NUMERICAL GENERATION OF SEMISIMPLE TORTILE CATEGORIES COMING FROM QUANTUM GROUPS

by

Ivelina Bobtcheva

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY
in
Mathematics

APPROVED:

Frank Quinn, Chair

Edward Green

Peter Haskell

Gail Letzter

Peter Linnell

August 29, 1996
Blacksburg, Virginia

Keywords: Quantum groups, representations, tortile category, 6j-symbols.

NUMERICAL GENERATION OF SEMISIMPLE TORTILE CATEGORIES COMING FROM QUANTUM GROUPS

by
Ivelina Bobtcheva
Committee Chairman: Frank S. Quinn
Mathematics

(ABSTRACT)

In this work we set up a general framework for exact computations of the associativity, commutativity and duality morphisms in a quite general class of tortile categories. The source of the categories we study is the work of Gelfand and Kazhdan, *Examples of tensor categories*, Invent.Math. **109** (1992), 595-617. They proved that, associated to the quantized enveloping algebra of any simple Lie group at a primitive prime root of unity, there is a semisimple monoidal braided category with finite number of simple objects. The prime p needs to be greater than the Coxeter number of the corresponding Lie algebra. We show that each of the Gelfand-Kazhdan categories has at least two subcategories which are tortile, and offer algorithms for computing the associativity, commutativity and duality morphisms in any of those categories. A careful choice of the bases of the simple objects and of the product of two such objects, make the exact computations possible. The algorithms have been implemented in Mathematica and tested for the categories $A_2, p = 5$, $A_3, p = 7$, $A_4, p = 7$, $C_2, p = 7$, and $G_2, p = 11$.

This work was supported by the Center for Mathematical Computations through NSF grant DMS-9207973.

Keywords: Quantum groups, representations, tortile category, 6j-symbols.

To my parents

Acknowledgments

In the first place, I would like to thank my advisor, Prof. Frank Quinn, whose ideas, illuminated guidance, continuous encouragement and assistance have been the main driving force behind this work. I had the high privilege to learn from Dr. Quinn mathematics, but also, from him and his wife Katherine Quinn, I was given the most valuable lesson in human tact, care, and generosity to keep and to cherish.

I would also like to thank the organizers of the Regional Geometry Institute in Park City, summer 1991, for giving me the opportunity to participate. There, I was introduced to the subject of quantum topology, and I owe my thanks to Prof. D. Freed, S. Garoufalidis and L. Jeffrey for patiently explaining to me many basic facts in this new field.

Fruitful discussions with Prof. Gail Letzter, Prof. P. Haskell, and Prof. R. Streater are gratefully acknowledged.

I express my gratitude to the Center for Transport Theory and Mathematical Physics for providing the students with a wonderful atmosphere for discussions and learning, as well as for financial support in the Summers of 1989 and 1990 through NSF grant DMS-8922002.

I would like also to thank my husband Lucio Demeio for carefully reading the manuscript and correcting numerous spelling mistakes.

Scientific work is strenuous and tense, and the decision to carry it on is never a personal one, but imposes heavy sacrifices on the whole family. Next to me, my husband and son carried quietly this burden. To them, and to all close friends with whom I shared those few years in Blacksburg, and whose encouragement and support allowed me to go on, my thanks.

This work was generously supported by the Center for Mathematical Computations through NSF grant DMS-9207973.

Contents

List of Figures

Chapter 1
Introduction

The notion of quantum groups was introduced in 1985 by Drinfeld [3] and Jimbo [12] as a deformation of the enveloping algebras of simple Lie algebras. These "quantum" enveloping algebras appeared in connection with problems in statistical mechanics, and later were shown to have interesting applications in conformal theory and topology. In particular, in 1989 Witten [42] suggested the existence of new topological invariants of closed 3-manifolds coming from topological quantum field theories (TQFT), defined on the corresponding bordism categories. An axiomatic approach to TQFT's was studied in [2] and [30]. The first general scheme for constructing the new quantum invariants, using presentation of closed three manifolds via framed links, was given by Reshetikhin and Turaev [33]. They showed that the so-called modular Hopf algebras produce 3-manifold invariants, and in particular, there exist a quotient of the quantized enveloping algebra of $sl_2(\mathbf{C})$ which has a modular structure.

A (quasi) modular structure has three main components. The first one is that there exits a subcategory of the category of representations of the Hopf algebra, whose quotient, in the terminology of [10], is a semisimple monoidal braided category with finite number of simple objects. Here, by semisimple we mean that the product of any two objects is isomorphic to a sum of simple ones. The second component is that the duality morphisms of the selfdual objects should satisfy a constraint which, using the terminology in [37], makes the above category into a tortile one. The third component of the modular structure is the invertability of the so-called S-matrix, and it won't be addressed in this work. The expectation is that the quantized enveloping algebra of any simple Lie group at any root of unity has such structure, and actually this was proved in [40] for algebras of classical type.

After the work of Reshetikhin and Turaev, few different approaches to 3-dimensional quantum invariants were developed. All of them start with the categorical data described above, but define the invariants in different ways. K.Walker [41] constructed the full TQFT on "extended" 3-manifolds, i.e. he described an algebraic construction of the state spaces associated with the extended surfaces, and the maps between these spaces corresponding to the elementary bordisms. Another approach was taken in [39] and [43],

where "state-sum" invariants of 3-manifolds were defined on the bases of a triangulation of the manifold. All these invariants are supposed to be closely related, if not the same, since they come from the same categorical data, but at this point little is known about them. In particular, even the question if they are actually new invariants, or if they can be expressed in terms of classical ones as is the case for the analogous invariants coming from finite groups ([7]), has not been answered. One reason for this is that computing the exact value of the quantum invariants is highly nontrivial. For Lens spaces and Seifert fibered manifolds, in the case of $sl_2(\mathbf{C})$, some computations and estimates were given in [6, 9, 13, 14], and a more general approach, applicable to the invariants coming from rank two Lie groups, was developed in [18].

Any exact computation of the 3-manifold invariants in [41, 39, 43] reduces to the knowledge of the associativity, commutativity and duality morphisms in the corresponding tortile category. The goal of this work is to set up a general framework for computing this categorical data. The source of the categories we study is the work of Gelfand and Kazdan [10]. They proved that, associated to the quantized enveloping algebra of any simple Lie group at a primitive prime root of unity, there is a semisimple monoidal category with finite number of simple objects. The prime p needs to be greater then the Coxeter number of the corresponding Lie algebra. We denote this category with $\mathcal{C}_p$. From the works of Lusztig [26] and Rosso [36], it follows that this category is actually braided. Here we show that there are subcategories ${\mathcal{C}_p}^0 \subset {\mathcal{C}_p}^{tor} \subset \mathcal{C}_p$ which are tortile, and offer algorithms for computing the associativity, commutativity and duality morphisms in these categories.

This work is organized as follows. In Chapter 2 we give the definition and describe the representation theory of the quantized enveloping algebra $\mathbf{U}$ of a simple Lie algebra over the ring $\mathbf{Q}(v)$ as developed in [26, 1]. This representation theory is very similar to the one of the corresponding Lie algebra in the sense that any integrable module is a direct sum of simple ones, and the simple modules have the same character formula as the ones of the Lie algebra of the same highest weight. The quantum modules, though, were shown in [15, 16, 28] to have canonical bases. Moreover, Lakshmibai [20] introduced monomial (L-S) bases of the simple integrable modules. With respect to the canonical bases, the L-S bases have a transition matrix which is upper triangular with ones on the diagonal. These are the bases used in the programs, so we describe them in detail together with the algorithm for their generation. By using the explicit description of the L-S basic element in

the lowest weight space, we deduce a relation between the duality morphism of a selfdual module and of its transpose. This relation allows us in Chapter 2 to specify the subcategory of $\mathcal{C}_p$ which is tortile. Also, the operators Ω' and Ω'' acting on any integrable module, are introduced. Those will later be related to the left and right twists.

In Chapter 2 we summarize the main results in [23, 1] which describe the representation theory of the algebra ${}_{\mathcal{R}'}\mathbf{U}$ where $\mathcal{R}' = \mathbf{Q}(v)$ with v primitive prime root of unity. Then we describe the monoidal braided structure on the category $\mathcal{C}_p$ and its tortile subcategories. In particular, it becomes clear that the main part of the computation of the commutativity and associativity morphisms in the category would consist of finding explicit bases for the spaces $hom_{\overline{\mathcal{C}_p}}(a, b \otimes c)$, where a, b, c are any simple objects in the category.

In Chapter 3 we introduce the diagrammatic notation for morphisms between tensor products of simple objects as in [37, 32], and show that the tortile property, in the way it has been defined here, is equivalent to the the equality of the left and right twists.

