RAMP

The Research Assistant for Maniplexes and Polytopes

0.3

28 July 2020

Gabe Cunningham

Gabe Cunningham
Email: gabriel.cunningham@umb.edu
Homepage: http://www.gabrielcunningham.com
Address: Gabe Cunningham
Department of Mathematics
University of Massachusetts Boston
100 William T. Morrissey Blvd.
Boston MA 02125

Copyright

Acknowledgements

We appreciate very much all past and future comments, suggestions and contributions to this package and its documentation provided by GAP users and developers.

Contents

Chapter 1

Constructions

1.1 Extensions, amalgamations, and quotients

1.1.1 UniversalPolytope (for IsInt)

▷ `UniversalPolytope(n)` (operation)

Returns the universal polytope of rank `n`.

1.1.2 FlatRegularPolyhedron (for IsInt, IsInt, IsInt, IsInt)

▷ `FlatRegularPolyhedron(p, q, i, j)` (operation)

Returns the flat regular polyhedron with automorphism group [p, q] / (r2 r1 r0 r1 = (r0 r1)^i (r1 r2)^j). This function does not currently validate the inputs to make sure that the output makes sense.

1.1.3 QuotientPolytope (for IsManiplex, IsList)

▷ `QuotientPolytope(M, rels)` (operation)

Returns the quotient of `M` by `rels`, which may be given as either a list of Tietze words, such as [[1,2,1,0,1,2,1,0]] or as a string like "(r0 r1 r2 r1)^2, (r0 r1 r2)^4".

1.1.4 UniversalExtension (for IsManiplex)

▷ `UniversalExtension(M)` (operation)

Returns the universal extension of `M`, i.e. the maniplex with facets isomorphic to `M` that covers all other maniplexes with facets isomorphic to `M`. Currently only defined for reflexible maniplexes.

1.1.5 UniversalExtension (for IsManiplex, IsInt)

▷ `UniversalExtension(M, k)` (operation)

Returns the universal extension of `M` with last entry of Schlafli symbol `k`. Currently only defined for reflexible maniplexes.

1.1.6 TrivialExtension (for IsManiplex)

▷ `TrivialExtension(M)` (operation)

Returns the trivial extension of `M`, also known as `{M/, 2}`.

1.1.7 FlatExtension (for IsManiplex, IsInt)

▷ `FlatExtension(M, k)` (operation)

Returns the flat extension of `M` with last entry of Schlafli symbol `k`. (As defined in "Flat Extensions of Abstract Polytopes".) Currently only defined for reflexible maniplexes.

1.1.8 Amalgamate (for IsManiplex, IsManiplex)

▷ `Amalgamate(M1, M2)` (operation)

Returns the amalgamation of `M1` and `M2`. Implicitly assumes that `M1` and `M2` are compatible. Currently only defined for reflexible maniplexes.

1.1.9 Medial (for IsManiplex)

▷ `Medial(M)` (operation)

Given a 3-maniplex `M`, returns its medial.

1.2 Duality

1.2.1 Dual (for IsManiplex)

▷ `Dual(M)` (attribute)

Returns: The maniplex that is dual to `M`.

1.2.2 IsSelfDual (for IsManiplex)

▷ `IsSelfDual(P)` (property)

Returns: Whether this polytope is isomorphic to its dual.

1.2.3 Petrial (for IsManiplex)

▷ `Petrial(P)` (attribute)

Returns: The Petrial (Petrie dual) of `P`. Note that this is not necessarily a polytope.

1.2.4 IsSelfPetrial (for IsManiplex)

▷ `IsSelfPetrial(P)` (property)

Returns: Whether this polytope is isomorphic to its Petrial.

Chapter 2

Databases

2.1 Regular polyhedra

2.1.1 DegeneratePolyhedra (for IsInt)

▷ `DegeneratePolyhedra(`*`maxsize`*`)` (operation)

Returns all degenerate polyhedra (of type {2, q} and {p, 2}) with up to *maxsize* flags.

