functions of self-similar branched groups*, ArXiv e-prints (2013).

[Bar16a] ———, *Algorithmic decidability of Engel's property for automaton groups*, Computer science—theory and applications, Lecture Notes in Comput. Sci., vol. 9691, Springer, [Cham], 2016, pp. 29–40. MR 3533843

[Bar16b] ———, *FR, computations with functionally recursive groups, Version 2.3.6*, `https://www.gap-system.org/Packages/fr.html`, Apr 2016, GAP package.

[BBSZ13] Ievgen V. Bondarenko, Natalia V. Bondarenko, Said N. Sidki, and Flavia R. Zapata, *On the conjugacy problem for finite–state automorphisms of regular rooted trees*, Noncommutative Geometry **7** (2013), no. 2, pp. 323–355.

[BdlH10] Laurent Bartholdi and Pierre de la Harpe, *Representation zeta functions of wreath products with finite groups*, Groups, Geometry, and Dynamics (2010).

[BG02] Laurent Bartholdi and Rostislav I. Grigorchuk, *On parabolic subgroups and Hecke algebras of some fractal groups*, Serdica Math. J. **28** (2002), no. 1, 47–90. MR 1899368

[BGL17] Laurent Bartholdi, Thorsten Groth, and Igor Lysenok, *Commutator width in the first Grigorchuk group*, ArXiv e-prints (2017).

[BGŠ03] Laurent Bartholdi, Rostislav I. Grigorchuk, and Zoran Šuniḱ, *Branch groups*, Handbook of algebra, Vol. 3, Handb. Algebr., vol. 3, Elsevier/North-Holland, Amsterdam, 2003, pp. 989–1112. MR 2035113

[BH16] Laurent Bartholdi and René Hartung, *Gap package LPRES, Version 0.3.0*, `https://laurentbartholdi.github.io/lpres`, Mai 2016, GAP package.

[BKN10] Laurent Bartholdi, Vadim A. Kaimanovich, and Volodymyr V. Nekrashevych, *On amenability of automata groups*, Duke Math. J. **154** (2010), no. 3, 575–598.

[BM17] L. Bartholdi and I. Mitrofanov, *The word and order problems for self-similar and automata groups*, ArXiv e-prints (2017).

[BN03] E. Bondarenko and V. Nekrashevych, *Post-critically finite self-similar groups*, Algebra Discrete Math. (2003), no. 4, 21–32. MR 2070400

[BS98] A. M. Brunner and Said Sidki, *The generation of* $\mathrm{GL}(n, \mathbf{Z})$ *by finite state automata*, Internat. J. Algebra Comput. **8** (1998), no. 1, 127–139. MR 1492064

[Bur55] William Burnside, *Theory of groups of finite order*, Dover Publications, Inc., New York, 1955, 2d ed. MR 0069818

[Cal09] Danny Calegari, *scl*, MSJ Memoirs, vol. 20, Mathematical Society of Japan, Tokyo, 2009. MR 2527432

[CE81] Leo P. Comerford, Jr. and Charles C. Edmunds, *Quadratic equations over free groups and free products*, J. Algebra **68** (1981), no. 2, 276–297. MR 608536

[Cul81] Marc Culler, *Using surfaces to solve equations in free groups*, Topology **20** (1981), no. 2, 133 – 145.

[CWM$^+$68] L. Carlitz, A. Wilansky, John Milnor, R. A. Struble, Neal Felsinger, J. M. S. Simoes, E. A. Power, R. E. Shafer, and R. E. Maas, *Advanced problems: 5600-5609*, The American Mathematical Monthly **75** (1968), no. 6, 685–687.

[Deh11] Max Dehn, *über unendliche diskontinuierliche Gruppen*, Math. Ann. **71** (1911), no. 1, 116–144. MR 1511645

[EHN81] David Eisenbud, Ulrich Hirsch, and Walter Neumann, *Transverse foliations of Seifert bundles and self-homeomorphism of the circle*, Comment. Math. Helv. **56** (1981), no. 4, 638–660. MR 656217

[Fin14] Elisabeth Fink, *Conjugacy growth and width of certain branch groups*, Internat. J. Algebra Comput. **24** (2014), no. 8, 1213–1231. MR 3296364

[FM11] Benson Farb and Dan Margalit, *A primer on mapping class groups*, Princeton University Press, 2011.

[GAP14] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.7.5*, 2014.

[GAP17] The GAP Group, Version 4.8.8, *Gap - reference manual*, 2017.

[Gil17] P. Gillibert, *An automaton group with undecidable order and Engel problems*, ArXiv e-prints (2017).

[GK92] R. I. Grigorchuk and P. F. Kurchanov, *On quadratic equations in free groups*, Proceedings of the International Conference on Algebra, Part 1 (Novosibirsk, 1989), Contemp. Math., vol. 131, Amer. Math. Soc., Providence, RI, 1992, pp. 159–171. MR 1175769

[GM05] Yair Glasner and Shahar Mozes, *Automata and square complexes*, Geom. Dedicata **111** (2005), 43–64. MR 2155175

[GOB13] Amir Gideon, Angel Omer, and Virág Bálint, *Amenability of linear-activity automaton groups*, Journal of the european Mathematical society **15** (2013), no. 3, 705 – 730.