Chapter 4 summarizes the algorithms on which the programs are based. The first step in these algorithms is the generation of the simple modules, in particular, the L-S bases for these modules, the action of the algebra generators on them, and the duality morphisms. Once computed, this data is stored. The second step is generating bases for the spaces $hom_{\overline{\mathcal{C}_p}}(a, b \otimes c)$. These spaces are in general very big, and the numerical approach to them is nontrivial. Here we use results in [31] which describe a specific choices of bases for the spaces of homomorphisms in question. These choices have two big advantages. One, that given any set of three simple objects $\{a, b, c\}$, the bases need to be generated only for two permutations of the ordered triple (a, b, c). The bases for any other permutation can be found by applying a "diagonal" operator on the ones already generated (diagonal with respect to the decomposition into pieces described in 4.3). The other advantage of using these choices is that most of the commutativity morphisms and the associativities on the trivial module can be evaluated analytically. Actually, the only commutativity morphisms which need to be evaluated numerically are the ones commuting two objects which are the same.

In Chapter 5 we discuss some numerical aspects of the calculations and state our conclusions.

Chapter 2
The Quantized Enveloping Algebra U

2.1 Definition of U

The Hopf algebra U has been defined in general by Jimbo [12], and Drinfeld [3]. The particular form of the definition here has been introduced by Lusztig [23], and has slightly different generators (see 2.6).

2.1.1 Let n be a positive nonzero integer, X a free abelian group of rank n, and $Y = Hom_{\mathbf{Z}}(X, \mathbf{Z})$ the dual of X. We are given the following data:

(a) a basis $\Omega = \{\omega_i\}_{i=1}^n$ of X, and a linearly independent subset $\Pi = \{\alpha_i\}_{i=1}^n$ of elements in X;

(b) a basis $\{h_i\}_{i=1}^n$ of Y;

(c) a symmetric bilinear form on X x, x' maps to $x.x'$ with values in $\mathbf{Q}$, such that the following conditions are satisfied:

$$\begin{aligned} \alpha_i.\alpha_i &= 2\mathbf{Z}_{>0} \quad \text{for all} \quad i \leq n \\ \alpha_i.\alpha_j &\leq 0 \quad \text{for all} \quad i, j \leq n \text{ and } i \neq j \\ \langle h_i, \lambda \rangle &= 2\,\frac{\alpha_i.\lambda}{\alpha_i.\alpha_i} \quad \text{for all} \quad i \leq n \text{ and } \lambda \in X \\ \langle h_i, \omega_i \rangle &= \delta_{ij} \quad \text{for all} \quad i, j \leq n \end{aligned}$$

where $\langle\ ,\ \rangle : Y \times X \to \mathbf{Z}$ is the dual pairing.

Using the standard Lie algebra terminology, we call X the weight space, Π the set of simple roots, and Ω the set of fundamental weights. The '.' form is the Killing form on X and the $n \times n$ matrix with i, j element $\langle h_j, \alpha_i \rangle$ is the Cartan matrix $\mathbf{C}$. Let X^+ be the submonoid of X consisting of all linear combinations of the elements in Π with non-negative integer coefficients. Then we introduce a partial order on X in the following way: $\lambda \leq \mu$ if and

only if $\mu \in \lambda + X^+$. Also for any $\nu = \sum_{i=1}^n \nu_i \alpha_i \in X$ we set the level of ν to be $lev(\nu) = \sum_{i=1}^n \nu_i$.

In order to make the expressions more compact in what follows we will write $i.j$ instead of $\alpha_i.\alpha_j$, and $\langle i,j\rangle$ instead of $\langle h_i, \alpha_j\rangle$.

2.1.2 We define $\mathbf{U}$ to be the associative $\mathbf{Q}(v)$-algebra with generators E_i, F_i, for $i \leq n$, K_μ for $\mu \in Y$, and relations:

$$
\begin{array}{lll}
\text{(a)} & K_0 = 1 & \\
 & K_\mu K_{\mu'} = K_{\mu+\mu'} & \text{for all } \mu, \mu' \in Y \\
\text{(b)} & K_\mu E_i = v^{\langle \mu, \alpha_i\rangle} E_i K_\mu & \\
 & K_\mu F_i = v^{-\langle \mu, \alpha_i\rangle} F_i K_\mu & \text{for all } \mu \in Y \text{ and } i \leq n \\
\text{(c)} & E_i F_j - F_j E_i = \delta_{ij} \frac{\tilde{K}_i - \tilde{K}_{-i}}{v_i - v_i^{-1}} & \text{for all } i, j \leq n \\
\text{(d)} & \sum_{r=0}^c (-1)^r E_i^{(r)} E_j E_i^{(c-r)} = 0 & \\
 & \sum_{r=0}^c (-1)^r F_i^{(r)} F_j F_i^{(c-r)} = 0 & \text{for } i \neq j \text{ and } c = 1 - \langle h_i, \alpha_j\rangle
\end{array}
$$

Here we have used the following notations:

$$
\begin{aligned}
& v_\mu = v^{\tilde{\mu}}, \quad v_i \equiv v^{\frac{i.i}{2}}, \quad \tilde{K}_\mu = K_{\tilde{\mu}}, \quad \tilde{K}_{\pm i} \equiv \tilde{K}_{\pm\alpha_i}, \\
& [k]_i = \frac{v_i^k - v_i^{-k}}{v_i - v_i^{-1}}, \quad [k]_i^! = \prod_{m=1}^k [m]_i \\
& E_i^{(k)} = \frac{E_i^k}{[k]_i^!}, \quad F_i^{(k)} = \frac{F_i^k}{[k]_i^!}.
\end{aligned}
$$

where for any $\mu = \sum_{i=1}^n \mu_i \alpha_i$, $\mu_i \in \mathbf{Z}$, $\tilde{\mu} = \sum_{i=1}^n (i.i/2)\mu_i h_i$.

2.1.3 There is unique algebra homomorphism (*comultiplication*) $\triangle : \mathbf{U} \to \mathbf{U} \otimes \mathbf{U}$, such that

$$
\begin{aligned}
\triangle(E_i) &= E_i \otimes 1 + \tilde{K}_i \otimes E_i \\
\triangle(F_i) &= F_i \otimes \tilde{K}_{-i} + 1 \otimes F_i \qquad \text{for } i \leq n \\
\triangle(K_\mu) &= K_\mu \otimes K_\mu \quad \text{for any } \mu \in Y.
\end{aligned}
$$

Then $\triangle$ has the coassociativity property, i.e. $(id \otimes \triangle) \circ \triangle = (\triangle \otimes id) \circ \triangle$. One can define a Hopf-algebra structure $(\triangle, S, e)$ on $\mathbf{U}$ with counit the unique algebra homomorphism $e : \mathbf{U} \to \mathbf{Q}(v)$, such that

$$
e(E_i) = e(F_i) = 0 \quad \text{and} \quad e(K_\mu) = 1,
$$

for $i \leq n$, $\mu \in Y$, and an antipode the unique algebra anti-homomorphism $S : \mathbf{U} \to \mathbf{U}$ such that

$$S(E_i) = -\tilde{K}_{-i}E_i, \qquad S(F_i) = -F_i\tilde{K}_i, \qquad S(K_\mu) = K_\mu,$$

for $i \leq n$, $\mu \in Y$. Then if we denote by m the multiplication in $\mathbf{U}$, the following identities hold:

$$\begin{aligned} &m(1 \otimes S)\triangle(x) = m(S \otimes 1)\triangle(x) = e(x)\,1, \\ &eS = e, \\ &(S \otimes S)\triangle(x) = \triangle^t(S(x)), \end{aligned}$$

for any $x \in \mathbf{U}$. Here $\triangle^t$ denotes the composition of maps $\mathbf{U} \xrightarrow{\triangle} \mathbf{U} \otimes \mathbf{U} \xrightarrow{x\otimes y \to y\otimes x} \mathbf{U} \otimes \mathbf{U}$.

2.1.4 Let $^- : \mathbf{Q}(v) \to \mathbf{Q}(v)$ be the $\mathbf{Q}$-algebra involution such that $\overline{v^r} = v^{-r}$ for all r. Then there is a unique homomorphism of $\mathbf{Q}(v)$-algebras $^- : \mathbf{U} \to \mathbf{U}$ such that

$$\overline{E_i} = E_i,\ \overline{F_i} = F_i,\ \overline{K_\mu} = K_{-\mu} \text{ and } \overline{rx} = \overline{r}\ \overline{x} \text{ for all } r \in \mathbf{Q}(v),\ x \in \mathbf{U}.$$

2.1.5 Let $\mathbf{U}^+$ ($\mathbf{U}^-$) be the subalgebra of $\mathbf{U}$ generated by the elements E_i (resp. F_i) for $i \leq n$, and $\mathbf{U}^0$ be the subalgebra of $\mathbf{U}$ generated by K_μ, $\mu \in Y$. Let also $\mathcal{A}$ be the ring $\mathbf{Z}[v, v^{-1}]$, ${}_{\mathcal{A}}\mathbf{U}^+({}_{\mathcal{A}}\mathbf{U}^-)$ the $\mathcal{A}$-subalgebra of $\mathbf{U}$ generated by the elements $E_i^{(k)}$ (resp.$F_i^{(k)}$), $k \geq 0$, $i \leq n$, and ${}_{\mathcal{A}}U^0$ the $\mathcal{A}$-subalgebra of $\mathbf{U}$ generated by $\tilde{K}_{\pm i}$ and

$$\begin{bmatrix} \tilde{K}_i; c \\ t \end{bmatrix} = \prod_{s=1}^{t} \frac{\tilde{K}_i v_i^{(c-s+1)} - \tilde{K}_{-i} v_i^{-(c-s+1)}}{v_i^s - v_i^{-s}}.$$