2.1.2 FlatRegularPolyhedra (for IsInt)

▷ `FlatRegularPolyhedra(`*`maxsize`*`)` (operation)

Returns all nondegenerate flat regular polyhedra with up to *maxsize* flags. Currently supports a maxsize of 4000 or less.

2.1.3 SmallRegularPolyhedra (for IsInt)

▷ `SmallRegularPolyhedra(`*`maxsize`*`)` (operation)

Returns all regular polyhedra with up to *maxsize* flags. Currently supports a maxsize of 4000 or less. You can also set options "nondegenerate" and "nonflat".

Example

```
L1 := SmallRegularPolyhedra(500);;
L2 := SmallRegularPolyhedra(1000 : nondegenerate);;
L3 := SmallRegularPolyhedra(2000 : nondegenerate, nonflat);;
```

Chapter 3

Combinatorics and Structure

3.1 Faces

3.1.1 NumberOfIFaces (for IsManiplex, IsInt)

▷ `NumberOfIFaces(M, i)` (operation)

Returns The number of i-faces of M.

3.1.2 NumberOfVertices (for IsManiplex)

▷ `NumberOfVertices(M)` (attribute)

Returns the number of vertices of M.

3.1.3 NumberOfEdges (for IsManiplex)

▷ `NumberOfEdges(M)` (attribute)

Returns the number of edges of M.

3.1.4 NumberOfFacets (for IsManiplex)

▷ `NumberOfFacets(M)` (attribute)

Returns the number of facets of M.

3.1.5 NumberOfRidges (for IsManiplex)

▷ `NumberOfRidges(M)` (attribute)

Returns the number of ridges ((n-2)-faces) of M.

3.1.6 Fvector (for IsManiplex)

▷ `Fvector(M)` (attribute)

Returns the f-vector of *M*.

3.1.7 Facets (for IsManiplex)

▷ `Facets(M)` (attribute)

Returns the facet-types of *M* (i.e. the maniplexes corresponding to the facets). Currently only works for reflexible maniplexes.

3.1.8 VertexFigures (for IsManiplex)

▷ `VertexFigures(M)` (attribute)

Returns the types of vertex-figures of *M* (i.e. the maniplexes corresponding to the vertex-figures). Currently only works for reflexible maniplexes.

3.2 Posets

3.2.1 PosetFromFaceListOfFlags (for IsList)

▷ `PosetFromFaceListOfFlags(list)` (operation)

Returns: IsPosetOfFlags. Not that the function is INTENTIONALLY agnostic about whether it is being given full poset or not. Given a *list* of lists of faces in increasing rank, where each face is described by the incident flags, gives you a IsPosetOfFlags object back.

3.2.2 IsFull (for IsPoset)

▷ `IsFull(poset)` (attribute)

Returns: truth value

Checks or creates the value of the attribute IsFull for an IsPoset.

3.2.3 IsFlaggable (for IsPoset)

▷ `IsFlaggable(arg)` (attribute)

3.2.4 IsFlaggablePoset (for IsPosetOfFlags)

▷ `IsFlaggablePoset(poset)` (operation)

Given a *poset* (whose elements are lists of flags) corresponding to a maniplex, this function will tell you if it is flaggable, i.e., if the flags can be recovered from the poset or not.

3.2.5 ListIsFullPoset (for IsList)

▷ `ListIsFullPoset(list)` (operation)

Given *list*, a poset as a list of faces ordered by rank, each face listing the flags on the face, this function will tell you if the poset is full or not.

3.2.6 RankOfPoset (for IsPoset)

▷ `RankOfPoset(poset)` (operation)

Given a *poset*, returns the rank of the poset. Note: There may be hidden assumptions here to untangle later.

3.2.7 IsNotFull (for IsPoset)

▷ `IsNotFull(poset)` (operation)

Lets me check to see if a poset is NOT full, for certain filtering operations.

