Elementary functions for association schemes on GAP

Akihide Hanaki

November 24, 2019

You can get GAP at `http://www.gap-system.org/`. We suppose the version of GAP is 4.x. Some functions need the package `grape 4.4`.

Read my file `association_scheme.gap` on GAP, `Read("association_scheme.gap")`. Then you can use the following functions.

We use notations and terminologies in [2] and [12]. Let `R` be a relation matrix.

We use the notation (X, S) for the association scheme with relation matrix `R`.

1 Basic functions

1.1 ClassOfAssociationScheme

`ClassOfAssociationScheme(R)` returns the class of `R`, namely $|S| - 1$. The numbers of the relations must be `[0..d]`.

1.2 Valency

`Valency(R, i)` returns the valency of the `i`-th relation of `R`, namely $p_{ii^*}^0$.

1.3 Valencies

`Valencies(R)` returns the list of valencies of `R`, namely the list $\{p_{ii^*}^0 \mid i = 0, 1, \cdots, d\}$.

1.4 SumOfValencies

`SumOfValencies(R, L)` returns the sum of the valencies in the list of relations `L`.

1.5 OrderOfScheme

`OrderOfScheme(R)` returns the order of `R`, namely $|X|$.

1.6 Relation

`Relation(R, x, y)` returns the relation r of `R` such that $(x, y) \in r$.

1.7 TransposedRelations

`TransposedRelations(R)` returns the list of transpositions of the relations.

1.8 SymmetricRelations

`SymmetricRelations(R)` returns the list of symmetric relations of `R`.

1.9 NonSymmetricRelations

`NonSymmetricRelations(R)` returns the list of non-symmetric relations of `R`.

1.10 Involutions

`Involutions(R)` returns the list of involutions of `R`. A relation `g` is called an *involution* if `[0, g]` is a closed subset.

1.11 AdjacencyMatrices

`AdjacencyMatrices(R)` returns the list of adjacency matrices of `R`.

1.12 SumOfAdjacencyMatrices

`SumOfAdjacencyMatrices(R, L)` returns the sum of adjacency matrices in the list of relation numbers `L`. For example, use `SumOfAdjacencyMatrices(R, [1,2])`.

1.13 IntersectionNumber

`IntersectionNumber(R, i, j, k)` returns the intersection number p_{ij}^k of `R`.

1.14 IntersectionMatrices

`IntersectionMatrices(R)` returns the list of the intersection matrices. Namely, the list of (p_{ij}^k).

1.15 IsAssociationScheme

`IsAssociationScheme(R)` returns whether `R` is the relation matrix of an association scheme.

1.16 ComplexProduct

`ComplexProduct(R, L1, L2)` returns the complex product of `L1` and `L2`. `L1` and `L2` must be lists of relation numbers.

1.17 HadamardProduct

`HadamardProduct(L1, L2)` returns the Hadamard product (entry-wise product) of matrices `L1` and `L2`.

1.18 Neighbors

`Neighbors(R, p, L)` returns the `L`-neighbors of a point `p`.

1.19 NrCharacters

`NrCharacters(R)` returns the number of the complex irreducible characters of `R`. Of course, this number is the dimension of the center of the adjacency algebra.

1.20 NumCharacters

Same as `NrCharacters`.

1.21 NrModularSimpleModules

`NrModularSimpleModules(R, p)` returns the number of simple modules of adjacency algebra over an algebraically closed field of characteristic p. The method is based on results by R. Brauer (see [7], for example).

1.22 AutomorphismGroupOfScheme

`AutomorphismGroupOfScheme(R)` returns the automorphism group of `R`.

1.23 AlgebraicAutomorphismGroupOfScheme

`AlgebraicAutomorphismGroupOfScheme(R)` returns the algebraic automorphism group of `R` (see [6]).

1.24 CoefficientsOfAdjacencyMatrices

Let `R` be an association scheme and let `F` be a field. If `F` is not given, then we assume that `F` is the rational number field. For an element `M` of the adjacency algebra of `R` over `F`, `CoefficientsOfAdjacencyMatrices(R, M [,F])` returns the coefficients of `M` with respect to the standard basis.

1.25 Examples

```
gap> R :=
> [ [ 0,1,2,2,2,3,3,3 ],
>   [ 1,0,3,3,3,2,2,2 ],
>   [ 3,2,0,2,3,1,2,3 ],
>   [ 3,2,3,0,2,2,3,1 ],
>   [ 3,2,2,3,0,3,1,2 ],
>   [ 2,3,1,3,2,0,3,2 ],
>   [ 2,3,3,2,1,2,0,3 ],
>   [ 2,3,2,1,3,3,2,0 ] ];;
gap> ClassOfAssociationScheme(R);
3
gap> Valency(R, 2);
3
gap> Valencies(R);
[ 1, 1, 3, 3 ]
gap> OrderOfScheme(R);
8
gap> Relation(R, 1, 3); Relation(R, 5, 4);
2
3
gap> TransposedRelations(R);
[ 1, 3, 2 ]
gap> SymmetricRelations(R);
[ 0, 1 ]
gap> NonSymmetricRelations(R);
[ 2, 3 ]
gap> Involutions(R);
[ 1 ]
gap> IntersectionNumber(R, 1, 2, 3);
1
gap> IntersectionMatrices(R);
[ [ [ 1, 0, 0, 0 ], [ 0, 1, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ],
  [ [ 0, 1, 0, 0 ], [ 1, 0, 0, 0 ], [ 0, 0, 0, 1 ], [ 0, 0, 1, 0 ] ],
  [ [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ], [ 0, 3, 1, 1 ], [ 3, 0, 1, 1 ] ],
  [ [ 0, 0, 0, 1 ], [ 0, 0, 1, 0 ], [ 3, 0, 1, 1 ], [ 0, 3, 1, 1 ] ] ]
gap> A := AdjacencyMatrices(R);
gap> CoefficientsOfAdjacencyMatrices(R, A[3] * A[4]);
[ 3, 0, 1, 1 ]
gap> NrCharacters(R);
4
gap> AutomorphismGroupOfScheme(R);
Group([ (3,4,5)(6,8,7), (1,2)(3,6)(4,8)(5,7), (1,3,2,6)(4,7,8,5) ])
gap> AlgebraicAutomorphismGroupOfScheme(R);
Group([ (2,3) ])
```

2 Construction of schemes

2.1 DirectProductScheme

`DirectProductScheme(R, S)` returns the direct product of `R` and `S`.

