A PACKAGE FOR COMPUTATIONS IN THE REPRESENTATION THEORY OF HOPF ALGEBRAS

BY

KATHLEEN DAKOTA WHITE

A Thesis Submitted to the Graduate Faculty of

WAKE FOREST UNIVERSITY GRADUATE SCHOOL OF ARTS AND SCIENCES

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF ARTS

Mathematics and Statistics

May, 2021

Winston-Salem, North Carolina

Approved By:

Frank Moore, Ph.D., Advisor

Ellen Kirkman, Ph.D., Chair

Sarah Raynor, Ph.D.

Acknowledgments

To Dr. Frank Moore, the `Macaulay2` wizard and a truly incredible research advisor, thank you for your endless patience, support, and guidance. I am forever grateful that I was able to grow as a mathematician under your extremely knowledgeable and wonderfully kind wings. It's rare to find an advisor who understands how critical laughter is to success. May your coffee cup never be empty.

To the esteemed doctors Ellen Kirkman and Sarah Raynor, thank you for not only serving on my thesis committee but for also being impeccable mentors and role models. Beacons, both of you, assuring me that there is light at the end of the academic tunnel.

To my parents, thank you both for your endless love and support, especially the impeccably timed care-packages and Sunday morning Zoom coffee dates.

To Logan Gray and Sarah Britton, my work wife and house spouse respectively, thank you for sticking with me through the highs and the lows. Friends who make sure you eat, sleep, and breathe are invaluable. I'm also fairly certain you're the little elves who snuck in at night to finish this thesis.

Table of Contents

Abstract

Kathleen Dakota White

Our goal is to understand the representation theory of a Hopf algebra, specifically the construction of the character table for semisimple almost cocommutative Hopf algebras. The end product of our work is a software package `HopfAlgebras2` for the computer algebra system `Macaulay2` with the functionality to automate the computation of the character table and other important representation-theoretic information about a Hopf algebra.

Chapter 1: Introduction

In this paper, we explore the infrastructure for representation theory of Hopf algebras with emphasis on Witherspoon's construction for the character table of a Hopf algebra. The final result is a package `HopfAlgebras2` for `Macaulay2` that automates the process of computing the representation theory information for a given Hopf algebra. The updated version of the package realizes the following:

- enhanced functionality and optimized run-time by lifting computations to the new `FinDimAlgebra` data type and corresponding functions
- solving for primitive central idempotents in a finite dimensional algebra and their corresponding representations and characters
- constructing the various components to a few of Witherspoon's theorems that function analogously to the conjugacy class of a group
- computing the character ring of the Hopf algebra
- a `RepresentationData` data type that caches and returns all of the critical data used to construct the character table (including the character table)
- methods for checking the orthogonality relations and class equation

Representation theory for groups is a well-studied subject and classical character tables are a widely used tool. As representation theory for Hopf algebras is relatively new, the computational infrastructure is not as robust. This package therefore fills a gap in current computational algebra software, broadening the accessibility of representation theory of Hopf algebras.

Chapter 2: Background and Motivation

We begin with a discussion of the algebraic structures of which the foundation for `HopfAlgebras2` package subsists. For the rest of the paper we assume that the reader has completed a standard sequence in abstract algebra and that k is a field. Starting with a review of representation theory for groups, we move on to the construction of an algebra via ring theory and tensor products before covering algebra representations and several important decomposition results. This chapter thus reviews the necessary infrastructure for building to a Hopf algebra and finding the requisite representation theory information for computing its character table.

2.1 Representation Theory of Groups

In this section we review the construction of a character table for a group and its importance as a tool in representation theory of groups.

Definition 2.1. *A **representation** of a group G on V, a vector space over a field k, is a homomorphism $\rho : G \to GL(V)$ where $GL(V)$ is the general linear group over V.*

Every group has the **trivial representation** ρ_{triv} which is the one-dimensional representation sending every element of G to $1 \in k$. The goal is to understand the building blocks out of which all representations are constructed; this requires the definition below to understand when we can break down a representation.

Definition 2.2. *Let W be a subspace of V. If $gW = W$ for all $g \in G$ then W is a **G-invariant subspace** of V. We define a **subrepresentation of** ρ as $\rho|_W := G \to GL(W)$.*

Definition 2.3. *If V has no proper G-invariant subspaces, then ρ is said to be* ***irreducible****. Otherwise, ρ is* ***reducible****.*

Another term that is frequently substituted for irreducible is **simple**. These irreducible representations serve as the aforementioned building blocks. The following theorem, oft referred to as Maschke's Theorem, states that group representations always decompose into irreducible representations.

Theorem 2.4. *[Art91] Every representation of a finite group G on a nonzero, finite-dimensional complex vector space is a direct sum of irreducible representations.*

The above holds also when the characteristic of k does not divide the order of the group, a generalization that is useful when moving from groups to algebras.

2.1.1 Characters

As important as representations are, working with maps into different general linear spaces can be unwieldy. In order to distill the critical information about a representation and how it acts on a group, we turn to characters.

Definition 2.5. *Given a representation ρ, the* ***character*** *of $\rho : G \to GL(V)$ is $\chi_V(g) = Tr(\rho(g), V)$. The value of $\chi_V(1)$ is the dimension of the representation, also referred to as the dimension of the character. If ρ is irreducible then χ_V is called an* ***irreducible character****.*

From the definition we can draw a few important conclusions. First, one dimensional representations and their corresponding characters are the same map. If ρ is one-dimensional then it sends every element to a one-by-one matrix M and thus the trace is equivalent to the matrix. Another term for one-dimensional characters is **linear characters** and we will use this term interchangeably with one-dimensional

representations when applicable (most of the time it will be more beneficial to think of the representations as characters). It also follows that one-dimensional characters are therefore homomorphisms, though any higher dimensional characters are not. Since only the linear characters are homomorphisms, we cannot rely on acting on solely group's basis elements to inform how the representations act on the group as a whole. However, we want to avoid needing to compute the character of every individual element of a group. We thus turn to the notions of conjugate elements and conjugacy classes.

Definition 2.6. *Let G be a group. We say a and b and* ***conjugate*** *if there exists $g \in G$ such that $a = gbg^{-1}$. This relation is an equivalence relation and partitions G into* ***conjugacy classes****. Therefore, whenever a and b are conjugate elements, they belong to the same conjugacy class.*

Theorem 2.7. *Let G be a group, ρ a representation, χ its corresponding character, C a conjugacy class of G, and V, W representations of G. Then the following statements hold.*

- $\chi(g) = Tr(\rho(g))$ *is the same* $\forall g \in C$ *(characters are constant on conjugacy classes)*
- $\chi_V + \chi_W = \chi_{V \oplus W}$
- $\chi_V \chi_W = \chi_{V \otimes W}$

Since the conjugacy classes form a partition, we can pick a representative element from each class. Then by the first item above, computing the character on the class representative tells us how the corresponding representation interacts with all elements of the class. So for any group G, we can describe how the irreducible representations act on the group by computing the corresponding characters on the

the conjugacy classes of G.

The second and third items tell us that irreducible characters are closed under pointwise addition and multiplication respectively. At the level of basis elements, the direct sum action is a disjoint union and the tensor product (to be covered later) is a Cartesian product. Most significantly, the closure under both operations imposes a group structure on the characters of a group. Therefore, when decomposing a group into irreducible representations, we are viewing the group as a group of its irreducible characters. Much of the work we do later in building the structure of a Hopf algebra will be to ensure that we can view the Hopf algebra as an algebra of its irreducible characters. We continue with more properties of characters over finite groups.

Theorem 2.8. *Let G be a finite group. Then the following are true:*

1. *The number of isomorphism classes of irreducible representations is the same as the number of conjugacy classes.*
2. *If $\rho_1, ..., \rho_n$ are the isomorphism classes of irreducible representations of G with corresponding characters $\chi_1, ..., \chi_n$, then the dimension d_i of χ_i divides the order of G and $|G| = \sum_{i=1}^{n} d_i^2$.*

As a corollary, we obtain the following theorem regarding representations of a finite abelian group.

Theorem 2.9 ([Art91])**.** *Let G be a finite abelian group. Then there exists $|G|$ many non-isomorphic one-dimensional representations of $|G|$.*

Therefore, when G is abelian, we automatically know how many one-dimensional representations to expect. We would like to translate this correlation between "abelianness" and representations to non-abelian groups. Specifically, if N is a normal subgroup of G, we can use representations of G/N to obtain representations of G.

Theorem 2.10 ([Art91])**.** *Let G be a group, N a normal subgroup, $\pi : G \to G/N$ the canonical projection map. Consider a representation of G/N, $\rho : G/N \to GL(V)$. The composition $\rho\dot{\pi} = \rho' : G \to G/N \to GL(V)$ is a representation of G. If ρ is an irreducible representation of G/N, then ρ' is an irreducible representation of G.*

In particular, if $N = [G, G]$, the commutator subgroup, then G/N is abelian, and the previous two theorems yield $[G : [G, G]]$ many one-dimensional representations of G. For the converse, if $\rho : G \to \mathbb{C}^*$ is any one-dimensional representation, $[G, G] \subseteq ker(\rho)$, and so ρ induces a homomorphism $\overline{\rho} : G/[G, G] \to \mathbb{C}^*$. Thus when $N = [G, G]$, G/N offers an easier way to find the one-dimensional representations and corresponding linear characters of G. Furthermore, we can use our knowledge about the commutator to determine how many one-dimensional representations to expect.

Theorem 2.11. *The number of one-dimensional representations of G is the index of the commutator subgroup $[G, G]$.*

The above result is particularly useful as it generates an approximation for how close to being abelian a given group is. The best case is when the index is equal to the order of the group, implying that G is abelian, and the worst case is when the index is equal to one, implying that no elements but the identity are central. If the index is in between these two values, the closer the value is to the order of the group, the more abelian the group and the easier it is to decompose into irreducible representations. We then close the section with another result related to the number and dimension of conjugacy classes, the class equation.

Definition 2.12. *The **class equation** of a group G with n conjugacy classes is $|G| = \sum_{i=1}^{n} |cc(g_i)|$, where $g_1, \ldots, g_n$ is a complete list of representatives of the conjugacy classes, and $cc(g_i)$ denotes the conjugacy class of g_i.*

This result allows us to take partial information about the number and size of known conjugacy classes and make guesses about the remaining classes. Since the number of conjugacy classes is equal to the number of irreducible representations, we can also combine this with the former result about one-dimensional representations to support finding the higher dimensional irreducible representations of our group.

2.1.2 Character Table

The **character table** of a group G assembles the irreducible characters, conjugacy classes, and values of the characters into a table. The rows are labeled by the irreducible characters, the columns by representatives of the conjugacy classes, and the entries are then the values of the character on said conjugacy classes. It is convention to have the first row correspond to the trivial character and the first column correspond to the identity. This ensures the entries in the first row are all one and the first column is the dimension of the irreducible representations.

