1 Introduction

1.1 The poset $\mathcal{NFI}_{PB_4}(B_4)$ 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. 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 [3] that $\mathcal{NFI}_{PB_4}(B_4)$ is the poset of finite index normal subgroups $N \trianglelefteq B_4$ such that $N \leq PB_4$. For every such $N$, we denote by $N_{PB_3}$ and $N_{PB_2}$ the following finite index normal subgroups in $PB_3$ and in $PB_2$

$$N_{PB_3} := \varphi_{123}(N) \cap \varphi_{12,3,4}(N) \cap \varphi_{1,2,3,4}(N) \cap \varphi_{234}(N), \quad (1.1)$$

$$N_{PB_2} := \varphi_{12}^{-1}(N_{PB_3}) \cap \varphi_{12,3}(N_{PB_4}) \cap \varphi_{1,2,3}(N_{PB_3}) \cap \varphi_{23}(N_{PB_3}), \quad (1.2)$$

respectively.

The symbols $\varphi_{123}$, $\varphi_{12,3,4}$, $\varphi_{1,2,3,4}$, $\varphi_{234}$ (resp. $\varphi_{12}$, $\varphi_{12,3}$, $\varphi_{1,2,3}$, $\varphi_{23}$) in (1.1) (resp. in (1.2)), denote the group homomorphisms $PB_3 \to PB_4$ (resp. $PB_2 \to PB_3$) defined by the formulas:

$$\varphi_{123}(x_{12}) = x_{12}, \quad \varphi_{123}(x_{23}) = x_{23}, \quad \varphi_{123}(x_{13}) = x_{13},$$

$$\varphi_{234}(x_{12}) = x_{23}, \quad \varphi_{234}(x_{23}) = x_{34}, \quad \varphi_{234}(x_{13}) = x_{24},$$

$$\varphi_{12,3,4}(x_{12}) = x_{13}x_{23}, \quad \varphi_{12,3,4}(x_{23}) = x_{34}, \quad \varphi_{12,3,4}(x_{13}) = x_{14}x_{24},$$

$$\varphi_{1,2,3,4}(x_{12}) = x_{12}x_{13}, \quad \varphi_{1,2,3,4}(x_{23}) = x_{24}x_{34}, \quad \varphi_{1,2,3,4}(x_{13}) = x_{14},$$

$$\varphi_{1,2,34}(x_{12}) = x_{12}, \quad \varphi_{1,2,34}(x_{23}) = x_{23}x_{24}, \quad \varphi_{1,2,34}(x_{13}) = x_{13}x_{14},$$

and

$$\varphi_{12}(x_{12}) = x_{12}, \quad \varphi_{23}(x_{12}) = x_{23}, \quad \varphi_{12,3}(x_{12}) = x_{13}x_{23}, \quad \varphi_{1,2,3}(x_{12}) = x_{12}x_{13}. \quad (1.4)$$

It is easy to see that $N_{PB_3} \in \mathcal{NFI}_{PB_3}(B_4)$, $N_{PB_2} \in \mathcal{NFI}_{PB_2}(B_2)$ and the subgroup $N_{PB_2}$ is completely determined by its index $N_{od} := |PB_2 : N_{PB_2}|$.

For every $N \in \mathcal{NFI}_{PB_4}(B_4)$, the triple $N, N_{PB_3}, N_{PB_2}$ gives us a compatible equivalence relation $\sim_N$ on the truncation $PaB^{\leq 4}$ of the operad $PaB$ [1, 3 Appendix A], [6] Chapter 6], [13]. The quotient $PaB^{\leq 4}/\sim_N$ is a truncated operad in the category of finite groupoids.
A GT-pair with the target $N$ is a morphism of truncated operads $PaB^{\leq 4} \to PaB^{\leq 4}/ \sim_N$ and such morphisms are in bijection with pairs
\[(m, f_{PB_3}) \in \{0, 1, \ldots, N_{\text{ord}} - 1\} \times PB_3/N_{PB_3}\] (1.5)
satisfying the hexagon relations
\[
\begin{align*}
\sigma_1 x_{12}^m f^{-1} \sigma_2 x_{23}^m N_{PB_3} &= f^{-1} \sigma_1 \sigma_2 (x_{13} x_{23})^m N_{PB_3}, \\
 f^{-1} \sigma_2 x_{23}^m \sigma_1 x_{12}^m N_{PB_3} &= \sigma_2 \sigma_1 (x_{12} x_{13})^m f N_{PB_3},
\end{align*}
(1.6)
\]
and the pentagon relation
\[
\varphi_{234}(f) \varphi_{1,23,4}(f) \varphi_{123}(f) N = \varphi_{1,2,34}(f) \varphi_{12,34}(f) N. \tag{1.8}
\]
Both sides of (1.6) and (1.7) are elements of $B_3/N_{PB_3}$ and both sides of (1.8) are elements of $PB_4/N$.

It is convenient to represent GT-pairs by tuples $(m, f) \in \mathbb{Z} \times PB_3$ 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
\[
T_{PB_4}^{PB_3} : PB_4 \to PB_4/N, \quad T_{PB_3}^{PB_3} : PB_3 \to PB_3/N_{PB_3}, \quad T_{PB_2}^{PB_2} : PB_2 \to PB_2/N_{PB_2}. \tag{1.9}
\]
Explicit formulas for these homomorphisms are given in [3, Corollary 2.8].

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

\begin{itemize}
  \item $(m, f)$ obeys relations (1.6), (1.7), (1.8),
  \item $2m + 1$ represents a unit in the ring $\mathbb{Z}/N_{\text{ord}} \mathbb{Z}$, and
  \item the group homomorphism $T_{PB_3}^{PB_3} : PB_3 \to PB_3/N_{PB_3}$ is onto.
\end{itemize}

In this note, we tacitly identify GT-shadows (with the target $N$) with pairs (1.5) satisfying the above conditions and we denote by $GT(N)$ the set of GT-shadows with the target $N$. For $[m, f] \in GT(N)$, $T_{m,f}$ denotes the corresponding onto morphism of truncated operads
\[
T_{m,f} : PaB^{\leq 4} \to PaB^{\leq 4}/ \sim_N. \tag{1.10}
\]

Due to [3, Proposition 2.11], for every $N \in NFI_{PB_4}(B_4)$ and $[m, f] \in GT(N)$, the “kernel” of the morphism $T_{m,f}$ coincides with $\sim_K$, where
\[
K := \ker(PB_4 T_{m,f}^{PB_4} \to PB_4/N).
\]
We call the kernel of $T_{m,f}^{PB_4}$ the source of the GT-shadow $[m, f] \in GT(N)$.

Since the morphism in (1.10) is onto, it induces the following isomorphism of truncated operads:
\[
T_{m,f}^{\text{isom}} : PaB^{\leq 4}/ \sim_K \xrightarrow{\sim} PaB^{\leq 4}/ \sim_N, \tag{1.11}
\]
where $K := \ker(T_{m,f}^{PB_4})$. Moreover,
\[
T_{m,f} = P_N \circ T_{m,f}^{\text{isom}},
\]
In this note, we assume that all \(GT\) elements shadows. Namely, if \([m,f] \in \text{GTSh}(\mathbb{N})\) where \(m,N \in \text{NFI}_{PB_3}(B_4)\), the set of morphisms \(\text{GTSh}(K,N)\) from \(K\) to \(N\) is
\[
\text{GTSh}(K,N) := \{[m,f] \in \text{GT}(N) \mid \ker(T_{m,f}^{\text{PB}_3}) = K\}.
\]

The composition of morphisms comes from the obvious identification of \(\text{GTSh}(K,N)\) with the set of isomorphisms of truncated operads
\[
\text{PaB}^{\leq 4}/\sim_K \xrightarrow{\sim} \text{PaB}^{\leq 4}/\sim_N.
\]

For example, the pair \((0,1_{PB_3})\) represents the identity morphism from \(N\) to \(N\), the isomorphism \(T_{0,1_{PB_3}}^{\text{isom}}\) is the identity map and
\[
T_{0,1_{PB_3}} = \mathcal{P}_N : \text{PaB}^{\leq 4} \rightarrow \text{PaB}^{\leq 4}/\sim_N.
\]

Just as in \([3]\), 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 \(f \in F_2\). In this note, we assume that all GT-shadows are practical. From now on, \(\text{GT}(N)\) denotes the set of all practical GT-shadows with the target \(N\). Moreover, \(\text{GTSh}\) denotes the groupoid of practical GT-shadows.

Due to \([3]\), we have an explicit composition formula for practical GT-shadows. Namely, if \([m_1,f_1] \in \text{GTSh}(N^{(2)},N^{(1)}), [m_2,f_2] \in \text{GTSh}(N^{(3)},N^{(2)})\), and
\[
m := 2m_1m_2 + m_1 + m_2, \quad f(x,y) := f_1(x,y)f_2(x^{2m_1+1},f_1(x,y)^{-1}y^{2m_1+1}f_1(x,y)),
\]
then \([m,f] := [m_1,f_1] \circ [m_2,f_2]\), i.e. the pair \((m,f)\) represents the GT-shadow \([m_1,f_1] \circ [m_2,f_2] \in \text{GTSh}(N^{(3)},N^{(1)})\).

For \(N \in \text{NFI}_{PB_3}(B_4)\), we set
\[
N_{F_2} := N_{PB_3} \cap \langle x_{12}, x_{23} \rangle.
\]

For \([m,f] \in \text{GT}(N)\), the notation \(T_{m,f}^{\text{PB}_3}\) is reserved for the group homomorphism
\[
T_{m,f}^{\text{PB}_3} \big|_{F_2} : F_2 \rightarrow F_2/N_{F_2}.
\]

This homomorphism is given explicitly by the formulas
\[
T_{m,f}^{\text{PB}_3}(x_{12}) := x_{12}^{2m_1+1}N_{F_2}, \quad T_{m,f}^{\text{PB}_3}(x_{23}) := f^{-1}x_{23}^{2m_1+1}fN_{F_2}.
\]

A GT-shadow \([m,f] \in \text{GT}(N)\) is called charming \footnote{See Proposition 2.1 and Remark 2.3 in [3].} if the coset \(fN_{F_2}\) can be represented by \(f_1 \in [F_2,F_2]\). Equivalently, \(fN_{F_2} \in [F_2/N_{F_2},F_2/N_{F_2}]\).

For \(N \in \text{NFI}_{PB_3}(B_4)\), we denote by \(\text{GT}^\circ(N)\) the set of charming GT-shadows in \(\text{GT}(N)\). Due to \([3]\), charming GT-shadows form a subgroupoid of \(\text{GTSh}\) and we denote this subgroupoid by \(\text{GTSh}^\circ\).
A GT-shadow \([m, f] \in \text{GT}(N)\) is called **settled**, if its source coincides with its target \(N\), i.e.  
\[
\ker(T_{m,f}^{PB_4}) = N. \tag{1.13}
\]

In other words, settled GT-shadows with the target \(N\) are precisely automorphisms of \(N \in \text{NFI}_{PB_4}(B_4)\) in the groupoid \(\text{GTSh}\). Similarly, settled elements of \(\text{GT}^\triangledown(N)\) are precisely automorphisms of \(N \in \text{NFI}_{PB_4}(B_4)\) in the groupoid \(\text{GTSh}^\triangledown\).

