UGALY

Universal Groups Acting LocallY

v2.0

1 June 2020

Khalil Hannouch

Stephan Tornier

Khalil Hannouch
Email: khalil.hannouch@newcastle.edu.au
Homepage: https://www.newcastle.edu.au/profile/khalil-hannouch

Stephan Tornier
Email: stephan.tornier@newcastle.edu.au
Homepage: https://www.newcastle.edu.au/profile/stephan-tornier

Abstract

UGALY (Universal Groups Acting LocallY) is a GAP package that provides methods to create, analyse and find local actions of universal groups acting on locally finite regular trees, following Burger-Mozes and Tornier.

Copyright

Acknowledgements

The second author owes thanks to Marc Burger and George Willis for their support and acknowledges contributions from the SNSF Doc.Mobility fellowship 172120 and the ARC Discovery Project DP120100996 to the development of an early version of this codebase. In its present form, the development of UGALY was made possible by the ARC Laureate Fellowship FL170100032 and the ARC DECRA Fellowship DE210100180.

Contents

Chapter 1

Introduction

Let Ω be a set of cardinality $d \in \mathbb{N}_{\geq 3}$ and let $T_d = (V, E)$ be the d-regular tree. We follow Serre's graph theory notation [Ser80]. Given a subgroup H of the automorphism group $\mathrm{Aut}(T_d)$ of T_d, and a vertex $x \in V$, the stabilizer H_x of x in H induces a permutation group on the set $E(x) := \{e \in E \mid o(e) = x\}$ of edges issuing from x. We say that H is locally "P" if for every $x \in V$ said permutation group satisfies the property "P", e.g. being transitive, semiprimitive, quasiprimitive or 2-transitive. In [BM00], Burger-Mozes develop a remarkable structure theory of closed, non-discrete, locally quasiprimitive subgroups of $\mathrm{Aut}(T_d)$, which resembles the theory of semisimple Lie groups. They complement this structure theory with a particularly accessible class of subgroups of $\mathrm{Aut}(T_d)$ with prescribed local action: Given $F \leq \mathrm{Sym}(\Omega)$ their universal group $\mathrm{U}(F)$ is closed in $\mathrm{Aut}(T_d)$, vertex-transitive, compactly generated and locally permutation isomorphic to F. It is discrete if and only if F is semiregular. When F is transitive, $\mathrm{U}(F)$ is maximal up to conjugation among vertex-transitive subgroups of $\mathrm{Aut}(T_d)$ that are locally permutation isomorphic to F, hence *universal*.

This construction was generalized by the second author in [Tor20]: In the spirit of k-closures of groups acting on trees developed in [BEW15], we generalize the universal group construction by prescribing the local action on balls of a given radius $k \in \mathbb{N}$, the Burger-Mozes construction corresponding to the case $k = 1$. Fix a tree $B_{d,k}$ which is isomorphic to a ball of radius k in the labelled tree T_d and let $l_x^k : B(x,k) \to B_{d,k}$ $(x \in V)$ be the unique label-respecting isomorphism. Then

$$\sigma_k : \mathrm{Aut}(T_d) \times V \to \mathrm{Aut}(B_{d,k}),\ (g,x) \to l_{gx}^k \circ g \circ (l_x^k)^{-1}$$

captures the *k-local action* of g at the vertex $x \in V$.

With this we can make the following defintition: Let $F \leq \mathrm{Aut}(B_{d,k})$. Define

$$\mathrm{U}_k(F) := \{g \in \mathrm{Aut}(T_d) \mid \forall x \in V :\ \sigma_k(g,x) \in F\}.$$

While $\mathrm{U}_k(F)$ is always closed, vertex-transitive and compactly generated, other properties of $\mathrm{U}(F)$ do *not* carry over. Foremost, the group $\mathrm{U}_k(F)$ need not be locally action isomorphic to F and we say that $F \leq \mathrm{Aut}(B_{d,k})$ satisfies condition (C) if it is. This can be viewed as an interchangeability condition on neighbouring local actions, see Section 3.1. There is also a discreteness condition (D) on $F \leq \mathrm{Aut}(B_{d,k})$ in terms of certain stabilizers in F under which $\mathrm{U}_k(F)$ is discrete, see Section 5.1.

UGALY provides methods to create, analyse and find local actions $F \leq \mathrm{Aut}(B_{d,k})$ that satisfy condition (C) and/or (D), including the constructions Γ, Δ, Φ, Σ, and Π developed in [Tor20]. This package was developed within the Zero-Dimensional Symmetry Research Group in the School of Mathematical and Physical Sciences at The University of Newcastle as part of a project course taken by the first author, supervised by the second author.

1.1 Purpose

UGALY serves both a research and an educational purpose. It consolidates a rudimentary codebase that was developed by the second author in the course of research undertaken towards the article [Tor20]. This codebase had been tremendously beneficial in achieving the results of [Tor20] in the first place, and so there has always been a desire to make it available to a wider audience.

From a research perspective, UGALY introduces computational methods to the world of locally compact groups. Due to the Cayley-Abels graph construction [KM08], groups acting on trees form a particularly significant class of totally disconnected locally compact groups. Burger-Mozes universal groups [BM00] and their generalisations $\mathrm{U}_k(F)$, where $F \leq \mathrm{Aut}(B_{d,k})$ satisfies the compatibility condition (C), are among the most accessible of these groups and form a significant subclass: indeed, due to [Tor20, Corollary 4.32] the locally transitive, generalised universal groups are exactly the closed, locally transitive subgroups of $\mathrm{Aut}(T_d)$ that contain an inversion of order 2 and satisfy one of the independence properties (P_k) (see [BEW15]) that generalise Tits' independence property (P), see [Tit70]. UGALY provides the means to generate a library of all these groups in terms of the k-local action representing them. This library naturally encompasses the library of finite transitive permutation groups TransGrp in the case $k = 1$ of classical Burger-Mozes groups. For example, in the case $d = 3$ we obtain the following for $k = 1$ and $k = 2$:

Example

```
gap> F1:=LocalAction(3,1,TransitiveGroup(3,1));
A3
gap> F2:=LocalAction(3,1,TransitiveGroup(3,2));
S3
gap> ConjugacyClassRepsCompatibleGroupsWithProjection(2,F1);
[ Group([ (1,4,5)(2,3,6) ]) ]
gap> ConjugacyClassRepsCompatibleGroupsWithProjection(2,F2);
[ Group([ (1,2)(3,5)(4,6), (1,4,5)(2,3,6) ]),
  Group([ (1,2)(3,4)(5,6), (1,2)(3,5)(4,6), (1,4,5)(2,3,6) ]),
  Group([ (3,4)(5,6), (1,2)(3,4), (1,4,5)(2,3,6), (3,5,4,6) ]),
  Group([ (3,4)(5,6), (1,2)(3,4), (1,4,5)(2,3,6), (3,5)(4,6) ]),
  Group([ (3,4)(5,6), (1,2)(3,4), (1,4,5)(2,3,6), (5,6), (3,5,4,6) ]) ]
```

We envision to add such a library in a future version and hope that it will be useful to all researchers in the area. On the educational side...

Chapter 2

Preliminaries

We recall the following notation from the Introduction which is essential throughout this manual, cf. [Tor20]. Let Ω be a set of cardinality $d \in \mathbb{N}_{\geq 3}$ and let $T_d = (V,E)$ denote the d-regular tree, following the graph theory notation in [Ser80]. A *labelling* l of T_d is a map $l : E \to \Omega$ such that for every $x \in V$ the restriction $l_x : E(x) \to \Omega,\ e \mapsto l(e)$ is a bijection, and $l(e) = l(\overline{e})$ for all $e \in E$. For every $k \in \mathbb{N}$, fix a tree $B_{d,k}$ which is isomorphic to a ball of radius k around a vertex in T_d and carry over the labelling of T_d to $B_{d,k}$ via the chosen isomorphism. We denote the center of $B_{d,k}$ by b.