It is important to note that an incomplete character table is still an extremely useful tool using the natural Hermitian product on characters,

$$\langle \chi_1, \chi_2 \rangle = \frac{1}{|G|} \sum_g \overline{\chi_1(g)} \chi_2(g).$$

For the character table we are only concerned with the irreducible characters, which makes the following result about the inner product on the irreducible characters very useful.

Theorem 2.13. *Irreducible characters are orthonormal under the Hermitian product.*

Applying the above theorem yields

$$\langle \chi_i, \chi_j \rangle = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$$

and thus the rows of the character table are orthogonal to each other. It is important when checking this relation over any table with complex entries to remember that the first character is conjugated. We can find a specific inner product definition for the columns as well where the sum is defined as (for every pair of elements g, h)

$$\sum_i \overline{\chi_i(g)}\chi_i(h) = \begin{cases} |C_G(g)| \text{ if } g, h \text{ are conjugate} \\ 0, \text{ else} \end{cases}$$

Since $g, h \in G$ are conjugate if and only if they are in the same conjugacy class then the columns of the character table are orthogonal as well.

It is therefore possible to complete the character table when only some of the irreducible representations or characters are known. Additional uses for the character table in representation theory of groups include finding the order of the centralizers, the order of the group, or decomposing an unknown character into a direct sum of known irreducible representations. The utilization of incomplete information is a huge boon, making the character table integral to the representation theory of groups, and later, more abstract algebraic structures including the focus of this paper, Hopf algebras.

Example

The **character table** compiles all of the representation theory information gathered and is constructed by indexing rows by characters, columns by conjugacy classes, and the entries are the values of the characters on the conjugacy classes. Let G be a group with n conjugacy classes – denoted by elements $e_G, d_2 \ldots, g_n$ and of dimension $d_1, \ldots, d_n$ – and characters $\chi_{triv}, \chi_2, \ldots, \chi_n$ with corresponding dimension $1 = \delta_{triv}, \delta_2, \ldots, \delta_n$. By convention, we order the rows with χ_{triv}, then the rest of the linear characters, and finally the higher order characters. For the columns we start with the conjugacy class determined by the identity of G and then the rest of the

conjugacy classes. Thus the first row of entries will contain only ones (recalling χ_{triv} sends every element to the identity) and the first column will contain the δ_i (recalling that $\chi_{triv}(e_G)$ is the dimension of the representation and thus character). We can then compute the general character table as so:

	d_1	d_2	d_3	$\dots$	d_n
	e_G	g_2	g_3	$\dots$	g_n
χ_{triv}	1	1	1	$\dots$	1
χ_2	δ_2	$\chi_2(g_2)$	$\chi_2(g_3)$	$\dots$	$\chi_2(g_n)$
χ_3	δ_3	$\chi_3(g_2)$	$\chi_3(g_3)$	$\dots$	$\chi_3(g_n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
χ_n	δ_n	$\chi_n(g_2)$	$\chi_n(g_3)$	$\dots$	$\chi_n(g_n)$

To better illustrate the construction, we will compute the table by hand for for D_4, the symmetries of a square, presented below where where e is the identity, r is a rotation by ninety degrees, and s is a reflection.

$$D_4 = \langle r, s \mid r^4 = s^2 = e, srs = r^{-1} \rangle$$

The conjugacy classes of D_4 are

$$\{1\}, \{r, r^3\}, \{r^2\}, \{s, r^2 s\}, \{rs, r^3 s\}$$

and we will use the first element listed as our representative element.

The next step is to decompose D_4 into its irreducible representations. Our first priority is to establish the number of one-dimensional representations which we can do by applying Theorem 2.9. Let $N = \langle r^2 \rangle$, the commutator subgroup of D_4. Modding out by N yields $D_4/N \cong \mathbb{Z}_2 \times \mathbb{Z}_2$, an abelian group of order four. Therefore we have four one-dimensional representations to find which are going to be lifts of maps $\rho : \mathbb{Z}_2 \times \mathbb{Z}_2 \to \mathbb{C}^*$. One of these is the trivial representation ρ_{triv} (the identity map) leaving only three more to find.

Here the order of the elements comes into play as the order of any element must

match its image. We can represent D_4/N as $\langle r\rangle \times \langle s\rangle = \{(1,1),(r,1),(1,s),(r,s)\}$. Thus, as every element but the identity has order two, we can create our three remaining one-dimensional representations as shown via the following table

	$(1,1)$	$(r,1)$	$(1,s)$	(r,s)
ρ_{triv}	1	1	1	1
ρ_1	1	-1	1	-1
ρ_2	1	1	-1	-1
ρ_3	1	-1	-1	1

While that accounts for any of the one dimensional representations, it remains to be seen if we have any higher dimensional representations.

This can be checked via the class equation. Each of the four representations above have dimension one, so the sum of the one-dimensional reps squared is $4 < 8$. Therefore we are missing at least one representation with corresponding dimension d. We can write the class equation as $8 = 1^2 + 1^2 + 1^2 + 1^2 + d^2 \Rightarrow d = 2$ ($2|8$ so that also checks out). Therefore we are looking for a two dimensional irreducible representation of D_4.

With a little bit of work, we could deduce that the final representation is $\rho_4 : D_4 \to GL_2$. However, we can complete the character table simply by using the one-dimensional representations and the properties of the character table. To that end, let $\chi_{triv}, \chi_1, \chi_2, \chi_3, \chi_4$ be the characters corresponding to their respective representations. Recalling that the first column is always the trace of the identity, we know that the first entry in the row for ρ_4 must be two. We can then partially construct the character table:

	(1) 1	(2) r	(2) s	(1) r^2	(2) rs
χ_{triv}	1	1	1	1	1
χ_1	1	-1	1	1	-1
χ_2	1	1	-1	1	-1
χ_3	1	-1	-1	1	1
χ_4	2				

Based on the orthogonality relations, we know $\langle \chi_i, \chi_4 \rangle = 0$ and that the dot product of the column denoted by r^2 with any other column is also zero. Consider the following series of equations.

$$(1 \cdot 1) + (1 \cdot -1) + (1 \cdot 1) + (1 \cdot -1) + 2\chi_4(r) = 0 + 2\chi_4(r) \qquad \Rightarrow \chi_4(r) = 0$$

$$(1 \cdot 1) + (1 \cdot 1) + (1 \cdot -1) + (1 \cdot -1) + 2\chi_4(s) = 0 + 2\chi_4(s) \qquad \Rightarrow \chi_4(s) = 0$$

$$(1 \cdot 1) + (1 \cdot 1) + (1 \cdot 1) + (1 \cdot 1) + 2\chi_2(r^2) = 4 + 2\chi_4(r^2) \qquad \Rightarrow \chi_4(r^2) = -2$$

$$(1 \cdot 1) + (1 \cdot -1) + (1 \cdot -1) + (1 \cdot 1) + 2\chi_4(rs) = 0 + 2\chi_4(rs) \qquad \Rightarrow \chi_2(rs) = 0$$

Therefore, we take the completed character table to be

	(1) 1	(2) r	(2) s	(1) r^2	(2) rs
χ_{triv}	1	1	1	1	1
χ_1	1	-1	1	1	-1
χ_2	1	1	-1	1	-1
χ_3	1	-1	-1	1	1
χ_4	2	0	0	-2	0

Thus we are able to compile all of the representation theory for a group into a single construct. This is a well-understood construct and widely used tool for groups, where the underlying set has no algebraic structure. If we impose the structure of a vector space on the set, we can generalize from groups to algebras. Hopf algebras arise from imposing all of the "nice" structure and relations available in a group. Therefore the purpose of this paper is to understand how to analogously compile the representation theory of a Hopf algebra into a character table and create a software package to

manage the computations. After briefly reviewing some linear algebra terms, the rest of this chapter will work to build a foundation for the representations theory of algebras which later chapters will grow into the representation theory of Hopf algebras.

2.1.3 Dual Spaces

Since the underlying structure of the set in an algebra is a vector space, working in representation theory of algebras requires working with maps on and into vector spaces. We therefore recall the following definitions from linear algebra concerning spaces of maps on vector spaces.

Definition 2.14. *Let V be a k-module. The **dual space of V** is the set of all linear maps $\varphi : V \to k$, denoted $V^* = Hom(V, k)$. We also define $L(V) = Hom_k(V, V)$.*

The left entry of the parenthesis is called the contravariant slot and the right entry is the covariant slot.

Definition 2.15. *Let V and W be k-vector spaces. Then we can generalize the above to $L(V, W) = Hom_k(V, W)$, where the elements are linear maps $\varphi : V \to W$.*

Note that unlike $GL(V)$, these maps are only linear and not necessarily bijective.

2.2 Algebras and the Tensor Product

As mentioned previously, one obtains the definition of an algebra by replacing the a binary operation defined on the set G with a binary (and bilinear) operation defined on a vector space V over a field k. While we can create more generalized representations of groups that allow for bijective "symmetries" amongst algebraic objects, when we change our underlying space to an algebra we can force linearity among the operations.

In this section we will introduce the structure and notation necessary for working with algebras, namely modules and the tensor product.

Definition 2.16. *Let V, W, and Y be vector spaces over a field k. Then a map $\phi : V \times W \to Y$ is **bilinear** if $\forall\ v \in V$, $w \in W$:*

$\phi_w : V \to Y$ such that $v \longmapsto \phi(v, w)$ is a linear map

$\phi_v : W \to Y$ such that $w \longmapsto \phi(v, w)$ is a linear map

Such a map preserves addition and scalar multiplication in both slots of the Cartesian Product. We can now define an algebra.

Definition 2.17. *Let k be a field and A a k-vector space.Then A is an **algebra** if there exists an k-bilinear map $\mu : A \times A \to A$ that is associative with identity.*

The issue with the above definition of an algebra is that bilinear maps are not vector space homomorphisms. To understand bilinear maps in terms of linear ones, we use the notion of a tensor product.

Returning to the notation in Definition 2.16, let V, W, and Y be vector spaces over k. Let $F(V \times W)$ be the vector space with basis given by the set $V \times W$ (there are no relations yet). We then construct a subspace I out of the bilinear properties we want to think of as linear relations. That is, $\forall v, v' \in V$, $w, w' \in W$, and $c \in k$, set I equal to the k-span of the following.

$$(v + v', w) - (v, w) - (v', w)$$

$$(v, w + w') - (v, w) - (v, w')$$

$$c(v, w) - (cv, w)$$

$$c(v, w) - (v, cw).$$

By construction, the elements of I are always in the kernel of any k-linear map $F(V \times W) \to Y$ induced by any k-bilinear map $V \times W \to Y$. Therefore we take the quotient space $F(V \times W)/I$ to get an induced map from $F(V \times W)/I \to Y$.

Definition 2.18. *Let V, W, $F(V \times W)$, and I be as defined above. Then $F(V \times W)/I$ is the* ***tensor product over k*** *of V and W, denoted $V \otimes_k W$. When k is understood, it is often dropped from the tensor product.*

We write $v \otimes w$ to denote the coset $(v, w) + I$ in $V \otimes_k W$. Such an element is called an **elementary tensor**, the set of which spans $V \otimes W$. The construction above proves the following result.

