# Paramodifications of abstract unitals

## GAP implementation and examples worksheet

0.1

8 March 2020

**Gábor P. Nagy, Dávid Mezőfi**

**Address:** Bolyai Institute of the University of Szeged (Hungary)
Department of Algebra of the Budapest University of Technology and Economics (Hungary)

## Abstract

GAP implementation of paramodifications of abstract unitals. With small modifications, it must work for other 2-designs as well.





## Acknowledgements

Support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2018-1.2.1-NKP funding scheme, within the SETIT Project 2018-1.2.1-NKP-2018-00004. Partially supported by OTKA grants 119687 and 115288.

# Contents

# Chapter 1

# Commands for paramodifications

## 1.1 Prerequisites

The GAP packages UnitalSZ and GRAPE are necessary.

Example

```
gap> LoadPackage( "UnitalSZ", false );
gap> LoadPackage( "grape", false );
```

You can use the info class `InfoParamod` to get extra information.

Example

```
gap> DeclareInfoClass( "InfoParamod" );
```

### 1.1.1 AllRegularBlockColorings

▷ `AllRegularBlockColorings(bls, k, gr)` (function)

**Returns:** the list of block colorings of the set `bls` of blocks, with `k` colors and color classes of size `|bls|/k`. The colorings are returned as GAP transformation objects from `[1..Size(bls)]` to `[1..k]`.

Let $P$ be a set, $B$ a set of subsets (called *blocks*), and $k$ a positive integer. The map $\chi : B \to \{1,\ldots,k\}$ is a block coloring if $\chi(b) = \chi(b')$ implies $b \cap b' = \emptyset$ for any $b, b' \in B$. The block coloring in *regular*, if each color class has size $|B|/k$. The last argument `gr` must be a block preserving permutation of the set $P$ of points.

## 1.2 Paramodifications

### 1.2.1 ParamodificationOfUnital

▷ `ParamodificationOfUnital(u, b, chi)` (function)

**Returns:** the paramodification of the unital `u` with respect to the block `b` and regular block coloring `chi` with `|b|` colors.

The coloring `chi` must be given as a GAP transformation object from `[1..q^2*(q^2-q+1)]` to `[1..q+1]`, where `q` is the order of `u`. This function has a slightly faster non-checking version `ParamodificationOfUnitalNC`.

### 1.2.2 ParamodificationsOfUnitalWithBlock

▷ `ParamodificationsOfUnitalWithBlock(u, b)` (function)

**Returns:** all paramodifications of the unital $u$ with respect to the block $b$. The results are reduced up to isomorphism of abstract unitals.

### 1.2.3 AllParamodificationsOfUnital

▷ `AllParamodificationsOfUnital(u)` (function)

**Returns:** all paramodifications of the unital $u$. The results are reduced up to isomorphism of abstract unitals.

## 1.3 Implementations

### 1.3.1 Regular block colorings

Example

```
gap> InstallGlobalFunction( AllRegularBlockColorings, function( bls, nr_colors, gr )
>         local Gamma, complete_subgraphs, graph_of_cliques, colorings, ret, new_blocks, c, c_vec
gap>            # construct the line graph and its cliques
>         Gamma := Graph( gr, bls, OnSets,
>                  function( x, y ) return x <> y and Intersection( x, y ) = [];
gap>            end );
gap>            complete_subgraphs := CompleteSubgraphs( Gamma, Size(bls)/nr_colors, 1);;
gap>            complete_subgraphs := Union( List( complete_subgraphs,
>                                      x -> Orbit( AutomorphismGroup( Gamma ), x, OnSets ) ) );
gap>            Info( InfoParamod, 3, "cliques of the line graph computed..." );
gap>            # construct the graph of cliques and its cliques
>         graph_of_cliques := Graph( Gamma.group, complete_subgraphs, OnSets,
>                                    function( x, y ) return x <> y and Intersection( x, y ) = [];
gap>                                       end );
gap>            colorings := CompleteSubgraphs( graph_of_cliques, nr_colors, 1 );
gap>            Info( InfoParamod, 3, Size( colorings ), " block colorings computed..." );
gap>            # construct colorings
>         ret := [];
gap>            for c in colorings do
>                 c_vec:=0*[1..Size(bls)];
gap>                    for i in [1..nr_colors] do
>                         for j in VertexNames( graph_of_cliques )[c[i]] do
>                                 c_vec[ Position( bls, VertexNames( Gamma )[j] ) ] := i;
gap>                            od;
gap>                    od;
gap>                    Add(ret, Transformation(c_vec));
gap>            od;
gap>            return ret;
gap> end );
```