2.1.6 Define ${}_{\mathcal{A}}\mathbf{U}$ to be the $\mathcal{A}$-subalgebra of $\mathbf{U}$ generated by $E_i^{(k)}, F_i^{(k)}$, and $\tilde{K}_{\pm i}$ for any $k \geq 0$, and $i \leq n$. Then the following relations are satisfied in ${}_{\mathcal{A}}\mathbf{U}$([25])

(a) For any $i \neq j$, $i, j \leq n$

$$E_i^{(k)} F_j^{(l)} = F_j^{(l)} E_i^{(k)} \text{ for } i \neq j,$$
$$E_i^{(k)} F_i^{(l)} = \sum_{t:0\leq t\leq min(k,l)} F_i^{(l-t)} \begin{bmatrix} \tilde{K}_i; 2t-k-l \\ t \end{bmatrix} E_i^{(k-t)},$$
$$F_i^{(k)} E_i^{(l)} = \sum_{t:0\leq t\leq min(k,l)} E_i^{(l-t)} \begin{bmatrix} \tilde{K}_i; k+l-t-1 \\ t \end{bmatrix} E_i^{(k-t)}.$$

(b) For any $i, j \leq n$

$$\begin{bmatrix} \tilde{K}_i; c \\ t \end{bmatrix} E_j^{(k)} = E_j^{(k)} \begin{bmatrix} \tilde{K}_i; c + k\langle i,j\rangle \\ t \end{bmatrix},$$
$$\begin{bmatrix} \tilde{K}_i; c \\ t \end{bmatrix} F_j^{(k)} = F_j^{(k)} \begin{bmatrix} \tilde{K}_i; c - k\langle i,j\rangle \\ t \end{bmatrix}.$$

Note also that

$$\triangle(E_i^{(r)}) = \sum_{r'+r''=r} v_i^{r'r''} E_i^{(r')} \tilde{K}_i{}^{r''} \otimes E_i^{(r'')},$$
$$\triangle(F_i^{(r)}) = \sum_{r'+r''=r} v_i^{r'r''} F_i^{(r')} \otimes \tilde{K}_{-i}{}^{r''} F_i^{(r'')},$$
$$S(E_i^{(r)}) = (-1)^r v_i^{r^2-r} \tilde{K}_{-i}{}^{r} E_i^{(r)},$$
$$S(F_i^{(r)}) = (-1)^r v_i^{-r^2+r} F_i^{(r)} \tilde{K}_i{}^{r}.$$

Hence the comutiplication and the antipode leave ${}_{\mathcal{A}}\mathbf{U}$ invariant, and in this way define a Hopf algebra structure on ${}_{\mathcal{A}}\mathbf{U}$.

2.2 The algebras $'\mathbf{f}$ and $\mathbf{f}$

The algebras $\mathbf{U}^+$ and $\mathbf{U}^-$ are actually isomorphic to each other, and it is useful to study them as one object. This is why the algebra $\mathbf{f}$ is introduced (isomorphic to $\mathbf{U}^+$ and $\mathbf{U}^-$) as a quotient of the free associative algebra $'\mathbf{f}$. Here we will list some properties and operations on $\mathbf{f}$ (see [26, Chapter 1]). The nondegenerate form on $\mathbf{f}$ described in 2.2.3 is the one introduced by the double construction of Drinfeld ([5]).

2.2.1 Given the data in 2.1.1, we denote by $'\mathbf{f}$ the free associative $\mathbf{Q}(v)$ -algebra with generators θ_i, $i \leq n$. Then there is a direct sum decomposition $'\mathbf{f} = \oplus_{\nu \in X^+} {}'\mathbf{f}_\nu$, such that for any $\nu = \sum_{j=1}^n \nu_j \alpha_j$, the monomial $\theta_{i_1} \dots \theta_{i_r}$ is in $'\mathbf{f}_\nu$ if and only if j appears ν_j times in the sequence $i_1, \dots, i_r$ An element x in $'\mathbf{f}$ is said to be *homogeneous* if it belongs to $'\mathbf{f}_\nu$ for some ν. Then following Lusztig we set $|x| = \nu$.

2.2.2 The tensor product $'\mathbf{f} \otimes {}'\mathbf{f}$ can be regarded as $\mathbf{Q}(v)$ -algebra with multiplication

$$(x_1 \otimes x_2)(x_1' \otimes x_2') = v^{|x_2| \cdot |x_1'|} x_1 x_1' \otimes x_2 x_2'.$$

Then there is unique algebra homomorphism $r : {}'\mathbf{f} \to {}'\mathbf{f} \otimes {}'\mathbf{f}$ such that $r(\theta_i) = \theta_i \otimes 1 + 1 \otimes \theta_i$ for all $i \leq n$.

2.2.3 There is a unique bilinear inner product $(\ ,\ )$ on $'\mathbf{f}$ with values in $\mathbf{Q}(v)$such that $(1,1) = 1$, and

(a) $(\theta_i, \theta_j) = \delta_{ij}(1 - v_i^{-2})^{-1}$, for $i, j \leq n$;

(b) $(x, y'y'') = (r(x), y' \otimes y'')$, for any $x, y', y'' \in {}'\mathbf{f}$;

(c) $(x'x'', y) = (x' \otimes x'', r(y))$, for any $x', x'', y \in {}'\mathbf{f}$,

where the bilinear form on $'\mathbf{f} \otimes' \mathbf{f}$ given by $x' \otimes x'', y' \otimes y'' \to (x', y')(x'', y'')$ is denoted again by $(,)$.

2.2.4 The radical $\mathcal{I}$ of the form $(\ ,\ )$ is a two-sided ideal of $'\mathbf{f}$, and we define $\mathbf{f} = {}'\mathbf{f}/\mathcal{I}$. Then there is a direct sum decomposition $\mathbf{f} = \oplus_\nu \mathbf{f}_\nu$, where $\mathbf{f}_\nu$ is the image of $'\mathbf{f}_\nu$ in $\mathbf{f}_\nu$, and the form $(\ ,\ )$ is nondegenerate on each summand $\mathbf{f}_\nu$ of $\mathbf{f}$. Also, if we regard $\mathbf{f} \otimes \mathbf{f}$ as an algebra by the rule 2.2.2. Then r induces an algebra homomorphism $\mathbf{f} \to \mathbf{f} \otimes \mathbf{f}$ which we denote again by r.

2.2.5 Denote by $^- : \mathbf{f} \to \mathbf{f}$ the unique $\mathbf{Q}$-algebra involution such that $\overline{q\theta_i} = \overline{q}\theta_i$ for any $i \leq n$, and $q \in \mathbf{Q}(v)$.

We also define ${}_{\mathcal{A}}\mathbf{f}$ to be the $\mathcal{A}$-subalgebra of $\mathbf{f}$ generated by $\theta_i^{(r)}$, for any $i \leq n$, and $r \geq 0$.

2.2.6 It can be verified that the generators θ_i of $\mathbf{f}$ satisfy the quantum Serre relations:

$$\sum_{k=0}^{c}(-1)^k\theta_i^{(k)}\theta_j\theta_i^{(c-k)} = 0,$$

where $c = 1 - \langle i, j\rangle$, and $i \neq j$. Then from 1.1.2 follows that there are homomorphisms $\mathbf{f} \xrightarrow{x\to x^+} \mathbf{U}^+$, ${\theta_i}^+ = E_i$ and $\mathbf{f} \xrightarrow{x\to x^-} \mathbf{U}^-$, ${\theta_i}^- = F_i$ respectively. A stronger result of Lusztig [26, 33.1.1] shows that these are actually isomorphisms. The images of $\mathbf{f}_\nu$ under these isomorphisms we denote with $\mathbf{U}_\nu^\pm$. Furthermore, the action of r on $\mathbf{f}$ can be connected with the comultiplication on $\mathbf{U}$. The exact statement is the following. For any $x \in X$

$$\begin{aligned}\triangle(x^+) &= \sum x_1^+\tilde{K}_{|x_2|} \otimes x_2^+, \text{ and}\\ \triangle(x^-) &= \sum x_3^- \otimes \tilde{K}_{-|x_3|}x_4^-,\end{aligned}$$

where $r(x) = \sum x_1 \otimes x_2$, and $\overline{r(\overline{x})} = \sum x_3 \otimes x_4$. We set $\overline{r}(x) \equiv \overline{r(\overline{x})}$.

2.2.7 By induction on ν one can verify the following identities for $x \in \mathbf{f}_\nu$

$$\begin{aligned}S(x^+) &= (-1)^{lev(\nu)}v^{c(\nu)}\tilde{K}_{-\nu}\sigma(x)^+, \text{ and}\\ S(x^-) &= (-1)^{lev(\nu)}v^{-c(\nu)}\sigma(x)^-\tilde{K}_\nu,\end{aligned}$$

where $c(\nu) = \nu.\nu/2 - \sum_{i=1}^n \nu_i(i.i/2)$, and $\sigma : \mathbf{f} \to \mathbf{f}^{opp}$ is the algebra homomorphism which takes θ_i to θ_i.