For every $x \in V$ there is a unique, label-respecting isomorphism $l_x^k : B(x,k) \to B_{d,k}$. We define the *k-local action* $\sigma_k(g,x) \in \mathrm{Aut}(B_{d,k})$ of an automorphism $g \in \mathrm{Aut}(T_d)$ at a vertex $x \in V$ via the map

$$\sigma_k : \mathrm{Aut}(T_d) \times V \to \mathrm{Aut}(B_{d,k}), \sigma_k(g,x) \mapsto \sigma_k(g,x) := l_{gx}^k \circ g \circ (l_x^k)^{-1}.$$

2.1 Local actions

In this package, local actions $F \leq \mathrm{Aut}(B_{d,k})$ are handled as objects of the category `IsLocalAction` (**??**) and have several attributes and properties introduced throughout this manual. Most importantly, a local action always stores the degree d and the radius k of the ball $B_{d,k}$ that it acts on.

2.1.1 IsLocalAction (for IsPermGroup)

▷ `IsLocalAction(`*arg*`)` (filter)

Returns: `true` or `false`

Local actions $F \leq \mathrm{Aut}(B_{d,k})$ are stored together with their degree (see `LocalActionDegree` (**??**)), radius (see `LocalActionRadius` (**??**)) as well as other attributes and properties in this category.

Example
```
gap> G:=WreathProduct(SymmetricGroup(2),SymmetricGroup(3));
Group([ (1,2), (3,4), (5,6), (1,3,5)(2,4,6), (1,3)(2,4) ])
gap> IsLocalAction(G);
false
gap> H:=AutBall(3,2);
Group([ (1,2), (3,4), (5,6), (1,3,5)(2,4,6), (1,3)(2,4) ])
gap> IsLocalAction(H);
true
gap> K:=LocalAction(3,2,G);
Group([ (1,2), (3,4), (5,6), (1,3,5)(2,4,6), (1,3)(2,4) ])
```

```
gap> IsLocalAction(K);
true
```

2.1.2 LocalAction (for IsInt, IsInt, IsPermGroup)

▷ LocalAction(*d*, *k*, *F*) (operation)

Returns: the regular rooted tree group G as an object of the category IsLocalAction (**??**), checking that F is indeed a subgroup of $\mathrm{Aut}(B_{d,k})$.

The arguments of this method are a degree $d \in \mathbb{N}_{\geq 3}$, a radius $k \in \mathbb{N}_0$ and a group $F \leq \mathrm{Aut}(B_{d,k})$.

Example

```
gap> G:=WreathProduct(SymmetricGroup(2),SymmetricGroup(3));
Group([ (1,2), (3,4), (5,6), (1,3,5)(2,4,6), (1,3)(2,4) ])
gap> IsLocalAction(G);
false
gap> G:=LocalAction(3,2,G);
Group([ (1,2), (3,4), (5,6), (1,3,5)(2,4,6), (1,3)(2,4) ])
gap> IsLocalAction(G);
```

2.1.3 LocalActionNC (for IsInt, IsInt, IsPermGroup)

▷ LocalActionNC(*d*, *k*, *F*) (operation)

Returns: the regular rooted tree group G as an object of the category IsLocalAction (**??**), without checking that F is indeed a subgroup of $\mathrm{Aut}(B_{d,k})$.

The arguments of this method are a degree $d \in \mathbb{N}_{\geq 3}$, a radius $k \in \mathbb{N}_0$ and a group $F \leq \mathrm{Aut}(B_{d,k})$.

2.1.4 LocalActionDegree (for IsLocalAction)

▷ LocalActionDegree(*F*) (attribute)

Returns: the degree d of the ball $B_{d,k}$ that F is acting on.

The argument of this attribute is a local action $F \leq \mathrm{Aut}(B_{d,k})$ (see IsLocalAction (**??**)).

Example

```
gap> F:=PHI(4,AlternatingGroup(4));
Group([ (1,5,7)(2,4,8)(3,6,9)(10,11,12), (1,2,3)(4,7,10)(5,9,11)(6,8,12),
  (1,2,3), (4,5,6), (7,8,9), (10,11,12) ])
gap> LocalActionDegree(F);
4
```

2.1.5 LocalActionRadius (for IsLocalAction)

▷ LocalActionRadius(*F*) (attribute)

Returns: the radius k of the ball $B_{d,k}$ that F is acting on.

The argument of this attribute is a local action $F \leq \mathrm{Aut}(B_{d,k})$ (see IsLocalAction (**??**)).

Example

```
gap> F:=PHI(4,AlternatingGroup(4));
Group([ (1,5,7)(2,4,8)(3,6,9)(10,11,12), (1,2,3)(4,7,10)(5,9,11)(6,8,12),
  (1,2,3), (4,5,6), (7,8,9), (10,11,12) ])
gap> LocalActionRadius(F);
2
```

2.1.6 LocalAction (for r, d, k, aut, addr)

▷ `LocalAction(r, d, k, aut, addr)` (operation)

Returns: the `r`-local action σ_r(`aut`,`addr`) of the automorphism `aut` of $B_{d,k}$ at the vertex represented by the address `addr`.

The arguments of this method are a radius `r`, a degree `d` $\in \mathbb{N}_{\geq 3}$, a radius `k` $\in \mathbb{N}$, an automorphism `aut` of $B_{d,k}$, and an address `addr`.

Example

```
gap> a:=(1,3,5)(2,4,6);; a in AutBall(3,2);
true
gap> LocalAction(2,3,2,a,[]);
(1,3,5)(2,4,6)
gap> LocalAction(1,3,2,a,[]);
(1,2,3)
gap> LocalAction(1,3,2,a,[1]);
(1,2)
```

Example

```
gap> b:=Random(AutBall(3,4));
(1,20,4,17,2,19,3,18)(5,22,8,23,6,21,7,24)(9,10)(13,16,14,15)
gap> LocalAction(2,3,4,b,[3,1]);
(1,4)(2,3)
gap> LocalAction(3,3,4,b,[3,1]);
Error, the sum of input argument r=3 and the length of input argument
addr=[ 3, 1 ] must not exceed input argument k=4
```

2.1.7 Projection (for F, r)

▷ `Projection(F, r)` (operation)

Returns: the restriction of the projection map $\mathrm{Aut}(B_{d,k}) \to \mathrm{Aut}(B_{d,r})$ to `F`.

The arguments of this method are a local action `F` $\leq \mathrm{Aut}(B_{d,k})$, and a projection radius `r` $\leq$ `k`.

Example

```
gap> F:=GAMMA(4,3,SymmetricGroup(3));
Group([ (1,16,19)(2,15,20)(3,13,18)(4,14,17)(5,10,23)(6,9,24)(7,12,22)
  (8,11,21), (1,9)(2,10)(3,12)(4,11)(5,15)(6,16)(7,13)(8,14)(17,21)(18,22)
  (19,24)(20,23) ])
gap> pr:=Projection(F,2);
<action homomorphism>
gap> a:=Random(F);; Image(pr,a);
(1,4,5)(2,3,6)
```

2.1.8 ImageOfProjection

▷ `ImageOfProjection(F, r)` (function)

Returns: the local action $\sigma_r(F,b) \leq \mathrm{Aut}(B_{d,r})$.

The arguments of this method are a local action `F` $\leq \mathrm{Aut}(B_{d,k})$, and a projection radius `r` $\leq$ `k`. This method uses `LocalAction` (2.1.6) on generators rather than `Projection` (2.1.7) on the group to compute the image.

Example
```
gap> AutBall(3,2);
Group([ (1,2), (3,4), (5,6), (1,3,5)(2,4,6), (1,3)(2,4) ])
gap> ImageOfProjection(AutBall(3,2),1);
Group([ (), (), (), (1,2,3), (1,2) ])
```

2.2 Finite balls

The automorphism groups of the finite labelled balls $B_{d,k}$ lie at the center of this package. The method `AutBall` (2.2.1) produces these automorphism groups as iterated wreath products. The result is a permutation group on the set of leaves of $B_{d,k}$.

2.2.1 AutBall

▷ `AutBall(d, k)` (function)

Returns: the local action $\mathrm{Aut}(B_{d,k})$ as a permutation group of the $d \cdot (d-1)^{k-1}$ leaves of $B_{d,k}$.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\geq 3}$ and a radius `k` $\in \mathbb{N}_0$.

