1 Generalities about group rings

There are several packages in GAP designed to work with group rings. For example:

- LAGUNA : **L**ie **A**l**G**ebras and **UN**its of group **A**lgebras
- UnitLib : **Lib**rary of normalized **unit** groups of modular group algebras
- Wedderga : **Wedder**burn Decomposition of **G**roup **A**lgebras
- HeLP : **He**rtweck-**L**uthar-**P**assi method.

For each package there is a documentation providing lot of additional information and instructions (see e.g. the webpages of the packages). To read, for instance, the package LAGUNA into a running GAP session one can just type

```
gap> LoadPackage("laguna");
true
```

In case the package was loaded successfully, true is returned. [1]
GAP provides the functionality to define group rings and group algebras:

```
gap> G := DihedralGroup(16);;
gap> ZG := GroupRing(Integers, G);
<free left module over Integers, and ring-with-one, with 4 generators>
gap> QG := GroupRing(Rationals, G);
<algebra-with-one over Rationals, with 4 generators>
gap> FG := GroupRing(GF(2), G);
<algebra-with-one over GF(2), with 4 generators>
```

Using LAGUNA one might check whether the group ring is actually a group algebra, i.e. the coefficient domain is a field:

```
gap> IsGroupAlgebra(ZG); IsGroupAlgebra(QG); IsGroupAlgebra(FG);
false
true
true
```

The commands UnderlyingGroup and UnderlyingRing can be used to retrieve the group and coefficient ring used to define the group ring.
To deal with the group elements as elements of the group ring, one might embed those element in the group ring:

```
gap> i := Embedding(G, ZG);;
```

For example, the center of $G = D_{16}$ is cyclic of order 2 generated by a rotation z of radian π and the rotation subgroup is generated by any element a of order 8:

[1] Note that several packages, such as LAGUNA are loaded by default in recent GAP versions. You can also customize which packages are loaded by default by editing the gap.ini or gaprc files, see Section 3.2 of the manual.

```
gap> z := MinimalGeneratingSet(Center(G))[1];
f4
gap> z in ZG;
false
gap> z^i in ZG;
true
gap> a := First(Elements(G), g -> Order(g) = 8);   # generator of rotations
f2
```

We can now use the arithmetic for elements of $\mathbb{Z}G$ just as expected:

```
gap> v := z^i + a^i;
(1)*f2+(1)*f4
```

LAGUNA allows to check whether an element of a group ring is symmetric, i.e. invariant under the canonical involution defined by $g \mapsto g^{-1},\ g \in G$. And we can of course force elements to become symmetric:

```
gap> IsSymmetric(v);
false
gap> IsSymmetric(v*Involution(v));
true
gap> IsSymmetric(v+Involution(v));
true
```

Is $v = a + z$ a unit in $\mathbb{Z}G$? Well, recall that the augmentation $\mathrm{aug}\colon \mathbb{Z}G \to \mathbb{Z}, g \mapsto 1$ (i.e. the $\mathbb{Z}$-linear extension of the trivial representation) is a ringhomomorphism. Clearly, $\mathrm{aug}(v) = 2$. This can be verified by GAP:

```
gap> Augmentation(v);
2
gap> v^-1;
fail
```

Assume that $b \in G$ is a non-central involution in G. Is $x = a - b + z$ a unit in $\mathbb{Z}G$? Now $\mathrm{aug}(x) = 1$, but:

```
gap> b := First(Elements(G), g -> Order(g) = 2 and not g in Center(G));
f1
gap> x := v - b^i;
(-1)*f1+(1)*f2+(1)*f4
gap> Augmentation(x);
1
gap> x^-1;
(1/3)*<identity> of ...+(-1/3)*f1+(1/3)*f2+(1/3)*f3+(1/3)*f4+(-1/3)*f1*f2+(-1/
3)*f1*f3+(2/3)*f1*f4+(1/3)*f2*f3+(-2/3)*f2*f4+(-2/3)*f3*f4+(2/3)*f1*f2*f3+(2/
3)*f1*f2*f4+(-1/3)*f1*f3*f4+(1/3)*f2*f3*f4+(-1/3)*f1*f2*f3*f4
```

Note that the inverse of x was calculated in $\mathbb{Q}G$, yet $x^{-1} \notin \mathbb{Z}G$.