Theorem 2.19. *(Universal Property of Tensor Products) Let V, W, Y, $F(V \times W)$, and I be as defined above. Let i be in the inclusion map, $i : V \times W \to V \otimes W$, $(v, w) \longmapsto v \otimes w$. Suppose there is a bilinear map $\phi : V \times W \to Y$. Then $\exists!$ linear transformation $\psi : V \otimes W \to Y$ where $\psi = \phi \circ i$ and the following diagram commutes.*

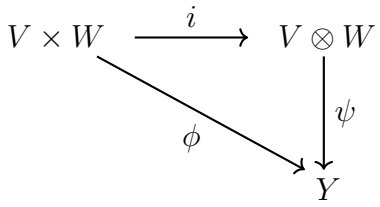

Figure 2.1: Tensor Product

The Universal Property of Tensor Products ensures that given a bilinear map, there exists a unique corresponding linear map out of the tensor product. We will now redefine an algebra using the tensor product as in Montgomery's text, which will make it easier to build up to the formal definition of a Hopf algebra.

Definition 2.20. *A k-**algebra** is a k-vector space A paired with k-linear maps **multiplication** $\mu : A \otimes A \to A$ (associative) and **unit** $\eta : k \to A$ (unit) such that the following diagrams commute.*

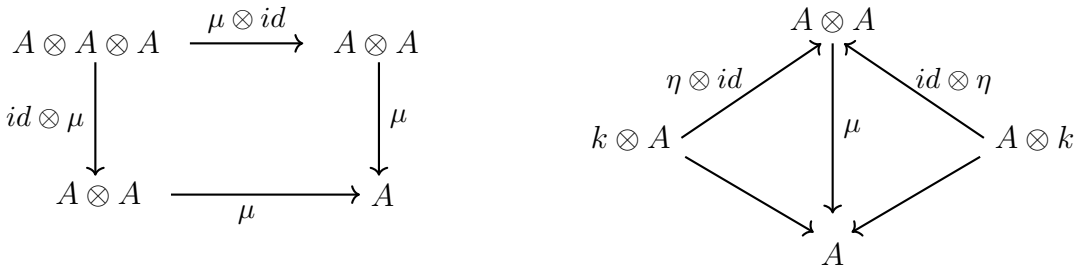

Figure 2.2: Associativity and 2-sided unit

For the purposes of this paper we will only be considering finite dimensional algebras.

2.3 Representation Theory of Algebras

Now that we have a background in both representation theory of groups and the structure of an algebra, we can begin to think about algebra representations.

Definition 2.21. *Let k be a field and A a k-algebra. An **algebra representation** is an algebra homomorphism $\rho : A \to L(V)$ for V a k-vector space. The corresponding **character** is $\chi_V = \mathrm{Tr}(\rho(a), V)$.*

The character is thus the trace of the composition of ρ and the appropriate map from $L(V)$. Rather than thinking of vector spaces over k equipped with such a homomorphism, we transition the notion of A-modules.

Definition 2.22. *Let k be a field and A a k-algebra. Then a* ***left A-module*** *M consists of a k-vector space M and an operation*

$$\cdot : A \times M \to M$$

such that $\forall m_1, m_2 \in M$ and $r, s \in A$ the following hold

$$r(m_1 + m_2) = r \cdot m_1 + r \cdot m_2,$$

$$(r + s)m_1 = r \cdot m_1 + s \cdot m_2,$$

$$(rs)m_1 = r(s \cdot m_1),$$

$$1 \cdot m_1 = m_1.$$

Thus, the algebraic structure of a vector space over a field generalizes to a module over a k-algebra. A right A-module is defined analogously by elements acting on the right. We also retain the notions of subrepresentations as submodules and simple representations as simple submodules.

Definition 2.23. *Let M be a module for a k-algebra A and N a subset of M. Then N is a* ***submodule*** *if N is closed under addition and the A-module action.*

Definition 2.24. *Let M be a module for a k-algebra A. We say that M is a* ***simple module*** *if M is non-zero and has no non-zero submodules.*

As the above definitions show, the shift from groups to algebras is not a complete departure from our original representation theory set-up. Conceptually, the set-up is the same, with the difficulty residing in re-formulating concepts to match the new underlying structure and retain linearity. In fact, the parallels are so well-defined that we can draw a direct connection between representations of groups and those of algebras.

Definition 2.25. *$k(G)$, also known as the **group algebra**, is the set of all linear combinations $\sum_{g\in G} a_g g$ for $a_i \in k, g_i \in G$ with*

1. *e_g the unit in kG*
2. $\sum_{g\in G} a_g g + \sum_{g\in G} b_g g = \sum_{g\in G} (a_g + b_g)g$
3. $c\sum_{g\in G} a_g g = \sum_{g\in G} (ca_g)g$
4. $\left(\sum_{g\in G} a_g g\right)\left(\sum_{g\in G} b_g g\right) = \sum_{g\in G}\left(\sum_{h\in G} (a_h b_{h^{-1}g})\right) g$

Recalling our earlier definition of representations and the relation between modules and vector spaces, there exists a natural bijection between kG-modules and vector spaces over k paired with a representation, $\varphi : G \to GL(V)$. This homomorphism thus gives a homomorphism from kG into the space of linear transformations on V, $L(V)$. In other words, given $\rho : G \to GL(V)$, we can always define $\tilde{\rho} : k(G) \to L(V)$ by setting $\tilde{\rho}(g) = \rho(g)$ and extending by linearity.

One of the most important results in group representation theory is Maschke's Theorem, which guarantees that every representation of a finite group G decomposes as a direct sum of irreducible representations. For certain algebras over an algebraically closed field (categorized as semisimple), a similar result holds, explored in the next section.

2.4 Wedderburn Decomposition

The following theorem gives us equivalent conditions for how a finite dimensional algebra can be decomposed.

Theorem 2.26 ([Dum04])**.** *Let R be a finite dimensional k-algebra with k algebraically closed. Then the following statements concerning modules over the algebra, R-modules, are equivalent:*

1. *any R-module is completely reducible*
2. *R considered as a left R-module is a direct sum, $R = \bigoplus_{i=1}^{n} L_i$ with L_i a simple module, $L_i = Re_i$ for $e_i \in R$ such that*
   (a) $e_i e_j = 0, i \neq j$
   (b) $e_i^2 = e_i$ *(idempotent)*
   (c) $\sum_{i=1}^{n} e_i = 1$
3. *as a ring, R is isomorphic to a direct sum of matrix rings over k.*

In particular, the second statement will be crucial in how the package creates the Hopf algebra's representation ring and character table.

Chapter 3: Coalgebras and Bialgebras

With the results from the previous chapter, we successfully generalized from a group to an algebra and from group representations to algebra representations. However, currently these representations are only closed under addition, not multiplication. Our goal for this chapter is to begin building the necessary structure to close representations under multiplication.

3.1 Coalgebras

The first step towards closure is to define how to map from an algebra to an algebra. Let A be a k algebra with multiplication and unit maps, $\mu : A \otimes A \to A$ and $\eta : k \to A$ respectively. If we wanted to think about the linear dual on A, we need maps that let us move from A into $A \otimes A$. To do this, we construct a coalgebra by creating the categorical dual of A. That is, we construct a coalgebra with the same underlying structure diagrams as A except that the arrows (and thus maps) have all been reversed.

Definition 3.1. *A **coalgebra** is a k-module C paired with k-linear maps **comultiplication** $\Delta : C \to C \otimes C$ (coassociative) and **counit** $\epsilon : C \to k$ (counit) such that the following diagrams commute.*

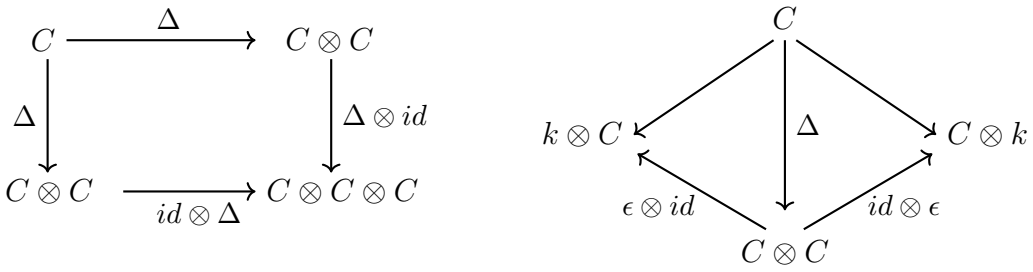

Figure 3.1: Coassociativity and 2-sided counit

With the added structure of a coalgebra, we can now define algebra dual spaces by letting an algebra A act in place of k. Since there is no natural definition for this composition of maps, we use the following operation.

Definition 3.2. *Let A be an algebra and C a coalgebra and consider $Hom_k(C, A)$. The **convolution product**, for all $f, g \in Hom_k(C, A)$ and $c \in C$ is*

$$(f * g)(c) = \mu \circ (f \otimes g)(\Delta c)$$

In particular, we can impose an algebra structure on $\mathrm{Hom}_k(C, A)$ using the convolution product as our binary operation. Referred to as the **convolution algebra**, $(\mathrm{Hom}_k(C, A), *)$ has unit $\eta\epsilon$. Similarly, if we revert to k instead of A, the dual of a coalgebra, C^*, is an algebra.

We also need to be careful with our definition of the comultiplication (or coproduct) map. The coproduct of an element, is an element of $C \otimes C$, which can be written as $\Delta(c) = \sum_{i=1}^{n} a_i \otimes b_i$. To avoid adding variables, we can instead write $\Delta(c) = \sum_{i=1}^{n} c_{i,(1)} \otimes c_{i,(2)}$, leaving a summation over "i". To clean up the final summation we use the following notation.

Definition 3.3. *Let C be a coalgebra. Then the **Sweedler notation** for Δ and any $c \in C$ is*

$$\Delta(c) = \sum c_{(1)} \otimes c_{(2)}$$

It is understood that our leading "c" is folded into the terms in the tensor.

