RAMP

The Research Assistant for Maniplexes and Polytopes

0.3

28 July 2020

Gabe Cunningham

Mark Mixer

Gordon Williams

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

Mark Mixer
Email: mixerm@wit.edu

Gordon Williams
Email: giwilliams@alaska.edu

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.

1.3 Products

1.3.1 PyramidOver (for IsManiplex)

▷ `PyramidOver(M)` (operation)

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

1.3.2 PrismOver (for IsManiplex)

▷ `PrismOver(M)` (operation)

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

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 Schlafli symbol

3.2.1 SchlafliSymbol (for IsManiplex)

▷ `SchlafliSymbol(M)` (attribute)

Returns the Schlafli symbol of the maniplex *M*. Each entry is either an integer or a set of integers, where entry number i shows the polygons that we obtain as sections of (i+1)-faces over (i-2)-faces.

3.2.2 IsEquivelar (for IsManiplex)

▷ `IsEquivelar(M)` (property)

Returns: the the maniplex *M* is equivelar; i.e., whether its Schlafli Symbol consists of integers at each position (no lists).

3.3 Basics

3.3.1 Size (for IsManiplex)

▷ `Size(M)` (attribute)

Returns the number of flags of the maniplex *M*.

3.3.2 RankManiplex (for IsManiplex)

▷ `RankManiplex(M)` (attribute)

Returns the rank of the maniplex *M*.

3.3.3 IsTight (for IsManiplex and IsPolytopal)

▷ `IsTight(P)` (property)

Returns: `true` or `false`

Returns whether the polytope P is tight, meaning that it has a Schlafli symbol {k_1, ..., k_{n-1}} and has 2 k_1 ... k_{n-1} flags, which is the minimum possible. This property doesn't make any sense for non-polytopal maniplexes, which aren't constrained by this lower bound.

3.3.4 IsDegenerate (for IsManiplex)

▷ `IsDegenerate(M)` (property)

Returns: `true` or `false`

Returns whether the maniplex M has any sections that are digons. We may eventually want to include maniplexes with even smaller sections.

3.3.5 SymmetryTypeGraph (for IsManiplex)

▷ `SymmetryTypeGraph(M)` (attribute)

Returns the Symmetry Type Graph of the maniplex M, encoded as a permutation group on Rank(M) generators.

3.3.6 NumberOfFlagOrbits (for IsManiplex)

▷ `NumberOfFlagOrbits(M)` (attribute)

Returns the number of orbits of the automorphism group of M on its flags.

3.3.7 IsReflexible (for IsManiplex)

▷ `IsReflexible(M)` (property)

Returns: Whether the maniplex M is reflexible (has one flag orbit).

3.3.8 IsChiral (for IsManiplex)

▷ `IsChiral(M)` (property)

Returns: Whether the maniplex M is chrial.

3.3.9 IsRotary (for IsManiplex)

▷ `IsRotary(M)` (property)

Returns: Whether the maniplex M is rotary; i.e., whether it is either reflexible or chiral.

3.3.10 Description (for IsManiplex)

▷ `Description(M)` (attribute)

Returns a short name for the maniplex M, if one is available. For example, Description(Simplex(3)) = "3-simplex".

3.4 Zigzags and holes

3.4.1 ZigzagLength (for IsManiplex, IsInt)

▷ `ZigzagLength(M, j)` (operation)

Returns: The lengths of `j`-zigzags of the 3-maniplex `M`. This corresponds to the lengths of orbits under r0 (r1 r2)^j.

3.4.2 ZigzagVector (for IsManiplex)

▷ `ZigzagVector(M)` (attribute)

Returns: The lengths of all zigzags of the 3-maniplex `M`. A rank 3 maniplex of type {p, q} has Floor(q/2) distinct zigzag lengths because the j-zigzags are the same as the (q-j)-zigzags.

3.4.3 PetrieLength (for IsManiplex)

▷ `PetrieLength(M)` (attribute)

Returns: The length of the petrie polygons of the maniplex `M`.

3.4.4 HoleLength (for IsReflexibleManiplex)

▷ `HoleLength(M)` (attribute)

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

Groups

5.1 Groups

5.1.1 AutomorphismGroup (for IsManiplex)

▷ `AutomorphismGroup(M)` (attribute)

Returns the automorphism group of `M`. This group is not guaranteed to be in any particular form.