[Gri80] Rostislav I. Grigorchuk, *On Burnside's problem on periodic groups*, Funktsional. Anal. i Prilozhen. **14** (1980), no. 1, 53–54. MR 565099

[Gri83] ———, *On the Milnor problem of group growth*, Dokl. Akad. Nauk SSSR **271** (1983), no. 1, 30–33. MR 712546

[Gri05] Rostislav Grigorchuk, *Solved and unsolved problems around one group*, Infinite groups: geometric, combinatorial and dynamical aspects, Progr. Math., vol. 248, Birkhäuser, Basel, 2005, pp. 117–218. MR 2195454

[Gs61] V. M. Glu ˇ skov, *Abstract theory of automata*, Uspehi Mat. Nauk **16** (1961), no. 5 (101), 3–62. MR 0138529

[GS83] Narain Gupta and Saï d Sidki, *On the Burnside problem for periodic groups*, Math. Z. **182** (1983), no. 3, 385–388. MR 696534

[Gur80] Robert M. Guralnick, *Expressing group elements as commutators*, Rocky Mountain J. Math. **10** (1980), no. 3, 651–654. MR 590227

[GW00] R. I. Grigorchuk and J. S. Wilson, *The conjugacy problem for certain branch groups*, Tr. Mat. Inst. Steklova **231** (2000), no. Din. Sist., Avtom. i Beskon. Gruppy, 215–230. MR 1841756

[GW03] Rostislav I. Grigorchuk and John S. Wilson, *A structural property concerning abstract commensurability of subgroups*, J. London Math. Soc. (2) **68** (2003), no. 3, 671–682. MR 2009443

[GZ01] Rostislav I. Grigorchuk and Andrzej Żuk, *The lamplighter group as a group generated by a 2-state automaton, and its spectrum*, Geom. Dedicata **87** (2001), no. 1-3, 209–244. MR 1866850

[HU79] John E. Hopcroft and Jeffrey D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley Publishing Co., Reading, Mass., 1979, Addison-Wesley Series in Computer Science. MR 645539

[Leo98] Yu. G. Leonov, *The conjugacy problem in a class of* 2*-groups*, Mat. Zametki **64** (1998), no. 4, 573–583. MR 1687212

[LMU16] Igor Lysenok, Alexei Miasnikov, and Alexander Ushakov, *Quadratic equations in the Grigorchuk group*, Groups Geom. Dyn. **10** (2016), no. 1, 201–239. MR 3460336

[LOST10] Martin W. Liebeck, Eamonn A. O'Brien, Aner Shalev, and Pham Huu Tiep, *The Ore conjecture*, J. Eur. Math. Soc. (JEMS) **12** (2010), no. 4, 939–1008. MR 2654085

[Lyn59] R. C. Lyndon, *The equation* $a^2b^2 = c^2$ *in free groups*, Michigan Math. J **6** (1959), 89–95. MR 0103218

[Mil99] G. A. Miller, *On the commutators of a given group*, Bull. Amer. Math. Soc. **6** (1899), no. 3, 105–109.

[Nek05] Volodymyr Nekrashevych, *Self-similar groups*, Mathematical Surveys and Monographs, vol. 117, American Mathematical Society, Providence, RI, 2005. MR 2162164

[Neu86] Peter M. Neumann, *Some questions of edjvet and pride about infinite groups*, Illinois J. Math. **30** (1986), no. 2, 301–316.

[Ore51] Oystein Ore, *Some remarks on commutators*, Proc. Amer. Math. Soc. **2** (1951), 307–314. MR 0040298

[Rhe68] A. H. Rhemtulla, *A problem of bounded expressibility in free products*, Mathematical Proceedings of the Cambridge Philosophical Society **64** (1968), no. 3, 573–584.

[Rhe69] ———, *Commutators of certain finitely generated soluble groups*, Can. J. Math. **21** (1969), no. 5, 1160–1164.

[Roz93] Alexander V. Rozhkov, *Centralizers of elements in a group of tree automorphisms*, Izv. Ross. Akad. Nauk Ser. Mat. **57** (1993), no. no. 6, 82–105. MR 1256568

[Roz98] ———, *The conjugacy problem in an automorphism group of an infinite tree*, Mat. Zametki **64** (1998), no. 4, 592–597. MR 1687204

[Seg00] Dan Segal, *The finite images of finitely generated groups*, Proceedings of the London Mathematical Society **82** (2000), 597–613.

[Seg09] Dan Segal, *Words. notes on verbal width in groups*, London Mathematical Society Lecture Note Series 361, Cambridge University Press, 2009.

[Sid87] Said Sidki, *On a 2-generated infinite 3-group: subgroups and automorphisms*, J. Algebra **110** (1987), no. 1, 24–55. MR 904180

[Sid00] ———, *Automorphisms of one-rooted trees: growth, circuit structure, and acyclicity*, J. Math. Sci. (New York) **100** (2000), no. 1, 1925–1943, Algebra, 12. MR 1774362

[SS05] P. V. Silva and B. Steinberg, *On a class of automata groups generalizing lamplighter groups*, Internat. J. Algebra Comput. **15** (2005), no. 5-6, 1213–1234. MR 2197829

[vSV12] Zoran ˇ Sunić and Enric Ventura, *The conjugacy problem in automaton groups is not solvable*, J. Algebra **364** (2012), 148–154. MR 2927052

[VV07] Mariya Vorobets and Yaroslav Vorobets, *On a free group of transformations defined by an automaton*, Geom. Dedicata **124** (2007), 237–249. MR 2318547

[WZ97] J. S. Wilson and P. A. Zalesskii, *Conjugacy separability of certain torsion groups*, Arch. Math. (Basel) **68** (1997), no. 6, 441–449. MR 1444655