AssociationSchemes

A GAP package for working with association schemes and homogeneous coherent configurations

3.1.0

7 October 2024

John Bamberg

Akihide Hanaki

Jesse Lansdown

John Bamberg
Email: john.bamberg@uwa.edu.au
Homepage: http://school.maths.uwa.edu.au/~bamberg/
Address: John Bamberg
School of Mathematics and Statistics
The University of Western Australia
35 Stirling Highway
Crawley WA 6009, Perth
Australia

Akihide Hanaki
Email: hanaki@shinshu-u.ac.jp
Homepage: http://math.shinshu-u.ac.jp/~hanaki/
Address: Akihide Hanaki
Department of Mathematics
Faculty of Science, Shinshu University
Matsumoto 390-8621, Japan

Jesse Lansdown
Email: jesse.lansdown.a1@tohoku.ac.jp
Homepage: http://www.jesselansdown.com
Address: Jesse Lansdown
Graduate School of Information Sciences
Tohoku University
Sendai 980-8579
Japan

Abstract

AssociationSchemes is a GAP package for working with association schemes and homogeneous coherent configurations.

Copyright

Acknowledgements

The third author would like to acknowledge the support of an Australian Government Research Training Program (RTP) Scholarship while writing this software. The first and third authors are also grateful for the 2019 CMSC Retreat for providing an opportunity and environment for some of the founding work on the package.

Contents

Chapter 1

Introduction

1.1 Welcome to AssociationSchemes

AssociationSchemes is a GAP[GAP16] package for working with association schemes and homogeneous coherent configurations.

For definitions and more information on the theory of association schemes and homogeneous coherent configurations, we refer you to [BI84] and [God93].

It is important to note that the term "association scheme" is used differently by different authors. We reserve the term "association scheme" to mean a symmetric coherent configuration, and use "homogeneous coherent configuration" to refer to the more general objects.

1.2 Citing AssociationSchemes

If you use AssociationSchemes in research leading to publication please cite it as you would a paper. Example citations and a BibTeX entry are given below. Please check that the version and DOI match the version of AssociationSchemes used in your research.

Please also inform us by email of the paper, as we are very interested to hear how AssociationSchemes is being used!

Example
```
@article{AssociationSchemes,
Author = {Bamberg,  J.  and  Hanaki,  A.  and  Lansdown, J.},
Doi = {10.5281/zenodo.2634954},
Key = {AssociationSchemes},
Title = {{AssociationSchemes -- AssociationSchemes: A GAP package for working
with association schemes and homogeneous coherent configurations, Version 3.1.0}},
Url = {https://doi.org/10.5281/zenodo.2634954},
Year = 2024        ,
}
```

1.3 Dependencies

AssociationSchemes requires

- GAP 4.13 (or later)

as well as the following GAP packages:

- Digraphs 0.13.0 (or later)

You may of course use AssociationSchemes without the above packages, however the corresponding functionality will be unavailable.

You may also want the following packages:

- NautyTracesInterface 0.2 (or later)

Note that NautyTracesInterface is not yet a deposited package. It can be obtained from the link in the appendix. It is not necessary, but it might improve speed of certain things such as finding the automorphism group, and checking isomorphisms.

1.4 Installation

To install AssociationSchemes, simply copy to the "pkg" directory of your GAP installation and unzip.

Alternatively, you may load the package from a location other than the GAP "pkg" directory by using the -L flag when opening GAP. Note that this requires the parent directory of AssociationSchemes to be called "pkg". See the GAP documentation for more details on how to do this. This is useful, for example, when administrative priveledges are required to access the GAP root directory.

Chapter 2

Getting Started

2.1 Tutorial - A first session with AssociationSchemes

In this section we provide a "first session" introduction to the AssociationSchemes package. It is intended to demonstrate the basic functions of the package through a series of small examples. More detailed descriptions of each of the methods are given in the chapter "Functionality". The fundamental method of describing a scheme in the AssociationSchemes package is via its relation matrix. Take for example the following relation matrix:

Example
```
gap> M:=
> [ [  0,  1,  2,  2,  3,  3,  3,  3,  3,  3,  3,  3 ],
>   [  1,  0,  2,  2,  3,  3,  3,  3,  3,  3,  3,  3 ],
>   [  2,  2,  0,  1,  3,  3,  3,  3,  3,  3,  3,  3 ],
>   [  2,  2,  1,  0,  3,  3,  3,  3,  3,  3,  3,  3 ],
>   [  3,  3,  3,  3,  0,  1,  2,  2,  3,  3,  3,  3 ],
>   [  3,  3,  3,  3,  1,  0,  2,  2,  3,  3,  3,  3 ],
>   [  3,  3,  3,  3,  2,  2,  0,  1,  3,  3,  3,  3 ],
>   [  3,  3,  3,  3,  2,  2,  1,  0,  3,  3,  3,  3 ],
>   [  3,  3,  3,  3,  3,  3,  3,  3,  0,  1,  2,  2 ],
>   [  3,  3,  3,  3,  3,  3,  3,  3,  1,  0,  2,  2 ],
>   [  3,  3,  3,  3,  3,  3,  3,  3,  2,  2,  0,  1 ],
>   [  3,  3,  3,  3,  3,  3,  3,  3,  2,  2,  1,  0 ] ];;
```

To construct a scheme from this matrix, we use the CoherentConfiguration command.

Example
```
gap> CC := HomogeneousCoherentConfiguration(M);;
```

CoherentConfiguration performs a number of checks as it constructs the scheme to make sure that it is in fact a homogeneous coherent configuration. However if you are confident that M does in fact define a scheme, then you can skip the checks by using CoherentConfigurationNC. Do not do this unless you are sure! We can display the scheme and see that GAP already knows the class and order of CC, as well that CC is symmetric and commutative.

Example
```
gap> Display(CC);
3-class association scheme of order 12.
  Symmetric: true
  Commutative: true
```

We can directly ask if CC is commutative or symmetric.

Example
```
gap> IsCommutative(CC);
true
gap> IsSymmetricCoherentConfiguration(CC);
true
```

We can retrieve the relation matrix of a scheme

Example
```
gap> relmat := RelationMatrix(CC);;
gap> relmat = M;
true
```

Example
```
gap> P := MatrixOfEigenvalues(CC);;
gap> Display(P);
[ [   1,   1,   2,   8 ],
  [   1,   1,   2,  -4 ],
  [   1,   1,  -2,   0 ],
  [   1,  -1,   0,   0 ] ]
```

If we try displaying again, we will also obtain the matrix of eigenvalues and the dual matrix of eigenvalues.

Example
```
gap> Display(CC);
3-class association scheme of order 12.
  Symmetric: true
  Commutative: true
  Metric: false
    Admits metric ordering: false
  Matrix of eigenvalues:
[ [   1,   1,   2,   8 ],
  [   1,   1,   2,  -4 ],
  [   1,   1,  -2,   0 ],
  [   1,  -1,   0,   0 ] ]
  Matrix of dual eigenvalues:
[ [   1,   2,   3,   6 ],
  [   1,   2,   3,  -6 ],
  [   1,   2,  -3,   0 ],
  [   1,  -1,   0,   0 ] ]
```

If you want to print CC, it will return the relation matrix. This is useful if you want to print to a file for exmaple.

Example
```
gap> Print(CC);
[ [ 0, 1, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3 ], [ 1, 0, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3 ],
  [ 2, 2, 0, 1, 3, 3, 3, 3, 3, 3, 3, 3 ], [ 2, 2, 1, 0, 3, 3, 3, 3, 3, 3, 3, 3 ],
  [ 3, 3, 3, 3, 0, 1, 2, 2, 3, 3, 3, 3 ], [ 3, 3, 3, 3, 1, 0, 2, 2, 3, 3, 3, 3 ],
  [ 3, 3, 3, 3, 2, 2, 0, 1, 3, 3, 3, 3 ], [ 3, 3, 3, 3, 2, 2, 1, 0, 3, 3, 3, 3 ],
  [ 3, 3, 3, 3, 3, 3, 3, 3, 0, 1, 2, 2 ], [ 3, 3, 3, 3, 3, 3, 3, 3, 1, 0, 2, 2 ],
  [ 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 0, 1 ], [ 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 1, 0 ] ]
```

You can obtain the adjacency matrices by doing:

Example
```
gap> AdjacencyMatrices(CC);;
```

If you were able to calculate the matrix of eigenvalues, then you can also construct the minimal idempotents E_i

Example
```
gap> MinimalIdempotents(CC);;
```

Note that if CC is Schurian (or has a transitive group associated with it) then MinimalIdempotents will be much faster! You can check if a scheme is schurian by doing

Example
```
gap> IsSchurian(CC);
true
```

In doing this, a graph is constructed and the automorphism group for CC is found. We can also find the automorphism group directly.

Example
```
gap> AutomorphismGroup(CC);
<permutation group with 11 generators>
```

We can define homogeneous coherent figurations from transitive groups. This is typically fast.

Example
```
gap> G := Group( [ ( 6,10)( 7,11)( 8,12)( 9,13)(15,28)(16,29)(17,30)(18,31)
>          (20,37)(21,38)(22,39)(23,40)(24,33)(25,34)(26,35)(27,36),
>      ( 1,15,22,31,18,26,16, 2, 5)( 3,24,21)( 4,20,40,29,11, 6,28,27,25)
>      ( 7,10,14)( 8,33,35,39,38,12,32,13,19)( 9,37,34)(23,36,30),
>      ( 3, 4)( 7,11)( 8, 9)(12,13)(15,28)(17,31)(18,30)(19,32)(20,33)
>      (21,25)(22,36)(23,35)(24,37)(26,40)(27,39)(34,38), () ] );;
gap> HomogeneousCoherentConfigurationByOrbitals(G);;
```

If G is transitive, we can construct a Schurian coherent configuration. If G is generously transitive, then we can constuct a Schurian association scheme (it will be symmetric)

Example
```
gap> IsTransitive(G);
true
gap> SchurianCoherentConfiguration(G);;
gap> IsGenerouslyTransitive(G);
true
gap> SchurianAssociationScheme(G);;
```

If we have a group G and subgroup H such that G acts transitively on G/H, then we can also use the following construction.

Example
```
gap> G:=SymmetricGroup(5);;
gap> H:=Stabiliser(G, 1);;
gap> HomogeneousCoherentConfigurationByOrbitals(G, H);;
```

There are a number of special constructors, such as for Johnson schemes

Example
```
gap> JohnsonScheme(10,3);
3-class association scheme of order 120.
```

AssociationSchemes also comes with a library of association schemes on small numbers of vertices, according to [HM].

Example
```
gap> m:=HomogeneousCoherentConfiguration(12, 7);;
```

We can test if two schemes are equal with "=". This will return true if the schemes have the same relation matrix. The previous example from the library is in fact the same as the example constructed from the matrix M at the start.

Example
```
gap> CC = m;
true
```

There is also the option to create a fusion scheme. This takes a partition of the relations, (where [0] must be cell of the partition. If the resulting fusion is not a valid scheme this will return fail;

Example
```
gap> FusionOfHomogeneousCoherentConfiguration(m, [[0], [1,2],[3]]);
2-class association scheme of order 12.
```

Chapter 3

Homogeneous Coherent Configuration objects

3.1 Core functionality

3.1.1 HomogeneousCoherentConfiguration (for IsMatrix)

▷ `HomogeneousCoherentConfiguration(M)` (operation)

Returns: homogeneous coherent configuration

Takes the relationship matrix, M, describing a homogeneous coherent configuration and returns a HomogeneousCoherentConfiguration object. The matrix $M = \sum_{i=0}^{d} iA_i$, where A_i are the adjacency matrices describing a coherent configuration. Checks that the matrix satisfies the axioms of a homogeneous coherent configuration.

3.1.2 HomogeneousCoherentConfigurationNC (for IsMatrix)

▷ `HomogeneousCoherentConfigurationNC(M)` (operation)

Returns: homogeneous coherent configuration

Same as HomogeneousCoherentConfiguration but without performing any checks. Use this method only if you know with certainty that M describes a coherent configuration.

