

The G2Comp Package

Equivariant 2-complexes

Version 1.0.2

Iván Sadofschi Costa

Iván Sadofschi Costa Email: isadofschi@dm.uba.ar

Homepage: <http://mate.dm.uba.ar/~isadofschi>

Copyright

© 2018 Iván Sadofski Costa.

Contents

1	G2Comp	4
1.1	Introduction	4
1.2	Components of a G 2-complex	4
1.3	Representation in GAP	5
1.4	Functions to construct G - 2-complexes	5
1.5	Action on the cells	6
1.6	Stabilizers	7
1.7	Orbits	7
1.8	Edges and edge paths	8
1.9	Subcomplex fixed by a subgroup H of G	9
1.10	Complex mod G	10
1.11	Fundamental group	10
1.12	Homotopy properties	11
1.13	Random attaching maps	12
1.14	More	12
1.15	Additional functions	13
	References	14
	Index	15

Chapter 1

G2Comp

1.1 Introduction

This package includes functions to construct, manipulate and study 2-complexes with an admissible action of a finite group G .

1.2 Components of a G 2-complex

1.2.1 GroupOfComplex

▷ `GroupOfComplex(K)` (function)

Returns the group acting on K .

1.2.2 VerticesOfComplex

▷ `VerticesOfComplex(K)` (function)

Returns the set of vertices of K .

1.2.3 EdgesOfComplex

▷ `EdgesOfComplex(K)` (function)

Returns the set of edges of K .

1.2.4 FacesOfComplex

▷ `FacesOfComplex(K)` (function)

Returns the set of faces of K .

1.2.5 LabelsOfComplex

▷ LabelsOfComplex(K) (function)

Returns the set of labels of the orbits of K .

1.3 Representation in GAP

Most of the time it should not be necessary to have in mind how objects are represented in G2Comp. An *oriented edge* is a list $[e, s]$ where e is an edge of K and s is 1 or -1 . An *edge path* is represented as a list of oriented edges (the target of each edge has to be equal to the source of the next edge).

1.4 Functions to construct G - 2-complexes

In this section we describe the main functions of this package.

1.4.1 NewEquivariantTwoComplex

▷ NewEquivariantTwoComplex(G) (function)

Returns an empty G 2-complex.

Example

```
gap> K:=NewEquivariantTwoComplex(AlternatingGroup(5));
[ Alt( [ 1 .. 5 ] ), [ ], [ ], [ ], [ ] ]
```

1.4.2 AddOrbitOfVertices

▷ AddOrbitOfVertices($K, H, label$) (function)

Adds an orbit of vertices to K of type G/H . Returns the new vertex corresponding to the coset $1.H$. A label $label$ for the new orbit must be provided.

Example

```
gap> v0 := AddOrbitOfVertices(K, Group((1,2)(3,4)), "A");
[ (), Group([ (1,2)(3,4) ]), "A" ]
```

1.4.3 AddOrbitOfEdges

▷ AddOrbitOfEdges($K, H, v1, v2, label$) (function)

Adds an orbit of edges to K of type G/H . The vertices $v1, v2$ are the endpoints of the attached edge. Returns the new edge corresponding to the coset $1.H$. A label $label$ for the new orbit must be provided.

Example

```
gap> e0 := AddOrbitOfEdges(K, Group(), v0, v0, "some edges");
[ (), Group(), "some edges", [ (), Group([ (1,2)(3,4) ]), "A" ],
  [ (), Group([ (1,2)(3,4) ]), "A" ] ]
```

1.4.4 AddOrbitOfTwoCells

▷ AddOrbitOfTwoCells(K , H , f , $label$) (function)

Adds an orbit of 2-cells to K with stabilizer H and attaching map f . Returns the new 2-cell corresponding to the coset $1.H$. A label $label$ for the new orbit must be provided.

Example

```
gap> f:=[[e0,1],[e0,1],[e0,-1]];;
gap> AddOrbitOfTwoCells(K,Group(()),f,"some faces");
[ (), Group(()), "some faces",
  [
    [ [ (), Group(()), "some edges", [ (), Group([ (1,2)(3,4) ]), "A" ],
      [ (), Group([ (1,2)(3,4) ]), "A" ] ], 1 ],
    [ [ (), Group(()), "some edges", [ (), Group([ (1,2)(3,4) ]), "A" ],
      [ (), Group([ (1,2)(3,4) ]), "A" ] ], 1 ],
    [ [ (), Group(()), "some edges", [ (), Group([ (1,2)(3,4) ]), "A" ],
      [ (), Group([ (1,2)(3,4) ]), "A" ] ], -1 ] ] ]
```

1.5 Action on the cells

1.5.1 ActionVertex

▷ ActionVertex(g , v) (function)

Returns the vertex $g.v$.

1.5.2 ActionEdge

▷ ActionEdge(g , e) (function)

Returns the edge $g.e$.

1.5.3 ActionOrientedEdge

▷ ActionOrientedEdge(g , e) (function)

Returns the oriented edge $g.e$.

1.5.4 ActionEdgePath

▷ ActionEdgePath(g , c) (function)

Returns the edge path $g.c$.