2 Modular group algebras

The package `LAGUNA` provides much more functionality for finite p-modular group algebras[2] like our `FG` $= \mathbb{F}_2 D_{16}$ defined above. The normalized unit group is the kernel of the augmentation map restricted to the group of units of a group ring, i.e. $V(RG) = \{u \in RG \mid u \text{ invertible in } RG, \mathrm{aug}(u) = 1\}$.

```
gap> V := NormalizedUnitGroup(FG);
<group of size 32768 with 15 generators>
```

When one wants to do calculations with groups it is frequently useful to have a power commutator (PC) presentation, this can be achieved through the command `VC := PcNormalizedUnitGroup(FG);`. For example, compare the runtime of the commands `NilpotencyClassOfGroup(V);` and `NilpotencyClassOfGroup(VC);`. Note however that the generators of the tow unit groups do not agree in general, one may use the command `NaturalBijectionToPcNormalizedUnitGroup` to convert from one to the other.
Even for p-groups G of moderate size, the computation of the *structure* of the group of units of $\mathbb{F}_p G$ might be pretty time and memory consuming (to find it as a set is easy). The package `UnitLib` provides pre-computed unit groups for all p-groups up to order 243 (having the same attributes as if they were computed in `LAGUNA`, note that for groups of order 128 these are only available on UNIX-type systems):

```
gap> LoadPackage("UnitLib");
gap> V := PcNormalizedUnitGroupSmallGroup(128,161);
```

where V will be the normalized unit group of $\mathbb{F}G$. Having this description at ones disposal one might calculate certain invariants:

```
gap> C := Center(V);;
gap> StructureDescription(C);
"C32 x C16 x C8 x C8 x C4 x C4 x C4 x C4 x C2 x C2 x C2 x C2 x C2 x C2 x C2 x C2 x C2 x C2"
gap> AbelianInvariants(C);
[ 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 8, 8, 16, 32 ]
gap> FG := UnderlyingGroupRing(V);
<algebra-with-one over GF(2), with 7 generators>
gap> UnderlyingRing(FG);
GF(2)
gap> UnderlyingGroup(FG);
<pc group of size 128 with 7 generators>
gap> StructureDescription(last);
"D128"
```

Note that in the case of abelian groups `AbelianInvariants` might be way more lucid than `StructureDescription`. Try to compute the unit group of $\mathbb{F}_2 G$ with G the group with SmallGroupID (64, 52).

[2] recall that a group algebra FG is called p-modular if G is a p-group and F is a field of characteristic p.

3 Units in integral group rings

The package `LAGUNA` also provides functionality that is particular useful when computing with units in group rings given by concrete constructions. For elements $g, h \in G$ the normalized units

$$b(h, \tilde{g}) = 1 + (1 - g)h\tilde{g},$$
$$b(\tilde{g}, h) = 1 + \tilde{g}h(1 - g)$$

are units in RG, called *bicyclic units* of type 1 and type 2, respectively. (Here $\tilde{g} = (1+g+g^2+...+g^{|g|-1}$.) They can be called in `GAP` by `BicyclicUnitOfType1(gî, hî)` and `BicyclicUnitOfType2(gî, hî)`, respectively (caution: note the change of order of the element in the arguments of bicyclic units of type 1). Note that $b(h, \tilde{g})$ and $b(\tilde{g}, h)$ are trivial, i.e. are elements of G, if and only if $h \in \mathrm{N}_G(\langle g \rangle)$. In case they are non trivial, these units are of infinite order.
Recall that in our example `G` is a dihedral group of order 16 generated by the elements `a` and `b` of order 8 and 2, respectively (where the latter inverts the former). We defined `i` to be the embedding of G in $\mathbb{Z}G$. One has:

```
gap> BicyclicUnitOfType1(a^i, b^i);
(1)*<identity> of ..
gap> u1 := BicyclicUnitOfType1(b^i, a^i);
(1)*<identity> of ...+(-1)*f2+(1)*f1*f2+(1)*f2*f3*f4+(-1)*f1*f2*f3*f4
```

