Documentation for the package GT

Vasily A. Dolgushev

Abstract

GT-shadows are tantalizing objects [4], [5] that may be thought of as approximations to elements of the mysterious Grothendieck-Teichmueller group $\widehat{\mathsf{GT}}$ [6, Section 4]. They form a groupoid GTSh and they act on Grothendieck's child's drawings. This is a detailed documentation of the software package for working with GT-shadows and their action on child's drawings.

Contents

1 Introduction

1.1 The poset $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and the groupoid of GT-shadows

Let B_n be the Artin braid group and PB_n be the pure braid group on n strands [3], [9]. The standard generators of B_n are denoted by $\sigma_1, \ldots, \sigma_{n-1}$ and the standard generators of PB_n are denote by x_{ij} for $1 \leq i < j \leq n$. Recall [5] that $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ is the poset of finite index normal subgroups $\mathsf{N} \trianglelefteq \mathrm{B}_4$ such that $\mathsf{N} \leq \mathrm{PB}_4$. For every such N, we denote by $\mathsf{N}_{\mathrm{PB}_3}$ and $\mathsf{N}_{\mathrm{PB}_2}$ the following finite index normal subgroups in PB_3 and in PB_2:

$$\mathsf{N}_{\mathrm{PB}_3} := \varphi_{123}^{-1}(\mathsf{N}) \cap \varphi_{12,3,4}^{-1}(\mathsf{N}) \cap \varphi_{1,23,4}^{-1}(\mathsf{N}) \cap \varphi_{1,2,34}^{-1}(\mathsf{N}) \cap \varphi_{234}^{-1}(\mathsf{N}), \tag{1.1}$$

$$\mathsf{N}_{\mathrm{PB}_2} := \varphi_{12}^{-1}(\mathsf{N}_{\mathrm{PB}_3}) \cap \varphi_{12,3}^{-1}(\mathsf{N}_{\mathrm{PB}_3}) \cap \varphi_{1,23}^{-1}(\mathsf{N}_{\mathrm{PB}_3}) \cap \varphi_{23}^{-1}(\mathsf{N}_{\mathrm{PB}_3}), \tag{1.2}$$

respectively.

The symbols φ_{123}, $\varphi_{12,3,4}$, $\varphi_{1,23,4}$, $\varphi_{1,2,34}$, φ_{234} (resp. φ_{12}, $\varphi_{12,3}$, $\varphi_{1,23}$, φ_{23}) in (1.1) (resp. in (1.2)), denote the group homomorphisms $\mathrm{PB}_3 \to \mathrm{PB}_4$ (resp. $\mathrm{PB}_2 \to \mathrm{PB}_3$) defined by the formulas:

$$\begin{gathered}
\varphi_{123}(x_{12}) = x_{12}, \quad \varphi_{123}(x_{23}) = x_{23}, \quad \varphi_{123}(x_{13}) = x_{13}, \\
\varphi_{234}(x_{12}) = x_{23}, \quad \varphi_{234}(x_{23}) = x_{34}, \quad \varphi_{234}(x_{13}) = x_{24}, \\
\varphi_{12,3,4}(x_{12}) = x_{13}x_{23}, \quad \varphi_{12,3,4}(x_{23}) = x_{34}, \quad \varphi_{12,3,4}(x_{13}) = x_{14}x_{24}, \\
\varphi_{1,23,4}(x_{12}) = x_{12}x_{13}, \quad \varphi_{1,23,4}(x_{23}) = x_{24}x_{34}, \quad \varphi_{1,23,4}(x_{13}) = x_{14}, \\
\varphi_{1,2,34}(x_{12}) = x_{12}, \quad \varphi_{1,2,34}(x_{23}) = x_{23}x_{24}, \quad \varphi_{1,2,34}(x_{13}) = x_{13}x_{14},
\end{gathered} \tag{1.3}$$

and

$$\varphi_{12}(x_{12}) = x_{12}, \quad \varphi_{23}(x_{12}) = x_{23}, \quad \varphi_{12,3}(x_{12}) = x_{13}x_{23}, \quad \varphi_{1,23}(x_{12}) = x_{12}x_{13}. \tag{1.4}$$

It is easy to see that $\mathsf{N}_{\mathrm{PB}_3} \in \mathsf{NFI}_{\mathrm{PB}_3}(\mathrm{B}_3)$, $\mathsf{N}_{\mathrm{PB}_2} \in \mathsf{NFI}_{\mathrm{PB}_2}(\mathrm{B}_2)$ and the subgroup $\mathsf{N}_{\mathrm{PB}_2}$ is completely determined by its index $N_{\mathrm{ord}} := |\mathrm{PB}_2 : \mathsf{N}_{\mathrm{PB}_2}|$.

For every $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the triple $\mathsf{N}, \mathsf{N}_{\mathrm{PB}_3}, \mathsf{N}_{\mathrm{PB}_2}$ gives us a compatible equivalence relation $\sim_{\mathsf{N}}$ on the truncation $\mathsf{PaB}^{\leq 4}$ of the operad PaB [1], [5, Appendix A], [7, Chapter 6], [14]. The quotient $\mathsf{PaB}^{\leq 4}/\sim_{\mathsf{N}}$ is a truncated operad in the category of finite groupoids.

A GT-**pair** with the target N is a morphism of truncated operads $\mathsf{PaB}^{\leq 4} \to \mathsf{PaB}^{\leq 4}/\sim_{\mathsf{N}}$ and such morphisms are in bijection with pairs

$$(m, f\mathsf{N}_{\mathrm{PB}_3}) \in \{0, 1, \dots, N_{\mathrm{ord}} - 1\} \times \mathrm{PB}_3/\mathsf{N}_{\mathrm{PB}_3} \tag{1.5}$$

satisfying the hexagon relations

$$\sigma_1 x_{12}^m f^{-1} \sigma_2 x_{23}^m f \, \mathsf{N}_{\mathrm{PB}_3} \;=\; f^{-1}\sigma_1\sigma_2 (x_{13}x_{23})^m \, \mathsf{N}_{\mathrm{PB}_3}\,, \tag{1.6}$$

$$f^{-1}\sigma_2 x_{23}^m f \, \sigma_1 x_{12}^m \, \mathsf{N}_{\mathrm{PB}_3} \;=\; \sigma_2\sigma_1 (x_{12}x_{13})^m \, f \, \mathsf{N}_{\mathrm{PB}_3}\,, \tag{1.7}$$

and the pentagon relation

$$\varphi_{234}(f)\, \varphi_{1,23,4}(f)\, \varphi_{123}(f)\, \mathsf{N} \;=\; \varphi_{1,2,34}(f)\varphi_{12,3,4}(f)\, \mathsf{N}. \tag{1.8}$$

Both sides of (1.6) and (1.7) are elements of $\mathrm{B}_3/\mathsf{N}_{\mathrm{PB}_3}$ and both sides of (1.8) are elements of PB_4/N.

It is convenient to represent GT-pairs by tuples $(m, f) \in \mathbb{Z} \times \mathrm{PB}_3$ and we denote by $[m, f]$ the GT-pair represented by the tuple (m, f).

For every GT-pair $[m, f]$, we have the group homomorphisms

$$T_{m,f}^{\mathrm{PB}_4} : \mathrm{PB}_4 \to \mathrm{PB}_4/\mathsf{N}, \qquad T_{m,f}^{\mathrm{PB}_3} : \mathrm{PB}_3 \to \mathrm{PB}_3/\mathsf{N}_{\mathrm{PB}_3}, \qquad T_{m,f}^{\mathrm{PB}_2} : \mathrm{PB}_2 \to \mathrm{PB}_2/\mathsf{N}_{\mathrm{PB}_2}. \tag{1.9}$$

Explicit formulas for these homomorphisms are given in [5, Corollary 2.8].

A GT-**shadow with the target** N is an onto morphism of truncated operads $\mathsf{PaB}^{\leq 4} \to \mathsf{PaB}^{\leq 4}/\sim_{\mathsf{N}}$ and such morphisms are in bijection with pairs (1.5) satisfying the following conditions:

- (m, f) obeys relations (1.6), (1.7), (1.8),
- $2m + 1$ represents a unit in the ring $\mathbb{Z}/N_{\mathrm{ord}}\mathbb{Z}$, and
- the group homomorphism $T_{m,f}^{\mathrm{PB}_3} : \mathrm{PB}_3 \to \mathrm{PB}_3/\mathsf{N}_{\mathrm{PB}_3}$ is onto.

In this note, we tacitly identify GT-shadows (with the target N) with pairs (1.5) satisfying the above conditions and we denote by $\mathsf{GT}(\mathsf{N})$ the set of GT-shadows with the target N. For $[m, f] \in \mathsf{GT}(\mathsf{N})$, $T_{m,f}$ denotes the corresponding onto morphism of truncated operads

$$T_{m,f} : \mathsf{PaB}^{\leq 4} \to \mathsf{PaB}^{\leq 4}/\sim_{\mathsf{N}}\ . \tag{1.10}$$

Due to [5, Proposition 2.11], for every $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and $[m, f] \in \mathsf{GT}(\mathsf{N})$, the "kernel" of the morphism $T_{m,f}$ coincides with $\sim_{\mathsf{K}}$, where

$$\mathsf{K} := \ker(\mathrm{PB}_4 \xrightarrow{T_{m,f}^{\mathrm{PB}_4}} \mathrm{PB}_4/\mathsf{N}).$$

We call the kernel of $T_{m,f}^{\mathrm{PB}_4}$ the **source** of the GT-shadow $[m, f] \in \mathsf{GT}(\mathsf{N})$.

Since the morphism in (1.10) is onto, it induces the following isomorphism of truncated operads:

$$T_{m,f}^{\mathrm{isom}} : \mathsf{PaB}^{\leq 4}/\sim_{\mathsf{K}} \xrightarrow{\simeq} \mathsf{PaB}^{\leq 4}/\sim_{\mathsf{N}}, \tag{1.11}$$

where $\mathsf{K} := \ker(T_{m,f}^{\mathrm{PB}_4})$. Moreover,

$$T_{m,f} = T_{m,f}^{\mathrm{isom}} \circ \mathcal{P}_{\mathsf{K}},$$

where $\mathcal{P}_{\mathsf{K}}$ is the standard onto morphism $\mathsf{PaB}^{\leq 4} \to \mathsf{PaB}^{\leq 4}/\sim_{\mathsf{K}}$.

GT-shadows form a groupoid GTSh. The objects of GTSh are elements of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$. For $\mathsf{K},\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the set of morphisms $\mathsf{GTSh}(\mathsf{K},\mathsf{N})$ from K to N is

$$\mathsf{GTSh}(\mathsf{K},\mathsf{N}) := \big\{ [m,f] \in \mathsf{GT}(\mathsf{N}) \mid \ker(T^{\mathrm{PB}_4}_{m,f}) = \mathsf{K} \big\}.$$

The composition of morphisms comes from the obvious identification of $\mathsf{GTSh}(\mathsf{K},\mathsf{N})$ with the set of isomorphisms of truncated operads

$$\mathsf{PaB}^{\leq 4}/\sim_{\mathsf{K}} \ \xrightarrow{\simeq}\ \mathsf{PaB}^{\leq 4}/\sim_{\mathsf{N}} \ .$$

For example, the pair $(0, 1_{\mathrm{PB}_3})$ represents the identity morphism from N to N, the isomorphism $T^{\mathrm{isom}}_{0,1_{\mathrm{PB}_3}}$ is the identity map and

$$T_{0,1_{\mathrm{PB}_3}} = \mathcal{P}_{\mathsf{N}} : \mathsf{PaB}^{\leq 4} \to \mathsf{PaB}^{\leq 4}/\sim_{\mathsf{N}} \ .$$

Just as in [5], we will tacitly identify F_2 with the subgroup of PB_3 generated by the elements x_{12}, x_{23}. We also often denote the generators of F_2 by x and y, i.e. $x := x_{12}$ and $y := x_{23}$.

A GT-shadow is called **practical**, if it can be represented by a pair (m,f) where $f \in \mathsf{F}_2$. In this note, we assume that all GT-shadows are practical. *From now on*, $\mathsf{GT}(\mathsf{N})$ denotes the set of all *practical* GT-shadows with the target N. Moreover, GTSh denotes the groupoid of *practical* GT-shadows.

Due to [5, Remark 2.15], we have an explicit composition formula for (practical) GT-shadows. Namely, if $[m_1,f_1] \in \mathsf{GTSh}(\mathsf{N}^{(2)},\mathsf{N}^{(1)})$, $[m_2,f_2] \in \mathsf{GTSh}(\mathsf{N}^{(3)},\mathsf{N}^{(2)})$, and

$$\begin{aligned} m &:= 2m_1m_2 + m_1 + m_2\,, \\ f(x,y) &:= f_1(x,y)\, f_2(x^{2m_1+1}, f_1(x,y)^{-1} y^{2m_1+1} f_1(x,y)), \end{aligned} \tag{1.12}$$

then $[m,f] := [m_1,f_1] \circ [m_2,f_2]$, i.e. the pair (m,f) represents the GT-shadow $[m_1,f_1] \circ [m_2,f_2] \in \mathsf{GTSh}(\mathsf{N}^{(3)},\mathsf{N}^{(1)})$.

For $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, we set

$$\mathsf{N}_{\mathsf{F}_2} := \mathsf{N}_{\mathrm{PB}_3} \cap \langle\, x_{12}, x_{23} \,\rangle.$$

For $[m,f] \in \mathsf{GT}(\mathsf{N})$, the notation $T^{\mathsf{F}_2}_{m,f}$ is reserved for the group homomorphism

$$T^{\mathrm{PB}_3}_{m,f}\Big|_{\mathsf{F}_2} : \mathsf{F}_2 \ \to\ \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}.$$

This homomorphism is given explicitly by the formulas

$$T^{\mathsf{F}_2}_{m,f}(x_{12}) := x_{12}^{2m+1}\mathsf{N}_{\mathsf{F}_2}\,, \qquad T^{\mathsf{F}_2}_{m,f}(x_{23}) := f^{-1}x_{23}^{2m+1} f \mathsf{N}_{\mathsf{F}_2}\,.$$

A GT-shadow $[m,f] \in \mathsf{GT}(\mathsf{N})$ is called **charming** if[1] the coset $f\mathsf{N}_{\mathsf{F}_2}$ can be represented by $f_1 \in [\mathsf{F}_2,\mathsf{F}_2]$. Equivalently, $f\mathsf{N}_{\mathsf{F}_2} \in [\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}, \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}]$.

For $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, we denote by $\mathsf{GT}^{\heartsuit}(\mathsf{N})$ the set of charming GT-shadows in $\mathsf{GT}(\mathsf{N})$. Due to [5, Proposition 2.22], charming GT-shadows form a subgroupoid of GTSh and we denote this subgroupoid by $\mathsf{GTSh}^{\heartsuit}$.

[1] See Proposition 2.1 and Remark 2.3 in [4].

1.2 GT-shadows coming from elements of $\widehat{\mathsf{GT}}$

Recall that the Grothendieck-Teichmueller group $\widehat{\mathsf{GT}}$ [6, Section 4] consists of pairs $(\hat{m}, \hat{f}) \in \widehat{\mathbb{Z}} \times \widehat{\mathsf{F}}_2$ that satisfy the profinite versions of the hexagon relations, the profinite version of the pentagon relation and the invertibility condition[2]. The set of such pairs can be identified with the set of continuous automorphisms of the operad $\widehat{\mathsf{PaB}}$ [5, Introduction] and this identification is used to define the multiplication on $\widehat{\mathsf{GT}}$.

Let $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$. We say [5, Section 2.4] that a GT-shadow $[m, f] \in \mathsf{GT}(\mathsf{N})$ **comes from** an element $(\hat{m}, \hat{f}) \in \widehat{\mathsf{GT}}$ if $m + N_{\mathrm{ord}}\mathbb{Z}$ (resp. $f\mathsf{N}_{\mathsf{F}_2}$) coincides with the image of the standard homomorphism $\widehat{\mathbb{Z}} \to \mathbb{Z}/N_{\mathrm{ord}}\mathbb{Z}$ (resp. with the image of the standard homomorphism $\widehat{\mathsf{F}}_2 \to \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$). We say that an element $[m, f] \in \mathsf{GT}(\mathsf{N})$ is **genuine** if $[m, f]$ comes from an element of $\widehat{\mathsf{GT}}$; otherwise, we say that $[m, f] \in \mathsf{GT}(\mathsf{N})$ is **fake**. If $[m, f] \in \mathsf{GT}(\mathsf{N})$ comes from an element $(\hat{m}, \hat{f}) \in \widehat{\mathsf{GT}}$, then $[m, f]$ is an approximation of $(\hat{m}, \hat{f})$.

Due to [5, Proposition 2.20], every genuine GT-shadow is charming. For this reason, it may make sense to study only charming GT-shadows.

Let $\mathsf{K}, \mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, $\mathsf{K} \leq \mathsf{N}$ and $(m, f) \in \mathbb{Z} \times \mathsf{F}_2$ be a pair that represents an element in $\mathsf{GT}^{\heartsuit}(\mathsf{K})$. Recall (see eq. (3.24) in [5]) that, in this situation, the same pair (m, f) also represents a charming GT-shadow with the target N. Hence we have a natural map (of sets)

$$\mathsf{GT}^{\heartsuit}(\mathsf{K}) \to \mathsf{GT}^{\heartsuit}(\mathsf{N}). \tag{1.13}$$

According to [5, Definition 3.12], a GT-shadow $[m_1, f_1] \in \mathsf{GT}^{\heartsuit}(\mathsf{N})$ **survives into** K, if $[m_1, f_1]$ belongs to the image of the map in (1.13), i.e. there exists $[m, f] \in \mathsf{GT}^{\heartsuit}(\mathsf{K})$ such that

$$m \equiv m_1 \mod N_{\mathrm{ord}} \qquad \text{and} \qquad f\mathsf{N}_{\mathsf{F}_2} = f_1\mathsf{N}_{\mathsf{F}_2}\,.$$

Due to [5, Corollary 3.13], a GT-shadow in $\mathsf{GT}^{\heartsuit}(\mathsf{N})$ is genuine if and only if it survives into K for all $\mathsf{K} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ such that $\mathsf{K} \leq \mathsf{N}$.

We should remark that, for all $\mathsf{K}, \mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ with $\mathsf{K} \leq \mathsf{N}$, the same construction also gives us a natural map of sets

$$\mathsf{GT}(\mathsf{K}) \to \mathsf{GT}(\mathsf{N}).$$

1.3 Isolated objects of the groupoid $\mathsf{GTSh}^{\heartsuit}$

It is easy to see [5, Section 3.1] that the groupoid $\mathsf{GTSh}^{\heartsuit}$ is highly disconnected. However, due to [5, Proposition 3.1], the connected component $\mathsf{GTSh}^{\heartsuit}_{\mathrm{conn}}(\mathsf{N})$ of N in $\mathsf{GTSh}^{\heartsuit}$ is a finite groupoid for every $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$.

A GT-shadow $[m, f] \in \mathsf{GT}^{\heartsuit}(\mathsf{N})$ is called **settled**, if its source coincides with its target N, i.e.

$$\ker(T^{\mathrm{PB}_4}_{m,f}) = \mathsf{N}. \tag{1.14}$$

In other words, settled elements of $\mathsf{GT}^{\heartsuit}(\mathsf{N})$ are precisely automorphisms of N in the groupoid $\mathsf{GTSh}^{\heartsuit}$.

An element $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ is called **isolated** if every GT-shadow in $\mathsf{GT}^{\heartsuit}(\mathsf{N})$ is settled. It is clear that $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ is isolated if and only if N is the only object of the connected component $\mathsf{GTSh}^{\heartsuit}_{\mathrm{conn}}(\mathsf{N})$. It is also clear that, in this case, $\mathsf{GT}^{\heartsuit}(\mathsf{N}) = \mathsf{GTSh}^{\heartsuit}(\mathsf{N}, \mathsf{N})$, i.e.

[2] In particular, $2\hat{m} + 1$ is a unit in the ring $\widehat{\mathbb{Z}}$.

$\mathsf{GT}^{\heartsuit}(\mathsf{N})$ is a (finite) group. The notation $\mathsf{NFI}^{isolated}_{\mathrm{PB}_4}(\mathrm{B}_4)$ is reserved for the sub-poset of isolated elements of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$.

Due to [5, Proposition 3.3], for every $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the subgroup

$$\mathsf{N}^{\sharp} := \bigcap_{[m,f]\in\mathsf{GT}(\mathsf{N})} \ker(T^{\mathrm{PB}_4}_{m,f})$$

is an isolated element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$. Moreover, due to [5, Proposition 3.6], the intersection of two isolated elements of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ is an isolated element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$.

Remark 1.1 Let $\mathsf{K}, \mathsf{N} \in \mathsf{NFI}^{isolated}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and $\mathsf{K} \leq \mathsf{N}$. It is easy to see that, in this case, the map in (1.13) is a group homomorphism.

1.4 The action of GT-shadows on child's drawings

For $d \in \mathbb{Z}_{\geq 1}$, S_d denotes the symmetric group on d letters and $\mathfrak{P}_d$ denotes the set of partitions of d. The notation $\mathfrak{ct}$ is reserved for the standard map $S_d \to \mathfrak{P}_d$ which assigns to a permutation its cycle structure. A subgroup $H \leq S_d$ is called **transitive** if it acts transitively on the set $\{1, 2, \ldots, d\}$.