An element \(N \in \text{NFI}_{PB_4}(B_4)\) is called **isolated** if every GT-shadow in \(\text{GT}^\triangledown(N)\) is settled.

It is clear that \(N \in \text{NFI}_{PB_4}(B_4)\) is isolated if and only if \(N\) is the only object of its connected component \(\text{GTSh}_{\text{con}}(N)\) in \(\text{GTSh}^\triangledown\). The notation \(\text{NFI}_{PB_4}^{\text{isolated}}(B_4)\) is reserved for the sub-poset of isolated elements of \(\text{NFI}_{PB_4}(B_4)\).

Due to [3] Proposition 3.3, for every \(N \in \text{NFI}_{PB_4}(B_4)\), the subgroup

\[
N^\sharp := \bigcap_{[m,f] \in \text{GT}(N)} \ker(T_{m,f}^{PB_4})
\]

is an isolated element of \(\text{NFI}_{PB_4}(B_4)\). Moreover, due to [3] Proposition 3.6, the intersection of two isolated elements of \(\text{NFI}_{PB_4}(B_4)\) is an isolated element of \(\text{NFI}_{PB_4}(B_4)\).

### 1.2 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 \(\mathfrak{B}_d\) denotes the set of partitions of \(d\). The notation \(ct\) is reserved for the standard map \(S_d \to \mathfrak{B}_d\) 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 \((ct(c_1), ct(c_2), ct(c_1^{-1}c_2^{-1}))\) is called the passport of \([c]\).

Just as in [4], we often represent child’s drawing of degree \(d\) by group homomorphisms \(\psi : F_2 \to S_d\) for which \(\psi(F_2)\) is a transitive subgroup of \(S_d\). The assignment

\[
\psi \mapsto (\psi(x), \psi(y)), \quad x := x_{12}, \quad y := x_{23}
\]

gives us the obvious bijection between such homomorphisms \(\psi : F_2 \to S_d\) 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 \(\ker(\psi) \leq F_2\) depends only on the child’s drawing \([\psi]\) but not on a particular choice of a representative \(\psi : F_2 \to S_d\).

Recall from [4] the following definition:

**Definition 1.1:** Let \(N \in \text{NFI}_{PB_4}(B_4)\) and \(\psi\) be a homomorphism \(F_2 \to S_d\) that represents a child’s drawing. We say that the child’s drawing \([\psi]\) is **subordinate** to \(N\) (or **subordinate** to the equivalence relation \(\sim_N\)) if

\[
N_{F_2} \leq \ker(\psi). \tag{1.14}
\]
If $\psi$ is subordinate to $N$, then we say that $N$ dominates $[\psi]$. We denote by $\text{Dessin}(N)$ the set of child’s drawings subordinate to $N$.

Let $\psi : F_2 \to S_d$ be a representative of a child’s drawing subordinate to $N$, $(m,f)$ be a pair that represents an element in $\mathbb{GT}(N)$ and $K := \ker(T_{m,f})$. We denote by $\psi^{(m,f)}$ the following homomorphism $F_2 \to S_d$

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

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 $\psi$ and $(m,f)$, and
- the assignments $\mathcal{A}^{\text{sh}}(N) := N$, $\mathcal{A}^{\text{sh}}([m,f])([\psi]) := [\psi^{(m,f)}]$ define a cofunctor $\mathcal{A}^{\text{sh}}$ from the category $\mathbb{GTSh}$ to the category $\text{Dessin}$.

In other words, the assignment $[\psi]^{[m,f]} := [\psi^{(m,f)}]$ defines a right action of $\mathbb{GT}$-shadows on the set of 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 $\mathbb{GTSh}$. Due to [4, Theorem 3.16], the passport of a child’s drawing is an invariant of the action of the subgroupoid $\mathbb{GTSh}^{\heartsuit}$ of charming $\mathbb{GT}$-shadows.

### 1.3 Formats of various objects related to the package

#### 1.3.1 Permutations and permutation groups

Just as C, Python starts counting at 0. For this reason, an element in $S_d$ is a bijection $\{0, 1, \ldots, d-1\} \rightarrow \{0, 1, \ldots, d-1\}$. In this package, permutations are represented by instances of the class `sympy.combinatorics.permutations.Permutation` (see [11]). The command `Permutation` from [11] 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 $Z_d$, 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].
1.3.2 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 $\text{PB}_4$ to $S_d$ is represented by a tuple

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

such that

$$ g_{23}g_{12}g_{13} \text{ commutes with } g_{12}, g_{23}, g_{13}, \quad (1.16) $$

$$ g_{12}^{-1}g_{14}g_{12} = g_{14}g_{24}g_{14}g_{24}^{-1}g_{14}^{-1}, \quad g_{12}^{-1}g_{24}g_{12} = g_{14}g_{24}g_{14}^{-1}, \quad g_{12}^{-1}g_{34}g_{12} = g_{34}, \quad (1.17) $$

$$ g_{23}^{-1}g_{14}g_{23} = g_{14}, \quad g_{23}^{-1}g_{24}g_{23} = g_{24}g_{34}g_{24}g_{34}^{-1}g_{24}^{-1}, \quad g_{23}^{-1}g_{34}g_{23} = g_{24}g_{34}g_{24}^{-1}, \quad (1.18) $$

$$ g_{13}^{-1}g_{14}g_{13} = g_{14}g_{34}g_{14}g_{34}^{-1}g_{14}^{-1}, \quad g_{13}^{-1}g_{24}g_{13} = g_{24}g_{34}^{-1}, \quad g_{13}^{-1}g_{34}g_{13} = g_{14}g_{34}g_{14}^{-1}, \quad (1.19) $$

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 $\psi : \text{PB}_4 \to S_d$ is a homomorphism represented by tuple [1.16], then this tuple also represents the normal subgroup $\ker(\psi) \trianglelefteq \text{PB}_4$.

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 $\psi_t$ 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 $F_k \to \langle X | R \rangle$, the normal subgroup $N_t^G \trianglelefteq G$ represented by a tuple $t$ corresponds to the normal subgroup $N_t \trianglelefteq F_k$ represented by the same tuple $t$. It is clear that $|F_k : N_t| = |G : N_t^G|$ 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 $t_{\text{cap}}$ be the tuple of elements in $S_{d_1} \times S_{d_2}$ that represents the homomorphism

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

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

These simple ideas are implemented in the definitions of the functions $\text{cap}(\ , \ )$, $N\text{subgrp.less.eq}(\ , \ )$, and $\text{sameNsubgrp}(\ , \ )$ 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 $y$. 

6
1.3.3 Formats for elements of \( \text{NFI}_{PB_4}(B_4) \)

We use two formats for elements of \( \text{NFI}_{PB_4}(B_4) \):

- **tuple format**: an element \( N \in \text{NFI}_{PB_4}(B_4) \) is represented by a group homomorphism \( \psi : PB_4 \rightarrow S_d \) such that \( \ker(\psi) = N \); the homomorphism \( \psi \) is, in turn, represented by a tuple of permutations satisfying the relations \((1.16), (1.17)\), \((1.18)\), \((1.19)\) and \((1.20)\).

- **object format**: an element \( N \in \text{NFI}_{PB_4}(B_4) \) is represented by an instance of the class \( \text{Equiv} \).

For example, if \( t \) is a tuple representing \( N \in \text{NFI}_{PB_4}(B_4) \), the command \( \text{Equiv}(t) \) returns the instance of the class \( \text{Equiv} \) that represents \( N \). Moreover, if \( E \) is an instance of the class \( \text{Equiv} \) representing \( N \in \text{NFI}_{PB_4}(B_4) \), the command \( E.PB_4 \) returns a tuple representing \( N \).

1.3.4 Formats for \( \text{GT} \)-shadows

Given \( N \equiv \text{NFI}_{PB_4}(B_4) \), we use two formats for elements of \( \text{GT}(N) \) (or candidates for elements of \( \text{GT}(N) \)):

- **tuple format**: an element of \( \text{GT}(N) \) 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 \( \mathbb{Z}/\text{Ord}\mathbb{Z} \);

- **object format**: an element of \( \text{GT}(N) \) is represented by an instance of the class \( \text{GTsh} \); for a tuple \( (w, m) \) and a tuple \( t \) that represents \( N \equiv \text{NFI}_{PB_4}(B_4) \), \( \text{GTsh}((w, m), t) \) is the instance of the class \( \text{GTsh} \) that represents the \( \text{GT} \)-shadow corresponding to \( (w, m) \).

For example, given an instance \( T \) of the class \( \text{GTsh} \) that represents a \( \text{GT} \)-shadow with the target \( N \), the command \( T.wm \) returns a representation \( (w, m) \) of this \( \text{GT} \)-shadow in the tuple format; the command \( T.tar \) returns the instance of the class \( \text{Equiv} \) that represents \( N \); \( T.g \) returns a permutation \( g \) corresponding to the coset \( wN_{F_2} \) \((g \) belongs to a permutation group isomorphic to the quotient \( F_2/N_{F_2} \)); finally, \( T.cc.ch \) returns the value of the virtual cyclotomic character.

1.3.5 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 \( \text{Dessin} \).
For example, if \( c \) is a tuple that represents a child’s drawing then \( \text{Dessin}(c) \) is an instance of the class \( \text{Dessin} \) that represents the child’s drawing \([c]\). If \( D \) is an instance of \( \text{Dessin} \) 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
\[
\text{Dessin}(\text{permut}(1, 2, 3, 6, 5), \text{permut}(0, 3, 6, 1, 5, 4)) ).full
\text{Dessin}(\text{permut}(1, 2, 3, 6, 5), \text{permut}(0, 3, 6, 1, 5, 4)) ).passport
\]
we get
\[
(\text{Permutation}(1, 2, 3, 6, 5), \text{Permutation}(0, 3, 6, 1, 5, 4), \text{Permutation}(0, 4, 5, 3, 2, 6))
\]
and
\[
( (5, 1, 1), (6, 1), (6, 1) )
\]
respectively.

**Remark 1.2** For all timed procedures, the time is given in **minutes**.

### 1.4 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 \( \mathcal{P}_{\mathcal{B}} \leq 4 \), 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 [3].