5.1.2 AutomorphismGroupFpGroup (for IsManiplex)

▷ `AutomorphismGroupFpGroup(M)` (attribute)

Returns the automorphism group of `M` as a finitely presented group.

5.1.3 AutomorphismGroupPermGroup (for IsManiplex)

▷ `AutomorphismGroupPermGroup(M)` (attribute)

Returns the automorphism group of `M` as a permutation group.

5.1.4 ConnectionGroup (for IsManiplex)

▷ `ConnectionGroup(M)` (attribute)

Returns the connection group of `M` as a permutation group. We may eventually allow other types of connection groups.

5.1.5 EvenConnectionGroup (for IsManiplex)

▷ `EvenConnectionGroup(M)` (attribute)

Returns the even-word subgroup of the connection group of `M` as a permutation group.

5.1.6 RotationGroup (for IsManiplex)

▷ `RotationGroup(M)` (attribute)

Returns the rotation group of `M`. This group is not guaranteed to be in any particular form.

5.1.7 ExtraRelators (for IsReflexibleManiplex)

▷ `ExtraRelators(M)` (attribute)

For a reflexible maniplex `M`, returns the relators needed to define its automorphism group as a quotient of the string Coxeter group given by its Schlafli symbol. Not particularly robust at the moment.

5.1.8 IsStringC (for IsGroup)

▷ `IsStringC(G)` (operation)

For an sggi `G`, returns whether the group is a string C group. It does not check whether `G` is an sggi.

Chapter 6

Properties

6.1 Orientability

6.1.1 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.

6.1.2 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.

6.1.3 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}.

6.1.4 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 7

Basics

7.1 Constructors

7.1.1 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.

7.1.2 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).

7.1.3 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");
```

If the option set_schlafli is set, then we set the Schlafli symbol to the one given. This may not be the correct Schlafli symbol, since the relations may cause a collapse, so this should only be used if you know that the Schlafli symbol is correct.

7.1.4 ReflexibleManiplex (for IsString)

▷ `ReflexibleManiplex(name)` (operation)

Returns the regular polytope with the given symbolic name. Examples: ReflexibleManiplex("{3,3,3}"); ReflexibleManiplex("{4,3}_3"); If the option set_schlafli is set, then we set the Schlafli symbol to the one given. This may not be the correct Schlafli symbol, since the relations may cause a collapse, so this should only be used if you know that the Schlafli symbol is correct.

7.1.5 Maniplex (for IsGroup)

▷ `Maniplex(G)` (operation)

Returns a maniplex with connection group *G*, where *G* is assumed to be a permutation group on the flags.

7.1.6 IsPolytopal (for IsManiplex)

▷ `IsPolytopal(M)` (property)

Returns: `true` or `false`

Returns whether the maniplex *M* is a polytope.

Chapter 8

Actions

8.1 Faithfulness

8.1.1 IsVertexFaithful (for IsReflexibleManiplex)

▷ `IsVertexFaithful(M)` (property)

Returns: `true` or `false`

Returns whether the reflexible maniplex *M* is vertex-faithful; i.e., whether the action of the automorphism group on the vertices is faithful.

8.1.2 IsFacetFaithful (for IsReflexibleManiplex)

▷ `IsFacetFaithful(M)` (property)

Returns: `true` or `false`

Returns whether the reflexible maniplex *M* is facet-faithful; i.e., whether the action of the automorphism group on the facets is faithful.

8.1.3 MaxVertexFaithfulQuotient (for IsReflexibleManiplex)

▷ `MaxVertexFaithfulQuotient(M)` (operation)

Returns the maximal vertex-faithful reflexible maniplex covered by *M*.

Chapter 9

Posets

I'm in the process of reconciling all of this, but there are going to be a number of ways to *define* a poset:

- As an *IsPosetOfFlags*, where the underlying description is an ordered list of length $n+2$. Each of the $n+2$ list elements is a list of faces, and the assumption is that these are the faces of rank $i-2$, where i is the index in the master list (e.g., *l[1][1]* would usually correspond to the unique -1 face of a polytope – and there won't be an *l[1][2]*). Each face is then a list of the flags incident with that face.

- As an *IsPosetOfIndices*, where the underlying description is a binary relation on a set of indices, which correspond to labels for the elements of the poset.

- If the poset is known to be atomic, then by a description of the faces in terms of the atoms... usually we'll just need the list of the elements of maximal rank, from which all other elements may be obtained.

- As an *IsPosetOfElements*, where the elements could be anything, and we have a known function determining the partial order on the elements.

Usually, we assume that the poset will have a natural rank function on it.

9.1 Poset constructors

9.1.1 PosetFromFaceListOfFlags (for IsList)

▷ `PosetFromFaceListOfFlags(list)` (operation)

Returns: *IsPosetOfFlags*. Note 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. Note that if you don't include faces or ranks, this function doesn't know about about them!

Here we have a poset using the *IsPosetOfFlags* description for the triangle.

Example
```
gap> poset:=PosetFromFaceListOfFlags([[[]],[[1,2],[3,6],[4,5]],[[1,4],[2,3],[5,6]],[[1,2,3,4,5,6]
A poset using the IsPosetOfFlags representation with 8 faces.
gap> FaceListOfPoset(poset);
[ [ [  ] ], [ [ 1, 2 ], [ 3, 6 ], [ 4, 5 ] ], [ [ 1, 4 ], [ 2, 3 ], [ 5, 6 ] ], [ [ 1, 2, 3, 4, 5
```

9.1.2 PosetOfConnectionGroup (for IsPermGroup)

▷ `PosetOfConnectionGroup(g)` (operation)

Returns: *IsPosetOfFlags* with *IsFull*=true.

Given a group, returns a poset with an internal representation 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 includes 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.

Example