Recall that a **child's drawing** of degree d is represented by a pair of permutations $c := (c_1, c_2)$ in S_d for which the subgroup $\langle\, c_1, c_2 \,\rangle \leq S_d$ is transitive. Two pairs $c := (c_1, c_2)$ and $\tilde{c} := (\tilde{c}_1, \tilde{c}_2)$ in S_d represent the same child's drawing if and only if there exists $h \in S_d$ such that $\tilde{c}_i = hc_ih^{-1}$, $i \in \{1, 2\}$. For such a pair $c := (c_1, c_2)$, $[c]$ denotes the corresponding child's drawing.

The (conjugacy class of the) permutation group $\langle\, c_1, c_2 \,\rangle \leq S_d$ is called the **monodromy group** of the child's drawing $[c]$. The triple $(\mathfrak{ct}(c_1), \mathfrak{ct}(c_2), \mathfrak{ct}(c_2^{-1}c_1^{-1}))$ is called the passport of $[c]$.

Just as in [4], we often represent child's drawing of degree d by group homomorphisms $\psi : \mathsf{F}_2 \to S_d$ for which $\psi(\mathsf{F}_2)$ is a transitive subgroup of S_d. The assignment

$$\psi \mapsto \big(\psi(x), \psi(y)\big), \qquad x := x_{12}, \quad y := x_{23}$$

gives us the obvious bijection between such homomorphisms $\psi : \mathsf{F}_2 \to S_d$ and permutation pairs $(c_1, c_2) \in S_d \times S_d$ for which $\langle\, c_1, c_2 \,\rangle$ is a transitive subgroup of S_d.

It is clear that $\ker(\psi) \trianglelefteq \mathsf{F}_2$ depends only on the child's drawing $[\psi]$ but not on a particular choice of a representative $\psi : \mathsf{F}_2 \to S_d$.

Recall from [4] the following definition:

Definition 1.2 *Let $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and ψ be a homomorphism $\mathsf{F}_2 \to S_d$ that represents a child's drawing. We say that the child's drawing $[\psi]$ is **subordinate** to N (or **subordinate** to the equivalence relation $\sim_{\mathsf{N}}$) if*

$$\mathsf{N}_{\mathsf{F}_2} \leq \ker(\psi). \tag{1.15}$$

*If $[\psi]$ is subordinate to N, then we say that N **dominates** $[\psi]$. We denote by $\mathsf{Dessin}(\mathsf{N})$ the set of child's drawings subordinate to N.*

Let $\psi : \mathsf{F}_2 \to S_d$ be a representative of a child's drawing subordinate to N, (m, f) be a pair that represents an element in $\mathsf{GT}(\mathsf{N})$ and $\mathsf{K} := \ker(T^{\mathrm{PB}_4}_{m,f})$. We denote by $\psi^{(m,f)}$ the following homomorphism $\mathsf{F}_2 \to S_d$

$$\psi^{(m,f)}(x) := \psi(x^{2m+1}), \qquad \psi^{(m,f)}(y) := \psi(f^{-1}y^{2m+1}f). \tag{1.16}$$

Due to [4, Theorem 3.1],

- $\psi^{(m,f)}$ represents a child's drawing that is subordinate to K,
- $[\psi^{(m,f)}]$ does not depend on the choice of the representatives ψ and (m, f), and
- the assignments $\mathscr{A}^{sh}(\mathsf{N}) := \mathsf{N}$, $\mathscr{A}^{sh}([m, f])([\psi]) := [\psi^{(m,f)}]$ define a cofunctor $\mathscr{A}^{sh}$ from the category GTSh to the category Dessin.

In other words, the assignment $[\psi]^{[m,f]} := [\psi^{(m,f)}]$ defines a right action of GT-shadows on child's drawings.

It is relatively easy to see that the monodromy group of a child's drawing is an invariant of the action of GTSh. Due to [4, Theorem 3.16], the passport of a child's drawing is an invariant of the action of the subgroupoid $\mathsf{GTSh}^{\heartsuit}$ of charming GT-shadows.

1.5 Formats of various objects related to the package

1.5.1 Permutations and permutation groups

Just as C, Python starts counting at 0. For this reason, an element in S_d is a bijection $\{0, 1 \ldots, d-1\} \xrightarrow{\simeq} \{0, 1 \ldots, d-1\}$. In this package, permutations are represented by instances of the class *sympy.combinatorics.permutations.Permutation* (see [12]). The command *Permutation* from [12] is abbreviated to *permut*. For example, $permut(0, 4)(3, 2, 5)$ represents the permutation $(0, 4)(3, 2, 5)$ in $S_6 \cong S_{\{0,\ldots,5\}}$. The commands

- $permut(6)(0, 2, 5)(1, 4)$,
- $permut([2, 4, 5, 3, 1, 0, 6])$,
- $permut([[0, 2, 5], [1, 4], [6]])$ and
- $permut(((0, 2, 5), (1, 4), (6,)))$

return the same permutation $(0, 2, 5)(1, 4)$ in S_7.

Permutation groups are represented by instances of the class

sympy.combinatorics.perm_groups.PermutationGroup

For example, the command $SG(d)$ (resp. $AG(d)$, $CG(d)$, $DG(d)$) returns the symmetric group S_d (resp. the alternating group A_d, the cyclic group $\mathcal{Z}_d$, the dihedral group D_d of order $2d$).

For a tuple (or a list) t of permutations in S_d, the command $PG(t)$ returns the subgroup of S_d generated by elements of t. For a permutation $g \in S_d$, the command $PG(g)$ returns the cyclic subgroup $\langle\, g \,\rangle \leq S_d$.

For selected commands related to permutations and permutation groups, we refer the reader to Section 2.

1.5.2 Homomorphisms from finitely presented groups to permutation groups

For a finitely presented group $G = \langle\, X|R \,\rangle$, a group homomorphism $\psi : G \to S_d$ is represented by the tuple of permutations $(\psi(x) \mid x \in X)$ that satisfies all relations in R.

For example, a group homomorphism from PB_4 to S_d is represented by a tuple

$$(g_{12}, g_{23}, g_{13}, g_{14}, g_{24}, g_{34}) \in (S_d)^6 \tag{1.17}$$

such that

$$g_{23}g_{12}g_{13} \text{ commutes with } g_{12},\ g_{23},\ g_{13}, \tag{1.18}$$

$$g_{12}^{-1}g_{14}g_{12} = g_{14}g_{24}g_{14}g_{24}^{-1}g_{14}^{-1}, \qquad g_{12}^{-1}g_{24}g_{12} = g_{14}g_{24}g_{14}^{-1}, \qquad g_{12}^{-1}g_{34}g_{12} = g_{34}, \tag{1.19}$$

$$g_{23}^{-1}g_{14}g_{23} = g_{14}, \qquad g_{23}^{-1}g_{24}g_{23} = g_{24}g_{34}g_{24}g_{34}^{-1}g_{24}^{-1}, \qquad g_{23}^{-1}g_{34}g_{23} = g_{24}g_{34}g_{24}^{-1}. \tag{1.20}$$

$$g_{13}^{-1}g_{14}g_{13} = g_{14}g_{34}g_{14}g_{34}^{-1}g_{14}^{-1}, \quad g_{13}^{-1}g_{24}g_{13} = gg_{24}g^{-1}, \quad g_{13}^{-1}g_{34}g_{13} = g_{14}g_{34}g_{14}^{-1}, \tag{1.21}$$

where $g := g_{14}g_{34}g_{14}^{-1}g_{34}^{-1}$.

We also use tuples of permutations to represent finite index normal subgroups in a finitely presented group $G = \langle X|R \rangle$. For example, if $\psi : \mathrm{PB}_4 \to S_d$ is a homomorphism represented by tuple (1.17) then this tuple also represents the normal subgroup $\ker(\psi) \trianglelefteq \mathrm{PB}_4$.

Let k be the size of X for a finitely presented group $G = \langle X|R \rangle$. Note that any tuple t (of elements in S_d) of length k represents a group homomorphism ψ_t from the free group F_k on k generators to S_d. If t satisfies the relations of $G = \langle X|R \rangle$ then t also represents a group homomorphism from G to S_d. Under the bijection between finite index normal subgroups of G and finite index normal subgroups N of F_k that contain the kernel of the standard onto homomorphism $\mathsf{F}_k \to \langle X|R \rangle$, the normal subgroup $\mathsf{N}_t^G \trianglelefteq G$ represented by a tuple t corresponds to the normal subgroup $\mathsf{N}_t \trianglelefteq \mathsf{F}_k$ represented by the same tuple t. It is clear that $|\mathsf{F}_k : \mathsf{N}_t| = |G : \mathsf{N}_t^G|$ coincides with the order of the permutation group generated by elements of the tuple t.

Two different tuples of permutations may represent the same normal subgroup of $G = \langle X|R \rangle$. Let t (resp. tt) be a tuple of permutations in S_{d_1} (resp. in S_{d_2}) and $\psi_t : G \to S_{d_1}$ (resp. $\psi_{tt} : G \to S_{d_2}$) be the corresponding group homomorphism. Let t_{cap} be the tuple of elements in $S_{d_1} \times S_{d_2}$ that represents the homomorphism

$$\psi_{cap}(g) := \big(\psi_t(g), \psi_{tt}(g)\big) : G \to S_{d_1} \times S_{d_2}.$$

Since $\ker(\psi_{cap}) = \ker(\psi_t) \cap \ker(\psi_{tt})$, tuples t and tt represent the same normal subgroup of G if and only if

- the order of the permutation group generated by elements of t coincides with the order of the permutation group generated by elements of tt (i.e. $|G : \ker(\psi_t)| = |G : \ker(\psi_{tt})|$) and
- the order of the permutation group generated by elements of t_{cap} coincides with the order of the permutation group generated by elements of t (i.e. $|G : \ker(\psi_{cap})| = |G : \ker(\psi_t)|$).

These simple ideas are implemented in the definitions of the functions *cap*(,), *Nsubgrp_less_eq*(,), and *sameNsubgrp*(,) from the file *PaB.py*. For more details, please see Section 4.

Elements of F_2 are represented by tuples of 0's and 1's. For example, the tuple $w = (0, 0, 1, 0)$ represents the element x^2yx. Since we typically consider images of elements of F_2 in finite groups, we ignore x^{-1} and y^{-1}, i.e. we only consider words in x and y.

1.5.3 Formats for elements of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$

We use two formats for elements of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$:

- **tuple format**: an element $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ is represented by a group homomorphism $\psi : \mathrm{PB}_4 \to S_d$ such that $\ker(\psi) = \mathsf{N}$; the homomorphism ψ is, in turn, represented by a tuple of permutations (1.17) satisfying the relations (1.18), (1.19), (1.20) and (1.21).
- **object format**: an element $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ is represented by an instance of the class *Equiv*.

For example, if t is a tuple representing $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the command $Equiv(t)$ returns the instance of the class *Equiv* that represents N. Moreover, if E is an instance of the class *Equiv* representing $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the command $E.PB4$ returns a tuple representing N.

If an instance E of *Equiv* represents $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the command $E.PB3$ returns a tuple that represents the subgroup $\mathsf{N}_{\mathrm{PB}_3} \in \mathsf{NFI}_{\mathrm{PB}_3}(\mathrm{B}_3)$, the command $E.xy$ returns a tuple of two permutations that represents the subgroup $\mathsf{N}_{\mathsf{F}_2} := \mathsf{N}_{\mathrm{PB}_3} \cap \langle\, x_{12}, x_{23} \,\rangle$, the command $E.N0$ returns the index $N_{\mathrm{ord}} := |\mathrm{PB}_2 : \mathsf{N}_{\mathrm{PB}_2}|$.

Of course, $E.PB4$ (resp. $E.PB3$) also represents a group homomorphism $\mathrm{PB}_4 \to S_{d_4}$ (resp. a group homomorphism $\mathrm{PB}_3 \to S_{d_3}$). The commands $E.d4$ and $E.d3$ return the corresponding degrees d_4 and d_3.

1.5.4 Formats for GT-shadows

Given $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, we use two formats for elements of $\mathsf{GT}(\mathsf{N})$ (or candidates for elements of $\mathsf{GT}(\mathsf{N})$):

- **tuple format**: an element of $\mathsf{GT}(\mathsf{N})$ is represented by a tuple (w, m), where w is a tuple (of 0's and 1's) that represents an element of F_2 and m is a non-negative integer that represents an element of $\mathbb{Z}/N_{\mathrm{ord}}\mathbb{Z}$;
- **object format**: an element of $\mathsf{GT}(\mathsf{N})$ is represented by an instance of the class $GTsh$; for a tuple (w, m) and a tuple t that represents $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, $GTsh((w, m), t)$ is the instance of the class $GTsh$ that represents the GT-shadow corresponding to (w, m).

For example, given an instance T of the class $GTsh$ that represents a GT-shadow with the target N, the command $T.wm$ returns a representative (w, m) of this GT-shadow; the command $T.tar$ returns the instance of the class *Equiv* that represents N; $T.g$ returns a permutation g corresponding to the coset $w\mathsf{N}_{\mathsf{F}_2}$ (g belongs to a permutation group isomorphic to the quotient $\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$); finally, $T.cc\_ch$ returns the value of the virtual cyclotomic character.

1.5.5 Formats for child's drawings

For a positive integer d, we use two formats for (representatives of) child's drawings of degree d:

- **tuple format**: a child's drawing of degree d is represented by a tuple $c = (c_1, c_2)$ of permutations in S_d such that the subgroup $\langle\, c_1, c_2 \,\rangle$ is transitive;
- **object format**: a child's drawing of degree d is represented by an instance of the class *Dessin*.

For example, if c is a tuple that represents a child's drawing, then $Dessin(c)$ is an instance of the class $Dessin$ that represents the child's drawing $[c]$. If D is an instance of $Dessin$ that represents a child's drawing $\mathcal{D}$, then the command $D.pr$ returns a tuple c that represents $\mathcal{D}$, the command $D.full$ returns the permutation triple

$$(c_1, c_2, c_2^{-1}c_1^{-1}),$$

and the command $D.passport$ returns the passport of $\mathcal{D}$ in the format of a nested tuple. For instance, executing the lines:

Dessin((permut(1, 2, 3, 6, 5), permut(0, 3, 6, 1, 5, 4))).full
Dessin((permut(1, 2, 3, 6, 5), permut(0, 3, 6, 1, 5, 4))).passport

we get

(Permutation(1, 2, 3, 6, 5), Permutation(0, 3, 6, 1, 5, 4), Permutation(0, 4, 5, 3, 2, 6))

and

((5, 1, 1), (6, 1), (6, 1))

respectively.

Remark 1.3 For all timed procedures, the time is given in **minutes**.

1.6 Brief outline of the package. Examples of what we can do

The package consists of the auxiliary Python file *Aux.py* and the main Python file *PaB.py*. The key classes of *PaB.py* are *Equiv*, *GTsh* and *Dessin*. As we mentioned above, instances of *Equiv* represent compatible equivalence relations on $\mathsf{PaB}^{\leq 4}$, instances of *GTsh* represent (candidates for) GT-shadows; finally, instances of *Dessin* represents child's drawings. In this documentation, we will freely use the terminology and notational conventions from [5].

When we run *PaB.py*, a computer creates the following objects:

- *listE* is the list of compatible equivalence relations corresponding to 35 distinct elements

$$\mathsf{N}^{(0)},\ \mathsf{N}^{(1)},\ \ldots,\ \mathsf{N}^{(33)},\ \mathsf{N}^{(34)} \tag{1.22}$$

  of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$; the equivalence relations are given in the object format; *listE* is obtained from the storage file *subGrPB4_org*35; Table 1 shows basic information about these 35 selected elements of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$;

- *GTcharm_wm* is the (nested) list which contains charming GT-shadows whose targets are elements of *listE*, i.e. $len(GTcharm\_wm) = 35$ and $GTcharm\_wm[i]$ is the list of charming GT-shadows with the target $\mathsf{N}^{(i)}$; the GT-shadows are given in the tuple format; *GTcharm_wm* is extracted from the storage file *wm_list_charm*35;

- *GTall_wm* is the (nested) list which contains all (practical) GT-shadows whose targets are elements of the first 31 entries of *listE*, i.e. $len(GTall\_wm) = 31$ and $GTall\_wm[i]$ is the complete list of (practical) GT-shadows with the target $\mathsf{N}^{(i)}$; the GT-shadows are given in the tuple format; *GTall_wm* is extracted from the storage file *wm_list_all*31.

If you choose to upload $GTcharm$, you will also have the nested list of the charming GT-shadows whose targets are elements of $listE$, i.e. $len(GTcharm) = 35$ and $GTcharm[i]$ is the complete list of charming GT-shadows with the target $\mathsf{N}^{(i)}$. GT-shadows in $GTcharm[i]$ are given in the object format.

Here is more information about Table 1. For every $0 \leq i \leq 34$, the quotient $\mathsf{F}_2/\mathsf{N}^{(i)}_{\mathsf{F}_2}$ is **non-Abelian**. Table 1 shows (in the order from left to right) the number of $\mathsf{N}^{(i)}$, the index of $\mathsf{N}^{(i)}$ in PB_4, the index of $\mathsf{N}^{(i)}_{\mathsf{F}_2}$ in F_2, the order of the commutator subgroup $[\mathsf{F}_2/\mathsf{N}^{(i)}_{\mathsf{F}_2}, \mathsf{F}_2/\mathsf{N}^{(i)}_{\mathsf{F}_2}]$, $N^{(i)}_{\mathrm{ord}} := |\mathrm{PB}_2 : \mathsf{N}^{(i)}_{\mathrm{PB}_2}|$, the size of $\mathsf{GT}(\mathsf{N}^{(i)})$ (i.e. the total number of (practical) GT-shadows with the target $\mathsf{N}^{(i)}$) and the size of $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(i)})$. Finally, the last column indicates whether $\mathsf{N}^{(i)}$ is isolated or not. Note that, for $\mathsf{N}^{(33)}$ and $\mathsf{N}^{(34)}$, the exact numbers of (practical) GT-shadows is not known.

i	$\lvert\mathrm{PB}_4 : \mathsf{N}^{(i)}\rvert$	$\lvert\mathsf{F}_2 : \mathsf{N}^{(i)}_{\mathsf{F}_2}\rvert$	$\lvert[\mathsf{F}_2/\mathsf{N}^{(i)}_{\mathsf{F}_2}, \mathsf{F}_2/\mathsf{N}^{(i)}_{\mathsf{F}_2}]\rvert$	$N^{(i)}_{\mathrm{ord}}$	$\lvert\mathsf{GT}(\mathsf{N}^{(i)})\rvert$	$\lvert\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(i)})\rvert$	isolated?
0	8	16	2	4	4	4	True
1	8	16	2	4	8	4	True
2	12	36	4	3	18	6	True
3	21	63	7	3	36	12	False
4	21	63	7	3	36	12	False
5	24	288	8	6	72	12	True
6	24	144	4	6	72	12	True
7	48	144	4	6	72	12	True
8	60	1500	60	5	100	20	True
9	60	900	4	15	360	24	True
10	72	144	18	4	16	8	False
11	72	144	18	4	16	8	False
12	108	972	27	6	72	12	True
13	120	6000	60	10	400	40	True
14	147	441	49	3	216	72	True
15	168	8232	168	7	294	42	True
16	168	1344	168	4	64	32	False
17	168	1344	168	4	64	32	False
18	180	13500	60	15	600	40	True
19	216	7776	216	6	72	12	True
20	240	6000	60	10	400	40	True
21	324	8748	108	9	486	54	True
22	504	40824	504	9	486	54	True
23	504	24696	504	7	294	42	True
24	648	1296	162	4	32	16	True
25	720	54000	240	15	1800	120	True
26	1512	40824	504	9	486	54	False
27	1512	40824	504	9	486	54	False
28	2520	63000	2520	5	200	40	True
29	2520	45360	2520	6	144	48	True
30	28224	225792	28224	4	512	256	True
31	181440	8890560	181440	7	588	84	True
32	181440	9072000	181440	10	800	160	True
33	181440	40824000	181440	15	$\geq$ **1800**	120	True
34	762048	20575296	254016	9	$\geq$ **4374**	486	True

Table 1: The basic information about selected 35 compatible equivalence relations

Executing the lines:

for i in range(35):
print(i,' ', listE[i].ind4(), ' ', listE[i].indF2(), ' ',
listE[i].commF2().order(),' ', listE[i].N0,' ', len(GTcharm[i]))

we get selected columns of Table 1: the number of the entry $\mathsf{N}^{(i)}$, $|\mathrm{PB}_4 : \mathsf{N}^{(i)}|$, $|\mathsf{F}_2 : \mathsf{N}^{(i)}_{\mathsf{F}_2}|$, the order of the commutator subgroup $[\mathsf{F}_2/\mathsf{N}^{(i)}_{\mathsf{F}_2}, \mathsf{F}_2/\mathsf{N}^{(i)}_{\mathsf{F}_2}]$, $N^{(i)}_{\mathrm{ord}} := |\mathrm{PB}_2 : \mathsf{N}^{(i)}_{\mathrm{PB}_2}|$ and the number of charming GT-shadows with the target $\mathsf{N}^{(i)}$.

For example, $\mathsf{N}^{(19)}$ is called the *Philadelphia subgroup* of PB_4 and it is represented by the tuple of 6 permutations in S_9:

$$\begin{array}{lll} g_{12} := (1,3,2)(4,6,5), & g_{23} := (1,4,9)(2,7,6), & g_{13} := (1,7,5)(3,6,9), \\ g_{14} := (2,6,7)(3,8,5), & g_{24} := (1,8,6)(3,4,7), & g_{34} := (1,2,3)(7,9,8). \end{array} \tag{1.23}$$

According to Table 1, $\mathsf{N}^{(19)}$ is an isolated element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$. Executing the command $\{$*T.settled() for T in GTcharm[19]*$\}$, we get $\{\mathsf{True}\}$. This confirms that $\mathsf{N}^{(19)}$ is indeed an isolated element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$.