### 1.3.2 Paramodifications

Example

```
gap> InstallGlobalFunction( ParamodificationOfUnitalNC, function( u, b, chi )
>         local Cb, n_Cb, C_star_b, intact_blks, B_star;
```

```
gap>          Cb := Filtered( BlocksOfUnital( u ),
>                  x -> Size( Intersection( x, b ) ) = 1 );
gap>          n_Cb := Length( Cb );
gap>          C_star_b := List( [1..n_Cb],
>                  i -> Union( Difference( Cb[i], b ), [ b[i^chi] ] ) );
gap>          intact_blks := Difference( BlocksOfUnital( u ), Cb );
gap>          B_star := Union( intact_blks, C_star_b );
gap>          return AbstractUnitalByDesignBlocks( B_star );
gap> end );
```

Example

```
gap> InstallGlobalFunction( ParamodificationOfUnital, function( u, b, chi )
>         local Cb;
gap>          if not b in BlocksOfUnital( u ) then
>                  Error( "argument 2 must be a block of argument 1");
gap>          fi;
gap>          Cb := Filtered( BlocksOfUnital( u ),
>                         x -> Size( Intersection( x, b ) ) = 1 );
gap>          Cb := List( Cb, x -> Difference( x, b) );
gap>          if not ForAll( Combinations( [1..Size(Cb)], 2 ), p ->
>                              Intersection(Cb{p})=[] or
>                              (p[1]^chi<>p[2]^chi)
>                  ) then Error( "argument 3 is not a proper block coloring" );
gap>          fi;
gap>          return ParamodificationOfUnitalNC( u, b, chi );
gap> end );
```

### 1.3.3 Paramodifications with respect to a block

Example

```
gap> InstallGlobalFunction( ParamodificationsOfUnitalWithBlock,
> function( u, b )
>         local q, Cb, b_stab, new_unitals, all, allchibmod, i, isom_class, colorings;
gap>          if not b in BlocksOfUnital( u ) then
>                  Error( "argument 2 must be a block of argument 1");
gap>          fi;
gap>          q := Order( u );
gap>          Cb := Filtered( BlocksOfUnital( u ),
>                  x -> Size( Intersection( x, b ) ) = 1 );
gap>          # Important: keep the order from BlocksOfUnital
>         Cb := List( Cb, x -> Difference( x, b ) );
gap>          # compute all colorings
>         b_stab := Stabilizer( AutomorphismGroup( u ), b, OnSets );
gap>          colorings := AllRegularBlockColorings( Cb, q + 1, b_stab );
gap>          Info( InfoParamod, 1, Size( colorings ), " coloring(s) for the given unital-block pa
gap>          new_unitals := List( colorings, c -> ParamodificationOfUnitalNC( u, b, c ) );
gap>          # reduction up to isomorphism
>         all := [1..Length( new_unitals )];
gap>          allchibmod := [];
gap>          while all <> [] do
>                  i := Remove( all );
gap>                     isom_class := Filtered( all, x -> Isomorphism( new_unitals[i],
>                          new_unitals[x] ) <> fail ) ;
```

```
gap>                 all := Difference( all, isom_class );
gap>                 Add( allchibmod, new_unitals[i] );
gap>         od;
gap>         return allchibmod;
gap> end );
```