3.2 Bialgebras

Definition 3.4. *Let V and W be k-modules. Then the* ***twist map*** *is $\tau : V \otimes W \to W \otimes v$, $\tau(v \otimes w) = (w \otimes v)$ for $v \in V$ and $w \in W$.*

Definition 3.5. *A* ***bialgebra*** *is a k-module B such that (B, μ, η) is an algebra, (B, Δ, ϵ) is a coalgebra, and μ and η are coalgebra morphisms or, equivalently, Δ and ϵ are algebra morphisms. In other words, the following diagrams commute:*

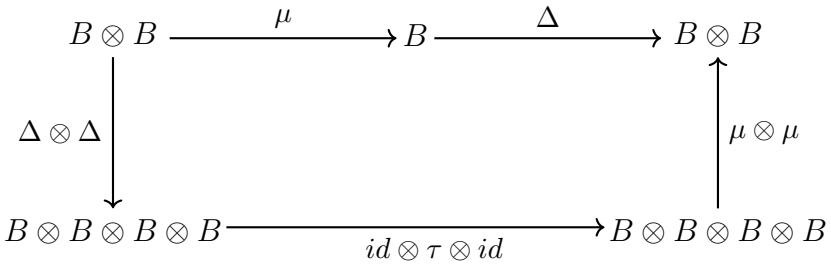

Figure 3.2: μ and Δ preserve each other

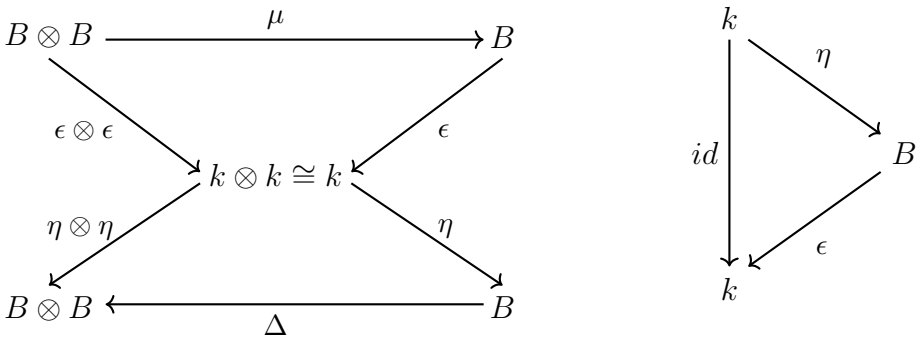

Figure 3.3: μ preserves ϵ and Δ preserves η; η and ϵ preserve each other

Definition 3.6. *Let B be a bialgebra and $b \in B$. We call b a* ***group-like*** *if $\Delta(b) = b \otimes b$.*

This definition actually only requires a coalgebra though we need the extra bialgebra structure before it becomes useful for us.

Specifically, the group-likes are useful when it comes to finding one dimensional representations. A nice result is that if B is a bialgebra then $B^* = \mathrm{Hom}_k(B, k)$ is not only algebra under the convolution product, it is a bialgebra. As mentioned in the section on representation theory of groups, finding the one-dimensional representations and corresponding linear characters is key to understanding a space's representations and character table. A group-like element of the dual is a one-dimensional representation and therefore presented an initial method for finding the linear characters. The final version of the code follows Witherspoon's construction, which relies on idempotents instead of group-likes.

3.3 Representation Ring of a Bialgebra

Previously, in the group case, we were able to construct the character table without consulting the representation ring. The interplay between the various data points made it unnecessary. However our code relies on the structure of the character ring in both the code for the character table and for checking the class equation.

Definition 3.7. *If V and W are representations of a bialgebra B, the tensor product $V \otimes W$ has a structure of a representation as well via the formula*

$$b(v \otimes w) = \sum b_{(1)}v \otimes b_{(2)}w,$$

for any $b \in B$, $v \in V$ and $w \in W$.

So let B be a finite dimensional, semisimple bialgebra. Since we're semisimple, we know there exists a decomposition of irreducible representations by Wedderburn's

Decomposition. Let $R(B) = \text{span}_{\mathbb{C}}\{[\rho_0], [\rho_1], \ldots, [\rho_n]\}$ where ρ_0 is always the trivial representation. A general element is a linear combination of the isomorphism classes of the irreducible representations: $a_0[\rho_0] + a_1[\rho_1] + \cdots + a_n[\rho_n]$.

These isomorphism classes are actually B-modules which allows us to define a relation between the direct sum of modules and addition of the representations, for example: $[\rho_i \oplus \rho_j] = [\rho_i] + [\rho_j]$. In general if V is a representation of B, $[V] = a_0[\rho_0] + a_1[\rho_1] + \cdots + a_n[\rho_n]$ where $V = \rho_0^{(a_0)} \oplus \rho_1^{(a_1)} \oplus \cdots \oplus \rho_n^{(a_n)}$. Similarly, we can define the product of two classes as the class of the tensor of the two representations, $[V][W] = [V \otimes W]$. Thus, we are able to to formally define $R(B)$ as a ring.

Definition 3.8. *$R(B)$, the **representation ring** of B, is the with identity given by the trivial representation and addition and multiplication given as follows for v, W representations of B*

1. $[V \oplus W] = [V] + [W]$
2. $[V][W] = [V \otimes W]$

However, what we end up using in the code for the character table is the character ring. Recall that $B^* = \text{Hom}_k(B, k)$, an algebra under the convolution product. Consider the map $\phi : R(B) \to B^*$ where, for V a representation, $V \longmapsto \chi(V) : b \longmapsto \text{Tr}(b, v)$. The map ϕ is a ring homomorphism such that $\phi([V \oplus W]) = \chi_V + \chi_W$ and $\phi([V \otimes W]) = \chi_V * \chi_W$ where $*$ is the convolution product. We can define the **character ring**, $C(H)$ as the image of ϕ. We will return to this when we discuss the construction of the character table.

Chapter 4: Hopf Algebras

With the structure of a bialgebra we are able to define representations and a structure on them closed under addition and multiplication. Specifically, when we use the convolution product from the last section we can impose an algebra structure. Ideally however, it would be beneficial to be able to build new representations out of existing ones.

Consider kG, the group algebra. We can define an action on $\mathrm{Hom}_k(V, W)$ as $(g \cdot f)(v) = gf(g^{-1}(v))$, reminiscent of conjugation. We would like to define a similar action on $\mathrm{Hom}_k(V, W)$ (V, W, bialgebra modules) that respects the tensor product on representations. In other words, an action such that it plays nicely with $\mathrm{Hom}_k(V, W) \cong V^* \otimes W$, where $V^* = \mathrm{Hom}_k(V, k)$ denotes the k-linear dual. However, we have no formal definition of inverses or conjugation with a bialgebra. Thus we introduce the antipode map and take the necessary induced structure on a bialgebra to form a Hopf algebra.

Definition 4.1. *Let $(H, \mu, \eta, \Delta, \epsilon)$ be a bialgebra. H is a **Hopf algebra** if there exists a unique **antipode** $S \in \mathrm{Hom}_k(H, H)$ such that S is the inverse to id_H under the convolution product. Note that for all $h \in H$, S satisfies*

$$\sum (Sh_1)h_2 = \epsilon(h)1_H = \sum h_1(Sh_2)$$

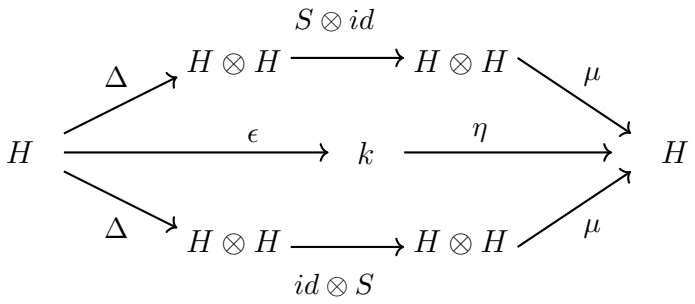

Figure 4.1: S is a left and right inverse of id under convolution

The definition of a Hopf algebra is therefore a categorification of the concept of an inverse. In particular, for g a group-like of H, $S(g)$ is exactly g^{-1}.

Theorem 4.2. *[Mon93] Let H be a Hopf algebra and S its antipode. Then*

1. *S is an anti-algebra homomorphism*

$$S(hk) = S(k)S(h) \forall h, k \in H,\ S(1) = 1$$

2. *S is an anti-coalgebra homomorphism*

$$\Delta \circ S = \tau \circ (S \otimes S) \circ \Delta,\ \epsilon \circ S = \epsilon$$

We can now return to our earlier question of defining an action on $\mathrm{Hom}_k(V, W)$ for V, W H-modules. Let $v \in V, h \in H, f \in \mathrm{Hom}_k(V, W)$. Then we can define the action as

$$(h \cdot f)(v) = \sum h_{(1)} f(S(h_{(2)})v).$$

In the special case $W = k$, we have:

$$\begin{aligned}(h \cdot f)(v) &= \sum h_{(1)} f(S(h_{(2)})v) \\ &= \sum \epsilon(h_{(1)}) f(S(h_{(2)})v) \\ &= f\Big(\sum \epsilon(h_{(1)}) S(h_{(2)})v\Big) \\ &= f(S\Big(\sum \epsilon(h_{(1)})h_{(2)}\Big)v) \\ &= f(S(h)v).\end{aligned}$$

4.1 Hopf Algebra Examples

4.1.1 Group Algebra

The group algebra, kG, is a Hopf algebra. The appropriate maps are

- $\mu : kG \otimes kG \to kG$, defined by $\mu(g \otimes h) = gh$.
- $\eta : k \to kG$, defined by $\eta(1) = e_G$.
- $\Delta(g) : kG \to kG \otimes kG$, $\Delta(g) = g \otimes g$
- $\epsilon : kG \to k$, $\epsilon(g) = 1 \ \forall g \in G$
- $S(g) = g^{-1}$, $\forall g \in G$ (all elements of G are group-like)

4.1.2 Dual of a Group Algebra

The dual of kG, $kG^* = \mathrm{Hom}_k(kG, k)$ is also a bialgebra.

- $\mu : kG^* \otimes kG^* \to kG^*$, defined by $(\alpha * \beta)(g) = \alpha(g)\beta(g)$.
- $\eta : k \to kG^*$, defined by $\eta(1) = \epsilon$, the counit of kG.

- $\Delta : kG^* \otimes kG^* \otimes kG^*$, defined by $\Delta(\alpha_g) = \sum_{\substack{a,b \in G \\ ab=g}} \alpha_a \otimes \alpha_b$, where α_g is the functional defined by $\alpha_g(h) = \delta_{gh}$.
- $\epsilon : kG^* \to k$ defined by $\epsilon(\alpha) = \alpha(e_G)$.
- $S : kG^* \longrightarrow kG^*$, defined by $\alpha_g \longmapsto \alpha_{g^{-1}}$.