```
gap> g:=Group([(1,4)(2,3)(5,6),(1,2)(3,6)(4,5)]);
Group([ (1,4)(2,3)(5,6), (1,2)(3,6)(4,5) ])
gap> PosetOfConnectionGroup(g);
A poset using the IsPosetOfFlags representation with 8 faces.
```

9.1.3 PosetOfManiplex (for IsManiplex)

▷ `PosetOfManiplex(mani)` (operation)

Returns: *IsPosetOfFlags*

Given a maniplex, returns a poset of the maniplex with an internal representation 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.

Example

```
gap> p:=HemiCube(3);
Regular 3-polytope of type [ 4, 3 ] with 24 flags
gap> PosetOfManiplex(p);
A poset using the IsPosetOfFlags representation with 15 faces.
```

9.1.4 PosetFromPartialOrder (for IsBinaryRelation)

▷ `PosetFromPartialOrder(partialOrder)` (operation)

Returns: *IsPosetOfIndices*

Given a partial order on a finite set of size n, this function will create a partial order on *[1..n]*.

Example

```
gap> l:=List([[1,1],[1,2],[1,3],[1,4],[2,4],[2,2],[3,3],[4,4]],x->Tuple(x));
gap> r:=BinaryRelationByElements(Domain([1..4]), l);
<general mapping: Domain([ 1 .. 4 ]) -> Domain([ 1 .. 4 ]) >
gap> poset:=PosetFromPartialOrder(r);
A poset using the IsPosetOfIndices representation
gap> h:=HasseDiagramBinaryRelation(PartialOrder(poset));
<general mapping: Domain([ 1 .. 4 ]) -> Domain([ 1 .. 4 ]) >
gap> UnderlyingRelation(h);
Domain([ DirectProductElement( [ 1, 2 ] ), DirectProductElement( [ 1, 3 ] ), DirectProductElement
```

Note that what we've accomplished here is the poset containing the elements 1, 2, 3, 4 with partial order determined by whether the first element divides the second. The essential information about the poset can be obtained from the Hasse diagram.

9.1.5 PosetFromElements (for IsList)

▷ `PosetFromElements(list_of_faces, {func})` (operation)

Returns: *IsPosetOfElements*

This is for gathering elements with a known ordering `func` on two variables into a poset. Note... you should expect to get complete garbage if you send it a list of faces of different types. If your list of faces HasFlagList or HasAtomList, you may omit the function. Also note, the expectation is that `func` behaves similarly to IsSubset, i.e., `func` (x,y)=true means y is less than x in the order. Also worth noting, is that the internal representation of this kind of poset can and does keep both the partial order on the indices, and the list of faces corresponding to those indices, and the binary relation `func` (if the `list_of_faces` elements all have HasFlagList or HasAtomList, this will be the operation *PairCompareFlagsList* or *PairCompareAtomsList*).

Example