### 1.3.4 All paramodifications of a unital

Example

```
gap> InstallGlobalFunction( AllParamodificationsOfUnital, function( u )
>         local blocks, rep_blocks, allchibmods, uus, b;
gap>         blocks := BlocksOfUnital( u );
gap>         rep_blocks := List( Orbits( AutomorphismGroup( u ), blocks, OnSets ),
>                     orb -> Representative( orb ) );
gap>         Info( InfoParamod, 2, Size( rep_blocks ), " block representatives for the unital com
gap>         allchibmods := [];
gap>         for b in rep_blocks do
>              uus := ParamodificationsOfUnitalWithBlock( u, b );
gap>                 uus := Filtered( uus, x -> Isomorphism( x, u ) = fail and
>                                 ForAll( allchibmods, y -> Isomorphism( y, x ) = fail ) );
gap>                 Append( allchibmods, uus );
gap>         od;
gap>         return allchibmods;
gap> end );
```

# Chapter 2

# Examples

## 2.1 Computing paramodifications

Example

```
gap> LoadPackage( "UnitalSZ", false );
gap> LoadPackage( "grape", false );
gap> u:=KNPAbstractUnital(1577);
KNPAbstractUnital(1577)
gap> AutomorphismGroup(u);
<permutation group with 6 generators>
gap> b:=BlocksOfUnital(u)[1];
[ 1, 2, 5, 6, 14 ]
gap> bls0:=Filtered(BlocksOfUnital(u),x->Size(Intersection(x,b))=1);;
gap> bls:=List(bls0,x->Difference(x,b));;
gap> cols:=AllRegularBlockColorings(bls,5,Group(())); time;
[ Transformation( [ 1, 1, 1, 4, 2, 5, 3, 5, 3, 4, 2, 2, 5, 3, 4, 4, 3, 2, 5, 5, 3, 4, 4, 5, 3, 2,
      4, 3, 2, 2, 3, 2, 5, 3, 4, 1, 1, 1, 4, 2, 4, 4, 5, 2, 3, 2, 5, 3, 1, 1, 1, 3, 3, 2, 5, 3, 5,
  Transformation( [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 3, 3, 4, 2, 4, 2, 3, 3, 2, 4,
      5, 3, 5, 3, 2, 2, 3, 5, 3, 4, 4, 4, 4, 3, 5, 3, 5, 4, 4, 5, 3, 5, 2, 2, 2, 5, 2, 4, 2, 4, 5,
  Transformation( [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 4, 4, 4, 5, 5, 5, 3, 3, 3, 5,
      5, 2, 4, 2, 5, 5, 4, 4, 4, 4, 2, 5, 5, 3, 3, 2, 3, 2, 2, 5, 2, 5, 5, 3, 2, 3, 2, 4, 3, 4, 4,
  Transformation( [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
      5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2,
  Transformation( [ 1, 2, 5, 5, 5, 5, 5, 2, 1, 2, 2, 1, 1, 2, 1, 4, 4, 4, 4, 2, 2, 2, 1, 1, 1, 2,
      5, 1, 4, 1, 5, 5, 4, 4, 4, 4, 1, 5, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 5, 2, 4, 5, 4, 4,
  Transformation( [ 1, 3, 4, 4, 4, 4, 4, 3, 1, 3, 3, 1, 1, 3, 1, 4, 4, 4, 4, 5, 5, 5, 1, 1, 1, 5,
      5, 1, 3, 1, 5, 5, 3, 3, 3, 3, 1, 5, 5, 3, 3, 4, 3, 4, 4, 5, 4, 5, 5, 3, 4, 2, 2, 2, 2, 2, 2,
  Transformation( [ 1, 2, 4, 4, 4, 4, 4, 2, 1, 2, 2, 1, 1, 2, 1, 4, 4, 4, 4, 2, 2, 2, 3, 3, 3, 2,