Executing the command $\{$*T.settled() for T in GTcharm[16]*$\}$, we get $\{\mathsf{False}, \mathsf{True}\}$. This confirms that $\mathsf{N}^{(16)}$ is not isolated.

In fact, executing the lines:

GT16_not_settled =[T for T in GTcharm[16] if not T.settled()]
len(GT16_not_settled)

we see that exactly half of the 32 charming GT-shadows with the target $\mathsf{N}^{(16)}$ are not settled and the remaining 16 charming GT-shadows are settled.

Executing the line:

$\{$*T.src()==listE[17] for T in GT16_not_settled*$\}$

we get $\{\mathsf{True}\}$. So we see that, for every GT-shadow T in the list $GT16\_not\_settled$, the source of T is $\mathsf{N}^{(17)}$.

Thus the connected component of $\mathsf{N}^{(16)}$ in the groupoid $\mathsf{GTSh}^{\heartsuit}$ has exactly two (isomorphic) objects: $\mathsf{N}^{(16)}$ and $\mathsf{N}^{(17)}$. There are exactly 16 elements in $\mathsf{GTSh}^{\heartsuit}(\mathsf{N}^{(16)}, \mathsf{N}^{(16)})$ and exactly 16 elements in $\mathsf{GTSh}^{\heartsuit}(\mathsf{N}^{(17)}, \mathsf{N}^{(16)})$.

Executing the command $\{$*p for p in comb(range(35),2) if listE[p[1]].finer(listE[p[0]])*$\}$, we get:

$$\begin{array}{c} \big\{(1,10),(1,11),(1,24),(2,5),(2,6),(2,7),(2,9),(2,19),(2,21), \\ (2,25),(3,14),(4,14),(5,19),(8,13),(8,18),(8,20),(8,25),(10,24), \\ (11,24),(16,30),(17,30),(18,25),(26,34),(27,34)\}. \end{array} \tag{1.24}$$

In other words, we get the set of all pairs (i,j) with $0 \le i < j \le 34$ such that $\mathsf{N}^{(j)} \subset \mathsf{N}^{(i)}$. For example, since $(8,20)$ belongs to the above set of pairs, $\mathsf{N}^{(20)} \subset \mathsf{N}^{(8)}$. Since $(5,20)$ is not in the above list, $\mathsf{N}^{(20)} \not\subset \mathsf{N}^{(5)}$.

Executing the command *furusho_test1(listE[5])* (resp. *furusho_test_comm1(listE[5])*), we get False (resp. True). This shows that element $\mathsf{N}^{(5)}$ does not satisfy the strong Furusho

property (see Property 5.1), however $\mathsf{N}^{(5)}$ satisfies the weak Furusho property (see Property 5.2).

The command *furusho_test1(listE[24])* returns True, which means that element $\mathsf{N}^{(24)}$ satisfies the strong Furusho property and hence $\mathsf{N}^{(24)}$ also satisfies the weak Furusho property.

The command *furusho_test_comm1(listE[25])* returns False which means that element $\mathsf{N}^{(25)}$ does not satisfy the weak Furusho property and hence $\mathsf{N}^{(25)}$ does not satisfy the strong Furusho property.

Executing the lines:

c8=(permut(7)(0,1,2)(3,4,5),permut(0,7,4)(1,3,6))
D8=Dessin(c8)

we create a child's drawing $\mathcal{D}_{8,0}$ of degree 8 and genus 0 represented by the permutation triple[3]

$$\big((1,2,3)(4,5,6),\ (1,8,5)(2,4,7),\ (1,3,7,4,6,8)(2,5)\big). \tag{1.25}$$

Executing the command *D8.passport*, we see that the passport of $\mathcal{D}_{8,0}$ is

$$\big((3,3,1,1),\ (3,3,1,1),\ (6,2)\big).$$

Executing the command *D8.is_Galois( )*, we get False. Hence the child's drawing $\mathcal{D}_{8,0}$ is not Galois[4].

Executing the command {*i for i in range(35) if listE[i].subord(c8)*}, we get $\{5, 19\}$. This means that $\mathcal{D}_{8,0}$ is subordinate to $\mathsf{N}^{(5)}$ and $\mathsf{N}^{(19)}$.

Executing the command {*Dessin(T.act(c8))==D8 for T in GTcharm[5]*}, we get {True}. This means that the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(5)})(\mathcal{D}_{8,0})$ is a singleton. Hence, for every element g of the absolute Galois group $G_{\mathbb{Q}}$ of rationals, $\mathcal{D}_{8,0}^{g} = \mathcal{D}_{8,0}$.

Organization of the documentation. In addition to the introduction, the documentation has 7 sections and two appendices. In Section 2, we listed selected commands related to permutations and permutation groups. In Section 3, we described various auxiliary functions. These functions are defined in the Python file *Aux.py*. Section 4 is devoted to functions, generators, and classes defined in the Python file *PaB.py*. In this section, we described a generator for GT-shadows and a generator for charming GT-shadows (these are the generators *gener_GT_sh( )* and *gener_GT_charm( )*, respectively). In Section 4, we also described the classes *Equiv*, *GTsh*, *Dessin* and various functions for working with child's drawings. In brief Subsection 4.1.1, we described how we obtained 35 selected elements of the poset $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ (see (1.22)).

In Section 5, we presented more examples of working with this package. We described possible ways of looking for charming GT-shadows that are fake. We discussed versions of the Furusho property. Finally, we presented more examples of using the commands related to child's drawings and to the action of the groupoid $\mathsf{GTSh}^{\heartsuit}$ on child's drawings: we showed how to generate all non-Abelian child's drawings of degrees 3, 4 and 5, and computed $\mathsf{GTSh}^{\heartsuit}$-orbits of selected child's drawings of degrees ≤ 5. We also showed how to produce child's drawings subordinate to a given element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$.

[3]This example is also stored in the file *dde8E5E19*.
[4]This is also clear from the passport of $\mathcal{D}_{8,0}$.

Section 6 contains detailed descriptions of all storage files. In Section 7, we described indirect ways of testing various functions, methods and outputs. In Appendix A, we briefly reviewed the braid groups B_n, PB_n and presented some calculations relevant to the package. Finally, in Appendix B, we proved an auxiliary statement that justifies certain lines of code in *PaB.py*.

Contributors: The following people contributed to this software[5] package: Chelsea Zackey, Aidan Lorenz, Khanh Le and V.A.D.

Acknowledgement. V.A.D. is thankful to John Voight for many clarifying discussions and for his patience with some (probably naive) questions. Some computational tricks V.A.D. learned from John Voight are used in the parts of the package related to child's drawings. V.A.D. is thankful to Leila Schneps for stimulating discussions and her unbounded enthusiasm about GT-shadows. V.A.D. is thankful to Sergey Plyasunov and Justin Y. Shi for showing him how to use the Python module pickle. V.A.D. is thankful to the Temple University Honors Program and the Undergraduate Research Program of the College of Science and Technology of Temple University for their active support of undergraduate researchers. V.A.D. acknowledges a partial support from NSF grant DMS-1501001 and a support from Temple University in the form of 2021 Summer Research Award.

2 Selected commands related to permutations and permutation groups

Here is the list of selected commands related to permutations and permutation groups from the Python library SymPy [11]:

- for a permutation g, $g.size$ returns its degree; for instance, the command $permut(7)(0,4,1)(2,3).size$ returns 8 because the permutation $(7)(0,4,1)(2,3)$ belongs to S_8;
- for a permutation $g \in S_d$ and $i \in \{0,1,\dots,d-1\}$, the command i^g returns $g(i)$; for instance, for $g = permut(7)(0,4,1)(2,3)$, the command

  $$[i^g \ for \ i \ in \ range(8)]$$

  returns the list $[4,0,3,2,1,5,6,7]$;
- permutations in S_d act on the set $\{1,2,\dots,d\}$ from the right; so, for $g,h \in S_d$, the command $g * h$ returns the composition $h \circ g$;
- for a permutation g and an integer n, the command $g{**}n$ returns the permutation g^n; for instance, $g{**}(-1)$ returns the inverse of the permutation g;
- for a permutation g, the command $g.cyclic\_form$ returns the list of cycles of length ≥ 2; each cycle is represented by a list; for instance, the command $permut(18)(0,2,5)(1,4).cyclic\_form$ returns $[[0,2,5],[1,4]]$;
- for a permutation g, the command $g.order()$ returns the order of g; for instance, the command $permut(7)(0,4,1)(2,3).order()$ returns 6;

[5]The names of contributors are given in the reverse alphabetic order.

- SymPy has a built-in bijection between S_d and the set $\{0, 1, \ldots, d! - 1\}$: for a permutation g in S_d the command $g.rank()$ returns the value of this bijection (i.e. its unique hashtag in $\{0, 1, \ldots, d! - 1\}$); for example, the hashtag $permut(d-1).rank()$ of the identity element $permut(d-1)$ is 0 for every $d \geq 1$; it is often more efficient to store hashtags than the corresponding instances of the class *sympy.combinatorics.permutations.Permutation*;

- for a permutation group G, the command $G.order()$ returns the order of G; for instance, the command $AG(5).order()$ returns 60 (i.e. the order of the alternating group A_5);

- for a permutation group $G \leq S_d$, the command $G.degree$ returns d;

- for a permutation group G, the command $G.elements$ returns the set of elements of G; for instance, the command $AG(3).elements$ returns

$$\{Permutation(2), Permutation(0,1,2), Permutation(0,2,1)\},$$

  i.e. the set of elements of the alternating group A_3;

- for a permutation group G, the command $G.is\_abelian$ returns True if G is Abelian, otherwise it returns False; for instance, the command $AG(3).is\_abelian$ returns True and the command $SG(3).is\_abelian$ returns False;

- for a permutation group $G \leq S_d$, the command $G.is\_transitive()$ returns True if G is a transitive subgroup of S_d; otherwise it returns False;

- for two permutation groups H and G, the command $H.is\_subgroup(G)$ returns True if $H \leq G$, otherwise it returns False;

- for a subgroup H of a permutation group G, the command $H.is\_normal(G)$ returns True if H is a normal subgroup of G, otherwise it returns False; for instance, the command $AG(5).is\_normal(SG(5))$ returns True, while the command $DG(5).is\_normal(SG(5))$ returns False;

- for a subgroup H of a permutation group G, the command $G.commutator(G, H)$ returns the commutator subgroup $[G, H]$; for instance, the command $DG(5).commutator(DG(5), DG(5))$ returns the cyclic subgroup of S_5 generated by $(0, 1, 2, 3, 4)$, while the command $SG(5).commutator(SG(5), SG(5))$ returns the alternating group A_5;

- for a subgroup H of a permutation group G, the command

$$G.coset\_transversal(H)$$

  returns a transversal of the left cosets of G by H (as a list); for instance, if $H = PG(permut(1,2))$, then the command

$$SG(3).coset\_transversal(H)$$

  returns

$$[Permutation(2),\ Permutation(0,2),\ Permutation(2)(0,1)],$$

  i.e. $S_3 = H \sqcup (0,2)H \sqcup (0,1)H$;

- for a subgroup H of a permutation group G and $g \in G$, the command $G.\_coset\_representative(g, H)$ returns the unique representative of the left coset gH in accordance with $G.coset\_transversal(H)$; for instance the command

$$\{G.\_coset\_representative(g, H) == g \ \ for \ \ g \ \ in \ \ G.coset\_transversal(H)\}$$

returns $\{\mathsf{True}\}$.

For more commands and examples, please see [12] and [13].

3 Selected functions of *Aux.py*

Here is the list of selected functions from *Aux.py*:

- for a permutation g, $display\_perm(g)$ prints the nested tuple whose entries are cycles of the permutation; the shift $i \mapsto i + 1$ is incorporated; for example, $display\_perm(permut(0, 4)(3, 2, 5))$ prints the nested tuple $((1, 5), (3, 6, 4))$; note that cycles of length 1 are not shown; in particular, the commands $display\_perm(permut(0, 4)(3, 2, 5))$ and $display\_perm(permut(18)(0, 4)(3, 2, 5))$ print the same nested tuple;
- for a list L of iterables, $cart\_pr(L)$ is a generator of all elements of the Cartesian product $L[0] \times L[1] \times L[2] \times \ldots$; for example, $cart\_pr([[5], [2, 1], [3, 4], [6]])$ generates the 4 tuples $(5, 2, 3, 6)$, $(5, 2, 4, 6)$, $(5, 1, 3, 6)$, $(5, 1, 4, 6)$;
- lcm (resp. $lcm3$) returns the least common multiple of two (resp. three) integers;
- for $n \in \mathbb{Z}_{\geq 2}$, $m\_units(n)$ generates all integers between 0 and $n - 1$ such that $2m + 1$ is a unit of the ring $\mathbb{Z}/n\mathbb{Z}$;
- for a positive integer d, $prm(d)$ returns a random permutation of degree d;
- $ran(n)$ returns the tuple $(0, 1, ..., n - 1)$;
- $split(t)$ generates all possible splittings of a tuple t; for instance, $split((1, 2, 3))$ generates $((1, 2, 3),)$, $((1, 2), (3,))$ and $((1,), (2,), (3,))$; for a tuple t of length n, the number of outputs of $split(t)$ is the total number of partitions of n;
- for permutations s and t of the same degree, $comp(s, t)$ returns the composition of two permutations s and t in the standard order, i.e. t acts first and s acts second;
- for a tuple (or a list) of permutations t (of the same degree), the command $compAll(t)$ returns the consecutive composition of all permutations in t;
- for a positive integer d, $one(d)$ returns the identity element in S_d;
- $is\_id(g)$ returns True if the permutation g is the identity element, otherwise False;
- not_id is the negation of is_id;
- $concat(g, h)$ implements the standard homomorphism $S_n \times S_k \to S_{n+k}$; for instance, the command $concat(permut(0, 2)(3, 4), permut(0, 2, 1))$ returns the permutation

$$(0, 2)(3, 4)(5, 7, 6);$$

- for a tuple t of permutations in S_d and a tuple tt of permutations in S_n, *concat_tup* returns the tuple whose entries are obtained by "concatenating" the corresponding entries of t and tt;

- $commut(g, h)$ returns the group commutator $ghg^{-1}h^{-1}$ of permutations g and h of the same degree;

- for permutations g, h of the same degree, $conj(g, h)$ returns the permutation ghg^{-1};

- *conj_tup* is the extension of the command *conj* to the case when the second argument is a tuple of permutations, i.e., for a permutation $g \in S_d$ and a tuple t of permutations of degree d, the command $conj\_tup(g, t)$ returns the tuple

$$tuple(conj(g, h) \;\; for \;\; h \;\; in \;\; t);$$

- let t be a tuple of permutations in S_d, G be the subgroup of S_d generated by elements of t and X be the set of indices $i \in \{0, 1, \ldots, d-1\}$ for which the G-orbit of i is **not** a singleton; then every permutation g in t is uniquely determined by the corresponding "trimmed" permutation in $S_X \cong S_q$, where q is the size of X; the function $trim\_perms(t)$ returns the tuple of the corresponding trimmed permutations; of course, the permutation groups $PG(trim\_perms(t))$ and $PG(t)$ are isomorphic; this command was used to "simplify" tuples of permutations that represent elements of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$;

- for a tuple t of three permutations s_1, s_2, s_3 (of the same degree), $relB4(t)$ returns True if the tuple t represents a homomorphism from B_4 to a symmetric group; otherwise, it returns False;

- for a tuple t representing a group homomorphism $\varphi : \mathrm{B}_4 \to S_d$, $restr\_PB4(t)$ returns the tuple $(g_{12}, g_{23}, g_{13}, g_{14}, g_{24}, g_{34})$ that represents the homomorphism $\varphi\big|_{\mathrm{PB}_4} : \mathrm{PB}_4 \to S_d$;

- for $d \in \mathbb{Z}_{\geq 1}$, $generB4(d, \mathrm{timed})$ generates all group homomorphisms from B_4 to S_d up to conjugation by elements of S_d; $generB4\_A(d, \mathrm{timed})$ is another version of this generator; $generB4\_A$ works faster than $generB4$ for $d > 5$;

- *dict2tup*, *tup2dict* allow us to reformat partitions from the dictionary format to the tuple format; for instance, the command $tup2dict((5, 3, 3, 1, 1))$ returns the dictionary $\{1 : 2, 3 : 2, 5 : 1\}$ and the command $dict2tup(\{1 : 2, 3 : 2, 5 : 1\})$ returns the tuple $(5, 3, 3, 1, 1)$;

- for tuples t and tt of permutations in S_d of the same length, $are\_conj\_easy(t, tt)$ returns True if there exists g in S_d such that $conj\_tup(g)(t)$ coincides with tt; otherwise, it returns False;

- for a non-increasing tuple p of positive integers $(k_1, k_2, \ldots)$, $toCanPerm(p)$ returns the permutation $(0, 1, \ldots, k_1 - 1)(k_1, \ldots, k_1 + k_2 - 1), \ldots$;

- for a permutation group G and subgroups $H1, H2$ of G, $double\_coset\_reps(H1, G, H2)$ generates representatives of double cosets in $H1 \backslash G / H2$; exactly one representative for each double coset.

4 Classes, functions and methods of *PaB.py*

Here is the list of selected functions and generators from the Python file *PaB.py*:

- for a tuple t of six permutations in S_d, $relPB4(t)$ returns True if t represents a homomorphism $\mathrm{PB}_4 \to S_d$; otherwise, the function returns False;
- for a tuple t that represents a homomorphism $\mathrm{PB}_4 \to S_d$, the command $cenPB4(t)$ returns the image of the generator

$$c_4 = x_{14}x_{24}x_{34}x_{12}x_{13}x_{23}$$

  of the center $\mathcal{Z}(\mathrm{PB}_4)$ of PB_4; see Proposition A.2 in Appendix A.1; this function was used for testing $relPB4(\ )$ indirectly;
- for tuples x, y of permutations representing homomorphisms from a free group on $len(x)$ generators to symmetric groups, the command, $cap(x, y)$ returns the tuple of permutations that represents the intersection of the kernels of the corresponding homomorphisms;
- for tuples x, y of permutations representing homomorphisms from a free group on $len(x)$ generators to symmetric groups, the command, $Nsubgrp\_less\_eq(x, y)$ returns True if the kernel of the homomorphism corresponding to x is contained in the kernel of the homomorphism corresponding to y; otherwise, the command returns False; the function is often applied to tuples that represent finite index normal subgroups of a finitely presented group (say, PB_n or B_n); for this function, we tacitly assume that $len(x) == len(y)$;
- for tuples x, y of permutations representing homomorphisms from a free group on $len(x)$ generators to symmetric groups, the command $sameNsubgrp(x, y)$ returns True if the kernels of the corresponding homomorphisms coincide; otherwise, the command $sameNsubgrp(x, y)$ returns False; the function is often applied to tuples that represent finite index normal subgroups of a finitely presented group (say, PB_n or B_n); for this function, we tacitly assume that $len(x) == len(y)$;
- the functions $fi1\_23\_4$, $fi123$, $fi12\_3\_4$, $fi1\_2\_34$, $fi234$ are related to the homomorphisms $\mathrm{PB}_3 \to \mathrm{PB}_4$ defined in (1.3); for instance, for a tuple t that represents a homomorphism $\psi_t : \mathrm{PB}_4 \to S_d$ the command $fi1\_23\_4(t)$ returns the tuple that represents the homomorphism

$$\psi_t \circ \varphi_{1,23,4} : \mathrm{PB}_3 \to S_d\,;$$

- similarly, the functions $fi12$, $fi12\_3$, $fi1\_23$, $fi23$ are related to the homomorphisms $\mathrm{PB}_3 \to \mathrm{PB}_4$ defined in (1.4); for instance, for a tuple t that represents a homomorphism $\psi_t : \mathrm{PB}_3 \to S_d$ the command $fi1\_23(t)$ returns a permutation that represents the homomorphism[6]

$$\psi_t \circ \varphi_{1,23} : \mathrm{PB}_2 \to S_d\,;$$

- for a tuple t that represents an element $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the command $N\_PB3(t)$ returns a tuple that represents $\mathsf{N}_{\mathrm{PB}_3} \in \mathsf{NFI}_{\mathrm{PB}_3}(\mathrm{B}_3)$ defined in (1.1);

[6] Recall that PB_2 is an infinite cyclic group generated by x_{12}.