2.2 WreathProductScheme

`WreathProductScheme(R, S)` returns the wreath product of `R` and `S`.

2.3 TransitivePermutationGroupScheme

Let `G` be a finite group and `H` a subgroup of `G`. `TransitivePermutationGroupScheme(G, H)` returns the Schurian scheme defined by them. If `G` is given as a transitive permutation group, then use it as `TransitivePermutationGroupScheme(G, Stabilizer(G, 1))`.

2.4 SchurianScheme

Same as `TransitivePermutationGroupScheme`.

2.5 GroupAssociationScheme

Let `G` be a finite group. `GroupAssociationScheme(G)` returns the group association scheme of `G`.

2.6 NormalSubgroupScheme

Let `G` be a finite group and `H` a normal subgroup of `G`. `NormalSubgroupScheme(G, H)` returns the scheme defined by `G`-conjugacy classes of `H`.

2.7 CyclotomicScheme

Let `q` be a prime power, and `d` a divisor of $\mathtt{q}-1$. `CyclotomicScheme(q,d)` returns the cyclotomic scheme $Cyc(\mathtt{q},\mathtt{d})$.

2.8 HammingScheme

`HammingScheme(n, q)` returns the Hamming scheme $H(\mathtt{n},\mathtt{q})$.

2.9 JohnsonScheme

`JohnsonScheme(v, k)` returns the Johnson scheme $J(\mathtt{v},\mathtt{k})$.

2.10 CompleteGraphScheme

`CompleteGraphScheme(n)` returns the scheme of order `n` and of class `1`.

2.11 CompleteMultipartiteGraphScheme

`CompleteMultipartiteGraphScheme(n, m)` returns the scheme defined by the complete multipartite graph.

2.12 ShrinkhandeGraphScheme

`ShrinkhandeGraphScheme()` returns the associationscheme obtained by the Shrinkhande Graph. This is the scheme `as16[6]` in our list.

2.13 DoobGraphScheme

`DoobGraphScheme(m, n)` returns the associationscheme obtained by the Doob graph. It is the direct product of `m` copies of the Shrinkhande graphs and `n` copies of the complete graphs on four points.

2.14 FusionScheme

Let `R` have class `d`, and `L` a partition of `[1..d]`. `FusionScheme(R, L)` returns the fusion of `R` by `L`. For example, use it like as `FusionScheme(R, [[1],[2,3],[4,5,6]])`. `0` is always fixed. Note that this function does not check whether the result is a scheme. To do it, use the function `IsAssociationScheme`.

2.15 AlgebraicFusionOfScheme

Let `R` be a relation matrix of an association scheme, and let `G` be a subgroup of the algebraic automorphism group of `R` (see `AlgebraicAutomorphismGroupOfAssociationScheme`). `AlgebraicFusionOfAssociationScheme(R, G)` return the algebraic fusion of `R` by the action of `G`.

2.16 SymmetrizationOfScheme

`SymmetrizationOfScheme(R)` returns the symmetrization of `R`. This function works also for noncommutative scheme, but does not check whether the result is a scheme. To do it, use the function `IsAssociationScheme`.

2.17 SemidirectProductScheme

Let `R` be a relation matrix of size n, `G` a finite group, and `hom` a group homomorphism from `G` to the symmetric group of degree n. `SemidirectProductScheme(R, G, hom)` returns the semidirect product of `R` by `G` in the sense of [12, §2.7]. This function does not check whether `hom` is a homomorphism to the automorphism group.

For example,

```
gap> R := CompleteGraphScheme(3);;
gap> gens := [(1,2,3)];;
gap> H := Group(gens);;
gap> G := Group(gens);;
gap> hom := GroupHomomorphismByImages(G, H, gens, gens);;
gap> R2 := SemidirectProductScheme(R, G, hom);;
gap> Display(R2);
[ [  0,  1,  1,  3,  2,  3,  5,  5,  4 ],
  [  1,  0,  1,  3,  3,  2,  4,  5,  5 ],
  [  1,  1,  0,  2,  3,  3,  5,  4,  5 ],
  [  5,  5,  4,  0,  1,  1,  3,  2,  3 ],
  [  4,  5,  5,  1,  0,  1,  3,  3,  2 ],
  [  5,  4,  5,  1,  1,  0,  2,  3,  3 ],
  [  3,  2,  3,  5,  5,  4,  0,  1,  1 ],
  [  3,  3,  2,  4,  5,  5,  1,  0,  1 ],
  [  2,  3,  3,  5,  4,  5,  1,  1,  0 ] ]
gap> IsAssociationScheme(R2);
true
```

3 Some properties

3.1 IsPrimitiveScheme

`IsPrimitiveScheme(R)` returns whether `R` is primitive.

3.2 IsCommutativeScheme

`IsCommutativeScheme(R)` returns whether `R` is commutative.

3.3 IsSymmetricScheme

`IsSymmetricScheme(R)` returns whether `R` is symmetric.

3.4 IsThin

`IsThin(R)` returns whether `R` is thin.

3.5 IsQuasiThin

`IsQuasiThin(R)` returns whether `R` is quasi-thin.

3.6 IsResiduallyThin

`IsResiduallyThin(R)` returns whether `R` is residually thin.

3.7 IsSchurian

`IsSchurian(R)` returns whether `R` is Schurian.

3.8 IsGroupLikeScheme

`IsGroupLikeScheme(R)` returns whether `R` is a group-like scheme. A scheme is called *group-like* if the center of the adjacency algebra is closed by the Hadamard product (see [4]).

