

OrbitalGraphs

Computations with orbital graphs in GAP

0.1.2

1 February 2022

Paula Hähndel
Markus Pfeiffer
Wilf A. Wilson

Paula Hähndel Email: paula.haehndel@mathematik.uni-halle.de
Homepage: <https://algebra.mathematik.uni-halle.de/haehndel>
Address: Martin-Luther-Universität Halle-Wittenberg
06099 Halle (Saale)

Markus Pfeiffer Email: markus.pfeiffer@st-andrews.ac.uk
Homepage: <https://markusp.morphism.de>
Address: School of Computer Science, University of St Andrews,
St Andrews, Fife, KY16 9SX, Scotland

Wilf A. Wilson Email: gap@wilf-wilson.net
Homepage: <https://wilf.me>
Address: School of Computer Science, University of St Andrews,
St Andrews, Fife, KY16 9SX, Scotland

Abstract

[]

Copyright

© 2018–present by Paula Hähndel, Markus Pfeiffer, and Wilf A. Wilson.
OrbitalGraphs is licensed under the Mozilla Public License, version 2.0.

Contents

1	Orbital graphs	4
1.1	Categories of orbital graphs	4
1.2	Constructing orbital graphs	4
1.3	Information stored about orbital graphs at creation	5
1.4	Values computed from the orbital graphs of a group	6
1.5	Recognising a group from its orbital graphs	7
1.6	Attributes and properties of individual orbital graphs	8
2	Visualising orbital graphs	9
2.1	Visualizing an orbital graph via the DOT format	9
	Index	10

Chapter 1

Orbital graphs

1.1 Categories of orbital graphs

1.1.1 IsOrbitalGraph (for IsDigraph)

▷ `IsOrbitalGraph( $D$ )` (Category)

Returns: true or false

Every orbital graph that is constructed with the `OrbitalGraphs` package is a `Digraphs` package digraph (see `IsDigraph` (**Digraphs:** `IsDigraph`)) that additionally lies in the category `IsOrbitalGraph`.

This makes it easy to recognise orbital graphs that were created with this package.

1.1.2 IsOrbitalGraphOfGroup (for IsOrbitalGraph)

▷ `IsOrbitalGraphOfGroup( $D$ )` (Category)

Returns: true or false

Every orbital graph that is constructed from a permutation group with the `OrbitalGraphs` package lies in the category `IsOrbitalGraphOfGroup`, which is a subcategory of `IsOrbitalGraph` (1.1.1).

1.1.3 IsOrbitalGraphOfSemigroup (for IsOrbitalGraph)

▷ `IsOrbitalGraphOfSemigroup( $D$ )` (Category)

Returns: true or false

Every orbital graph that is constructed from a transformation semigroup with the `OrbitalGraphs` package lies in the category `IsOrbitalGraphOfGroup` (1.1.2), which is a subcategory of `IsOrbitalGraph` (1.1.1).

1.2 Constructing orbital graphs

1.2.1 OrbitalGraphs (for IsPermGroup)

▷ `OrbitalGraphs( $G$ )` (attribute)

▷ `OrbitalGraphs( $S$ )` (attribute)

▷ `OrbitalGraphs( $G$ ,  $vertices$ )` (operation)

▷ `OrbitalGraphs( $G$ ,  $max$ )` (operation)

Returns: A list of orbital graphs

This returns a list of all orbital graphs of the permutation group G or transformation semigroup S . The order of the returned list is not specified.

Example

```
gap> D8 := Group([ (1,2,3,4), (2,4) ]);; StructureDescription(D8);
"D8"
gap> OrbitalGraphs(D8);
[ <self-paired orbital graph of D8 on 4 vertices with base-pair (1,3), 4 arcs>
  , <self-paired orbital graph of D8 on 4 vertices
    with base-pair (1,2), 8 arcs> ]
```

1.2.2 OrbitalGraph (for IsPermGroup, IsHomogeneousList, IsPosInt)

▷ `OrbitalGraph( $G$ ,  $basepair$ ,  $k$ )` (operation)

Returns: An orbital graph

If G is a permutation group, $basepair$ is a pair of positive integers, and k is a positive integer such that $[1..<A>k]$ is preserved by G and contains the entries of $basepair$, then this function returns the orbital graph of G with the given $basepair$, on the vertices $[1..<A>k]$.

The resulting orbital graph will have $basepair$ set as its `BasePair` (1.3.1) attribute.