3.1.3 AssociationScheme (for IsMatrix)

▷ `AssociationScheme(M)` (operation)

Returns: homogeneous coherent configuration

Takes the relationship matrix, M, describing an associatioin scheme and returns an association scheme (symmetric coherent configuration). This is simply a HomogeneousCoherentConfiguration object, but with the known property of being symmetric. The matrix $M = \sum_{i=0}^{d} iA_i$, where A_i are the adjacency matrices describing an association scheme. Checks that the matrix satisfies the association scheme axioms.

3.1.4 AssociationSchemeNC (for IsMatrix)

▷ `AssociationSchemeNC(M)` (operation)

Returns: homogeneous coherent configuration

Same as AssociationScheme but without performing any checks. Use this method only if you know with certainty that M describes an association scheme (symmetric coherent configuration).

3.1.5 ReorderRelations (for IsHomogeneousCoherentConfiguration, IsList)

▷ `ReorderRelations(CC, L)` (operation)

Returns: coherent configuration

Takes a homogeneous coherent configuration CC and a list L, where L is a reordering of the relations. Returns a homogeneous coherent configuration where the i-th relation of the CC becomes the j-th relation in the new homogeneous coherent configuration, where $j = L_i$. Note that L must be equal to $\{0,\dots,d\}$ as a set, and additionally requires that $L_1 = 0$.

3.1.6 RelationMatrix (for IsHomogeneousCoherentConfiguration)

▷ `RelationMatrix(CC)` (operation)

Returns: M

Takes a homogeneous coherent configuration and returns the underlying relation matrix $M = \sum_{i=0}^{d} iA_i$, where A_i are the adjacency matrices of the coherent configuration

3.1.7 Relation (for IsHomogeneousCoherentConfiguration, IsPosInt, IsPosInt)

▷ `Relation(CC, x, y)` (operation)

Returns: i

Takes a CC and two points, x and y, and returns i such that $(x,y) \in R_i$.

3.1.8 Neighbours (for IsHomogeneousCoherentConfiguration, IsPosInt, IsInt)

▷ `Neighbours(CC, p, k)` (operation)

Returns: L

Returns a list L of all the points y of CC such that $(p,y) \in R_k$.

3.1.9 Neighbours (for IsHomogeneousCoherentConfiguration, IsInt, IsList)

▷ `Neighbours(CC, p, L)` (operation)

Returns: L

Returns a list L of all the points y of CC such that $(p,y) \in R_k$ for some $k \in L$.

3.1.10 IsCommutative (for IsHomogeneousCoherentConfiguration)

▷ `IsCommutative(CC)` (property)

Returns: true or false

Checks if the input is a commutative coherent configuration.

3.1.11 IsSymmetricCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `IsSymmetricCoherentConfiguration(CC)` (property)

Returns: true or false

Checks if the input is a symmetric coherent configuration.

3.1.12 IsAssociationScheme (for IsHomogeneousCoherentConfiguration)

▷ `IsAssociationScheme(CC)` (operation)

Returns: true or false

Alias for IsSymmetricCoherentConfiguration

3.1.13 NumberOfClasses (for IsHomogeneousCoherentConfiguration)

▷ `NumberOfClasses(CC)` (attribute)

Returns: d

Returns d for a d-class homogeneous coherent configuration.

3.1.14 Rank (for IsHomogeneousCoherentConfiguration)

▷ `Rank(CC)` (operation)

Returns: d

Returns d for a d-class homogeneous coherent configuration.

3.1.15 AdjacencyMatrices (for IsHomogeneousCoherentConfiguration)

▷ `AdjacencyMatrices(CC)` (attribute)

Returns: L

Returns a list L, where the i-th entry of L is the adjacency matrix A_{i-1}, where $(A_i)_{xy} = 1$ if $(x,y) \in R_i$ and $(A_i)_{xy} = 0$ otherwise.

3.1.16 Order (for IsHomogeneousCoherentConfiguration)

▷ `Order(CC)` (attribute)

Returns: n

Returns the order n (number of vertices) of the coherent configuration.

3.1.17 IntersectionNumber (for IsHomogeneousCoherentConfiguration, IsInt, IsInt, IsInt)

▷ `IntersectionNumber(CC, i, j, k)` (operation)

Returns: p_{ij}^k

Returns the intersection number p_{ij}^k for a coherent configuration CC.

3.1.18 Valencies (for IsHomogeneousCoherentConfiguration)

▷ `Valencies(CC)` (attribute)

Returns: L

Returns a list L of valencies of a coherent configuration CC. The i-th entry of L is k_{i-1}.

3.1.19 IntersectionAlgebraOfHomogeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `IntersectionAlgebraOfHomogeneousCoherentConfiguration(CC)` (attribute)

Returns: L

Returns an IntersectionAlgebra object for CC

3.1.20 IntersectionMatrices (for IsHomogeneousCoherentConfiguration)

▷ `IntersectionMatrices(CC)` (attribute)

Returns: L

Returns a list L of the intersection matrices of a homogeneous coherent configuration *CC*, where the *i*-th entry of *L* is B_{i-1} and $(B_i)_{jk} = p_{ij}^k$.

3.1.21 NumberOfCharacters (for IsHomogeneousCoherentConfiguration)

▷ `NumberOfCharacters(CC)` (attribute)

Returns: n

Returns the number *n* of characters of CC.

3.1.22 SplittingField (for IsHomogeneousCoherentConfiguration)

▷ `SplittingField(CC)` (attribute)

Returns: F

Returns the splitting field of the CC

3.1.23 HasRationalSplittingField (for IsHomogeneousCoherentConfiguration)

▷ `HasRationalSplittingField(CC)` (property)

Returns: true or false

Returns true if the splitting field is the rationals, false otherwise.

3.1.24 KreinParameter (for IsHomogeneousCoherentConfiguration, IsInt, IsInt, IsInt)

▷ `KreinParameter(CC, i, j, k)` (operation)

Returns: $q_{i,j}^k$

Compute the krein parameter $q_{i,j}^k$ of a commutative homogeneous coherent configuration.

3.1.25 KreinParameters (for IsHomogeneousCoherentConfiguration)

▷ `KreinParameters(CC)` (attribute)

Returns: L

Return a list *L* of all Krein parameters of a commutative homogeneous coherent configuration, where $L[i][j,k] = q_{i,j}^k$.

3.1.26 SaveHomogeneousCoherentConfigurationWithCertainAttributes (for IsString, IsHomogeneousCoherentConfiguration, IsList)

▷ `SaveHomogeneousCoherentConfigurationWithCertainAttributes(`*`file, A, L`*`)` (operation)

Returns: true

Saves homogeneous coherent configuration A to file F with the attributes listed in L. Note that L must be a list of strings, where each entry is an attribute known for A. Note that Print or PrintTo will only return the relation matrix of a homogeneous coherent configuration, which contains all necessary information about the homogeneous coherent configuration, but may require a lot of computation to reobtain its attributes. Hence this method is intended to alow saving of computationally difficult or time consuming attributes directly. It also alows the user to choose which attributes to save, since some attributes are very large, but easily recomputed. For example, it is often desirable to save the matrix of eigenvalues, and perhaps the automorphism group and intersection matrices, while it is not generally desirable to also save the adjacency matrices or minimal idempotents.

3.1.27 ReadHomogeneousCoherentConfigurationWithCertainAttributes (for IsString)

▷ `ReadHomogeneousCoherentConfigurationWithCertainAttributes(`*`file, A, L`*`)` (operation)

Returns: homogeneous coherent configuration

Reads in a homogenous coherent configuration from file and sets it to have the attributes stored in the file. This reads files of the type formed by SaveHomogeneousCoherentConfigurationWithCertainAttributes.

3.1.28 IsThin (for IsHomogeneousCoherentConfiguration)

▷ `IsThin(`*`CC`*`)` (property)

Returns: true or false

Check if the homogeneous coherent configuration is thin.

3.1.29 IsQuasiThin (for IsHomogeneousCoherentConfiguration)

▷ `IsQuasiThin(`*`CC`*`)` (property)

Returns: true or false

Check if the homogeneous coherent configuration is quasi thin.

3.1.30 IsPrimitive (for IsHomogeneousCoherentConfiguration)

▷ `IsPrimitive(`*`CC`*`)` (property)

Returns: true or false

Check if the homogeneous coherent configuration is primitve.

3.1.31 ReorderMinimalIdempotents (for IsHomogeneousCoherentConfiguration, IsList)

▷ `ReorderMinimalIdempotents(`*`CC, L`*`)` (operation)

Returns: coherent configuration

Takes a homogeneous coherent configuration CC and a list L, where L is a reordering of the minimal idempotents. Returns a homogeneous coherent configuration where the i-th idempotent of the CC becomes the j-th idempotent in the new homogeneous coherent configuration, where $j = L_i$. Note that L_i must be equal to $\{0, \ldots, d\}$ as a set, and additionally requires that $L_1 = 0$.

3.1.32 ViewRelationDistributionDiagram (for IsHomogeneousCoherentConfiguration)

▷ `ViewRelationDistributionDiagram(CC)` (operation)

Returns: true (Displays relation diistribution diagram)

Take a CC and display the relation-distribution diagram with respect to R_1.

3.1.33 Description (for IsHomogeneousCoherentConfiguration)

▷ `Description(CC)` (attribute)

Returns: lis

Take a CC and returns a list containing various descriptions/names of the CC, if available. Note that most homogeneous coherent configurations will not have a description. Some famous homogeneous coherent configurations, association schemes, or distance regular graphs in the library, as well as families that have constructor methods, will have names. Some will have multiple descriptions, hence they are given as a list. You can check if a homogeneous coherent configuration has assigned descriptions with HasDescription, or set one with SetDescription.

3.2 Constructor methods

3.2.1 DistanceRegularGraphScheme (for IsMatrix)

▷ `DistanceRegularGraphScheme(A)` (operation)

Returns: homogeneous coherent configuration

Constructs an association scheme from the adjacency matrix A of a distance regular graph.

3.2.2 DistanceRegularGraphSchemeNC (for IsMatrix)

▷ `DistanceRegularGraphSchemeNC(A)` (operation)

Returns: homogeneous coherent configuration

Same as DistanceRegularGraphScheme but without checking that a valid association scheme arises.

3.2.3 StronglyRegularGraphScheme (for IsMatrix)

▷ `StronglyRegularGraphScheme(A)` (operation)

Returns: homogeneous coherent configuration

Constructs an association scheme from the adjacency matrix A of a strongly regular graph.

3.2.4 StronglyRegularGraphSchemeNC (for IsMatrix)

▷ `StronglyRegularGraphSchemeNC(A)` (operation)

Returns: homogeneous coherent configuration

Same as StronglyRegularGraphScheme but without checking that a valid association scheme arises.

3.2.5 BilinearFormsScheme (for IsField, IsPosInt, IsPosInt)

▷ `BilinearFormsScheme(F, n, k)` (operation)

Returns: homogeneous coherent configuration

Returns the bilinear forms scheme for the finite field F with a bilinear form from $F^n \times F^n$ to F^k.

3.2.6 CyclotomicScheme (for IsPosInt, IsPosInt)

▷ `CyclotomicScheme(q, d)` (operation)

Returns: homogeneous coherent configuration

Let q be a prime power, and d a divisor of $q-1$. Returns the cyclotomic scheme $Cyc(q,d)$.

3.2.7 GrassmannScheme (for IsPosInt, IsPosInt, IsPosInt)

▷ `GrassmannScheme(n, k, q)` (operation)

Returns: homogeneous coherent configuration

Returns the Grassmann scheme, $J_q(n,k)$.

3.2.8 GroupCoherentConfiguration (for IsGroup)

▷ `GroupCoherentConfiguration(G)` (operation)

Returns: homogeneous coherent configuration

Returns the coherent configuration on the conjugacy classes of a group G.

3.2.9 HammingScheme (for IsPosInt, IsPosInt)

▷ `HammingScheme(n, q)` (operation)

Returns: homogeneous coherent configuration

Returns the Hamming scheme, $H(n,q)$.

3.2.10 JohnsonScheme (for IsPosInt, IsPosInt)

▷ `JohnsonScheme(n, k)` (operation)

Returns: homogeneous coherent configuration

Returns the Johnson scheme, $J(n,k)$.

3.2.11 DirectProductOfHomogeneousCoherentConfigurations (for IsHomogeneousCoherentConfiguration, IsHomogeneousCoherentConfiguration)

▷ `DirectProductOfHomogeneousCoherentConfigurations(CC1, CC2)` (operation)

Returns: homogeneous coherent configuration

Takes two homogeneous coherent configurations CC1 and CC2 and returns their direct product.

