

SRGroups

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

To do.

Copyright

??? is free software; you can redistribute it and/or modify it under the terms of the [GNU General Public License](#) as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

Acknowledgements

DE210100180, FL170100032.

Contents

1	The package	4
1.1	Framework	4
1.2	Auxiliary methods	7
2	The library	9
2.1	Methods	9
	Index	10

Chapter 1

The package

?? is a package which does some interesting and cool things. To be continued...

1.1 Framework

Introduction... to do. Testing references: See `AutT` (1.2.2). See `RegularRootedTreeGroupDegree` (??). Why do function references work and attribute references don't?

1.1.1 `IsRegularRootedTreeGroup` (for `IsPermGroup`)

▷ `IsRegularRootedTreeGroup(arg)` (filter)

Returns: true or false

Groups acting on regular rooted trees are stored together with their degree (`RegularRootedTreeGroupDegree` (??)), depth (`RegularRootedTreeGroupDepth` (??)) and other attributes in this category.

Example

```
gap> G:=SymmetricGroup(3);
Sym( [ 1 .. 3 ] )
gap> IsRegularRootedTreeGroup(G);
false
gap> H:=RegularRootedTreeGroup(3,1,SymmetricGroup(3));
Sym( [ 1 .. 3 ] )
gap> IsRegularRootedTreeGroup(H);
true
```

1.1.2 `RegularRootedTreeGroup` (for `IsInt`, `IsInt`, `IsPermGroup`)

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

Returns: the regular rooted tree group G as an object of the category `IsRegularRootedTreeGroup` (??), checking that G is indeed a subgroup of $\text{Aut}(T_{k,n})$.

The arguments of this method are a degree $k \in \mathbb{N}_{\geq 2}$, a depth $n \in \mathbb{N}$ and a subgroup G of $\text{Aut}(T_{k,n})$.

Example

```
to do
```

1.1.3 RegularRootedTreeGroupNC (for IsInt, IsInt, IsPermGroup)

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

Returns: the regular rooted tree group G as an object of the category `IsRegularRootedTreeGroup (??)`, without checking that G is indeed a subgroup of $\text{Aut}(T_{k,n})$.

The arguments of this method are a degree $k \in \mathbb{N}_{\geq 2}$, a depth $n \in \mathbb{N}$ and a subgroup G of $\text{Aut}(T_{k,n})$.

Example

to do

1.1.4 RegularRootedTreeGroupDegree (for IsRegularRootedTreeGroup)

▷ `RegularRootedTreeGroupDegree(G)` (attribute)

Returns: the degree k of the regular rooted tree that G is acting on.

The argument of this attribute is a regular rooted tree group $G \leq \text{Aut}(T_{k,k})$ (`IsRegularRootedTreeGroup (??)`).

Example

to do

1.1.5 RegularRootedTreeGroupDepth (for IsRegularRootedTreeGroup)

▷ `RegularRootedTreeGroupDepth(G)` (attribute)

Returns: the depth n of the regular rooted tree that G is acting on.

The argument of this attribute is a regular rooted tree group $G \leq \text{Aut}(T_{k,k})$ (`IsRegularRootedTreeGroup (??)`).

Example

to do

1.1.6 ParentGroup (for IsRegularRootedTreeGroup)

▷ `ParentGroup(G)` (attribute)

Returns: the regular rooted tree group that arises from G by restricting to $T_{k,n-1}$.

The argument of this attribute is a regular rooted tree group $G \leq \text{Aut}(T_{k,k})$ (`IsRegularRootedTreeGroup (??)`).

Example

```
gap> G:=AutT(2,4);
<permutation group of size 32768 with 15 generators>
gap> ParentGroup(G)=AutT(2,3);
true
```

1.1.7 IsSelfReplicating (for IsRegularRootedTreeGroup)

▷ `IsSelfReplicating(G)` (property)

Returns: true, if G is self-replicating, and false otherwise.

The argument of this property is a regular rooted tree group $G \leq \text{Aut}(T_{k,k})$ (`IsRegularRootedTreeGroup (??)`).

Example

```

gap> G:=AutT(2,2);
Group([ (1,2), (3,4), (1,3)(2,4) ])
gap> subgroups:=AllSubgroups(G);
gap> Apply(subgroups,H->RegularRootedTreeGroup(2,2,H));
gap> for H in subgroups do Print(IsSelfReplicating(H),"\n"); od;
false
false
false
false
false
false
false
false
true
true
true

```

1.1.8 HasSufficientRigidAutomorphisms (for IsRegularRootedTreeGroup)

▷ HasSufficientRigidAutomorphisms(G) (property)

Returns: true, if G has sufficient rigid automorphisms, and false otherwise.

The argument of this property is a regular rooted tree group $G \leq \text{Aut}(T_{k,k})$ (IsRegularRootedTreeGroup (??)).

Example

```
to do
```

1.1.9 RepresentativeWithSufficientRigidAutomorphisms (for IsRegularRootedTreeGroup)

▷ RepresentativeWithSufficientRigidAutomorphisms(G) (attribute)

Returns: a regular rooted tree group which is conjugate to G in $\text{Aut}(T_{k,n})$ and which has sufficient rigid automorphisms, i.e. it satisfies HasSufficientRigidAutomorphisms (??). This returned group is G itself, if G already has sufficient rigid automorphisms. Furthermore, the returned group has the same parent group as G if the parent group of G has sufficient rigid automorphisms.

The argument of this attribute is a regular rooted tree group $G \leq \text{Aut}(T_{k,k})$ (IsRegularRootedTreeGroup (??)), which is self-replicating (IsSelfReplicating (??)).

