SRGroups: Self-replicating groups of regular rooted trees.

Self-replicating groups of regular rooted trees.

0.1

17 December 2020

Sam King

Sarah Shotter

Sam King
Email: sam.king@newcastle.edu.au
Address: University Drive, Callaghan NSW 2308

Sarah Shotter
Email: sarah.shotter@newcastle.edu.au
Address: University Drive, Callaghan NSW 2308

Abstract

SRGroups is a package for searching up self-replicating groups of regular rooted trees and performing computations on these groups. This package allows the user to generate more self-replicating groups at greater depths with its in-built functions, and is an extension of the transgrp package.

Acknowledgements

DE210100180, FL170100032.

Contents

Chapter 1

Introduction

Let G be a subgroup of $\mathrm{Aut}(T_{k,k})$ with its group action, α, defined as $\alpha(g,x) = g(x)$, where $g \in G$ are the automorphisms of G and $x \in X$ the vertices of $T_{k,k}$. Let $\mathrm{stab}_G(0) = \{g \in G : \alpha(g,0) = 0\}$, and $T_0 \subset T_{k,k}$ be the set of all vertices below and including the vertex 0. Additonally, let $\varphi_0 : \mathrm{stab}_G(0) \to G$ be a group homomorphism with the mapping $g \mapsto g|_{T_0}$. Then G is called self-replicating if and only if the following two conditions, $\mathscr{R}_k$, are satisfied: G is vertex transitive on level 1 of $T_{k,k}$, and $\varphi_0\left(\mathrm{stab}_G(0)\right) = G$.

Chapter 2

Functionality

2.1 Methods

2.1.1 IsRegularRootedTreeGroup (for IsPermGroup)

▷ `IsRegularRootedTreeGroup(G)` (filter)

Returns: `true` or `false`

The argument of this category is any permutation group, *G*. Checks whether *G* is a regular rooted tree group.

2.1.2 RegularRootedTreeGroupDegree (for IsRegularRootedTreeGroup)

▷ `RegularRootedTreeGroupDegree(G)` (attribute)

Returns: The degree of *G*.

The argument of this attribute is any regular rooted tree group, *G*.

Example

```
gap> RegularRootedTreeGroupDepth(AutT(2,3));
3
```

2.1.3 RegularRootedTreeGroupDepth (for IsRegularRootedTreeGroup)

▷ `RegularRootedTreeGroupDepth(G)` (attribute)

Returns: The depth of *G*.

The argument of this attribute is any regular rooted tree group, *G*.

Example

```
gap> RegularRootedTreeGroupDegree(AutT(2,3));
2
```

2.1.4 RegularRootedTreeGroup (for IsInt, IsInt, IsPermGroup)

▷ `RegularRootedTreeGroup(k, n, G)` (operation)

Returns: The regular rooted tree group *G* as an object of the category `IsRegularRootedTreeGroup` (??), with attributes `RegularRootedTreeGroupDegree` (??) and `RegularRootedTreeGroupDepth` (??).

The arguments of this operation are a regular rooted tree group, *G*, and its degree *k* and depth *n*.

2.1.5 IsSelfReplicating (for IsRegularRootedTreeGroup)

▷ `IsSelfReplicating(G)` (property)

Returns: `true` or `false`

The argument of this property is any regular rooted tree group, `G`. Tests whether `G` satisfies the self-replicating conditions.

Example
```
gap> IsSelfReplicating(AutT(2,3));
true
```

2.1.6 HasSufficientRigidAutomorphisms (for IsRegularRootedTreeGroup)

▷ `HasSufficientRigidAutomorphisms(G)` (property)

Returns: `true` or `false`

The argument of this property is any regular rooted tree group, `G`. Tests whether `G` has sufficient rigid automorphisms.

Example
```
gap> HasSufficientRigidAutomorphisms(AutT(2,3));
true
```

2.1.7 ParentGroup (for IsRegularRootedTreeGroup)

▷ `ParentGroup(G)` (attribute)

Returns: The image of `G` when projected onto the automorphism group of degree `k` and depth `n-1`.

The argument of this attribute is any regular rooted tree group, `G`, of degree `k` and depth `n`.

Example
```
gap> G:=AutT(2,3); H:=AutT(2,2);
Group([ (1,2), (3,4), (5,6), (7,8), (1,3)(2,4), (5,7)(6,8), (1,5)(2,6)(3,7)(4,8) ])
Group([ (1,2), (3,4), (1,3)(2,4) ])
gap> ParentGroup(G);
Group([ (1,2), (1,3)(2,4), (3,4) ])
gap> H=last;
true
```

2.1.8 MaximalExtension (for IsRegularRootedTreeGroup)

▷ `MaximalExtension(G)` (attribute)

Returns: The maximal extension of `G`, `M(G)`, that is a subgroup of the automorphism group of degree `k` and depth `n+1`.

The argument of this attribute is any regular rooted tree group, `G`, of degree `k` and depth `n`.

Example
```
gap> G:=AutT(2,3); H:=AutT(2,4);
Group([ (1,2), (3,4), (5,6), (7,8), (1,3)(2,4), (5,7)(6,8), (1,5)(2,6)(3,7)(4,8) ])
<permutation group of size 32768 with 15 generators>
gap> MaximalExtension(G);
<permutation group with 11 generators>
gap> H=last;
true
```

2.1.9 RepresentativeWithSufficientRigidAutomorphisms (for IsRegularRootedTreeGroup)

▷ `RepresentativeWithSufficientRigidAutomorphisms(`*`G`*`)` (attribute)

Returns: A conjugate of *`G`* with sufficient rigid automorphisms.