- for a tuple x that represents an element $\mathsf{N}_{\mathrm{PB}_3} \in \mathsf{NFI}_{\mathrm{PB}_3}(\mathrm{B}_3)$, the command $Nord(x)$ returns the index $|\mathrm{PB}_2 : \mathsf{N}_{\mathrm{PB}_2}|$ of $\mathsf{N}_{\mathrm{PB}_2}$, where $\mathsf{N}_{\mathrm{PB}_2}$ is defined in (1.2);

- for a tuple $wm = (w, m)$ and a tuple t of permutations that represents $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the command $penta(wm, t)$ returns True if (w, m) satisfies pentagon relation (1.8) modulo N; otherwise $penta(wm, t)$ returns False; here w is a tuple of 0's and 1's that represents an element of F_2 and m is an integer;

- for a tuple $wm = (w, m)$ and a tuple tt of permutations that represents $\mathsf{K} \in \mathsf{NFI}_{\mathrm{PB}_3}(\mathrm{B}_3)$, the command $hexa1(wm, tt)$ returns True if (w, m) satisfies the first hexagon relation (see (1.6)) modulo K; otherwise $hexa1(wm, tt)$ returns False; as above, w is a tuple of 0's and 1's that represents an element of F_2 and m is an integer; please see Proposition B.1 in Appendix B for the explanation of line 552 in *PaB.py*

$$tup = (x23 * *m, fxy, x12 * *m, fxz\_inv, z * *m, fyz);$$

- for a tuple $wm = (w, m)$ and a tuple tt of permutations that represents $\mathsf{K} \in \mathsf{NFI}_{\mathrm{PB}_3}(\mathrm{B}_3)$, the command $hexa2(wm, tt)$ returns True if (w, m) satisfies the second hexagon relation (see (1.7)) modulo K; otherwise $hexa2(wm, tt)$ returns False; as above, w is a tuple of 0's and 1's that represents an element of F_2 and m is an integer; please see Proposition B.1 in Appendix B for the explanation of line 569 in *PaB.py*

$$tup = (fux\_inv, x12 * *m, fxy\_inv, x23 * *m, fuy, u * *m);$$

- for a (possibly empty) tuple w of elements in $\{0, 1, \ldots, q-1\}$ and a tuple t that represents a group homomorphism from F_q to S_d, the command $w2g(w, t)$ returns the value $\varphi(w)$ in S_d; for instance, the command $w2g((0, 0, 1), (permut(0, 1, 2), permut(1, 2)))$ returns $Permutation(0, 2)$, i.e. the product $(0, 1, 2) \cdot (0, 1, 2) \cdot (1, 2)$; the command $w2g(( ), t)$ returns the identity element of S_d; if q does not coincide with the length of t, the command will *not* work;

- for a tuple tt representing a homomorphism $\varphi : \mathsf{F}_2 \to S_d$, $generWF2(tt, \text{timed=None})$ generates all words in F_2 corresponding to distinct elements of the permutation group $\varphi(\mathsf{F}_2) \leq S_d$; for each element $g \in \varphi(\mathsf{F}_2)$, $generWF2(tt, \text{timed=None})$ yields exactly one $w \in \mathsf{F}_2$ such that $\varphi(w) = g$; at each iteration, this generator uses a list W of words in F_2 of a fixed length and a list L of the corresponding permutations in $\varphi(\mathsf{F}_2)$; for every word $w \in W$, the generator tests whether $\varphi(w + (0,))$ (resp. $\varphi(w + (1,))$) is a new permutation of $\varphi(\mathsf{F}_2)$; if this is the case, the word $w + (0,)$ (resp. $w + (1,)$) is appended to $Wnew$ and the permutation $\varphi(w + (0,))$ (resp. $\varphi(w + (1,))$) is appended to $Lnew$; the hashtag of each new permutation is appended to the list G; at the end of each iteration, W becomes $Wnew$ and L becomes $Lnew$; if the second argument of $generWF2(,)$ is True, the generator prints the status update in the form

  time elapsed (in minutes) the length of the list G;

  the last status update for $generWF2(tt, \mathsf{True})$ is

  time elapsed (in minutes) the order of the group $\varphi(\mathsf{F}_2)$;

- for a tuple $t = (g_0, g_1)$ representing a homomorphism $\varphi : \mathsf{F}_2 \to S_d$, the function $generWComm(t, \text{timed=None})$ generates words in F_2 that represent distinct elements of the commutator subgroup $[\varphi(\mathsf{F}_2), \varphi(\mathsf{F}_2)]$ of the permutation group $\varphi(\mathsf{F}_2)$; the generator goes through all words in F_2 of the form $x^{k_1}y^{t_1}x^{k_2}y^{t_2}\dots$ such that

$$k_1 + k_2 + \dots \equiv 0 \mod \text{ord}(g_0) \qquad \text{and} \qquad t_1 + t_2 + \dots \equiv 0 \mod \text{ord}(g_1);$$

  if the second variable of $generWComm(\ ,\ )$ is True, the generator prints the update every time it stores the next 1000 hashtags of new permutations;

- for a tuple t representing $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, $gener\_GT\_pr(t)$ generates all GT-pairs with the target N satisfying the condition $\gcd(2m+1, N_{\text{ord}}) = 1$; the outputs are in the tuple format;

- if wm is a tuple that represents a GT-pair with the target $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and tt is a tuple that represents $\mathsf{N}_{\mathrm{PB}_3}$, then the command $sourcePB3(wm, tt)$ returns a tuple that represents

$$\ker(\mathrm{PB}_3 \xrightarrow{T^{\mathrm{PB}_3}_{m,f}} \mathrm{PB}_3/\mathsf{N}) \trianglelefteq \mathrm{PB}_3\,;$$

- if wm is a tuple that represents a GT-pair with the target $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and tt is a tuple that represents N, then the command $sourcePB4(wm, tt)$ returns a tuple that represents

$$\ker(\mathrm{PB}_4 \xrightarrow{T^{\mathrm{PB}_4}_{m,f}} \mathrm{PB}_4/\mathsf{N}) \trianglelefteq \mathrm{PB}_4\,;$$

- for a tuple t representing $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, $gener\_GT\_sh(t)$ generates all GT-shadows with the target N; the outputs are in the tuple format;

- for a tuple t representing $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, $gener\_GT\_charm(t)$ generates all charming GT-shadows with the target N; the outputs are in the tuple format.

4.1 The class *Equiv*. Its attributes and methods

The class *Equiv* has exactly one instance variable and this instance variable is a tuple of 6 permutations. Two instances E and EE of *Equiv* are equal if and only if the corresponding (normal) subgroups of PB_4 coincide.

In the description of attributes and methods of *Equiv* given below, N denotes the element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ represented by instance "$self$" of the class *Equiv*. This class has the following data attributes:

- $self.PB4$ is the instance variable, i.e. $self.PB4$ is a tuple of 6 permutations that represents $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$;

- $self.PB3$ is a tuple of 3 permutations that represents $\mathsf{N}_{\mathrm{PB}_3} \in \mathsf{NFI}_{\mathrm{PB}_3}(\mathrm{B}_3)$;

- $self.N0$ is N_{ord}, i.e. the index $|\mathrm{PB}_2 : \mathsf{N}_{\mathrm{PB}_2}|$;

- $self.xy$ is a tuple of two permutations that represents $\mathsf{N}_{\mathsf{F}_2} := \mathsf{F}_2 \cap \mathsf{N}_{\mathrm{PB}_3} \trianglelefteq \mathsf{F}_2$;

- $self.d4$ is the degree of permutations in $self.PB4$;

- $self.d3$ is the degree of permutations in $self.PB3$.

The class $Equiv$ has the following methods:

- $self.ind4()$ returns the index $|\mathrm{PB}_4 : \mathsf{N}|$;
- $self.ind3()$ returns the index $|\mathrm{PB}_3 : \mathsf{N}_{\mathrm{PB}_3}|$;
- $self.indF2()$ returns the index $|\mathsf{F}_2 : \mathsf{N}_{\mathsf{F}_2}|$;
- $self.commF2()$ returns the commutator subgroup of a permutation group isomorphic to $\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$; the output is an instance of the class *sympy.combinatorics.perm_groups.PermutationGroup*; for examples, the command $self.commF2().order()$ returns the order of the commutator subgroup $[\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}, \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}]$;
- $self.relPB4()$ returns True if the tuple $self.PB4$ indeed represents a homomorphism from PB_4 to a permutation group; otherwise the command returns False;
- for an instance E of $Equiv$, the command $self.finer(E)$ returns True if $self$ is a finer equivalence relation than E (i.e. if $\mathsf{N} \leq \mathsf{N}_E$, where N_E is the element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ represented by E); otherwise, the command returns False;
- for an instance E of the class $Equiv$, the command $self.cap(E)$ returns an instance of $Equiv$ that represents $\mathsf{N} \cap \mathsf{N}_E$; here N_E is the element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ represented by E;
- for a permutation pair c that represents a child's drawing, the command $self.subord(c)$ returns True if the child's drawing is subordinate to the equivalence relation "$self$", i.e. if $\mathsf{N}_{\mathsf{F}_2}$ is a subgroup of the kernel of the homomorphism from F_2 corresponding to c; otherwise, the command returns False.

4.1.1 How we obtained 35 selected elements in $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$

Recall that $listE$ consists of 35 instances of the class $Equiv$ and elements of $listE$ correspond to 35 selected element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ (see (1.22)). The procedure $fishingE\_nonAb(\ ,\ ,\ )$ (see line 947 in *PaB.py*) was used to produce most of elements of $listE$. We will now explain what the procedure $fishingE\_nonAb$ does.

Let L be a (possibly empty) list of instances of the class $Equiv$, d be a positive integer and $timed \in \{\mathsf{False}, \mathsf{True}\}$. The procedure $fishingE\_nonAb(L, d, timed)$ looks through (conjugacy classes of) group homomorphisms $\varphi : \mathrm{B}_4 \to S_d$ and "restricts" them to PB_4. Let us set $\mathsf{N} := \ker(\varphi\big|_{\mathrm{PB}_4})$. If $|\mathrm{PB}_4 : \mathsf{N}| \geq 6$, the finite group $\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$ is non-Abelian and the equivalence relation corresponding to N is not in L, then the procedure appends this equivalence relation to L. As it runs, the procedure prints updates. If $timed$ is True, then, at the very end, the procedure will print the time elapsed (in minutes).

To produce most of elements of $listE$, we used $fishingE\_nonAb$ for $d = 4, 5, 6, 7, 8$. We also let a compute work for day for $d = 9$ to get more examples of elements $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ for which $\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$ is non-Abelian. Elements $\mathsf{N}^{(14)}, \mathsf{N}^{(24)}, \mathsf{N}^{(30)}, \mathsf{N}^{(34)}$ were not produced with the help of $fishingE\_nonAb$. They were produced by intersecting already obtained subgroups of PB_4. More precisely,

$$\mathsf{N}^{(14)} := \mathsf{N}^{(3)} \cap \mathsf{N}^{(4)}, \quad \mathsf{N}^{(24)} := \mathsf{N}^{(10)} \cap \mathsf{N}^{(11)}, \quad \mathsf{N}^{(30)} := \mathsf{N}^{(16)} \cap \mathsf{N}^{(17)}, \quad \mathsf{N}^{(34)} := \mathsf{N}^{(26)} \cap \mathsf{N}^{(27)}.$$

See the beginning of [5, Section 4]. As we see in Table 1, $listE$ is sorted by the index $|\mathrm{PB}_4 : \mathsf{N}|$.

We encourage the reader to execute these lines:

Test = [ ]
fishingE_nonAb(Test,6,True)

After that, executing the lines:

len(Test)
{E in listE for E in Test}

we get 2 and {True}. In other words, for $d = 6$, the procedure $fishingE\_nonAb$ gives us two elements of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and they are already in $listE$.

After executing the above commands, the reader may also try to run $fishingE\_nonAb$ for $d = 7$.

It is possible that there are elements $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ represented by group homomorphisms $\mathrm{PB}_4 \to S_9$ that do not belong to the list in (1.22) and such that $\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$ is non-Abelian.

4.2 The class $GTsh$. Its attributes and methods

As we mentioned above, the class $GTsh$ has two instance variables: (w, m) and t. w is a tuple of 0's and 1's and it represents an element of F_2, m is a non-negative integer and t is a tuple of 6 permutations; t represents an element $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$. Two instances T and TT of $GTsh$ represent the same GT-shadows if and only if the command $T == TT$ returns True.

In the description of attributes and methods of $GTsh$ given below, "$self$" represents a GT-shadow $[m, f]$ (or a candidate for GT-shadow) with the target N. The class $GTsh$ has the following data attributes:

- $self.wm$ returns the first instance variable of $self$, i.e. the tuple (w, m), where the tuple w represents $f \in \mathsf{F}_2$;
- $self.w$ returns the tuple w, i.e. $self.wm[0]$;
- $self.tar$ returns the instance of the class $Equiv$ that represents $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$; for instance, the command $self.tar.PB4$ returns the tuple of 6 permutations that also represents N;
- $self.cc\_ch$ returns the value of the virtual cyclotomic character of $self$, i.e. the remainder of the division of $2m + 1$ by N_{ord};
- $self.g$ returns the image of $self.w$ (i.e. the image of f) in a permutation group isomorphic to $\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$.

The class $GTsh$ has the following methods:

- $self.is\_GTpr()$ returns True if the pair (m, f) satisfies hexagon relations (1.6), (1.7), pentagon relation (1.8), and $2m + 1$ represents a unit in the ring $\mathbb{Z}/N_{\mathrm{ord}}\mathbb{Z}$; otherwise, the command returns False;

- if $self$ represents a GT-shadow, then $self.src()$ returns the source of $[m, f]$ viewed as the morphism in the groupoid GTSh; the output is an instance of the class $Equiv$;
- $self.is\_GTsh()$ returns True if $self$ represents[7] a (practical) GT-shadow; otherwise, the command returns False;
- $self.is\_charm()$ returns True if $self$ represents a charming GT-shadow; otherwise, the command returns False;
- $self.settled()$ returns True if $self$ represents a settled GT-shadow, i.e. the source of the morphism $[m, f]$ in the groupoid GTSh coincides with its target; note that the command $self.settled()$ does not verify whether $[m, f]$ is charming or not, so it can be applied to morphisms of the groupoid GTSh and to morphisms of the sub-groupoid $\mathsf{GTSh}^{\heartsuit}$; if the GT-shadow $[m, f]$ is not settled then the command $self.settled()$ returns False;
- E be an instance of the class $Equiv$ that represents $\mathsf{N}^E \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and $\mathsf{N} \leq \mathsf{N}^E$; $self.proj(E)$ returns an instance of $GTsh$ that represents the image of $[m, f]$ in $\mathsf{GT}^{\heartsuit}(\mathsf{N}^E)$ with respect to the natural map $\mathsf{GT}^{\heartsuit}(\mathsf{N}) \to \mathsf{GT}^{\heartsuit}(\mathsf{N}^E)$ (see (1.13)); note that this method can also be applied to instances of $GTsh$ that represent elements of $\mathsf{GT}(\mathsf{N})$;
- let N^E be an element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ represented by an instance E of the class $Equiv$ and $\mathsf{N}^E \leq \mathsf{N}$; the command $self.survives(E)$ returns True if $self$ belongs to the image of the natural projection $\mathsf{GT}^{\heartsuit}(\mathsf{N}^E) \to \mathsf{GT}^{\heartsuit}(\mathsf{N})$ (see (1.13)); this command may work slowly if the order of the commutator subgroup $[\mathsf{F}_2/\mathsf{N}^E_{\mathsf{F}_2},\ \mathsf{F}_2/\mathsf{N}^E_{\mathsf{F}_2}]$ is greater than 10^5; for this method, it is important that $self$ represents a *charming* GT-shadow with the target N;
- let K be the source of the GT-shadow $[m, f]$ and $other$ be an instance of the class $GTsh$ that represents a GT-shadow $[m', f']$ with the target K; the command $self.compose(other)$ returns an instance of the class $GTsh$ that represents the composition $[m, f] \circ [m', f']$ (see eq. (2.55) in [5, Section 2.5]); the method tests whether the target of $other$ coincides with the source of $self$; if they do not coincide the command $self.compose(other)$ returns an error message;
- assuming that $[m, f]$ is a charming GT-shadow, the command $self.inv()$ returns an instance of $GTsh$ that represents the inverse of $[m, f]$ in the groupoid $\mathsf{GTSh}^{\heartsuit}$; note that the method finds the inverse by generating GT-shadows in $\mathsf{GT}^{\heartsuit}(\mathsf{K})$ where K is the source of $[m, f]$; this command may work slowly if the order of the commutator subgroup $[\mathsf{F}_2/\mathsf{K}_{\mathsf{F}_2},\ \mathsf{F}_2/\mathsf{K}_{\mathsf{F}_2}]$ is greater than 10^5;
- let c be a tuple (of two permutations) that represents a child's drawing $\mathcal{D}$; it is assumed that $\mathcal{D}$ is subordinate to N; the command $self.act(c)$ returns a tuple (of two permutations) that represents the child's drawing $\mathcal{D}^{[m,f]}$ (see [4, Theorem 3.1]).

4.3 The class *Dessin*. Its attributes and methods. Additional functions from *PaB.py*

Instances of the class *Dessin* represent child's drawings. The instance variable *pr* is a tuple of two permutations (c_1, c_2) in S_d such that the subgroup $\langle\, c_1, c_2 \,\rangle \leq S_d$ is transitive. For

[7] This part of the code is justified by [5, Proposition 2.10].

two instances D and DD of *Dessin* represented by pairs (c_1, c_2) and $(\tilde{c}_1, \tilde{c}_2)$, the command $D == DD$ returns True if and only if D and DD have the same degree d and there exists $h \in S_d$ such that $\tilde{c}_1 = hc_1h^{-1}$ and $\tilde{c}_2 = hc_2h^{-1}$.

The class *Dessin* has the following data attributes:

- $self.pr$ returns the instance variable, i.e. a tuple (c_1, c_2) of permutations that represents the child's drawing;
- $self.d$ returns the degree of the child's drawing represented by $self$;
- $self.full$ returns the permutation triple $(c_1, c_2, c_2^{-1}c_1^{-1})$;
- $self.passport$ returns the passport of the child's drawing represented by $self$; the output of $self.passport$ is a nested tuple of length 3; each entry of this tuple is a partition of d; for instance, $Dessin((permut(2,3), permut(0,1,2))).passport$ returns the tuple $((2,1,1),(3,),(4,))$.

The class *Dessin* has the following methods:

- $self.monG()$ returns the monodromy group $\langle\, c_1, c_2 \,\rangle$ of the child's drawing $self$;
- $self.is\_transitive()$ returns True if $self$ indeed represents a child's drawing, i.e. if the permutation group $\langle\, c_1, c_2 \,\rangle$ is transitive; otherwise, the method returns False;
- $self.genus()$ returns the genus of the covering of $\mathbb{CP}^1 \setminus \{0, 1, \infty\}$ corresponding to the child's drawing $self$;
- $self.is\_Galois()$ returns True if the covering map corresponding to $self$ is Galois; otherwise, the method returns False.

Here is the description of additional functions[8] from the Python file *PaB.py*:

- Let N be an isolated element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and L be a list of instances of the class $GTsh$ that represent all elements of $\mathsf{GT}^{\heartsuit}(\mathsf{N})$; the command $shadows2perm\_group(L, timed = None)$ returns the corresponding permutation group as a subgroup of S_d where d is the size of $\mathsf{GT}^{\heartsuit}(\mathsf{N})$; the output is an instance of the class *sympy.combinatorics.perm_groups.PermutationGroup*; note that d is also the order of this permutation group; if the second (optional) argument is True, then the command also prints how much time it took to form the permutation group;
- $shadows2perm\_group\_choice(\ , timed = None)$ is a version of $shadows2perm\_group(\ , timed = None)$ in which non-identity elements of $\mathsf{GT}^{\heartsuit}(\mathsf{N})$ are chosen randomly; here it is assumed that $L[0]$ represents the identity element of the group $\mathsf{GT}^{\heartsuit}(\mathsf{N})$; sometimes, this function works faster than $shadows2perm\_group(\ , timed = None)$;
- Let N be an isolated element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, L be a list of instances of the class $GTsh$ that represent all elements of $\mathsf{GT}^{\heartsuit}(\mathsf{N})$, and T be an element of L; the command $shadow2perm(T, L)$ returns the permutation in $S_{len(L)}$ corresponding to the left action of the corresponding GT-shadow on $\mathsf{GT}^{\heartsuit}(\mathsf{N})$; we assume that $L[0]$ represents the identity element of the group $\mathsf{GT}^{\heartsuit}(\mathsf{N})$; the output is an instance of the class *sympy.combinatorics.permutations.Permutation*;

[8] Most of these functions are for working with child's drawings.