3.9 IsPValencedScheme

A scheme is called a *p-valenced scheme* if every valency is p-power. `IsPValencedScheme(R, p)` returns whether `R` is p-valenced.

3.10 IsPPrimeValencedScheme

A scheme is called a *p'-valenced scheme* if every valency is coprime to p. `IsPPrimeValencedScheme(R, p)` returns whether `R` is p'-valenced.

3.11 IsPiValencedScheme

A scheme is called a *π-valenced scheme* if every valency is a π-number. `IsPValencedScheme(R, pi)` returns whether `R` is `pi`-valenced, where `pi` is a set of prime numbers.

3.12 IsPScheme

A scheme is called a *p-scheme* if it is p-valenced and its order is p-power. `IsPScheme(R, p)` returns whether `R` is a p-scheme.

3.13 IsTriplyRegularAssociationScheme

`IsTriplyRegularAssociationScheme(R)` returns whether `R` is triply regular in the sense of [8].

4 Closed subsets

Let `L` be a list of relation numbers.

4.1 IsClosedSubset

`IsClosedSubset(R, L)` returns whether `L` is a closed subset of `R`.

4.2 IsNormalClosedSubset

`IsNormalClosedSubset(R, L)` returns whether `L` is a normal closed subset of `R`.

4.3 GeneratedClosedSubset

`GeneratedClosedSubset(R, L)` returns the list of relation numbers in the closed subset generated by the relations in `L`.

4.4 ThinRadical

`ThinRadical(R)` returns the list of relation numbers in the thin radical of `R`.

4.5 ThinResidue

`ThinResidue(R)` returns the list of relation numbers in the thin residue of `R`.

4.6 PartitionOfPointsByClosedSubset

`PartitionOfPointsByClosedSubset(R, L)` returns a partition of points determined by a closed subset `L`.

4.7 RelationMatrixSortedByClosedSubset

`RelationMatrixSortedByClosedSubset(R, L)` returns a relation matrix sorted by a partition determined by a closed subset `L`. For example,

```
gap> Display(R);
[ [  0,  1,  2,  2,  2,  2,  2,  3,  3,  3,  3,  3 ],
  [  1,  0,  3,  3,  3,  3,  3,  2,  2,  2,  2,  2 ],
  [  2,  3,  0,  2,  2,  3,  3,  1,  2,  2,  3,  3 ],
  [  2,  3,  2,  0,  3,  2,  3,  3,  2,  3,  1,  2 ],
  [  2,  3,  2,  3,  0,  3,  2,  3,  3,  2,  2,  1 ],
  [  2,  3,  3,  2,  3,  0,  2,  2,  3,  1,  3,  2 ],
  [  2,  3,  3,  3,  2,  2,  0,  2,  1,  3,  2,  3 ],
  [  3,  2,  1,  3,  3,  2,  2,  0,  3,  3,  2,  2 ],
  [  3,  2,  2,  2,  3,  3,  1,  3,  0,  2,  3,  2 ],
  [  3,  2,  2,  3,  2,  1,  3,  3,  2,  0,  2,  3 ],
  [  3,  2,  3,  1,  2,  3,  2,  2,  3,  2,  0,  3 ],
  [  3,  2,  3,  2,  1,  2,  3,  2,  2,  3,  3,  0 ] ]
gap> R2 := RelationMatrixSortedByClosedSubset(R, [0,1]);;
gap> Display(R2);
```

```
[ [  0,  1,  2,  3,  2,  3,  2,  3,  2,  3,  2,  3 ],
  [  1,  0,  3,  2,  3,  2,  3,  2,  3,  2,  3,  2 ],
  [  2,  3,  0,  1,  2,  3,  2,  3,  3,  2,  3,  2 ],
  [  3,  2,  1,  0,  3,  2,  3,  2,  2,  3,  2,  3 ],
  [  2,  3,  2,  3,  0,  1,  3,  2,  2,  3,  3,  2 ],
  [  3,  2,  3,  2,  1,  0,  2,  3,  3,  2,  2,  3 ],
  [  2,  3,  2,  3,  3,  2,  0,  1,  3,  2,  2,  3 ],
  [  3,  2,  3,  2,  2,  3,  1,  0,  2,  3,  3,  2 ],
  [  2,  3,  3,  2,  2,  3,  3,  2,  0,  1,  2,  3 ],
  [  3,  2,  2,  3,  3,  2,  2,  3,  1,  0,  3,  2 ],
  [  2,  3,  3,  2,  3,  2,  2,  3,  2,  3,  0,  1 ],
  [  3,  2,  2,  3,  2,  3,  3,  2,  3,  2,  1,  0 ] ]
```

4.8 Subscheme

`Subscheme(R, L [, p])` returns the subscheme of `R` at a point `p` by a closed subset `L`. If `p` is not given, then the function assume `p` to be `1`.

4.9 FactorScheme

`FactorScheme(R, L)` returns the factor scheme of `R` by a closed subset `L`.

4.10 QuotientScheme

Same as `FactorScheme`.

4.11 CommutatorOfScheme

`CommutatorOfScheme(R, s, t)` returns the commutator s^*t^*st.

4.12 CommutatorOfSubsets

tt CommutatorOfSubsets(R, S, T) returns the closed subset generated by commutators $\{s^*t^*st \mid s \in S,\ t \in T\}$. If S and T are whole of the scheme, then this closed subset is the smallest closed subset of R such that the quotient scheme is commutative thin.

4.13 CosetDecompositionOfScheme

`CosetDecompositionOfScheme(R, L)` returns the coset decomposition of the relations. `R` is a relation matrix and `L` must be a closed subset. This function does not check whether `L` is closed.

4.14 CosetRepresentativesOfScheme

`CosetRepresentativesOfScheme(R, L)` returns a representatives of coset decomposition of the relations. `R` is a relation matrix and `L` must be a closed subset. This function does not check whether `L` is closed.