Example

```
gap> D8 := DihedralGroup(IsPermGroup, 8);
Group([ (1,2,3,4), (2,4) ])
gap> OrbitalGraph(D8, [1, 3], 4);
<self-paired orbital graph of Group([ (1,2,3,4), (2,4) ]) on 4 vertices
with base-pair (1,3), 4 arcs>
gap> OrbitalGraph(D8, [1, 3], 5);
<self-paired orbital graph of Group([ (1,2,3,4), (2,4) ]) on 5 vertices
with base-pair (1,3), 4 arcs>
gap> G := Group([ (1,2)(3,4) ]);;
gap> OrbitalGraph(G, [1, 2], 2);
<self-paired orbital graph of Group([ (1,2)(3,4) ]) on 2 vertices
with base-pair (1,2), 2 arcs>
```

1.3 Information stored about orbital graphs at creation

1.3.1 BasePair (for IsOrbitalGraph)

▷ `BasePair( $D$ )` (attribute)

Returns: A list of two positive integers

If D is an orbital graph that was constructed with respect to a specific base pair, then this attribute stores that value.

Otherwise, if D is an orbital graph of a group, then this attribute stores the least edge of D , i.e. `Minimum(DigraphEdges(<A>D</A>))`; see `DigraphEdges` (**Digraphs: DigraphEdges**). If D is an orbital graph of a group, then this attribute stores an arbitrary base-pair of D .

Note that equal orbital graphs may have different base pairs, depending on how they were constructed.

Example

```
gap> true;
true
```

1.3.2 UnderlyingGroup (for IsOrbitalGraphOfGroup)

▷ UnderlyingGroup(D) (attribute)

Returns: A permutation group

For an orbital graph D created from a permutation group G , this attribute stores the value G . Note that equal orbital graphs may have been created from different groups, and may therefore have different underlying groups.

Example

```
gap> true;
true
```

1.3.3 UnderlyingSemigroup (for IsOrbitalGraphOfSemigroup)

▷ UnderlyingSemigroup(D) (attribute)

Returns: A transformation semigroup

For an orbital graph D created from a transformation semigroup S , this attribute stores the value S . Note that equal orbital graphs may have been created from different semigroups, and may therefore have different underlying semigroups.

Example

```
gap> true;
true
```

1.4 Values computed from the orbital graphs of a group

1.4.1 OrbitalClosure (for IsPermGroup)

▷ OrbitalClosure(G) (attribute)

Returns: A permutation group

The *orbital closure* of a nontrivial permutation group G is the intersection of the automorphism groups of all orbital graphs of the group. See OrbitalGraphs (1.2.1). A trivial permutation group is defined to be its own orbital closure.

For a transitive permutation group, OrbitalClosure returns the same as the GAP function TwoClosure (**Reference: TwoClosure**) (which only applies to transitive groups).

Example

```
gap> OrbitalClosure(PSL(2,5)) = SymmetricGroup(6);
true
gap> C6 := CyclicGroup(IsPermGroup, 6);
gap> OrbitalClosure(C6) = C6;
true
gap> A4_6 := Action(AlternatingGroup(4), Combinations([1..4], 2), OnSets);
gap> closure := OrbitalClosure(A4_6);
Group([ (3,4), (2,5), (1,2,3)(4,6,5) ])
gap> IsConjugate(SymmetricGroup(6),
>               closure, WreathProduct(Group([ (1,2) ]), Group([ (1,2,3) ])));
true
```

1.4.2 OrbitalIndex (for IsPermGroup)

▷ `OrbitalIndex( $G$ )` (attribute)

Returns: A positive integer

The *orbital index* of a permutation group is its Index (**Reference:** Index for a group and its subgroup) in its OrbitalClosure (1.4.1).

Example

```
gap> OrbitalIndex(PSL(2,5));
12
gap> OrbitalIndex(PGL(2,5));
6
gap> OrbitalIndex(AlternatingGroup(6));
2
gap> OrbitalIndex(DihedralGroup(IsPermGroup, 6));
1
```

1.5 Recognising a group from its orbital graphs

1.5.1 IsOrbitalGraphRecognisable (for IsPermGroup)

▷ `IsOrbitalGraphRecognisable( $G$ )` (property)

Returns: true or false

A permutation group is *orbital graph recognisable* if and only if it is equal to its OrbitalClosure (1.4.1), i.e. if and only if its OrbitalIndex (1.4.2) is 1.

IsOGR is a synonym for IsOrbitalGraphRecognisable.

Example

```
gap> IsOrbitalGraphRecognisable(QuaternionGroup(IsPermGroup, 8));
true
gap> IsOGR(AlternatingGroup(8));
false
gap> IsOGR(TrivialGroup(IsPermGroup));
true
```

1.5.2 IsStronglyOrbitalGraphRecognisable (for IsPermGroup)

▷ `IsStronglyOrbitalGraphRecognisable( $G$ )` (property)

Returns: true or false

The nontrivial permutation group G is *strongly orbital graph recognisable* (*strongly OGR*) if and only if there exists *some* orbital graph of G whose automorphism group is G . The trivial permutation group is defined to be strongly OGR.