```
gap> g:=SymmetricGroup(3);
Sym( [ 1 .. 3 ] )
gap> asg:=AllSubgroups(g);
[ Group(()), Group([ (2,3) ]), Group([ (1,2) ]), Group([ (1,3) ]), Group([ (1,2,3) ]), Group([ (1
gap> poset:=PosetFromElements(asg,IsSubgroup);
A poset using the IsPosetOfIndices representation
gap> HasseDiagramBinaryRelation(PartialOrder(poset));
<general mapping: Domain([ 1 .. 6 ]) -> Domain([ 1 .. 6 ]) >
gap> UnderlyingRelation(last);
Domain([ DirectProductElement( [ 1, 2 ] ), DirectProductElement( [ 1, 3 ] ), DirectProductElement
gap> ElementsList(poset){[2,6]};
[ Group([ (2,3) ]), Group([ (1,2,3), (2,3) ]) ]
```

Here we have an example of how we can store a partially ordered set, and recover information about which objects are incident in the partial order.

9.1.6 PairCompareFlagsList (for IsList,IsList)

▷ `PairCompareFlagsList(list1, list2)` (operation)

Returns: *true* or *false*

Function assumes `list1` and `list2` are of the form [*listOfFlags*,*i*] where *listOfFlags* is a list of flags in the face and *i* is the rank of the face.

9.1.7 PairCompareAtomsList (for IsList,IsList)

▷ `PairCompareAtomsList(list1, list2)` (operation)

Returns: *true* or *false*

Function assumes `list1` and `list2` are of the form *[listOfAtoms,int]* where *listOfAtoms* is a list of flags in the face and *int* is the rank of the face.

9.2 Poset attributes

9.2.1 IsFlaggable (for IsPoset)

▷ `IsFlaggable(poset)` (attribute)

Returns: *true* or *false*

Checks or creates the value of the attribute IsFlaggable for an IsPoset. Point here is to see if the structure of the poset is sufficient to determine the flag graph. For IsPosetOfFlags this is another way of saying that the intersection of the faces (thought of as collections of flags) containing a flag is that selfsame flag. (Might be equivalent to prepolytopal... but Gabe was tired and Gordon hasn't bothered to think about it yet.)

9.2.2 IsAtomic (for IsPoset)

▷ `IsAtomic(`*poset*`)` (attribute)

Returns: *true* or *false*

Checks if `poset` is atomic. *Note, currently something that is not computed, just declared.*

9.2.3 PartialOrder (for IsPoset)

▷ `PartialOrder(`*poset*`)` (attribute)

Returns: *partial order*

HasPartialOrder Checks if `poset` has a declared partial order (binary relation). SetPartialOrder assigns a partial order to the `poset`. *Note, currently something that is not computed, just declared.*

9.2.4 ListIsFullPoset (for IsList)

▷ `ListIsFullPoset(`*list*`)` (operation)

Returns: *true* or *false*

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.

9.2.5 RankOfPoset (for IsPoset)

▷ `RankOfPoset(`*poset*`)` (operation)

Returns: *integer*

Given a `poset`, returns the rank of the poset. Note: There may be hidden assumptions here to untangle later. NOT IMPLEMENTED YET.

9.2.6 IsNotFull (for IsPoset)

▷ `IsNotFull(`*poset*`)` (operation)

Returns: *true* or *false*

Lets me check to see if a poset is NOT full. For use in certain filtering operations.

9.2.7 IsP1 (for IsPoset)

▷ `IsP1(`*poset*`)` (attribute)

Returns: *true* or *false*

Determines whether a poset has property P1 from ARP.

9.3 Working with posets

9.3.1 AreIncidentFlagFaces (for IsObject,IsObject)

▷ `AreIncidentFlagFaces(`*`object1, object2`*`)` (operation)

Returns: *true* or *false*

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

9.3.2 FlagsAsListOfFacesFromPoset (for IsPoset)

▷ `FlagsAsListOfFacesFromPoset(`*`poset`*`)` (operation)

Returns: *IsList*

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.

9.3.3 AdjacentFlag (for IsPosetOfFlags,IsList,IsInt)

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

Returns: *flag(s)*

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. Appending *true* to the arguments will give the position of the flag instead of its description from *FlagsAsListOfFaces-FromPoset*.

9.3.4 ConnectionGeneratorOfPoset (for IsPoset,IsInt)

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

Returns: A permutation on the flags.