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

- \( \text{listE} \) is the list of compatible equivalence relations corresponding to 35 distinct elements

\[
N^{(0)}, N^{(1)}, \ldots, N^{(33)}, N^{(34)}
\]

of \( \text{NFI}_{\mathcal{P}_{\mathcal{B}}} \); the equivalence relations are given in the object format; Table 1 shows basic information about these 35 selected elements of \( \text{NFI}_{\mathcal{P}_{\mathcal{B}}} \);

- \( \text{GTcharm}_{\text{wm}} \) is the (nested) list which contains charming GT-shadows whose targets are elements of \( \text{listE} \), i.e. \( \text{len}(\text{GTcharm}_{\text{wm}}) = 35 \) and \( \text{GTcharm}_{\text{wm}}[i] \) is the list of charming GT-shadows with the target \( N^{(i)} \); the GT-shadows are given in the tuple format;

- \( \text{GTall}_{\text{wm}} \) is the (nested) list which contains all (practical) GT-shadows whose targets are elements of the first 31 entries of \( \text{listE} \), i.e. \( \text{len}(\text{GTall}_{\text{wm}}) = 31 \) and \( \text{GTall}_{\text{wm}}[i] \) is the complete list of (practical) GT-shadows with the target \( N^{(i)} \); the GT-shadows are given in the tuple format;
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 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 F_2/N_{F_2}^{(i)} is non-Abelian. Table 1 shows (in the order from left to right) the number of N^{(i)}, the index of N^{(i)} in PB_4, the index of N^{(i)}F_2 in F_2, the order of the commutator subgroup [F_2/N_{F_2}^{(i)}, F_2/N_{F_2}^{(i)}], N_{ord} := |PB_2 : N_{PB_2}^{(i)}|, the size of GT(N^{(i)}) (i.e. the total number of (practical) GT-shadows with the target N^{(i)}) and the size of GT^\diamond(N^{(i)}). Finally, the last column indicates whether N^{(i)} is isolated or not. Note that, for N^{(33)} and N^{(34)}, the exact numbers of (practical) GT-shadows is not known.

| i  | |PB_4 : N^{(i)}| |F_2 : N^{(i)}| |[F_2/N_{F_2}^{(i)}, F_2/N_{F_2}^{(i)}]| N_{ord} | |GT(N^{(i)})| |GT^\diamond(N^{(i)})| isolated?
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>16</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>16</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>36</td>
<td>4</td>
<td>3</td>
<td>18</td>
<td>6</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>63</td>
<td>7</td>
<td>3</td>
<td>36</td>
<td>12</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>63</td>
<td>7</td>
<td>3</td>
<td>36</td>
<td>12</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>288</td>
<td>8</td>
<td>6</td>
<td>72</td>
<td>12</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>24</td>
<td>144</td>
<td>4</td>
<td>6</td>
<td>72</td>
<td>12</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>48</td>
<td>144</td>
<td>4</td>
<td>6</td>
<td>72</td>
<td>12</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>1500</td>
<td>60</td>
<td>5</td>
<td>100</td>
<td>20</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>60</td>
<td>900</td>
<td>4</td>
<td>15</td>
<td>360</td>
<td>24</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>72</td>
<td>144</td>
<td>18</td>
<td>4</td>
<td>16</td>
<td>8</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>72</td>
<td>144</td>
<td>18</td>
<td>4</td>
<td>16</td>
<td>8</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>108</td>
<td>972</td>
<td>27</td>
<td>6</td>
<td>72</td>
<td>12</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>120</td>
<td>6000</td>
<td>60</td>
<td>10</td>
<td>400</td>
<td>40</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>147</td>
<td>441</td>
<td>49</td>
<td>3</td>
<td>216</td>
<td>72</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>168</td>
<td>8232</td>
<td>168</td>
<td>7</td>
<td>294</td>
<td>42</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>168</td>
<td>1344</td>
<td>168</td>
<td>4</td>
<td>64</td>
<td>32</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>168</td>
<td>1344</td>
<td>168</td>
<td>4</td>
<td>64</td>
<td>32</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>180</td>
<td>13500</td>
<td>60</td>
<td>15</td>
<td>600</td>
<td>40</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>216</td>
<td>7776</td>
<td>216</td>
<td>6</td>
<td>72</td>
<td>12</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>240</td>
<td>6000</td>
<td>60</td>
<td>10</td>
<td>400</td>
<td>40</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>324</td>
<td>8748</td>
<td>108</td>
<td>9</td>
<td>486</td>
<td>54</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>504</td>
<td>40824</td>
<td>504</td>
<td>9</td>
<td>486</td>
<td>54</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>504</td>
<td>24696</td>
<td>504</td>
<td>7</td>
<td>294</td>
<td>42</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>648</td>
<td>1296</td>
<td>162</td>
<td>4</td>
<td>32</td>
<td>16</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>720</td>
<td>54000</td>
<td>240</td>
<td>15</td>
<td>1800</td>
<td>120</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>1512</td>
<td>40824</td>
<td>504</td>
<td>9</td>
<td>486</td>
<td>54</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>1512</td>
<td>40824</td>
<td>504</td>
<td>9</td>
<td>486</td>
<td>54</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2520</td>
<td>63000</td>
<td>2520</td>
<td>5</td>
<td>200</td>
<td>40</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>2520</td>
<td>45360</td>
<td>2520</td>
<td>6</td>
<td>144</td>
<td>48</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>28224</td>
<td>225792</td>
<td>28224</td>
<td>4</td>
<td>512</td>
<td>256</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>181440</td>
<td>880560</td>
<td>181440</td>
<td>7</td>
<td>588</td>
<td>84</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>181440</td>
<td>9072000</td>
<td>181440</td>
<td>10</td>
<td>800</td>
<td>160</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>181440</td>
<td>40824000</td>
<td>181440</td>
<td>15</td>
<td>\geq 1800</td>
<td>120</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>762048</td>
<td>20575296</td>
<td>254016</td>
<td>9</td>
<td>\geq 4374</td>
<td>486</td>
<td>True</td>
<td></td>
</tr>
</tbody>
</table>

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 $N^{(i)}$, $|PB_4 : N^{(i)}|$, $|F_2 : N^{(i)}_F|$, the order of the commutator subgroup $[F_2/N^{(i)}_F, F_2/N^{(i)}]$, $N^{(i)}_\text{ord} := |PB_2 : N^{(i)}_P|$, and the number of charming GT-shadows with the target $N^{(i)}$.

For example, $N^{(19)}$ is called the Philadelphia subgroup of $PB_4$ and it is represented by the tuple of 6 permutations in $S_6$:

$$g_{12} := (1, 3, 2)(4, 6, 5), \quad g_{23} := (1, 4, 9)(2, 7, 6), \quad g_{13} := (1, 7, 5)(3, 6, 9),$$
$$g_{14} := (2, 6, 7)(3, 8, 5), \quad g_{24} := (1, 8, 6)(3, 4, 7), \quad g_{34} := (1, 2, 3)(7, 9, 8).$$

According to Table 1, $N^{(19)}$ is an isolated element of $NFI_{PB_4}(B_4)$. Executing the command, \{ $T$.settled() for $T$ in GTcharm[19] \}, we get \{ True \}. This confirms that $N^{(19)}$ is indeed an isolated element of $NFI_{PB_4}(B_4)$.

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

In fact, executing the commands

$$GT16,not,settled = [T \text{ for } T \text{ in } GTcharm[16] \text{ if not } T.\text{settled()}]$$
$$len(GT16,not,settled)$$

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

Executing the command

\{ $T$.src() == listE[17] for $T$ in GT16,not,settled \}

we get \{ True \}. So we see that, for every GT-shadow $T$ in the list GT16,not,settled, the source of $T$ is $N^{(17)}$.

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

Executing the command:

\{ $p$ for $p$ in comb(range(35), 2) if listE[p[1]].finer(listE[p[0]]) \}

we get

\{ (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) \}.

In other words, we got the set of all pairs $(i, j)$ with $0 \leq i < j \leq 34$ such that $N^{(j)} \subset N^{(i)}$.

For example, since $(8, 20)$ belongs to the above set of pairs, $N^{(20)} \subset N^{(8)}$. Since $(5, 20)$ is not in the above list, $N^{(20)} \not\subset N^{(5)}$. 

10
Executing the command \texttt{furusho\_test1(listE[5])} (resp. \texttt{furusho\_test\_comm1(listE[5])}) we get \texttt{False} (resp. \texttt{True}). This shows that the element \(N^{(5)}\) does not satisfy the strong Furusho property (see Property \ref{property:strong_furusho}), however \(N^{(5)}\) satisfies the weak Furusho property (see Property \ref{property:weak_furusho}).

The command \texttt{furusho\_test1(listE[24])} returns \texttt{True}, which means that the element \(N^{(24)}\) satisfies the strong Furusho property and hence \(N^{(24)}\) also satisfies the weak Furusho property.

The command \texttt{furusho\_test\_comm1(listE[25])} returns \texttt{False} which means that the element \(N^{(25)}\) does not satisfy the weak Furusho property and hence \(N^{(25)}\) does not satisfy the strong Furusho property.

Executing the commands
\begin{align*}
c8 &= (\text{permut}(7)(0,1,2)(3,4,5), \text{permut}(0,7,4)(1,3,6)) \\
D8 &= Dessin(c8)
\end{align*}
we create a child’s drawing \(D_{8,0}\) of degree 8 and genus 0 represented by the permutation triple
\begin{equation}
((1,2,3)(4,5,6), (1,8,5)(2,4,7), (1,3,7,4,6,8)(2,5)). \tag{1.23}
\end{equation}

Executing the command
\begin{equation*}
D8\text{.passport}
\end{equation*}
we see that \(D_{8,0}\) has the passport:
\begin{equation*}
((3,3,1,1), (3,3,1,1), (6,2)).
\end{equation*}

Executing the command \texttt{D8\_is\_Galois()}, we get \texttt{False}. Hence the child’s drawing \(D_{8,0}\) is not Galois.

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

Executing the command \{\texttt{Dessin(T.act(c8))==D8 for T in GTcharm[5]}\}, we get \{\texttt{True}\}. This means that the orbit \(GT^\circ(N^{(5)})(D_{8,0})\) is a singleton. Hence, for every element \(g\) of the absolute Galois group \(G_Q\) of rationals, \(D^g_{8,0} = D_{8,0}\).

\textbf{Contributors:} The following people contributed to this software\footnote{This example is also stored in the file \textit{dde}\_8E5E19.} package: Chelsea Zackey, Aidan Lorenz, Khanh Le and Vasily Dolgushev.

\textbf{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.

\footnote{The names of contributors are given in the reverse alphabetic order.}
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 [10]:

- 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 \( i \in \{0, 1, \ldots, d - 1\} \), the command \( i^g \) returns \( g(i) \); for instance, for \( g = permut(7)(0, 4, 1)(2, 3) \), the command

  \[
  [i^g \text{ for } i \text{ in range}(8)]
  \]

  returns the list \([4, 0, 3, 2, 1, 5, 6, 7]\);
- permutations in \( S_d \) act on the set \( \{1, 2, \ldots, d\} \) from the right; so, for \( g, h \in S_d \), the command \( g \ast h \) returns the composition of \( h \circ g \);
- 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 \( \geq 2 \); each cycle is represented as 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;
- SymPy has a 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 $\text{True}$ if $G$ acts transitively on $\{0, 1, \ldots, d - 1\}$, otherwise it returns $\text{False}$;

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

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

for a subgroup $H$ of a permutation group $G$, the command $G.commutator(G, H)$ returns the commutator subgroup $\left[G, H\right]$; 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 right cosets of $G$ by $H$ (as a list); for instance, if $H = PG(\text{permut}(1, 2))$, then the command

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

returns

$$[\text{Permutation}(2), \text{Permutation}(0, 2), \text{Permutation}(2)(0, 1)],$$

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

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 right coset $Hg$ in accordance with $G.coset\_transversal(H)$; for instance the command

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

returns $\{\text{True}\}$.