2.2.8 The following result of Lusztig describes what is called the triangular decomposition of $\mathbf{U}$ in terms of $\mathbf{f}$, and $\mathbf{U}^0$ ([24, 25]).

(a) The $\mathbf{Q}(v)$-linear map $\mathbf{f} \otimes \mathbf{U}^0 \otimes \mathbf{f} \to \mathbf{U}$ given by $x \otimes K_\mu \otimes y \to x^-K_\mu y^+$ is an isomorphism.

(b) The $\mathbf{Q}(v)$-linear map $\mathbf{f} \otimes \mathbf{U}^0 \otimes \mathbf{f} \to \mathbf{U}$ given by $x \otimes K_\mu \otimes y \to x^+K_\mu y^-$ is an isomorphism.

2.3 Integrable U-modules

2.3.1 An $\mathbf{U}$-module M is said to be *integrable* if it is finite dimensional, and there is a direct sum decomposition $M = \oplus_{\lambda \in X} M^\lambda$ as a $\mathbf{Q}(v)$-vector space such that for any $\mu \in Y$, $\lambda \in X$, and $m \in M^\lambda$

$$K_\mu m = v^{\langle \mu, \lambda \rangle} m.$$

M^λ are called the weight spaces of M, and their dimensions are called multiplicities. A homomorphism between modules is a $\mathbf{U}$-linear map, and hence preserves the weight space decomposition. We will denote the category of integrable $\mathbf{U}$-modules with $\mathcal{C}$.

2.3.2 Let $M \in \mathcal{C}$. Note that from the finite dimensionality of M, and the fact that $x^+ M^\lambda \subset M^{\lambda + |x|}$ and $x^- M^\lambda \subset M^{\lambda - |x|}$, follows that there exists $N \geq 0$ and for any $m \in M$ and any $x \in \mathbf{f}$ with $lev(|x|) \geq N$, $x^+ m = 0$, and $x^- m = 0$.

2.3.3 The simplest example of integrable $\mathbf{U}$-module is the following. Let $M = \mathbf{Q}(v)$. Then for any $u \in \mathbf{U}$ and $m \in M$ define $um = e(u)m$. We will call this module the trivial module (representation) of $\mathbf{U}$.

2.3.4 The tensor product of two $\mathbf{U}$-modules M', and M'' is naturally a $\mathbf{U} \otimes \mathbf{U}$ -module with $(u' \otimes u'')(m' \otimes m'') = u'm' \otimes u''m''$, for any $u', u'' \in \mathbf{U}$, $m' \in M'$, and $m'' \in M''$. We restrict this to a $\mathbf{U}$-module via the comultiplication $\triangle : \mathbf{U} \to \mathbf{U} \otimes \mathbf{U}$, i.e.

$$u(m' \otimes m'') = \triangle(u)(m' \otimes m'').$$

If M' and M'' are in $\mathcal{C}$, their tensor product is again in $\mathcal{C}$, and

$$(M' \otimes M'')^\lambda = \sum_{\lambda' + \lambda'' = \lambda} M^{\lambda'} \otimes M''^{\lambda''}.$$

The coassociativity property of $\triangle$ insures the associativity of the tensor product above, i.e. the standard map $(M' \otimes M'') \otimes M''' \to M' \otimes (M'' \otimes M''')$, given by $(m' \otimes m'') \otimes m''' \to m' \otimes (m'' \otimes m''')$ is an algebra homomorphism.

2.3.5 Given a $\mathbf{U}$-module M we define its dual M^* to be the $\mathbf{U}$-module with vector space $M^* = Hom(M, \mathbf{Q}(v))$, and the following $\mathbf{U}$-structure: for any $u \in \mathbf{U}, \quad y \in M^*$, and $m \in M$

$$u(y)(m) = y(S(u)x).$$

Since the antipode is an algebra anti-homomorphism, this is well defined.

If M is an element in $\mathcal{C}$, M^* is again in $\mathcal{C}$, and

$$(M^*)^\lambda = Hom(M^{-\lambda}, \mathbf{Q}(v)).$$

To see this, note that for any $\mu \in Y, \quad g \in Hom(M^{-\lambda}, \mathbf{Q}(v))$, and $m \in M^{-\lambda}$

$$(K_\mu g)(m) = g(S(K_\mu)m) = g(K_{-\mu}m) = v^{\langle \mu, \lambda \rangle} g(m).$$

2.3.6 Any integrable $\mathbf{U}$-module is isomorphic to a direct sum of simple $\mathbf{U}$-modules (see [26, Chapter 6]). Let $\mathcal{W} = \{\lambda = \sum_{i=1}^n \lambda_i \omega_i \in X \mid \lambda_i \in \mathbf{N}, \quad i \leq n\}$. Then the set of simple $\mathbf{U}$-modules corresponds to the elements in $\mathcal{W}$ in the following way. For any $\lambda \in \mathcal{W}$, set $\mathcal{J} = \sum_{i=1}^n \mathbf{U}E_i + \sum_{\mu \in Y} \mathbf{U}(K_\mu - v^{\langle \mu, \lambda \rangle}1)$. Then from 2.2.8 it follows that the map $\mathbf{f} \to \mathbf{U}/\mathcal{J}$, $x \to x^- + \mathcal{J}$ is an isomorphism of $\mathbf{Q}(v)$ spaces. Via this isomorphism $\mathbf{f}$ becomes a $\mathbf{U}$-module, called Verma module. The Verma module is infinite dimensional, but has a submodule - the left ideal of $\mathbf{f}$ generated by $\theta_i^{\langle h_i, \lambda \rangle + 1}$ for various i. If we denote this ideal with $\mathcal{J}'$, then the quotient $L_\lambda = \mathbf{f}/\mathcal{J}'$ with the $\mathbf{U}$-structure described above, is a simple integrable $\mathbf{U}$-module. Moreover, these are all of the simple integrable $\mathbf{U}$-modules. From the definition we can see that the subspace ${L_\lambda}^\lambda$ is equal to $\{x \in L_\lambda \mid E_i(x) = 0 \text{ for any } i \leq n\}$, and it is one-dimensional with basis the image of 1 in L_λ (whenever confusion is unlikely we will use the same notation for the elements of $\mathbf{f}$ and their image in L_λ) . ${L_\lambda}^\lambda$ is called the highest weight space, and λ the highest weight of L_λ. In general, for any homogeneous y in $\mathbf{f}$,

$$K_\mu(y) = v^{\langle \mu, \lambda - |y| \rangle} y,$$

i.e. ${L_\lambda}^{\lambda'}$ is equal to $\mathbf{f}_{\lambda - \lambda'}$. Also, observe that for any y in $\mathbf{f}$, $F_i(y) = \theta_i y$, and therefore L_λ is contained in $\mathbf{U}^- 1$.

Obviously the automorphism group of any quotient of the Verma module is isomorphic to $\mathbf{Q}(v)$.

2.4 Weyl Group and its applications

Here we list some definitions and properties of the Weyl group (see [8, 21] and the references therein).

2.4.1 Given the data in 1.1.1, define the Weyl group $\mathbf{W}$ to be the group with generators s_i, $i \leq n$, and relations:

$$(s_i s_j s_i)^{h(i,j)} = (s_j s_i s_j)^{h(i,j)}, \text{ and } s_i^2 = 1$$

where $i \neq j$, and $cos^2 \frac{\pi}{h(i,j)} = \frac{(i.j)^2}{(i.i)(j.j)}$. Then $\mathbf{W}$ is a finite group, and there are homomorphisms $\mathbf{W} \to Aut(X)$ and $\mathbf{W} \to Aut(Y)$, such that

$$s_i(\mu) = \mu - \langle \mu, \alpha_i \rangle h_i, \text{ for any } \mu \in Y, i \leq n,$$
$$s_i(\lambda) = \lambda - \langle h_i, \lambda \rangle \alpha_i, \text{ for any } \lambda \in X, i \leq n$$

i.e. s_i is a reflection of X in a plane perpendicular to the i-th simple root (with respect to the Killing form).

We have $\lambda.\lambda = s_i(\lambda).s_i(\lambda)$ for any $i \leq n$ and $\lambda \in X$.

2.4.2 If $w \in \mathbf{W}$, let $l(w)$ be the smallest integer r such that $w = s_{i_1} \dots s_{i_r}$ for some sequence $i_1, \dots, i_r$, where $i_j \leq n$. Then $l(w)$ is called the length of w, and $s_{i_1} \dots s_{i_r}$ a reduced expression of w.

Given a reduced expression $s_{i_1} \dots s_{i_r}$ of w, define $A_w = \{w' \in \mathbf{W} | w' = s_{i_{j_1}} \dots s_{i_{j_k}}, \quad 1 \leq j_1 < j_2 \dots < j_k \leq r\}$. Then A_w depends only on w, and not on the choice of reduced expression, and we define a partial order on $\mathbf{W}$ as follows: $w' \leq w$ if and only if $w' \in A_w$. This is called the Bruhat order on $\mathbf{W}$. Given $w' \leq w$, we define the *canonical reduced expression of* w' with respect to to a given reduced expression $s_{i_1} \dots s_{i_r}$ of w to be the one which is lexicographically the biggest [1].