We show an example.

```
gap> R :=
> [ [  0,  1,  2,  3,  4,  4,  4,  4,  5,  5,  5,  5 ],
>   [  1,  0,  3,  2,  4,  4,  4,  4,  5,  5,  5,  5 ],
>   [  2,  3,  0,  1,  5,  5,  5,  5,  4,  4,  4,  4 ],
>   [  3,  2,  1,  0,  5,  5,  5,  5,  4,  4,  4,  4 ],
>   [  4,  4,  5,  5,  0,  1,  4,  4,  2,  3,  5,  5 ],
>   [  4,  4,  5,  5,  1,  0,  4,  4,  3,  2,  5,  5 ],
>   [  4,  4,  5,  5,  4,  4,  0,  1,  5,  5,  2,  3 ],
>   [  4,  4,  5,  5,  4,  4,  1,  0,  5,  5,  3,  2 ],
>   [  5,  5,  4,  4,  2,  3,  5,  5,  0,  1,  4,  4 ],
>   [  5,  5,  4,  4,  3,  2,  5,  5,  1,  0,  4,  4 ],
>   [  5,  5,  4,  4,  5,  5,  2,  3,  4,  4,  0,  1 ],
>   [  5,  5,  4,  4,  5,  5,  3,  2,  4,  4,  1,  0 ] ];;
gap> IsClosedSubset(M, [0, 1]);
true
gap> CosetDecompositionOfScheme(M, [0,1]);
[ [ 0, 1 ], [ 2, 3 ], [ 4 ], [ 5 ] ]
gap> CosetRepresentativesOfScheme(M, [0,1]);
[ 0, 2, 4, 5 ]
```

5 *P*-Polynomial schemes

5.1 IsPPolynomialScheme

`IsPPolynomialScheme(R)` returns whether `R` is a P-polynomial scheme.

5.2 AllPPolynomialOrdering

`AllPPolynomialOrdering(R)` returns the list of P-polynomial orderings of the relations.

5.3 IntersectionArray

`IntersectionArray(R, a)` returns the intersection array of the P-polynomial scheme defined by the relation `a`.

5.4 DRG2PPolynomialScheme

Let `Gra` be a distance-regular graph. `DRG2PPolynomialScheme(Gra)` returns the relation matrix of the P-polynomial scheme defined by `Gra`.

5.5 AMofDRG2PPolynomialScheme

Let `A` be the adjacency matrix of a distance-regular graph. `AMofDRG2PPolynomialScheme(A)` returns the relation matrix of the P-polynomial scheme defined by `A`.

5.6 Example

For example, use them as the following.

```
gap> G := JohnsonScheme(5,2);;
gap> IsPPolynomialScheme(G);
true
gap> AllPPolynomialOrdering(G);
[ [ 1, 2 ], [ 2, 1 ] ]
gap> PrintArray(IntersectionArray(G, 1));
[ [  *,  1,  4 ],
  [  0,  3,  2 ],
  [  6,  2,  * ] ]
```

6 Character tables

6.1 CharacterTableOfAssociationScheme

`CharacterTableOfAssociationScheme(R [, F])` returns the character table of `R`, if every entry is in the field `F`. The field `F` must be of characteristic zero. If the table contains entries not in `F`, then this function returns `false`. If `F` is not given, then the function assume `F` the rational number field. The last column is a list of multiplicities. If you want to delete multiplicities, use it as `CharacterTableOfAssociationScheme(R, F, 1)`. If you will use the table for other functions, don't delete the multiplicities.

6.2 CharacterTableOfHammingScheme

`CharacterTableOfHammingScheme(n, q)` returns the character table of the Hamming scheme (`HammingScheme(n, q)`). The last column is a list of multiplicities. If you want to delete multiplicities, use it as `CharacterTableOfHammingScheme(v, k, 1)`. This function is faster than the general function `CharacterTableOfAssociationScheme`.

6.3 CharacterTableOfJohnsonScheme

`CharacterTableOfJohnsonScheme(v, k)` returns the character table of the Johnson scheme (`JohnsonScheme(v, k)`). The last column is a list of multiplicities. If you want to delete multiplicities, use it as `CharacterTableOfJohnsonScheme(v, k, 1)`. This function is faster than the general function `CharacterTableOfAssociationScheme`.

6.4 IsCharacterTableAS

`IsCharacterTableAS(R, T)` returns whether `T` is a character table of `R`.

6.5 RegularCharacterOfAssociationScheme

`RegularCharacterOfAssociationScheme(R)` returns the regular character of the scheme `R`. Namely, it returns the list of $\{\sum_{t \in S} p_{st}^t \mid s \in S\}$.

6.6 FrameQuotient

`FrameQuotient(T)` return the Frame quotient ([1], [3]) of the character table `T`. Note that `T` must be a character table, not a relation matrix.

6.7 FrameNumber

`FrameNumber(T)` return the Frame number ([1], [3]) of the character table `T`. Note that `T` must be a character table, not a relation matrix. It is known that the adjacency algebra over a field of characteristic p is semisimple if and only if p does not divide the Frame number.

6.8 FrobeniusSchurIndicatorAS

Let `R` be a relation matrix, `T` the character table. `FrobeniusSchurIndicatorAS(R, T, a)` returns the Frobenius-Schur indicator of the `a`-th irreducible character [5]. It is defined by

$$\nu_2(\chi) = \frac{m_\chi}{n_S \chi(1)} \sum_{s \in S} \frac{1}{n_s} \chi({\sigma_s}^2).$$

6.9 CentralPrimitiveIdempotentByCharacterTable

Let `R` be an association scheme, and let `T` be the character table of `R`. `CentralPrimitiveIdempotentByCha` `T, i)` returns the central primitive idempotent of the adjacency algebra corresponding to the `i`-th row of `T`.

6.10 CentralPrimitiveIdempotentsByCharacterTable

Let `R` be an association scheme, and let `T` be the character table of `R`. `CentralPrimitiveIdempotentsByCh` `T)` returns the list of central primitive idempotents of the adjacency algebra.