For more commands and examples, please see [11] and [12].

3 Selected functions of Aux.py

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

for a permutation $g$, $\text{display\_perm}(g)$ prints the nested tuple whose entries are cycles of the permutation; the shift $i \mapsto i + 1$ is incorporated; for example, $\text{display\_perm(\text{permut}(0, 4)(3, 2, 5))}$ returns the nested tuple $((1, 5), (3, 6, 4))$; note that cycles of length 1 are not shown; in particular, the commands $\text{display\_perm(\text{permut}(0, 4)(3, 2, 5))}$ and $\text{display\_perm(\text{permut}(18)(0, 4)(3, 2, 5))}$ return 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 $comp\_All(t)$ 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

$$tuple(conj(g, h) \text{ for } h \text{ in } t)$$

• 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.
for a tuple \( t \) of three permutations \( s_1, s_2, s_3 \) (of the same degree), \( relB4(t) \) returns \textbf{True} if the tuple \( t \) represents a homomorphism from \( B_4 \) to a symmetric group; otherwise, it returns \textbf{False};

for a tuple \( t \) representing a group homomorphism \( \varphi : B_4 \to S_d \), \( restr\_PB4(t) \) returns the tuple \( (g_{12}, g_{23}, g_{13}, g_{14}, g_{24}, g_{34}) \) which represents the homomorphism \( \varphi|_{PB_4} : PB_4 \to S_d \);

for \( d \in \mathbb{Z}_{\geq 1} \), \( generB4(d, \text{timed}) \) generates all group homomorphisms from \( B_4 \) to \( S_d \) up to conjugation by elements of \( S_d \); \( generB4_A(d, \text{timed}) \) 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 \( \text{tup2dict}((5, 3, 3, 1, 1)) \) returns the dictionary \{1 : 2, 3 : 2, 5 : 1\} and the command \( \text{dict2tup}\{1 : 2, 3 : 2, 5 : 1\} \) 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 \textbf{True} if there exists \( g \) in \( S_d \) such that \( \text{conj\_tup}(g)(t) \) coincides with \( tt \); otherwise, it returns \textbf{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 \( H_1, H_2 \) of \( G \), \( double\_coset\_reps(H_1, G, H_2) \) generates representatives of double cosets \( H_1 \backslash G/H_2 \); exactly one representative for each double coset.

\section{Classes, functions and methods of \textit{PaB.py}}

Here is the list of selected functions and generators in \textit{PaB.py}:

- for a tuple \( t \) of six permutations in \( S_d \), \( relPB4(t) \) returns \textbf{True} if \( t \) represents a homomorphism \( PB_4 \to S_d \); otherwise, the function returns \textbf{False};

- for a tuple \( t \) that represents a homomorphism \( PB_4 \to S_d \), \( cenPB4(t) \) returns the image of the generator

\[
c_4 = x_{14}x_{24}x_{34}x_{12}x_{13}x_{23}
\]

of the center \( Z(PB_4) \) of \( PB_4 \); see Proposition \[A.3\] in Appendix \[A.1\] this function was used for testing \textit{relPB4( )} indirectly;

- for tuples \( x, y \) of permutations representing homomorphisms from a free group on \( len(x) \) generators to symmetric groups, the command, \textit{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, \textit{Nsubgrpless}(x, y) \) returns \textbf{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 \textbf{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 \( \text{len}(x) == \text{len}(y) \);

- for tuples \( x, y \) of permutations representing homomorphisms from a free group on \( \text{len}(x) \) generators to symmetric groups, the command \( \text{sameNsubgrp}(x, y) \) returns \( \text{True} \) if the kernels of the corresponding homomorphisms coincide; otherwise, the command \( \text{sameNsubgrp}(x, y) \) returns \( \text{False} \); 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 \( \text{len}(x) == \text{len}(y) \);

- the functions \( fi1_23_4, fi123, fi12_3_4, fi123_4, fi234 \) are implementations of the homomorphisms \( \varphi_{123}, \varphi_{123_4}, \varphi_{123}, \varphi_{123_4} \); \( PB_3 \rightarrow PB_4 \) defined in \( (1.3) \); for instance, for a tuple \( t \) that represents a homomorphism \( \psi_t : PB_3 \rightarrow S_d \) the command \( fi1_23_4(t) \) returns the tuple that represents the homomorphism

\[
\psi_t \circ \varphi_{123_4} : PB_3 \rightarrow S_d;
\]

- similarly, the functions \( fi12, fi12_3, fi1_23, fi23 \) are implementations of the homomorphisms \( PB_3 \rightarrow PB_4 \) defined in \( (1.4) \); for instance, for a tuple \( t \) that represents a homomorphism \( \psi_t : PB_3 \rightarrow S_d \) the command \( fi1_23(t) \) returns a permutation that represents the homomorphism

\[
\psi_t \circ \varphi_{123} : PB_3 \rightarrow S_d;
\]

- for a tuple \( t \) that represents an element \( N \in \text{NFI}_{PB_4}(B_4) \), the command \( N\_PB3(t) \) returns a tuple that represents \( N_{PB_3} \in \text{NFI}_{PB_3}(B_3) \) defined in \( (1.1) \);

- for a tuple \( x \) that represents an element \( N_{PB_4} \in \text{NFI}_{PB_4}(B_3) \), the command \( \text{Nord}(x) \) returns the index \( |PB_2 : N_{PB_2}| \) of \( N_{PB_2} \), where \( N_{PB_2} \) is defined in \( (1.2) \);

- for a tuple \( \text{wm} = (w, m) \) and a tuple \( t \) of permutations that represents \( N \in \text{NFI}_{PB_4}(B_4) \), the command \( \text{penta}(\text{wm}, t) \) returns \( \text{True} \) if \( (w, m) \) satisfies pentagon relation \( (1.8) \) modulo \( N \); otherwise \( \text{penta}(\text{wm}, t) \) returns \( \text{False} \); 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 \( \text{wm} = (w, m) \) and a tuple \( tt \) of permutations that represents \( K \in \text{NFI}_{PB_4}(B_3) \), the command \( \text{hexa1}(\text{wm}, tt) \) returns \( \text{True} \) if \( (w, m) \) satisfies the first hexagon relation (see \( (1.6) \)) modulo \( K \); otherwise \( \text{hexa1}(\text{wm}, tt) \) returns \( \text{False} \); 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 \( \cite{B1} \) in Appendix \( \cite{B} \) for the explanation of line 552 in \( PaB.py \)

\[
tup = (x23 * * m, f x y, x12 * * m, f x z _ { i n v }, z * * m, f y z);
\]

- for a tuple \( \text{wm} = (w, m) \) and a tuple \( tt \) of permutations that represents \( K \in \text{NFI}_{PB_4}(B_3) \), the command \( \text{hexa2}(\text{wm}, tt) \) returns \( \text{True} \) if \( (w, m) \) satisfies the second hexagon relation (see \( (1.7) \)) modulo \( K \); otherwise \( \text{hexa2}(\text{wm}, tt) \) returns \( \text{False} \); 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 \( \cite{B1} \) in Appendix \( \cite{B} \) for the explanation of line 569 in \( PaB.py \)

\[
tup = (f u x _ { i n v }, x12 * * m, f x y _ { i n v }, x23 * * m, f u y, u * * m);
\]

\footnote{Recall that \( PB_2 \) is an infinite cyclic group generated by \( x_{12} \).}
• 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 $w2g((0, 0, 1), (\text{permut}(0, 1, 2), \text{permut}(1, 2)))$ returns $\text{Permutation}(0, 2)$, 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 $\varphi : F_2 \to S_d$, $\text{generWF2}(tt, \text{timed}=\text{None})$ generates all words in $F_2$ corresponding to distinct elements of the permutation group $\varphi(F_2) \leq S_d$; for each element $g \in \varphi(F_2)$, $\text{generWF2}(tt, \text{timed}=\text{None})$ yields exactly one $w \in F_2$ 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 $\varphi(F_2)$; for every word $w \in W$, the generator tests whether $\varphi(w + (0,))$ (resp. $\varphi(w + (1,))$) is a new permutation of $\varphi(F_2)$; if this is the case, the word $w + (0,)$ (resp. $w + (1,)$) is appended to $W\text{new}$ and the permutation $\varphi(w + (0,))$ (resp. $\varphi(w + (1,))$) is appended to $L\text{new}$; the “rank” of each new permutation is appended to the list $G$; at the end of each iteration, $W$ becomes $W\text{new}$ and $L$ becomes $L\text{new}$; if the second argument of $\text{generWF2}(\ , \ )$ is $\text{True}$, the generator prints the status update in the form

$\begin{align*}
\text{time elapsed (in minutes)} & \quad \text{the length of the list } G; \\
\text{the last status update for } & \text{generWF2}(tt, \text{True}) \text{ is}
\end{align*}$

$\begin{align*}
\text{time elapsed (in minutes)} & \quad \text{the order of the group } \varphi(F_2); \\
\end{align*}$

• for a tuple $t = (g_0, g_1)$ representing a homomorphism $\varphi : F_2 \to S_d$, the function $\text{generWComm}(t, \text{timed}=\text{None})$ generates words in $F_2$ that represent distinct elements of the commutator subgroup $[\varphi(F_2), \varphi(F_2)]$ of the permutation group $\varphi(F_2)$; the generator goes through all words in $F_2$ of the form $x^{k_1}y^{l_1}x^{k_2}y^{l_2} \ldots$ such that $k_1 + k_2 + \cdots \equiv 0 \text{ mod ord}(g_0)$ and $t_1 + t_2 + \cdots \equiv 0 \text{ mod ord}(g_1)$; if the second variable of $\text{generWComm}(\ , \ )$ is $\text{True}$, the generator prints the update every time it stores the next 1000 hashtags of new permutations;

• for a tuple $t$ representing $N \in \text{NFI}_{PB_4}(B_4)$, $\text{gener}_{GT, pr}(t)$ 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 $N \in \text{NFI}_{PB_4}(B_4)$ and $tt$ is a tuple that represents $N_{PB_3}$, then the command $\text{sourcePB3}(wm, tt)$ returns a tuple that represents $\ker(PB_3 \xrightarrow{\gamma_{PB_3}} PB_3/N) \leq PB_3$;

• if $wm$ is a tuple that represents a GT-pair with the target $N \in \text{NFI}_{PB_4}(B_4)$ and $tt$ is a tuple that represents $N$, then the command $\text{sourcePB4}(wm, tt)$ returns a tuple that represents $\ker(PB_4 \xrightarrow{\gamma_{PB_4}} PB_4/N) \leq PB_4$;
• AAA
• for a tuple \( t \) representing \( N \in \text{NFI}_{PB_4}(B_4) \), \( \text{gener}_{GT,sh}(t) \) generates all GT-shadows with the target \( N \); the outputs are in the tuple format;

• for a tuple \( t \) representing \( N \in \text{NFI}_{PB_4}(B_4) \), \( \text{gener}_{GT,charm}(t) \) generates all charming GT-shadows with the target \( N \); the outputs are in the tuple format;

• AAA

4.1 Elements of \( \text{NFI}_{PB_4}(B_4) \) and the class \( \text{Equiv} \)

The class \( \text{Equiv} \) has exactly one instance variable and this instance variable is a tuple of 6 permutations. Two instances \( E \) and \( EE \) of \( \text{Equiv} \) are equal if and only if the corresponding (normal) subgroups of \( PB_4 \) coincide. The class \( \text{Equiv} \) has the following data attributes:

• \textit{self.PB4} is the instance variable; in the remaining items, \( N \) denotes the element of \( \text{NFI}_{PB_4}(B_4) \) (i.e. the kernel of the homomorphism represented by \textit{self.PB4});

• \textit{self.PB3} is the tuple of 3 permutations; this tuple represents a group homomorphism from \( PB_3 \) to a symmetric group and the kernel of this homomorphism is \( N_{PB_3} \);

• \textit{self.N0} is \( N_{ord} \);

• \textit{self.xy} is the tuple of two permutations; it represents a group homomorphism \( F_2 \) to a symmetric group; the kernel of this homomorphism is \( N_{F_2} \);

• \textit{self.d4} is the degree of permutations in \textit{self.PB4};

• \textit{self.d3} is the degree of permutations in \textit{self.PB3}.