3.2.12 WreathProductOfHomogeneousCoherentConfigurations (for IsHomogeneousCoherentConfiguration, IsHomogeneousCoherentConfiguration)

▷ `WreathProductOfHomogeneousCoherentConfigurations(CC1, CC2)` (operation)

Returns: homogeneous coherent configuration

Takes two homogeneous coherent configurations CC1 and CC2 and returns their wreath product.

3.2.13 BipartiteDoubleOfAssociationScheme (for IsHomogeneousCoherentConfiguration)

▷ `BipartiteDoubleOfAssociationScheme(A)` (operation)

Returns: Association scheme

Returns the bipartite double of an association scheme.

3.2.14 ExtendedQBipartiteDouble (for IsHomogeneousCoherentConfiguration)

▷ `ExtendedQBipartiteDouble(A)` (operation)

Returns: Association scheme

Given a cometric association scheme satisfying $b^*_j + c^*_{j+1} = m+1$ for $0 \leq j \leq d-1$, returns the extended Q-bipartite double.

3.2.15 HomogeneousCoherentConfigurationByOrbitals (for IsPermGroup)

▷ `HomogeneousCoherentConfigurationByOrbitals(G)` (operation)

Returns: homogeneous coherent configuration

Constructs a "group-case" coherent configuration, where the relations are defined by the orbitals of G on $\{1,\dots,n\} \times \{1,\dots,n\}$. G must be a permutation group which is transitive on $\{1,\dots,n\}$.

3.2.16 HomogeneousCoherentConfigurationByOrbitals (for IsGroup, IsGroup)

▷ `HomogeneousCoherentConfigurationByOrbitals(G, H)` (operation)

Returns: homogeneous coherent configuration

Constructs a "group-case" coherent configuration, where the relations are defined by the orbitals of G on G/H. G is a group, H is a subgroup of G, G/H is the set of right cosets of G on H, and G must be transitive on G/H.

3.2.17 SchurianAssociationScheme (for IsPermGroup)

▷ `SchurianAssociationScheme(G)` (operation)

Returns: homogeneous coherent configuration

Returns the Schurian scheme defined by G, where G is a generously transitive permutation group. A Schurian scheme is a special case of CoherentConfigurationByOrbitals and is symmetric.

3.2.18 SchurianCoherentConfiguration (for IsPermGroup)

▷ `SchurianCoherentConfiguration(G)` (operation)

Returns: homogeneous coherent configuration

Alias for HomogeneousCoherentConfigurationByOrbitals

3.3 Library

3.3.1 HomogeneousCoherentConfiguration (for IsPosInt, IsPosInt)

▷ `HomogeneousCoherentConfiguration(n, k)` (operation)

Returns: homogeneous coherent configuration

Returns the k-th homogeneous coherent configuration of order n. Library is complete for $5 \leq n \leq 38$ excluding $n \in \{35, 36, 37\}$, corresponding to [HM]. Thin schemes are are assigned numbers after those in [HM].

3.3.2 NumberOfHomogeneousCoherentConfigurations (for IsPosInt)

▷ `NumberOfHomogeneousCoherentConfigurations(n)` (operation)

Returns: m

Returns the number m of homogeneous coherent configuration of order n contained in the library.

3.3.3 AvailableHomogeneousCoherentConfigurations

▷ `AvailableHomogeneousCoherentConfigurations()` (operation)

Returns: L

Returns a list L of the orders for which the library contains homogeneous coherent configurations.

3.3.4 AllHomogeneousCoherentConfigurations (for IsPosInt)

▷ `AllHomogeneousCoherentConfigurations(n)` (operation)

Returns: L

Returns a list L of all homogeneous coherent configuration of order n.

3.3.5 SmallSchemeIdentification (for IsHomogeneousCoherentConfiguration)

▷ `SmallSchemeIdentification(CC)` (attribute)

Returns: id

Returns the identification, id, of the homogeneous coherent configuration in the library which is is isomorphic to CC. Thus HomogeneousCoherentConfiguration(n, id) will be isomorphic to CC, where n is the order of CC.

3.4 Graphs, automorphisms, and maps

3.4.1 Digraph (for IsHomogeneousCoherentConfiguration, IsPosInt)

▷ `Digraph(CC, k)` (operation)

Returns: homogeneous coherent configuration

Returns the digraph object associated with the k-th relation of a homogeneous coherent configuration CC. Note that the identity relation is excluded.

3.4.2 Digraph (for IsHomogeneousCoherentConfiguration, IsList)

▷ `Digraph(CC, S)` (operation)

Returns: homogeneous coherent configuration

Returns the digraph object which is a union of the relations of a homogeneous coherent configuration CC given by the set S. Note that the identity relation is excluded.

3.4.3 AutomorphismGroup (for IsHomogeneousCoherentConfiguration)

▷ `AutomorphismGroup(CC)` (attribute)

Returns: G

Returns the automorphism group G of the coherent configuration CC. G is a permutation group acting on the index set of the vertices.

3.4.4 AlgebraicAutomorphismGroup (for IsHomogeneousCoherentConfiguration)

▷ `AlgebraicAutomorphismGroup(CC)` (attribute)

Returns: G

Returns the algebraic automorphism group G of the coherent configuration CC. G is a permutation group acting on the set of non-trivial relations.

3.4.5 ImageOfHomogeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration, IsPerm, IsPerm)

▷ `ImageOfHomogeneousCoherentConfiguration(CC, p, $\sigma$)` (operation)

Returns: true or false

For a d-class homogeneous coherent configuration with relation matrix M and order n, takes a permutation p on the set $\{1..n\}$ and permutation σ on the set $\{1..d\}$ and returns the d-class homogenous coherent configuration with relation matrix $P^{-1}M^{\sigma}P$, where P is the permutation matrix defined by σ.

3.4.6 IsomorphismHomogeneousCoherentConfigurations (for IsHomogeneousCoherentConfiguration, IsHomogeneousCoherentConfiguration)

▷ `IsomorphismHomogeneousCoherentConfigurations(A, B)` (operation)

Returns: true or false

If there exists a permutation matrix P and permutation σ such that if M is the relation matrix of A, then $P^{-1}M^{\sigma}P$ is the relation matrix of B, then the operation will return $[p,\sigma]$, where p is the permutation defining P. If no such P and σ exist, then the operation will return fail.

3.4.7 AlgebraicIsomorphismHomogeneousCoherentConfigurations (for IsHomogeneousCoherentConfiguration, IsHomogeneousCoherentConfiguration)

▷ `AlgebraicIsomorphismHomogeneousCoherentConfigurations(A, B)` (operation)

Returns: true or false

Returns a permutation which maps the set of relations of A to B such that the intersection numbers are preserved. Returns fail if such a bijection between relations does not exist.

3.4.8 AreIsomorphicHomogeneousCoherentConfigurations (for IsHomogeneousCoherentConfiguration, IsHomogeneousCoherentConfiguration)

▷ `AreIsomorphicHomogeneousCoherentConfigurations(A, B)` (operation)

Returns: true or false

If there exists a permutation matrix P and permutation σ such that if M is the relation matrix of A, then $P^{-1}M^{\sigma}P$ is the relation matrix of B, then the operation will return true. Returns false otherwise.

3.4.9 AreAlgebraicallyIsomorphicHomogeneousCoherentConfigurations (for IsHomogeneousCoherentConfiguration, IsHomogeneousCoherentConfiguration)

▷ `AreAlgebraicallyIsomorphicHomogeneousCoherentConfigurations(A, B)` (operation)

Returns: true or false

Returns true if there exists a permutation which maps the set of relations of A to B such that the intersection numbers are preserved. Returns false otherwise.

3.4.10 CanonisingMap (for IsHomogeneousCoherentConfiguration)

▷ `CanonisingMap(CC)` (attribute)

Returns: [perm1, perm2]

Returns two permutations which will produce the canonical form of the homogeneous coherent configuration CC. The canonical form can be obtained by ImageOfHomogeneousCoherentConfiguration(CC, perm1, perm2) Any homogenouse coherent configuration which is isomorphic to CC will the same canonical form.

3.4.11 CanonicalFormOfHomogeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `CanonicalFormOfHomogeneousCoherentConfiguration(CC)` (operation)

Returns: CC2

Returns the canonical form, CC2, of the homogeneous coherent configuration CC. Any homogenouse coherent configuration which is isomorphic to CC will have CC2 as the canonical form.

3.4.12 ConstructorGroup (for IsHomogeneousCoherentConfiguration)

▷ `ConstructorGroup(CC)` (attribute)

Returns: group or false

Checks if the coherent configuration was constructed by a group and returns it if it was, or returns false otherwise.

3.4.13 IsGenerouslyTransitive (for IsPermGroup)

▷ `IsGenerouslyTransitive(G)` (property)

Returns: true or false

Checks if the permutation group G is generously transitive.

3.4.14 IsGenerouslyTransitive (for IsPermGroup, IsList)

▷ `IsGenerouslyTransitive(G, L)` (operation)

Returns: true or false

Checks that the permutation group G acts generously transitive on the set L.

3.4.15 IsSchurian (for IsHomogeneousCoherentConfiguration)

▷ `IsSchurian(CC)` (property)

Returns: true or false

Checks if the input is a Schurian scheme, that is, if the automorphism group is transitive.

3.5 Fusions

3.5.1 IsFusionOfHomogeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration, IsList)

▷ `IsFusionOfHomogeneousCoherentConfiguration(CC, L)` (operation)

Returns: true or false

Takes a d-class homogeneous coherent configuration CC, and checks if the partion L of $\{0,\dots,d\}$ corresponds to a valid fusion.

3.5.2 FusionOfHomogeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration, IsList)

▷ `FusionOfHomogeneousCoherentConfiguration(CC, L)` (operation)

Returns: homogeneous coherent configuration

Takes a d-class homogeneous coherent configuration CC and returns a fusion scheme corresponding to L, where L is a partion of $\{0,\dots,d\}$. Note that the ordering of the cells of L is irrelevant. The method will sort the fused relations according to the smallest value in each cell.

3.5.3 ConverseRelationPairs (for IsHomogeneousCoherentConfiguration)

▷ `ConverseRelationPairs(CC)` (attribute)

Returns: L

Returns a list L of either tuples or singletons, corresponding to relations and their converses or symmetric relations.

3.5.4 ConverseRelation (for IsHomogeneousCoherentConfiguration, IsInt)

▷ `ConverseRelation(CC, i)` (operation)

Returns: j

Returns j such that $R_j = R_i^\top$, the converse relation of i .

3.5.5 IsStratifiable (for IsHomogeneousCoherentConfiguration)

▷ `IsStratifiable(`*`CC`*`)` (attribute)

Returns: true or false

If the fusion of transposed relations produces a valid association scheme, then CC is stratifiable.

3.5.6 SymmetrisationOfHomogeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `SymmetrisationOfHomogeneousCoherentConfiguration(`*`CC`*`)` (operation)

Returns: Association scheme

Given a homogeneous coherent configuration, CC, the symmetrisation is computed if possible, otherwise fail is returned. The symmetrisation of a homogeneous coherent configuration takes any non-symmetric relations and fuses them together. The result may or may not be a valid homogeneous coherent configuration. If it is valid, then it is an association scheme (Symmetric coherent configuration). If CC is commutative, then it can be symmetrised.

3.5.7 FusingPartitionOfHomogeneousCoherentConfigurations (for IsHomogeneousCoherentConfiguration, IsHomogeneousCoherentConfiguration)

▷ `FusingPartitionOfHomogeneousCoherentConfigurations(`*`CC1, CC2`*`)` (operation)

Returns: partition

Takes two homogeneous coherent configurations, CC1 and CC2, where CC1 is d-class. If CC2 is equal to a homogeneous coherent configuration formed by fusing the relations of CC2, this will return the partition of $0,\dots,d$ corresponding to this fusion. If CC2 cannot be produced as a fusion, then "fail" is returned. This operation does not consider isomorphic homogeneous coherent configurations - CC2 must be exactly equal to a fusion.