Example
```
gap> G:=AutBall(3,2);
Group([ (1,2), (3,4), (5,6), (1,3,5)(2,4,6), (1,3)(2,4) ])
gap> Size(G);
48
```

2.3 Addresses and leaves

The vertices at distance n from the center b of $B_{d,k}$ are addressed as elements of the set

$$\Omega^{(n)} := \{(\omega_1,\dots,\omega_n) \in \Omega^n \mid \forall l \in \{1,\dots,n-1\} : \ \omega_l \neq \omega_{l+1}\},$$

i.e. as lists of length n of elements from `[1..d]` such that no two consecutive entries are equal. They are ordered according to the lexicographic order on $\Omega^{(n)}$. The center b itself is addressed by the empty list `[]`. Note that the leaves of $B_{d,k}$ correspond to elements of $\Omega^{(k)}$.

2.3.1 BallAddresses

▷ `BallAddresses(d, k)` (function)

Returns: a list of all addresses of vertices in $B_{d,k}$ in ascending order with respect to length, lexicographically ordered within each level. See `AddressOfLeaf` (2.3.3) and `LeafOfAddress` (2.3.4) for the correspondence between the leaves of $B_{d,k}$ and addresses of length `k`.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\geq 3}$ and a radius `k` $\in \mathbb{N}_0$.

Example
```
gap> BallAddresses(3,1);
[ [  ], [ 1 ], [ 2 ], [ 3 ] ]
gap> BallAddresses(3,2);
[ [  ], [ 1 ], [ 2 ], [ 3 ], [ 1, 2 ], [ 1, 3 ], [ 2, 1 ], [ 2, 3 ],
[ 3, 1 ], [ 3, 2 ] ]
```

2.3.2 LeafAddresses

▷ `LeafAddresses(d, k)` (function)

Returns: a list of addresses of the leaves of $B_{d,k}$ in lexicographic order.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\geq 3}$ and a radius `k` $\in \mathbb{N}_0$.

Example

```
gap> LeafAddresses(3,2);
[ [ 1, 2 ], [ 1, 3 ], [ 2, 1 ], [ 2, 3 ], [ 3, 1 ], [ 3, 2 ] ]
```

2.3.3 AddressOfLeaf

▷ `AddressOfLeaf(d, k, lf)` (function)

Returns: the address of the leaf `lf` of $B_{d,k}$ with respect to the lexicographic order.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\geq 3}$, a radius `k` $\in \mathbb{N}$, and a leaf `lf` of $B_{d,k}$.

Example

```
gap> AddressOfLeaf(3,2,1);
[ 1, 2 ]
gap> AddressOfLeaf(3,3,1);
[ 1, 2, 1 ]
```

2.3.4 LeafOfAddress

▷ `LeafOfAddress(d, k, addr)` (function)

Returns: the smallest leaf (integer) whose address has `addr` as a prefix.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\geq 3}$, a radius `k` $\in \mathbb{N}$, and an address `addr`.

Example

```
gap> LeafOfAddress(3,2,[1,2]);
1
gap> LeafOfAddress(3,2,[3]);
5
gap> LeafOfAddress(3,2,[]);
1
```

2.3.5 ImageAddress

▷ `ImageAddress(d, k, aut, addr)` (function)

Returns: the address of the image of the vertex represented by the address `addr` under the automorphism `aut` of $B_{d,k}$.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\geq 3}$, a radius `k` $\in \mathbb{N}$, an automorphism `aut` of $B_{d,k}$, and an address `addr`.

Example

```
gap> ImageAddress(3,2,(1,2),[1,2]);
[ 1, 3 ]
gap> ImageAddress(3,2,(1,2),[1]);
[ 1 ]
```

2.3.6 ComposeAddresses

▷ `ComposeAddresses(addr1, addr2)` (function)

Returns: the concatenation of the addresses *addr1* and *addr2* with reduction as per [Tor20, Section 3.2].

The arguments of this method are two addresses *addr1* and *addr2*.

Example
```
gap> ComposeAddresses([1,3],[2,1]);
[ 1, 3, 2, 1 ]
gap> ComposeAddresses([1,3,2],[2,1]);
[ 1, 3, 1 ]
```

Chapter 3

Compatibility

3.1 The compatibility condition (C)

A subgroup $F \leq \mathrm{Aut}(B_{d,k})$ satifies the compatibility condition (C) if and only if if $\mathrm{U}_k(F)$ is locally action isomorphic to F, see [Tor20, Proposition 3.8]. The term *compatibility* comes from the following translation of this condition into properties of the $(k-1)$-local actions of elements of F: The group F satisfies (C) if and only if

$$\forall \alpha \in F\ \forall \omega \in \Omega\ \exists \beta \in F: \ \sigma_{k-1}(\alpha,b) = \sigma_{k-1}(\beta,b_\omega),\ \sigma_{k-1}(\alpha,b_\omega) = \sigma_{k-1}(\beta,b).$$

3.2 Compatible elements

This section is concerned with testing compatibility of two given elements (see `AreCompatibleBallElements` (3.2.1)) and finding an/all elements that is/are compatible with a given one (see `CompatibleBallElement` (3.2.2), `CompatibilitySet` (3.2.3)).

3.2.1 AreCompatibleBallElements

▷ `AreCompatibleBallElements(d, k, aut1, aut2, dir)` (function)

Returns: `true` if *aut1* and *aut2* are compatible with each other in direction *dir*, and `false` otherwise.

The arguments of this method are a degree $d \in \mathbb{N}_{\geq 3}$, a radius $k \in \mathbb{N}$, two automorphisms *aut1*, *aut2* $\in \mathrm{Aut}(B_{d,k})$, and a direction *dir* $\in$ `[1..d]`.

Example
```
gap> AreCompatibleBallElements(3,1,(1,2),(1,2,3),1);
true
gap> AreCompatibleBallElements(3,1,(1,2),(1,2,3),2);
false
```

Example
```
gap> a:=(1,3,5)(2,4,6);; a in AutBall(3,2);
true
gap> LocalAction(1,3,2,a,[]); LocalAction(1,3,2,a,[1]);
(1,2,3)
```

```
(1,2)
gap> b:=(1,4)(2,3);; b in AutBall(3,2);
true
gap> LocalAction(1,3,2,b,[]); LocalAction(1,3,2,b,[1]);
(1,2)
(1,2,3)

gap> AreCompatibleBallElements(3,2,a,b,1);
true
gap> AreCompatibleBallElements(3,2,a,b,3);
false
```

3.2.2 CompatibleBallElement

▷ `CompatibleBallElement(F, aut, dir)` (function)

Returns: an element of F that is compatible with `aut` in direction `dir` if one exists, and `fail` otherwise.

The arguments of this method are a local action $F \leq \mathrm{Aut}(B_{d,k})$, an element `aut` $\in F$, and a direction `dir` $\in$ `[1..d]`.

Example
```
gap> a:=Random(AutBall(5,1)); dir:=Random([1..5]);
(1,3,2,5)
4
gap> CompatibleBallElement(AutBall(5,1),a,dir);
(1,3,2,5)
```

Example
```
gap> a:=(1,3,5)(2,4,6);; a in AutBall(3,2);
true
gap> CompatibleBallElement(AutBall(3,2),a,1);
(1,4,2,3)
```

3.2.3 CompatibilitySet

▷ `CompatibilitySet(F, aut, dir)` (operation)
▷ `CompatibilitySet(F, aut, dirs)` (operation)

for the arguments F, `aut`, `dir`

Returns: the list of elements of F that are compatible with `aut` in direction `dir`.

The arguments of this method are a local action F of $\leq \mathrm{Aut}(B_{d,k})$, an automorphism `aut` $\in F$, and a direction `dir` $\in$ `[1..d]`.

for the arguments F, `aut`, `dirs`

Returns: the list of elements of F that are compatible with `aut` in all directions of `dirs`.