In the description of methods given below, \( N \) denotes the element of \( \text{NFI}_{PB_4}(B_4) \) represented by instance “\textit{self}” of the class \( \text{Equiv} \). This class has the following methods:

• \textit{self.ind4()} returns the index \( |PB_4 : N| \);

• \textit{self.ind3()} returns the index \( |PB_3 : N_{PB_3}| \);

• \textit{self.indF2()} returns the index \( |F_2 : N_{F_2}| \);

• \textit{self.commF2()} returns the commutator subgroup of \( F_2/N_{F_2} \); the type of the output is \textit{sympy.combinatorics.perm_groups.PermutationGroup}; for examples, the command \textit{self.commF2().order()} returns the order of the commutator subgroup \( [F_2/N_{F_2}, F_2/N_{F_2}] \) of \( F_2/N_{F_2} \).

• AAA I have to describe methods relPB4, finer, cap, subord.

4.2 GT-shadows and the class \( \text{GTsh} \)

AAA
4.3 The class \textit{GTsh}

As we mentioned above, the class \textit{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 \(N \in \text{NFI}_{PB_4}(B_4)\).

Note that two instances \(T\) and \(TT\) represent the same \textit{GT}-shadows in \(\text{GT}(N)\) for a fixed \(N\) if and only if the command \(T == TT\) returns \text{True}.

The class \textit{GTsh} has the following data attributes:

- \(\text{self.wm}\) returns the first instance variable of \(\text{self}\), i.e. the tuple \((w, m)\), where the tuple \(w\) represents an element of \(F_2\) and \(m\) is a non-negative integer;
- \(\text{self.w}\) returns the corresponding word in \(F_2\), i.e. \(\text{self.wm}[0]\);
- \(\text{self.tar}\) returns the instance of the class \textit{Equiv}; this instance represents the element in \(\text{NFI}_{PB_4}(B_4)\) which is the target \(\text{self}\) viewed as the morphism in the groupoid \(\text{GTSh}\); for instance, the command \(\text{self.tar}.PB4\) returns the tuple, which represents \(N \leq PB_4\) we need to write \(\text{self.tar}.PB4\);
- \(\text{self.cc.ch}\) returns the value of the virtual cyclotomic character of \(\text{self}\);
- \(\text{self.g}\) returns the image of \(\text{self.w}\) in the permutation group isomorphic to \(F_2/NF_2\).

The class \textit{GTsh} has the following methods:

- \(\text{self.is_GTpr()}\) returns \text{True} if the instance is a friendly \(\text{GT}\)-pair; otherwise, \text{False};
- if \(\text{self}\) represents a \textit{GT}-shadow, then \(\text{self.src()}\) returns the source of \(\text{self}\) viewed as the morphism in the groupoid \(\text{GTSh}\);
- \(\text{self.is_GTsh()}\) returns \text{True} if \(\text{self}\) represents a (practical) \(\text{GT}\)-shadow; otherwise \text{False};
- \(\text{self.is_charm()}\) returns \text{True} if \(\text{self}\) is a charming \(\text{GT}\)-shadow; otherwise \text{False};
- \(\text{self.settled()}\) returns \text{True} if the \(\text{GT}\)-shadow \(\text{self}\) is settled, i.e. the source of \(\text{self}\) coincides with the target of \(\text{self}\);
- let \(N\) be the target of \(\text{self}\) and \(E\) be an instance of the class \textit{Equiv} which represents \(N^E \in \text{NFI}_{PB_4}(B_4)\) and satisfies the property \(N \leq N^E\); \(\text{self.proj}(E)\) returns the image of \(\text{self}\) in \(\text{GT}(N^E)\) with respect to the natural projection \(\text{GT}(N) \rightarrow \text{GT}(N^E)\).
- let \(N\) be the target of \(\text{self}\) and \(E\) be an instance of the class \textit{Equiv} which represents \(N^E \in \text{NFI}_{PB_4}(B_4)\) and satisfies the property \(N^E \leq N\); the command \(\text{self.survives}(E)\) returns \text{True} if \(\text{self}\) belongs to the image of the natural projection \(\text{GT}(N^E) \rightarrow \text{GT}(N)\) (i.e. \(\text{self}\) survives into \(N^E\));
- \(\text{other}\) is an instance of the class \textit{GTsh} whose source coincides with the target of \(\text{self}\); the command \(\text{self.compose(other)}\) returns the composition of \(\text{GT}\)-shadows \(\text{self}\) and \(\text{other}\); the composition works according to the rule: if \(\text{self} \in \text{GTSh}(N(1), N(2))\) and \(\text{other} \in \text{GTSh}(N(2), N(3))\) then \(\text{self} \circ \text{other} \in \text{GTSh}(N(1), N(3))\); the method tests whether the source of \(\text{other}\) coincides with the target of \(\text{self}\); if they do not coincide the command \(\text{self.compose(other)}\) returns an error message;
• assuming that \textit{self} is a charming GT-shadow, the command \textit{self.inv()} returns the inverse of \textit{self} in the groupoid \textit{GTSh}; note that the method finds the inverse by generating GT-shadows in \textit{GT}^{\textit{S}}(\textit{N}) where \textit{N} is the source of \textit{self}, so it may work very slowly if the index \(|\text{PB}_4 : \textit{N}|\) is large;

• let \(c\) be a child’s drawing represented by a tuple of two permutations; it is assumed that the isomorphism class \([c]\) of \(c\) is subordinate to the target of \textit{self}; the command \textit{self.act(c)} returns the result of the (right) action of \textit{self} on \(c\).

4.4 The class \textit{Dessin}

Instances of the class \textit{Dessin} represent child’s drawings. The instance variable \textit{pr} is a tuple of two permutations \((c_1, c_2)\) in \(S_d\) such that the group \(\langle c_1, c_2 \rangle\) acts transitively on the set \(\{1, \ldots, d\}\). For two instances \(D\) and \(DD\) of \textit{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 \textit{Dessin} has the following data attributes:

• \textit{self.pr} returns the instance variable, i.e. a tuple \((c_1, c_2)\) of permutations which represents the child’s drawing;

• \textit{self.d} returns the degree of the child’s drawing represented by \textit{self};

• \textit{self.full} returns the permutation triple \((c_1, c_2, c_2^{-1}c_1^{-1})\);

• \textit{self.passport} returns the passport of the child’s drawing represented by \textit{self}; the output of \textit{self.passport} is a nested tuple of length 3; each entry of this tuple is a partition of \(d\); for instance, \(Dessin((\text{permut}(2, 3), \text{permut}(0, 1, 2))).passport\) returns the tuple \(((2, 1, 1), (3,), (4,))\).

The class \textit{Dessin} has the following methods:

• \textit{self.monG()} returns the monodromy group \(\langle c_1, c_2 \rangle\) of the child’s drawing \textit{self};

• \textit{self.is_transitive()} returns True if \textit{self} indeed represents a child’s drawing, i.e. if the group \(\langle c_1, c_2 \rangle\) acts transitively on \(\{1, \ldots, d\}\); otherwise, the method returns False;

• \textit{self.genus()} returns the genus of the covering of \(\mathbb{CP}^1 \setminus \{0, 1, \infty\}\) corresponding to the child’s drawing \textit{self};

• \textit{self.is_Galois()} returns True if the covering map corresponding to \textit{self} is Galois; otherwise, the method returns False;

• for a tuple \(c\) that represents a child’s drawing of degree \(d\), the command \textit{dessin2PB4(c)} returns a tuple that represents a group homomorphism \(\varphi : \text{PB}_4 \rightarrow S_d\); the intersection of \(F_2 \leq \text{PB}_3 \leq \text{PB}_4\) with the kernel of \(\varphi\) is the kernel of the group homomorphism \(F_2 \rightarrow S_d\) corresponding to \(c\).

• \(B4\_inv()\)
5 Playing with selected examples of elements of NFI_{PB_4}(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 \([m, f] \in \text{GT}(N)\) is \textbf{fake} if it does not come from an element of \(\text{GT}\). A GT-shadow \([m, f] \in \text{GT}(N)\) if and only if the exists \(K \in \text{NFI}_{PB_4}(B_4)\) such that \(K \leq N\) and \([m, f]\) does not belong to the image of the natural map

\[
\text{GT}(K) \rightarrow \text{GT}(N).
\]

(5.1)

If \([m, f]\) does not belong to the image of (5.1) then we say that \([m, f]\) \textbf{does not survive} to \(K\).

The function \texttt{charm\_fake4isolated}( , , ) has 3 arguments: \(E, EE, num\). \(E\) is an isolated compatible equivalence relation, \(EE\) is a finer equivalence relation and \(num\) is the total number of charming GT-shadows whose target is \(E\). The function returns \texttt{True} if every GT-shadow in \(\text{GT}(E)\) survives into \(EE\). Otherwise, it returns \texttt{False}. The function generates all GT-shadows in \(\text{GT}(EE)\) and projects then to \(E\). If it gets > \(num//2\) distinct charming GT-shadows in \(\text{GT}(E)\) then the function returns \texttt{True}.

We looked at the example \(N := N(19) \cap N(23)\). Both \(N(19)\) and \(N(23)\) are isolated. \(\text{GT}(N(19))\) (resp. \(\text{GT}(N(23))\)) is a group of order 12 (resp. 42). For the quotient \(F_2/N_{F_2}\) has order 192, 036, 096 and its commutator subgroup has order 108, 864. On a MacBook Air (the processor: 1.6 GHz Intel Core i5), it took almost 2 hours to verify (using \texttt{charm\_fake4isolated}( , , )) that every charming GT-shadow in \(\text{GT}(N(19))\) survives into \(N\). On the other hand, it took roughly a minute to verify (using \texttt{charm\_fake4isolated}( , , )) that every charming GT-shadow in \(\text{GT}(N(23))\) survives into \(N\).