3.5.8 FeasibleNonTrivialFusionsOfHomgeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `FeasibleNonTrivialFusionsOfHomgeneousCoherentConfiguration(`*`CC[, k[, flag]]`*`)` (attribute)

Returns: list of feasibly fusionable relations

Returns a list where each entry is a collection of relations which may be fused to form a feasible homogeneous coherent configuration Trivial means either no relations are fused, or all non-identity relations are fused. If the additional argument k is given, only fusions with k-classes are returned. If flag is also given and is equal to true, then all fusions with at most k-classes are returned.

3.5.9 AllNonTrivialFusionsOfHomgeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `AllNonTrivialFusionsOfHomgeneousCoherentConfiguration(`*`CC`*`)` (operation)

Returns: List of all non-trivial fusions of CC

Returns a list of all homogeneous coherent configurations such that each element of the list is a non-trivial fusion of CC. Trivial means either no relations are fused, or all non-identity relations are fused. If the additional argument k is given, only fusions with k-classes are returned. If flag is also given and is equal to true, then all fusions with at most k-classes are returned.

3.5.10 AllFusionsOfHomgeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `AllFusionsOfHomgeneousCoherentConfiguration(`*`CC`*`)` (operation)

Returns: List of all fusions of CC

Returns a list of all homogeneous coherent configurations such that each element of the list is a fusion of CC. Includes trivial fusions, i.e the original homogeneous coherent configuration, and the coherent configuration resulting from the fusion of all non-identity relations

3.5.11 FeasibleNonTrivialSymmetricFusionsOfHomgeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `FeasibleNonTrivialSymmetricFusionsOfHomgeneousCoherentConfiguration(`*`CC`*`)` (attribute)

Returns: list of feasibly fusionable relations

Returns a list where each entry is a collection of relations which may be fused to form a feasible association scheme (i.e. relations are symmetric) Trivial means either no relations are fused, or all non-identity relations are fused. If the additional argument k is given, only fusions with k-classes are returned. If flag is also given and is equal to true, then all fusions with at most k-classes are returned.

3.5.12 AllNonTrivialSymmetricFusionsOfHomgeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `AllNonTrivialSymmetricFusionsOfHomgeneousCoherentConfiguration(`*`CC`*`)` (operation)

Returns: List of all non-trivial fusions of CC

Returns a list of all association schemes (i.e symmetric relations) such that each element of the list is a non-trivial fusion of CC. Trivial means either no relations are fused, or all non-identity relations are fused. If the additional argument k is given, only fusions with k-classes are returned. If flag is also given and is equal to true, then all fusions with at most k-classes are returned.

3.5.13 AllSymmetricFusionsOfHomgeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `AllSymmetricFusionsOfHomgeneousCoherentConfiguration(`*`CC`*`)` (operation)

Returns: List of all fusions of CC

Returns a list of all association schemes (i.e symmetric relations) such that each element of the list is a fusion of CC. Includes trivial fusions, i.e the original homogeneous coherent configuration (if it is an association scheme), and the coherent configuration resulting from the fusion of all non-identity relations

3.5.14 IsomorphismToFusionScheme (for IsHomogeneousCoherentConfiguration, IsHomogeneousCoherentConfiguration)

▷ `IsomorphismToFusionScheme(`*`A`*`, `*`B`*`)` (operation)

Returns: p1, p2, f

If A is isomorphic to a fusion of B, then f is the fusion of *B*, and [p1, p2] is the map which carries *B* to this fusion.

3.5.15 IsIsomorphicToFusionScheme (for IsHomogeneousCoherentConfiguration, IsHomogeneousCoherentConfiguration)

▷ `IsIsomorphicToFusionScheme(A, B)` (operation)

Returns: p1, p2, f

Returns true if *A* is isomorphic to a fusion of *B*. Returns false otherwise.

3.5.16 FirstFeasibleNonTrivialSymmetricFusionOfHomogeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `FirstFeasibleNonTrivialSymmetricFusionOfHomogeneousCoherentConfiguration(CC)` (attribute)

Returns: list of feasibly fusionable relations

Returns a list of relations which may be fused to form a feasible association scheme (i.e. relations are symmetric), if possible. Returns fail if no nontrivial fusion exists. Trivial means either no relations are fused, or all non-identity relations are fused.

3.6 Bose-Mesner algebra

3.6.1 MapFromAdjacencyMatricesToMinimalIdempotents (for IsHomogeneousCoherentConfiguration)

▷ `MapFromAdjacencyMatricesToMinimalIdempotents(CC)` (attribute)

Returns: M

Takes a homogeneous coherent configuration object, CC, and returns a matrix, M, which maps the adjacency matrices to the minimal idempotents of the adjacency algebra. The central idempotent $E_i = \sum_{i=0}^{d} M_{ij} A_j$.

3.6.2 MapFromAdjacencyMatricesToMinimalIdempotentsOverRationals (for IsHomogeneousCoherentConfiguration)

▷ `MapFromAdjacencyMatricesToMinimalIdempotentsOverRationals(CC)` (attribute)

Returns: M

Takes a homogeneous coherent configuration, CC, and returns a matrix, M, which maps the adjacency matrices to the minimal idempotents of the adjacency algebra over the rationals. The minimal idempotent $E_i = \sum_{i=0}^{d} M_{ij} A_j$.

3.6.3 MinimalIdempotents (for IsHomogeneousCoherentConfiguration)

▷ `MinimalIdempotents(CC)` (attribute)

Returns: minimal idempotents

Returns the minimal idempotents of the homogeneous coherent configuration.

3.6.4 MinimalIdempotentsOverRationals (for IsHomogeneousCoherentConfiguration)

▷ `MinimalIdempotentsOverRationals(CC)` (attribute)

Returns: minimal idempotents

Returns the minimal idempotents of the homogeneous coherent configuration over the rationals.

3.6.5 MatrixOfEigenvalues (for IsHomogeneousCoherentConfiguration)

▷ `MatrixOfEigenvalues(`*`CC`*`)` (attribute)

Returns: P

Returns a the matrix of eigenvalues (or character table), *P*, for a homogeneous coherent configuration CC.

3.6.6 MatrixOfDualEigenvalues (for IsHomogeneousCoherentConfiguration)

▷ `MatrixOfDualEigenvalues(`*`CC`*`)` (attribute)

Returns: Q

Returns a the matrix of dual eigenvalues, *Q*, for a homogeneous coherent configuration CC.

3.6.7 Multiplicities (for IsHomogeneousCoherentConfiguration)

▷ `Multiplicities(`*`CC`*`)` (attribute)

Returns: n

Returns the multiplicities of characters of CC.

3.6.8 CharacterTableOfHomogeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `CharacterTableOfHomogeneousCoherentConfiguration(`*`CC`*`)` (attribute)

Returns: P

TODO.

3.6.9 FitMatrixOfEigenvalues (for IsHomogeneousCoherentConfiguration, IsMatrix)

▷ `FitMatrixOfEigenvalues(`*`CC, P`*`)` (operation)

Returns: P2

Checks if P is the matrix of eigenvalues of homogeneous coherent configuration CC, upto some reordering of the columns. In such a case, P2, the reordered matrix is returned. If not, returns fail.

3.6.10 CharacterTableOfSchurianHomogeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration)

▷ `CharacterTableOfSchurianHomogeneousCoherentConfiguration(`*`CC, P`*`)` (operation)

Returns: P2

Computes the character table of a Schurian coherent configuration by using the group. The ordering of the columns does not respect the ordering of the coherent configuration, so "FitMatrixOfEigenvalues" must be used. Sometimes the group theoretic method is much faster and sometimes it is much slower than the other methods.

3.6.11 IsCharacterTableOfHomogeneousCoherentConfiguration (for IsHomogeneousCoherentConfiguration, IsMatrix, IsList)

▷ `IsCharacterTableOfHomogeneousCoherentConfiguration(`*`CC, T, mults`*`)` (operation)

Returns: P2

Checks if T is the matrix of eigenvalues of homogeneous coherent configuration CC, given multiplicities mults.

3.6.12 MatrixOfEigenvaluesOfCyclotomicScheme (for IsPosInt, IsPosInt)

▷ `MatrixOfEigenvaluesOfCyclotomicScheme(`*`n, k, q`*`)` (operation)

Returns: P

Returns the matrix of eigenvalues P of the scheme $Cyc(q,d)$.

3.6.13 MatrixOfEigenvaluesOfGrassmannScheme (for IsPosInt, IsPosInt, IsPosInt)

▷ `MatrixOfEigenvaluesOfGrassmannScheme(`*`n, k, q`*`)` (operation)

Returns: P

Returns the matrix of eigenvalues P of the Grassmann scheme $J_q(n,k)$.

3.6.14 MatrixOfEigenvaluesOfHammingScheme (for IsPosInt, IsPosInt)

▷ `MatrixOfEigenvaluesOfHammingScheme(`*`n, q`*`)` (operation)

Returns: P

Returns matrix of eigenvalue P for the Hamming scheme, $H(n,q)$.

3.6.15 MatrixOfEigenvaluesOfJohnsonScheme (for IsPosInt, IsPosInt)

▷ `MatrixOfEigenvaluesOfJohnsonScheme(`*`n, k`*`)` (operation)

Returns: P

Returns the matrix of eigenvalues P of the Johnson scheme $J(n,k)$.

3.7 Metric schemes

3.7.1 IsPPolynomial (for IsHomogeneousCoherentConfiguration)

▷ `IsPPolynomial(`*`CC`*`)` (property)

Returns: true or false

Returns if the homogeneous coherent configuration CC is P-polynomial.

3.7.2 FirstPPolynomialOrdering (for IsHomogeneousCoherentConfiguration)

▷ `FirstPPolynomialOrdering(`*`CC`*`)` (attribute)

Returns: P-polynomial ordering or fail

Returns the first P-polynomial ordering admitted by the homogeneous coherent configuration CC, and fail otherwise.

3.7.3 AdmitsPPolynomialOrdering (for IsHomogeneousCoherentConfiguration)

▷ `AdmitsPPolynomialOrdering(`*CC*`)` (property)

Returns: true or false

Returns if the homogeneous coherent configuration CC admits a P-polynomial ordering.

3.7.4 IsMetric (for IsHomogeneousCoherentConfiguration)

▷ `IsMetric(`*CC*`)` (operation)

Returns: true or false

Alias for IsPPolynomial.

3.7.5 FirstMetricOrdering (for IsHomogeneousCoherentConfiguration)

▷ `FirstMetricOrdering(`*CC*`)` (operation)

Returns: metric ordering or fail

Alias for FirstPPolynomialOrdering.

3.7.6 AdmitsMetricOrdering (for IsHomogeneousCoherentConfiguration)

▷ `AdmitsMetricOrdering(`*CC*`)` (operation)

Returns: true or false

Alias for AdmitsPPolynomialOrdering.

3.7.7 AllPPolynomialOrderings (for IsHomogeneousCoherentConfiguration)

▷ `AllPPolynomialOrderings(`*CC*`)` (attribute)

Returns: L

Calculate the list L of all P-polynomial orderings of a homogeneous coherent configuration.

3.7.8 AllMetricOrderings (for IsHomogeneousCoherentConfiguration)

▷ `AllMetricOrderings(`*CC*`)` (operation)

Returns: L

Alias for AllPPolynomialOrderings.

3.7.9 IsStronglyRegularGraph (for IsHomogeneousCoherentConfiguration)

▷ `IsStronglyRegularGraph(`*CC*`)` (property)

Returns: true or false

Check if a coherent configuration is a strongly regular graph (a 2-class primitive association scheme).

3.7.10 IntersectionArray (for IsHomogeneousCoherentConfiguration)

▷ `IntersectionArray(`*CC*`)` (attribute)

Returns: List

Returns the intersection array if CC is P-polynomial.

3.7.11 ClassicalParameters (for IsHomogeneousCoherentConfiguration)

▷ `ClassicalParameters(`*CC*`)` (attribute)

Returns: [d, b, α, β]

Returns the classical parameters if the CC is metric with classical parameters.

3.7.12 StronglyRegularGraphParameters (for IsHomogeneousCoherentConfiguration)

▷ `StronglyRegularGraphParameters(`*CC*`)` (attribute)

Returns: [d, b, α, β]

Returns the parameters $\{n,k;\lambda,\mu\}$ if the CC is an association scheme with 2 classes.