- for a positive integer d, the command $dessin\_star(d)$ returns a tuple that represents the child's drawing with one black vertex of valency d and d univalent white vertices;
- for a positive integer d, the command $dessin\_path(d)$ returns a tuple that represents the child's drawing, whose graph is the path graph with d edges; if d is even, then the corresponding graph has 2 black univalent vertices;
- for a tuple tup of three partitions of an integer $d \geq 2$, $gener\_dessin\_pt(tup)$ is a generator of all child's drawing (in the tuple format) with the passport tup; note that there are triples of partitions that are not passports for any child's drawing;
- for a positive integer d, $gener\_dessin(d)$ is a generator of all child's drawing of degree d; the outputs are given in the tuple format; this generator has a limited practical value for $d > 7$;
- for a positive integer d, the command $rand\_dessin(d)$ returns a permutation pairs that represents a "relatively random" child's drawing of degree d;
- for a tuple c that represents a child's drawing $\mathcal{D}$ subordinate to $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and a list L of GT-shadows with the target N, the command $orbit(c, L)$ returns the corresponding orbit of $\mathcal{D}$; elements of L are instances of the class $GTsh$; the output of $orbit(\ ,\ )$ is a list whose entries are instances of the class $Dessin$;
- for an instance of the class $Equiv$ and a positive integer d, $srch\_dessin(E, d)$ is a generator of all (if any) child's drawings of degree d subordinate to the equivalence relation represented by E; the outputs of the generator are tuples of permutations;
- for a tuple x that represents a homomorphism $\psi : \mathrm{PB}_4 \to S_n$, the command $conjBySig1(x)$ (resp. $conjBySig2(x)$, $conjBySig3(x)$) returns a tuple that represents the homomorphism $\tilde{\psi} : \mathrm{PB}_4 \to S_n$ defined by the formula $\tilde{\psi}(w) := \psi(\sigma_1 w \sigma_1^{-1})$ (resp. by the formula $\tilde{\psi}(w) := \psi(\sigma_2 w \sigma_2^{-1})$, $\tilde{\psi}(w) := \psi(\sigma_3 w \sigma_3^{-1})$); of course, $\ker(\tilde{\psi}) = \sigma_1^{-1} \ker(\psi)\sigma_1$ (resp. $\ker(\tilde{\psi}) = \sigma_2^{-1} \ker(\psi)\sigma_2$, $\ker(\tilde{\psi}) = \sigma_3^{-1} \ker(\psi)\sigma_3$);
- for a tuple x that represents a finite index normal subgroup H in PB_4, the command $isNormB4(x)$ returns True if $\mathsf{H} \trianglelefteq \mathrm{B}_4$; otherwise, the command returns False;
- for a tuple c that represents a child's drawing of degree d, the command $dessin2PB4(c)$ returns a tuple that represents a group homomorphism $\varphi : \mathrm{PB}_4 \to S_d$ with the following property: the intersection of $\mathsf{F}_2 \leq \mathrm{PB}_3 \leq \mathrm{PB}_4$ with the kernel of φ is the kernel of the group homomorphism $\mathsf{F}_2 \to S_d$ corresponding to c;
- for a tuple tt that represents a normal subgroup (of finite index) K in PB_4, the command $B4\_inv(tt)$ returns a tuple that represents a normal core of K in B_4; thus, for every tuple c that represents a child's drawing, the tuple $B4\_inv(dessin2PB4(c))$ represents an element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ the dominates the child's drawing $[c]$;
- let xy be a tuple of two permutations and H be a subgroup of a permutation group G generated by elements of xy (tuple xy represents a group homomorphism $\mathsf{F}_2 \to G$); the command $subgrp2dessin(H, xy)$ returns a tuple that represents a child's drawing corresponding to the action of F_2 on the set G/H of left cosets of H in G; the index of H in G is the degree of this child's drawing.

For example, the commands $len(list(gener\_dessin(3)))$ and $len(list(gener\_dessin(4)))$ return 7 and 26, respectively. This means that there are exactly 7 child's drawings of degree 3 and exactly 26 child's drawings of degree 4.

Executing the lines:

```
for i in range(5):
    t = dessin2PB4(rand_dessin(8))
    print(isNormB4(t))
```

we see that a "typical" normal subgroup (of finite index) in PB_4 is not normal in B_4.

5 Playing with selected examples of elements of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, GT-shadows and their action on child's drawings

5.1 Looking for charming GT-shadows that are fake

Recall that an GT-shadow $[m, f] \in \mathsf{GT}^{\heartsuit}(\mathsf{N})$ is **fake** if it does not come from an element of $\widehat{\mathsf{GT}}$. Due to [5, Corollary 3.13], a GT-shadow $[m, f] \in \mathsf{GT}^{\heartsuit}(\mathsf{N})$ is fake if and only if there exists $\mathsf{K} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ such that

- $\mathsf{K} \leq \mathsf{N}$ and
- $[m, f]$ does not survive into K, i.e. $[m, f]$ does not belong to the image of the natural map in (1.13).

We will now describe functions from *PaB.py* that may be used to look for charming GT-shadows that are fake.

5.1.1 Looking for fake GT-shadows using function *charm_fake_search*

The function $charm\_fake\_search(\ ,\ ,\ )$ has 3 arguments: E, EE, num. E and EE are instances of the class *Equiv* that represent $\mathsf{N}, \mathsf{K} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and num is the total number of charming GT-shadows with the target N. We assume that $\mathsf{K} \leq \mathsf{N}$, i.e. the equivalence relation represented by EE is finer than the one represented by E. The command $charm\_fake\_search(E, EE, num)$ returns True if all charming GT-shadows with the target N survive into K; otherwise, the command returns False.

The following command gives us the list of pairs (i, j) such that $i < j$ and $\mathsf{N}^{(j)} \leq \mathsf{N}^{(i)}$:

```
pairs=[p for p in comb(range(35),2) if listE[p[1]].finer(listE[p[0]])]
```

Executing[9] the line:

```
{ charm_fake_search(listE[p[0]],listE[p[1]], len(GTcharm[p[0]])) for p in pairs }
```

we get $\{\mathsf{True}\}$. This implies that, for every pair $(i, j) \in \{0, \ldots, 34\}^2$ such that $i < j$ and $\mathsf{N}^{(j)} \leq \mathsf{N}^{(i)}$, the natural map

$$\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(j)}) \to \mathsf{GT}^{\heartsuit}(\mathsf{N}^{(i)})$$

[9] This command may take more than a minute.

is onto.

5.1.2 Looking for fake GT-shadows using function *charm_fake4isolated*

The function *charm_fake4isolated*(, ,) has 3 arguments: E, EE, num. E and EE are instances of the class *Equiv* that represent elements $\mathsf{N}, \mathsf{K} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and num is the total number of charming GT-shadows whose target is N. It is assumed that $\mathsf{K} \leq \mathsf{N}$ and N, K are isolated objects of $\mathsf{GTSh}^{\heartsuit}$ (i.e. the connected component $\mathsf{GTSh}^{\heartsuit}_{\mathrm{conn}}(\mathsf{N})$ (resp. $\mathsf{GTSh}^{\heartsuit}_{\mathrm{conn}}(\mathsf{K})$) of N (resp. of K) in the groupoid $\mathsf{GTSh}^{\heartsuit}$ has exactly one object).

The command *charm_fake4isolated*(E, EE, num) returns True if every GT-shadow in $\mathsf{GT}^{\heartsuit}(\mathsf{N})$ survives into K; otherwise the command returns False.

The function *charm_fake4isolated*(, ,) generates all elements of $\mathsf{GT}^{\heartsuit}(\mathsf{K})$ and produces their images in $\mathsf{GT}^{\heartsuit}(\mathsf{N})$. If, in the process of execution, the function gets $> num//2$ distinct charming GT-shadows with the target N, then the function returns True. Here, it is important that $\mathsf{GT}^{\heartsuit}(\mathsf{K})$ and $\mathsf{GT}^{\heartsuit}(\mathsf{N})$ are finite groups and the natural map $\mathsf{GT}^{\heartsuit}(\mathsf{K}) \to \mathsf{GT}^{\heartsuit}(\mathsf{N})$ is a group homomorphism. (See Remark 1.1.) Remark 1.1 implies that the elements of $\mathsf{GT}^{\heartsuit}(\mathsf{N})$ that survive into K form a subgroup of $\mathsf{GT}^{\heartsuit}(\mathsf{N})$. Since $\mathsf{GT}^{\heartsuit}(\mathsf{N})$ cannot have proper subgroups of order $> |\mathsf{GT}^{\heartsuit}(\mathsf{N})|//2$, the code for the function *charm_fake4isolated*(, ,) works correctly.

Let us consider the intersection $\mathsf{N} := \mathsf{N}^{(19)} \cap \mathsf{N}^{(23)}$. Both $\mathsf{N}^{(19)}$ and $\mathsf{N}^{(23)}$ are isolated. $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(19)})$ (resp. $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(23)})$) is a group of order 12 (resp. 42). The quotient $\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$ has order $192,036,096 = 2^8 \cdot 3^7 \cdot 7^3$ and the commutator subgroup $[\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}, \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}]$ has order $108,864 = 2^6 \cdot 3^5 \cdot 7$.

Using *charm_fake4isolated*(, ,) on iMac with the processor 3.4 GHz (Quad-Core Intel Core i5) it took over an hour to verify that every charming GT-shadow in $\mathsf{GT}(\mathsf{N}^{(19)})$ survives into N. On the other hand, on the same iMac, it took less than a minute to verify that every charming GT-shadow in $\mathsf{GT}(\mathsf{N}^{(23)})$ survives into N.

Let us consider element $\mathsf{N}^{(7)}$ from list (1.22) and element N^{dde6} stored (in the tuple format) in the file *E_dde6genus0*. Both $\mathsf{N}^{(7)}$ and N^{dde6} are isolated objects of the groupoid $\mathsf{GTSh}^{\heartsuit}$; $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(7)})$ is a group of order $12 = 2^2 \cdot 3$ and $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{dde6})$ is a group of order $32 = 2^5$. Since $\mathsf{N}^{(7)}$ and N^{dde6} are isolated, so is the element $\mathsf{K} := \mathsf{N}^{(7)} \cap \mathsf{N}^{dde6}$.

Executing the following[10] lines:

```
Edde6 = Equiv(load_now(E_dde6genus0))
EE = listE[7].cap(Edde6)
GT = [GTsh(wm,Edde6.PB4) for wm in gener_GT_charm(Edde6.PB4)]
charm_fake4isolated(listE[7],EE,len(GTcharm[7]))
charm_fake4isolated(Edde6,EE,len(GT))
```

we get True and True. This shows that the natural maps $\mathsf{GT}^{\heartsuit}(\mathsf{K}) \to \mathsf{GT}^{\heartsuit}(\mathsf{N}^{(7)})$ and $\mathsf{GT}^{\heartsuit}(\mathsf{K}) \to \mathsf{GT}^{\heartsuit}(\mathsf{N}^{dde6})$ are onto, i.e. every charming GT-shadows with the target $\mathsf{N}^{(7)}$ (resp. with the target N^{dde6}) survives into $\mathsf{K} := \mathsf{N}^{(7)} \cap \mathsf{N}^{dde6}$.

5.2 On various versions of the Furusho property

Various versions of the Furusho property are motivated by the statement which says roughly that, in the pro-unipotent setting, the pentagon relation implies the two hexagon relations.

[10] The last command may take more than 5 minutes.

For a precise formulation of this statement, we refer the reader to [2, Theorem 3.1] and [8, Theorem 1].

We say that an element $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ satisfies *the strong Furusho property* (see [5, Section 4.3]) if

Property 5.1 *For every $f\mathsf{N}_{\mathsf{F}_2} \in \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$ satisfying pentagon relation* (1.8) *modulo* N*, there exists $m \in \mathbb{Z}$ such that*

- *$2m+1$ represents a unit in $\mathbb{Z}/N_{\mathrm{ord}}\mathbb{Z}$ and*
- *the pair (m, f) satisfies hexagon relations* (1.6)*,* (1.7)*.*

Since every genuine GT-shadow can be represented by a pair (m, f) with $f \in [\mathsf{F}_2, \mathsf{F}_2]$, it makes sense to consider the weaker version of Property 5.1:

Property 5.2 *For every $f\mathsf{N}_{\mathsf{F}_2} \in [\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}, \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}]$ satisfying pentagon relation* (1.8) *modulo* N*, there exists $m \in \mathbb{Z}$ such that*

- *$2m+1$ represents a unit in $\mathbb{Z}/N_{\mathrm{ord}}\mathbb{Z}$ and*
- *the pair (m, f) satisfies hexagon relations* (1.6)*,* (1.7)*.*

Since there are examples of $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ that satisfy neither Property 5.1 nor Property 5.2, one could say that there is **no** version of Furusho's theorem for GT-shadows. Still, it is worth mentioning that some elements $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ satisfy Property 5.2 and some of these elements satisfy Property 5.1.

Here are selected functions for working with questions related to the Furusho property:

- for a tuple w that represents an element of F_2 and an instance E of the class *Equiv* that represents $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the command $onto(w, E)$ returns True if there exists m such that $(2m+1)+N_{\mathrm{ord}}\mathbb{Z} \in \big(\mathbb{Z}/N_{\mathrm{ord}}\mathbb{Z}\big)^{\times}$ and the cosets $x_{12}^{2m+1}\mathsf{N}_{\mathrm{PB}_3}$, $w^{-1}x_{23}^{2m+1}w\mathsf{N}_{\mathrm{PB}_3}$, $c^{2m+1}\mathsf{N}_{\mathrm{PB}_3}$ generate the group $\mathrm{PB}_3/\mathsf{N}_{\mathrm{PB}_3}$; otherwise, the command returns False;
- for an instance E of the class *Equiv* that represents $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and a tuple w that represents $f \in \mathsf{F}_2$ satisfying pentagon relation (1.8), the command $furusho(w, E)$ returns True if the set

  $$\big\{m \in \{0, 1, \ldots, N_{\mathrm{ord}}-1\} \;\big|\; \overline{2m+1} \in \big(\mathbb{Z}/N_{\mathrm{ord}}\mathbb{Z}\big)^{\times},\ (m, f) \text{ satisfies (1.6) and (1.7)}\big\}$$

  is non-empty; otherwise, the command returns False;
- for an instance E of the class *Equiv* that represents $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, $furusho\_test(E)$ is a generator of tuples of length 2: the first entry is a word w (of 0's and 1's) that satisfies pentagon relation (1.8) and the second entry is the output $furusho(w, E)$;
- for an instance E of the class *Equiv* that represents $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, $furusho\_test\_comm(E)$ is a generator of tuples of length 2: the first entry is a word w (of 0's and 1's) that represents an element of $[\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}, \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}]$ satisfying pentagon relation (1.8) and the second entry is the output $furusho(w, E)$;

- for an instance E of the class $Equiv$ that represents $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the command $furusho\_test1(E)$ returns False if there exists $f\mathsf{N}_{\mathsf{F}_2} \in \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$ that satisfies pentagon relation (1.8) and such that the set

$$\big\{m \in \{0,1,\dots,N_{\mathrm{ord}}-1\} \;\big|\; \overline{2m+1} \in \big(\mathbb{Z}/N_{\mathrm{ord}}\mathbb{Z}\big)^{\times},\ (m,f) \text{ satisfies (1.6) and (1.7)}\big\}$$

  is empty; otherwise, the command returns True;

- for an instance E of the class $Equiv$ that represents $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the command $furusho\_test\_comm1(E)$ returns False if there exists $f\mathsf{N}_{\mathsf{F}_2} \in [\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}, \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}]$ that satisfies pentagon relation (1.8) and such that the set

$$\big\{m \in \{0,1,\dots,N_{\mathrm{ord}}-1\} \;\big|\; \overline{2m+1} \in \big(\mathbb{Z}/N_{\mathrm{ord}}\mathbb{Z}\big)^{\times},\ (m,f) \text{ satisfies (1.6) and (1.7)}\big\}$$

  is empty; otherwise, the command returns True.

Using the function $furusho\_test1(\ )$, we showed that every N in the following list of 11 elements (out of 35 elements in (1.22))

$$\mathsf{N}^{(1)},\ \mathsf{N}^{(2)},\ \mathsf{N}^{(3)},\ \mathsf{N}^{(4)},\ \mathsf{N}^{(6)},\ \mathsf{N}^{(7)},\ \mathsf{N}^{(9)},\ \mathsf{N}^{(10)},\ \mathsf{N}^{(11)},\ \mathsf{N}^{(14)},\ \mathsf{N}^{(24)} \tag{5.1}$$

satisfy Property 5.1.

Similarly, using the function $furusho\_test\_comm1$ we showed that every N in the following list of 13 elements (out of 35 elements in (1.22))

$$\mathsf{N}^{(0)},\ \mathsf{N}^{(1)},\ \mathsf{N}^{(2)},\ \mathsf{N}^{(3)},\ \mathsf{N}^{(4)},\ \mathsf{N}^{(5)},\ \mathsf{N}^{(6)},\ \mathsf{N}^{(7)},\ \mathsf{N}^{(9)},\ \mathsf{N}^{(10)},\ \mathsf{N}^{(11)},\ \mathsf{N}^{(14)},\ \mathsf{N}^{(24)} \tag{5.2}$$

satisfy Property 5.2.

For example $\mathsf{N}^{(5)}$ does not belong to list (5.1). Executing the lines:

```
Test = list(furusho_test(listE[5]))
Test_ok = [p for p in Test if p[1]]
len(Test)
len(Test_ok)
```

we see that exactly 72 elements $f\mathsf{N}_{\mathsf{F}_2}^{(5)} \in \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}^{(5)}$ satisfy pentagon relation (1.8) (modulo $\mathsf{N}^{(5)}$) and, for exactly half of these elements, the set

$$\big\{m \in \{0,\dots,5\} \;\big|\; \overline{2m+1} \in \big(\mathbb{Z}/6\mathbb{Z}\big)^{\times},\ (m,f) \text{ satisfies (1.6) and (1.7) modulo } \mathsf{N}_{\mathrm{PB}_3}^{(5)}\big\}$$

is non-empty ($N_{\mathrm{ord}}^{(5)} = 6$).

Element $\mathsf{N}^{(8)}$ does not belong to list (5.2). Executing the lines:

```
Test = list(furusho_test_comm(listE[8]))
Test_ok = [p for p in Test if p[1]]
len(Test)
len(Test_ok)
```

we see that exactly 16 elements $f\mathsf{N}_{\mathsf{F}_2}^{(8)} \in [\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}^{(8)}, \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}^{(8)}]$ satisfy pentagon relation (1.8) (modulo $\mathsf{N}^{(8)}$) and, for exactly 10 of these 16 elements, the set

$$\big\{m \in \{0,\dots,4\} \;\big|\; \overline{2m+1} \in \big(\mathbb{Z}/5\mathbb{Z}\big)^{\times},\ (m,f) \text{ satisfies (1.6) and (1.7) modulo } \mathsf{N}_{\mathrm{PB}_3}^{(8)}\big\}$$

is non-empty ($N_{\mathrm{ord}}^{(8)} = 5$).

We should remark, even if $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ satisfies Property 5.1, it does **not** mean that every element in $(m, f\mathsf{N}_{\mathsf{F}_2}) \in \{0,1,\dots,N_{\mathrm{ord}}-1\} \times \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$ satisfying the pentagon relation and the hexagon relations is a GT-shadow. More precisely, for some of these pairs, the homomorphism $T_{m,f}^{\mathrm{PB}_3} : \mathrm{PB}_3 \to \mathrm{PB}_3/\mathsf{N}_{\mathrm{PB}_3}$ is **not onto**.

For instance, $\mathsf{N}^{(11)}$ satisfies Property 5.1. However, executing the lines:

```
Test = [ ]
for w in generWF2(listE[11].xy,None):
    if penta((w,0),listE[11].PB4):
        for m in m_units(listE[11].N0):
            if hexa1((w,m),listE[11].PB3) and hexa2((w,m),listE[11].PB3):
                Test.append((w,m))

Test_ok = [wm for wm in Test if GTsh(wm,listE[11].PB4).is_GTsh()]
len(Test)
len(Test_ok)
```

we see that the set[11]

$$\{(m, f\mathsf{N}_{\mathsf{F}_2}^{(11)}) \in \{0,1,2,3\} \times \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}^{(11)} \mid (m, f\mathsf{N}_{\mathsf{F}_2}^{(11)}) \text{ satisfies (1.6), (1.7), (1.8)}\}$$

has 24 elements and only 16 of these 24 elements represent GT-shadows.

Similarly, executing the lines:

```
Test = [ ]
for w in generWComm(listE[11].xy,None):
    if penta((w,0),listE[11].PB4):
        for m in m_units(listE[11].N0):
            if hexa1((w,m),listE[11].PB3) and hexa2((w,m),listE[11].PB3):
                Test.append((w,m))

Test_ok = [wm for wm in Test if GTsh(wm,listE[11].PB4).is_GTsh()]
len(Test)
len(Test_ok)
```

we see that the set

$$\{(m, f\mathsf{N}_{\mathsf{F}_2}^{(11)}) \in \{0,1,2,3\} \times [\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}^{(11)}, \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}^{(11)}] \mid (m, f\mathsf{N}_{\mathsf{F}_2}^{(11)}) \text{ satisfies (1.6), (1.7), (1.8)}\}$$

has 12 elements and only 8 of these 12 elements represent GT-shadows (in fact, these 8 elements represent charming GT-shadows).

[11] $N_{\mathrm{ord}}^{(11)} = 4$ and, for every $0 \le m \le 3$, $2m+1$ represents a unit in $\mathbb{Z}/4\mathbb{Z}$.

5.3 Playing with child's drawings and their $\mathsf{GTSh}^{\heartsuit}$-orbits

Several selected examples of child's drawings and their $\mathsf{GTSh}^{\heartsuit}$-orbits are described in [4, Section 5].

In this section, we present more examples of using the commands related to child's drawings and to the action of the groupoid $\mathsf{GTSh}^{\heartsuit}$ on child's drawings.

Since the groupoid GTSh (and the groupoid $\mathsf{GTSh}^{\heartsuit}$) acts trivially on Abelian child's drawings (see [4, Corollary 4.7]), we will focus only on non-Abelian examples.

Executing the lines:

```
LD3 = [c for c in gener_dessin(3) if c[0]*c[1]!=c[1]*c[0]]
LD4 = [c for c in gener_dessin(4) if c[0]*c[1]!=c[1]*c[0]]
LD5 = [c for c in gener_dessin(5) if c[0]*c[1]!=c[1]*c[0]]
```

we form a list $LD3$ (resp. $LD4$, $LD5$) of all non-Abelian child's drawings of degree 3 (resp. degree 4, degree 5). Executing the commands $len(LD3)$, $len(LD4)$ and $len(LD5)$, we see that there are exactly 3 (resp. 19, 91) non-Abelian child's drawings of degree 3 (resp. degree 4, 5).