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. For a precise formulation of this statement, we refer the reader to \([2\text{, Theorem 3.1]}\) and \([7\text{, Theorem 1]}\).

Following \([3]\), we say that an element \(N \in \text{NFI}_{PB_4}(B_4)\) satisfies the \textit{strong Furusho property} if

\textbf{Property 5.1} For every \(fN_{F_2} \in F_2/N_{F_2}\) satisfying pentagon relation (1.8) modulo \(N\), there exists \(m \in \mathbb{Z}\) such that

\begin{itemize}
  \item \(2m + 1\) represents a unit in \(\mathbb{Z}/N_{\text{ord}}\mathbb{Z}\) and
  \item the pair \((m, f)\) satisfies hexagon relations (1.6), (1.7).
\end{itemize}

Since every genuine GT-shadow can be represented by a pair \((m, f)\) with \(f \in [F_2,F_2]\), it makes sense to consider the weaker version of Property 5.1

\textbf{Property 5.2} For every \(fN_{F_2} \in [F_2/N_{F_2},F_2/N_{F_2}]\) satisfying pentagon relation (1.8) modulo \(N\), there exists \(m \in \mathbb{Z}\) such that

\begin{itemize}
  \item \(2m + 1\) represents a unit in \(\mathbb{Z}/N_{\text{ord}}\mathbb{Z}\) and
the pair \((m, f)\) satisfies hexagon relations \((1.6), (1.7)\).

Since there are examples of \(N \in \mathbf{NFI}_{PB_4}(B_4)\) that satisfy neither Property 5.1 nor Property P:Furusho-weak, one could say that there is NO version of Furusho’s theorem for GT-shadows. Still, it is worth mentioning that some elements \(N \in \mathbf{NFI}_{PB_4}(B_4)\) satisfy Property 5.2 and some of these elements even satisfy Property 5.1.

Using the function \(furusho\_test1\), we showed that every \(N\) in the following list of 11 elements (out of 35 elements in (1.21))

\[
N^{(1)}, N^{(2)}, N^{(3)}, N^{(4)}, N^{(6)}, N^{(7)}, N^{(9)}, N^{(10)}, N^{(11)}, N^{(14)}, N^{(24)}
\]  

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 (1.21))

\[
N^{(0)}, N^{(1)}, N^{(2)}, N^{(3)}, N^{(4)}, N^{(5)}, N^{(6)}, N^{(7)}, N^{(9)}, N^{(10)}, N^{(11)}, N^{(14)}, N^{(24)}
\]  

satisfy Property 5.2.

We should remark, even if \(N \in \mathbf{NFI}_{PB_4}(B_4)\) satisfies Property 5.1, it does not mean that every element in \((m, fN_{F_2}) \in \{0, 1, \ldots, N_{ord} - 1\} \times F_2/N_{F_2}\) satisfying the pentagon relation and the hexagon relations is a GT-shadow. More precisely, for some of these pairs, the homomorphism \(T^{PB_3}_{m, f} : PB_3 \to PB_3/N_{PB_3}\) is not onto. For instance, if \(N\) is the Philadelphia subgroup \(N^{(19)}\) and

\[
\text{Fur}(N^{(19)}) := \{(m, fN_{F_2}) \in \{0, \ldots, N_{ord} - 1\} \times F_2/N_{F_2} \mid (m, fN_{F_2}) \text{ satisfies } (1.6), (1.7), (1.8)\}
\]

5.3 Selected examples of child’s drawings

The child’s drawing \(D_{15}\) represented by the permutation triple

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

is subordinate to the equivalence relation represented by \(listE[29]\).

Its passport is

\[(6, 6, 3), (6, 6, 3), (6, 6, 3)\]

and its genus is 4.

The orbit of \(D_{15}\) with respect to the action of \(GT^\odot(N^{29})\) has size 2. The second child’s drawing in the orbit of \(D_{15}\) is represented by the permutation triple

\[
((1, 6, 5, 4, 3, 2)(7, 12, 11, 10, 9, 8)(13, 15, 14),
(1, 15, 9, 12, 6, 2)(3, 13, 7)(4, 10, 8, 5, 14, 11),
(1, 6, 2, 7, 10, 14)(3, 11, 9, 4, 8, 15)(5, 12, 13)).
\]  

\(D_{15}\) is stored (in the tuple format) in the storage file \(dde15E29\).
6 Descriptions of storage files

*subGrPB4_org35* contains the list of 35 selected elements (1.21) of NFI_{PB4}(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 GT^{\diamond}(N^{(i)}) (in the tuple format), where 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 ≤ i ≤ 30) is the list of all GT-shadows (in the tuple format) in GT(N^{(i)}), where N^{(i)} is the i-th entry of the list stored in *subGrPB4_org35*.

*wm_list_all31* contains the list of all elements in GT(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 GT(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 N^{(34)}; each GT-shadow is given in the tuple format; note that GT(N^{(34)}) may contain more (practical) GT-shadows.

*Leila_PB4* contains Leila’s subgroup N^C ∈ NFI_{PB4}(B_4) and it is given in the tuple format; here is the basic information about N^C:

- N^C is the kernel of a group homomorphism PB_4 → S_{130}, N^C_{PB4} is the kernel of a homomorphism PB_3 → S_{130} and \(N^C_{ord} = 12\).
- \(|PB_4 : N^C| = 285, 315, 214, 344, 192 = 2^2 \cdot 3^1 \cdot 2 \approx 3 \cdot 10^{14}\),
- \(|PB_3 : N^C_{PB3}| = 2, 985, 984 \approx 3 \cdot 10^6\),
- \(|F_2 : N^C_{F2}| = 248, 832\), the order of the commutator subgroup of F_2/N^C_{F2} is 1728 = 2^6 \cdot 3^3,
- N^C is an isolated element of NFI_{PB4}(B_4) and GT^{\diamond}(N^C) is a non-Abelian group of order 48 = 2^4 \cdot 3.

*wm_list_Leila* contains the list of all elements of GT^{\diamond}(N^C) in the tuple format. As we mentioned above, GT^{\diamond}(N^C) 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. Every 2-Sylow subgroup of GT^{\diamond}(N^C) is non-Abelian.

*dde8E5E19* contains a child’s drawing D_{8,0} (in the tuple format) of degree 8 and genus 0; D_{8,0} is subordinate to N^{(5)} and N^{(19)}; its passport is \(((3, 3, 1, 1), (3, 3, 1, 1), (6, 2))\);

there are 5 child’s drawings with the same passport; the orbit GT^{\diamond}(N^{(5)})(D_{8,0}) is a singleton (hence the GQ-orbit of D_{8,0} is also a singleton).
• $dde15E29$ contains a child’s drawing (in the tuple format) of degree 15 that is subordinate to $N(29)$ (i.e. the equivalence relation represented by $listE[29]$); the passport of this child’s drawing is

$((6, 6, 3), (6, 6, 3), (6, 6, 3))$

and its genus is 4; the size of its orbit, with respect to $N(29)$, is 2.

• $dde6genus0$ contains the child’s drawing (in the tuple format)

$((1, 4, 5, 2)(3, 6), (1, 6, 3, 2)(4, 5), ((1, 3), (2, 4)))$

of degree 6 corresponding to Belyi map $\text{https://beta.lmfdb.org/Belyi/6T10/4.2/4.2/2.2.1.1/a}$; the passport of this child’s drawing is

$((4, 2), (4, 2), (2, 2, 1, 1))$

and its genus is 0; the size of its $G_Q$-orbit is 2; its Galois conjugate is represented by the permutation triple

$((1, 4, 5, 2)(3, 6), (1, 2, 5, 6)(3, 4), (3, 5)(4, 6))$

• $E_{dde6genus0}$ contains an element of $NFI_{PB_4}(B_4)$ (in the tuple format) that dominates the child’s drawing stored in $dde6genus0$; the element $N \in NFI_{PB_4}(B_4)$ stored in $E_{dde6genus0}$ is isolated and $GT^{G}(N)$ is a non-Abelian group of order $32 = 2^5$.

• $dde7E29$ contains the child’s drawing (in the tuple format)

$((1, 2, 3)(4, 5)(6, 7), (1, 5, 6)(2, 7)(3, 4), (1, 4)(2, 6)(3, 7, 5))$

this child’s drawing is subordinate to $N(29)$, its passport is $((3, 2, 2), (3, 2, 2), (3, 2, 2))$ and its genus is zero. A Belyi map that represents this child’s drawing can be found at $\text{https://beta.lmfdb.org/Belyi/7T6/3.2.2/3.2.2/3.2.2/a}$.

• $dde18E29$ contains the child’s drawing $D_{18,4}$ represented by the permutation triple:

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

• More interesting examples of child’s drawings should be added.

7 Testing

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

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

returns $\{\text{True}\}$. This confirms that, for every equivalence relation $E$ in list listE of 35 elements, the corresponding subgroup $N_E \leq PB_4$ 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 \( \chi : G \to \hat{\mathbb{Z}}^\times \) is an onto group homomorphism. Hence, for every \( N \in \text{NF}_\text{PB}_4(B_4) \), the virtual cyclotomic character

\[
\text{Ch}_\text{cyclot} : \text{GT}^\diamond(N) \to (\mathbb{Z}/\text{ordZ})^\times
\]  

(7.1)
is onto.

For a tuple \( t \) representing \( N \in \text{NF}_\text{PB}_4(B_4) \), the command \texttt{test\_cyclotomic}(t) prints the image of the virtual cyclotomic character and returns \texttt{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 \texttt{False}.

The function \texttt{test\_cyclotomic} was used to test all elements of the list \( \text{listE} \) of 35 equivalence relations.

**Indirect testing of relPB4( )**

The function \texttt{relPB4} was tested (indirectly) using the explicit formula for the standard generator \( c_4 \) of the center \( Z(PB_4) \) of \( PB_4 \). (See (A.12) in Appendix [A.1]). For a tuple of \( t \) representing a group homomorphism \( PB_4 \to S_d \), the command \texttt{cenPB4}(t) returns the image of \( c_4 \) in \( S_d \). For a tuple of 6 permutations in \( S_d \), the command \texttt{test\_relPB4}(t) returns \texttt{True} if the permutation \( \text{cenPB4}(t) \) commutes with each entry of the tuple \( t \); otherwise, it returns \texttt{False}.

If you execute the command \{\texttt{test\_relPB4}(E.PB4) for E in listE\}, you get \{\texttt{True}\}.