3.7.13 IsPBipartite (for IsHomogeneousCoherentConfiguration)

▷ `IsPBipartite(`*CC*`)` (property)

Returns: true or false

Returns if the homogeneous coherent configuration CC is bipartite.

3.7.14 IsPAntipodal (for IsHomogeneousCoherentConfiguration)

▷ `IsPAntipodal(`*CC*`)` (property)

Returns: true or false

Returns if the homogeneous coherent configuration CC is antipodal.

3.8 Cometric schemes

3.8.1 AdmitsQPolynomialOrdering (for IsHomogeneousCoherentConfiguration)

▷ `AdmitsQPolynomialOrdering(`*CC*`)` (property)

Returns: true or false

Returns if the homogeneous coherent configuration CC admits a Q-polynomial ordering.

3.8.2 AdmitsCometricOrdering (for IsHomogeneousCoherentConfiguration)

▷ `AdmitsCometricOrdering(`*CC*`)` (operation)

Returns: true or false

Alias for AdmitsQPolynomialOrdering.

3.8.3 IsQPolynomial (for IsHomogeneousCoherentConfiguration)

▷ `IsQPolynomial(`*CC*`)` (property)

Returns: true or false

Returns if the commutative coherent configuration CC is Q-polynomial.

3.8.4 IsCometric (for IsHomogeneousCoherentConfiguration)

▷ `IsCometric(CC)` (operation)

Returns: true or false

Alias for is Q-polynomial.

3.8.5 AllQPolynomialOrderings (for IsHomogeneousCoherentConfiguration)

▷ `AllQPolynomialOrderings(CC)` (attribute)

Returns: L

Calculate a list *L* of all Q-polynomial orderings of a homogeneous coherent configuration.

3.8.6 AllCometricOrderings (for IsHomogeneousCoherentConfiguration)

▷ `AllCometricOrderings(CC)` (operation)

Returns: L

Alias for AllQPolynomialOrderings.

3.8.7 KreinArray (for IsHomogeneousCoherentConfiguration)

▷ `KreinArray(CC)` (attribute)

Returns: List

Returns the Krein (or dual intersection) array if CC is Q-polynomial.

3.8.8 DualIntersectionArray (for IsHomogeneousCoherentConfiguration)

▷ `DualIntersectionArray(CC)` (operation)

Returns: List

Alias for KreinArray.

3.8.9 IsQBipartite (for IsHomogeneousCoherentConfiguration)

▷ `IsQBipartite(CC)` (property)

Returns: true or false

Returns if the homogeneous coherent configuration CC is Q-bipartite.

3.8.10 IsQAntipodal (for IsHomogeneousCoherentConfiguration)

▷ `IsQAntipodal(CC)` (property)

Returns: true or false

Returns if the homogeneous coherent configuration CC is Q-antipodal.

3.9 Subsets

3.9.1 CharacteristicVector (for IsList, IsList)

▷ `CharacteristicVector(Omega, X)` (operation)

Returns: χ_X

Takes a subset X of Omega and returns the characteristic vector. The characteristic vector is a 0,1-vector indexed by the entries of Omega, with a 1 at position x if x is in X, and 0 otherwise.

3.9.2 CharacteristicVector (for IsList, IsPosInt)

▷ `CharacteristicVector(X, n)` (operation)

Returns: χ_X

Takes a subset X of of [1 .. n] and returns the characteristic vector chi_X.

3.9.3 InnerDistribution (for IsHomogeneousCoherentConfiguration, IsList)

▷ `InnerDistribution(v, CC)` (operation)

Returns: a

Returns the inner distribution *a* of a vector v with respect to the adjacency matrices of the coherent configuration CC. Note that v must be a vector over R^n where n is the order of CC. CC must be commutative.

3.9.4 MacWilliamsTransform (for IsHomogeneousCoherentConfiguration, IsList)

▷ `MacWilliamsTransform(v, CC)` (operation)

Returns: aQ

Returns the MacWilliams transform aQ of a vector v with respect to a coherent confiiguration CC. Takes either a vector v in R^n and converts it to its inner distribution vector first, or takes the inner distribution directly.

3.9.5 DualBoseMesnerBasis (for IsHomogeneousCoherentConfiguration, IsPosInt)

▷ `DualBoseMesnerBasis(CC, p)` (operation)

Returns: L

Returns a list L with the dual Bose-Mesner basis of a homogeneous coherent configuration with respect to the point p, such that $L_i = \tilde{E}_{i-1}$.

3.9.6 DualBoseMesnerBasis (for IsHomogeneousCoherentConfiguration)

▷ `DualBoseMesnerBasis(CC)` (operation)

Returns: L

Returns a list L with the dual Bose-Mesner basis of a homogeneous coherent configuration with respect to the first point, such that $L_i = \tilde{E}_{i-1}$.

3.9.7 OuterDistribution (for IsHomogeneousCoherentConfiguration, IsList)

▷ `OuterDistribution(v, CC)` (operation)

Returns: B

Returns the outer distribution B of a vector v with respect to the adjacency matrices of the coherent configuration CC. Note that v must be a vector over R^n where n is the order of CC. CC must be commutative.

3.9.8 DelsarteDesignType (for IsHomogeneousCoherentConfiguration, IsList)

▷ `DelsarteDesignType(CC, S)` (operation)

Returns: T

Returns T such that S is a Delsarte T-design, that is, $\chi_S E_i = 0$ for all $i \in T$. S must be a subset of the vertices.

3.9.9 WeightedDelsarteDesignType (for IsHomogeneousCoherentConfiguration, IsList)

▷ `WeightedDelsarteDesignType(CC, S)` (operation)

Returns: T

Returns T such that v is a (weighted) Delsarte T-design, that is, $vE_i = 0$ for all $i \in T$. v must be a weighted characteristic vector with respect to the vertices.

3.9.10 IsDelsarteTDesign (for IsHomogeneousCoherentConfiguration, IsList, IsList)

▷ `IsDelsarteTDesign(CC, S)` (operation)

Returns: true or false

Checks that S is a (weighted) Delsarte T-design, that is, $\chi_S E_i = 0$ for all $i \in T$. S must be either a subset of the vertices, or a weighted characteristic vector with respect to the vertices.

3.9.11 DualDegreeSet (for IsHomogeneousCoherentConfiguration, IsList)

▷ `DualDegreeSet(CC, S)` (operation)

Returns: K

Returns the dual degree set K for a set S, that is, $\chi_S E_i \neq 0$ for all $i \in K$. S must be a subset of the vertices.

3.9.12 WeightedDualDegreeSet (for IsHomogeneousCoherentConfiguration, IsList)

▷ `WeightedDualDegreeSet(CC, S)` (operation)

Returns: K

Returns the dual degree set K for a set S, that is, $vE_i \neq 0$ for all $i \in K$. v must be a (weighted) characteristic vector with restpect to the vertices.

3.9.13 AreDesignOrthogonal (for IsHomogeneousCoherentConfiguration, IsList, IsList)

▷ `AreDesignOrthogonal(CC, S1, S2)` (operation)

Returns: true or false

If S1 and S2 are either subsets of vertices, or (weighted) characteristic vectors, this checks that they are design orthogonal, that is, their dual degree sets are disjoint.

3.10 Approximations

3.10.1 ApproximateRealMinimalIdempotent (for IsHomogeneousCoherentConfiguration, IsInt)

▷ `ApproximateRealMinimalIdempotent(`*`CC, i`*`)` (operation)

Returns: approximation of E_i

Returns the i-th idempotent with entries approximated by floats. All entries must be real values.

3.10.2 ApproximateRealMinimalIdempotents (for IsHomogeneousCoherentConfiguration)

▷ `ApproximateRealMinimalIdempotents(`*`CC`*`)` (operation)

Returns: approximation of minimal idempotents

Returns a list of idempotents with entries approximated by floats. All entries must be real values.

3.10.3 ApproximateRealMatrixOfEigenvalues (for IsHomogeneousCoherentConfiguration)

▷ `ApproximateRealMatrixOfEigenvalues(`*`CC`*`)` (operation)

Returns: approximation of matrix of eigenvalues

Returns the matrix of eigenvalues with entries approximated by floats. All entries must be real values.

3.11 Algebras

3.11.1 BoseMesnerAlgebra (for IsHomogeneousCoherentConfiguration)

▷ `BoseMesnerAlgebra(`*`CC`*`)` (operation)

Returns: A

Returns the Bose-Mesner algebra A of a homogeneous coherent configuration.

3.11.2 AdjacencyAlgebra (for IsHomogeneousCoherentConfiguration)

▷ `AdjacencyAlgebra(`*`CC`*`)` (operation)

Returns: A

Returns the adjacency algebra A of a homogeneous coherent configuration. This is an alias for BoseMesnerAlgebra.

3.11.3 TerwilligerAlgebra (for IsHomogeneousCoherentConfiguration, IsInt)

▷ `TerwilligerAlgebra(`*`CC, p`*`)` (operation)

Returns: T

Returns the Terwilliger algebra T of a homogeneous coherent configuration with respect to the point p.

3.11.4 TerwilligerAlgebra (for IsHomogeneousCoherentConfiguration)

▷ `TerwilligerAlgebra(`*CC*`)` (operation)

Returns: T

Returns the Terwilliger algebra T of a homogeneous coherent configuration with respect to the first point.

Chapter 4

Intersection Algebra objects

4.1 Core functionality

4.1.1 HasCyclotomicSplittingField (for IsHomogeneousCoherentConfiguration)

▷ `HasCyclotomicSplittingField(CC)` (property)

Returns: true or false

Returns if the CC has a cyclotomic splitting field

4.1.2 IntersectionAlgebra (for IsList)

▷ `IntersectionAlgebra(B)` (operation)

Returns: homogeneous coherent configuration

Takes a list of intersection matrices, B, and returns the Intersection Algebra. The intersection matrix $(B_i)_{jk} = p^k_{ij}$ must be for valid intersection numbers p^k_{ij} for some homogeneous coherent configuration.

4.1.3 IntersectionMatrices (for IsIntersectionAlgebraObject)

▷ `IntersectionMatrices(CC)` (attribute)

Returns: L

Returns a list L of the intersection matrices of an intersecgion algebra object, *CC*, where the i-th entry of L is B_{i-1} and $(B_i)_{jk} = p^k_{ij}$.

4.1.4 IntersectionNumber (for IsIntersectionAlgebraObject, IsInt, IsInt, IsInt)

▷ `IntersectionNumber(CC, i, j, k)` (operation)

Returns: p^k_{ij}

Returns the intersection number p^k_{ij} for a intersection algebra.

4.1.5 NumberOfClasses (for IsIntersectionAlgebraObject)

▷ `NumberOfClasses(CC)` (attribute)

Returns: d

Returns d for a d-class intersection algebra.

4.1.6 Valencies (for IsIntersectionAlgebraObject)

▷ `Valencies(CC)` (attribute)

Returns: L

Returns a list L of valencies of a coherent configuration CC. The i-th entry of L is k_{i-1}.

4.1.7 Order (for IsIntersectionAlgebraObject)

▷ `Order(CC)` (attribute)

Returns: n

Returns the order n (number of vertices) of the intersection algebra.

4.1.8 SplittingField (for IsIntersectionAlgebraObject)

▷ `SplittingField(CC)` (attribute)

Returns: F

Returns the splitting field of the CC

4.1.9 HasCyclotomicSplittingField (for IsIntersectionAlgebraObject)

▷ `HasCyclotomicSplittingField(CC)` (property)

Returns: true or false

Returns if the CC has a cyclotomic splitting field

4.1.10 HasRationalSplittingField (for IsIntersectionAlgebraObject)

▷ `HasRationalSplittingField(CC)` (property)

Returns: true or false

Returns true if the splitting field is the rationals, false otherwise.

4.1.11 KreinParameter (for IsIntersectionAlgebraObject, IsInt, IsInt, IsInt)

▷ `KreinParameter(CC, i, j, k)` (operation)

Returns: $q_{i,j}^{k}$

Compute the krein parameter $q_{i,j}^{k}$ of a commutative intersection algebra.

4.1.12 KreinParameters (for IsIntersectionAlgebraObject)

▷ `KreinParameters(CC)` (attribute)

Returns: L

Return a list L of all Krein parameters of a commutative intersection algebra, where $L[i][j,k] = q_{i,j}^{k}$.