This matches what theory predicts: $\mathtt{b} \in \mathrm{N}_{\mathtt{G}}(\langle \mathtt{a} \rangle)$, since $\langle \mathtt{a} \rangle \trianglelefteq \mathtt{G}$ but, $\mathtt{a} \notin \mathrm{N}_{\mathtt{G}}(\langle \mathtt{b} \rangle)$. Here the arguments were actually elements from $\mathbb{Z}G$. There is also a three-argument version that takes the group ring in question together with elements from the underlying group:

```
gap> v1 := BicyclicUnitOfType1(ZG, b, a);
(1)*<identity> of ...+(-1)*f2+(1)*f1*f2+(1)*f2*f3*f4+(-1)*f1*f2*f3*f4
gap> u1 = v1;
true
```

Of course, there also exists the command `BicyclicUnitOfType2` with the obvious adaptions.
Bicyclic units can also be used to produce torsion units: If $u_1 = b(h, \tilde{g})$ is a bicyclic unit of type 1 based on h and g, then

$$u = u_1 g = g + (1 - g)h\tilde{g}$$

is a torsion unit of order $|g|$. Furthermore, u is non-trivial, i.e. $u \notin G$, if and only if $b(h, \tilde{g}) \notin G$, that is, $h \notin \mathrm{N}_G(\langle g \rangle)$.

```
gap> u := u1*b^i;
(1)*f1+(-1)*f2+(1)*f1*f2+(1)*f2*f3*f4+(-1)*f1*f2*f3*f4
gap> Order(u);
2
```

The units of this form are called *Bovdi units* (named after Victor Bovdi). In case they are not contained in G, "how far away" are they from G? We will meet this question later in the realm of the Zassenhaus Conjectures (or Zassenhaus Problems).
There is another extremely useful generic construction of units in integral group rings, the so-called *Bass cyclic units*. In `GAP` the Bass units is based on two parameters: $g \in G$ a group element of order and an integer $1 < k < n$ such that k and n are coprime. Then $k^{\varphi(n)} \equiv 1 \pmod{n}$. Then in

GAP the Bass cyclic unit is

$$u_{k,\varphi(n)}(g) = \left(\sum_{\ell=0}^{k-1} g^{\ell}\right)^{\varphi(n)} + \frac{1-k^{\varphi(n)}}{n}\hat{g}$$

```
gap> v := BassCyclicUnit(a^i, 3);
(-8)*<identity> of ...+(-6)*f2+(9)*f4+(6)*f2*f3+(6)*f2*f4+(-6)*f2*f3*f4
```

In the generation of large subgroups of the unit group, the Bass cyclic units are important to cover the center of the units of $\mathbb{Z}G$. The unit v defined above actually happens to be central in $\mathrm{U}(\mathbb{Z}G)$ (how do you verify this?), however in general Bass cyclic units are far from being central, cf. the lectures of Eric Jespers.

4 Wedderburn decompositions of group algebras

The GAP package `wedderga` provides functionality to calculate the Wedderbrun decomposition of group algebras FG, where G is a finite group and F is either an abelian number field or a finite field of characteristic not dividing $|G|$. The two main functions are `WedderburnDecomposition` and `WedderburnDecompositionInfo`. Both calculate the Wedderburn decomposition of the group algebra FG, the former returns a list of simple algebras as GAP objects such that their direct product is isomorphic to the group algebra FG, the latter returns a similar list, but here the entries of the list are GAP algebras, but are descriptions of the algebras that are usually more appealing for the human reader (but cannot be further processed in GAP as simple algebras).

```
gap> LoadPackage("wedderga");
gap> WedderburnDecomposition( GroupRing(Rationals, CyclicGroup(3)) );
[ Rationals, CF(3) ]
gap> WedderburnDecompositionInfo( GroupRing(Rationals, CyclicGroup(3)) );
[ [ 1, Rationals ], [ 1, CF(3) ] ]
```