The arguments of this method are a local action F of $\leq \mathrm{Aut}(B_{d,k})$, an automorphism `aut` $\in F$, and a sublist of directions `dirs` $\subseteq$ `[1..d]`.

Example

```
gap> F:=LocalAction(4,1,TransitiveGroup(4,3));
D(4)
gap> G:=LocalAction(4,1,SymmetricGroup(4));
Sym( [ 1 .. 4 ] )
gap> CompatibilitySet(G,aut,1);
RightCoset(Sym( [ 2 .. 4 ] ),(1,3))
gap> CompatibilitySet(F,aut,1);
RightCoset(Group([ (2,4) ]),(1,3))
gap> CompatibilitySet(F,aut,[1,3]);
RightCoset(Group([ (2,4) ]),(1,3))
gap> CompatibilitySet(F,aut,[1,2]);
RightCoset(Group(()),(1,3))
```

3.2.4 AssembleAutomorphism

▷ AssembleAutomorphism(*d*, *k*, *auts*) (function)

Returns: the automorphism $(\mathtt{aut},(\mathtt{auts}[\mathrm{i}])_{i=1}^{d})$ of $B_{d,k+1}$, where aut is implicit in $(\mathtt{auts}[\mathrm{i}])_{i=1}^{d}$.

The arguments of this method are a degree $d \in \mathbb{N}_{\geq 3}$, a radius $k \in \mathbb{N}$, and a list *auts* of d automorphisms $(\mathtt{auts}[\mathrm{i}])_{i=1}^{d}$ of $B_{d,k}$ which comes from an element $(\mathtt{aut},(\mathtt{auts}[\mathrm{i}])_{i=1}^{d})$ of $\mathrm{Aut}(B_{d,k+1})$.

Example

```
gap> aut:=Random(AutBall(3,2));
(1,2)(3,6)(4,5)
gap> auts:=[];;
gap> for i in [1..3] do auts[i]:=CompatibleBallElement(3,2,AutBall(3,2),aut,i); od;
gap> auts;
[ (1,2)(3,5)(4,6), (1,3,5)(2,4,6), (1,5,3)(2,6,4) ]
gap> a:=AssembleAutomorphism(3,2,auts);
(1,3)(2,4)(5,11)(6,12)(7,9)(8,10)
gap> a in AutBall(3,3);
true
gap> LocalAction(2,3,3,a,[]);
(1,2)(3,6)(4,5)
```

3.3 Compatible subgroups

Using the methods of Section 3.2, this section provides methods to test groups for the compatibility condition and search for compatible subgroups inside a given group, e.g. $\mathrm{Aut}(B_{d,k})$, or with a certain image under some projection.

3.3.1 MaximalCompatibleSubgroup (for IsLocalAction)

▷ MaximalCompatibleSubgroup(*F*) (attribute)

Returns: The local action $C(F) \leq \mathrm{Aut}(B_{d,k})$, which is the maximal compatible subgroup of F.

The argument of this attribute is a local action $F \leq \mathrm{Aut}(B_{d,k})$ (see IsLocalAction (**??**)).

Example

```
gap> F:=LocalAction(3,1,Group((1,2)));
Group([ (1,2) ])
gap> MaximalCompatibleSubgroup(F);
```

```
Group([ (1,2) ])
gap> G:=LocalAction(3,2,Group((1,2)));
Group([ (1,2) ])
gap> MaximalCompatibleSubgroup(G);
Group(())
```

3.3.2 SatisfiesC (for IsLocalAction)

▷ `SatisfiesC(`F`)` (property)

Returns: `true` if F satisfies the compatibility condition (C), and `false` otherwise.

The argument of this property is a local action $F \leq \mathrm{Aut}(B_{d,k})$ (see `IsLocalAction` (**??**)).

Example

```
gap> D:=DELTA(3,SymmetricGroup(3));
Group([ (1,3,6)(2,4,5), (1,3)(2,4), (1,2)(3,4)(5,6) ])
gap> SatisfiesC(D);
true
```

3.3.3 CompatibleSubgroups

▷ `CompatibleSubgroups(`F`)` (function)

Returns: the list of all compatible subgroups of F.

The argument of this method is a local action $F \leq \mathrm{Aut}(B_{d,k})$. This method calls `AllSubgroups` on F and is therefore slow. Use for instructional purposes on small examples only, and use `ConjugacyClassRepsCompatibleSubgroups` (**??**) or `ConjugacyClassRepsCompatibleSubgroupsWithProjection` (**??**) for computations.

Example

```
gap> G:=GAMMA(3,SymmetricGroup(3));
Group([ (1,4,5)(2,3,6), (1,3)(2,4)(5,6) ])
gap> list:=CompatibleSubgroups(G);
[ Group(()), Group([ (1,2)(3,5)(4,6) ]), Group([ (1,3)(2,4)(5,6) ]),
  Group([ (1,6)(2,5)(3,4) ]), Group([ (1,4,5)(2,3,6) ]), Group([ (1,4,5)
  (2,3,6), (1,3)(2,4)(5,6) ]) ]
gap> Size(list);
6
gap> Size(AllSubgroups(SymmetricGroup(3)));
6
```

3.3.4 ConjugacyClassRepsCompatibleSubgroups (for IsLocalAction)

▷ `ConjugacyClassRepsCompatibleSubgroups(`F`)` (attribute)

Returns: a list of compatible representatives of conjugacy classes of F that contain a compatible subgroup.

The argument of this method is a local action F of $\mathrm{Aut}(B_{d,k})$.

Example

```
gap> ConjugacyClassRepsCompatibleSubgroups(AutBall(3,2));
[ Group(()), Group([ (1,2)(3,5)(4,6) ]), Group([ (1,4,5)(2,3,6) ]),
  Group([ (3,5)(4,6), (1,2) ]), Group([ (1,2)(3,5)(4,6), (1,3,6)
  (2,4,5) ]), Group([ (3,5)(4,6), (1,3,5)(2,4,6), (1,2)(3,4)(5,6) ]),
```

```
  Group([ (1,2)(3,5)(4,6), (1,3,5)(2,4,6), (1,2)(5,6), (1,2)(3,4) ]),
  Group([ (3,5)(4,6), (1,3,5)(2,4,6), (1,2)(5,6), (1,2)(3,4) ]),
  Group([ (5,6), (3,4), (1,2), (1,3,5)(2,4,6), (3,5)(4,6) ]) ]
```

3.3.5 ConjugacyClassRepsCompatibleGroupsWithProjection

▷ `ConjugacyClassRepsCompatibleGroupsWithProjection(l, F)` (function)

Returns: a list of compatible representatives of conjugacy classes of $\mathrm{Aut}(B_{d,l})$ that contain a compatible group which projects to $F \le \mathrm{Aut}(B_{d,r})$.

The arguments of this method are a radius $l \in \mathbb{N}$, and a local action $F \le \mathrm{Aut}(B_{d,k})$ for some $k \le l$.

Example

```
gap> S3:=LocalAction(3,1,SymmetricGroup(3));
Sym( [ 1 .. 3 ] )
gap> ConjugacyClassRepsCompatibleGroupsWithProjection(2,S3);
[ Group([ (1,2)(3,5)(4,6), (1,4,5)(2,3,6) ]),
  Group([ (1,2)(3,4)(5,6), (1,2)(3,5)(4,6), (1,4,5)(2,3,6) ]),
  Group([ (3,4)(5,6), (1,2)(3,4), (1,4,5)(2,3,6), (3,5,4,6) ]),
  Group([ (3,4)(5,6), (1,2)(3,4), (1,4,5)(2,3,6), (3,5)(4,6) ]),
  Group([ (3,4)(5,6), (1,2)(3,4), (1,4,5)(2,3,6), (5,6), (3,5,4,6) ]) ]
gap> A3:=LocalAction(3,1,AlternatingGroup(3));
Alt( [ 1 .. 3 ] )
gap> ConjugacyClassRepsCompatibleGroupsWithProjection(2,A3);
[ Group([ (1,4,5)(2,3,6) ]) ]
```

Example

```
gap> F:=SymmetricGroup(3);;
gap> rho:=SignHomomorphism(F);;
gap> H1:=PI(2,3,F,rho,[0,1]);;
gap> H2:=PI(2,3,F,rho,[1]);;
gap> Size(ConjugacyClassRepsCompatibleGroupsWithProjection(3,H1));
2
gap> Size(ConjugacyClassRepsCompatibleGroupsWithProjection(3,H2));
4
```

Chapter 4

Examples

Several classes of examples of subgroups of $\mathrm{Aut}(B_{d,k})$ that satisfy (C) and or (D) are constructed in [Tor20] and implemented in this section. For a given permutation group $F \le S_d$, there are always the three local actions $\Gamma(F)$, $\Delta(F)$ and $\Phi(F)$ on $\mathrm{Aut}(B_{d,2})$ that project onto F. For some F, these are all distinct and yield all universal groups that have F as their 1-local action, see [Tor20, Theorem 3.32]. More examples arise in particular when either point stabilizers in F are not simple, F preserves a partition, or F is not perfect.