2.4.3 $\mathbf{W}$ has a unique element w^0 of largest length. Set $l^0 = l(w^0)$. Then we have :

(a) $w \leq w^0$ for any $w \in \mathbf{W}$;

[1] The lexicographical ordering on the subsequences $\underline{j} = \{j_1, j_2, \dots, j_s\}$ of $\{1, 2, \dots, r\}$ is the one where $\underline{j} \leq \underline{k}$ if there exists a $t \leq s$ such that $j_p = k_p$, $p < t$, and $j_t < k_t$.

(b) There exists a permutation $\{1,\dots,n\} \xrightarrow{P_0} \{1,\dots,n\}$ such that $w^0(\alpha_i) = -\alpha_{P_0(i)}$;

(c) Let $w^0 = s_{i_1}\dots s_{i_{l^0}}$ be a reduced expression for w^0 and

$$\mathbf{R}^+ = \{\alpha_{i_1}, s_{i_1}(\alpha_{i_2}),\dots, s_{i_1}\dots s_{i_{l^0-1}}(\alpha_{i_{l^0}})\}.$$

Then the set $\mathbf{R}^+$ doesn't depend on the choice of reduced expression for w^0, and each element in $\mathbf{R}^+$ is a linear combination of elements in $\mathbf{\Pi}$ with coefficients in $\mathbf{N}$. We call $\mathbf{R}^+$ the set of positive roots. (This definition, of course, coincides with the usual one for Lie algebra with the same Cartan matrix). Given an element $\beta = w(\alpha_i) \in \mathbf{R}^+$, set $s_\beta = ws_iw^{-1}$ to be the reflection of X in a plane perpendicular to β.

2.4.4 The set $\{h_\beta \in Y \mid \pm\beta \in \mathbf{R}^+ \text{ and} \langle h_\beta, \lambda\rangle = 2\frac{\beta.\lambda}{\beta.\beta}\}$ is called the set of coroots. Note that $h_{\alpha_i} = h_i$. The following statement follows from [21, Proposition1.4]. Let $s_{i_1}\dots s_{i_k}$ be a reduced expression for an element in $\mathbf{W}$. Then $s_{i_1}\dots s_{i_k}s_{i_{k+1}}$ is reduced if and only if $s_{i_1}\dots s_{i_k}(h_{i_{k+1}}) = \sum_{i=1}^n a_ih_i$, $a_i \in \mathbf{N}$ for any $i \le n$ (i.e. it is *positive coroot*). Otherwise, $s_{i_1}\dots s_{i_k}(h_{i_{k+1}}) = \sum_{i=1}^n a_ih_i$, $-a_i \in \mathbf{N}$ for any $i \le n$ (*negative coroot*).

2.4.5 Let $\lambda \in X$, and $\mathbf{W}_\lambda$ be the stabilizer of λ in $\mathbf{W}$. Then define $\tilde{\mathbf{W}}_\lambda = \mathbf{W}/\mathbf{W}_\lambda$ to be the group of left cosets in $\mathbf{W}$. Given $\tilde{w} \in \tilde{\mathbf{W}}_\lambda$, we define the length $l(\tilde{w})$ of $\tilde{w}$ to be the minimum of $l(w)$ over $w \in \tilde{w}$. Since $\mathbf{W}$ is a finite group, this is well defined. Then a reduced expression for $\tilde{w}$ would be a reduced expression for $w \in \tilde{w}$ of minimal length. Now, we can extend the definition of Bruhat order to $\tilde{\mathbf{W}}_\lambda$ as follows: Define $A_{\tilde{w}}$ to be the image of A_w in $\tilde{\mathbf{W}}_\lambda$, where $w \in \tilde{w}$ is of minimal length. Then $A_{\tilde{w}}$ doesn't depend on the choice of the element of minimal length, and we define $\tilde{w}' \le \tilde{w}$ if and only if $\tilde{w}' \in A_{\tilde{w}}$.

2.4.6 Let M be an integrable $\mathbf{U}$-module. Then for any $i \le n$ there exists a linear operator $T_i : M \to M$ which defines an isomorphism of the λ-weight space of M onto the $s_i(\lambda)$-weight space of M (see [26]).

There are two observations to be made in connection with the statement above:

Corolary 2.4.7 *For any* $i \le n$ $F_i{L_\lambda}^{w^0(\lambda)} = 0$. *Hence* $L_\lambda = \mathbf{U}^+{L_\lambda}^{w^0(\lambda)}$.

The first assertion follows from the fact that the multiplicities of ${L_\lambda}^{w^0(\lambda)-\alpha_i}$ are zero for any i. Then from theorem 2.2.8 and the fact that L_λ is simple, follows that $L_\lambda = \mathbf{U}^+ {L_\lambda}^{w^0(\lambda)}$.

Corolary 2.4.8 *${L_\lambda}^*$ is isomorphic to $L_{-w^0(\lambda)}$.*

First we want to show that ${L_\lambda}^*$ is simple. Let $\mathcal{I}$ be an ideal in ${L_\lambda}^*$, and μ be maximal such that $\mathcal{I} \cap (L_\lambda^*)^\mu$ is nonzero. From the maximality condition follows that the intersection above is in the kernel of E_i for any $i \leq n$. By duality the map $\ominus E_i : \oplus {L_\lambda}^{-\mu-\alpha_i} \to {L_\lambda}^{-\mu}$ is not onto. Then from (i) follows that $\mu = -w^0(\lambda)$. From 2.4.6 $({L_\lambda}^*)^{w^0(\lambda)}$ is one dimensional. Let y be a nonzero vector in this weight space. Then $\mathcal{I} = U^- y$. To conclude that $\mathcal{I} = {L_\lambda}^*$ it is enough to show that for any $\mu < w^0(\lambda)$, the map $\oplus F_i : \ominus({L_\lambda}^*)^{\mu+\alpha_i} \to ({L_\lambda}^*)^\mu$ is onto. This follows from the same argument as above. If it wasn't, the intersection of the kernels of $F_i : {L_\lambda}^{-\mu} \to {L_\lambda}^{-\mu-\alpha_i}$ would be nonempty. But since the module L_λ is simple this is true only when $\mu = w^0(\lambda)$ and we have a contradiction. Hence the dual representation is simple. Then the dual representation must be isomorphic to $L_{-w^0(\lambda)}$ since all simple object are of this form (see 2.3.6).

2.5 Lakshmibai-Seshadri bases of simple integrable representations

2.5.1 From 2.4.6 one sees that the simple modules L_λ have structure very similar to the one of the simple modules of the corresponding Lie algebra with the same Cartan matrix. In particular, all the weights in the orbit of λ under the action of $\tilde{\mathbf{W}}_\lambda$ are of multiplicity one. Actually, it can be shown ([27, 35]) that the character formula of L_λ is the same as the one of the simple modules of the Lie algebra of highest weight λ. The modules of the quantum algebra though, are richer, in the sense that the lattices ${}_{\mathcal{A}}L_\lambda$ were shown to have canonical bases ([28, 15, 16]). Unfortunately, the nice properties of these bases doesn't preserve after changing the ring to $\mathbf{Q}(v)$ with v – root of unity (the case we are interested in). Hence, we found more convenient to work with the bases introduced by Lakshmibai ([19, 20]) which, following [20], we call Lakshmibai-Seshadri (L-S) bases. The elements of the L-S bases are monomials in $\mathbf{f}$, and this leads to considerable simplification of the computations, and speed up in finding the injections (see Chapter

3). The transition matrix of the L-S basis of L_λ to the canonical basis of Kashiwara-Lusztig is upper triangular with ones on the diagonal (for a suitable indexing). There seems to be a difference in the basis we define here and the one in [20]. The difference comes from the different definition of canonical reduced expression in 2.4.2 (cf. [20, 5.5]). We do this in order to have an inductive procedure of generating the basis. The main theorem of Lakshmibai though, applies to this case basically without change.

2.5.2 For any $w \in \check{\mathbf{W}}_\lambda$ let $e_w \in {L_\lambda}^{w(\lambda)}$ be the element in the canonical basis (2.5.9 together with the fact that the transition matrix from the L-S basis to the canonical one is upper triangular with ones on the diagonal, will give an explicit expression for e_w; for now, it is enough to know that these elements are canonical.) Then the $\mathbf{U}^+$-modules $V_{\lambda,w} = \mathbf{U}^+ e_w$ are called Demazure modules. Each of them contains a lattice ${}_{\mathcal{A}}L_{\lambda,w} = {}_{\mathcal{A}}\mathbf{U}^+ e_w$.