Example

```
to do
```

1.1.10 MaximalExtension (for IsRegularRootedTreeGroup)

▷ MaximalExtension(G) (attribute)

Returns: the regular rooted tree group $M(G) \leq \text{Aut}(T_{k,n})$ which is the unique maximal self-replicating extension of G to $T_{k,n+1}$.

The argument of this attribute is a regular rooted tree group $G \leq \text{Aut}(T_{k,k})$ (IsRegularRootedTreeGroup (??)), which is self-replicating (IsSelfReplicating (??)) and has sufficient rigid automorphisms (HasSufficientRigidAutomorphisms (??)).

Example

```
to do
```

1.1.11 ConjugacyClassRepsSelfReplicatingGroupsWithProjection (for IsRegularRootedTreeGroup)

▷ ConjugacyClassRepsSelfReplicatingGroupsWithProjection(G) (attribute)

Returns: a list $\text{Aut}(T_{k,n+1})$ -conjugacy class representatives of regular rooted tree groups which are self-replicating, have sufficient rigid automorphisms and whose parent group is G .

The argument of this attribute is a regular rooted tree group $G \leq \text{Aut}(T_{k,k})$ ($\text{IsRegularRootedTreeGroup}(\text{??})$), which is self-replicating ($\text{IsSelfReplicating}(\text{??})$) and has sufficient rigid automorphisms ($\text{HasSufficientRigidAutomorphisms}(\text{??})$).

Example

to do

1.1.12 ConjugacyClassRepsSelfReplicatingGroupsWithConjugateProjection (for IsRegularRootedTreeGroup)

▷ ConjugacyClassRepsSelfReplicatingGroupsWithConjugateProjection(G) (attribute)

Returns: a list $\text{Aut}(T_{k,n+1})$ -conjugacy class representatives of regular rooted tree groups which are self-replicating, have sufficient rigid automorphisms and whose parent group is conjugate to G .

The argument of this attribute is a regular rooted tree group $G \leq \text{Aut}(T_{k,k})$ ($\text{IsRegularRootedTreeGroup}(\text{??})$), which is self-replicating ($\text{IsSelfReplicating}(\text{??})$) and has sufficient rigid automorphisms ($\text{HasSufficientRigidAutomorphisms}(\text{??})$).

Example

to do

1.2 Auxiliary methods

This section explains the methods of this package.

1.2.1 RemoveConjugates

▷ RemoveConjugates(G , $subgroups$) (function)

Returns: n/a. This method removes G -conjugates from the mutable list $subgroups$.

The arguments of this method are a group G and a mutable list $subgroups$ of subgroups of G .

Example

```
gap> G:=SymmetricGroup(3);
Sym( [ 1 .. 3 ] )
gap> subgroups:=[Group((1,2)),Group((2,3))];
[ Group([ (1,2) ]), Group([ (2,3) ]) ]
gap> RemoveConjugates(G,subgroups);
gap> subgroups;
[ Group([ (1,2) ]) ]
```

1.2.2 AutT

▷ AutT(k , n) (function)

Returns: the regular rooted tree group $\text{Aut}(T_{k,n})$ ($\text{IsRegularRootedTreeGroup}(\text{??})$) as a permutation group of the k^n leaves of $T_{k,n}$, generated as an iterated wreath product.

The arguments of this method 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> RegularRootedTreeGroupDegree(G);
2
gap> RegularRootedTreeGroupDepth(G);
2
```

1.2.3 BelowAction

▷ BelowAction(k, n, aut, i)

(function)

Returns: the automorphism of $\text{Aut}(T_{k,n})$ that arises from aut by restricting to the subtree below the i -th vertex at depth 1.

The arguments of this method are a degree $k \in \mathbb{N}_{\geq 2}$, a depth $n \in \mathbb{N}_{\neq 0}$, an automorphism $aut \in \text{Aut}(T_{k,n})$ and an index $i \in [1..k]$.

Example

```
gap> G:=AutT(2,2);
Group([ (1,2), (3,4), (1,3)(2,4) ])
gap> a:=Random(G);
(1,3,2,4)
gap> BelowAction(2,2,a,1);
()
gap> BelowAction(2,2,a,2);
(1,2)
```


Chapter 2

The library

2.1 Methods

2.1.1 bar (for IsObject)

▷ `bar(arg)`

Returns: true or false
foo

(filter)

Index

AutT, [7](#)

bar

for IsObject, [9](#)

BelowAction, [8](#)

ConjugacyClassRepsSelfReplicating-
GroupsWithConjugateProjection

for IsRegularRootedTreeGroup, [7](#)

ConjugacyClassRepsSelfReplicating-
GroupsWithProjection

for IsRegularRootedTreeGroup, [7](#)

HasSufficientRigidAutomorphisms

for IsRegularRootedTreeGroup, [6](#)

IsRegularRootedTreeGroup

for IsPermGroup, [4](#)

IsSelfReplicating

for IsRegularRootedTreeGroup, [5](#)

MaximalExtension

for IsRegularRootedTreeGroup, [6](#)

ParentGroup

for IsRegularRootedTreeGroup, [5](#)

RegularRootedTreeGroup

for IsInt, IsInt, IsPermGroup, [4](#)

RegularRootedTreeGroupDegree

for IsRegularRootedTreeGroup, [5](#)

RegularRootedTreeGroupDepth

for IsRegularRootedTreeGroup, [5](#)

RegularRootedTreeGroupNC

for IsInt, IsInt, IsPermGroup, [5](#)

RemoveConjugates, [7](#)

RepresentativeWithSufficientRigid-
Automorphisms

for IsRegularRootedTreeGroup, [6](#)