Note that every strongly OGR group is also orbital graph recognisable, see IsOrbitalGraphRecognisable (1.5.1).

IsStronglyOGR is a synonym for IsStronglyOrbitalGraphRecognisable.

Example

```
gap> IsStronglyOrbitalGraphRecognisable(CyclicGroup(IsPermGroup, 8));
true
gap> IsStronglyOGR(QuaternionGroup(IsPermGroup, 8));
false
gap> IsStronglyOGR(TrivialGroup(IsPermGroup));
true
```

1.5.3 IsAbsolutelyOrbitalGraphRecognisable (for IsPermGroup)

▷ IsAbsolutelyOrbitalGraphRecognisable(*G*) (property)

Returns: true or false

The permutation group *G* is *absolutely orbital graph recognisable* (*absolutely OGR*) if and only if every orbital graph of *G* has automorphism group equal to *G*.

Note that every absolutely OGR group is also strongly orbital graph recognisable, see IsStronglyOrbitalGraphRecognisable (1.5.2).

IsAsolutelyOGR is a synonym for IsAbsolutelyOrbitalGraphRecognisable.

Example

```
gap> IsAbsolutelyOrbitalGraphRecognisable(DihedralGroup(IsPermGroup, 8));
true
gap> IsAbsolutelyOGR(CyclicGroup(IsPermGroup, 8));
false
gap> IsAbsolutelyOGR(TrivialGroup(IsPermGroup));
true
```

1.6 Attributes and properties of individual orbital graphs

1.6.1 IsSelfPaired (for IsOrbitalGraph)

▷ IsSelfPaired(*arg*) (property)

Returns: true or false

Chapter 2

Visualising orbital graphs

2.1 Visualing an orbital graph via the DOT format

2.1.1 DotOrbitalGraph (for IsOrbitalGraph)

▷ DotOrbitalGraph(D) (attribute)

Returns: A string

WARNING: This function requires version 1.4.0 of the Digraphs package, or newer.

Given an orbital graph D , this function returns a string that contains a description of D in DOT format.

The edge of D corresponding to its BasePair (1.3.1) is coloured red, while the remaining edges are coloured black.

The DOT string for D can then be compiled and displayed with the Splash (**Digraphs: Splash**) function of the Digraphs package.

See [https://en.wikipedia.org/wiki/DOT_\(graph_description_language\)](https://en.wikipedia.org/wiki/DOT_(graph_description_language)) for more information about this format.

Example

```
gap> G := Group([(1,2,3)]);;
gap> o := OrbitalGraphs(G);;
gap> Print(DotOrbitalGraph(o[1]));
//dot
digraph hgn{
node [shape=circle]
1 [label="1"]
2 [label="2"]
3 [label="3"]
1 -> 2 [color=red]
2 -> 3
3 -> 1
}
gap> Splash(DotOrbitalGraph(o[1])); # To visualise
```

Index

BasePair
 for IsOrbitalGraph, [5](#)

DotOrbitalGraph
 for IsOrbitalGraph, [9](#)

IsAbsolutelyOGR, for IsPermGroup, [8](#)
IsAbsolutelyOrbitalGraphRecognisable
 for IsPermGroup, [8](#)
IsOGR, for IsPermGroup, [7](#)
IsOrbitalGraph
 for IsDigraph, [4](#)
IsOrbitalGraphOfGroup
 for IsOrbitalGraph, [4](#)
IsOrbitalGraphOfSemigroup
 for IsOrbitalGraph, [4](#)
IsOrbitalGraphRecognisable
 for IsPermGroup, [7](#)
IsSelfPaired
 for IsOrbitalGraph, [8](#)
IsStronglyOGR, for IsPermGroup, [7](#)
IsStronglyOrbitalGraphRecognisable
 for IsPermGroup, [7](#)

OrbitalClosure
 for IsPermGroup, [6](#)

OrbitalGraph
 for IsPermGroup, IsHomogeneousList, Is-
 PosInt, [5](#)

OrbitalGraphs
 for IsPermGroup, [4](#)
 for IsPermGroup, IsHomogeneousList, [4](#)
 for IsPermGroup, IsInt, [5](#)
 for IsTransformationSemigroup, [4](#)

OrbitalIndex
 for IsPermGroup, [7](#)

UnderlyingGroup
 for IsOrbitalGraphOfGroup, [6](#)

UnderlyingSemigroup
 for IsOrbitalGraphOfSemigroup, [6](#)