Proposition *Let w_λ^0 be the maximal element in $\check{\mathbf{W}}_\lambda$, then ${}_{\mathcal{A}}L_{\lambda,w_\lambda^0}$ is stable under the action of ${}_{\mathcal{A}}\mathbf{U}$.*

It is enough to show that ${}_{\mathcal{A}}L_{\lambda,w_\lambda^0}$ is invariant under the generators of the algebra. The statement for $E_i^{(k)}$ and $\tilde{K}_{\pm i}$ is trivial. Let $Y = u^+ e_{w^0}$ for some $u \in {}_{\mathcal{A}}\mathbf{f}$. One shows the statement for $F_i^{(k)}$ using induction over $lev(|u|)$. The statement is obvious for $lev(|u|) = 0$. Suppose it holds for $lev(|u|) \leq l$. Now let y be as above with $lev(|u|) = l + 1$. Without loss of generality it can be assumed that $y = E_I^{(k')} y'$ for some $i \leq n$ and $k' \in \mathbf{N}$. Then from 2.1.6 and the induction hypothesis it follows that $F_i^{(k)} y$ is in ${}_{\mathcal{A}}L_{\lambda,w^0}$.

2.5.3 Let $\underline{w} : w_1 > w_2 > \ldots > w_r$ be a sequence of linearly ordered cosets in $\check{\mathbf{W}}_\lambda$, and $\underline{a} : 0 = a_0 < a_1 \ldots < a_r = 1$ be a sequence of rational numbers. Then the pair $\pi = (\underline{w}; \underline{a})$ is called L-S path of shape λ if for each pair (w_i, w_{i+1}) there exists a sequence $\{k_i\}_{i=0}^s$ of elements in $\tilde{\mathbf{W}}_\lambda$, such that the following conditions are satisfied:

(a) $k_0 = w_i > k_1 = s_{\beta_1} w_i > k_2 = s_{\beta_2} s_{\beta_1} w_i > \ldots k_s = s_{\beta_s} \ldots s_{\beta_1} w_i = w_{i+1}$,

(b) $l(k_i) = l(k_{i-1}) - 1$, and $a_{i+1}\langle h_{\beta_i}, k_i(\lambda)\rangle \in \mathbf{Z}$,

where $\beta_i \in \mathbf{R}^+$ are positive roots, and h_{β_i} are as in 2.4.4. Let $X_{\mathbf{Q}} = X \otimes_{\mathbf{Z}} \mathbf{Q}$. Then π describes a piecewise linear path $\pi : [0, 1] \to X_{\mathbf{Q}}$ given by

$$\pi(t) = \sum_{i=1}^{j-1} (a_i - a_{i-1}) w_i(\lambda) + (t - a_{j-1}) w_j(\lambda) \quad \text{for} \quad a_{j-1} \leq t \leq a_j.$$

Note that the path starts at the origin, and its end point $\pi(1)$ is in X. Following [22], we denote the set of all L-S paths of shape λ by $\mathcal{P}_\lambda$, the map $\mathcal{P}_\lambda \to \tilde{\mathbf{W}}_\lambda$, $\pi \to w_1$ by Φ, and the map $\mathcal{P}_\lambda \to X$, $\pi \to \pi(1)$ by ν.

A simple, but important example of an L-S path is the path $\pi_\lambda = (\underline{1}; 0, 1)$ – a straight line which connects the origin with λ. Here $\underline{1}$ denotes the image of the identity in $\check{\mathbf{W}}_\lambda$. As it was shown in [22], all the other L-S paths of shape λ can be received by repeatedly applying a set of "decreasing" operators f_i, $i \leq n$ on π_λ. The f_i's are designed in such a way that

$$\nu(f_i(\pi)) = \nu(\pi) - \alpha_i.$$

Now we are going to define these operators in the way they are implemented in the algorithm. For the proof that they have the above property see [22].

2.5.4 Let $\pi = (w_1, \ldots, w_r; a_0, \ldots, a_r) \in \mathcal{P}_\lambda$, and $i \leq n$. Define a function $g_i : [0,1] \to \mathbf{R}$, where $g_i(t) = \langle h_i, \pi(t) \rangle$. Let Q be the minimal integer attained by g_i, and $P = g_i(1) - Q$. Let also p be maximal such that $g_i(a_p) = Q$, and if $p > 0$, denote by x the minimal integer greater or equal to p such that $g_i(t) \geq Q + 1$ for any $t \geq a_x$.

Now, if $P \leq 0$ set $f_i(\pi) = 0$. Otherwise let $f_i(\pi)$ be the following L-S path:

(a) If $g_i(a_x) > Q + 1$, $p > 0$, and $s_i w_{p+1} = w_p$

$$\begin{aligned} f_i(\pi) \;=\; & (w_1, \ldots, w_{p-1}, s_i w_{p+1}, s_i w_{p+2}, \ldots, s_i w_{x-1}, s_i w_x, w_x, \ldots, w_r; \\ & a_0, a_1, \ldots, a_{p-1}, a_{p+1}, \ldots, a_{x-1}, a, a_x, \ldots, a_r), \end{aligned}$$

where $a_{x-1} < a < a_x$ is such that $g_i(a) = Q + 1$.

(b) If $g_i(a_x) > Q + 1$, and either $p = 0$, or $s_i w_{p+1} < w_p$

$$\begin{aligned} f_i(\pi) \;=\; & (w_1, \ldots, w_{p-1}, w_p, s_i w_{p+1}, s_i w_{p+2}, \ldots, s_i w_{x-1}, s_i w_x, w_x, \ldots, w_r; \\ & a_0, a_1, \ldots, a_{x-1}, a, a_x, \ldots, a_r), \end{aligned}$$

where $a_{x-1} < a < a_x$ is such that $g_i(a) = Q + 1$.

(c) If $g_i(a_x) = Q + 1$, $p > 0$, and $s_i w_{p+1} = w_p$

$$\begin{aligned} f_i(\pi) \;=\; & (w_1, \ldots, w_{p-1}, w_p = s_i w_{p+1}, s_i w_{p+2}, \ldots, s_i w_x, w_{x+1}, \ldots, w_r; \\ & a_0, a_1, \ldots, a_{p-1}, a_{p+1}, \ldots, a_r); \end{aligned}$$

(d) If $g_i(a_x) = Q + 1$, and either $p = 0$, or $s_i w_{p+1} < w_p$

$$f_i(\pi) = (w_1, \ldots, w_{p-1}, w_p, s_i w_{p+1}, s_i w_{p+2}, \ldots, s_i w_x, w_{x+1}, \ldots, w_r; a_0, a_1, \ldots, a_r).$$

2.5.5 For any $w \in \tilde{\mathbf{W}}_\lambda$, let $\mathcal{P}_{\lambda,w} = \{\pi \in \mathcal{P}_\lambda \mid \Phi(\pi) \leq w\}$. The following results are proved in [22].

(a) If $\pi \in \mathcal{P}_\lambda$, and $i \leq n$. Then $f_i^m(\pi) = 0$ if and only if $m \geq g_i(1) - \min_t g_i(t)$.

(b) Let $s_i w > w$ and $\pi \in \mathcal{P}_{\lambda,w}$ be such that $\min_t g_i(t) = 0$. Then if $f_i \pi \in \mathcal{P}_{\lambda,w}$, $\{f_i^k \pi, \quad 0 \leq k \leq g_i(1)\} \subset \mathcal{P}_{\lambda,w}$.

(c) If $s_i w > w$, $\mathcal{P}_{\lambda,s_i w} = \{f_i^k \pi, \quad$ where $\pi \in \mathcal{P}_{\lambda,w}$ such that $\min_t g_i(t) = 0, \quad$ and $0 \leq k \leq g_i(1)\}$.

2.5.6 Let $\pi \in \mathcal{P}_\lambda \lambda$. Then given $\mu \in \mathcal{W}$ we say that π is *μ-dominant* if the shifted image $\lambda + \pi(t)$, $t \in [0,1]$ of the path is contained in $\mathcal{W}$. Then (see [22]):

(a) The multiplicity of given weight of L_λ is equal to the number of different L-S paths ending at this weight.

(b) $L_\mu \otimes L_\lambda \simeq \oplus_\pi L_{\mu+\pi(1)}$, where the sum runs over all μ-dominant L-S paths in $\mathcal{P}_\lambda$.

2.5.7 The following result can be found in [20]. Let $w = s_{i_1} \ldots s_{i_r} \in \tilde{\mathbf{W}}_\lambda$ be a reduced expression for w, and let $\pi \in \mathcal{P}_\lambda$ be such that $\Phi(\pi) = w$. Then there exist unique $n_j > 0$, $1 \leq j \leq r$ such that if we set $\pi_j = f_{i_j}^{n_j} \ldots f_{i_r}^{n_r} \pi_\lambda$, $\pi_1 = \pi$, and

(a) $min_t \langle h_{i_j}, \pi_{j-1} \rangle = 0$, $1 < j \leq r$.

(b) $\Phi(\pi_j) = s_{i_j} s_{i_{j+1}} \ldots s_{i_r}$, $1 \leq j \leq r$.

Let $\pi \in \mathcal{P}_{\lambda,w}$, i.e. $\Phi(\pi) \leq w$. Then the unique expression for π given by the above proposition, using the canonical reduced expression for $\Phi(\pi)$ with respect to w, is called the canonical reduced expression for π with respect to w.