4.1 Discrete groups

Here, we implement the local actions $\Gamma(F), \Delta(F) \le \mathrm{Aut}(B_{d,2})$, both of which satisfy both (C) and (D), see [Tor20, Section 3.4.1].

4.1.1 gamma

▷ `gamma(d, a)` (operation)
▷ `gamma(l, d, a)` (operation)
▷ `gamma(l, d, s, addr)` (operation)
▷ `gamma(d, k, aut, z)` (operation)

for the arguments `d`, `a`

Returns: the automorphism $\gamma(\mathtt{a}) = (\mathtt{a}, (\mathtt{a})_{\omega\in\Omega}) \in \mathrm{Aut}(B_{d,2})$.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\ge 3}$ and a permutation `a` $\in S_d$.

for the arguments `l`, `d`, `a`

Returns: the automorphism $\gamma^l(\mathtt{a}) \in \mathrm{Aut}(B_{d,l})$ all of whose 1-local actions are given by `a`.

The arguments of this method are a radius `l` $\in \mathbb{N}$, a degree `d` $\in \mathbb{N}_{\ge 3}$ and a permutation `a` $\in S_d$.

for the arguments `l`, `d`, `s`, `addr`

Returns: the automorphism of $B_{d,k}$ whose 1-local actions are given by `s` at vertices whose address has `addr` as a prefix and are trivial elsewhere.

The arguments of this method are a radius `l` $\in \mathbb{N}$, a degree `d` $\in \mathbb{N}_{\ge 3}$, a permutation `s` $\in S_d$ and an address `addr` of a vertex in $B_{d,l}$ whose last entry is fixed by `s`.

for the arguments `d`, `k`, `aut`, `z`

Returns: the automorphism $\gamma_z(\mathtt{aut}) = (\mathtt{aut}, (z(\mathtt{aut}, \omega))_{\omega\in\Omega}) \in \mathrm{Aut}(B_{d,k+1})$.

The arguments of this method are a degree $d \in \mathbb{N}_{\geq 3}$, a radius $k \in \mathbb{N}$, an automorphism `aut` of $B_{d,k}$, and an involutive compatibility cocycle `z` of a subgroup of $\mathrm{Aut}(B_{d,k})$ that contains `aut` (see `InvolutiveCompatibilityCocycle` (**??**)).

Example
```
gap> gamma(3,(1,2));
(1,3)(2,4)(5,6)
```

Example
```
gap> gamma(2,3,(1,2));
(1,3)(2,4)(5,6)
gap> gamma(3,3,(1,2));
(1,5)(2,6)(3,8)(4,7)(9,11)(10,12)
```

Example
```
gap> gamma(3,3,(1,2),[1,3]);
(3,4)
gap> gamma(3,3,(1,2),[]);
(1,5)(2,6)(3,8)(4,7)(9,11)(10,12)
```

Example
```
gap> S3:=SymmetricGroup(3);;
gap> z1:=AllInvolutiveCompatibilityCocycles(3,1,S3)[1];;
gap> gamma(3,1,(1,2),z1);
(1,4)(2,3)(5,6)
gap> z3:=AllInvolutiveCompatibilityCocycles(3,1,S3)[3];;
gap> gamma(3,1,(1,2),z3);
(1,3)(2,4)(5,6)
```

4.1.2 GAMMA

▷ `GAMMA(d, F)` (operation)
▷ `GAMMA(l, d, F)` (operation)
▷ `GAMMA(F, z)` (operation)

for the arguments `d`, `F`

Returns: the local action $\Gamma(F) = \{(a, (a)_\omega) \mid a \in F\} \leq \mathrm{Aut}(B_{d,2})$.

The arguments of this method are a degree $d \in \mathbb{N}_{\geq 3}$, and a subgroup `F` of S_d.

for the arguments `l`, `d`, `F`

Returns: the group $\Gamma^l(F) \leq \mathrm{Aut}(B_{d,l})$.

The arguments of this method are a radius $l \in \mathbb{N}$, a degree $d \in \mathbb{N}_{\geq 3}$, and a subgroup `F` of S_d.

for the arguments `d`, `k`, `F`, `z`

Returns: the group $\Gamma_z(F) = \{(a, (z(a, \omega))_{\omega\in\Omega}) \mid a \in F\} \leq \mathrm{Aut}(B_{d,k+1})$.

The arguments of this method are a local action $F \leq \mathrm{Aut}(B_{d,k})$ and an involutive compatibility cocycle `z` of `F` (see `InvolutiveCompatibilityCocycle` (**??**)).

Example

```
gap> F:=TransitiveGroup(4,3);;
gap> GAMMA(4,F);
Group([ (1,5,9,10)(2,6,7,11)(3,4,8,12), (1,8)(2,7)(3,9)(4,5)(10,12) ])
```

Example

```
gap> GAMMA(3,SymmetricGroup(3));
Group([ (1,4,5)(2,3,6), (1,3)(2,4)(5,6) ])
gap> GAMMA(2,3,SymmetricGroup(3));
Group([ (1,4,5)(2,3,6), (1,3)(2,4)(5,6) ])
gap> GAMMA(3,3,SymmetricGroup(3));
Group([ (1,8,10)(2,7,9)(3,5,12)(4,6,11), (1,5)(2,6)(3,8)(4,7)(9,11)(10,12) ])
```

Example

```
gap> F:=SymmetricGroup(3);;
gap> rho:=SignHomomorphism(F);;
gap> H:=PI(2,3,F,rho,[1]);;
gap> z:=InvolutiveCompatibilityCocycle(3,2,H);;
gap> GAMMA(H,z);
Group([ (), (), (1,8,9)(2,7,10)(3,5,11)(4,6,12), (1,8,9)(2,7,10)(3,5,11)(4,6,12),
  (1,7,3,5)(2,8,4,6)(9,11,10,12) ])
```

4.1.3 DELTA

▷ DELTA(*d*, *F*) (operation)
▷ DELTA(*d*, *F*, *C*) (operation)

for the arguments d, F

Returns: the group $\Delta(F) \leq \mathrm{Aut}(B_{d,2})$.

The arguments of this method are a degree $d \in \mathbb{N}_{\geq 3}$, and a *transitive* subgroup F of S_d.

for the arguments d, F, C

Returns: the group $\Delta(F,C) \leq \mathrm{Aut}(B_{d,2})$.

The arguments of this method are a degree $d \in \mathbb{N}_{\geq 3}$, a *transitive* subgroup F of S_d, and a central subgroup C of the stabilizer F_1 of 1 in F.