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

9.3.5 ConnectionGroupOfPoset (for IsPoset)

▷ `ConnectionGroupOfPoset(`*`poset`*`)` (operation)

Returns: *IsPermGroup*

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

9.3.6 FacesOfPosetAsBinaryRelationOnFaces (for IsPoset)

▷ `FacesOfPosetAsBinaryRelationOnFaces(`*`poset`*`)` (operation)

Returns: A binary relation on the integers 1 through n, where n is the number of faces of the full poset.

FacesOfPosetAsBinaryRelationOnFaces

9.3.7 FaceListOfPoset (for IsPoset)

▷ `FaceListOfPoset(`*poset*`)` (operation)

Returns: *list*

Gives a list of faces collected into lists ordered by increasing rank.

9.4 Elements of posets, also known as faces.

9.4.1 RankPosetElement (for IsPosetElement)

▷ `RankPosetElement(`*posetelement, {face}*`)` (attribute)

Returns: *true* or *false*

The rank of a poset element. Alternately *RankFace*(`IsPosetElement`).

9.4.2 FlagList (for IsPosetElement)

▷ `FlagList(`*posetelement, {face}*`)` (attribute)

Returns: *list*

Description of `posetelement` n as a list of incident flags (when present).

9.4.3 FromPoset (for IsPosetElement)

▷ `FromPoset(`*posetelement, {face}*`)` (attribute)

Returns: *poset*

Gives the poset to which the face belongs (when present).

9.4.4 AtomList (for IsPosetElement)

▷ `AtomList(`*posetelement, {face}*`)` (attribute)

Returns: *list*

Description of `posetelement` n as a list of atoms (when present).

9.4.5 Index (for IsPosetElement)

▷ `Index(`*arg*`)` (attribute)

9.4.6 PosetElementFromListOfFlags (for IsList,IsInt)

▷ `PosetElementFromListOfFlags(`*list, n*`)` (operation)

Returns: *IsPosetElement*

This is used to create a face of rank `n` from a `list` of flags of poset. If an IsPoset object is appended to the input will tell the element what poset it belongs to.

9.4.7 PosetElementFromAtomList (for IsList,IsInt)

▷ `PosetElementFromAtomList(list, n)` (operation)

Returns: *IsFace*

Creates a *face* with `list` of atoms at rank `n`. If an IsPoset object is appended to the input will tell the element what poset it belongs to.

9.4.8 PosetElementFromIndex (for IsObject,IsInt)

▷ `PosetElementFromIndex(obj, n)` (operation)

Returns: *IsFace*

Creates a *face* with index `obj` at rank `n`. If an IsPoset object is appended to will tell the element what poset it belongs to.

9.4.9 RankedFaceListOfPoset (for IsPoset)

▷ `RankedFaceListOfPoset(poset)` (operation)

Returns: *list*

Gives a list of [`face`,`rank`] pairs for all the faces of `poset`.

9.4.10 IsSubface (for IsFace,IsFace)

▷ `IsSubface([face1, face1])` (operation)

Returns: *true* or *false*

`face1` and `face2` are IsFace or IsPosetElement. Subface will check to make sure `face2` is a subface of `face1`.

Chapter 10

Comparing maniplexes

10.1 Quotients and covers

10.1.1 IsQuotientOf (for IsManiplex, IsManiplex)

▷ `IsQuotientOf(M1, M2)` (operation)

Returns whether *M1* is a quotient of *M2*.

10.1.2 IsCoverOf (for IsManiplex, IsManiplex)

▷ `IsCoverOf(M1, M2)` (operation)

Returns whether *M1* is a cover of *M2*.

10.1.3 IsIsomorphicTo (for IsManiplex, IsManiplex)

▷ `IsIsomorphicTo(M1, M2)` (operation)

Returns whether *M1* is isomorphic to *M2*.

10.1.4 SmallestRegularCover (for IsManiplex)

▷ `SmallestRegularCover(M)` (attribute)

Returns the smallest regular cover of *M*, which is the maniplex whose automorphism group is the connection group of *M*.

Chapter 11

ramp automatic generated documentation

11.1 ramp automatic generated documentation of methods

11.1.1 UniversalRotationGroup (for IsInt)

▷ `UniversalRotationGroup(`*n*`)` (operation)

Returns the rotation subgroup of the universal Coxeter Group of rank n.

11.1.2 UniversalRotationGroup (for IsList)

▷ `UniversalRotationGroup(`*sym*`)` (operation)

Returns the rotation subgroup of the Coxeter Group with Schlafli symbol sym.

11.1.3 RotaryManiplex (for IsGroup)

▷ `RotaryManiplex(`*arg*`)` (operation)

11.1.4 RotaryManiplex (for IsList)

▷ `RotaryManiplex(`*arg*`)` (operation)

Index