4.1.3 H_8

As nice as the above two examples are, due to their group structure, neither are particularly illuminating. The Kac-Palyutkin Hopf algebra [KP66] H_8 is the smallest dimensional Hopf algebra that is neither a commutative or cocommutative Hopf algebra nor a twist of one that is commutative or cocommutative. It is therefore the first example of an algebra possessing this new structure; one may call it the S_3 of (semisimple) Hopf algebras.

$$H_8 = k\langle x, y, x\rangle / \langle 1 - x^2, 1 - y^2, \frac{1}{2}(1 + x + y - xy) - z^2, xy - yx, zx - yz, zy - xz\rangle$$

The Δ, ϵ, and S structures are

- $\Delta(x) = x \otimes x$, $\Delta(y) = y \otimes y$, $\Delta(z) = \frac{1}{2}(1 \otimes 1 + 1 \otimes x + y \otimes 1 - y \otimes x)(z \otimes z)$
- $\epsilon(x) = \epsilon(y) = \epsilon(z) = 1$
- $S(x) = x$, $S(y) = y$, $S(z) = z$

Chapter 5: Representation Theory of Hopf Algebras

We are now able to discuss representation theory of Hopf algebras, specifically, the construction of their character tables. Our code is automating the character table construction outlined in Sarah J. Witherspoon's paper "The Representation Ring and the Centre of a Hopf Algebra". Witherspoon's construction is not the only available construction for finding characters of a Hopf algebra. Martin Lorenz's "Representations of Finite-Dimensional Hopf Algebras" is another such example. However, we found that Witherspoon's construction is more analogous to the classical construction for groups. Since the scope of the paper goes beyond just the construction of character tables, we the following two sections describe the two algorithms we used as a foundation for the code. The final section of this chapter presents formal examples.For the purposes of this construction, we are concerned only with finite dimensional semisimple almost cocommutative algebras.

Definition 5.1 ([Dum04])**.** *A finite dimensional Hopf algebra is* ***semisimple*** *if it meets the criteria for Wedderburn's Decomposition.*

Definition 5.2. *Let H be a Hopf algebra with comultiplication map Δ. Recall the twist map, τ. Then H is* ***almost cocommutative*** *if there exists and invertible element $R \in H \otimes H$ such that $\tau\Delta(a) = R\Delta(a)R^{-1}$.*

The important thing about almost cocommutative Hopf algebras is that their character ring is commutative. For the rest of this section, we will take H to be a semisimple almost cocommutative Hopf algebra over k algebraically closed with comultiplication Δ, counit ϵ, and antipode S. Recall that for comultiplication we use Sweedler notation so $\Delta(h) = \sum h_{(1)} \otimes h_{(2)}$.

5.1 Representation Ring of a Hopf Algebra

Thankfully all of the structure for $R(B)$ and $C(B)$ carries over for a Hopf algebra H. The antipode is the only augmentation of the bialgebra structure for the bialgebra to be a Hopf algebra, and this has no bearing on the representation or character ring. Therefore, we can define $R(H)$ and $C(H)$ as we did in the section on bialgebras, only now for H a Hopf algebra. We do add the following result.

Theorem 5.3 ([Zhu94]). *If H is a finite-dimensional semisimple Hopf algebra then the representation ring $R(H)$ is also semisimple.*

Since we are only looking at semisimple almost cocommutative Hopf algebras, we know that our representation ring $R(H)$ will always be semisimple and commutative. This preemptively informs us that we will be able to use linear characters to support our definition of the representation ring.

5.2 Character Tables

In [Wit99], Witherspoon presents two different constructions of the character table as well as a construction for the class equation. Her secondary construction is the basis for the `HopfAlgebras2` package as it fits better with the construction for the class equation.

Let $R(H)$ be the representation ring for H and $k = V_1, \ldots, V_n$ be the irreducible H-modules. Technically, these are isomorphism classes $[V_i]$ as we discussed in the section on $R(B)$. However, Witherspoon drops the brackets for readability in the following formulas (equivalent to picking a representative of the class). Witherspoon's initial construction utilizes only the irreducible H-modules of H and the characters of $R(H)$, labeled μ. The rows are indexed by the isomorphism classes $k = V_1, \ldots, V_n$, and the columns by the μ's. The entries are therefore the values of the characters on the

modules, $\mu_j(V_i)$. Of course, this construction requires that we have the representation ring at hand. `HopfAlgebras2` uses a slightly different construction that finds the representation ring for the user. We will come back to this after setting up the orthogonality relations and the center of H, $Z(H)$.

Theorem 5.4 ([NR96]). *Let μ be a character of $R(H)$ and $V \in R(H)$. Then the* ***complex conjugate*** *of μ, $\overline{\mu(V)} = \mu(V^*)$.*

Witherspoon later changes the notation to read $\mu^*(V)$ for $\mu(V^*)$.

Definition 5.5.

$$M = \bigoplus_{i=1}^{n} (V_i^* \otimes V_i)$$

Further, $M \cong H$ if we think of H as an H-module over itself via the **adjoint action** of $h \in H$ on $h' \in H$: $h' \cdot h = \sum S(h_{(1)})h'h_{(2)}$. (For G a group, M would be G acting on itself by conjugation.)

Theorem 5.6 (Column Orthogonality [Wit99]). *For all $1 \leq i, j \leq n$ one has:*

$$\sum_{l=1}^{n} \mu_i(V_l)\overline{\mu_j(V_l)} = \delta_{ij}\mu_i(M)$$

Theorem 5.7 (Row Orthogonality [Wit99]). *For all $1 \leq i, j \leq n$ one has:*

$$\sum_{l=1}^{n} \frac{\overline{\mu_l(V_i)}\mu_l(V_j)}{\mu_l(M)} = \delta_{ij}$$

Theorem 5.8 ([Wit99]). *$Z(H)$ is the* ***center of*** *H and a basis is given by*

$$z_j := \sum_{i=1}^{n} \frac{\mu_j(V_i)}{dim(V_i)} e_i$$

where e_i is the primitive central idempotent corresponding to V_i.

In the case of a group, the z_j are normalized sums of elements in the conjugacy class.

Witherspoon states that as $(\mu_j(V_i))$ is nonsingular via the given orthogonality relations, and in fact, $\mu_j = \mathrm{Tr}(z_j, \cdot)$. We can then label the columns of the character table with the z_j rather than the μ_j. Below we verify that $\mu_i = \mathrm{Tr}(z_j, \cdot)$.

$$\begin{aligned}
\mathrm{Tr}(z_j, V_i) &= \mathrm{Tr}\Big(\sum_{k=1}^{n} \frac{\mu_j(V_k)}{\dim V_k} e_k, V_k\Big) \quad \text{by definition of } z_j \\
&= \sum_{k=1}^{n} \frac{\mu_j(V_k)}{\dim V_k} \mathrm{Tr}(e_k, V_k) \quad \text{by linearity of trace} \\
&= \mu_j(V_i)
\end{aligned}$$

Thus, we now have a second way to structure the character table: rows still denoted by the V_i but columns now labeled with the z_j. The z_j are also instrumental in defining a class equation for a Hopf algebra; specifically, we can modify the z_j's into a new basis for $Z(H)$ as follows:

Definition 5.9.

$$\zeta_j := \frac{dim H}{\mu_j(M)} z_j.$$

In the case of a group, ζ_j is exactly the sum of elements in a conjugacy class. This allows us to now define a class equation for H analogous to that of the group case.

Theorem 5.10 (Class Equation [Wit99])**.** *If H is a finite dimensional semisimple almost cocommutative Hopf algebra over $\mathbb{C}$ then*

$$dim(H) = \sum_{i=1}^{n} \epsilon(\zeta_i).$$

Further, for all $1 \leq i \leq n$, $\epsilon(\zeta_i) = \frac{dim(H)}{\mu_i(M)}$ is an integer dividing $dim(H)$.

A drawback to Witherspoon's construction is the assumption of knowledge concerning the the representation ring $R(H)$. It also makes the computations for the

class equation "extraneous" as the z_j''s and ζ_j's are not part of the character table construction, even though Witherspoon's secondary definition for the μ_j makes that possible. Since the goal of the package is to streamline the computations for the representation data of a Hopf algebra, we chose to assume that the user may not know the representation ring. Our construction builds the representation ring from scratch and then utilizes that data to construct the rest of the representation theory data.

It also "linearizes" the search for and use of the idempotents and bases for $Z(H)$. That is, the package first decomposes the given H into its irreducible representations and finds the corresponding linear idempotents (a partial list of e_i's). These can then be used to find any remaining idempotents corresponding to higher dimensional representations of H, yielding a full list of e_i's. From there we have all of the information we need to move through the constructions for $R(H)$, $C(H)$, the characters of $C(H)$ (μ_j's), M, and the two different bases for $Z(H)$ (the z_j's and the ζ_j's). We then have all of the necessary pieces to construct a character table and matching class equation for H. A detailed account of how the code manages the calculations is given in Chapter 6.

5.3 Examples

This section serves to demonstrate the package's output in a nicer format than can be achieved with the print functions in `Macaulay2`. Each examples we will show their standard presentation, the character table, the orthogonality relation matrices, and the class equation. Several of the examples are titled "Kashina #", first appearing in [Kas00]. The presentation used here is from a more recent paper from Wake Forest [FKMW21]. Chapter 6 contains a demonstration of the code for H_8.

5.3.1 Group, D_4

Recall the presentation of D_4,

$$D_4 = \langle r, s | r^4 = s^2 = e, srs = r^{-1} \rangle$$

and the character table:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 \\ 2 & 0 & 0 & 0 & -2 \end{bmatrix}$$

The code easily allows for us to check the orthogonality relations, verified below.

$$\begin{bmatrix} 8 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 8 \end{bmatrix} \qquad \begin{bmatrix} 8 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 8 \end{bmatrix}$$

Additionally, there is no change to the class equation as D_4 is a group:

$$8 = 1 + 2 + 2 + 2 + 1.$$

5.3.2 H_8

Recall the presentation of H_8 for $k = \mathbb{Q}[i]$

$$H_8 = k\langle x, y, x \rangle / \langle 1 - x^2, 1 - y^2, \frac{1}{2}(1 + x + y - xy) - z^2, xy - yx, zx - yz, zy - xz \rangle$$

with the coproduct given by

$\Delta(x) = x \otimes x$

$\Delta(y) = y \otimes y$

$\Delta(z) = \frac{1}{2}(1 \otimes 1 + 1 \otimes x + y \otimes 1 - y \otimes x)(z \otimes z).$

The z_j and ζ_j elements for H_8 are

z_j	ζ_j
1	1
$(\frac{1}{4})(1+i)(1-iy-ix+xy)z$	$(\frac{1}{2})(1+i)(1-iy-ix+xy)z$
$(\frac{1}{4})(1-i)(1+iy+ix+xy)z$	$(\frac{1}{2})(1-i)(1+iy+ix+xy)z$
$\frac{1}{2}(y+x)$	$y+x$
xy	xy

The character table is as follows:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 \\ 2 & 0 & 0 & 0 & -2 \end{bmatrix}$$

The orthogonality relations are verified below:

$$\begin{bmatrix} 8 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 8 \end{bmatrix} \qquad \begin{bmatrix} 8 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 8 \end{bmatrix}$$

The class equation is

$$8 = 1 + 2 + 2 + 2 + 1.$$

5.3.3 Kashina 3

The presentation for the Hopf algebra Kashina 3 with $k = \mathbb{Q}[i]$ is

$K_3 = k\langle x, y, z, w\rangle / \langle x^2-1, y^2-1, z^2-1, xy-yx, xz-zx, yz-zy, w^2-\frac{1}{2}(1+x+y-xy)z,$

$$wx - yw, wy - xw, wz - zw\rangle$$

with coproduct given by

$\Delta(x) = x \otimes x$

$$\Delta(y) = y \otimes y$$

$$\Delta(z) = z \otimes z$$

$$\Delta(w) = \frac{1}{4}\Bigg(\sum_{b,c,\alpha,\beta\in\{0,1\}} (-1)^{b\alpha+b\beta+c\beta} y^b z^c \otimes x^\alpha y^\beta\Bigg)(w \otimes w).$$