1.5.5 ActionTwoCell

▷ ActionTwoCell(g , f) (function)

Returns the 2-cell $g.f$.

1.6 Stabilizers

1.6.1 StabilizerVertex

▷ `StabilizerVertex(v)` (function)

Returns the stabilizer of v .

1.6.2 StabilizerEdge

▷ `StabilizerEdge(e)` (function)

Returns the stabilizer of e .

1.6.3 StabilizerOrientedEdge

▷ `StabilizerOrientedEdge(e)` (function)

Returns the stabilizer of e .

1.6.4 StabilizerEdgePath

▷ `StabilizerEdgePath(f)` (function)

Returns stabilizer of f .

1.6.5 StabilizerTwoCell

▷ `StabilizerTwoCell(f)` (function)

Returns the stabilizer of f .

1.6.6 StabilizerCell

▷ `StabilizerCell(e)` (function)

Returns the stabilizer of a k -cell e .

1.7 Orbits

1.7.1 OrbitOfVertex

▷ `OrbitOfVertex(H , v)` (function)

Returns the orbit of the vertex v by the subgroup H .

1.7.2 OrbitOfEdge

▷ `OrbitOfEdge(H , e)` (function)

Returns the orbit of the edge e by the subgroup H .

1.7.3 OrbitOfOrientedEdge

▷ `OrbitOfOrientedEdge(H , e)` (function)

Returns the orbit of the oriented edge e by the subgroup H .

1.7.4 OrbitOfEdgePath

▷ `OrbitOfEdgePath(H , f)` (function)

Returns the orbit of the edge path f by the subgroup H .

1.7.5 OrbitOfTwoCell

▷ `OrbitOfTwoCell(H , f)` (function)

Returns the orbit of the 2-cell f by the subgroup H .

1.8 Edges and edge paths

1.8.1 VerticesOfEdge

▷ `VerticesOfEdge(e)` (function)

Returns the set of vertices of the edge e .

1.8.2 SourceOrientedEdge

▷ `SourceOrientedEdge(e)` (function)

Returns the source of the oriented edge e .

1.8.3 TargetOrientedEdge

▷ `TargetOrientedEdge(e)` (function)

Returns the target of the oriented edge e .

1.8.4 VerticesOrientedEdge

▷ `VerticesOrientedEdge(e)` (function)

Returns a list with the source and target of the oriented edge e (in this order).

1.8.5 OppositeEdge

▷ `OppositeEdge(e)` (function)

Returns the opposite edge of an oriented edge e .

1.8.6 IsEdgePath

▷ `IsEdgePath(c)` (function)

Checks if a list of edges c is an edge path.

1.8.7 IsClosedEdgePath

▷ `IsClosedEdgePath(c)` (function)

Checks if a list of edges c is a closed edge path.

1.8.8 InverseEdgePath

▷ `InverseEdgePath(c)` (function)

Returns the inverse edge path of an edge path c .

1.8.9 ReducedEdgePath

▷ `ReducedEdgePath(f)` (function)

Reduces the edge path f .

1.8.10 CyclicallyReducedEdgePath

▷ `CyclicallyReducedEdgePath(f)` (function)

Cyclically reduces the edge path f .

1.9 Subcomplex fixed by a subgroup H of G

1.9.1 FixedSubcomplex

▷ `FixedSubcomplex(K, H)` (function)

Returns the subcomplex K^H of K fixed by H . This is represented as a 2-complex with an action of the trivial group (this function may be improved in a future version to yield a complex with an action of $N_G(H)$ or $N_G(H)/H$).

1.10 Complex mod G

These functions allow to work with the complex K/G .

1.10.1 TwoComplexModG

▷ `TwoComplexModG(K)` (function)

Returns the complex K/G . This is represented as a 2-complex with an action of the trivial group.

Example

```
gap> KmodG:=TwoComplexModG(K);
```

1.10.2 VertexModG

▷ `VertexModG(v)` (function)

1.10.3 EdgeModG

▷ `EdgeModG(e)` (function)

1.10.4 DirectedEdgeModG

▷ `DirectedEdgeModG(e)` (function)

1.10.5 EdgePathModG

▷ `EdgePathModG(c)` (function)

1.10.6 TwoCellModG

▷ `TwoCellModG(c)` (function)

1.11 Fundamental group

1.11.1 Pi1

▷ `Pi1(K[, T])` (function)

Returns the fundamental group of a connected complex K . Optionally, to compute the fundamental group a specific spanning tree T may be provided. Returns *fail* if K is not connected.

1.11.2 ElementOfPi1FromClosedEdgePath

▷ `ElementOfPi1FromClosedEdgePath(K , c)` (function)

Returns the element of the fundamental group of K representing the class of the closed edge path c .

1.11.3 Pi1RandomSpanningTree

▷ `Pi1RandomSpanningTree(K)` (function)

Returns the fundamental group of a connected complex K , computed using a random spanning tree of K . Returns *fail* if K is not connected.

1.11.4 Pi1XModX0

▷ `Pi1XModX0(K)` (function)

Returns the fundamental group of K/K^0 .