In the output of `WedderburnDecompositionInfo`, each entry is a list `[n, F]`, representing a $n \times n$-matrix ring over the filed F. Note that `CF(n)` represents the nth cyclotomic field, i.e. the field extension of $\mathbb{Q}$ obtained by adjoining a primitive nth root of unity. So that $\mathbb{Q}C_3 \simeq \mathbb{Q} \times \mathbb{Q}(\zeta_3)$, as expected.
In case of finite fields, the field extension is usually indicated by the cardinality of the field.

```
gap> WedderburnDecompositionInfo( GroupRing(GF(5), CyclicGroup(3)) );
[ [ 1, 5 ], [ 1, 25 ] ]
gap> WedderburnDecompositionInfo( GroupRing(GF(7), CyclicGroup(3)) );
[ [ 1, 7 ], [ 1, 7 ], [ 1, 7 ] ]
```

Hence $\mathbb{F}_5C_3 \simeq \mathbb{F}_5 \times \mathbb{F}_{25}$ and $\mathbb{F}_7C_3 \simeq \mathbb{F}_7 \times \mathbb{F}_7 \times \mathbb{F}_7$.
In some cases, as for example for the quaternion group of order 8, the output might be more involved:

```
gap> WedderburnDecomposition( GroupRing(Rationals, QuaternionGroup(8)) );
[ Rationals, Rationals, Rationals, Rationals,
 <crossed product with center Rationals over GaussianRationals of a group of
  size 2> ]
gap> WedderburnDecompositionInfo( GroupRiobjectsng(Rationals, QuaternionGroup(8)) );
[ [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ],
  [ 1, Rationals, 4, [ 2, 3, 2 ] ] ]
```

Here the last entry indicates a 1×1 matrix ring over a cyclic algebra. To understand the notation we can have a look at the documentation of the command:

```
?WedderburnDecompositionInfo
[...]
```

Here it might be useful to use the .html-version of the documentation (or the .pdf version):

```
gap> SetHelpViewer("firefox");
```

which should work on Linux systems and Macs. On windows use `SetHelpViewer("browser");` which should display the documentation in the default browser.

```
?WedderburnDecompositionInfo
Help: Showing 'Wedderga: WedderburnDecompositionInfo'
```

now displays the documentation (of the `GAP` version used) in the corresponding webbrowser. In wedderga 4.9.3 we find the description of the cyclic algebra in Sections 2.1.2 and 9.10.
According to that the simple algebra indicated by `[ 1, Rationals, 4, [ 2, 3, 2 ] ]` $= [n, F, k, [d, \alpha, \beta]]$ is a 1×1 matrix ring over the cyclic algebra A given by

$$\begin{aligned} A &= F(\zeta_d)[u|\zeta_k^u = \zeta_k^\alpha, u^d = \zeta_k^\beta] \\ &= \mathbb{Q}(i)[u|iu = ui^3, u^2 = -1] \simeq \left(\frac{-1,-1}{\mathbb{Q}}\right) \end{aligned}$$

where i and u in A corresponds to i and j in the definition of $\left(\frac{-1,-1}{\mathbb{Q}}\right)$ as given in the lecture of Eric Jespers.
To display the documentation again as per default in the terminal type:

```
gap> SetHelpViewer("screen");
```

(To learn more on how documentations are displayed see the documentation of `SetHelpViewer` by typing `?SetHelpViewer`.)
Recently Allen Herman added functionality to `wedderga` to unfold such a description and obtain a description as matrix ring over a division algebra:

```
gap> WedderburnDecompositionWithDivAlgParts( GroupRing(Rationals, QuaternionGroup(8)) );
[ [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ],
  [ 1, rec( Center := Rationals, DivAlg := true, LocalIndices := [ [ 2, 2 ], [ infinity, 2 ] ],
          SchurIndex := 2 ) ] ]
gap> WedderburnDecompositionWithDivAlgParts( GroupRing(CF(4), QuaternionGroup(8)) );
[ [ 1, GaussianRationals ], [ 1, GaussianRationals ], [ 1, GaussianRationals ],
  [ 1, GaussianRationals ], [ 2, GaussianRationals ] ]
```