Example

```
gap> F:=SymmetricGroup(3);;
gap> D:=DELTA(3,F);
Group([ (1,3,6)(2,4,5), (1,3)(2,4), (1,2)(3,4)(5,6) ])
gap> F1:=Stabilizer(F,1);;
gap> D1:=DELTA(3,F,F1);
Group([ (1,4,5)(2,3,6), (1,3)(2,4)(5,6), (1,2)(3,4)(5,6) ])
gap> D=D1;
false
gap> G:=AutBall(3,2);;
gap> D^G=D1^G;
true
```

Example

```
gap> F:=PrimitiveGroup(5,3);
AGL(1, 5)
```

```
gap> F1:=Stabilizer(F,1);
Group([ (2,3,4,5) ])
gap> C:=Group((2,4)(3,5));
Group([ (2,4)(3,5) ])
gap> Index(F1,C);
2
gap> Index(DELTA(5,F,F1),DELTA(5,F,C));
2
```

4.2 Maximal extensions

For any $F \leq \mathrm{Aut}(B_{d,k})$ that satisfies (C), the group $\Phi(F) \leq \mathrm{Aut}(B_{d,k+1})$ is the maximal extension of F that satisfies (C) as well. It stems from the action of $\mathrm{U}_k(F)$ on balls of radius $k+1$ in T_d.

4.2.1 PHI

▷ `PHI(d, F)` (operation)
▷ `PHI(F)` (operation)
▷ `PHI(l, F)` (operation)

for the arguments `d`, `F`

Returns: the group $\Phi(F) = \{(a,(a_\omega)_\omega) \mid a \in F,\ \forall \omega \in \Omega:\ a_\omega \in C_F(a,\omega)\} \leq \mathrm{Aut}(B_{d,2})$.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\geq 3}$ and a permutation group `F` $\leq S_d$.

for the arguments `d`, `k`, `F`

Returns: the group $\Phi_k(F) = \{(a,(a_\omega)_\omega) \mid a \in F,\ \forall \omega \in \Omega:\ a_\omega \in C_F(a,\omega)\} \leq \mathrm{Aut}(B_{d,k+1})$.

The argument of this method is a local action `F` $\leq \mathrm{Aut}(B_{d,k})$.

for the arguments `l`, `d`, `k`, `F`

Returns: the group $\Phi^l(F) = \Phi_{l-1} \circ \cdots \circ \Phi_{k+1} \circ \Phi_k(F) \leq \mathrm{Aut}(B_{d,l})$.

The arguments of this method are a radius `l` $\in \mathbb{N}$ and a local action `F` $\leq \mathrm{Aut}(B_{d,k})$.

Example

```
gap> PHI(3,SymmetricGroup(3));
Group([ (1,4,5)(2,3,6), (1,3)(2,4)(5,6), (1,2), (3,4), (5,6) ])
gap> last=AutBall(3,2);
true
gap> PHI(3,AlternatingGroup(3));
Group([ (1,4,5)(2,3,6) ])
gap> last=GAMMA(3,AlternatingGroup(3));
true
```

Example

```
gap> S3:=LocalAction(3,1,SymmetricGroup(3));;
gap> groups:=ConjugacyClassRepsCompatibleSubgroupsWithProjection(2,S3);
[ Group([ (1,2)(3,5)(4,6), (1,4,5)(2,3,6) ]),
  Group([ (1,2)(3,4)(5,6), (1,2)(3,5)(4,6), (1,4,5)(2,3,6) ]),
  Group([ (3,4)(5,6), (1,2)(3,4), (1,4,5)(2,3,6), (3,5,4,6) ]),
  Group([ (3,4)(5,6), (1,2)(3,4), (1,4,5)(2,3,6), (3,5)(4,6) ]),
```

```
  Group([ (3,4)(5,6), (1,2)(3,4), (1,4,5)(2,3,6), (5,6), (3,5,4,6) ]) ]
gap> for G in groups do Print(Size(G),",",Size(PHI(G)),"\n"); od;
6,6
12,12
24,192
24,192
48,3072
```

Example
```
gap> PHI(3,LocalAction(4,1,SymmetricGroup(4)));
<permutation group with 34 generators>
gap> last=AutBall(4,3);
true
```

Example
```
ap> rho:=SignHomomorphism(SymmetricGroup(3));;
gap> F:=PI(2,3,SymmetricGroup(3),rho,[1]);; Size(F);
24
gap> P:=PHI(4,F);; Size(P);
12288
gap> IsSubgroup(AutBall(3,4),P);
true
gap> SatisfiesC(P);
true
```

4.3 Normal subgroups and partitions

When point stabilizers in $F \leq S_d$ are not simple, or F preserves a partition, more universal groups can be constructed as follows.

4.3.1 PHI

▷ `PHI(d, F, N)` (operation)
▷ `PHI(d, F, P)` (operation)
▷ `PHI(F, P)` (operation)

for the arguments `d`, `F`, `N`

Returns: the group $\Phi(F,N) \leq \mathrm{Aut}(B_{d,2})$.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\geq 3}$, a *transitive* permutation group $F \leq S_d$ and a normal subgroup N of the stabilizer F_1 of 1 in F.

for the arguments `d`, `F`, `P`

Returns: the group $\Phi(F,P) = \{(a,(a_\omega)_\omega) \mid a \in F,\ a_\omega \in C_F(a,\omega) \text{ constant w.r.t. } P\} \leq \mathrm{Aut}(B_{d,2})$.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\geq 3}$ and a permutation group $F \leq S_d$ and a partition P of `[1..d]` preserved by F.

for the arguments `d`, `k`, `F`, `P`

Returns: the group $\Phi_k(F,P) = \{(\alpha,(\alpha_\omega)_\omega) \mid \alpha \in F,\ \alpha_\omega \in C_F(\alpha,\omega) \text{ constant w.r.t. } P\} \leq \mathrm{Aut}(B_{d,k+1})$.

The arguments of this method are a local action $F \leq \mathrm{Aut}(B_{d,k})$ and a partition P of `[1..d]` preserverd by $\pi F \leq S_d$. This method assumes that all compatibility sets with respect to a partition element are non-empty and that all compatibility sets of the identity with respect to a partition element are non-trivial.

Example
```
gap> F:=SymmetricGroup(4);;
gap> F1:=Stabilizer(P,1);
Sym( [ 2 .. 4 ] )
gap> grps:=NormalSubgroups(F1);
[ Sym( [ 2 .. 4 ] ), Alt( [ 2 .. 4 ] ), Group(()) ]
gap> N:=grps[2];
Alt( [ 2 .. 4 ] )
gap> PHI(4,F,N);
Group([ (1,5,9,10)(2,6,7,11)(3,4,8,12), (1,4)(2,5)(3,6)(7,8)(10,11),
  (1,2,3) ])
gap> Index(F1,N);
2
gap> Index(PHI(4,F,F1),PHI(4,F,N));
16
```

Example
```
gap> F:=TransitiveGroup(4,3);
D(4)
gap> P:=Blocks(F,[1..4]);
[ [ 1, 3 ], [ 2, 4 ] ]
gap> G:=PHI(4,F,P);
Group([ (1,5,9,10)(2,6,7,11)(3,4,8,12), (1,8)(2,7)(3,9)(4,5)(10,12), (1,3)
  (8,9), (4,5)(10,12) ])
gap> aut:=Random(G);
(1,5,9,10)(2,6,7,11)(3,4,8,12)
gap> LocalAction(1,4,2,a,[1]); LocalAction(1,4,2,a,[3]);
(1,2,3,4)
(1,2,3,4)
gap> LocalAction(1,4,2,a,[2]); LocalAction(1,4,2,a,[4]);
(1,4)(2,3)
(1,4)(2,3)
```

Example
```
gap> H:=TransitiveGroup(4,3);
D(4)
gap> P:=Blocks(H,[1..4]);
[ [ 1, 3 ], [ 2, 4 ] ]
gap> F:=PHI(4,H,P);;
gap> G:=PHI(F,P);
<permutation group with 5 generators>
gap> SatisfiesC(G);
true
```

4.4 Abelian quotients

When a permutation group $F \leq S_d$ is not perfect, i.e. it admits an abelian quotient $\rho : F \twoheadrightarrow A$, more universal groups can be constructed by imposing restrictions of the form $\prod_{r\in R}\prod_{x\in S(b,r)} \rho(\sigma_1(\alpha,x)) = 1$ on elements $\alpha \in \Phi^k(F) \leq \mathrm{Aut}(B_{d,k})$.

4.4.1 SignHomomorphism

▷ `SignHomomorphism(F)` (function)

Returns: the sign homomorphism from F to S_2.