Executing the lines:

```
LD3_E = [[i for i in range(35) if listE[i].subord(c)] for c in LD3]
LD4_E = [[i for i in range(35) if listE[i].subord(c)] for c in LD4]
LD5_E = [[i for i in range(35) if listE[i].subord(c)] for c in LD5]

for L in LD3_E:
    print(L)

LD4_ok = [LD4[i] for i in range(len(LD4)) if len(LD4_E[i])>0]
LD5_ok = [LD5[i] for i in range(len(LD5)) if len(LD5_E[i])>0]

for c in LD4_ok:
    display_perm(c[0]); display_perm(c[1])
    print(i for i in range(35) if listE[i].subord(c))
    print(' ')
    print(' ')

for c in LD5_ok:
    display_perm(c[0]); display_perm(c[1])
    print(i for i in range(35) if listE[i].subord(c))
```

we see that

- every non-Abelian child's drawing of degree 3 is subordinate to $\mathsf{N}^{(12)}$ and to $\mathsf{N}^{(19)}$ from the list in (1.22);
- $LD4\_ok$ is a list of 3 non-Abelian child's drawings of degree 4 and each child's drawing in $LD4\_ok$ is subordinate to element $\mathsf{N}^{(0)}$ or to element $\mathsf{N}^{(2)}$ from the list in (1.22);

all the remaining non-Abelian child's drawings of degree 4 are not subordinate to any element from (1.22);

- $LD5\_ok$ has length 1 and $LD5\_ok[0]$ represents a non-Abelian child's drawings of degree 5 that is subordinate to elements $\mathsf{N}^{(8)}, \mathsf{N}^{(13)}, \mathsf{N}^{(18)}, \mathsf{N}^{(20)}, \mathsf{N}^{(25)}$ from (1.22); all the remaining non-Abelian child's drawings of degree 5 are not subordinate to any element from (1.22).

Let us denote by $\mathcal{D}_{4,12}$, $\mathcal{D}_{4,8,1}$, $\mathcal{D}_{4,8,2}$ the child's drawings represented by elements of $LD4\_ok$. The child's drawings $\mathcal{D}_{4,12}$, $\mathcal{D}_{4,8,1}$ and $\mathcal{D}_{4,8,2}$ are also represented[12] by the following permutation pairs:

$$((1,2,3),\ (2,4,3)), \tag{5.3}$$

$$((1,2)(3,4),\ (2,3)), \tag{5.4}$$

$$((1,2),\ (1,3)(2,4)), \tag{5.5}$$

respectively. The monodromy group of $\mathcal{D}_{4,12}$ has order 12 (it is obviously the alternating group A_4) and the monodromy groups of $\mathcal{D}_{4,8,1}$ and $\mathcal{D}_{4,8,2}$ are of order 8. The results of the above commands show that $\mathcal{D}_{4,12}$ is subordinate to elements

$$\mathsf{N}^{(2)},\ \mathsf{N}^{(5)},\ \mathsf{N}^{(6)},\ \mathsf{N}^{(7)},\ \mathsf{N}^{(9)},\ \mathsf{N}^{(19)},\ \mathsf{N}^{(21)},\ \mathsf{N}^{(25)},$$

and $\mathcal{D}_{4,8,1}$, $\mathcal{D}_{4,8,2}$ are subordinate to elements $\mathsf{N}^{(0)}, \mathsf{N}^{(1)}, \mathsf{N}^{(10)}, \mathsf{N}^{(11)}, \mathsf{N}^{(24)}$ from the list in (1.22). In particular, the element $\mathsf{N}^{(0)} \cap \mathsf{N}^{(2)}$ dominates all 3 child's drawings $\mathcal{D}_{4,12}$, $\mathcal{D}_{4,8,1}$ and $\mathcal{D}_{4,8,2}$.

Let us denote by $\mathcal{D}_{5,2}$ the child's drawing represented by $LD5\_ok[0]$. This child's drawing has genus 2 and it is also represented by the permutation pair:

$$((1,2,3,4,5),\ (1,2,4,5,3)). \tag{5.6}$$

As we mentioned above, $\mathcal{D}_{5,2}$ is subordinate to elements $\mathsf{N}^{(8)}, \mathsf{N}^{(13)}, \mathsf{N}^{(18)}, \mathsf{N}^{(20)}, \mathsf{N}^{(25)}$ from the list in (1.22).

The child's drawing $\mathcal{D}_{5,2}$ can be represented a Belyi pair defined over $\mathbb{Q}$ and one such Belyi pair can be found in `https://beta.lmfdb.org/Belyi/5T4/5/5/5/a/`. Indeed, by manually copying the permutation pair from `https://beta.lmfdb.org/Belyi/5T4/5/5/5/a/` and executing the line:

Dessin((permut(0,1,2,3,4), permut(0,2,3,1,4))) == Dessin(LD5_ok[0])

we see that $\mathcal{D}_{5,2}$ and the Belyi pair from `https://beta.lmfdb.org/Belyi/5T4/5/5/5/a/` represent the same child's drawing.

Executing the lines:

EE = listE[0].cap(listE[2])
GT0_2 = [GTsh(wm,EE.PB4) for wm in gener_GT_charm(EE.PB4)]

we get an instance EE of $Equiv$ that represents the element $\mathsf{N}^{(0)} \cap \mathsf{N}^{(2)}$ and the list $GT0\_2$ of all charming GT-shadows with the target $\mathsf{N}^{(0)} \cap \mathsf{N}^{(2)}$.

[12] Since the output of the command $list(G.elements)$ for a permutation group G depends on the computer session, you may get different representatives for child's drawings $\mathcal{D}_{4,12}$, $\mathcal{D}_{4,8,1}$ and $\mathcal{D}_{4,8,2}$.

Executing the commands $\{len(orbit(c, GTcharm[12]))\ \ for\ \ c\ \ in\ \ LD3\}$, $\{len(orbit(c, GT0\_2))\ \ for\ \ c\ \ in\ \ LD4\_ok\}$ and $len(orbit(LD5\_ok[0], GTcharm[8]))$, we see that

- $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(12)})$ acts trivially on all non-Abelian child's drawing of degree 3;
- $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(0)} \cap \mathsf{N}^{(2)})$ acts trivially on $\mathcal{D}_{4,12}$, $\mathcal{D}_{4,8,1}$ and $\mathcal{D}_{4,8,2}$;
- $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(8)})$ acts trivially on $\mathcal{D}_{5,2}$.

It is known (see, for example, [10]) that $G_{\mathbb{Q}}$ acts trivially on all child's drawings of degree ≤ 4. However, there are child's drawings of degree 5 on which $G_{\mathbb{Q}}$ acts non-trivially. Several examples of such child's drawings can be found in [10] (as well as in many other articles and surveys on this topics). Using this package, one can show that the child's drawing $\mathcal{D}_{5,0}$ stored in file $dde5genus0$ is subordinate to an element N^{dde5} stored (in the tuple format) in the file $E\_dde5genus0$ and the orbit

$$\mathsf{GT}^{\heartsuit}(\mathsf{N}^{dde5})(\mathcal{D}_{5,0}) = G_{\mathbb{Q}}(\mathcal{D}_{5,0})$$

has size 2. More details about $\mathcal{D}_{5,0}$ can be found in [4, Section 5].

Recall that the Philadelphia subgroup $\mathsf{N}^{(19)}$ is an isolated object of the groupoid $\mathsf{GTSh}^{\heartsuit}$. In particular, $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(19)})$ is a group. Executing the lines:

```
LD6_19 = [ ]
for c in srch_dessin(listE[19], 6):
    if c[0]*c[1] != c[1]*c[0]:
        LD6_19.append(c)

len(LD6_19)
{len(orbit(c, GTcharm[19])) for c in LD6_19}
```

we see that there are exactly 28 non-Abelian child's drawings of degree 6 subordinate to $\mathsf{N}^{(19)}$ and the group $\mathsf{GT}(\mathsf{N}^{(19)})$ acts trivially on all these child's drawings.

Executing the command $c6 = load\_now('dde6genus0')$, we pull yet another example $\mathcal{D}_{6,0}$ of a child's drawing of degree 6. $\mathcal{D}_{6,0}$ is represented by the permutation triple

$$\big((1,4,5,2)(3,6),\ (1,6,3,2)(4,5),\ (1,3)(2,4)\big) \tag{5.7}$$

and by the Belyi pair from `https://beta.lmfdb.org/Belyi/6T10/4.2/4.2/2.2.1.1/a/`

Executing[13] the lines:

```
tt = B4_inv(dessin2PB4(c6))
EE = Equiv(tt)
GTdde6=[GTsh(wm, EE.PB4) for wm in gener_GT_charm(EE.PB4)]
```

we get an instance EE of $Equiv$ that represents an element N^{dde6} that dominates $\mathcal{D}_{6,0}$. Executing the command $orbit(c6, GTdde6)$, we see that the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{dde6})(\mathcal{D}_{6,0})$ has size two.

[13] The third line may take more than a minute.

Copying the permutation pair that represents the Galois conjugate of $\mathcal{D}_{6,0}$ from `https://beta.lmfdb.org/Belyi/6T10/4.2/4.2/2.2.1.1/a/` manually and executing the line:

Dessin((permut(0,3,4,1)(2,5), permut(0,1,4,5)(2,3))) == orbit(c6, GTdde6)[1]

we see that the $G_{\mathbb{Q}}$-orbit of $\mathcal{D}_{6,0}$ coincides with $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{dde6})(\mathcal{D}_{6,0})$.

5.3.1 Producing child's drawings subordinate to N using proper subgroups of $\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$

Another possible way of producing child's drawings subordinate to a given element $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ is based on using subgroups of $\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$. Indeed, we have the standard bijection between subgroups $H \leq \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$ of index d and subgroups $\tilde{H} \leq \mathsf{F}_2$ of index d such that $\mathsf{N}_{\mathsf{F}_2} \leq \tilde{H}$. For every such $H \leq \mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$, the left action of F_2 on the set $\mathsf{F}_2/\tilde{H}$ of left cosets of $\tilde{H}$ gives us a child's drawing subordinate to N. This idea is implemented in the function *subgrp2dessin(,)* described in Section 4.3.

Here is an example of looking for child's drawings subordinate to $\mathsf{N}^{(21)}$ from the list in (1.22). Executing the lines:

```
G = PG(listE[21].xy); LG = list(G.elements); gen = [ ]
for i in range(10):
    a=choice(LG); b=choice(LG)
    d=PG(a,b).order()
    if d < G.order( ):
        gen.append((a,b))
        print('I found a proper subgroup of index ', G.order( )//d)
```

we found[14] 6 pairs (a, b) of elements of G for which the subgroups $\langle\, a, b \,\rangle$ have indicies:

$$54,\ 36,\ 54,\ 12,\ 36,\ 27.$$

Since 12 is the smallest index, we produced a child's drawing $\mathcal{D}_{12,3}$ of degree 12 subordinate to $\mathsf{N}^{(21)}$ by executing the command $c12 = subgrp2dessin(PG(gen[3]), listE[21].xy)$. (It is also a good idea to execute the command $listE[21].subord(c12)$.)

By executing the command $len(orbit(c12, GTcharm[21]))$, we verified that $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(21)})$ acts trivially on $\mathcal{D}_{12,3}$. Hence $G_{\mathbb{Q}}$ also acts trivially on $\mathcal{D}_{12,3}$. (The child's drawing $\mathcal{D}_{12,3}$ is stored (in the tuple format) in file $dde12E21$.)

Using the function *subgrp2dessin(,)* in the similar way, we produced the child's drawing $\mathcal{D}_{36,10}$ subordinate to $\mathsf{N}^{(21)}$ and stored in file $dde36E21$. Executing[15] the lines:

```
c36 = load_now('dde36E21'); T = GTcharm[21][1]; T.wm
Dessin(T.act(c36)) == Dessin(c36)
```

we see that (the GT-shadow coming from) the complex conjugation acts non-trivially on $\mathcal{D}_{36,10}$. In particular, the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(21)})(\mathcal{D}_{36,10})$ has size ≥ 2. (If you are patient and/or you have a relatively fast computer, you may also try to execute the command

[14] The specific results depend heavily on the computer session.
[15] The second line may take more than a minute.

$len(orbit(c36, GTcharm[21]))$. The author guesses that the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(21)})(\mathcal{D}_{36,10})$ has size 2.)

6 Descriptions of storage files

- *subGrPB4_org35* contains the list of 35 selected elements (1.22) of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$; each element is stored as a tuple that represents a group homomorphism from PB_4 to a permutation group;
- *wm_list_charm35* contains the nested list of length 35; its i-th entry is the list of all elements in $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(i)})$ (in the tuple format), where $\mathsf{N}^{(i)}$ is the i-th entry of the list stored in *subGrPB4_org35*;
- *wm_list_all31* contains the nested list of length 31; its i-th entry (for $0 \le i \le 30$) is the list of all GT-shadows (in the tuple format) in $\mathsf{GT}(\mathsf{N}^{(i)})$, where $\mathsf{N}^{(i)}$ is the i-th entry of the list stored in *subGrPB4_org35*;
- *wm_list31_all* contains the list of all elements in $\mathsf{GT}(\mathsf{N}^{(31)})$ (in the tuple format); this list has 588 elements; for the iMac with the processor 3.4 GHz, Intel Core i5, it took approximately 9.5 full days to complete this task;
- *wm_list32_all* contains the list of all elements in $\mathsf{GT}(\mathsf{N}^{(32)})$ (in the tuple format); this list has 800 elements; for the iMac with the processor 3.4 GHz, Intel Core i5, it took almost 10 full days to complete this task;
- *Mighty_Dandy_wm_list* contains the list of the found 4374 (practical) GT-shadows for the Mighty Dandy $\mathsf{N}^{(34)}$; each GT-shadow is given in the tuple format; note that $\mathsf{GT}(\mathsf{N}^{(34)})$ may contain more (practical) GT-shadows;
- *G_Mighty_Dandy* contains an instance of the class *sympy.combinatorics.perm_groups.PermutationGroup* that represents a subgroup of S_{486} isomorphic to the group $\mathsf{GTSh}^{\heartsuit}(\mathsf{N}^{(34)}, \mathsf{N}^{(34)}) = \mathsf{GT}^{\heartsuit}(\mathsf{N}^{(34)})$;
- *Leila_PB4* contains Leila's subgroup $\mathsf{N}^{\mathcal{L}} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and it is given in the tuple format; here is the basic information about $\mathsf{N}^{\mathcal{L}}$:
  - $\mathsf{N}^{\mathcal{L}}$ is the kernel of a group homomorphism $\mathrm{PB}_4 \to S_{130}$, $\mathsf{N}^{\mathcal{L}}_{\mathrm{PB}_3}$ is the kernel of a homomorphism $\mathrm{PB}_3 \to S_{130}$ and $N^{\mathcal{L}}_{\mathrm{ord}} = 12$,
  - $|\mathrm{PB}_4 : \mathsf{N}^{\mathcal{L}}| = 285,315,214,344,192 = 2^{29} \cdot 3^{12} \approx 3 \cdot 10^{14}$,
  - $|\mathrm{PB}_3 : \mathsf{N}^{\mathcal{L}}_{\mathrm{PB}_3}| = 2,985,984 = 2^{12} \cdot 3^6 \approx 3 \cdot 10^6$,
  - $|\mathsf{F}_2 : \mathsf{N}^{\mathcal{L}}_{\mathsf{F}_2}| = 248,832 = 2^{10} \cdot 3^5$, the order of the commutator subgroup of $\mathsf{F}_2/\mathsf{N}^{\mathcal{L}}_{\mathsf{F}_2}$ is $1728 = 2^6 \cdot 3^3$,
  - $\mathsf{N}^{\mathcal{L}}$ is an isolated element of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ and $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{\mathcal{L}})$ is a non-Abelain group of order $48 = 2^4 \cdot 3$.
- *wm_list_Leila* contains the list of all elements of $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{\mathcal{L}})$ in the tuple format; as we mentioned above, $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{\mathcal{L}})$ is a non-Abelian group of order $48 = 2^4 \cdot 3$; its 3-Sylow subgroup is normal but its 2-Sylow subgroup is not normal; a 2-Sylow subgroup of $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{\mathcal{L}})$ is non-Abelian;

- *dde*4*Many* contains the child's drawing $\mathcal{D}_{4,0}$ (in the tuple format) of degree 4 represented by the first bipartite ribbon graph in figure 6.1 on page 39; $\mathcal{D}_{4,0}$ is also represented by the permutation triple:
$$\big((1,2,3),\ (2,4,3),\ (1,4,2)\big),$$
and the Belyi pair from `https://beta.lmfdb.org/Belyi/4T4/3.1/3.1/3.1/a/`; its passport is $((3,1),(3,1),(3,1))$ and its genus is 0; $\mathcal{D}_{4,0}$ is subordinate to the elements $\mathsf{N}^{(2)}$, $\mathsf{N}^{(5)}$, $\mathsf{N}^{(6)}$, $\mathsf{N}^{(7)}$, $\mathsf{N}^{(9)}$, $\mathsf{N}^{(19)}$, $\mathsf{N}^{(21)}$, $\mathsf{N}^{(25)}$ from the list in (1.22); the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(2)})(\mathcal{D}_{4,0})$ (as well as the orbit $G_{\mathbb{Q}}(\mathcal{D}_{4,0})$) is a singleton;

- *dde*5*genus*0 contains the child's drawing $\mathcal{D}_{5,0}$ (in the tuple format) of degree 5; $\mathcal{D}_{5,0}$ is represented by the permutation triple
$$\big((1,4,5,2),\ (2,3,5,4),\ (1,4)(2,3)\big)$$
and by the Belyi pair from `https://beta.lmfdb.org/Belyi/5T3/4.1/4.1/2.2.1/a`; the passport of $\mathcal{D}_{5,0}$ is
$$\big((4,1),\ (4,1),\ (2,2,1)\big)$$
and its genus is 0; $\mathcal{D}_{5,0}$ is subordinate to the element N^{dde5} stored in file *E_dde*5*genus*0 and the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{dde5})(\mathcal{D}_{5,0})$ (as well as the orbit $G_{\mathbb{Q}}(\mathcal{D}_{5,0})$) has size 2;

- *dde*6*genus*0 contains the child's drawing $\mathcal{D}_{6,0}$ (in the tuple format) of degree 6; $\mathcal{D}_{6,0}$ is represented by the permutation triple
$$\big((1,4,5,2)(3,6),\ (1,6,3,2)(4,5),\ (1,3)(2,4)\big)$$
and by the Belyi pair from `https://beta.lmfdb.org/Belyi/6T10/4.2/4.2/2.2.1.1/a`; the passport of $\mathcal{D}_{6,0}$ is
$$\big((4,2),\ (4,2),\ (2,2,1,1)\big)$$
and its genus is 0; $\mathcal{D}_{6,0}$ is subordinate to the element N^{dde6} stored in file *E_dde*6*genus*0 and the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{dde6})(\mathcal{D}_{6,0})$ (as well as the orbit $G_{\mathbb{Q}}(\mathcal{D}_{6,0})$) has size 2;

- *dde*7*E*28 contains the child's drawing $\mathcal{D}_{7,0,28}$ (in the tuple format); $\mathcal{D}_{7,0,28}$ is represented by the permutation triple:
$$\big((1,2,3,4,5),\ (3,6,7,5,4),\ (1,7,6,3,2)\big)$$
and it is subordinate to $\mathsf{N}^{(28)}$; its passport is $((5,1,1),(5,1,1),(5,1,1))$ and its genus is zero; the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(28)})(\mathcal{D}_{7,0,28})$ is a singleton (hence the $G_{\mathbb{Q}}$-orbit of $\mathcal{D}_{7,0,28}$ is also a singleton); a Belyi pair that represents $\mathcal{D}_{7,0,28}$ can be found at `https://beta.lmfdb.org/Belyi/7T6/5.1.1/5.1.1/5.1.1/a/`;

- *dde*7*E*29 contains the child's drawing $\mathcal{D}_{7,0,29}$ (in the tuple format); $\mathcal{D}_{7,0,29}$ is represented by the permutation triple:
$$\big((1,2,3)(4,5)(6,7),\ (1,5,6)(2,7)(3,4),\ (1,4)(2,6)(3,7,5)\big)$$
and it is subordinate to $\mathsf{N}^{(29)}$; its passport is $((3,2,2),(3,2,2),(3,2,2))$ and its genus is zero; the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(29)})(\mathcal{D}_{7,0,29})$ is a singleton (hence the $G_{\mathbb{Q}}$-orbit of $\mathcal{D}_{7,0,29}$ is also a singleton); a Belyi pair that represents $\mathcal{D}_{7,0,29}$ can be found at `https://beta.lmfdb.org/Belyi/7T6/3.2.2/3.2.2/3.2.2/a`;

- *dde*8*E*5*E*19 contains the child's drawing $\mathcal{D}_{8,0}$ (in the tuple format) of degree 8 that is subordinate to $\mathsf{N}^{(5)}$ and $\mathsf{N}^{(19)}$; $\mathcal{D}_{8,0}$ is represented by the permutation triple

$$\big((1,2,3)(4,5,6),\ (1,8,5)(2,4,7),\ (1,3,7,4,6,8)(2,5)\big),$$

its passport is $\big((3,3,1,1),(3,3,1,1),(6,2)\big)$ and its genus is zero; altogether, there are 5 child's drawings with this passport; the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(5)})(\mathcal{D}_{8,0})$ is a singleton (hence the $G_{\mathbb{Q}}$-orbit of $\mathcal{D}_{8,0}$ is also a singleton); the author could not find $\mathcal{D}_{8,0}$ in database [10]; however, the author guesses that a Belyi pair that represents $\mathcal{D}_{8,0}$ may be obtained from the one at `https://beta.lmfdb.org/Belyi/8T12/6.2/3.3.1.1/3.3.1.1/a/` by applying a Moebius transformation; $\mathcal{D}_{8,0}$ is also represented by the second bipartite ribbon graph shown in figure 6.1 on page 39;