1.11.5 SpanningTreeOfComplex

▷ `SpanningTreeOfComplex(K)` (function)

Returns a spanning tree for the 1-skeleton of K .

1.11.6 RandomSpanningTreeOfComplex

▷ `RandomSpanningTreeOfComplex(K)` (function)

Returns a spanning tree for the 1-skeleton of K chosen randomly.

1.11.7 IsSpanningTreeOfComplex

▷ `IsSpanningTreeOfComplex(K , T)` (function)

Returns true if T is a spanning tree of K , false otherwise.

1.12 Homotopy properties

1.12.1 IsEmptyComplex

▷ `IsEmptyComplex(K)` (function)

Returns true if K is empty, false otherwise.

1.12.2 IsConnected

▷ IsConnected(K) (function)

Returns true if K is connected, false otherwise.

1.12.3 IsAcyclic

▷ IsAcyclic(K) (function)

Returns true if K is acyclic, false otherwise.

1.12.4 IsAsphericalComplex

▷ IsAsphericalComplex(K) (function)

If it returns true, then K is aspherical. It may return fail. Uses the function IsAspherical from the package HAP.

1.12.5 IsContractible

▷ IsContractible(K , $time_limit$) (function)

If it returns true, then K is contractible. It may return fail. The optional argument $time_limit$ allows to set a time limit for the computation of the fundamental group.

1.13 Random attaching maps

1.13.1 RandomAttachingMaps

▷ RandomAttachingMaps(K , $lengths$) (function)

Returns a list of randomly chosen closed edge paths in K of the specified lengths $lengths$.

1.14 More

1.14.1 H2AsGModule

▷ H2AsGModule(K) (function)

Returns the representation of the group G given by the action on $H_2(K)$. It is represented as a morphism $G \rightarrow GL(m, \mathbb{Z})$ where m is the rank of $H_2(K)$.

1.14.2 CoveringSpaceFromHomomorphism

▷ CoveringSpaceFromHomomorphism(H , G , phi) (function)

If H is finitely presented, G is finite and ϕ is an epimorphism $f: H \rightarrow G$, returns the covering space of the presentation complex of H corresponding to the subgroup $\text{Kernel}(\phi)$ of H . This covering space is represented as a G 2-complex.

1.15 Additional functions

1.15.1 CanonicalLeftCosetElement

▷ `CanonicalLeftCosetElement(g, H)` (function)

Returns a "canonical" representative of the left coset gH which is independent of the given representative g . This can be used to compare cosets by comparing their canonical representatives. The representative chosen to be the "canonical" one is representation dependent and only guaranteed to remain the same within one GAP session. See also `CanonicalRightCosetElement` (**Reference: CanonicalRightCosetElement**)

Example

```
gap> CanonicalLeftCosetElement((2,3,5),H);
(3,5)
```

1.15.2 Epimorphism

▷ `Epimorphism(H, G)` (function)

Returns true if there is an epimorphism from the FpGroup H to the finite group G . Otherwise returns false.

1.15.3 PresentationLength

▷ `PresentationLength(P)` (function)

Returns the length of the presentation P .

References

Index

ActionEdge, 6
ActionEdgePath, 6
ActionOrientedEdge, 6
ActionTwoCell, 6
ActionVertex, 6
AddOrbitOfEdges, 5
AddOrbitOfTwoCells, 6
AddOrbitOfVertices, 5

CanonicalLeftCosetElement, 13
CoveringSpaceFromHomomorphism, 12
CyclicallyReducedEdgePath, 9

DirectedEdgeModG, 10

EdgeModG, 10
EdgePathModG, 10
EdgesOfComplex, 4
ElementOfPi1FromClosedEdgePath, 11
Epimorphism, 13

FacesOfComplex, 4
FixedSubcomplex, 9

GroupOfComplex, 4

H2AsGModule, 12

InverseEdgePath, 9
IsAcyclic, 12
IsAsphericalComplex, 12
IsClosedEdgePath, 9
IsConnected, 12
IsContractible, 12
IsEdgePath, 9
IsEmptyComplex, 11
IsSpanningTreeOfComplex, 11

LabelsOfComplex, 5

NewEquivariantTwoComplex, 5

OppositeEdge, 9
OrbitOfEdge, 8
OrbitOfEdgePath, 8
OrbitOfOrientedEdge, 8
OrbitOfTwoCell, 8
OrbitOfVertex, 7

Pi1, 10
Pi1RandomSpanningTree, 11
Pi1XModX0, 11
PresentationLength, 13

RandomAttachingMaps, 12
RandomSpanningTreeOfComplex, 11
ReducedEdgePath, 9

SourceOrientedEdge, 8
SpanningTreeOfComplex, 11
StabilizerCell, 7
StabilizerEdge, 7
StabilizerEdgePath, 7
StabilizerOrientedEdge, 7
StabilizerTwoCell, 7
StabilizerVertex, 7

TargetOrientedEdge, 8
TwoCellModG, 10
TwoComplexModG, 10

VertexModG, 10
VerticesOfComplex, 4
VerticesOfEdge, 8
VerticesOrientedEdge, 8