2.5.8 Let $w \in \tilde{\mathbf{W}}_\lambda$, and $w = s_{i_1} \dots s_{i_r}$ be a reduced expression for w. Set $w_{s-j+1} = s_{i_j} s_{i_{j+1}} \dots s_{i_r}$, $1 \leq j \leq r$, and $w_0 = 1$. Let $\Xi_{\lambda,w} : \mathcal{P}_\lambda \to {}_{\mathcal{A}}\mathbf{U}^- e_1$ be the map given by $\pi \to Q_\pi$, where if $\pi = f_{j_1}^{n_j} \dots f_{j_s}^{n_s} \pi_\lambda$ is the canonical expression for π with respect to w, $Q_\pi = F_{j_1}^{(n_j)} \dots F_{j_s}^{(n_s)} e_1$. Note that by definition $\mathcal{P}_{\lambda,w_j} \subset \mathcal{P}_{\lambda,w_{j+1}}$. Denote with $\mathcal{B}_{\lambda,w_j}$ the image of $\mathcal{P}_{\lambda,w_j}$ under $\Xi_{\lambda,w}$. Then we have

(a) $\mathcal{B}_{\lambda,w_r} \supset \mathcal{B}_{\lambda,w_{r-1}} \supset \dots \supset \mathcal{B}_{\lambda,w_1} \supset \mathcal{B}_{\lambda,w_0}$,

(b)

$$\mathcal{B}_{\lambda,w_j} \backslash \mathcal{B}_{\lambda,w_{j-1}} = \{ \quad F_{i_j}^{(m_j)} Q_{\pi'},\ 1 \leq m_j \leq \langle h_{i_j}, \nu(\pi') \rangle \mid \pi' \in \mathcal{P}_{\lambda,w_{j-1}} \text{ with } \min_t \langle h_{i_j}, \pi'(t) \rangle = 0 \text{ and } f_{i_j}\pi' \text{ not in } \mathcal{P}_{\lambda,w_{j-1}} \}.$$

(a) follows from the observation that if $\tau \leq w_{j-1}$, the canonical expression for τ with respect to w_j is the same as the canonical expression for τ with respect to w_{j-1}. Then (b) follows from 2.5.5 and 2.5.7.

2.5.9 To state the result of Lakshmibai in [20], note that ([17, 11]) there is a subset of the canonical basis of L_λ which is an $\mathcal{A}$-basis of ${}_{\mathcal{A}}L_{\lambda,w}$. Then if $w = s_{i_1} \dots s_{i_r}$ is a reduced expression for an element in $\tilde{\mathbf{W}}_\lambda$, $\mathcal{B}_{\lambda,w}$ is a basis for the $\mathcal{A}$-module ${}_{\mathcal{A}}L_{\lambda,w}$. Moreover, there is an ordering of bases such that the transition matrix from $\mathcal{B}_{\lambda,w}$ to the canonical basis of ${}_{\mathcal{A}}L_{\lambda,w}$ is upper triangular with 1's on the diagonal.

2.5.10 The result above shows that choice of a presentation $w_\lambda^0 = s_{i_1} s_{i_2} \dots s_{i_l}$ of the maximal element in $\tilde{\mathbf{W}}_\lambda$ fixes a choice of basis $\mathcal{B}_{\lambda,\mathbf{i}}$ of L_λ. Since the transition matrix from this basis to the canonical one is upper triangular with 1's on the diagonals, for any presentation of w_λ^0 as above, any nonzero element $F_{i_1}^{(m_1)} \dots F_{i_l}^{(m_l)}$ with $m_j > 0$ in ${L_\lambda}^{w^0(\lambda)}$ such that $\pi_j = f_{i_j}^{m_j} \dots f_{i_r}^{m_l} \pi_\lambda$ satisfies (a) and (b) from 2.5.7, is equal to the canonical basic element $e_{w_\lambda^0}$ in this weight space. We set $\epsilon_\lambda = e_{w_\lambda^0}$

2.6 Some properties of the duality morphisms

The duality morphisms in the categories which lead to three dimensional topological quantum field theories should satisfy a "symmetry condition"

(see [32, 33]). This condition puts restrictions on the type of selfdual simple objects in the category. Here we establish which selfdual objects in $\mathcal{C}$ satisfy the symmetry condition. The result must be well known since this is obviously what lies in the difference between the definitions of the quantum enveloping algebra we gave here (introduced by Lusztig) and the original one of Drinfeld and Jimbo, where, among other differences, instead of $\tilde{K}_{\pm i}$, their squares are used as algebra generators.

2.6.1 Let ${}_{\mathcal{A}}L_\lambda{}^*$ be the $\mathcal{A}$-lattice of $L_\lambda{}^*$ generated by the dual basis of $\mathcal{B}_{\lambda \mathbf{j}}$. Since $u \in {}_{\mathcal{A}}\mathbf{U}$ acts on ${}_{\mathcal{A}}L_\lambda{}^*$ by the dual of $S(u)$, from 2.1.6 and 2.5.2 follows that ${}_{\mathcal{A}}L_\lambda{}^*$ is invariant under the action of ${}_{\mathcal{A}}\mathbf{U}$. Let $\vartheta_\lambda : L_\lambda \to L_{-w^0(\lambda)}{}^*$ be the isomorphism in $\mathcal{C}$ such that $1 \to \epsilon_\lambda^*$, where ϵ_λ^* is the dual vector of ϵ_λ. Such map exists because of 2.4.8 and 2.3.5. Observe that because the elements of the Laksmibai basis are of the form u^-1, with $u \in {}_{\mathcal{A}}\mathbf{f}$, ${}_{\mathcal{A}}L_\lambda \subset {}_{\mathcal{A}}\mathbf{U}^-1$. Then from the ${}_{\mathcal{A}}\mathbf{U}$-invariance of ${}_{\mathcal{A}}L_{-w^0(\lambda)}{}^*$ follows that $\vartheta_\lambda : {}_{\mathcal{A}}L_\lambda \to {}_{\mathcal{A}}L_{-w^0(\lambda)}{}^*$. From now on, given any $\lambda \in X$, we set $\hat{\lambda} = -w^0(\lambda)$.

2.6.2 Given any function $f : X \to \mathbf{Z}$ and any $M \in \mathcal{C}$, denote with $\hat{v}^f : M \to M$ the linear operator given by $\hat{v}^f m = v^{f(\lambda)} m$ for any $m \in M^\lambda$.

Proposition 2.6.3 *Let $\zeta_\lambda : L_\lambda \to L_\lambda{}^{**}$ be the standard $\mathbf{Q}(v)$-isomorphism, and $\rho : X \to \mathbf{Z}$ be given by $\rho(\sum_{i=1}^n \mu_i \alpha_i) = -\sum_{i=1}^n \mu_i$ i.i. Then*

(a) The composition $\pi_\lambda = \vartheta_{\hat{\lambda}}^* \circ \zeta_\lambda \circ \hat{v}^\rho : L_\lambda \to L_{\hat{\lambda}}{}^*$ is an isomorphism in $\mathcal{C}$.

(b) If $\lambda = \hat{\lambda}$, $\pi_\lambda = (-1)^{lev(2\lambda)} \vartheta_\lambda$.

Since all maps are $\mathbf{Q}(v)$-isomorphisms, so is π_λ. Now we want to show that for any $u \in \mathbf{U}$, and $m \in L_\lambda{}^\mu$, $\pi_\lambda(um) = u\pi_\lambda(m)$. From 2.2.8 follows that it is enough to show the statement for $u \in \mathbf{U}_\nu^- \mathbf{U}^0 \mathbf{U}_{\nu'}^+$ for any $\nu, \nu' \in X^+$. Given such u set $|u| = \nu - \nu' \in X$. Then if $m' \in L_{\hat{\lambda}}{}^{-\mu}$,

$$(\pi_\lambda um)(m') = v^{\rho(\mu+|u|)}(\theta_{\hat{\lambda}} S^{-1}(u)m')(m) \text{ and}$$
$$(u\pi_\lambda m)(m') = v^{\rho(\mu)}(\theta_{\hat{\lambda}} S(u)m')(m).$$

Hence, it is enough to show that $v^{\rho(|u|)} S^{-1}(u) = S(u)$. Because of the additivity of ρ if this is true for u_1, u_2, it is true for their product. So, it is

enough to check it on the generators, where it follows from the definition of S.