The character table is given as follows:

$$\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -i & i & -i & i & 1 & 1 & 1 \\
1 & -1 & 1 & -i & -i & i & i & 1 & 1 & 1 \\
1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 \\
1 & -1 & 1 & i & -i & i & -i & 1 & 1 & 1 \\
1 & -1 & 1 & i & i & -i & -i & 1 & 1 & 1 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & -2 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & -2
\end{bmatrix}$$

The orthogonality relations are verified below:

$$\begin{bmatrix}
16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16
\end{bmatrix}
\quad
\begin{bmatrix}
16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16
\end{bmatrix}$$

The class equation is

$$16 = 1 + 2 + 2 + 2 + 2 + 2 + 2 + 1 + 1 + 1.$$

5.3.4 Kashina 11

The presentation for the Hopf algebra Kashina 11 with $k = \mathbb{Q}[i]$ is

$$K_{11} = k\langle x, y, z\rangle / \langle x^4 - 1, y^2 - 1, xy - yx, z^2 - x^2 y, zx - x^3 z, zy - yz\rangle$$

with coproduct given by:

$\Delta(x) = x \otimes x$

$\Delta(y) = y \otimes y$

$\Delta(z) = \frac{1}{2}(1 \otimes 1 + y \otimes 1 + 1 \otimes x^2 - y \otimes x^2)(z \otimes z).$

The character table is as follows:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -i & i & -i & i & 1 & 1 & 1 \\ 1 & -1 & 1 & -i & -i & i & i & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & i & -i & i & -i & 1 & 1 & 1 \\ 1 & -1 & 1 & i & i & -i & -i & 1 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & -2 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & -2 \end{bmatrix}$$

The orthogonality relations are verified below:

$$\begin{bmatrix} 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 \end{bmatrix} \quad \begin{bmatrix} 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 \end{bmatrix}$$

The class equation is

$$16 = 1 + 2 + 2 + 2 + 2 + 2 + 2 + 1 + 1 + 1.$$

5.3.5 Kashina 16

The presentation for the Hopf algebra Kashina 16 with $k = \mathbb{Q}[z]/\langle z^4 + 1\rangle$ is

$$K_{16} = k\langle s, t, a\rangle/\langle s^2 - 1, t^2 - 1, sa - as, ta - at, a^2 - 1, (st)^4 - a\rangle$$

with coproduct given by:

$\Delta(s) = \frac{1}{2}(1 \otimes 1 + s \otimes as + t \otimes s - t \otimes as)$

$\Delta(t) = \frac{1}{2}(t \otimes t + t \otimes at + s \otimes t - s \otimes at)$

$\Delta(a) = a \otimes a$. The character table is as follows:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 2 & 0 & 0 & -2 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 & -2 & -z^3 + z & z^3 - z \\ 2 & 0 & 0 & 0 & -2 & z^3 - z & -z^3 + z \end{bmatrix}$$

The orthogonality relations are verified below:

$$\begin{bmatrix} 16 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 \end{bmatrix} \quad \begin{bmatrix} 16 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 \end{bmatrix}$$

The class equation is

$$16 = 1 + 4 + 4 + 2 + 1 + 2 + 2.$$

Chapter 6: The Code

This chapter formally introduces the `HopfAlgebras2` package for `Macaulay2`, a software system designed by Daniel Grayson and Michael Stillman for computations in commutative algebra [GS]. We use the Emacs developing environment to interface with the `Macaulay2` language; the language itself combines the best of script language and object-oriented structure to generate a syntax mirroring common mathematical notation.

Our package builds off of two pre-existing packages: the `HopfAlgebras` package written by Dr. Moore and former graduate student Colin Martin [Mar19], and the `NCAlgebra` package that is part of `Macaulay2`. While Dr. Moore and Colin Martin's package was primarily designed for invariant actions on Hopf algebras, it also contains the functionality for manipulating tensors, computing antipodes, and generally working with bialgebras. We utilize the `NCAlgebra` to support some of the noncommutative algebra structure and compute noncommutative Gröbner bases via a `CLISP` package titled `bergman`.

The rest of this chapter is dedicated to discussing the functionality in our package, `HopfAlgebras2`. The first section highlights the methods and data structures integral to constructing the character table and housing the representation data information. The second and third sections highlight the more important background functions that support quick computations and form the foundation for some of our more complex computations.

6.1 Character Table Construction

In this section we highlight the methods and constructions used to build the character table and representation data for a given finite dimensional semisimple almost cocommutative Hopf algebra. At the end of the section we will do a demonstration of how the code works on H_8, comparing a "hand-held" run through and our final version. The major methods and data types have been broken into subsections based on their roles in the construction.

Recalling our brief discussion at the end of Chapter 5, our code's starting point is to find the one dimensional irreducible representations of a given Hopf algebra H (V_i's) and then the corresponding linear central idempotents (e_i's). We then use the sum of the e_i's to find and decompose any idempotents corresponding to higher dimensional irreducible representations of H. Once we have obtained a complete list of central idempotents, we can find the corresponding characters of H, maps from H to k in H^*. The next step is to construct the character ring, $C(H)$, which is done by computing the multiplication table and storing it as a matrix. The functionality for multiplying characters is carried over from the original `HopfAlgebras` package and were updated using a new finite dimensional algebra data type. Once we have the character ring structure we can find the characters of the character ring, the μ_j's, which are maps from $C(H)$ to k. At this point we have all of the data necessary to use Witherspoon's theorems and compute the two bases, the z_j's and ζ_j's for the center of H, the character table, and the class equation.

6.1.1 One Dimensional Objects

We start by passing the package a Hopf algebra H which is stored as a bialgebra data type (as a Hopf algebra is a bialgebra paired with the antipode and we calculate the antipode separately). There are two versions of the method titled `findOneDimlReps`,

one that takes a noncommutative ring element and one that takes a quotient ring. The latter version of the method returns a list of characters (sorted with the trivial character first) and was designed for the later work of building the characters of the character ring. However, since the one dimensional irreducible representations of H are also the linear characters of H, we can utilize the functionality twice. The former version of the method pulls the ring structure from H and abelianizes the ring as the representations of the ablieanized ring are the same as the non-ablieanizaed. We can then call the second version of the method to obtain the representations. The final step is to redefine the domain of representations to be the non-abelianized ring.

```
findOneDimlReps NCRing := B -> (
   abB := toM2Ring B;
   repList := findOneDimlReps abB;
   kk := coefficientRing B;
   apply(repList, r -> ncMap(kk,B,apply(flatten entries matrix r, e -> e_kk)))
)

findOneDimlReps QuotientRing := C -> (
   BaseRing := coefficientRing C;
   musList := apply(minimalPrimes ideal C, J -> map(BaseRing,C,sub(matrix {gens
       ambient C} % J,BaseRing)));
   trivMu := first select(musList, mu -> all(drop(flatten entries matrix mu, -(
       numgens BaseRing)), i -> i == sub(i,ZZ) and sub(i,ZZ) > 0));
   sortedMusList := {trivMu} | select(musList, mu -> mu =!= trivMu);
   sortedMusList
)
```

We now have a partial list of the V_i and need to pair the algebra homomorphisms to their corresponding linear central idempotents, done via a method called `findLinearCharacterIdem`.

```
findLinearCharacterIdem = method() --needed a more general function to find counits
    when passing 1dim
findLinearCharacterIdem (NCBialgebra, NCRingMap) := (B, phi) -> (
   H := B.ring;
   basisH := totalBasis(B);
   n := #(basisH.source);
   kk := coefficientRing H;
   matList := apply(gens H, x -> totalLeftMultMap(x,B)-(phi x)*id_(kk^n));
   intCoeff := gens ker matrix apply(matList, m -> {m});
   idem := first first entries (basisH*intCoeff);
   zeroOut := ncMap (H, H, toList (numgens H : 0_H)); -- isolate coefficient of 1
   actualIdem := (leadCoefficient (zeroOut(idem)))^(-1)* idem * (1/n); --fixing the
       coefficients
   actualIdem
)
```

We pass the function H (the `NCBialgebra` element) and a one dimensional represen-

tation (the `NCRingMap` phi). Finding a linear central idempotent means first finding an element $h \in H$ such that for all $a \in H$, $ah = \phi(a)h$. The method builds a list of matrices that stacks all of the possible systems of equations and allows us to solve them all at once. Line 8 builds this list and then the rest of the method works to impose the appropriate idempotent conditions and pull the correct element.

6.1.2 Higher Dimensional Objects

The next task is to account for any idempotents corresponding to non-linear characters of our Hopf algebra H. To do so, we attempt to decompose an `NCRingElement` e, which is the result of subtracting the sum of the linear central idempotents from one. The resulting method is `decomposeIdempotent`.

```
1  decomposeIdempotent = method()
2  decomposeIdempotent (NCBialgebra, NCRingElement, ZZ) := (H, e, numIdems) -> (
3     -- decompose e as a sum of numIdems central orthogonal idempotents
4
5     -- add variables to the base to solve for idempotents
6     basisH := totalBasis H;
7     dimH := #(flatten entries basisH);
8     B := H.ring;
9     kk := coefficientRing B;
10    Hbig := extendCoeffRing(H,numIdems*dimH,getSymbol "xxx");
11
12    -- trying to speed things up a bit
13    psi := map(coefficientRing Hbig.ring, coefficientRing H.ring);
14    phi := f -> fastBaseChange(Hbig.ring, f, psi);
15    basisHbig := ncMatrix applyTable(entries basisH, phi);
16    eBig := phi e;
17
18    bigkk := coefficientRing Hbig.ring;
19    genericMat := matrix table(numIdems, dimH, (i,j) -> bigkk_(i*dimH + j));
20    genericIdemMat := genericMat * (transpose basisHbig);
21    genericIdems := flatten entries genericIdemMat;
22    idemEqns := apply(genericIdems, eg -> eg^2 - eg);
23    sumToE := eBig - sum genericIdems;
24    orthogEqns := apply(subsets(numIdems, 2), p -> genericIdems#(p#0)*genericIdems#(p
          #1));
25    centralEqns := flatten apply(genericIdems, eg -> apply(gens B, v -> (phi v)*eg -
          eg*(phi v)));
26    allEqns := idemEqns | {sumToE} | orthogEqns | centralEqns;
27
28    eqnCoeffs := sparseCoeffs (allEqns, Monomials => flatten entries basisHbig);
29    eqs := ideal eqnCoeffs;
30    primDec := minimalPrimes eqs;
31
32    local specMat;
33    local specIdems;
34    isGood := false;
35    for P in primDec do (
36       specMat = sub(genericMat % P,kk);
```

```
37          specIdems = flatten entries (specMat * (transpose basisH));
38          specIdems = unique select(specIdems, f -> f != 0);
39          isGood = (#specIdems == numIdems);
40          if isGood then break;
41      );
42      if not isGood then (
43          isBad := true; -- :(
44          << "Unable to find nontrivial idempotents." << endl;
45          << "Try extending the field to include certain roots of unity." << endl;
46          return;
47      );
48      specIdems
49  )
```

This method takes the H, e, and the expected number of idempotents (`numIdems`). (The number of expected idempotents will be given by the user rather than an internal function. There are supporting functions in the full package to support the user finding that number.) In order to find a general idempotent, we need to enlarge the base ring to account for an appropriate number of variables (number of expected idempotents times the order of H) in lines 6 - 16. Once we have a generic idempotent built, lines 19 - 21, we impose the necessary relations in order for the set of idempotents to be a collection of orthogonal central idempotents that add to the input e (all collected in `allEqns` in line 26). Since zero is an idempotent, the method also checks that the set of idempotents found are all nonzero. If the list does include zero, the method outputs an error message directing the user to try extending the base field by a root of unity.