- *dde*12*E*21 contains the child's drawing $\mathcal{D}_{12,3}$ (in the tuple format) of degree 12 that is subordinate to $\mathsf{N}^{(21)}$; $\mathcal{D}_{12,3}$ is represented by the permutation triple

$$\big((1,11,6,3,9,2,5,7,4),\ (1,6,12,3,2,10,5,4,8)(7,11,9),$$

$$(1,5,3)(2,11,8,4,9,12,6,7,10)\big),$$

its passport is $((9,1,1,1),(9,3),(9,3))$ and its genus is 3; the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(21)})(\mathcal{D}_{12,3})$ is a singleton (hence the $G_{\mathbb{Q}}$-orbit of $\mathcal{D}_{12,3}$ is also a singleton);

- *dde*12*wheel* contains the child's drawing $\mathcal{D}_{12,0}$ (in the tuple format) of degree 12 represented by the third bipartite ribbon graph shown in figure 6.1 on page 39; unfortunately, $\mathcal{D}_{12,0}$ is not subordinate to any element from the list in (1.22); executing the commands $c12 = load\_now('dde12wheel')$, $E = Equiv(B4\_inv(dessin2PB4(c12)))$, $E.commF2(\ ).order(\ )$, we produce an instance of *Equiv* that represents an element N that dominates $\mathcal{D}_{12,0}$; we also see that the commutator subgroup of $\mathsf{F}_2/\mathsf{N}_{\mathsf{F}_2}$ has order $78732 = 2^2 \cdot 3^9$; if the reader has enough patience or a stronger computer, he/she may try to find all charming GT-shadows with the target N;

- *dde*14*E*15 contains the child's drawing $\mathcal{D}_{14,5}$ (in the tuple format) of degree 14 that is subordinate to $\mathsf{N}^{(15)}$; $\mathcal{D}_{14,5}$ is represented by the permutation triple

$$\big((1,2,3,4,5,6,7)(8,9,10,11,12,13,14),\ (1,2,5,9,3,8,13)(4,10,11,14,7,12,6),$$

$$(1,14,8,11,4,9,3)(2,13,7,12,10,5,6)\big),$$

its passport is $((7,7),(7,7),(7,7))$ and its genus is 5; the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(15)})(\mathcal{D}_{14,5})$ has size 2;

- *dde*15*E*29 contains the child's drawing $\mathcal{D}_{15,4}$ (in the tuple format) of degree 15 that is subordinate to $\mathsf{N}^{(29)}$; $\mathcal{D}_{15,4}$ is represented by the permutation triple

$$\big((1,2,3,4,5,6)(7,8,9,10,11,12)(13,14,15),\ (1,2,6,12,9,15)(3,7,13)(4,11,14,5,8,10),$$

$$(1,2,15,11,8,3)(4,13,9,5,10,12)(6,14,7)\big),$$

its passport is $((6,6,3),(6,6,3),(6,6,3))$ and its genus is 4; the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(29)})(\mathcal{D}_{15,4})$ has size 2;

- *dde*18*E*29 contains a tuple that represents the child's drawing $\mathcal{D}_{18,4}$; this child's drawing is represented by the permutation triple:

$$\big((1,10,17,2,9,18)(3,12,13,4,11,14)(5,8,15,6,7,16),$$
$$(1,16,11,2,15,12)(3,18,7,4,17,8)(5,14,9,6,13,10),$$
$$(1,3,5)(2,4,6)(7,9,11)(8,10,12)(13,15,17)(14,16,18)\big);$$

  $\mathcal{D}_{18,4}$ has genus 4 and its passport is $((6,6,6),(6,6,6),(3,3,3,3,3,3))$; $\mathcal{D}_{18,4}$ is subordinate to $\mathsf{N}^{(29)}$ and the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(29)})(\mathcal{D}_{18,4})$ is a singleton; the covering corresponding to $\mathcal{D}_{18,4}$ is Galois;

- *dde*20*E*8*_many* contains the child's drawing $\mathcal{D}_{20,5}$ (in the tuple format) of degree 20 that is subordinate to $\mathsf{N}^{(8)}, \mathsf{N}^{(13)}, \mathsf{N}^{(18)}, \mathsf{N}^{(20)}, \mathsf{N}^{(25)}$; its passport is

$$((5,5,5,5),\ (5,5,5,5),\ (5,5,5,5))$$

  and its genus is 5; the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(8)})(\mathcal{D}_{20,5})$ is a singleton; although $\mathcal{D}_{20,5}$ has a uniform passport, $\mathcal{D}_{20,5}$ is not Galois;

- *dde*21*E*15 contains the child's drawing $\mathcal{D}_{21,7}$ (in the tuple format) of degree 21 that is subordinate to $\mathsf{N}^{(15)}$; its passport is $((7,7,7),(7,7,7),(7,7,7))$ and its genus is 7; the orbit $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(15)})(\mathcal{D}_{21,7})$ is a singleton; although $\mathcal{D}_{21,7}$ has a uniform passport, $\mathcal{D}_{21,7}$ is not Galois;

- *dde*36*E*21 contains the child's drawing $\mathcal{D}_{36,10}$ (in the tuple format) of degree 36 and genus 10 that is subordinate to $\mathsf{N}^{(21)}$; its passport is

$$\big((9,9,9,3,3,3),\ (9,9,9,3,3,3),\ (9,9,9,3,3,3)\big)$$

  and the orbit $\mathsf{GT}(\mathsf{N}^{(21)})(\mathcal{D}_{36,10})$ has size ≥ 2;

- *E_dde*5*genus*0 contains an element $\mathsf{N}^{dde5} \in \mathsf{NFl}_{\mathrm{PB}_4}(\mathrm{B}_4)$ (in the tuple format) that dominates the child's drawing stored in *dde*5*genus*0;

  - N^{dde5} is the kernel of a group homomorphism $\mathrm{PB}_4 \to S_{160}$, $\mathsf{N}^{dde5}_{\mathrm{PB}_3}$ is the kernel of a homomorphism $\mathrm{PB}_3 \to S_{160}$ and $N^{dde5}_{\mathrm{ord}} = 4$,
  - $|\mathrm{PB}_4 : \mathsf{N}^{dde5}| = 25 \cdot 10^{10} = 2^{10} \cdot 5^{12}$,
  - $|\mathrm{PB}_3 : \mathsf{N}^{dde5}_{\mathrm{PB}_3}| = 8,000 = 2^6 \cdot 5^3$,
  - $|\mathsf{F}_2 : \mathsf{N}^{dde5}_{\mathsf{F}_2}| = 2,000 = 2^4 \cdot 5^3$, the order of the commutator subgroup of $\mathsf{F}_2/\mathsf{N}^{dde5}_{\mathsf{F}_2}$ is $125 = 5^3$,
  - N^{dde5} is not an isolated object of the groupoid $\mathsf{GTSh}^{\heartsuit}$; the connected component $\mathsf{GTSh}^{\heartsuit}_{\mathrm{conn}}(\mathsf{N}^{dde5})$ has two objects;

- *E_dde*6*genus*0 contains an element $\mathsf{N}^{dde6} \in \mathsf{NFl}_{\mathrm{PB}_4}(\mathrm{B}_4)$ (in the tuple format) that dominates the child's drawing stored in *dde*6*genus*0;

  - N^{dde6} is the kernel of a group homomorphism $\mathrm{PB}_4 \to S_{192}$, $\mathsf{N}^{dde6}_{\mathrm{PB}_3}$ is the kernel of a homomorphism $\mathrm{PB}_3 \to S_{192}$ and $N^{dde6}_{\mathrm{ord}} = 4$,

– $|\mathrm{PB}_4 : \mathsf{N}^{dde6}| = 2^{10} \cdot 3^{24}$,
– $|\mathrm{PB}_3 : \mathsf{N}^{dde6}_{\mathrm{PB}_3}| = 46,656 = 2^6 \cdot 3^6$,
– $|\mathsf{F}_2 : \mathsf{N}^{dde6}_{\mathsf{F}_2}| = 11,664 = 2^4 \cdot 3^6$, the order of the commutator subgroup of $\mathsf{F}_2/\mathsf{N}^{dde6}_{\mathsf{F}_2}$ is $729 = 3^6$,
– the element N^{dde6} is isolated and $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{dde6})$ is a non-Abelian group of order $32 = 2^5$.

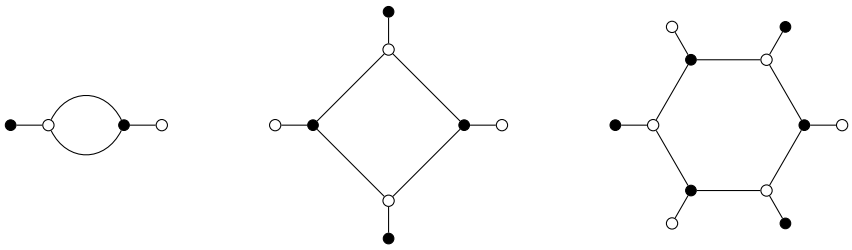

Fig. 6.1: The child's drawings $\mathcal{D}_{4,0}$, $\mathcal{D}_{8,0}$ and $\mathcal{D}_{12,0}$ stored in files *dde*4*Many*, *dde*8*E*5*E*19 and *dde*12*wheel*, respectively

7 Testing

Many functions, methods and outputs were tested directly. For example, the command

$$\{isNormB4(E.PB4) \;\; for \;\; E \;\; in \;\; listE\}$$

returns $\{\mathsf{True}\}$. This confirms that, for every equivalence relation E in $listE$ of 35 elements, the corresponding subgroup $\mathsf{N}_E \leq \mathrm{PB}_4$ is normal in B_4.

In the rest of this section, we outline indirect ways of testing various functions, methods and outputs.

Testing the lists of charming GT-shadows using the cyclotomic character

It is well known that the cyclotomic character $\chi : G_{\mathbb{Q}} \to \widehat{\mathbb{Z}}^{\times}$ is an onto group homomorphism. Hence, for every $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the virtual cyclotomic character

$$\mathsf{Ch}_{cyclot} : \mathsf{GT}^{\heartsuit}(\mathsf{N}) \to \left(\mathbb{Z}/N_{\mathrm{ord}}\mathbb{Z}\right)^{\times} \tag{7.1}$$

is onto.

For a tuple t representing $\mathsf{N} \in \mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$, the command $test\_cyclotomic(t)$ prints the image of the virtual cyclotomic character and returns True if the map (7.1) is onto. Otherwise, the command prints the alarming statement "Something is wrong with the values of the virtual cyclotomic character." and returns False.

To test the package, we applied the function *test_cyclotomic* to all elements of $\mathsf{NFI}_{\mathrm{PB}_4}(\mathrm{B}_4)$ represented by the entries of $listE$.

Indirect testing of $relPB4(\ )$

The function $relPB4$ was tested (indirectly) using the explicit formula for the standard generator c_4 of the center $\mathcal{Z}(\mathrm{PB}_4)$ of PB_4. (See (A.10) in Appendix A.1). For a tuple of t

representing a group homomorphism $\mathrm{PB}_4 \to S_d$, the command $cenPB4(t)$ returns the image of c_4 in S_d. For a tuple of 6 permutations in S_d, the command $test\_relPB4(t)$ returns True if the permutation $cenPB4(t)$ commutes with each entry of the tuple t; otherwise, it returns False.

Executing the command $\{test\_relPB4(E.PB4)\ for\ E\ in\ listE\}$, we get {True}.

Indirect testing of N_PB3 and *cap*

Let t be a tuple representing a homomorphism from PB_4 to S_d and $\mathsf{N} \trianglelefteq \mathrm{PB}_4$ be the kernel of this homomorphism. Both commands $N\_PB3(t)$ and $N\_PB3\_1(t)$ return a homomorphism φ from PB_3 to a symmetric group whose kernel is

$$\mathsf{N}_{\mathrm{PB}_3} := \varphi_{123}^{-1}(\mathsf{N}) \cap \varphi_{12,3,4}^{-1}(\mathsf{N}) \cap \varphi_{1,23,4}^{-1}(\mathsf{N}) \cap \varphi_{1,2,34}^{-1}(\mathsf{N}) \cap \varphi_{234}^{-1}(\mathsf{N}).$$

Unlike N_PB3, the function N_PB3_1 does not use *cap*. This is why, the degree of the permutation group $\varphi(\mathrm{PB}_3)$ for N_PB3_1 may be bigger than the degree of the corresponding permutation group for N_PB3. Of course, the kernel of the homomorphism $\mathrm{PB}_3 \to S_d$ corresponding to $N\_PB3(t)$ coincides with the kernel of the homomorphism $\mathrm{PB}_3 \to S_{d_1}$ corresponding to $N\_PB3\_1(t)$. This observation was used for testing N_PB3 and *cap* indirectly.

Indirect testing of $sameNsubgrp(\ ,\ )$

For tuples x and y representing homomorphisms φ_x and φ_y from a free group on $len(x)$ generators to symmetric groups, $sameNsubgrp(x, y)$ returns True if $\ker(\varphi_x) = \ker(\varphi_y)$; otherwise it returns False. This function is often applied to tuples representing homomorphisms from a finitely presented group to symmetric groups.

Here is an example of testing $sameNsubgrp(\ ,\ )$: let t be a tuple representing a homomorphism[16] $\varphi : \mathrm{PB}_4 \to S_d$ (for concreteness, we could use $listE[19].PB4$) and g be a random permutation in S_d; using the command $tt = conj\_tup(g, t)$, we form another homomorphism $\varphi' : \mathrm{PB}_4 \to S_d$ with the same kernel; the command $sameNsubgrp(t, tt)$ (as well as the command $sameNsubgrp(tt, t)$) returns True.

Indirect testing of the generators $generWF2$ and $generWComm$

The function $test\_generWF2(\ )$ was used for testing the generator $generWF2$. For a positive integer d, the command $test\_generWF2(d)$ forms a tuple t of two random permutations in S_d; then a computer uses $generWF2$ to form the complete list $Wlist$ of words that represent elements of the permutation group[17] G generated by elements of t; a computer checks that the length of $Wlist$ coincides with the order of G; finally, a computer checks that the set of permutations corresponding to words in $Wlist$ coincides with the set of elements of G.

Similarly the function $test\_generWComm(\ )$ was used for testing the generator $generWComm$. For a positive integer d, the command $test\_generWComm(d)$ forms a tuple t of two random permutations in S_d; a computer forms the permutation group G generated by elements of t and forms the commutator subgroup H of G; then a computer uses $generWComm$ to form the complete list $Wlist$ of words that represent elements of H; a computer checks that the length of $Wlist$ coincides with the order of H; finally, a computer

[16] Of course, the same tuple t also represents a homomorphism from F_6 to S_d.
[17] It is very likely that $G = S_d$

checks that the set of permutations corresponding to words in $Wlist$ coincides with the set of elements of H.

Indirect testing of *conjBySig*1**,** *conjBySig*2 **and** *conjBySig*3

*conjBySig*1( ), *conjBySig*2( ) and *conjBySig*3( ) were tested (indirectly) using the function *test_conj_braid_rel*( ). The input t of *test_conj_braid_rel*( ) is a tuple (of permutations) that represents a homomorphism from PB_4 to a symmetric group. The command *test_conj_braid_rel*(t) returns True if

- $conjBySig1(conjBySig2(conjBySig1(t)))$ coincides with $conjBySig2(conjBySig1(conjBySig2(t)))$ **and**
- $conjBySig2(conjBySig3(conjBySig2(t)))$ coincides with $conjBySig3(conjBySig2(conjBySig3(t)))$ **and**
- $conjBySig1(conjBySig3(t))$ coincides with $conjBySig3(conjBySig1(t))$.

We used these lines:

```
for i in range(5):
    t=dessin2PB4(rand_dessin(d))
    print(test_conj_braid_rel(t))
```

for $d = 4, 5, 6, 7$ to test the functions *conjBySig*1( ), *conjBySig*2( ), *conjBySig*3( ) via the braid relations.

Indirect testing of *gener_dessin_pt*, *gener_dessin*, *gener_dessin_slow*, *all_dessin* **and** *all_dessin_slow*

The generators of child's drawings *gener_dessin* and *gener_dessin_slow* use different ways of producing child's drawings of a given degree. For an integer $d \geq 2$, the command *test_dessin*(d) compares the results of these two generators. More precisely, it forms the list of child's drawings of degree d (in the tuple format) using *gener_dessin* and the list of child's drawings of degree d (in the tuple format) using *gener_dessin_slow*. Then the function compares the lists of the corresponding child's drawings in the object format.

The total number of child's drawings of degree d (for $d \leq 6$) was also compared to the corresponding entry of the sequence in [15].

Indirect testing of the methods *compose* **and** *act* **of the class** *GTsh*

Consider $\mathsf{N}^{(21)}$. Since $\mathsf{N}^{(21)}$ is isolated, $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(21)})$ is a (finite) group. The list *GTcharm*[21] has length $54 = 2 \cdot 3^3$ and it contains all elements of $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(21)})$ in the object format.

Executing the line:

```
{p[0].compose(p[1]) in GTcharm[21] for p in prod(GTcharm[21],GTcharm[21])}
```

we get {True}. Thus we verified that the list *GTcharm*[21] is closed under composition.

The command *GTcharm*[21][0].*wm* returns $((), 0)$. In other words, *GTcharm*[21][0] represents the identity morphism of $\mathsf{N}^{(21)}$ in $\mathsf{GTSh}^{\heartsuit}$. Executing the lines:

{GTcharm[21][0].compose(T) == T for T in GTcharm[21]}
{T.compose(GTcharm[21][0]) == T for T in GTcharm[21]}

we get {True} and {True}. This way we tested the method *compose* using the identity property.

Executing[18] the lines:

```
triples = list(prod(GTcharm[21][1: ], GTcharm[21][1: ], GTcharm[21][1: ]))
Test =[ ]
for i in range(180):
    t=choice(triples)
    Test.append(t[0].compose(t[1]).compose(t[2])==t[0].compose(t[1].compose(t[2])))
    triples.remove(t)
set(Test)
```

we get {True}. This way we tested the method *compose* using the associativity property for 180 randomly selected triples of non-identity elements of $\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(21)})$.

Executing the lines:

```
def inv_exists(T):
    for TT in GTcharm[21][1: ]:
        if T.compose(TT) == GTcharm[21][0] and TT.compose(T) == GTcharm[21][0]:
            return True
    return False
{inv_exists(T) for T in GTcharm[21][1: ]}
```

we get {True}. This way we tested the method *compose* using the invertibility of each GT-shadow.

Let us now consider elements $\mathsf{N}^{(16)}$ and $\mathsf{N}^{(17)}$ from the list in (1.22). Executing the lines:

```
E16=listE[16]; E17=listE[17]
GT16=GTcharm[16]; GT17=GTcharm[17]
{T.src( )==E17 for T in GT16 if T.src( )!=E16}
{T.src( )==E16 for T in GT17 if T.src( )!=E17}
```

we get {True} and {True}. Thus $\mathsf{N}^{(16)}$ and $\mathsf{N}^{(17)}$ are not isolated, and $\{\mathsf{N}^{(16)}, \mathsf{N}^{(17)}\}$ is the set of objects of the connected component $\mathsf{GTSh}^{\heartsuit}_{\mathrm{conn}}(\mathsf{N}^{(16)})$ of $\mathsf{N}^{(16)}$ in the groupoid $\mathsf{GTSh}^{\heartsuit}$.

Executing the lines:

```
GT16_16=[T for T in GT16 if T.settled( )]
GT17_17=[T for T in GT17 if T.settled( )]
GT16_17=[T for T in GT17 if T.src( )==E16]
GT17_16=[T for T in GT16 if T.src( )==E17]
```

[18] You may need to wait for several minutes.

we form the lists $GT16\_16$, $GT17\_17$, $GT16\_17$ and $GT17\_16$ with the obvious content: for instance, $GT16\_17$ is the list of instances that represent all elements in $\mathsf{GTSh}^{\heartsuit}(\mathsf{N}^{(16)}, \mathsf{N}^{(17)})$.

Since the list $GT16\_16$ has 16 elements, the group $\mathsf{GTSh}^{\heartsuit}(\mathsf{N}^{(16)}, \mathsf{N}^{(16)})$ of automorphisms of $\mathsf{N}^{(16)}$ in the groupoid $\mathsf{GTSh}^{\heartsuit}$ has order 16. It is not surprising that all four lists $GT16\_16$, $GT17\_17$, $GT16\_17$ and $GT17\_16$ have the same length.

Executing the lines:

{T.inv( ) in GT17_16 for T in GT16_17}
{T.inv( ) in GT16_17 for T in GT17_16}
{T.inv( ) in GT16_16 for T in GT16_16}
{T.inv( ) in GT17_17 for T in GT17_17}

we get the outputs $\{\mathsf{True}\}\ldots\{\mathsf{True}\}$. This way we tested the method *inv*( ) of the class $GTsh$ in the set-up when targets of GT-shadows are not isolated.

Executing[19] the lines:

T=choice(GT16_17)
{T.compose(p[0]).compose(p[1])==
T.compose(p[0].compose(p[1])) for p in prod(GT16_16,GT16_16)}
{T.compose(p[0]).compose(p[1])==
T.compose(p[0].compose(p[1])) for p in prod(GT16_16,GT17_16)}
{T.compose(p[0]).compose(p[1])==
T.compose(p[0].compose(p[1])) for p in prod(GT17_16,GT17_17)}
{T.compose(p[0]).compose(p[1])==
T.compose(p[0].compose(p[1])) for p in prod(GT17_16,GT16_17)}

we get the outputs $\{\mathsf{True}\}\ldots\{\mathsf{True}\}$. This way we tested the method *compose*( ) of the class $GTsh$ using the associativity in the set-up when targets of GT-shadows are not isolated.