6.11 Example

```
gap> R1 := GroupAssociationScheme(SymmetricGroup(4));;
gap> T1 := CharacterTableOfAssociationScheme(R1);;
gap> Display(T1);
```

```
[ [   1,   6,   3,   8,   6,   1 ],
  [   1,  -6,   3,   8,  -6,   1 ],
  [   1,   0,   3,  -4,   0,   4 ],
  [   1,  -2,  -1,   0,   2,   9 ],
  [   1,   2,  -1,   0,  -2,   9 ] ]
gap> IsCharacterTableAS(R1, T1);
true
gap> FrameQuotient(T1);
36864
gap> FrameNumber(T1);
21233664
gap> FactorsInt(last);
[ 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3 ]
gap> R2 := CyclotomicScheme(5, 2);;
gap> CharacterTableOfAssociationScheme(R2);
false
gap> Display(CharacterTableOfAssociationScheme(R2, CF(5)));
[ [              1,              2,              2,              1 ],
  [              1,  E(5)^2+E(5)^3,    E(5)+E(5)^4,              2 ],
  [              1,    E(5)+E(5)^4,  E(5)^2+E(5)^3,              2 ] ]
gap> G := SymmetricGroup(5);;
gap> H := Subgroup(G, [(1,2,3,4)]);;
gap> R3 := SchurianScheme(G, H);;
gap> T3 := CharacterTableOfAssociationScheme(R3);;
gap> Display(T3);
[ [   1,   1,   4,   4,   4,   4,   4,   4,   4,   1 ],
  [   2,   0,  -2,  -2,  -2,   3,  -1,   3,  -1,   5 ],
  [   1,  -1,   0,  -3,   3,  -1,   1,  -1,   1,   4 ],
  [   1,   1,   4,  -1,  -1,  -1,  -1,  -1,  -1,   4 ],
  [   1,   1,  -2,   2,   2,  -1,  -1,  -1,  -1,   5 ],
  [   1,  -1,   0,   2,  -2,  -1,   1,  -1,   1,   6 ] ]
gap> FrameNumber(T3);
1074954240000
```

7 Q-Polynomial schemes

7.1 IsQPolynomialScheme

`IsQPolynomialScheme(T)` returns whether `T` is the character table of a Q-polynomial scheme. Note that this function requires the character table, and does not check whether the scheme is symmetric.

7.2 AllQPolynomialOrdering

`AllQPolynomialOrdering(T)` returns the list of Q-polynomial orderings of the characters. Note that this function requires the character table, and does not check whether the scheme is symmetric.

7.3 KreinParameters

`KreinParameters(T)` returns the Krein parameters of the character table `T`. Note that this function requires the character table. If the scheme is non-commutative, then it returns `false`.

8 Some algebras

8.1 AdjacencyAlgebra

`AdjacencyAlgebra(R [, F])` returns the adjacency algebra (Bose-Mesner algebra) of `R` over the field `F`. If `F` is not given, then the function assume that the coefficient field is the rational number field.

8.2 IntersectionAlgebra

`IntersectionAlgebra(R [, F])` returns the intersection algebra of `R` over the field `F`. If `F` is not given, then the function assume that the coefficient field is the rational number field. This algebra is isomorphic to the adjacency algebra, but the size of matrices is smaller.

8.3 TerwilligerAlgebra

`TerwilligerAlgebra(R, [, F [, p]])` returns the Terwilliger algebra (subconstituent algebra [9], [10], [11]) of `R` over the field `F` at the point `p`. If `F` is not given, then the function assume that the coefficient field is the rational number field. If `p` is not given, then the function assume that the point is `1`.

8.4 CenterOfAdjacencyAlgebra

`CenterOfAdjacencyAlgebra(R [, F])` returns the center of the adjacency algebra of `R` over the field `F`. If `F` is not given, then the function assume that the coefficient field is the rational number field.

9 Rational integers and rational numbers

9.1 IsPrimePower

`IsPrimePower(n, p)` returns whether `n` is a power of the prime number `p`. See also the GAP's function `IsPrimePowerInt`.

9.2 P_Valuation

For an integer `n` and a rational prime number `p`, `P_Valuation(n, p)` returns the value of the `p`-valuation of `n`. For example,

```
gap> P_Valuation(12, 2);
2
```

9.3 P_ValuationRat

For a rational number `r` and a rational prime number `p`, `P_ValuationRat(r, p)` returns the value of the `p`-valuation of `r`. For example,

```
gap>  P_ValuationRat(120/21, 7);
-1
```

10 Coherent configurations

10.1 IsCoherentConfiguration

`IsCoherentConfiguration(R)` returns whether `R` is a relation matrix of a coherent configuration [5]. This function check whether the number of relations of `R` is equal to the dimension of the algebra generated by the relation matrices.

10.2 AssociationScheme2Configuration

Let `R` be a relation matrix of an association scheme of order `n`, and let `P` be a partition of `[1..n]`. `IsCoherentConfiguration(R, P)` returns the configuration determined by `R` and `P`.

We will show an example.

```
gap> R := CompleteGraphScheme(6);;
gap> CC := AssociationScheme2Configuration(R, [[1],[2,3,4,5,6]]);;
gap> Display(CC);
[ [  0,  1,  1,  1,  1,  1 ],
  [  2,  3,  4,  4,  4,  4 ],
  [  2,  4,  3,  4,  4,  4 ],
  [  2,  4,  4,  3,  4,  4 ],
  [  2,  4,  4,  4,  3,  4 ],
  [  2,  4,  4,  4,  4,  3 ] ]
gap> IsCoherentConfiguration(CC);
true
```

10.3 STS2CC

Let `M` be a incidence matrix of a Steiner triple system (2-$(v, k, 1)$ design). Then we can define a coherent configuration.