6.1.3 Character Ring

With a complete list of the e_i's, we can turn our attention to building the character ring, the first step of which is using the method `charOfLeftSubmodule` to turn the idempotents into the appropriate characters of H.

```
1  charOfLeftSubmodule = method()
2  charOfLeftSubmodule (NCBialgebra, NCRingElement) := (H, e) -> (
3      Hdual := dualize H;
4      basisH := flatten entries totalBasis H;
5      piSpan := sparseCoeffs(apply(basisH, b -> b*e), Monomials=> basisH);
6      prunePiSpan := prune image piSpan;
7      piBasis := gens image prunePiSpan.cache.pruningMap;
```

```
8       piBasisElts := apply(numcols piBasis, i -> first flatten entries ((ncMatrix
            {basisH})*piBasis_{i}));
9       dimOfRep := 1;
10      charValues := apply(basisH, m -> (
11          mActCoeffs := sparseCoeffs(apply(piBasisElts, f -> m*f), Monomials => basisH)
                ;
12          charValue := trace(mActCoeffs // piBasis);
13          if (m == 1_(H.ring)) then dimOfRep = round sqrt(sub(charValue,ZZ));
14          charValue // dimOfRep));
15      dualElement(Hdual, charValues)
16  )
```

The above method takes the H and an idempotent e_i as inputs and returns an irreducible character in H^*. After dualizing H, the method computes the action of e_i on the basis of H and builds an $H-dual$ module to host the information. Applying this method to the entire list of idempotents yields a basis of characters for the character ring. The next method, `characterRing`, takes the list of characters of H as a input and returns the fully structured character ring, $C(H)$.

```
1   characterRing = method(Options=>{"BaseRing"=>QQ})
2   characterRing (NCBialgebra, List, Symbol) := opts -> (H,charList,cc) -> (
3       -- H a bialgebra
4       -- charList is the complete list of characters of irreducible representations of
            H
5       -- the first element of charList should be the trivial representation
6       -- output will be the character ring over QQ
7       -- WARNING:  This function assumes the character ring is commutative.
8       BaseRing := opts#"BaseRing";
9       finDimAlgebra H;
10      chis := drop(charList,1);
11      if not all(subsets(chis,2), p -> p#0*p#1 == p#1*p#0) then
12         error "Expected a commutative character ring.";
13      charOverRing := BaseRing[cc_1..cc_(#chis)];  -- add variables here...
14      basisDeg2 := basis(2,charOverRing);
15      basisDeg0and1 := basis(0,1,charOverRing);
16      splitBasisDeg2 := apply((flatten entries basisDeg2) / support, p -> if #p == 1
            then {p#0,p#0} else p);
17      charHash := hashTable apply(#chis, i -> (charOverRing_i, chis#i));
18      charDeg2 := splitBasisDeg2 / (s -> apply(s, v -> charHash#v));
19      productsDeg2 := matrix {charDeg2 / product / (c -> transpose c.matrix)};
20      charListBasis := matrix {charList / (c -> transpose c.matrix)};
21      multCoeffs := productsDeg2 // charListBasis;
22      defIdeal := ideal (basisDeg2 - (basisDeg0and1 * sub(multCoeffs, QQ)));
23      --defIdeal = defIdeal + eqnsInCharRing;
24      charRing := charOverRing / defIdeal;
25      charRing.cache#"character list" = charList;
26      charRing
27  )
```

The method uses the characters of H to build the multiplication table for the character ring, stored in the hash table on line 17, `charHash`. Lines 18 - 22 then work to identify the relations in the multiplication table and lines 24 - 26 mod out by these

relations to construct the final ring. We can now apply the quotient ring version of `findOneDimlReps` to the character ring, acquiring the irreducible characters of $C(H)$, the μ_j's.

6.1.4 $Z(H)$ Data

At this point in the code, it is possible to compute the character table using the μ_j's and the representations of H, the V_i's. However, by computing the two bases for the center of H, the z_j's and ζ_j's, it is possible to compute the character table and the class equation using the same set of information. The following method, `getZsAndZetas`, uses the equations in Witherspoon's theorems to compute the two. Together, the two lists stand in for the conjugacy classes of a group, with the z_j's indexing the columns of the character table and the ζ_j's representing the classes for the class equation.

```
getZsAndZetas = (H, M, charRing, musList, idemList) ->
(
   fdH := finDimAlgebra H;
   n := #fdH.basis;
   m := #(charRing.cache#"character list");
   if m != #musList then error "Expected musList to be the same length as the number
       of irreducible characters.";
   sortedIdemList := sortIdempotents(H,charRing,idemList);
   trivMu := musList#0;
   charRingBasis := prepend(1_charRing, gens charRing);
   coeffRingH := coefficientRing H.ring;
   charBasis := matrix {charRing.cache#"character list" / (d -> transpose d.matrix)};
   chM := first flatten entries ((basis charRing)*sub((transpose M.matrix) //
       charBasis, charRing));
   zList := apply (musList, mu -> sum apply(m, j -> sub(mu(charRingBasis#j),
       coeffRingH) * (sub(trivMu(charRingBasis#j),coeffRingH))^(-1) * sortedIdemList
       #j));
   zetaList := apply (m, i -> n * sub((musList#i(chM))^(-1),coeffRingH) * zList#i);
   (zList, zetaList)
)
```

6.1.5 Representation Data Method and Data Type

All told, the package computes eight key pieces of representation data for a given Hopf Algebra H: the linear characters of H, the central idempotents, the character ring, the characters of the character ring, the z's, the ζ's, the character table,

and H as a module over itself under the adjoint action (M). Each of these is useful beyond the construction of the character table and combining the functionality above into one method streamlines the process for the user. We created a new data type, `RepresentationData`, which is a hash table that safely stores all of the representation theory data above once it has been computed. A user can then use the `getRepresentationData` method to run the computations by passing the method the Hopf algebra and a symbol (for method calls within `getRepresentationData` that require a general variable).

```
getRepresentationData = method(Options=>{"NumNonlinear"=>-1})
getRepresentationData(NCBialgebra,Symbol) := opts -> (H,c) -> (
   B := H.ring;
   kk := coefficientRing B;
   Hdual := dualize H;
   fdH := finDimAlgebra H;
   numNonlinear := opts#"NumNonlinear";

   oneDimCharMapList := findOneDimlReps H;
   oneDimCharList := apply(oneDimCharMapList, f -> character(Hdual,f));

   -- applying maps
   oneDimIdemList := apply (oneDimCharMapList, f -> findLinearCharacterIdem (H, f));

   e := 1_B - (sum oneDimIdemList);
   m := if numNonlinear == -1 then
           numberOfNonlinearCharacters(H, #oneDimCharList)
        else
           numNonlinear;
   if m == -1 then error "Unable to determine number of nonlinear characters.";

   newEs := decomposeIdempotent(H,e,m);
   if newEs === null then return;

   idemList := oneDimIdemList | newEs;

   charList := apply(idemList, e -> charOfLeftSubmodule(H,e));
   -- check that the dimensions of the characters are correct.
   chH := characterRing(H, charList, c, "BaseRing" => kk);

   muList := findOneDimlReps chH;
   if (#muList != #idemList) then
      error "Number of idempotents does not match number of characters.";

   M := sum apply(charList, c -> c~ * c);

   (zList,zetaList) := getZsAndZetas(H,M,chH,muList,idemList);
   -- note: charList#i(zList#j) = Tr_(charList#i)(zList#j) = mu#j(elt of char ring
       corresponding to charList#i)
   charTable := matrix table(#charList, #zList, (i,j) -> (charList#i)(zList#j));

   repData := new RepresentationData from hashTable { "NCBialgebra" => H,
           "characters" => charList,
           "idempotents" => idemList,
           "character ring" => chH,
```

```
            "mus" => muList,
            "zs" => zList,
            "zetas" => zetaList,
            "character table" => charTable,
            "chM" => M };
    repData
)
```

6.1.6 Orthogonality Relations and Class Equation

A user can verify the output of the `getRepresentationData` method by computing the orthogonality relations and the class equation. The following two blocks are the automization of the theorems presented by Witherspoon for the orthogonality relations, `checkOrthoRelations`, and the class equation, `classEquation`, respectively.

```
checkOrthoRelations = method()

checkOrthoRelations RepresentationData := repData -> (
    kk := coefficientRing repData#"NCBialgebra".ring;
    checkOrthoRelations(repData, id_kk)
)

checkOrthoRelations (RepresentationData,RingMap) := (repData,conj) -> (
    H := repData#"NCBialgebra";
    fdH := finDimAlgebra H;
    charTable := repData#"character table";
    charList := repData#"characters";
    muList := repData#"mus";
    M := repData#"chM";
    chH := repData#"character ring";
    charBasis := matrix{apply(charList, c -> transpose c.matrix)};
    chM := first flatten entries ((basis chH)*sub((transpose M.matrix) // charBasis,
        chH));
    diagWts := apply(muList, mu -> #(fdH.basis)*(mu(chM))^(-1)); -- go back and get
        fdH
    rowOrtho := charTable * (diagonalMatrix diagWts)*(conj transpose charTable);
    colOrtho := (diagonalMatrix diagWts)*(conj transpose charTable)*charTable;
    (rowOrtho, colOrtho)
)
```

The first version of the function allows a user to ignore the conjugation piece when the characters have no complex values. However, when the characters contain complex values, users must also pass the function the appropriate conjugation map from k to k in order for the function to appropriately handle any value with a complex component.

```
classEquation = method()
classEquation RepresentationData := repData -> (
    H := repData#"NCBialgebra";
    fdH := finDimAlgebra H;
    zetaList := repData#"zetas";
```

```
6       classEq := sum apply(zetaList, z -> expression H.counit z) == expression (#(fdH.
            basis));
7       classEq
8   )
```