The argument of this attribute is any regular rooted tree group, *`G`*.

Example
```
gap>
```

2.2 Library Functions

2.2.1 AllSRGroups

▷ `AllSRGroups(`*`Input1, val1, Input2, val2, ...`*`)` (function)

Returns: All of the self-replicating group(s) stored as objects satisfying all of the provided input arguments.

Main library search function. Has several possible input arguments such as *`Degree`*, *`Level`* (or *`Depth`*), *`Number`*, *`Projection`*, *`Subgroup`*, *`Size`*, *`NumberOfGenerators`*, and *`IsAbelian`*. Order of the inputs do not matter.

Example
```
gap> AllSRGroups(Degree, 2, Level, 4, IsAbelian, true);
[ SRGroup(2,4,2), SRGroup(2,4,9), SRGroup(2,4,12), SRGroup(2,4,14) ]
gap> Size(last[1]);
16
gap> AllSRGroups(Degree, 2, Level, 4, NumberOfGenerators, 4);
[ SRGroup(2,4,11), SRGroup(2,4,12), SRGroup(2,4,16), SRGroup(2,4,20), SRGroup(2,4,23), SRGroup(2,
 SRGroup(2,4,25), SRGroup(2,4,26), SRGroup(2,4,40), SRGroup(2,4,43), SRGroup(2,4,46), SRGroup(2,4
 SRGroup(2,4,50), SRGroup(2,4,66), SRGroup(2,4,70), SRGroup(2,4,71), SRGroup(2,4,72), SRGroup(2,4
 SRGroup(2,4,74), SRGroup(2,4,75), SRGroup(2,4,76), SRGroup(2,4,84), SRGroup(2,4,90), SRGroup(2,4
 SRGroup(2,4,93), SRGroup(2,4,95), SRGroup(2,4,97), SRGroup(2,4,102), SRGroup(2,4,108) ]
```

2.2.2 AllSRGroupsInfo

▷ `AllSRGroupsInfo(`*`Input1, val1, Input2, val2, ...`*`)` (function)

Returns: Information about the self-replicating group(s) satisfying all of the provided input arguments in list form: [*`Generators`*, *`Name`*, *`Parent Name`*, *`Children Name(s)`*]. If the *`Position`* input is provided, only the corresponding index of this list is returned.

Inputs work the same as the main library search function `AllSRGroups` (2.2.1), with one additional input: *`Position`*.

Example
```
gap> AllSRGroupsInfo(Degree, 2, Level, 3, IsAbelian, true);
[ [ [ (1,5,4,8,2,6,3,7), (1,4,2,3)(5,8,6,7), (1,2)(3,4)(5,6)(7,8) ], "SRGroup(2,3,1)", "SRGroup(2
[ [ (1,5,2,6)(3,7,4,8), (1,3)(2,4)(5,7)(6,8), (1,2)(3,4)(5,6)(7,8) ], "SRGroup(2,3,4)", "SRGroup(
[ [ (1,3)(2,4)(5,7)(6,8), (1,5)(2,6)(3,7)(4,8), (1,2)(3,4)(5,6)(7,8) ], "SRGroup(2,3,5)", "SRGrou
gap> AllSRGroupsInfo(Degree, 2, Level, 3, IsAbelian, true, Position, 1);
[ [ (1,5,4,8,2,6,3,7), (1,4,2,3)(5,8,6,7), (1,2)(3,4)(5,6)(7,8) ],
  [ (1,5,2,6)(3,7,4,8), (1,3)(2,4)(5,7)(6,8), (1,2)(3,4)(5,6)(7,8) ],
  [ (1,3)(2,4)(5,7)(6,8), (1,5)(2,6)(3,7)(4,8), (1,2)(3,4)(5,6)(7,8) ] ]
```

2.2.3 SRDegrees

▷ `SRDegrees()` (function)

Returns: All of the degrees currently stored in the SRGroups library (duplicates included).

There are no inputs to this function.

Example

```
gap> SRDegrees();
[ 2, 2, 2, 2, 3, 3, 3, 4, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 ]
```

2.2.4 SRLevels

▷ `SRLevels(k)` (function)

Returns: All of the levels currently stored in the SRGroups library for an input RegularRootedTreeGroupDegree, *deg*.

The input to this function is the degree of the regular rooted tree, *k*.

Example

```
gap> SRLevels(2);
[ 1, 2, 3, 4 ]
```

2.3 Package Functions

2.3.1 AutT

▷ `AutT(k, n)` (function)

Returns: The regular rooted tree group $\mathrm{Aut}(T_{k,n})$ as a permutation group of the k^n leaves of $T_{k,n}$.

The arguments of this function are a degree $k \in \mathbb{N}_{\geq 2}$ and a depth $n \in \mathbb{N}$.

Example

```
gap> G:=AutT(2,2);
Group([ (1,2), (3,4), (1,3)(2,4) ])
gap> Size(G);
8
```

2.3.2 BelowAction

▷ `BelowAction(k, n, aut, i)` (function)

Returns: The restriction of *aut* to the subtree below the level 1 vertex *i*, as an element of `AutT(k,n-1)`.

The arguments of this function are a degree, $k \in \mathbb{N}_{\geq 2}$, a depth, $n \in \mathbb{N}$, an element of `AutT(k,n)`, *aut*, and a level 1 vertex, $i \in \{1, \cdots, k\}$.

Example

```
gap> BelowAction(2,2,(1,2)(3,4),2);
(1,2)
```

Index