11 Other functions

11.1 AllMultiplicityFreeSubgroups

`AllMultiplicityFreeSubgroups(G)` returns the set of representatives of all conjugacy classes of proper subgroups `H` such that the Schurian scheme defined by `G` and `H` is commutative. For example, for the symmetric group of degree 5, we have the following result :

```
gap> AllMultiplicityFreeSubgroups(SymmetricGroup(5));
[ Group([ (1,2,3,4), (1,2) ]), Group([ (1,2,3), (1,2), (4,5) ]),
  Group([ (3,4), (1,4)(2,3), (1,3)(2,4) ]), Group([ (4,5), (1,2,3) ]),
  Alt( [ 1 .. 4 ] ), Alt( [ 1 .. 5 ] ), Group([ (1,2,3), (1,2)(4,5) ]),
  Group([ (2,3,4,5), (2,4)(3,5), (1,2,3,5,4) ]),
  Group([ (2,4)(3,5), (1,2,3,5,4) ]) ]
```

11.2 AllMultiplicityFreeSelfPairedSubgroups

`AllMultiplicityFreeSelfPairedSubgroups(G)` returns the set of representatives of all conjugacy classes of proper subgroups `H` such that the Schurian scheme defined by `G` and `H` is symmetric. For example, we have the following result :

```
gap> AllMultiplicityFreeSelfPairedSubgroups(SpecialLinearGroup(2,4));
[ Group([ [ [ Z(2^2)^2, 0*Z(2) ], [ 0*Z(2), Z(2^2) ] ],
      [ [ 0*Z(2), Z(2)^0 ], [ Z(2)^0, 0*Z(2) ] ] ]),
  Group([ [ [ Z(2^2), Z(2)^0 ], [ Z(2)^0, 0*Z(2) ] ],
      [ [ Z(2)^0, Z(2^2) ], [ 0*Z(2), Z(2)^0 ] ] ]),
  Group([ [ [ Z(2)^0, Z(2)^0 ], [ 0*Z(2), Z(2)^0 ] ],
      [ [ Z(2)^0, Z(2^2) ], [ 0*Z(2), Z(2)^0 ] ],
      [ [ Z(2^2)^2, 0*Z(2) ], [ 0*Z(2), Z(2^2) ] ] ]),
  Group([ [ [ Z(2^2), Z(2)^0 ], [ Z(2)^0, 0*Z(2) ] ] ]) ]
```

11.3 PValuationFiltration

Let `p` be a rational prime number. `PValuationFiltration(R, p)` returns the list of R_i, where

$$R_i = \{s \in \{0, 1, 2, \cdots, d\} \mid p^i \mid n_{\sigma_s} \quad \text{and} \quad p^{i+1} \nmid n_{\sigma_s}\}.$$

For example,

```
gap> Valencies(R);
[ 1, 1, 3, 3, 6, 6 ]
gap> PValuationFiltration(R, 2);
[ [ 0, 1, 2, 3 ], [ 4, 5 ] ]
gap> PValuationFiltration(R, 3);
[ [ 0, 1 ], [ 2, 3, 4, 5 ] ]
```

11.4 PermutePoints

`PermutePoints(R, P)` returns the relation matrix whose rows and columns are permuted by `P`. If you want to use a list `L` instead of a permutation, then use `PermList(L)`.

11.5 RenumberRelations

`RenumberRelations(R, P)` returns the relation matrix whose relation numbers are permuted by `P`. If you want to use a list `L` instead of a permutation, then use `PermList(L)`.

11.6 RenameRelations

`RenameRelations(R)` returns the relation matrix defined by `R`. For example,

```
gap> M := [["red", "blue"],["blue", "red"]];;
gap> RenameRelations(M);
[ [ 0, 1 ], [ 1, 0 ] ]
gap> M := [[5, 3, 1], [1, 5, 3], [3, 1, 5]];;
gap> RenameRelations(M);
[ [ 0, 2, 1 ], [ 1, 0, 2 ], [ 2, 1, 0 ] ]
```

11.7 CanonicalDualBasisAS

Let $A_0, A_1, \cdots, A_d$ be adjacency matrices of an association scheme `R`. `CanonicalDualBasisAS(R)` returns the dual basis $[[A_0, A_1, \cdots, A_d], [{n_0}^{-1}A_{0^*}, {n_1}^{-1}A_{1^*}, \cdots, {n_d}^{-1}A_{d^*}]]$.

11.8 CasimirOperatorAS

Let $A_0, A_1, \cdots, A_d$ be adjacency matrices of an association scheme `R`. Let `a` be an element of the adjacency algebra of `R` over the complex number field. `CasimirOperatorAS(a, R)` returns

$$\sum_{i=0}^{d} \frac{1}{n_i} A_i a A_{i^*}.$$

It is known that this element is in the center of the adjacency algebra.

11.9 CasimirElementAS

`CasimirElementAS(R)` returns `CasimirOperatorAS(I, R)`, where `I` is the identity martix.

11.10 VolumeAS

Same as `CasimirElementAS(R)`.

11.11 CommutatorOfRingElements

`CommutatorOfRingElements(a, b)` returns $ab - ba$ for ring elements a and b.

11.12 CommutatorOfAlgebra

Let K be a field , A an K-algebra, and let V and W be subspaces of A. `CommutatorOfAlgebra(K, V, W)` returns the subspace $[V, W] = \langle vw - wv \mid v \in V,\ w \in W \rangle_K$. `CommutatorOfAlgebra(K, V)` returns $[V, V]$.

11.13 Mat2TeX

`Mat2TeX(M)` displays the matrix `M` by TeX format. For example,

```
gap> R := GroupAssociationScheme(SymmetricGroup(3));
[ [ 0, 1, 1, 2, 2, 1 ], [ 1, 0, 2, 1, 1, 2 ], [ 1, 2, 0, 1, 1, 2 ],
  [ 2, 1, 1, 0, 2, 1 ], [ 2, 1, 1, 2, 0, 1 ], [ 1, 2, 2, 1, 1, 0 ] ]
gap> Mat2TeX(R);
0 & 1 & 1 & 2 & 2 & 1 \\
1 & 0 & 2 & 1 & 1 & 2 \\
1 & 2 & 0 & 1 & 1 & 2 \\
2 & 1 & 1 & 0 & 2 & 1 \\
2 & 1 & 1 & 2 & 0 & 1 \\
1 & 2 & 2 & 1 & 1 & 0 \\
```