6.1.7 Example on H_8

The following code listing details an "abridged" and "unabridged" version of how the package can be utilized from the user's end, letting $H = H_8$. The unabridged version is first (lines 1 - 38), walking through each of the steps discussed above. The abridged version spans lines 40 - 51 and is the recommended set-up for utilizing the functionality.

```
1   --- H_8, unabridged version
2   restart
3   debug loadPackage "HopfAlgebras"
4   kk = QQ[i]/ideal{i^2+1}
5   A = kk{x,y,z}
6   I = ncIdeal {x^2 - 1_A, y^2 - 1_A, x*y - y*x, z*x - y*z,
7                z*y - x*z, z^2 - (1/2)*(1_A + x + y - x*y)}
8   Igb = ncGroebnerBasis(I, InstallGB => true)
9   B = A/I
10  H = bialgebra(B,{{(x,x)},
11                   {(y,y)},
12                   {(1/2*z,z),(1/2*z,x*z),(1/2*y*z,z),(-1/2*y*z,x*z)}},
13                {1_B,1_B,1_B})
14  -- find the one dimensional characters
15  oneDimlHReps = findOneDimlReps H
16
17  -- find idempotents of these characters
18  oneDimlIdems = apply(oneDimlHReps, ch -> findLinearCharacterIdem(H,ch));
19
20  -- build the list of idempotents
21  theRest = 1_B - sum oneDimlIdems
22  idemList = oneDimlIdems | decomposeIdempotent(H,theRest,1);
23
24  -- find the character of these idempotents
25  charList = apply(idemList, e -> charOfLeftSubmodule(H,e))
26
27  -- build the character ring
28  chH = characterRing(H,charList,c)
29
30  -- find the mu_j, characters of the character ring
31  musList = findOneDimlReps chH;
32
33  -- build z_j and zeta_j
34  chM = sum apply(charList, c -> c~ * c)
35  (zList,zetaList) = getZsAndZetas(H,chM,chH,musList,idemList);
36
37  -- can now build the character table
38  charTable = matrix table(#charList, #zList, (i,j) -> (charList#i)(zList#j))
39
40  --- or: H_8 abridged
41  debug loadPackage "HopfAlgebras"
42  kk = QQ[a]/ideal{a^2+1}
```

```
A = kk{x,y,z}
I = ncIdeal {x^2 - 1_A, y^2 - 1_A, x*y - y*x, z*x - y*z, z*y - x*z, z^2 - (1/2)*(1_A
    + x + y - x*y)}
Igb = ncGroebnerBasis(I, InstallGB => true)
B = A/I
H = bialgebra(B,{{(x,x)},{(y,y)},{(1/2*z,z),(1/2*z,x*z),(1/2*y*z,z),(-1/2*y*z,x*z)}},
    {1_B,1_B,1_B})
isBialgebra H
elapsedTime repData = getRepresentationData(H,c)
classEquation repData
checkOrthoRelations repData
```

6.2 Finite Dimensional Algebras

It is worth noting that the functionality for the representation data is not the only new code in the package. All of the multiplicative computations in the `HopfAlgebras` treated objects as ring elements. Therefore, when multiplying two elements of the algebra for example, `Macaulay2` would multiply the ring elements and then try to reduce the computations using the given relations. However, the more complex the Hopf algebra structure, the harder it is for `Macaulay2` to perform the reductions. Additionally, none of the reductions were stored, forcing `Macaulay2` to re-run a single calculation multiple times. In early versions of the `HopfAlgebras2` package, running methods as stand-alone computations could take up to five minutes. Thankfully we were able to speed up the computations by taking advantage of linearity when possible in the form of a new finite dimensional algebra data type (`FinDimAlgebra`)and corresponding functionality. Using this new data type, `Macaulay2` can choose a basis for the algebra and store elements as vectors by separating an element's coefficients from its underlying ring structure. An added benefit is that we could compute all of the reductions at once by storing the multiplicative relations as matrices. The following table contains a list of the most important added and enhanced features to the package.

finDimAlgebra	fdComultMatrix	findLeftIntegral
finDimElementFromMatrix	multMatrix	findAntipode
finDimTensorProduct	totalLeftMultMap	applyRingHomomorphism
finDimAlgMult	extendCoeffRing	applyRingAntihomomorphism

6.3 Future Availability

The entire `HopfAlgebras2` package will be hosted and maintained by Dr. Moore. Anyone interested in working with the package or looking over specific details can contact him for access.

Bibliography

[Art91] Michael Artin. *Algebra*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1991.

[Dum04] Richard M. Dummit, David S. ; Foote. *Abstract Algebra*. John Wiley & Sons, Inc., 2004.

[FKMW21] Luigi Ferraro, Ellen Kirkman, W. Frank Moore, and Robert Won. Semisimple reflection hopf algebras of dimension sixteen. *Algebras and Representation Theory*, 2021.

[GS] Daniel R. Grayson and Michael E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at https://math.uiuc.edu/Macaulay2/.

[Kas00] Yevgenia Kashina. Classification of semisimple Hopf algebras of dimension 16. *J. Algebra*, 232(2):617–663, 2000.

[KP66] G. I. Kac and V. G. Paljutkin. Finite ring groups. *Trudy Moskov. Mat. Obšč.*, 15:224–261, 1966.

[Mar19] Colin Martin. A package for calculating and manipulating Hopf algebras in Macaulay2. Master's thesis, Wake Forest University, 2019.

[Mon93] Susan Montgomery. *Hopf algebras and their actions on rings*, volume 82 of *CBMS Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences, Washington, DC; by the American Mathematical Society, Providence, RI, 1993.

[NR96] Warren D. Nichols and M. Bettina Richmond. The Grothendieck group of a Hopf algebra. *J. Pure Appl. Algebra*, 106(3):297–306, 1996.

[Wit99] Sarah J. Witherspoon. The representation ring and the centre of a Hopf algebra. *Canad. J. Math.*, 51(4):881–896, 1999.

[Zhu94] Yongchang Zhu. Hopf algebras of prime dimension. *Internat. Math. Res. Notices*, (1):53–59, 1994.

Kathleen Dakota White
whitkd19@wfu.edu

EDUCATION

- **Wake Forest University** *Aug. 2019 – Present* Winston-Salem, NC
  Masters in Mathematics, Thesis Track; GPA 3.5;
  Teaching Assistantship
  Course List: Advanced Linear Algebra, Algebraic Number Theory, Algebra I-II, Commutative Algebra, Topology, Combinatorics - Generating Functions, Complex Analysis
- **University of North Carolina at Asheville** *Aug. 2014 – May 2018* Asheville, NC
  Bachelor of Arts in Applied Mathematics with a Minor in Computer Science; GPA: 3.7
  UNC Asheville Laurel Scholarship Recipient; UNC Asheville Honors Program Student

RESEARCH INTERESTS

RESEARCH EXPERIENCE

- **Character Theory of Hopf Algebras** *July 2020 – Present* Winston Salem, NC
  -
- **Honors Research** *Jan. 2017 – May. 2017* Asheville, NC
  - My independent project's final product is a full course curriculum deconstructing the witch archetype within supernatural horror film and fiction. Necessary skills for the project included source analysis, information synthesis, data extrapolation, and the ability to assess and plan student-learning outcomes. The Laurel's Scholarship funded the research and Dr. Ellen Pearson oversaw the project.
- **Boise State University REU** *June – July 2016* Boise, ID
  - Under Dr. Jyh-Haw Yeh and BSU graduate student Srisarguru Sridhar, I worked on a sub-project of the National Science Foundation funded Boise State Software Security REU. Our project focused on a new point-to-point email encryption system utilizing elliptic curves. I created, tested, and implemented a proof-of-concept JAVA GUI interface that linked the encryption system with the user's existing Gmail account. This research was funded by NSF Award No. CNS-1461133.

TEACHING EXPERIENCE

- **Wake Forest University**
  - Teaching Assistant *Aug. 2019 – Present* Winston Salem, NC
    - Math & Statistics Center (MSC) Tutoring; *Calculus I-II, Discrete Math, Linear Algebra, Abstract Algebra, Cryptography*
    - Calculus I-II Grader & Study Session Leader
  - Math & Statistics Center Lead TA
    - Coordinate TA and MSC tutor communications and work schedules
    - Plan and lead tutor training and community activities
    - Support MSC Director in technology updates and testing
    - Identify needs and capacity for tutors and TA's semester to semester
    - research and such workshops as an intensive training from the Racial Equity Collective at Asheville.

- **Johns Hopkins University, CTY Online**
  - Instructor, Linear Algebra *Aug. 2020 - Present*
- **Johns Hopkins University, Center for Talented Youth Summer Program**
  - Teaching Assistant *July – Aug. 2019* Lancaster, PA
    - Number Theory
  - Teaching Assistant *June – Aug. 2018* Lancaster, PA
    - Theory of Computation & Number Theory
  - Teaching Assistant, Number Theory *June – July 2017* Lancaster, PA
  - Teaching Assistant, Introductions to Robotics *June – July 2015* Chestertown, MD
- **AmeriCorps, University of North Carolina at Asheville**
  - AmeriCorps VISTA *Nov. 2018 – July 2019* Asheville, NC
  - As an AmeriCorps VISTA, I worked to build capacity for Marvelous Math Club. Marvelous Math Club is an after-school club for elementary age students of color, providing a safe environment focused on math and personal confidence, holistic wellbeing, and equity in education. Skills used during this time included grant writing, networking and community outreach, fundraiser planning and hosting, Math Champion/community partner training and coordination (the club's volunteers), communications and finance management, and marketing. I also volunteered as a Math Champion during club hours and supported the growth of the club's approach to equity in education through

Professional Experience Continued

- **UNC Asheville Student Employee**
  - Writing Center Consultant *Jan. 2016 – May 2018* Asheville, NC
- **Programming Language Proficiency**
  - Java
  - LateX

Formal Research Presentations

- **Committee of Public Liberal Arts Colleges Annual Meeting**
  - Student talks on coursework and impact *June 2017* Mansfield, PA
- **Southern Regional Honors Council Conference**
  - "Something Wicked This Way Comes" *March 2017* Asheville, NC
- **North Carolina Honors Association Conference**
  - "To the Point: Accessible P2P…CLOW-CKA" *Sept. 2016* Asheville, NC
  - "Sehnsucht: Art and Analyzing the Holocaust" *Sept. 2015* Asheville, NC
- **Idaho Conference on Undergraduate Research**
  - "Developing Accessible…CLOW-CKA" *July 2016* Asheville, NC

Extracurricular Activities and Awards

- **American Mathematical Society** *Sept. 2019 - Present*
- **Association for Women in Mathematics, Wake Forest Chapter** *Aug. 2019 - Present*
- **Association for Women in Mathematics** *Aug. 2019 – Aug. 2020*
- **UNC Asheville's Women's Ultimate President** *May 2015 – Dec. 2017*
- **UNC Asheville Student Honors Advisory Council Secretary** *2016 – Sept. 2017*
- **National Science Foundation REU Award Recipient** *Summer 2016*