**Indirect testing of N_PB3 and cap**

Let \( t \) be a tuple representing a homomorphism from \( PB_4 \) to \( S_d \) and \( N \subseteq PB_4 \) be the kernel of this homomorphism. Both commands \texttt{N\_PB3}(t) and \texttt{N\_PB3\_1}(t) return a homomorphism \( \varphi \) from \( PB_3 \) to a symmetric group whose kernel is

\[
N_{PB_4} := \varphi_{123}^{-1}(N) \cap \varphi_{12,3,4}^{-1}(N) \cap \varphi_{1,23,4}^{-1}(N) \cap \varphi_{1,2,34}^{-1}(N) \cap \varphi_{234}^{-1}(N)
\]

Unlike \texttt{N\_PB3}, the function \texttt{N\_PB3\_1} does not use \texttt{cap}. This is why, the degree of the permutation group \( \varphi(PB_3) \) for \texttt{N\_PB3\_1} may be bigger than the degree of the corresponding permutation group for \texttt{N\_PB3}. Of course, the kernel of the homomorphism \( PB_3 \to S_d \) corresponding to \texttt{N\_PB3}(t) coincides with the kernel of the homomorphism \( PB_3 \to S_{d_1} \) corresponding to \texttt{N\_PB3\_1}(t). This observation was used for testing for testing \texttt{N\_PB3} and \texttt{cap} indirectly.

**Indirect testing of sameNsubgrp( , )**

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

Here is an example of testing \texttt{sameNsubgrp}( , ): let \( t \) be a tuple representing a homomorphism\[^{2}\] \( \varphi : PB_4 \to S_d \) (for concreteness, we could use \text{listE}[19].PB4) and \( g \) be a random

\[^{2}\]Of course, the same tuple \( t \) also represents a homomorphism from \( F_6 \) to \( S_d \).
permutation in $S_d$; using the command $tt = \text{conj}_\text{tup}(g, t)$, we form another homomorphism $\varphi' : \text{PB}_4 \to S_d$ with the same kernel; the command $\text{sameNsubgrp}(t, tt)$ (as well as the command $\text{sameNsubgrp}(tt, t)$) returns True.

**Indirect testing of the generators** $\text{generW}F2$ and $\text{generW}Comm$

The function $\text{test}\_\text{generW}F2(\ )$ was used for testing the generator $\text{generW}F2$. For a positive integer $d$, the command $\text{test}\_\text{generW}F2(d)$ forms a tuple $t$ of two random permutations in $S_d$; then a computer uses $\text{generW}F2$ to form the complete list $Wlist$ of words which represent elements of the permutation group $G$ 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 $\text{test}\_\text{generW}Comm(\ )$ was used for testing the generator $\text{generW}Comm$. For a positive integer $d$, the command $\text{test}\_\text{generW}Comm(d)$ 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 $\text{generW}Comm$ to form the complete list $Wlist$ of words which represent elements of $H$. a computer checks that the length of $Wlist$ coincides with the order of $H$; finally, a computer checks that the set of permutations corresponding to words in $Wlist$ coincides with the set of elements of $H$.

**Indirect testing of** $\text{conjBySig}1$, $\text{conjBySig}2$ and $\text{conjBySig}3$

$\text{conjBySig}1(\ ), \text{conjBySig}2(\ )$ and $\text{conjBySig}3(\ )$ were tested (indirectly) using the function $\text{test}\_\text{conj}\_\text{braid}\_\text{rel}(\ )$. The input $t$ of $\text{test}\_\text{conj}\_\text{braid}\_\text{rel}(\ )$ is a tuple (of permutations) that represents a homomorphism from $\text{PB}_4$ to a symmetric group. The command $\text{test}\_\text{conj}\_\text{braid}\_\text{rel}(t)$ returns True if

- $\text{conjBySig}1(\text{conjBySig}2(\text{conjBySig}1(t)))$ coincides with $\text{conjBySig}2(\text{conjBySig}1(\text{conjBySig}2(t)))$ and
- $\text{conjBySig}2(\text{conjBySig}3(\text{conjBySig}2(t)))$ coincides with $\text{conjBySig}3(\text{conjBySig}2(\text{conjBySig}3(t)))$ and
- $\text{conjBySig}1(\text{conjBySig}3(t))$ coincides with $\text{conjBySig}3(\text{conjBySig}1(t))$.

Executing the command \{test\_conj\_braid\_rel(E.PB4) for E in listE\}, we get \{True\}.

**Indirect testing of** $\text{gener\_dessin\_pt}$, $\text{gener\_dessin}$, $\text{gener\_dessin\_slow}$, $\text{all\_dessin}$ and $\text{all\_dessin\_slow}$

The generator of child’s drawings $\text{gener\_dessin}$ and $\text{gener\_dessin\_slow}$ are based on different methods.

For an integer $d \geq 2$, the command $\text{test\_dessin}(d)$ 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 $\text{gener\_dessin}$ and the list of child’s drawings of degree $d$ (in the tuple format) using $\text{gener\_dessin\_slow}$. 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 [14].

\[\text{It is very likely that } G = S_d\]
Indirect testing of the composition of GT-shadows and the action on child’s drawings

Consider $N^{(21)}$. Since $N^{(21)}$ is isolated we tested our commands for composition we using the associativity property. For example, executing the commands

```python
for i in range(42):
    # code
```

In addition to the direct testing of the method `act` of the class `GTsh`, we also tested this method using the composition of GT-shadows.