12 Examples

```
[hanaki@kissme home]$ gap
    GAP, Version 4.5.5 of 16-Jul-2012 (free software, GPL)
   GAP     http://www.gap-system.org
    Architecture: i686-pc-linux-gnu-gcc-default32
 Libs used:  gmp
 Loading the library and packages ...
#I  You are using an old /home/hanaki/.gaprc file.
#I  See '?Ref: The former .gaprc file' for hints to upgrade.
 Components: trans 1.0, prim 2.1, small* 1.0, id* 1.0
 Packages:   AClib 1.2, Alnuth 3.0.0, AutPGrp 1.5, CRISP 1.3.5,
             Cryst 4.1.10, CrystCat 1.1.6, CTblLib 1.2.1, FactInt 1.5.3,
             FGA 1.1.1, GAPDoc 1.5.1, IRREDSOL 1.2.1, LAGUNA 3.6.1,
             Polenta 1.3.1, Polycyclic 2.10.1, RadiRoot 2.6,
             ResClasses 3.1.1, Sophus 1.23, TomLib 1.2.2
 Try '?help' for help. See also  '?copyright' and  '?authors'
gap> Read("association_scheme.gap");

Loading  GRAPE 4.6.1 (GRaph Algorithms using PErmutation groups)
by Leonard H. Soicher (http://www.maths.qmul.ac.uk/~leonard/).
```

```
Homepage: http://www.maths.qmul.ac.uk/~leonard/grape/

gap> R := HammingScheme(4,2);;
gap> Display(R);
[ [  0,  1,  1,  2,  1,  2,  2,  3,  1,  2,  2,  3,  2,  3,  3,  4 ],
  [  1,  0,  2,  1,  2,  1,  3,  2,  2,  1,  3,  2,  3,  2,  4,  3 ],
  [  1,  2,  0,  1,  2,  3,  1,  2,  2,  3,  1,  2,  3,  4,  2,  3 ],
  [  2,  1,  1,  0,  3,  2,  2,  1,  3,  2,  2,  1,  4,  3,  3,  2 ],
  [  1,  2,  2,  3,  0,  1,  1,  2,  2,  3,  3,  4,  1,  2,  2,  3 ],
  [  2,  1,  3,  2,  1,  0,  2,  1,  3,  2,  4,  3,  2,  1,  3,  2 ],
  [  2,  3,  1,  2,  1,  2,  0,  1,  3,  4,  2,  3,  2,  3,  1,  2 ],
  [  3,  2,  2,  1,  2,  1,  1,  0,  4,  3,  3,  2,  3,  2,  2,  1 ],
  [  1,  2,  2,  3,  2,  3,  3,  4,  0,  1,  1,  2,  1,  2,  2,  3 ],
  [  2,  1,  3,  2,  3,  2,  4,  3,  1,  0,  2,  1,  2,  1,  3,  2 ],
  [  2,  3,  1,  2,  3,  4,  2,  3,  1,  2,  0,  1,  2,  3,  1,  2 ],
  [  3,  2,  2,  1,  4,  3,  3,  2,  2,  1,  1,  0,  3,  2,  2,  1 ],
  [  2,  3,  3,  4,  1,  2,  2,  3,  1,  2,  2,  3,  0,  1,  1,  2 ],
  [  3,  2,  4,  3,  2,  1,  3,  2,  2,  1,  3,  2,  1,  0,  2,  1 ],
  [  3,  4,  2,  3,  2,  3,  1,  2,  2,  3,  1,  2,  1,  2,  0,  1 ],
  [  4,  3,  3,  2,  3,  2,  2,  1,  3,  2,  2,  1,  2,  1,  1,  0 ] ]
gap> trad := ThinRadical(R);
[ 0, 4 ]
gap> tres := ThinResidue(R);
[ 0, 2, 4 ]
gap> R2 := RelationMatrixSortedByClosedSubset(R, trad);;
gap> Display(R2);
[ [  0,  4,  1,  3,  1,  3,  2,  2,  1,  3,  2,  2,  2,  2,  3,  1 ],
  [  4,  0,  3,  1,  3,  1,  2,  2,  3,  1,  2,  2,  2,  2,  1,  3 ],
  [  1,  3,  0,  4,  2,  2,  1,  3,  2,  2,  1,  3,  3,  1,  2,  2 ],
  [  3,  1,  4,  0,  2,  2,  3,  1,  2,  2,  3,  1,  1,  3,  2,  2 ],
  [  1,  3,  2,  2,  0,  4,  1,  3,  2,  2,  3,  1,  1,  3,  2,  2 ],
  [  3,  1,  2,  2,  4,  0,  3,  1,  2,  2,  1,  3,  3,  1,  2,  2 ],
  [  2,  2,  1,  3,  1,  3,  0,  4,  3,  1,  2,  2,  2,  2,  1,  3 ],
  [  2,  2,  3,  1,  3,  1,  4,  0,  1,  3,  2,  2,  2,  2,  3,  1 ],
  [  1,  3,  2,  2,  2,  2,  3,  1,  0,  4,  1,  3,  1,  3,  2,  2 ],
  [  3,  1,  2,  2,  2,  2,  1,  3,  4,  0,  3,  1,  3,  1,  2,  2 ],
  [  2,  2,  1,  3,  3,  1,  2,  2,  1,  3,  0,  4,  2,  2,  1,  3 ],
  [  2,  2,  3,  1,  1,  3,  2,  2,  3,  1,  4,  0,  2,  2,  3,  1 ],
  [  2,  2,  3,  1,  1,  3,  2,  2,  1,  3,  2,  2,  0,  4,  1,  3 ],
  [  2,  2,  1,  3,  3,  1,  2,  2,  3,  1,  2,  2,  4,  0,  3,  1 ],
  [  3,  1,  2,  2,  2,  2,  1,  3,  2,  2,  1,  3,  1,  3,  0,  4 ],
  [  1,  3,  2,  2,  2,  2,  3,  1,  2,  2,  3,  1,  3,  1,  4,  0 ] ]
gap> R3 := RelationMatrixSortedByClosedSubset(R, tres);;
gap> Display(R3);
[ [  0,  2,  2,  2,  2,  2,  2,  4,  1,  1,  1,  3,  1,  3,  3,  3 ],
  [  2,  0,  2,  2,  2,  2,  4,  2,  1,  1,  3,  1,  3,  1,  3,  3 ],
```

```
  [  2,  2,  0,  2,  2,  4,  2,  2,  1,  3,  1,  1,  3,  3,  1,  3 ],
  [  2,  2,  2,  0,  4,  2,  2,  2,  3,  1,  1,  1,  3,  3,  3,  1 ],
  [  2,  2,  2,  4,  0,  2,  2,  2,  1,  3,  3,  3,  1,  1,  1,  3 ],
  [  2,  2,  4,  2,  2,  0,  2,  2,  3,  1,  3,  3,  1,  1,  3,  1 ],
  [  2,  4,  2,  2,  2,  2,  0,  2,  3,  3,  1,  3,  1,  3,  1,  1 ],
  [  4,  2,  2,  2,  2,  2,  2,  0,  3,  3,  3,  1,  3,  1,  1,  1 ],
  [  1,  1,  1,  3,  1,  3,  3,  3,  0,  2,  2,  2,  2,  2,  2,  4 ],
  [  1,  1,  3,  1,  3,  1,  3,  3,  2,  0,  2,  2,  2,  2,  4,  2 ],
  [  1,  3,  1,  1,  3,  3,  1,  3,  2,  2,  0,  2,  2,  4,  2,  2 ],
  [  3,  1,  1,  1,  3,  3,  3,  1,  2,  2,  2,  0,  4,  2,  2,  2 ],
  [  1,  3,  3,  3,  1,  1,  1,  3,  2,  2,  2,  4,  0,  2,  2,  2 ],
  [  3,  1,  3,  3,  1,  1,  3,  1,  2,  2,  4,  2,  2,  0,  2,  2 ],
  [  3,  3,  1,  3,  1,  3,  1,  1,  2,  4,  2,  2,  2,  2,  0,  2 ],
  [  3,  3,  3,  1,  3,  1,  1,  1,  4,  2,  2,  2,  2,  2,  2,  0 ] ]
gap> R4 := Subscheme(R, trad);;
gap> Display(R4);
[ [  0,  1 ],
  [  1,  0 ] ]
gap> R5 := Subscheme(R, tres, 1);;
gap> Display(R5);
[ [  0,  1,  1,  1,  1,  1,  1,  2 ],
  [  1,  0,  1,  1,  1,  1,  2,  1 ],
  [  1,  1,  0,  1,  1,  2,  1,  1 ],
  [  1,  1,  1,  0,  2,  1,  1,  1 ],
  [  1,  1,  1,  2,  0,  1,  1,  1 ],
  [  1,  1,  2,  1,  1,  0,  1,  1 ],
  [  1,  2,  1,  1,  1,  1,  0,  1 ],
  [  2,  1,  1,  1,  1,  1,  1,  0 ] ]
gap> R6 := FactorScheme(R, trad);;
gap> Display(R6);
[ [  0,  1,  1,  2,  1,  2,  2,  1 ],
  [  1,  0,  2,  1,  2,  1,  1,  2 ],
  [  1,  2,  0,  1,  2,  1,  1,  2 ],
  [  2,  1,  1,  0,  1,  2,  2,  1 ],
  [  1,  2,  2,  1,  0,  1,  1,  2 ],
  [  2,  1,  1,  2,  1,  0,  2,  1 ],
  [  2,  1,  1,  2,  1,  2,  0,  1 ],
  [  1,  2,  2,  1,  2,  1,  1,  0 ] ]
gap> R7 := FactorScheme(R, tres);;
gap> Display(R7);
[ [  0,  1 ],
  [  1,  0 ] ]
gap> IsThin(R6);
false
gap> IsThin(R7);
true
```

```
gap> IsQuasiThin(R6);
false
gap> IsQuasiThin(R7);
false
gap> quit;
[hanaki@kissme home]$
```