The argument of this method is a permutation group $F \leq S_d$. This method can be used as an example for the argument *rho* in the methods `SpheresProduct` (4.4.3) and `PI` (4.4.4).

Example
```
gap> F:=SymmetricGroup(3);;
gap> sign:=SignHomomorphism(F);
MappingByFunction( Sym( [ 1 .. 3 ] ), Sym( [ 1 .. 2 ] ), function( g ) ... end )
gap> Image(sign,(2,3));
(1,2)
gap> Image(sign,(1,2,3));
()
```

4.4.2 AbelianizationHomomorphism

▷ `AbelianizationHomomorphism(F)` (function)

Returns: the homomorphism from F to $F/[F,F]$.

The argument of this method is a permutation group $F \leq S_d$. This method can be used as an example for the argument *rho* in the methods `SpheresProduct` (4.4.3) and `PI` (4.4.4).

Example
```
gap> F:=PrimitiveGroup(5,3);
AGL(1, 5)
gap> ab:=AbelianizationHomomorphism(PrimitiveGroup(5,3));
[ (2,3,4,5), (1,2,3,5,4) ] -> [ f1, <identity> of ... ]
gap> Elements(Range(ab));
[ <identity> of ..., f1, f2, f1*f2 ]
gap> StructureDescription(Range(ab));
"C4"
```

4.4.3 SpheresProduct

▷ `SpheresProduct(d, k, aut, rho, R)` (function)

Returns: the product $\prod_{r\in R}\prod_{x\in S(b,r)} rho(\sigma_1(aut,x)) \in \mathrm{im}(rho)$.

The arguments of this method are a degree $d \in \mathbb{N}_{\geq 3}$, a radius $k \in \mathbb{N}$, an automorphism *aut* of $B_{d,k}$ all of whose 1-local actions are in the domain of the homomorphism *rho* from a subgroup of S_d to an abelian group, and a sublist R of `[0..k-1]`. This method is used in the implementation of `PI` (4.4.4).

Example
```
gap> rho:=SignHomomorphism(SymmetricGroup(3));;
gap> SpheresProduct(3,2,gamma(2,3,(1,2)),rho,[0]);
(1,2)
gap> SpheresProduct(3,2,gamma(2,3,(1,2)),rho,[0,1]);
()
```

Example
```
gap> F:=PrimitiveGroup(5,3);
AGL(1, 5)
gap> rho:=AbelianizationHomomorphism(F);;
gap> Elements(Range(rho));
```

```
[ <identity> of ..., f1, f2, f1*f2 ]
gap> StructureDescription(Range(rho));
"C4"
gap> aut:=Random(F);
(1,2,4,5)
gap> SpheresProduct(5,3,gamma(3,5,aut),rho,[2]);
<identity> of ...
gap> SpheresProduct(5,3,gamma(3,5,aut),rho,[1,2]);
f1*f2
gap> SpheresProduct(5,3,gamma(3,5,aut),rho,[0,1,2]);
f2
```

4.4.4 PI

▷ `PI(l, d, F, rho, R)` (function)

Returns: the group $\Pi^l(F,rho,R) = \{\alpha \in \Phi^l(F) \mid \prod_{r\in R}\prod_{x\in S(b,r)} rho(\sigma_1(\alpha,x)) = 1\} \leq \mathrm{Aut}(B_{d,l})$.

The arguments of this method are a degree `l` $\in \mathbb{N}_{\geq 2}$, a radius `d` $\in \mathbb{N}_{\geq 3}$, a permutation group `F` $\leq S_d$, a homomorphism ρ from `F` to an abelian group that is surjective on every point stabilizer in `F`, and a non-empty, non-zero subset `R` of `[0..l-1]` that contains $l-1$.

Example
```
gap> F:=LocalAction(5,1,PrimitiveGroup(5,3));
AGL(1, 5)
gap> rho1:=AbelianizationHomomorphism(F);;
gap> rho2:=SignHomomorphism(F);;
gap> PI(3,5,F,rho1,[0,1,2]);
<permutation group with 4 generators>
gap> Index(PHI(3,F),last);
4
gap> PI(3,5,F,rho2,[0,1,2]);
<permutation group with 6 generators>
gap> Index(PHI(3,F),last);
2
```

4.5 Semidirect products

When a subgroup $F \leq \mathrm{Aut}(B_{d,k})$ satisfies (C) and admits an involutive compatibility cocycle z (which is automatic when $k=1$) one can characterise the kernels $K \leq \Phi_k(F) \cap \ker(\pi_k)$ that fit into a z-split exact sequence $1 \to K \to \Sigma(F,K) \to F \to 1$ for some subgroup $\Sigma(F,K) \leq \mathrm{Aut}(B_{d,k+1})$ that satisfies (C). This characterisation is implemented in this section.

4.5.1 CompatibleKernels

▷ `CompatibleKernels(d, F)` (operation)
▷ `CompatibleKernels(F, z)` (operation)

for the arguments `d, F`

Returns: the list of kernels $K \leq \prod_{\omega\in\Omega} F_\omega \cong \ker\pi \leq \mathrm{Aut}(B_{d,2})$ that are preserved by the action $F \curvearrowright \prod_{\omega\in\Omega} F_\omega$, $a \cdot (a_\omega)_\omega := (a a_{a^{-1}\omega} a^{-1})_\omega$.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\geq 3}$, and a permutation group `F` $\leq S_d$. The kernels output by this method are compatible with `F` with respect to the standard cocycle (see `InvolutiveCompatibilityCocycle` (**??**)) and can be used in the method `SIGMA` (4.5.2).

for the arguments `d`**,** `k`**,** `F`**,** `z`

Returns: the list of kernels $K \leq \Phi_k(F) \cap \ker(\pi_k) \leq \mathrm{Aut}(B_{d,k+1})$ that are normalized by $\Gamma_z(F)$ and such that for all $k \in K$ and $\omega \in \Omega$ there is $k_\omega \in K$ with $\mathrm{pr}_\omega k_\omega = z(\mathrm{pr}_\omega k, \omega)^{-1}$.

The arguments of this method are a local action `F` $\leq \mathrm{Aut}(B_{d,k})$ that satisfies (C) and an involutive compatibility cocycle `z` of `F` (see `InvolutiveCompatibilityCocycle` (**??**)). It can be used in the method `SIGMA` (4.5.2).

Example
```
gap> CompatibleKernels(3,SymmetricGroup(3));
[ Group(()), Group([ (1,2)(3,4)(5,6) ]), Group([ (3,4)(5,6), (1,2)(5,6) ]),
  Group([ (5,6), (3,4), (1,2) ]) ]
```

Example
```
gap> P:=SymmetricGroup(3);;
gap> rho:=SignHomomorphism(P);;
gap> F:=PI(2,3,P,rho,[1]);;
gap> z:=InvolutiveCompatibilityCocycle(F);;
[ Group(()), Group([ (1,2)(3,4)(5,6)(7,8)(9,10)(11,12) ]),
  Group([ (1,2)(3,4)(5,6)(7,8), (5,6)(7,8)(9,10)(11,12) ]),
  Group([ (5,6)(7,8), (1,2)(3,4), (9,10)(11,12) ]) ]
```

4.5.2 SIGMA

▷ `SIGMA(d, F, K)` (operation)

▷ `SIGMA(F, K, z)` (operation)

for the arguments `d`**,** `F`**,** `K`

Returns: the semidirect product $\Sigma(F,K) \leq \mathrm{Aut}(B_{d,2})$.

The arguments of this method are a degree `d` $\in \mathbb{N}_{\geq 3}$, a subgroup `F` of S_d and a compatible kernel `K` for `F` (see `CompatibleKernels` (4.5.1)).

for the arguments `d`**,** `k`**,** `F`**,** `K`**,** `z`

Returns: the semidirect product $\Sigma_z(F,K) \leq \mathrm{Aut}(B_{d,k+1})$.

The arguments of this method are a local action `F` of $\mathrm{Aut}(B_{d,k})$ that satisfies (C) and a kernel `K` that is compatible for `F` with respect to the involutive compatibility cocycle `z` (see `InvolutiveCompatibilityCocycle` (**??**) and `CompatibleKernels` (4.5.1)) of `F`.

