The G2Comp Package

Version 1.0.1

Iván Sadofschi Costa

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

Copyright

Contents

Chapter 1

G 2-complexes

1.1 Representation in GAP

A G-2-complex is represented as a tuple [G, V, E, F, labels] # each element r in F is a list of pairs [edge, sign] # that represents a closed edge path used to attach a 2-cell

1.2 Functions to construct G - 2-complexes

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

1.2.1 NewEquivariantTwoComplex

▷ `NewEquivariantTwoComplex(G)` (function)

Returns an empty *G* 2-complex.

Example

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

1.2.2 AddOrbitOfVertices

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

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

Example

```
gap> AddOrbitOfVertices(K, Group((1,2)(3,4)), "A");
```

1.2.3 AddOrbitOfEdges

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

Adds an orbit of edges to *K* with stabilizer *H*. The vertices *v1,v2* are the endpoints of the attached edge. A label *label* for the new orbit must be provided.

Example

1.2.4 AddOrbitOfTwoCells

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

Adds an orbit of 2-cells to `K` with stabilizer `H` and attaching map `f`. A label `label` for the new orbit must be provided.

Example

1.3 Components of a G 2-complex

1.3.1 GroupOfComplex

▷ `GroupOfComplex(K)` (function)

Returns the group acting on `K`.

1.3.2 VerticesOfComplex

▷ `VerticesOfComplex(K)` (function)

Returns the set of vertices of `K`.

1.3.3 EdgesOfComplex

▷ `EdgesOfComplex(K)` (function)

Returns the set of edges of `K`.

1.3.4 FacesOfComplex

▷ `FacesOfComplex(K)` (function)

Returns the set of faces of `K`.

1.3.5 LabelsOfComplex

▷ `LabelsOfComplex(K)` (function)

Returns the set of labels of `K`.

1.4 Action on the cells

1.4.1 ActionVertex

▷ `ActionVertex(g, v)` (function)

Returns the vertex `g.v`.

1.4.2 ActionEdge

▷ `ActionEdge(g, e)` (function)

Returns the edge `g.e`.

1.4.3 ActionOrientedEdge

▷ `ActionOrientedEdge(g, e)` (function)

Returns the oriented edge `g.e`.

1.4.4 ActionEdgePath

▷ `ActionEdgePath(g, c)` (function)

Returns the edge path `g.c`.

1.4.5 ActionTwoCell

▷ `ActionTwoCell(g, f)` (function)

Returns the 2-cell `g.f`.

1.5 Stabilizers

1.5.1 StabilizerVertex

▷ `StabilizerVertex(v)` (function)

Returns the stabilizer of `v`.

1.5.2 StabilizerEdge

▷ `StabilizerEdge(e)` (function)

Returns the stabilizer of `e`.

1.5.3 StabilizerOrientedEdge

▷ `StabilizerOrientedEdge(e)` (function)

Returns the stabilizer of `e`.

1.5.4 StabilizerEdgePath

▷ `StabilizerEdgePath(g, f)` (function)

Returns the 2-cell `g.f`.

1.5.5 StabilizerTwoCell

▷ `StabilizerTwoCell(f)` (function)

Returns the stabilizer of `f`.

1.5.6 StabilizerCell

▷ `StabilizerCell(e)` (function)

Returns the stabilizer of a k-cell `e`.

1.6 Edges and edge paths

1.6.1 VerticesOfEdge

▷ `VerticesOfEdge(e)` (function)

Returns the set of vertices of the edge `e`.

1.6.2 SourceOrientedEdge

▷ `SourceOrientedEdge(e)` (function)

Returns the source of the oriented edge `e`.

1.6.3 TargetOrientedEdge

▷ `TargetOrientedEdge(e)` (function)

Returns the target of the oriented edge `e`.

1.6.4 VerticesOrientedEdge

▷ `VerticesOrientedEdge(e)` (function)

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

1.6.5 OppositeEdge

▷ `OppositeEdge(e)` (function)

Returns the opposite edge of an oriented edge `e`.

1.6.6 IsEdgePath

▷ `IsEdgePath(c)` (function)

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

1.6.7 IsClosedEdgePath

▷ `IsClosedEdgePath(c)` (function)

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

1.6.8 InverseEdgePath

▷ `InverseEdgePath(c)` (function)

Returns the inverse edge path of an edge path `c`.

1.6.9 ReducedEdgePath

▷ `ReducedEdgePath(c)` (function)

Reduces the edge path `c` (destructive).

1.6.10 CyclicallyReducedEdgePath

▷ `CyclicallyReducedEdgePath(c)` (function)

Cyclically reduces the edge path `c` (destructive).

1.7 Complex mod G

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

1.7.1 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);
```

1.7.2 VertexModG

▷ VertexModG(*v*) (function)

1.7.3 EdgeModG

▷ EdgeModG(*e*) (function)

1.7.4 DirectedEdgeModG

▷ DirectedEdgeModG(*e*) (function)

1.7.5 EdgePathModG

▷ EdgePathModG(*c*) (function)

1.8 Fundamental group

1.8.1 Pi1

▷ Pi1(*K*) (function)

Returns the fundamental group of a connected complex *K*. Returns *fail* if *K* is not connected.

1.8.2 ElementOfPi1FromClosedEdgePath

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

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

1.8.3 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.

1.8.4 Pi1XModX0

▷ Pi1XModX0(*K*) (function)

Returns the fundamental group of *K/K^0*.

1.8.5 SpanningTreeOfComplex

▷ `SpanningTreeOfComplex(K)` (function)

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

1.8.6 RandomSpanningTreeOfComplex

▷ `RandomSpanningTreeOfComplex(K)` (function)

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

1.9 Homotopy properties

1.9.1 IsAcyclic

▷ `IsAcyclic(K)` (function)

Returns `true` if K is acyclic, `false` otherwise.

1.9.2 IsAsphericalComplex

▷ `IsAsphericalComplex(K)` (function)

Uses the function `IsAspherical` from the package HAP. Returns `true` if K is aspherical. It may return `fail`.

1.9.3 IsContractible

▷ `IsContractible(K[, time_limit])` (function)

Returns `true` if 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.

1.10 Random attaching maps

1.10.1 RandomAttachingMaps

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

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

1.11 More

1.11.1 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 -> GL(m,Z)` where `m` is the rank of `H_2(K)`.

1.11.2 CoveringSpaceFromHomomorphism

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

If `H` is finitely presented, `G` is finite and `phi` is an epimorphism `f: H -> G`, returns the covering space of the presentation complex of `H` corresponding to the subgroup `Kernel(phi)` of `H`. This covering space is represented as a `G` 2-complex.

1.12 Additional functions

1.12.1 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.

Example

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

1.12.2 Epimorphism

▷ `Epimorphism(P, G)` (function)

Returns `true` if there is an epimorphism from the group given by the presentation `P` to the finite group `G`. Otherwise returns `false`.

1.12.3 PresentationLength

▷ `PresentationLength(P)` (function)

Returns the length of the presentation `P`.

References

Index