Due to [4, Theorem 3.16], the passport of a child's drawing is invariant with respect to the action of $\mathsf{GTSh}^{\heartsuit}$. This statement allowed us to test the method *act* indirectly as follows. Let us denote by

$$\mathcal{D}_{14,5},\quad \mathcal{D}_{15,4},\quad \mathcal{D}_{5,0},\quad \mathcal{D}_{6,0} \tag{7.2}$$

the child's drawing stored in files $dde14E15$, $dde15E29$, $dde5genus0$ and $dde6genus0$, respectively. These child's drawings are subordinate to $\mathsf{N}^{(15)}$, $\mathsf{N}^{(29)}$, the element N^{dde5} stored in the file $E\_dde5genus0$ and the element N^{dde6} stored in the file $E\_dde6genus0$, respectively. We showed that the corresponding orbits

$$\mathsf{GT}^{\heartsuit}(\mathsf{N}^{(15)})(\mathcal{D}_{14,5}),\quad \mathsf{GT}^{\heartsuit}(\mathsf{N}^{(29)})(\mathcal{D}_{15,4}),\quad \mathsf{GT}^{\heartsuit}(\mathsf{N}^{dde5})(\mathcal{D}_{5,0}),\quad \mathsf{GT}^{\heartsuit}(\mathsf{N}^{dde6})(\mathcal{D}_{6,0})$$

all have size 2. We verified that, for every child's drawing $\mathcal{D}$ in list (7.2), the passport of the child's drawing conjugate to $\mathcal{D}$ (via $\mathsf{GTSh}^{\heartsuit}$) coincides with the passport of $\mathcal{D}$.

We also observed that $\mathsf{GTSh}^{\heartsuit}$-orbits of selected child's drawings coincide with the corresponding $G_{\mathbb{Q}}$-orbits presented in [10]. For more details about such selected child's drawings, please see Section 5.3 of this note or [4, Section 5].

[19] Each command may take more than a minute.

A Some calculations related to the braid groups

Our conventions for the Artin braid group B_n and the pure braid group PB_n agree with those in [5, Appendix A]. In particular, we denote by $\sigma_1, \dots, \sigma_{n-1}$ the standard generators of B_n. We assume that the stands of geometric braids move up. The standard generators of PB_n are given by the formula

$$x_{ij} := \sigma_{j-1} \dots \sigma_{i+1} \sigma_i^2 \sigma_{i+1}^{-1} \dots \sigma_{j-1}^{-1}, \qquad 1 \le i < j \le n. \tag{A.1}$$

It is known [9, Section 1.3] that any relation on the standard generators of PB_n is a consequence of these relations

$$x_{rs}^{-1} x_{ij} x_{rs} = \begin{cases} x_{ij} & \text{if } \ s < i \text{ or } i < r < s < j, \\ x_{rj} x_{ij} x_{rj}^{-1} & \text{if } \ s = i, \\ x_{rj} x_{sj} x_{ij} x_{sj}^{-1} x_{rj}^{-1} & \text{if } \ r = i < s < j, \\ x_{rj} x_{sj} x_{rj}^{-1} x_{sj}^{-1}\, x_{ij}\, x_{sj} x_{rj} x_{sj}^{-1} x_{rj}^{-1} & \text{if } \ r < i < s < j. \end{cases} \tag{A.2}$$

Using [9, Theorem 1.16] and the above relations, one can see that, for every $n \ge 3$, PB_n is isomorphic to the semi-direct product of PB_{n-1} and the free group F_{n-1} on $n-1$ generators. More precisely,

$$\mathrm{PB}_n \cong K_n \ltimes U_n, \qquad K_n \cong \mathrm{PB}_{n-1}\,, \tag{A.3}$$

where U_n is freely generated by $x_{1,n}, \dots, x_{n-1,n}$ and K_n is generated by x_{ij} with $1 \le i < j \le n-1$.

For $n = 4$, we have

$$\begin{gathered} x_{12} = \sigma_1^2, \quad x_{23} = \sigma_2^2, \quad x_{13} = \sigma_2 \sigma_1 \sigma_2^{-1} = \sigma_1^{-1} \sigma_2 \sigma_1, \\ x_{14} = \sigma_3 \sigma_2 \sigma_1^2 \sigma_2^{-1} \sigma_3^{-1} = \sigma_1^{-1} \sigma_2^{-1} \sigma_3^2 \sigma_2 \sigma_1 = \sigma_1^{-1} \sigma_3 \sigma_2^2 \sigma_3^{-1} \sigma_1, \\ x_{24} = \sigma_3 \sigma_2^2 \sigma_3^{-1} = \sigma_2^{-1} \sigma_3^2 \sigma_2, \qquad x_{34} = \sigma_3^2\,. \end{gathered} \tag{A.4}$$

It is known [9, Corollary 1.20], that the Abelianization $\mathrm{PB}_n/[\mathrm{PB}_n, \mathrm{PB}_n]$ of PB_n is freely generated by images of $\{x_{ij}\}_{1 \le i < j \le n}$. In other words, $\mathrm{PB}_n/[\mathrm{PB}_n, \mathrm{PB}_n]$ is isomorphic to

$$\mathbb{Z}^{\frac{n(n-1)}{2}}\,.$$

For any $1 \le i \le n-2$, we have

$$(\sigma_i \sigma_{i+1})^3 = (\sigma_{i+1} \sigma_i)^3\,. \tag{A.5}$$

Indeed,

$$(\sigma_i \sigma_{i+1})^3 = (\sigma_i \sigma_{i+1} \sigma_i)(\sigma_{i+1} \sigma_i \sigma_{i+1}) = (\sigma_{i+1} \sigma_i \sigma_{i+1})(\sigma_i \sigma_{i+1} \sigma_i) = (\sigma_{i+1} \sigma_i)^3\,.$$

Moreover, if $n = 3$, then the element

$$c_3 := (\sigma_1 \sigma_2)^3 = (\sigma_2 \sigma_1)^3$$

belongs to the center of B_3 (and the center of PB_3).

Indeed,

$$\sigma_1 c_3 = \sigma_1\sigma_2\sigma_1\sigma_2\sigma_1\sigma_2\sigma_1 = \sigma_1\sigma_2\sigma_1\sigma_2\sigma_1\sigma_2\sigma_1 = c_3\,\sigma_1$$

and

$$\sigma_2 c_3 = \sigma_2\sigma_1\sigma_2\sigma_1\sigma_2\sigma_1\sigma_2 = \sigma_2\sigma_1\sigma_2\sigma_1\sigma_2\sigma_1\sigma_2 = c_3\,\sigma_2\,.$$

It is known [9, Theorem 1.24] that $\mathcal{Z}(\mathrm{B}_3) = \mathcal{Z}(\mathrm{PB}_3) = \langle\, c_3 \,\rangle$.

The group PB_3 is generated by

$$x_{12} = \sigma_1^2, \qquad x_{23} = \sigma_2^2, \qquad x_{13} = \sigma_2\sigma_1^2\sigma_2^{-1} = \sigma_1^{-1}\sigma_2^2\sigma_1\,, \tag{A.6}$$

where the identity $\sigma_2\sigma_1^2\sigma_2^{-1} = \sigma_1^{-1}\sigma_2^2\sigma_1$ follows from this calculation:

$$\sigma_2\sigma_1^2\sigma_2^{-1} = (\sigma_2\sigma_1\sigma_2^{-1})(\sigma_2\sigma_1\sigma_2^{-1}) = (\sigma_1^{-1}\sigma_2\sigma_1)(\sigma_1^{-1}\sigma_2\sigma_1) = \sigma_1^{-1}\sigma_2^2\sigma_1\,.$$

Proposition A.1 *In* PB_3*, we have*

$$x_{23}x_{12}x_{13} = c_3 = x_{12}x_{13}x_{23}\,. \tag{A.7}$$

Proof. Using (A.6), we get

$$x_{23}x_{12}x_{13} = \sigma_2^2\sigma_1^2\sigma_2\sigma_1^2\sigma_2^{-1} = \sigma_2\sigma_2\sigma_1\sigma_1\sigma_2\sigma_1\sigma_1\sigma_2^{-1} = \sigma_2\sigma_2\sigma_1\sigma_2\sigma_1\sigma_2\sigma_1\sigma_2^{-1}$$
$$= \sigma_2(\sigma_2\sigma_1)^3\sigma_2^{-1} = (\sigma_2\sigma_1)^3 = c_3.$$

The identity $x_{23}x_{12}x_{13} = x_{12}x_{13}x_{23}$ follows easily from $x_{23}^{-1}c_3x_{23} = c_3$. □

A.1 The generator of the center of B_4

Recall [9, Theorem 1.24] that, for every $n \geq 3$, the center $\mathcal{Z}(\mathrm{B}_n)$ of B_n coincides with the center $\mathcal{Z}(\mathrm{PB}_n)$ of PB_n. Moreover, $\mathcal{Z}(\mathrm{B}_n)$ is an infinite cyclic group generated by Δ_n^2 where

$$\Delta_n := (\sigma_1 \ldots \sigma_{n-1})(\sigma_1 \ldots \sigma_{n-2}) \ldots (\sigma_1\sigma_2)\sigma_1. \tag{A.8}$$

In particular, the center $\mathcal{Z}(\mathrm{B}_4) = \mathcal{Z}(\mathrm{PB}_4)$ is generated by the element

$$c_4 := \sigma_1\sigma_2\sigma_3\sigma_1\sigma_2\sigma_1^2\sigma_2\sigma_3\sigma_1\sigma_2\sigma_1. \tag{A.9}$$

Let us prove that

Proposition A.2 *The generator* c_4 *of* $\mathcal{Z}(\mathrm{PB}_4)$ *can be rewritten as*

$$c_4 = x_{14}x_{24}x_{34}x_{12}x_{13}x_{23}. \tag{A.10}$$

Proof. Since $\sigma_1\sigma_3 = \sigma_3\sigma_1$, we have

$$c_4 = \sigma_1\sigma_2\sigma_1(\sigma_3\sigma_2\sigma_1^2\sigma_2\sigma_3)\sigma_1\sigma_2\sigma_1\,. \tag{A.11}$$

Since $c_4 \in \mathcal{Z}(\mathrm{B}_4)$, $(\sigma_1\sigma_2\sigma_1)^{-1}c_4\sigma_1\sigma_2\sigma_1 = c_4$. Combining this observation with (A.7) and (A.11), we get

$$c_4 = (\sigma_3\sigma_2\sigma_1^2\sigma_2\sigma_3)\,\sigma_1\sigma_2\sigma_1\sigma_1\sigma_2\sigma_1 = \sigma_3\sigma_2\sigma_1^2\sigma_2\sigma_3\,x_{12}x_{13}x_{23},$$

i.e.

$$c_4 = \sigma_3\sigma_2\sigma_1^2\sigma_2\sigma_3\,x_{12}x_{13}x_{23}\,. \tag{A.12}$$

Using the definitions of x_{14}, x_{24} and x_{34}, we rewrite the element $\sigma_3\sigma_2\sigma_1^2\sigma_2\sigma_3$ as follows:

$$\sigma_3\sigma_2\sigma_1^2\sigma_2\sigma_3 = \sigma_3\sigma_2\sigma_1^2\sigma_2^{-1}\sigma_3^{-1}(\sigma_3\sigma_2^2\sigma_3) = x_{14}\sigma_3\sigma_2^2\sigma_3 = x_{14}(\sigma_3\sigma_2^2\sigma_3^{-1})\sigma_3^2 = x_{14}x_{24}x_{34}\,.$$

Combining this calculation with (A.12), we see that (A.10) indeed holds. □

B Justification of the code for $hexa1(\ ,\ )$ and $hexa2(\ ,\ )$

In this section, we work only with B_3 and PB_3. Moreover, we set $c := c_3$, where c_3 is the standard generator of the center of B_3.

Using the braid relation $\sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2$ and the definitions

$$x_{12} := \sigma_1^2, \qquad x_{23} := \sigma_2^2, \qquad c := (\sigma_1\sigma_2\sigma_1)^2\,,$$

it is easy to deduce the following identities

$$\begin{array}{ll} \sigma_1 x_{12}\sigma_1^{-1} = x_{12}, & \sigma_1 x_{23}\sigma_1^{-1} = zc, \\ \sigma_2 x_{12}\sigma_2^{-1} = uc, & \sigma_2 x_{23}\sigma_2^{-1} = x_{23}, \\ \sigma_1^{-1} x_{12}\sigma_1 = x_{12}, & \sigma_1^{-1} x_{23}\sigma_1 = uc, \\ \sigma_2^{-1} x_{12}\sigma_2 = zc, & \sigma_2^{-1} x_{23}\sigma_2 = x_{23}, \\ \sigma_1\sigma_2\, x_{12}\, \sigma_2^{-1}\sigma_1^{-1} = x_{23}, & \sigma_1\sigma_2\, x_{23}\, \sigma_2^{-1}\sigma_1^{-1} = zc, \\ \sigma_2\sigma_1\, x_{12}\, \sigma_1^{-1}\sigma_2^{-1} = uc, & \sigma_2\sigma_1\, x_{23}\, \sigma_1^{-1}\sigma_2^{-1} = x_{12}, \end{array} \tag{B.1}$$

where

$$z := x_{23}^{-1}x_{12}^{-1}, \qquad u := x_{12}^{-1}x_{23}^{-1}.$$

Let us use (B.1) to prove the following statement:

Proposition B.1 *Let* $\mathsf{K} \in \mathsf{NFI}_{\mathrm{PB}_3}(\mathrm{B}_3)$ *and* $(m, f) \in \mathbb{Z} \times \mathsf{F}_2$. *Then the first hexagon relation*

$$\sigma_1 x_{12}^m\, f^{-1}\sigma_2 x_{23}^m f\,\mathsf{K} \;=\; f^{-1}\sigma_1\sigma_2(x_{13}x_{23})^m\,\mathsf{K} \tag{B.2}$$

is equivalent to

$$x_{23}^m\, f x_{12}^m,\, f(x_{12}, z)^{-1} z^m f(x_{23}, z) \;\in\; \mathsf{K}, \tag{B.3}$$

and the second hexagon relation

$$f^{-1}\sigma_2 x_{23}^m f\,\sigma_1 x_{12}^m\,\mathsf{K} \;=\; \sigma_2\sigma_1(x_{12}x_{13})^m\, f\,\mathsf{K}\,, \tag{B.4}$$

is equivalent to

$$f(u, x_{12})^{-1} x_{12}^m f^{-1} x_{23}^m f(u, x_{23}) u^m \;\in\; \mathsf{K}. \tag{B.5}$$

Proof. Due to Proposition A.1, we have $x_{13}x_{23} = x_{12}^{-1}c$ and $x_{12}x_{13} = x_{23}^{-1}c$. Therefore, (B.2) and (B.4) are equivalent to

$$\sigma_1 x_{12}^m\, f^{-1}\sigma_2 x_{23}^m f\,\mathsf{K} \;=\; f^{-1}\sigma_1\sigma_2 x_{12}^{-m} c^m\,\mathsf{K} \tag{B.6}$$

and

$$f^{-1}\sigma_2 x_{23}^m f\,\sigma_1 x_{12}^m\,\mathsf{K} \;=\; \sigma_2\sigma_1 x_{23}^{-m} c^m\, f\,\mathsf{K}\,, \tag{B.7}$$

respectively.

Using (B.1), we rewrite the left hand side of (B.6) as follows

$$\sigma_1 x_{12}^m\, f^{-1}\sigma_2 x_{23}^m f\,\mathsf{K} = x_{12}^m\,\sigma_1 f^{-1}\sigma_1^{-1}\,\sigma_1\sigma_2 x_{23}^m f\,\mathsf{K} =$$

$$x_{12}^m\,\sigma_1 f^{-1}\sigma_1^{-1}\,\sigma_1\sigma_2 x_{23}^m f\sigma_2^{-1}\sigma_1^{-1}\sigma_1\sigma_2\,\mathsf{K} = x_{12}^m f^{-1}(x_{12}, zc) z^m f(x_{23}, zc)\sigma_1\sigma_2 c^m\,\mathsf{K}. \tag{B.8}$$

Similarly, using (B.1), we rewrite the right hand side of (B.6) as follows

$$f^{-1}\sigma_1\sigma_2 x_{12}^{-m} c^m\,\mathsf{K} = f^{-1} x_{23}^{-m}\sigma_1\sigma_2 c^m\,\mathsf{K}. \tag{B.9}$$

Combining (B.8) with (B.9), we conclude that (B.2) is equivalent to

$$x_{23}^m f\, x_{12}^m f^{-1}(x_{12}, zc)\, z^m f(x_{23}, zc) \;\in\; \mathsf{K}. \tag{B.10}$$

Similarly, applying (B.1) to (B.7), we get

$$f^{-1}\sigma_2 x_{23}^m f\, \sigma_1 x_{12}^m = f^{-1} x_{23}^m f(uc, x_{23}) u^m \sigma_2\sigma_1 c^m$$

and

$$\sigma_2\sigma_1 x_{23}^{-m} c^m\, f = x_{12}^{-m} f(uc, x_{12})\, \sigma_2\sigma_1 c^m\,.$$

Thus (B.7) is equivalent to

$$f(uc, x_{12})^{-1} x_{12}^m\, f^{-1} x_{23}^m\, f(uc, x_{23}) u^m \in \mathsf{K}. \tag{B.11}$$

Let q_1 (resp. q_2) be the sum of the exponents of x_{12} (resp. x_{23}) in the reduced form of $f \in \langle\, x_{12}, x_{23}\,\rangle$.

Since $c \in \mathcal{Z}(\mathrm{PB}_3)$, we can rewrite the expression $f^{-1}(x_{12}, zc) z^m f(x_{23}, zc)$ as follows

$$f^{-1}(x_{12}, zc) z^m f(x_{23}, zc) = f^{-1}(x_{12}, z) c^{-q_2} z^m f(x_{23}, z) c^{q_2} = f^{-1}(x_{12}, z) z^m f(x_{23}, z).$$

Similarly, we can rewrite the expression $f(uc, x_{12})^{-1} x_{12}^m\, f^{-1} x_{23}^m\, f(uc, x_{23})$ as follows

$$f(uc, x_{12})^{-1} x_{12}^m\, f^{-1} x_{23}^m\, f(uc, x_{23}) =$$

$$c^{-q_1} f(u, x_{12})^{-1} x_{12}^m\, f^{-1} x_{23}^m\, f(u, x_{23}) c^{q_1} = f(u, x_{12})^{-1} x_{12}^m\, f^{-1} x_{23}^m\, f(u, x_{23}).$$

Thus (B.10) (resp. (B.11)) is equivalent to (B.3) (resp. to (B.5)). □

References

[1] D. Bar-Natan, On Associators and the Grothendieck-Teichmueller Group I, Selecta Math. (N.S.) **4**, 2 (1998) 183–212; arXiv:q-alg/9606021

[2] D. Bar-Natan and Z. Dancso, Pentagon and hexagon equations following Furusho, Proc. Amer. Math. Soc. **140**, 4 (2012) 1243–1250.

[3] J.S. Birman, Braids, links, and mapping class groups, *Annals of Mathematics Studies*, **82**. Princeton University Press, Princeton, N.J.; University of Tokyo Press, Tokyo, 1974. ix+228 pp.

[4] V. A. Dolgushev, The Action of GT-Shadows on Child's Drawings, `https://arxiv.org/abs/2106.06645`

[5] V. A. Dolgushev, K.Q. Le and A.A. Lorenz, What are GT-shadows? `https://arxiv.org/abs/2008.00066`

[6] V. Drinfeld, On quasitriangular quasi-Hopf algebras and on a group that is closely connected with $\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$, Algebra i Analiz **2**, 4 (1990) 149–181.

[7] B. Fresse, Homotopy of operads and Grothendieck-Teichmueller groups. Part 1, The algebraic theory and its topological background, *Mathematical Surveys and Monographs,* **217**. AMS, Providence, RI, 2017. xlvi+532 pp.

[8] H. Furusho, Pentagon and hexagon equations, Ann. of Math. (2) **171**, 1 (2010) 545–556.

[9] C. Kassel and V. Turaev, Braid groups, *with the graphical assistance of Olivier Dodane,* Graduate Texts in Mathematics, **247.** Springer, New York, 2008. xii+340 pp.

[10] M. Musty, S. Schiavone, J. Sijsling and J. Voight, A database of Belyi maps, `https://beta.lmfdb.org/Belyi/`; a detailed description of its construction can be found in `https://arxiv.org/abs/1805.07751`

[11] SymPy, a Python library for symbolic mathematics, `https://www.sympy.org/en/index.html`

[12] Sympy Documentation. Permutations `https://docs.sympy.org/latest/modules/combinatorics/permutations.html`

[13] Sympy Documentation. Permutation Groups. `https://docs.sympy.org/latest/modules/combinatorics/perm_groups.html`

[14] D.E. Tamarkin, Formality of chain operad of little discs, Lett. Math. Phys. **66**, 1-2 (2003) 65–72.

[15] The On-Line Encyclopedia of Integer Sequences (OEIS), sequence A057005, `https://oeis.org/A057005`

Department of Mathematics, Temple University,
Wachman Hall Rm. 638
1805 N. Broad St.,
Philadelphia PA, 19122 USA
E-mail address: **vald@temple.edu**