3.2.8 PosetOfConnectionGroup (for IsPermGroup)

▷ `PosetOfConnectionGroup(g)` (operation)

Given a group, returns a poset as a list of faces ordered by rank, where each face is represented as a list of the flags it contains. Note that this function does not include the minimal (empty) face nor the maximal face of the maniplex. Note that the *i*-faces correspond to the *i+1* item in the list because of how GAP indexes lists.

3.2.9 FullPosetOfConnectionGroup (for IsPermGroup)

▷ `FullPosetOfConnectionGroup(g)` (operation)

Returns a full poset as a list of faces ordered by rank, where each face is represented as a list of the flags it contains. This function does include the minimal (empty) face nor the maximal face of the maniplex, so the list has *n+2* ranks if the maniplex is of rank *n*. Note that the *i*-faces correspond to the *i+1* item in the list because of how GAP indexes lists.

3.2.10 PosetOfManiplex (for IsManiplex)

▷ `PosetOfManiplex(mani)` (operation)

Given a maniplex, returns a poset as a list of faces ordered by rank, where each face is represented as a list of the flags it contains. Note that this function does not include the minimal (empty) face nor the maximal face of the maniplex. Note that the *i*-faces correspond to the *i+1* item in the list because of how GAP indexes lists.

3.2.11 FullPosetOfManiplex (for IsManiplex)

▷ `FullPosetOfManiplex(mani)` (operation)

Given a maniplex, returns a poset as a list of faces ordered by rank, where each face is represented as a list of the flags it contains. Note that this function does include the minimal (empty) face and the maximal face of the maniplex. Note that the `i`-faces correspond to the `i+1` item in the list because of how GAP indexes lists.

3.2.12 AreIncidentFaces (for IsObject,IsObject)

▷ `AreIncidentFaces(object1, object2)` (operation)

Given two faces, will tell you if they are incident. Currently only supports faces as list of their incident flags.

3.2.13 FlagsAsListOfFacesFromPoset (for IsPoset)

▷ `FlagsAsListOfFacesFromPoset(poset)` (operation)

Given a `poset`, this will give you a version of the list of flags in terms of the faces described in the `poset`. Note that the flag list does not include the empty face or the maximal face.

3.2.14 AdjacentFlag (for IsList,IsPosetOfFlags,IsInt)

▷ `AdjacentFlag(flag, poset, i)` (operation)

Given a flag (represented as chains of faces comprised of lists of flags) and a poset and a rank, this function will give you the `i`-adjacent flag. Note that adjacencies are listed from ranks 0 to one less than the dimension. You can replace `flag` with the integer corresponding to that flag.

3.2.15 ConnectionGeneratorOfPoset (for IsPoset,IsInt)

▷ `ConnectionGeneratorOfPoset(poset, i)` (operation)

Given a `poset` and an integer `i`, this function will give you the associated permutation for the rank `i`-connection.

3.2.16 ConnectionGroupOfPoset (for IsPoset)

▷ `ConnectionGroupOfPoset(poset)` (operation)

Given a `poset` corresponding to a maniplex, this function will give you the connection group.

Chapter 4

Families of Polytopes

4.1 Classical Polytopes

4.1.1 Vertex

▷ `Vertex()` (operation)

4.1.2 Edge

▷ `Edge()` (operation)

4.1.3 Pgon (for IsInt)

▷ `Pgon(`*p*`)` (operation)

4.1.4 Cube (for IsInt)

▷ `Cube(`*n*`)` (operation)

4.1.5 HemiCube (for IsInt)

▷ `HemiCube(`*n*`)` (operation)

4.1.6 CrossPolytope (for IsInt)

▷ `CrossPolytope(`*n*`)` (operation)

4.1.7 HemiCrossPolytope (for IsInt)

▷ `HemiCrossPolytope(`*n*`)` (operation)

4.1.8 Simplex (for IsInt)

▷ `Simplex(`*n*`)` (operation)

4.1.9 CubicTiling (for IsInt)

▷ `CubicTiling(`*n*`)` (operation)