implying $\mathbb{Q}Q_8 \simeq 4\mathbb{Q} \times \left(\frac{-1,-1}{\mathbb{Q}}\right)$ and $\mathbb{Q}(i)Q_8 \simeq 4\mathbb{Q}(i) \times M_2(\mathbb{Q}(i))$, where $\left(\frac{-1,-1}{\mathbb{Q}}\right)$ are the Hamiltonian quaternions over $\mathbb{Q}$, see the lecture notes of Eric Jespers, Section 4, for definition. There is a nice article by Herman describing the algorithm used to calculate the global and local Schur indices as used in `wedderga`: `https://arxiv.org/abs/1407.4426`.
`wedderga` also allows to compute central idempotents:

```
gap> S3 := SymmetricGroup(3);
gap> QS3 := GroupRing( Rationals, S3 );;
gap> idems := PrimitiveCentralIdempotentsByCharacterTable(QS3);
[ (1/6)*()+(-1/6)*(2,3)+(-1/6)*(1,2)+(1/6)*(1,2,3)+(1/6)*(1,3,2)+(-1/6)*(1,3),
```

```
  (2/3)*()+(-1/3)*(1,2,3)+(-1/3)*(1,3,2), (1/6)*()+(1/6)*(2,3)+(1/6)*(1,2)+(1/
    6)*(1,2,3)+(1/6)*(1,3,2)+(1/6)*(1,3) ]
gap> IsCompleteSetOfOrthogonalIdempotents(QS3, idems);
true
gap> IsCompleteSetOfOrthogonalIdempotents(QS3, [One(QS3)]);
true
```

So `IsCompleteSetOfOrthogonalIdempotents` does not test whether the idempotents are primitive. It does test neither whether they are central:

```
gap> M := MatrixAlgebra(Rationals, 2);
( Rationals^[ 2, 2 ] )
gap> ElMats := [
     [[1, 0], [0, 0]],
     [[0, 0], [0, 1]]];;
gap> IsCompleteSetOfOrthogonalIdempotents(M, ElMats);
true
```

We can also calculate the value a character takes on a particular idempotent. For this we may write an auxiliary function:

```
ApplyCharacterToGroupRingElement := function(chi, x)
  # takes a character chi of G and an element x of FG (F ANF) and returns chi(x)
  local supp, coeff;
  supp := Support(x);
  coeff := CoefficientsBySupport(x);
  return coeff*List(supp, g -> g^chi);
end;
```

and then we may calculate and display the character table by

```
gap> CT := CharacterTable(S3);
CharacterTable( Sym( [ 1 .. 3 ] ) )
gap> Display(CT);
CT1

     2  1  1  .
     3  1  .  1

       1a 2a 3a
    2P 1a 1a 3a
    3P 1a 2a 1a

X.1     1 -1  1
X.2     2  . -1
X.3     1  1  1

gap> List(idems, e -> List(Irr(CT), chi -> ApplyCharacterToGroupRingElement(chi, e)));
[ [ 1, 0, 0 ], [ 0, 2, 0 ], [ 0, 0, 1 ] ]
gap> C3 := CyclicGroup(3);
<pc group of size 3 with 1 generators>
```

```
gap> QC3 := GroupRing(Rationals, C3);
<algebra-with-one over Rationals, with 1 generators>
gap> CT2 := CharacterTable(C3);
CharacterTable( <pc group of size 3 with 1 generators> )
gap> idems2 := PrimitiveCentralIdempotentsByCharacterTable(QC3);
[ (1/3)*<identity> of ...+(1/3)*f1+(1/3)*f1^2, (2/3)*<identity> of ...+(-1/3)*f1+(-1/3)*f1^2 ]
gap> List(idems2, e -> List(Irr(CT2), chi -> ApplyCharacterToGroupRingElement(chi, e)));
[ [ 1, 0, 0 ], [ 0, 1, 1 ] ]
```

Acknowledgement. Several examples are based on the examples in the documentations of the corresponding GAP packages.