References

[1] Z. Arad, E. Fisman, and M. Muzychuk. Generalized table algebras. *Israel J. Math.*, 114:29–60, 1999.

[2] E. Bannai and T. Ito. *Algebraic combinatorics. I.* The Benjamin/Cummings Publishing Co. Inc., Menlo Park, CA, 1984.

[3] A. Hanaki. Semisimplicity of adjacency algebras of association schemes. *J. Algebra*, 225(1):124–129, 2000.

[4] A. Hanaki. Characters of association schemes and normal closed subsets. *Graphs Combin.*, 19(3):363–369, 2003.

[5] D. G. Higman. Coherent configurations. I. Ordinary representation theory. *Geometriae Dedicata*, 4(1):1–32, 1975.

[6] M. Klin, M. Muzychuk, C. Pech, A. Woldar, and P.-H. Zieschang. Association schemes on 28 points as mergings of a half-homogeneous coherent configuration. *European J. Combin.*, 28(7):1994–2025, 2007.

[7] B. Külshammer. Group-theoretical descriptions of ring-theoretical invariants of group algebras. In *Representation theory of finite groups and finite-dimensional algebras (Bielefeld, 1991)*, volume 95 of *Progr. Math.*, pages 425–442. Birkhäuser, Basel, 1991.

[8] A. Munemasa. An application of Terwilliger's algebra. unpublished, 1993.

[9] P. Terwilliger. The subconstituent algebra of an association scheme. I. *J. Algebraic Combin.*, 1(4):363–388, 1992.

[10] P. Terwilliger. The subconstituent algebra of an association scheme. II. *J. Algebraic Combin.*, 2(1):73–103, 1993.

[11] P. Terwilliger. The subconstituent algebra of an association scheme. III. *J. Algebraic Combin.*, 2(2):177–210, 1993.

[12] P.-H. Zieschang. *An algebraic approach to association schemes*, volume 1628 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1996.