Let $c_{15}$ be tuple that represents the child’s drawing $D_{15}$ of degree 15 stored in $dde_{15}E_{29}$. We know that $D_{15}$ is subordinate to $N^{(29)}$ represented by listE[29]. Moreover, $N^{(29)}$ is an isolated element of $NF_{PB_4}(B_4)$ and hence $GT^{\circ}(N^{(29)})$ form a group. The following piece of code was used to test the methods `act` and `compose`:

```python
Test=[ ]; i=0
GT29 = GTcharm[29]
for T in prod(GT29,GT29):
    D1=Dessin(T[0].compose(T[1]).act(c15))
    D2=Dessin(T[0].act(T[1].act(c15)))
    Test.append(D1==D2)
    i=i+1
    if i%500==0:
        print(i)

After running this piece of code, the command `set(Test)` returns `{True}`.

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 [3, Appendix A]. In particular, we denote by $\sigma_1,\ldots,\sigma_{n-1}$ the standard generators of $B_n$. We assume that the stands of geometric braid move up. The standard generators of $PB_n$ are given by the formula

$$x_{ij} := \sigma_{j-1} \cdots \sigma_{i+1} \sigma_i^2 \sigma_i^{1-1} \cdots \sigma_{j-1}^1, \quad 1 \leq i < j \leq n.$$  \hspace{1cm} (A.1)

It is known [8, Section 1.3] that any relation on the standard generators are consequences of these relations

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

Using [8, Theorem 1.16] and the above relations, one can see that, for every $n \geq 3$, $PB_n$ is isomorphic to the semi-direct product of $PB_{n-1}$ and the free group $F_{n-1}$ on $n-1$ generators. More precisely,

$$PB_n \cong K_n \ltimes U_n, \quad K_n \cong PB_{n-1} \hspace{1cm} (A.3)$$

$^7$GT$^{\circ}(N^{(29)})$ has 48 elements, so the shown loop has $48 \cdot 48 = 2304$ iterations.
where $U_n$ is freely generated by $x_{1,n}, \ldots, x_{n-1,n}$ and $K_n$ is generated by $x_{ij}$ with $1 \leq i < j \leq n - 1$.

For $n = 4$, we have
\[
x_{12} = \sigma_1^2, \quad x_{23} = \sigma_2^2, \quad x_{13} = \sigma_2 \sigma_1 \sigma_2^{-1} = \sigma_1^{-1} \sigma_2 \sigma_1, \quad x_{4} = \sigma_3 \sigma_2 \sigma_1^2 \sigma_2^{-1} \sigma_3^{-1} = \sigma_1^{-1} \sigma_2^{-1} \sigma_3 \sigma_2 \sigma_1 = \sigma_1^{-1} \sigma_2 \sigma_3 \sigma_2^{-1} \sigma_1.
\]

(A.4)

It is known \cite[Corollary 1.20]{8}, that the Abelianization $PB_n/[PB_n, PB_n]$ of $PB_n$ is freely generated by images of $\{x_{ij}\}_{1 \leq i < j \leq n}$. In other words, $PB_n/[PB_n, PB_n]$ is isomorphic to $\mathbb{Z}^{n(n-1)/2}$.

**Observation:** For any $1 \leq i \leq n - 2$, we have
\[
(\sigma_i \sigma_{i+1})^3 = (\sigma_{i+1} \sigma_i)^3.
\]

(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
\[
c := (\sigma_1 \sigma_2)^3 = (\sigma_2 \sigma_1)^3
\]
generates the center of $B_3$ (and the center of $PB_3$).

Indeed,
\[
\sigma_1 c = \sigma_1 \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 \sigma_1 = \sigma_1 \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 = c \sigma_1
\]
and
\[
\sigma_2 c = \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 = \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 = c \sigma_2.
\]

The group $PB_3$ is generated by
\[
x_{12} = \sigma_1^2, \quad x_{23} = \sigma_2^2, \quad x_{13} = \sigma_2 \sigma_1 \sigma_2^{-1} = \sigma_1^{-1} \sigma_2 \sigma_1.
\]

(A.6)

Here is how one can prove that $\sigma_2 \sigma_1 \sigma_2^{-1} = \sigma_1^{-1} \sigma_2 \sigma_1$:
\[
\sigma_2 \sigma_1 \sigma_2^{-1} = (\sigma_2 \sigma_1 \sigma_2^{-1}) \sigma_2 \sigma_1 \sigma_2^{-1} = (\sigma_1^{-1} \sigma_2 \sigma_1)(\sigma_1^{-1} \sigma_2 \sigma_1) = \sigma_1^{-1} \sigma_2 \sigma_1.
\]

**Proposition A.1** Let us prove that
\[
x_{23} x_{12} x_{13} = c = x_{12} x_{13} x_{23}
\]

(A.7)

**Proof.** Using (A.6), we get
\[
x_{23} x_{12} x_{13} = \sigma_2 \sigma_1 \sigma_2^{-1} \sigma_2 \sigma_1 \sigma_2^{-1} = \sigma_2 \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2^{-1} \sigma_2 \sigma_1 \sigma_2^{-1} = \sigma_2 \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2^{-1} = \sigma_2 \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 = \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 = (\sigma_2 \sigma_1)^3 = c.
\]

Similarly,
\[
x_{12} x_{13} x_{23} = \sigma_1 \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 = \sigma_1 \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 = \sigma_1 \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 = \sigma_1 \sigma_2 \sigma_1 \sigma_2 \sigma_1 \sigma_2 = (\sigma_1 \sigma_2)^3 = c.
\]

The proposition is proved.

The proof of the following proposition is straightforward:
Proposition A.2 We claim that

\begin{align}
x_{13} &= \sigma_2 x_{12} \sigma_2^{-1}, & x_{13} &= \sigma_1^{-1} x_{23} \sigma_1, & x_{13} &= x_{12}^{-1} x_{23}^{-1}, \quad (A.8) \\
\sigma_2^{-1} x_{12} \sigma_2 &= x_{23}^{-1} x_{12}^{-1}, & \sigma_1 x_{23} \sigma_1^{-1} &= x_{23}^{-1} x_{12}^{-1}, & \sigma_1 x_{13} \sigma_1^{-1} &= x_{23}. \quad (A.9)
\end{align}

\[ \] 

A.1 The generator of the center of \( B_4 \)

Recall [8, Theorem 1.24] that, for every \( n \geq 3 \), the center \( Z(B_n) \) of \( B_n \) coincides with the center \( Z(PB_n) \) of \( PB_n \). Moreover, \( Z(B_n) \) is an infinite cyclic group generated by \( \Delta_n \) where

\[
\Delta_n := (\sigma_1 \ldots \sigma_{n-1})(\sigma_1 \ldots \sigma_{n-2}) \ldots (\sigma_1, \sigma_2) \sigma_1
\]  

(A.10)

In particular, the center \( Z(B_4) = Z(PB_4) \) is generated by the element

\[
c_4 := \sigma_1 \sigma_2 \sigma_3 \sigma_1 \sigma_2 \sigma_1^2 \sigma_2 \sigma_3 \sigma_1 \sigma_2 \sigma_1.
\]  

(A.11)

Let us prove that

Proposition A.3 The generator \( c_4 \) of \( Z(PB_4) \) can be rewritten as

\[ c_4 = x_{14} x_{24} x_{34} x_{12} x_{13} x_{23}. \]  

(A.12)

**Proof.** Since \( \sigma_1 \sigma_3 = \sigma_3 \sigma_1 \), we have

\[
c_4 = \sigma_1 \sigma_2 \sigma_3 \sigma_1 \sigma_2 \sigma_1^2 \sigma_2 \sigma_1 \sigma_3 \sigma_2 \sigma_1.
\]  

(A.13)

The “blue” part of the expression in the right hand side is the generator \( c_3 \) of the center of \( B_3 \), i.e.

\[
\sigma_1 \sigma_2 \sigma_1^2 \sigma_2 \sigma_1 = x_{12} x_{13} x_{23}.
\]

Using this observation and the identity \( \sigma_3 x_{12} = x_{12} \sigma_3 \) we rewrite the expression \( \sigma_3 c_3 \sigma_3 \) as follows

\[
\sigma_3 c_3 \sigma_3 = \sigma_3 x_{12} x_{13} x_{23} \sigma_3 = x_{12} \sigma_3 x_{13} x_{23} \sigma_3 = x_{12} \sigma_3 (\sigma_2 \sigma_1^{-1} \sigma_2^{-1}) \sigma_2 \sigma_3
\]

\[
= x_{12} (\sigma_3 \sigma_2 \sigma_1^{-1} \sigma_2^{-1} \sigma_3) \sigma_2 \sigma_3 = x_{12} x_{14} \sigma_3 \sigma_2 \sigma_3
\]

\[
= x_{12} x_{14} \sigma_3 \sigma_2 \sigma_3^{-1} \sigma_3 = x_{12} x_{14} x_{24} x_{34}.
\]

Thus

\[
c_4 = \sigma_1 \sigma_2 x_{12} x_{14} x_{24} x_{34} \sigma_2 \sigma_1.
\]  

(A.14)

Using (A.14), we rewrite \( c_4 \) as follows:

\[
c_4 = \sigma_1 \sigma_2 x_{12} x_{14} x_{24} x_{34} \sigma_2 \sigma_1 = \sigma_1 \sigma_2 x_{12} x_{14} x_{24} x_{34} \sigma_2 \sigma_1^{-1} \sigma_1^{-1} \sigma_1 \sigma_2 \sigma_1
\]

To simplify the “blue” part of the last equation we observe that

\[
\sigma_1 \sigma_2 x_{12} \sigma_2^{-1} \sigma_1^{-1} = x_{23},
\]

\[
\sigma_1 \sigma_2 x_{14} \sigma_2^{-1} \sigma_1^{-1} = x_{24},
\]

\[
\sigma_1 \sigma_2 x_{24} \sigma_2^{-1} \sigma_1^{-1} = x_{34}.
\]
\[ \sigma_1 \sigma_2 x_{34} \sigma_2^{-1} \sigma_1^{-1} = x_{34}^{-1} x_{24}^{-1} x_{14} x_{24} x_{34}. \]

Combining the last 4 identities with

\[ \sigma_1 \sigma_2^2 \sigma_1 = \sigma_1^2 \sigma_1^{-1} \sigma_2 \sigma_1 = x_{12} x_{13} \]

we deduce that

\[ c_4 = x_{23} x_{14} x_{24} x_{34} x_{12} x_{13}. \]  \hspace{1cm} (A.15)

Finally

\[ x_{14} x_{24} x_{34} x_{12} x_{13} x_{23} = x_{23}^{-1} x_{23} x_{14} x_{24} x_{34} x_{12} x_{13} x_{23} = x_{23}^{-1} c_4 x_{23} = c_4. \]

The proposition is proved. \[ \square \]

\section*{B Justification of the code for hexa1( , ) and hexa2( , )}

Using the braid relation \( \sigma_1 \sigma_2 \sigma_1 = \sigma_2 \sigma_1 \sigma_2 \) and the definitions

\[ x_{12} := \sigma_1^2, \quad x_{23} := \sigma_2^2, \quad c := (\sigma_1 \sigma_2 \sigma_1)^2, \]

it is easy to deduce the following identities

\[ \begin{align*}
\sigma_1 x_{12} \sigma_1^{-1} &= x_{12}, \\
\sigma_2 x_{12} \sigma_2^{-1} &= uc,
\end{align*} \quad \begin{align*}
\sigma_1 x_{23} \sigma_1^{-1} &= zc, \\
\sigma_2 x_{23} \sigma_2^{-1} &= x_{23},
\end{align*} \]

\[ \begin{align*}
\sigma_1^{-1} x_{12} \sigma_1 &= x_{12}, \\
\sigma_2^{-1} x_{12} \sigma_2 &= zc,
\end{align*} \quad \begin{align*}
\sigma_1^{-1} x_{23} \sigma_1 &= uc, \\
\sigma_2^{-1} x_{23} \sigma_2 &= x_{23},
\end{align*} \]

\[ \begin{align*}
\sigma_1 \sigma_2 x_{12} \sigma_2^{-1} \sigma_1^{-1} &= x_{23}, \\
\sigma_2 \sigma_1 x_{12} \sigma_1^{-1} \sigma_2^{-1} &= uc,
\end{align*} \quad \begin{align*}
\sigma_1 \sigma_2 x_{23} \sigma_2^{-1} \sigma_1^{-1} &= zc, \\
\sigma_2 \sigma_1 x_{23} \sigma_1^{-1} \sigma_2^{-1} &= x_{12},
\end{align*} \]

(B.1)

where

\[ z := x_{23}^{-1} x_{12}^{-1}, \quad u := x_{12}^{-1} x_{23}^{-1}. \]

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

\begin{proposition}
Let \( K \in NF_{PB_3} (B_3) \) and \((m, f) \in \mathbb{Z} \times F_2\). Then the first hexagon relation

\[ \sigma_1 x_{12}^m f^{-1} \sigma_2 x_{23}^m f K = f^{-1} \sigma_1 \sigma_2 (x_{13} x_{23})^m K \]  \hspace{1cm} (B.2)

is equivalent to

\[ x_{23}^m f x_{12}^m f(x_{12}, z)^{-1} z^m f(x_{23}, z) \in K, \]  \hspace{1cm} (B.3)

and the second hexagon relation

\[ f^{-1} \sigma_2 x_{23}^m f \sigma_1 x_{12}^m K = \sigma_2 \sigma_1 (x_{12} x_{13})^m f K, \]  \hspace{1cm} (B.4)

is equivalent to

\[ f(u, x)^{-1} x_{12}^m f^{-1} x_{23}^m f(u, y) u^m \in K. \]  \hspace{1cm} (B.5)
\end{proposition}
Proof. First, it is easy to see that
\[ c = x_{12}x_{13}x_{23} = x_{23}x_{12}x_{13} . \]
Hence \( x_{13}x_{23} = x_{12}^{-1}c \) and \( x_{12}x_{13} = x_{23}^{-1}c \). Therefore, (B.2) and (B.4) are equivalent to
\[ \sigma_1 x_{12}^m f^{-1} \sigma_2 x_{23}^m f K = f^{-1} \sigma_1 \sigma_2 x_{12}^m c^m K \quad (B.6) \]
and
\[ f^{-1} \sigma_2 x_{23}^m f \sigma_1 x_{12}^m K = \sigma_2 \sigma_1 x_{23}^m c^m f K, \quad (B.7) \]
respectively.

Using (B.1), we rewrite the left hand side of (B.6) as follows
\[ \sigma_1 x_{12}^m f^{-1} \sigma_2 x_{23}^m f K = x_{12}^m \sigma_1 f^{-1} \sigma_1^{-1} \sigma_2 x_{23}^m f K = \]
\[ x_{12}^m \sigma_1 f^{-1} \sigma_1^{-1} \sigma_1 \sigma_2 x_{23}^m f \sigma_2^{-1} \sigma_1 \sigma_2 K = x_{12}^m f^{-1}(x_{12}, zc) z^m f(x_{23}, zc) \sigma_1 \sigma_2 c^m K. \quad (B.8) \]
Similarly, using (B.1), we rewrite the right hand side of (B.6) as follows
\[ f^{-1} \sigma_1 \sigma_2 x_{12}^m c^m K = f^{-1} x_{23}^m \sigma_1 \sigma_2 c^m K. \quad (B.9) \]
Combining (B.8) with (B.9), we conclude that (B.2) is equivalent to
\[ x_{23}^m f x_{12}^m f^{-1}(x_{12}, zc) z^m f(x_{23}, zc) \in K. \quad (B.10) \]
Similarly, applying (B.1) to (B.7), we get
\[ f^{-1} \sigma_2 x_{23}^m f \sigma_1 x_{12}^m = f^{-1} x_{23}^m f(u, x_{23}) u^m \sigma_2 \sigma_1 c^m \]
and
\[ \sigma_2 \sigma_1 x_{23}^m c^m f = x_{12}^m f(u, x_{12}) \sigma_2 \sigma_1 c^m . \]
Thus (B.7) is equivalent to
\[ f(u, x_{12})^{-1} x_{12}^m f^{-1} x_{23}^m f(u, x_{23}) u^m \in K. \quad (B.11) \]

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 \( c \in Z(PB_3) \), 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 \( f(u, x_{12})^{-1} x_{12}^m f^{-1} x_{23}^m f(u, x_{23}) \) as follows
\[ f(u, x_{12})^{-1} x_{12}^m f^{-1} x_{23}^m f(u, x_{23}) =
\]
\[ c^{-q_1} f(u, x_{12})^{-1} x_{12}^m f^{-1} x_{23}^m f(u, x_{23}) c^{q_1} = f(u, x_{12})^{-1} x_{12}^m f^{-1} x_{23}^m f(u, x_{23}) . \]
Thus (B.10) (resp. (B.11)) is equivalent to (B.3) (resp. to (B.5)).
References


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