4.1.13 IsQBipartite (for IsIntersectionAlgebraObject)

▷ `IsQBipartite(CC)` (property)

Returns: true or false

Returns if the intersection algebra CC is Q-bipartite.

4.1.14 IsPBipartite (for IsIntersectionAlgebraObject)

▷ `IsPBipartite(CC)` (property)

Returns: true or false

Returns if the intersection algebra CC is bipartite.

4.1.15 IsQAntipodal (for IsIntersectionAlgebraObject)

▷ `IsQAntipodal(CC)` (property)

Returns: true or false

Returns if the intersection algebra CC is Q-antipodal.

4.1.16 IsPAntipodal (for IsIntersectionAlgebraObject)

▷ `IsPAntipodal(CC)` (property)

Returns: true or false

Returns if the intersection algebra CC is antipodal.

4.1.17 ReorderRelations (for IsIntersectionAlgebraObject, IsList)

▷ `ReorderRelations(CC, L)` (operation)

Returns: coherent configuration

Takes a intersection algebra CC and a list L, where L is a reordering of the relations. Returns an intersection algebra where the i-th relation of the CC becomes the j-th relation in the intersection algebra, where $j = L_i$. Note that L_i must be equal to $\{0,\dots,d\}$ as a set, and additionally requires that $L_1 = 0$.

4.1.18 ReorderMinimalIdempotents (for IsIntersectionAlgebraObject, IsList)

▷ `ReorderMinimalIdempotents(CC, L)` (operation)

Returns: coherent configuration

Takes an intersection algebra CC and a list L, where L is a reordering of the minimal idempotents. Returns an intersection algebra where the i-th idempotent of the CC becomes the j-th idempotent in the new intersection algebra, where $j = L_i$. Note that L_i must be equal to $\{0,\dots,d\}$ as a set, and additionally requires that $L_1 = 0$.

4.1.19 ViewRelationDistributionDiagram (for IsIntersectionAlgebraObject)

▷ `ViewRelationDistributionDiagram(CC)` (operation)

Returns: true (Displays relation distribution diagram)

Take a CC and display the relation-distribution diagram with respect to R_1.

4.1.20 IsCommutative (for IsIntersectionAlgebraObject)

▷ `IsCommutative(CC)` (property)

Returns: true or false

Checks if the input is a commutative intersection algebra.

4.1.21 IsSymmetricIntersectionAlgebra (for IsIntersectionAlgebraObject)

▷ `IsSymmetricIntersectionAlgebra(CC)` (property)

Returns: true or false

Checks if the input is a symmetric intersection algebra.

4.1.22 NumberOfCharacters (for IsIntersectionAlgebraObject)

▷ `NumberOfCharacters(CC)` (attribute)

Returns: n

Returns the number n of characters of CC.

4.2 Constructor methods

4.2.1 IntersectionAlgebraFromMatrixOfEigenvalues (for IsMatrix)

▷ `IntersectionAlgebraFromMatrixOfEigenvalues(P)` (operation)

Returns: intersection algebra object

Returns the intersection algebra determined by a matrix of eigenvalues.

4.2.2 HammingSchemeIntersectionAlgebra (for IsPosInt, IsPosInt)

▷ `HammingSchemeIntersectionAlgebra(n, q)` (operation)

Returns: intersection algebra objectn

Returns the intersection algebra of the Hamming scheme, $H(n,q)$.

4.2.3 GrassmanSchemeIntersectionAlgebra (for IsPosInt, IsPosInt, IsPosInt)

▷ `GrassmanSchemeIntersectionAlgebra(n, k, q)` (operation)

Returns: intersection algebra object

Returns the intersection algebra of the Grassmann scheme, $J_q(n,k)$.

4.2.4 CyclotomicSchemeIntersectionAlgebra (for IsPosInt, IsPosInt)

▷ `CyclotomicSchemeIntersectionAlgebra(n, d)` (operation)

Returns: intersection algebra object

Returns the intersection algebra of the Cyclotomic scheme, $Cyc(n,d)$.

4.2.5 IntersectionAlgebraFromIntersectionArray (for IsList)

▷ `IntersectionAlgebraFromIntersectionArray(n, k, q)` (operation)

Returns: intersection algebra object

Returns the intersection algebra of a DRG given by its intersection array.

4.2.6 IntersectionAlgebraFromClassicalParameters (for IsList)

▷ `IntersectionAlgebraFromClassicalParameters(n, k, q)` (operation)

Returns: intersection algebra object

Returns the intersection algebra of a DRG given by its classical parameters.

4.2.7 IntersectionAlgebraFromStronglyRegularGraphParameters (for IsPosInt, IsPosInt, IsInt, IsInt)

▷ `IntersectionAlgebraFromStronglyRegularGraphParameters(`*n, k, q*`)` (operation)

Returns: intersection algebra object

Returns the intersection algebra of a SRG given by its parameters $[n,k,\lambda,\mu]$.

4.2.8 IsFusionOfHomogeneousCoherentConfiguration (for IsIntersectionAlgebraObject, IsList)

▷ `IsFusionOfHomogeneousCoherentConfiguration(`*CC, L*`)` (operation)

Returns: true or false

Takes the intersection algebra object of a d-class homogeneous coherent configuration CC, and checks if the partion L of $\{0,\dots,d\}$ corresponds to a valid fusion.

4.2.9 SchurianSchemeIntersectionAlgebra (for IsPermGroup)

▷ `SchurianSchemeIntersectionAlgebra(`*G*`)` (operation)

Returns: intersection algebra

Returns the Schurian scheme defined by G, where G is a transitive permutation group. A Schurian scheme is a special case of CoherentConfigurationByOrbitals and is symmetric.

4.3 Automorphisms and maps

4.3.1 MapFromIntersectionMatricesToCentralIdempotents (for IsIntersectionAlgebraObject)

▷ `MapFromIntersectionMatricesToCentralIdempotents(`*I*`)` (attribute)

Returns: M

Takes an intersection algebra object, I, and returns a matrix, M, which maps the intersection matrices to the central idempotents of the intersection algebra. The central idempotent $F_i = \sum_{i=1}^{d+1} M_{ij}B_j$.

4.3.2 MapFromIntersectionMatricesToCentralIdempotentsOverRationals (for IsIntersectionAlgebraObject)

▷ `MapFromIntersectionMatricesToCentralIdempotentsOverRationals(`*I*`)` (attribute)

Returns: M

Takes an intersection algebra object, I, and returns a matrix, M, which maps the intersection matrices to the central idempotents of the intersection algebra over the rationals. The central idempotent $F_i = \sum_{i=1}^{d+1} M_{ij}B_j$.

4.3.3 ImageOfIntersectionAlgebra (for IsIntersectionAlgebraObject, IsPerm)

▷ `ImageOfIntersectionAlgebra(`*A, σ*`)` (operation)

Returns: true A^σ

Take a d-class intersection algebra A and return its image under the permutation $\sigma \in Sym([1..d])$. If p_{ij}^k is an intersection number of A, then in the image the intersection number is $p_{i^\sigma j^\sigma}^{k^\sigma}$

4.3.4 IsomorphismIntersectionAlgebras (for IsIntersectionAlgebraObject, IsIntersectionAlgebraObject)

▷ `IsomorphismIntersectionAlgebras($A$, $B$)` (operation)

Returns: σ

Take two d-class intersection algebras A and B and return $\sigma \in Sym([1..d])$ such that $A^{\sigma} = B$.

4.3.5 AreIsomorphicIntersectionAlgebras (for IsIntersectionAlgebraObject, IsIntersectionAlgebraObject)

▷ `AreIsomorphicIntersectionAlgebras($A$, $B$)` (operation)

Returns: σ

Take two d-class intersection algebras A and B and return true if they are isomorphic. Return false otherwise.

4.3.6 CanonisingMap (for IsIntersectionAlgebraObject)

▷ `CanonisingMap(A)` (attribute)

Returns: perm

Returns two permutations which will produce the canonical form of the intersection algebra A. The canonical form can be obtained by ImageOfIntersectionAlgebra(A, perm) Any intersection algebra which is isomorphic to A will the same canonical form.

4.3.7 CanonicalFormOfIntersectionAlgebra (for IsIntersectionAlgebraObject)

▷ `CanonicalFormOfIntersectionAlgebra(A)` (operation)

Returns: B

Returns the canonical form, B, of the intersection algebra A. Any intersection algebra which is isomorphic to A will have B as the canonical form.

4.3.8 AutomorphismGroup (for IsIntersectionAlgebraObject)

▷ `AutomorphismGroup(A)` (attribute)

Returns: G

Returns the automorphism group G of the intersection algebra object A. G is a permutation group acting on the relations, such that for all $g \in G$, $p_{i^g j^g}^{k^g} = p_{ij}^{k}$.

4.4 Fusions

4.4.1 IsFusionOfIntersectionAlgebra (for IsIntersectionAlgebraObject, IsList)

▷ `IsFusionOfIntersectionAlgebra(CC, L)` (operation)

Returns: true or false

Takes a d-class homogeneous coherent configuration CC, and checks if the partion L of $\{0,\ldots,d\}$ corresponds to a valid fusion.

4.4.2 FusionOfIntersectionAlgebra (for IsIntersectionAlgebraObject, IsList)

▷ `FusionOfIntersectionAlgebra(CC, L)` (operation)

Returns: homogeneous coherent configuration

Takes a d-class homogeneous coherent configuration CC and returns a fusion scheme corresponding to L, where L is a partion of $\{0,\ldots,d\}$. Note that the ordering of the cells of L is irrelevant. The method will sort the fused relations according to the smallest value in each cell.

4.4.3 ConverseRelationPairs (for IsIntersectionAlgebraObject)

▷ `ConverseRelationPairs(CC)` (attribute)

Returns: L

Returns a list L of either tuples or singletons, corresponding to relations and their converses or symmetric relations.

4.4.4 ConverseRelation (for IsIntersectionAlgebraObject, IsInt)

▷ `ConverseRelation(CC, i)` (operation)

Returns: j

Returns j such that $R_j = R_i^\top$, the converse relation of i .

4.4.5 IsStratifiable (for IsIntersectionAlgebraObject)

▷ `IsStratifiable(CC)` (attribute)

Returns: true or false

If the fusion of transposed relations produces a valid association scheme, then CC is stratifiable.

4.4.6 SymmetrisationOfIntersectionAlgebra (for IsIntersectionAlgebraObject)

▷ `SymmetrisationOfIntersectionAlgebra(CC)` (operation)

Returns: Association scheme

Given a homogeneous coherent configuration, CC, the symmetrisation is computed if possible, otherwise fail is returned. The symmetrisation of a homogeneous coherent configuration takes any non-symmetric relations and fuses them together. The result may or may not be a valid homogeneous coherent configuration. If it is valid, then it is an association scheme (Symmetric coherent configuration). If CC is commutative, then it can be symmetrised.

4.4.7 FeasibleNonTrivialFusionsOfIntersectionAlgebra (for IsIntersectionAlgebraObject)

▷ `FeasibleNonTrivialFusionsOfIntersectionAlgebra(CC[, k[, flag]])` (attribute)

Returns: list of feasibly fusionable relations

Returns a list where each entry is a collection of relations which may be fused to form a feasible homogeneous coherent configuration Trivial means either no relations are fused, or all non-identity relations are fused. If the additional argument k is given, only fusions with k-classes are returned. If flag is also given and is equal to true, then all fusions with at most k-classes are returned.

4.4.8 AllNonTrivialFusionsOfIntersectionAlgebra (for IsIntersectionAlgebraObject)

▷ `AllNonTrivialFusionsOfIntersectionAlgebra(`*CC*`)` (operation)

Returns: List of all non-trivial fusions of CC

Returns a list of all homogeneous coherent configurations such that each element of the list is a non-trivial fusion of CC. Trivial means either no relations are fused, or all non-identity relations are fused. If the additional argument k is given, only fusions with k-classes are returned. If flag is also given and is equal to true, then all fusions with at most k-classes are returned.