Example
```
gap> S3:=SymmetricGroup(3);;
gap> kernels:=CompatibleKernels(3,S3);
[ Group(()), Group([ (1,2)(3,4)(5,6) ]), Group([ (3,4)(5,6), (1,2)(5,6) ]),
  Group([ (5,6), (3,4), (1,2) ]) ]
gap> for K in kernels do Print(Size(SIGMA(3,S3,K)),"\n"); od;
6
12
24
48
```

Example

```
gap> P:=SymmetricGroup(3);;
gap> rho:=SignHomomorphism(P);;
gap> F:=PI(2,3,P,rho,[1]);;
gap> z:=InvolutiveCompatibilityCocycle(F);;
gap> kernels:=CompatibleKernels(F,z);
[ Group(()), Group([ (1,2)(3,4)(5,6)(7,8)(9,10)(11,12) ]),
  Group([ (1,2)(3,4)(5,6)(7,8), (5,6)(7,8)(9,10)(11,12) ]),
  Group([ (5,6)(7,8), (1,2)(3,4), (9,10)(11,12) ]) ]
gap> for K in kernels do Print(Size(SIGMA(F,K,z)),"\n"); od;
24
48
96
192
```

Chapter 5

Discreteness

This chapter contains functions that are related to the discreteness property (D) presented in Proposition 3.12 of [Tor20].

5.1 The discreteness condition (D)

Said proposition shows that for a given $F \leq \mathrm{Aut}(B_{d,k})$ the group $\mathrm{U}_k(F)$ is discrete if and only if the maximal compatible subgroup $C(F)$ of F satisfies condition (D):

$$\forall \omega \in \Omega : F_{T_\omega} = \{\mathrm{id}\},$$

where T_ω is the $k-1$-neighbourhood of the the edge (b, b_ω) inside $B_{d,k}$. In other words, F satisfies (D) if and only if the compatibility set $C_F(\mathrm{id}, \omega) = \{\mathrm{id}\}$. We distinguish between F satisfying condition (D) and $\mathrm{U}_k(F)$ being discrete with the methods `SatisfiesD` (**??**) and `IsDiscrete` (**??**) below.

5.2 Discreteness

5.2.1 SatisfiesD (for IsLocalAction)

▷ `SatisfiesD(F)` (property)

Returns: `true` if F satisfies the discreteness condition (D), and `false` otherwise.

The argument of this attribute is a local action $F \leq \mathrm{Aut}(B_{d,k})$ (see `IsLocalAction` (**??**)).

Example

```
gap> G:=GAMMA(3,SymmetricGroup(3));
Group([ (1,4,5)(2,3,6), (1,3)(2,4)(5,6) ])
gap> SatisfiesD(G);
true
```

5.2.2 IsDiscrete (for IsLocalAction)

▷ `IsDiscrete(F)` (property)

Returns: `true` if $\mathrm{U}_k(F)$ is discrete, and `false` otherwise.

The argument of this attribute is a local action $F \leq \mathrm{Aut}(B_{d,k})$ (see `IsLocalAction` (**??**)).

Example
```
gap> G:=GAMMA(3,SymmetricGroup(3));
Group([ (1,4,5)(2,3,6), (1,3)(2,4)(5,6) ])
gap> IsDiscrete(G);
true
```

Example
```
gap> F:=LocalAction(3,2,Group((1,2)));
Group([ (1,2) ])
gap> IsDiscrete(F);
true
gap> SatisfiesD(F);
false
gap> C:=MaximalCompatibleSubgroup(F);
Group(())
gap> SatisfiesD(C);
true
```

5.3 Cocycles

Subgroups $F \leq \mathrm{Aut}(B_{d,k})$ that satisfy both (C) and (D) admit an involutive compatibility cocycle, i.e. a map $z : F \times \{1, \dots, d\} \to F$ that satisfies certain properties, see [Tor20, Section 3.2.2]. When F satisfies just (C), it may still admit an involutive compatibility cocycle. In this case, F admits an extension $\Gamma_z(F) \leq \mathrm{Aut}(B_{d,k})$ that satisfies both (C) and (D). Involutive compatibility cocycles can be searched for using `InvolutiveCompatibilityCocycle` (**??**) and `AllInvolutiveCompatibilityCocycles` (**??**) below.

5.3.1 InvolutiveCompatibilityCocycle (for IsLocalAction)

▷ `InvolutiveCompatibilityCocycle(F)` (attribute)

Returns: an involutive compatibility cocycle of F, which is a mapping $F \times$`[1..d]`$\to F$ with certain properties, if it exists, and `fail` otherwise. When $k = 1$, the standard cocycle is returned.

The argument of this attribute is a local action $F \leq \mathrm{Aut}(B_{d,k})$ (see `IsLocalAction` (**??**)), which is compatible (see `SatisfiesC` (**??**)).

Example
```
gap> F:=LocalAction(3,1,AlternatingGroup(3));;
gap> InvolutiveCompatibilityCocycle(F);
MappingByFunction( Domain([ [ (), 1 ], [ (), 2 ], [ (), 3 ], [ (1,3,2), 1 ],
  [ (1,3,2), 2 ], [ (1,3,2), 3 ], [ (1,2,3), 1 ], [ (1,2,3), 2 ],
  [ (1,2,3), 3 ] ]), Alt( [ 1 .. 3 ] ), function( s ) ... end )
gap> a:=Random(AlternatingGroup(3));; dir:=Random([1..3]);;
gap> a; Image(z,[a,dir]);
(1,3,2)
(1,3,2)
```

Example
```
gap> G:=GAMMA(3,AlternatingGroup(3));
Group([ (1,4,5)(2,3,6) ])
gap> InvolutiveCompatibilityCocycle(G);
MappingByFunction( Domain([ [ (), 1 ], [ (), 2 ], [ (), 3 ],
  [ (1,5,4)(2,6,3), 1 ], [ (1,5,4)(2,6,3), 2 ], [ (1,5,4)(2,6,3), 3 ],
```

```
  [ (1,4,5)(2,3,6), 1 ], [ (1,4,5)(2,3,6), 2 ], [ (1,4,5)(2,3,6), 3 ]
 ]), Group([ (1,4,5)(2,3,6) ]), function( s ) ... end )
gap> InvolutiveCompatibilityCocycle(AutBall(3,2));
fail
```

5.3.2 AllInvolutiveCompatibilityCocycles (for IsLocalAction)

▷ `AllInvolutiveCompatibilityCocycles(`F`)` (attribute)

Returns: the list of all involutive compatibility cocycles of F.

The argument of this attribute is a local action $F \leq \mathrm{Aut}(B_{d,k})$ (see `IsLocalAction` (**??**)), which is compatible (see `SatisfiesC` (**??**)).

Example
```
gap> S3:=LocalAction(3,1,SymmetricGroup(3));;
gap> Size(AllInvolutiveCompatibilityCocycles(S3));
4
gap> Size(AllInvolutiveCompatibilityCocycles(GAMMA(3,SymmetricGroup(3))));
1
```

References

[BEW15] C. Banks, M. Elder, and G. Willis. Simple groups of automorphisms of trees determined by their actions on finite subtrees. *Journal of Group Theory*, 18(2):235–261, 2015. 4, 5

[BM00] M. Burger and S. Mozes. Groups acting on trees: from local to global structure. *Publications Mathématiques de l'IHÉS*, 92(1):113–150, 2000. 4, 5

[KM08] B. Krön and R. Möller. Analogues of cayley graphs for topological groups. *Mathematische Zeitschrift*, 258(3):637, 2008. 5

[Ser80] J. P. Serre. *Trees*. Springer, 1980. 4, 6

[Tit70] J. Tits. Sur le groupe des automorphismes d'un arbre. In *Essays on topology and related topics*, pages 188–211. Springer, 1970. 5

[Tor20] S. Tornier. Groups acting on trees with prescribed local action. *arxiv preprint: 2002.09876*, 2020. 4, 5, 6, 11, 12, 17, 27, 28

Index