      5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 1, 3, 3, 4, 3, 4, 4, 1, 4, 1, 1, 3, 4, 3, 2, 1, 3, 1, 1,
  Transformation( [ 1, 3, 5, 5, 5, 5, 5, 3, 1, 3, 3, 1, 1, 3, 1, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
      5, 2, 3, 2, 5, 5, 3, 3, 3, 3, 2, 5, 1, 3, 3, 2, 3, 2, 2, 1, 2, 1, 1, 3, 2, 5, 2, 1, 5, 1, 1,
  Transformation( [ 1, 5, 2, 1, 5, 1, 5, 2, 2, 2, 1, 2, 5, 1, 5, 4, 1, 1, 4, 1, 4, 1, 3, 3, 3, 4,
      5, 5, 2, 5, 1, 1, 1, 2, 1, 4, 4, 4, 4, 3, 3, 5, 3, 4, 4, 5, 5, 5, 1, 3, 2, 3, 2, 4, 3, 4, 2,
  Transformation( [ 1, 4, 3, 1, 4, 1, 4, 3, 3, 3, 1, 3, 4, 1, 4, 4, 1, 1, 4, 1, 3, 1, 3, 3, 4, 3,
      5, 2, 5, 2, 1, 1, 1, 5, 1, 4, 2, 3, 3, 3, 5, 2, 5, 2, 2, 5, 2, 5, 1, 4, 2, 5, 2, 4, 4, 4, 5,
  Transformation( [ 1, 4, 2, 1, 4, 1, 4, 2, 2, 2, 1, 2, 4, 1, 4, 4, 3, 3, 4, 5, 5, 5, 3, 3, 4, 5,
      5, 3, 2, 3, 5, 5, 3, 2, 3, 4, 1, 5, 5, 3, 1, 3, 1, 1, 1, 5, 3, 5, 5, 4, 2, 1, 2, 4, 4, 4, 2,
  Transformation( [ 1, 5, 3, 1, 5, 1, 5, 3, 3, 3, 1, 3, 5, 1, 5, 4, 4, 4, 4, 2, 3, 2, 3, 3, 2, 3,
      5, 5, 4, 5, 2, 2, 4, 4, 4, 4, 1, 3, 3, 3, 1, 5, 1, 1, 1, 5, 5, 5, 2, 2, 2, 1, 2, 4, 2, 4, 4,
```

```
342
gap> List(cols,c->ParamodificationOfUnitalNC(u,b,c)); time;
[ AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign
  AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign
32
gap> List(cols,c->ParamodificationOfUnital(u,b,c)); time;
[ AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign
  AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign
88
gap> ParamodificationsOfUnitalWithBlock(u,b); time;
[ AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign
219
gap> AllParamodificationsOfUnital(u); time;
[ AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign
  AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4>, AU_UnitalDesign<4> ]
961
```

## 2.2 Para-rigidity of cyclic unitals

We say that a unital is *para-rigid*, if all block colorings of $C(b)$ are equivalent with the trivial one. The following example shows that the cyclic unitals of order 4 and 6 by Bagchi and Bagchi are para-rigid.

Example

```
gap> SetInfoLevel(InfoParamod,2);
gap> u:=BagchiBagchiCyclicUnital(4);
BagchiBagchiCyclicUnital(4)
gap> AllParamodificationsOfUnital(u);
#I  2 block representatives for the unital computed...
#I  1 coloring(s) for the given unital-block pair computed...
#I  1 coloring(s) for the given unital-block pair computed...
[ ]
gap> u:=BagchiBagchiCyclicUnital(6);
BagchiBagchiCyclicUnital(6)
gap> AllParamodificationsOfUnital(u);
#I  2 block representatives for the unital computed...
#I  1 coloring(s) for the given unital-block pair computed...
#I  1 coloring(s) for the given unital-block pair computed...
[ ]
```