4.4.9 AllFusionsOfIntersectionAlgebra (for IsIntersectionAlgebraObject)

▷ `AllFusionsOfIntersectionAlgebra(`*CC*`)` (operation)

Returns: List of all fusions of CC

Returns a list of all homogeneous coherent configurations such that each element of the list is a fusion of CC. Includes trivial fusions, i.e the original homogeneous coherent configuration, and the coherent configuration resulting from the fusion of all non-identity relations

4.4.10 FeasibleNonTrivialSymmetricFusionsOfIntersectionAlgebra (for IsIntersectionAlgebraObject)

▷ `FeasibleNonTrivialSymmetricFusionsOfIntersectionAlgebra(`*CC*`)` (attribute)

Returns: list of feasibly fusionable relations

Returns a list where each entry is a collection of relations which may be fused to form a feasible association scheme (i.e. relations are symmetric) Trivial means either no relations are fused, or all non-identity relations are fused. If the additional argument k is given, only fusions with k-classes are returned. If flag is also given and is equal to true, then all fusions with at most k-classes are returned.

4.4.11 AllNonTrivialSymmetricFusionsOfIntersectionAlgebra (for IsIntersectionAlgebraObject)

▷ `AllNonTrivialSymmetricFusionsOfIntersectionAlgebra(`*CC*`)` (operation)

Returns: List of all non-trivial fusions of CC

Returns a list of all association schemes (i.e symmetric relations) such that each element of the list is a non-trivial fusion of CC. Trivial means either no relations are fused, or all non-identity relations are fused. If the additional argument k is given, only fusions with k-classes are returned. If flag is also given and is equal to true, then all fusions with at most k-classes are returned.

4.4.12 AllSymmetricFusionsOfIntersectionAlgebra (for IsIntersectionAlgebraObject)

▷ `AllSymmetricFusionsOfIntersectionAlgebra(`*CC*`)` (operation)

Returns: List of all fusions of CC

Returns a list of all association schemes (i.e symmetric relations) such that each element of the list is a fusion of CC. Includes trivial fusions, i.e the original homogeneous coherent configuration (if it is an association scheme), and the coherent configuration resulting from the fusion of all non-identity relations

4.4.13 FirstFeasibleNonTrivialSymmetricFusionOfIntersectionAlgebra (for IsIntersectionAlgebraObject)

▷ `FirstFeasibleNonTrivialSymmetricFusionOfIntersectionAlgebra(CC)` (attribute)

Returns: list of feasibly fusionable relations

Returns a list of relations which may be fused to form a feasible association scheme (i.e. relations are symmetric), if possible. Returns fail if no nontrivial fusion exists. Trivial means either no relations are fused, or all non-identity relations are fused.

4.5 Intersection algebra

4.5.1 CentralIdempotentsOfIntersectionAlgebra (for IsIntersectionAlgebraObject)

▷ `CentralIdempotentsOfIntersectionAlgebra(I)` (attribute)

Returns: central idempotents

Returns the central idempotents of the intersection algebra.

4.5.2 CentralIdempotentsOfIntersectionAlgebraOverRationals (for IsIntersectionAlgebraObject)

▷ `CentralIdempotentsOfIntersectionAlgebraOverRationals(I)` (attribute)

Returns: central idempotents

Returns the central idempotents of the intersection algebra over the rationals.

4.5.3 MatrixOfDualEigenvalues (for IsIntersectionAlgebraObject)

▷ `MatrixOfDualEigenvalues(CC)` (attribute)

Returns: Q

Returns a the dual matrix of eigenvalues, Q, for a homogeneous coherent configuration CC.

4.5.4 MatrixOfEigenvalues (for IsIntersectionAlgebraObject)

▷ `MatrixOfEigenvalues(CC)` (attribute)

Returns: P

Returns a the matrix of eigenvalues (or character table), P, for the intersection algebra of a homogeneous coherent configuration CC.

4.5.5 FitMatrixOfEigenvalues (for IsIntersectionAlgebraObject, IsMatrix)

▷ `FitMatrixOfEigenvalues(A, P)` (operation)

Returns: P2

Checks if P is the matrix of eigenvalues of intersection algebra A, upto some reordering of the columns. In such a case, P2, the reordered matrix is returned. If not, returns fail.

4.6 Metric schemes

4.6.1 IsPPolynomial (for IsIntersectionAlgebraObject)

▷ `IsPPolynomial(CC)` (property)

Returns: true or false

Returns if the homogeneous coherent configuration CC is P-polynomial.

4.6.2 FirstPPolynomialOrdering (for IsIntersectionAlgebraObject)

▷ `FirstPPolynomialOrdering(CC)` (attribute)

Returns: P-polynomial ordering or fail

Returns the first P-polynomial ordering admitted by the homogeneous coherent configuration CC, and fail otherwise.

4.6.3 AdmitsPPolynomialOrdering (for IsIntersectionAlgebraObject)

▷ `AdmitsPPolynomialOrdering(CC)` (property)

Returns: true or false

Returns if the homogeneous coherent configuration CC admits a P-polynomial ordering.

4.6.4 IsMetric (for IsIntersectionAlgebraObject)

▷ `IsMetric(CC)` (operation)

Returns: true or false

Alias for IsPPolynomial.

4.6.5 FirstMetricOrdering (for IsIntersectionAlgebraObject)

▷ `FirstMetricOrdering(CC)` (operation)

Returns: metric ordering or fail

Alias for FirstPPolynomialOrdering.

4.6.6 AdmitsMetricOrdering (for IsIntersectionAlgebraObject)

▷ `AdmitsMetricOrdering(CC)` (operation)

Returns: true or false

Alias for AdmitsPPolynomialOrdering.

4.6.7 AllPPolynomialOrderings (for IsIntersectionAlgebraObject)

▷ `AllPPolynomialOrderings(CC)` (attribute)

Returns: L

Calculate the list *L* of all P-polynomial orderings of a homogeneous coherent configuration.

4.6.8 AllMetricOrderings (for IsIntersectionAlgebraObject)

▷ `AllMetricOrderings(CC)` (operation)

Returns: L

Alias for AllPPolynomialOrderings.

4.6.9 IntersectionArray (for IsIntersectionAlgebraObject)

▷ `IntersectionArray(CC)` (attribute)

Returns: List

Returns the intersection array if CC is P-polynomial.

4.6.10 ClassicalParameters (for IsIntersectionAlgebraObject)

▷ `ClassicalParameters(CC)` (attribute)

Returns: [d, b, α, β]

Returns the classical parameters if the CC is metric with classical parameters.

4.6.11 StronglyRegularGraphParameters (for IsIntersectionAlgebraObject)

▷ `StronglyRegularGraphParameters(CC)` (attribute)

Returns: [d, b, α, β]

Returns the parameters $\{n,k;\lambda,\mu\}$ if the CC is an association scheme with 2 classes.

4.7 Cometric schemes

4.7.1 AdmitsQPolynomialOrdering (for IsIntersectionAlgebraObject)

▷ `AdmitsQPolynomialOrdering(CC)` (property)

Returns: true or false

Returns if the homogeneous coherent configuration CC admits a Q-polynomial ordering.

4.7.2 AdmitsCometricOrdering (for IsIntersectionAlgebraObject)

▷ `AdmitsCometricOrdering(CC)` (operation)

Returns: true or false

Alias for AdmitsQPolynomialOrdering.

4.7.3 IsQPolynomial (for IsIntersectionAlgebraObject)

▷ `IsQPolynomial(CC)` (property)

Returns: true or false

Returns if the commutative coherent configuration CC is Q-polynomial.

4.7.4 IsCometric (for IsIntersectionAlgebraObject)

▷ `IsCometric(CC)` (operation)

Returns: true or false

Alias for is Q-polynomial.

4.7.5 AllQPolynomialOrderings (for IsIntersectionAlgebraObject)

▷ `AllQPolynomialOrderings(CC)` (attribute)

Returns: L

Calculate a list *L* of all Q-polynomial orderings of a homogeneous coherent configuration.

4.7.6 AllCometricOrderings (for IsIntersectionAlgebraObject)

▷ `AllCometricOrderings(CC)` (operation)

Returns: L

Alias for AllQPolynomialOrderings.

4.7.7 KreinArray (for IsIntersectionAlgebraObject)

▷ `KreinArray(CC)` (attribute)

Returns: List

Returns the Krein (or dual intersection) array if CC is Q-polynomial.

4.7.8 DualIntersectionArray (for IsIntersectionAlgebraObject)

▷ `DualIntersectionArray(CC)` (operation)

Returns: List

Alias for KreinArray.

Chapter 5

Examples

5.1 Example 1 – Constructing groups

In this example, we show how we can use coherent configurations to construct an entriely different almost simple permutation group from another one. We first show how $PSU(4,3)$ can be made out of its subgroup $PSL(3,4)$.

Example
```
gap> psl34 := PSL(3,4);;
gap> sylow3 := SylowSubgroup(psl34, 3);;
gap> normaliser := Normaliser(psl34, sylow3);;
gap> G := Image( FactorCosetAction(psl34, normaliser) );;
```

At this stage, we have constructed the unique permutation representation of degree 280, for $PSL(3,4)$.

Example
```
gap> A := HomogeneousCoherentConfigurationByOrbitals(G);
7-class homogeneous coherent configuration of order 280
gap> mat := RelationMatrix(A);;
gap> P := MatrixOfEigenvalues(A);;
gap> Print(P);
[ [ 1, 18, 18, 18, 72, 72, 72, 9 ], [ 1, 4, 4, 4, 16, -12, -12, -5 ],
  [ 1, -2, -2, 10, -8, 0, 0, 1 ], [ 1, -2, 10, -2, -8, 0, 0, 1 ],
  [ 1, 10, -2, -2, -8, 0, 0, 1 ],
  [ 1, -2, -2, -2, 0, -8*E(7)^3-8*E(7)^5-8*E(7)^6, -8*E(7)-8*E(7)^2-8*E(7)^4, -3 ],
  [ 1, -2, -2, -2, 0, -8*E(7)-8*E(7)^2-8*E(7)^4, -8*E(7)^3-8*E(7)^5-8*E(7)^6, -3 ],
  [ 1, -2, -2, -2, 7, -3, -3, 4 ] ]
```

We now take a particular fusion of this coherent configuration to obtain a 2-class association scheme.

Example
```
gap> valency18 := Filtered([1..7], j -> Number(mat[1], i -> i = j) = 18);
[1,2,3]
gap> fusions := List(Combinations(valency18,2), t ->
>                 FusionOfHomogeneousCoherentConfiguration(A, [[0], t,
>                         Difference([1..7],t)]) );;
```

Any of these three fusions will do:

Example
```
gap> autgroup := AutomorphismGroup( fusions[1] );;
gap> DisplayCompositionSeries( autgroup );
```

```
G (11 gens, size 26127360)
 | Z(2)
S (4 gens, size 13063680)
 | Z(2)
S (3 gens, size 6531840)
 | Z(2)
S (2 gens, size 3265920)
 | 2A(3,3) = U(4,3) ~ 2D(3,3) = O-(6,3)
1 (0 gens, size 1)
gap> socle := Socle(autgroup);;
gap> StructureDescription(socle);
"PSU(4,3)"
```

5.2 Example 2 – Dual polar spaces and their graphs

For this example, we also use the package FinInG [BBDB+18]. We will construct a metric association scheme coming from a dual polar space.

Example

```
gap> LoadPackage("FinInG", false);;
gap> quadric := EllipticQuadric(7, 2);
Q-(7, 2)
gap> points := AsList( Planes(quadric) );;
gap> mat := NullMat(Length(points), Length(points));;
gap> for i in [1..Length(points)] do
>        for j in [i+1..Length(points)] do
>                intersection := Meet( points{[i,j]} );
gap>                mat[i][j] := 2 - ProjectiveDimension( intersection );
gap>                mat[j][i] := mat[i][j];
gap>        od;
gap> od;
```

So far we have constructed the relation matrix arising from the dual polar space.