4.1.10 Dodecahedron

▷ `Dodecahedron()` (operation)

4.1.11 HemiDodecahedron

▷ `HemiDodecahedron()` (operation)

4.1.12 Icosahedron

▷ `Icosahedron()` (operation)

4.1.13 HemiIcosahedron

▷ `HemiIcosahedron()` (operation)

4.1.14 24Cell

▷ `24Cell()` (operation)

4.1.15 Hemi24Cell

▷ `Hemi24Cell()` (operation)

4.1.16 120Cell

▷ `120Cell()` (operation)

4.1.17 Hemi120Cell

▷ `Hemi120Cell()` (operation)

4.1.18 600Cell

▷ `600Cell()` (operation)

4.1.19 Hemi600Cell

▷ `Hemi600Cell()` (operation)

Chapter 5

Properties

5.1 Orientability

5.1.1 IsOrientable (for IsManiplex)

▷ `IsOrientable(p)` (property)

Returns: `true` or `false`

A polytope is orientable if its flag graph is bipartite. Currently only implemented for regular polytopes.

5.1.2 IsIOrientable (for IsManiplex, IsList)

▷ `IsIOrientable(p, I)` (operation)

For a subset I of {0, ..., n-1}, a polytope if I-orientable if every closed path in its flag graph contains an even number of edges with colors in I. Currently only implemented for regular polytopes.

5.1.3 IsVertexBipartite (for IsManiplex)

▷ `IsVertexBipartite(p)` (property)

Returns: `true` or `false`

A polytope is vertex-bipartite if its 1-skeleton is bipartite. This is equivalent to being I-orientable for I = {0}.

5.1.4 IsFacetBipartite (for IsManiplex)

▷ `IsFacetBipartite(p)` (property)

Returns: `true` or `false`

A polytope is facet-bipartite if the 1-skeleton of its dual is bipartite. This is equivalent to being I-orientable for I = {n-1}.

Chapter 6

Basics

6.1 Constructors

6.1.1 UniversalSggi

▷ `UniversalSggi(`*`n`*`)` (operation)
▷ `UniversalSggi(`*`sym`*`)` (operation)

In the first form, returns the universal Coxeter Group of rank n. In the second form, returns the Coxeter Group with Schlafli symbol sym.

6.1.2 ReflexibleManiplex (for IsGroup)

▷ `ReflexibleManiplex(`*`g`*`)` (operation)

Given a group g (which should be a string C-group), returns the abstract regular polytope with that automorphism group, where the privileged generators are those returned by GeneratorsOfGroup(g).

6.1.3 ReflexibleManiplex (for IsList, IsList)

▷ `ReflexibleManiplex(`*`symbol, relations`*`)` (operation)

Returns an abstract regular polytope with the given Schlafli symbol and with the given relations. The formatting of the relations is quite flexible. All of the following work:

Example

```
q := ReflexibleManiplex([4,3,4], "(r0 r1 r2)^3, (r1 r2 r3)^3");
q := ReflexibleManiplex([4,3,4], "(r0 r1 r2)^3 = (r1 r2 r3)^3 = 1");
p := ReflexibleManiplex([infinity], "r0 r1 r0 = r1 r0 r1");
```

6.1.4 ReflexibleManiplex (for IsString)

▷ `ReflexibleManiplex(`*`name`*`)` (operation)

Returns the regular polytope with the given symbolic name. Examples: ReflexibleManiplex("{3,3,3}"); ReflexibleManiplex("{4,3}_3");

Chapter 7

ramp automatic generated documentation

7.1 ramp automatic generated documentation of methods

7.1.1 PyramidOver (for IsManiplex)

▷ `PyramidOver(`*M*`)` (operation)

Returns the pyramid over *M*. Currently only works for finite maniplexes.

7.1.2 PrismOver (for IsManiplex)

▷ `PrismOver(`*M*`)` (operation)

Returns the prism over *M*. Currently only works for finite maniplexes.

Index