Example

```
gap> a := HomogeneousCoherentConfiguration( mat );
3-class association scheme of order 765
gap> P := MatrixOfEigenvalues(a);;
gap> Q := MatrixOfDualEigenvalues(a);;
gap> Display(P);
[ [    1,   28,  224,  512 ],
  [    1,   11,   20,  -32 ],
  [    1,   -7,   14,   -8 ],
  [    1,    1,  -10,    8 ] ]
gap> Display(Q);
[ [       1,      84,      204,      476 ],
  [       1,      33,      -51,       17 ],
  [       1,    15/2,     51/4,    -85/4 ],
  [       1,   -21/4,   -51/16,   119/16 ] ]
gap> IsPPolynomial(a);
true
gap> IsQPolynomial(a);
true
```

A simpler way (perhaps) uses the automorphism group of the ambient polar space:

Example
```
gap> cgroup := CollineationGroup(quadric);
PGO(-1,8,2)
gap> G := Action(cgroup, points);
<permutation group with 3 generators>
gap> a := SchurianAssociationScheme(G);
3-class homogeneous coherent configuration of order 765
gap> IsPPolynomial(a);
true
```

The automorphism group of the association scheme should be the same:

Example
```
gap> autgroup := AutomorphismGroup(a);;
gap> autgroup = G;
true
```

Now (for the purist!) we see if there are interesting subsets. Take a nondegenerate hyperplane section defining a parabolic quadric.

Example
```
gap> hyperplane := First(Hyperplanes(PG(7,2)), h ->
>                  TypeOfSubspace(quadric, h) = "parabolic");
<a proj. 6-space in ProjectiveSpace(7, 2)>
gap> insidehyp := Filtered(points, t -> t * hyperplane);;
gap> vector := CharacteristicVector(points, insidehyp);;
gap> dist := InnerDistribution(a, vector);
[ 1, 56, 64, 14 ]
gap> macw := MacWilliamsTransform(a, dist);
[ 135, 630, 0, 0 ]
```

Therefore, a hyperplane section gives rise to a design that is not a code, in this association scheme. Now we produce the dual polar graph.

Example
```
gap> P := MatrixOfEigenvalues(a);;
gap> Display(P);
[ [    1,  224,  512,   28 ],
  [    1,   20,  -32,   11 ],
  [    1,   14,   -8,   -7 ],
  [    1,  -10,    8,    1 ] ]
gap> position := Position(P[1], 28);
4
gap> M := AdjacencyMatrices(a)[ position ];;
gap> graph := Graph(G, [1..Order(a)], OnPoints, {x,y} -> M[x][y] = 1);;
gap> IsDistanceRegular(graph);
true
gap> GlobalParameters(graph);
[ [ 0, 0, 28 ], [ 1, 3, 24 ], [ 3, 9, 16 ], [ 7, 21, 0 ] ]
```

5.3 Example 3 – Codes

For this example, we use the package Guava[BBC+18] for its facility with block codes. We will see that the inner distribution vector of a subset coincides with the weight enumerator of a code when the association scheme is a Hamming scheme.

Example
```
gap> hammingscheme := HammingScheme(7,2);
7-class homogeneous coherent configuration of order 128
gap> LoadPackage("Guava", false);;
gap> hammingcode := HammingCode(3, GF(2));
a linear [7,4,3]1 Hamming (3,2) code over GF(2)
```

We now use an operation from Guava:

Example
```
gap> InnerDistribution(hammingcode);
[ 1, 0, 0, 7, 7, 0, 0, 1 ]
```

From the association scheme perspective ...

Example
```
gap> codewords := List( hammingcode, VectorCodeword );;
gap> vector := CharacteristicVector( AsList(GF(2)^7), codewords );;
gap> Collected(vector);
[ [ 0, 112 ], [ 1, 16 ] ]
gap> inndist := InnerDistribution( hammingscheme, vector);
[ 1, 0, 0, 7, 7, 0, 0, 1 ]
```

The MacWilliams transform coincides with the distribution vector of the dual code:

Example
```
gap> 1/16 * MacWilliamsTransform( hammingscheme, inndist);
[ 1, 0, 0, 0, 7, 0, 0, 0 ]
gap> dualcode := DualCode( hammingcode );
a linear [7,3,4]2..3 dual code
gap> InnerDistribution( dualcode );
[ 1, 0, 0, 0, 7, 0, 0, 0 ]
```

5.4 Example 4 – Using the library

In this package, we also have a library of all small homogeneous coherent configurations, of order at most 38 (except 31, ,35, 36, 37), corresponding to [HM].

Example
```
gap> for i in [5..20] do
>        Print(i,"    ",NumberOfHomogeneousCoherentConfigurations(i),"\n");
gap> od;
5    2
6    6
7    3
8    16
9    10
10    11
11    3
12    54
13    5
14    14
15    24
```

```
16    208
17    4
18    90
19    6
20    90
gap> order7 := List([1..3], i -> HomogeneousCoherentConfiguration(7, i));
 1-class homogeneous coherent configuration of order 7,
 2-class homogeneous coherent configuration of order 7,
 3-class homogeneous coherent configuration of order 7 ]
```

The first of these is trivial, so we look at the other two. The first arises from the Paley graph of order 7.

```
gap> a1 := order7[2];
2-class homogeneous coherent configuration of order 7
gap> autgroup := AutomorphismGroup(a1);
Group([ (2,3,4)(5,7,6), (1,2,3,5,4,6,7) ])
gap> StructureDescription(autgroup);
"C7 : C3"
```

The last one is a 3-class association scheme:

```
gap> a2 := order7[3];
3-class homogeneous coherent configuration of order 7
gap> IsAssociationScheme(a2);
true
gap> IsPPolynomial( a2 );
true
gap> IsPrimitive(a2);
rue
gap> Valencies(a2);
[ 1, 2, 2, 2 ]
gap> autgroup := AutomorphismGroup(a2);
Group([ (2,3)(4,5)(6,7), (1,2)(3,4)(5,6) ])
gap> StructureDescription(autgroup);
"D14"
gap> P := MatrixOfEigenvalues(a2);;
gap> Display(P);
[ [              1,              2,              2,              2 ],
  [              1,  E(7)^3+E(7)^4,    E(7)+E(7)^6,  E(7)^2+E(7)^5 ],
  [              1,  E(7)^2+E(7)^5,  E(7)^3+E(7)^4,    E(7)+E(7)^6 ],
  [              1,    E(7)+E(7)^6,  E(7)^2+E(7)^5,  E(7)^3+E(7)^4 ] ]
gap> AllPPolynomialOrderings(a2);
[ [ 0, 1, 2, 3 ], [ 0, 2, 3, 1 ], [ 0, 3, 1, 2 ] ]
gap> AdmitsQPolynomialOrdering(a2);
true
gap> AllQPolynomialOrderings(a2);
[ [ 0, 1, 3, 2 ], [ 0, 2, 1, 3 ], [ 0, 3, 2, 1 ] ]
```

5.5 Example 5 – Constructing HS (advanced example)

We redo an example that appears in Section 3.6 of Peter Cameron's "Permutation Groups" book [Cam99] and construct the Higman-Sims group.

First we construct the Hoffman-Singleton graph from the alternating group of degree 7.

Example

```
gap> A7 := AlternatingGroup(7);;
gap> Pi := [ [ 1, 2, 4 ], [ 1, 3, 7 ], [ 1, 5, 6 ],
>  [ 2, 3, 5 ], [ 2, 6, 7 ], [ 3, 4, 6 ], [ 4, 5, 7 ] ];;
gap> OnSetsRecursive := function(x,g)
>        if not IsSet(x) then
>                return x^g;
gap>        else
>                return Set(x,y->OnSetsRecursive(y,g));
gap>        fi;
gap> end;;
gap> triples := Combinations([1..7], 3);;
gap> allFanos := Orbit(A7, Pi, OnSetsSets);;
gap> fifty := Concatenation(triples, allFanos);;
gap> A7action := Action(A7, fifty, OnSetsRecursive);
<permutation group with 2 generators>
gap> orbitals := Orbits(A7action, Combinations([1..50],2), OnSets);;
gap> List(orbitals, Size);
[ 210, 315, 70, 420, 105, 105 ]
```

We will now make a homogeneous coherent configuration from scratch, from these orbitals.

Example

```
gap> mat := NullMat(50,50);;
gap> for i in [1..50] do
>        for j in [i+1..50] do
>                pos := First([1..Length(orbitals)], k -> [i,j] in orbitals[k]);
gap>                   mat[i][j] := pos;
gap>                   mat[j][i] := pos;
gap>        od;
gap> od;
```

This is not a CC yet. We will fuse the relations of valency 3 and 4:

Example

```
gap> l := Collected(mat[1]);
[ [ 0, 1 ], [ 1, 12 ], [ 2, 18 ], [ 3, 4 ], [ 4, 12 ], [ 5, 3 ] ]
gap> to_fuse := Filtered([1..Length(l)], t -> l[t][2] in [3,4])-1;
[ 3, 5 ]
gap> to_fuse2 := Difference([1..6],to_fuse);
[ 1, 2, 4, 6 ]
gap> poly := InterpolatedPolynomial(Rationals, Concatenation([0], to_fuse,
>        to_fuse2), [0,1,1,2,2,2,2] );;
gap> newmat := List(mat, row -> List(row, x -> Value(poly,x)));;
gap> Collected(newmat[1]);
[ [ 0, 1 ], [ 1, 7 ], [ 2, 42 ] ]
```

This now leads us directly to the Hoffman-Singleton graph:

Example
```
gap> cc := HomogeneousCoherentConfiguration( newmat );
2-class association scheme of order 50
gap> autHoffSing := AutomorphismGroup( cc );
<permutation group with 7 generators>
gap> StructureDescription( autHoffSing );
"PSU(3,5) : C2"
```

We will now construct the Mesner-Higman-Sims graph

Example
```
gap> vals := Valencies(cc);
[ 1, 7, 42 ]
gap> adjmat := AdjacencyMatrices(cc)[ Position(vals, 42) ];;
gap> graph := Graph(autHoffSing, [1..50], OnPoints, {x,y} -> adjmat[x][y]=1);;
gap> one_coclique := CompleteSubgraphsOfGivenSize(graph, 15)[1];;
gap> all_cocliques := Orbit(autHoffSing,
>          Set(VertexNames(graph){one_coclique}), OnSets);;
gap> Size(all_cocliques);
100
gap> G := Action(autHoffSing, all_cocliques, OnSets);;
gap> a := SchurianAssociationScheme(G);
4-class homogeneous coherent configuration of order 100
```

Now fuse the relations with valencies 7 and 15 (and the complement)

Example
```
gap> vals := Valencies(a);
[ 1, 35, 42, 15, 7 ]
gap> to_fuse := Filtered([1..Length(vals)], t -> vals[t] in [7,15])-1;;
gap> to_fuse2 := Difference([1..4], to_fuse);;
gap> fusion := FusionOfHomogeneousCoherentConfiguration(a, [[0], to_fuse,
>          to_fuse2]);
2-class association scheme of order 100
gap> autgroup2 := AutomorphismGroup(fusion);
<permutation group with 10 generators>
gap> StructureDescription(autgroup2);
"HS : C2"
```

Chapter 6

Appendix

6.1 AssociationSchemes Links

- Homepage: http://www.jesselansdown.com/AssociationSchemes
- Issue tracker: https://github.com/jesselansdown/AssociationSchemes/issues
- DOI: 10.5281/zenodo.2634954

6.2 GAP Links

- Homepage: http://gap-system.org
- NautyTracesInterface: https://github.com/sebasguts/NautyTracesInterface

References

[BBC^{+}18] R. Baart, T. Boothby, J. Cramwinckel, J. Fields, D. Joyner, R. Miller, E. Minkes, E. Roijackers, L. Ruscio, and C. Tjhai. GUAVA, a gap package for computing with error-correcting codes, Version 3.14. `https://gap-packages.github.io/guava`, Mar 2018. Refereed GAP package. 49

[BBDB^{+}18] J. Bamberg, A. Betten, J. De Beule, P. Cara, M. Lavrauw, and M. Neunhoeffer. FinInG, finite incidence geometry, Version 1.4.1. `http://www.fining.org`, Mar 2018. Refereed GAP package. 48

[BI84] Eiichi Bannai and Tatsuro Ito. *Algebraic combinatorics. I.* The Benjamin/Cummings Publishing Co., Inc., Menlo Park, CA, 1984. Association schemes. 5

[Cam99] Peter J. Cameron. *Permutation Groups*. London Mathematical Society Student Texts. Cambridge University Press, 1999. 52

[GAP16] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.8.6*, 2016. 5

[God93] C. D. Godsil. *Algebraic combinatorics*. Chapman and Hall Mathematics Series. Chapman & Hall, New York, 1993. 5

[HM] Akihide Hanaki and Izumi Miyamoto. 10, 19, 50

Index