To show the statement in (b), first note that since $\lambda - w^0(\lambda) \in X^+$ in this case $2\lambda \in X^+$, and $(-1)^{lev(2\lambda)}$ is well defined. Because of (a) it is enough to check (b) on ${L_\lambda}^\lambda$ (Note that since L_λ is a quotient of $\mathbf{U}^-1$, the value of the homomorphism into the highest weigh space determines the whole map). Let 1 denote the basic vector in ${L_\lambda}^\lambda$. If $w_\lambda^0 = s_{j_1}s_{j_2}\dots s_{j_l}$ is a reduced expression of the maximal element in $\tilde{\mathbf{W}}_\lambda$ we have (see 2.5.9) $\epsilon_\lambda = x^-1$, where $x = \theta_{j_1}^{(m_1)}\dots\theta_{j_l}^{(m_l)} \in \mathbf{f}$. Using 2.2.7, and the fact that $|x| = 2\lambda$, we get

$$\begin{aligned}(\pi_\lambda 1)(\epsilon_\lambda) &= v^{\rho(\lambda)}(\theta_\lambda\epsilon_\lambda)(1)\\ &= v^{\rho(\lambda)}(x^-\theta_\lambda 1)(1) = v^{\rho(\lambda)}\epsilon_\lambda^*(S(x^-)1)\\ &= (-1)^{lev(2\lambda)}v^{-2\lambda.\lambda}\epsilon_\lambda^*(\sigma(x)^-\tilde{K}_{|x|}1)\\ &= (-1)^{lev(2\lambda)}\epsilon_\lambda^*(\sigma(x)^-1).\end{aligned}$$

Now it is enough to observe that since $w_\lambda^0 = (w_\lambda^0)^{-1}$, $s_{j_l}s_{j_{l-1}}\dots s_1$ is also a reduced expression for w_λ^0. Moreover, from 2.5.5 one can deduce that $f_{j_l}^{m_l}\dots f_{j_1}^{m_1}\pi_\lambda$ is the canonical reduced expression for the corresponding path. Then from 2.5.10 it follows that $\sigma(x)^-1 = \epsilon_\lambda$. The statement follows.

2.6.4 The selfdual simple module L_λ is called *symmetric* if $\pi_\lambda = \vartheta_\lambda$. The proposition above states that L_λ is symmetric if and only if $lev(2\lambda) \in 2\mathbf{Z}$.

2.7 The quasi R-matrix

The definition of universal R-matrix is due to Drinfeld ([3, 4]). Its modified form - the quasi R-matrix, was introduced by Lusztig ([29]), and differs from the universal R-matrix by an element in $\mathbf{U}^0$. Here we follow the exposition in [26].

2.7.1 Let $\widehat{(\mathbf{U}\otimes\mathbf{U})}$ be the completion of the vector space $\mathbf{U}\otimes\mathbf{U}$ with respect to the descending sequence of vector spaces

$$\mathcal{H}_N = (\mathbf{U}^+\mathbf{U}^0(\textstyle\sum_{lev(\nu)\geq N}\mathbf{U}_\nu^-))\otimes\mathbf{U} + \mathbf{U}\otimes(\mathbf{U}^-\mathbf{U}^0(\textstyle\sum_{lev(\nu)\geq N}\mathbf{U}_\nu^+)),$$

where ${\mathbf{U}^-}_\nu$ (resp. ${\mathbf{U}^+}_\nu$), $\nu \in X^+$ is the image of $\mathbf{f}_\nu$ under the maps in 2.2.5. Then the $\mathbf{Q}(v)$-algebra structure extends by continuity on $\widehat{(\mathbf{U}\otimes\mathbf{U})}$.

Let $^- : \mathbf{U} \otimes \mathbf{U} \to \mathbf{U} \otimes \mathbf{U}$ be the $\mathbf{Q}$-algebra automorphism given by $^- \otimes ^-$. This extends by continuity to a $\mathbf{Q}$-algebra automorphism $^- : (\mathbf{U} \otimes \mathbf{U})\hat{} \to (\mathbf{U} \otimes \mathbf{U})\hat{}$.

Let $\overline{\triangle} : \mathbf{U} \to \mathbf{U}$ be the $\mathbf{Q}(v)$-algebra homomorphism given by $\overline{\triangle(x)} = \overline{\triangle}(\overline{x})$ for all $x \in \mathbf{U}$.

2.7.2 The following result (see [26, 29]) defines an element $\Theta \in (\mathbf{U} \otimes \mathbf{U})\hat{}$, called the quasi R-matrix.

(a) There is a unique family of elements $\Theta_\nu \in {\mathbf{U}^-}_\nu \otimes {\mathbf{U}^+}_\nu$, $\nu \in X^+$, such that $\Theta_0 = 1 \otimes 1$, and $\Theta = \sum_{\nu \in X} \Theta_\nu \in (\mathbf{U} \otimes \mathbf{U})\hat{}$ satisfies $\triangle(u)\Theta = \Theta\overline{\triangle}(u)$ for all $u \in \mathbf{U}$.

(b) Let B be a $\mathbf{Q}(v)$-basis of $\mathbf{f}$ such that $B_\nu = B \cap \mathbf{f}_\nu$ is a basis of $\mathbf{f}_\nu$ for any ν. Let $\{b^* | b \in B\}$ be the basis of $\mathbf{f}_\nu$ dual to B_ν under $(,)$. We have

$$\Theta_\nu = (-1)^{lev(\nu)} v_\nu \sum_{b \in B_\nu} b^- \otimes b^+ \in {\mathbf{U}^-}_\nu \otimes {\mathbf{U}^+}_\nu.$$

(c) $\Theta\overline{\Theta} = \overline{\Theta}\Theta = 1 \otimes 1$.

2.7.3 For an element $P = \sum x \otimes y \in \mathbf{U} \otimes \mathbf{U}$ denote the elements $\sum x \otimes y \otimes 1$, $\sum 1 \otimes x \otimes y$, $\sum x \otimes 1 \otimes y$ of $\mathbf{U} \otimes \mathbf{U} \otimes \mathbf{U}$ by P^{12}, P^{23} and P^{13} respectively. Then

(a) $(\overline{\triangle} \otimes 1)(\Theta_\nu) = \sum_{\nu+\nu''=\nu} \Theta^{13}_{\nu'}(1 \otimes \tilde{K}_{\nu'} \otimes 1)\Theta^{23}_{\nu''}$.

(b) $(1 \otimes \triangle)\Theta_\nu = \sum_{\nu'+\nu''=\nu} \Theta^{12}_{\nu'}(1 \otimes \tilde{K}_{\nu''} \otimes 1)\Theta^{13}_{\nu''}$.

2.7.4 Specializing the identities $\triangle(E_i)\Theta = \Theta\overline{\triangle}(E_i)$ and $\triangle(F_i)\Theta = \Theta\overline{\triangle}(F_i)$ on $\mathbf{U}^-_{\nu-\alpha_i} \otimes \mathbf{U}^+_\nu$ and $\mathbf{U}^-_\nu \otimes \mathbf{U}^+_{\nu-\alpha_i}$ we deduce

(a) $(E_i \otimes 1)\Theta_\nu + (\tilde{K}_i \otimes E_i)\Theta_{\nu-\alpha_i} = \Theta_\nu(E_i \otimes 1) + \Theta_{\nu-\alpha_i}(\tilde{K}_{-i} \otimes E_i)$,

(b) $(1 \otimes F_i)\Theta_\nu + (F_i \otimes \tilde{K}_{-i})\Theta_{\nu-\alpha_i} = \Theta_\nu(1 \otimes F_i) + \Theta_{\nu-\alpha_i}(F_i \otimes \tilde{K}_i)$.

2.8 The operators Ω' and Ω''

The operators Ω' and Ω'' are introduced by analogy with Lusztig's operator Ω ([26, 6.1]). Actually, Ω' is the inverse of Ω. These operators are later used in constructing the Casimir operator and proving the symmetry condition. In this section B denotes a basis of $\mathbf{f}$ as in 2.7.2(b).

Proposition 2.8.1 *For any object $M \in \mathcal{C}$ there are operators $\Omega' : M \to M$, and $\Omega'' : M \to M$ such that*

(a) $\Omega'(m) = \sum_{b\in B}(-1)^{tr|b|}v_{-|b|}\tilde{K}_{-|b|}b^-b^{*+}m$, *for any* $m \in M$,

(b) As operators on M

$$E_i\Omega' = \tilde{K}_{-2i}\Omega'E_i, \quad \Omega F_i = \tilde{K}_{-2i}F_i\Omega', \quad \Omega'K_\mu = K_\mu\Omega'.$$

(c) $\Omega''(m) = \sum_{b\in B}(-1)^{tr|b|}v_{-|b|}\tilde{K}_{|b|}b^{*+}b^-m$, *for any* $m \in M$,

(d) As operators on M

$$E_i\Omega''\tilde{K}_{2i} = \Omega''E_i, \quad \tilde{K}_{2i}\Omega''F_i = F_i\Omega'', \quad \Omega''K_\mu = K_\mu\Omega''.$$

The proof is adaptation of arguments in [26, 6.1.1]. Let

$$\Omega'_{\leq N} = \sum_{\nu:lev(\nu)\leq N}(-1)^{lev(\nu)}v_\nu v^{-\sum_{i=1}^n \nu_i i.i}\tilde{K}_{-\nu}\sum_{b\in B_\nu} b^-b^{*+}, \text{ and}$$

$$\Omega''_{\leq N} = \sum_{\nu:lev(\nu)\leq N}(-1)^{lev(\nu)}v_\nu v^{-\sum_{i=1}^n \nu_i i.i}\tilde{K}_{\nu}\sum_{b\in B_\nu} b^{*+}b^-.$$

From 2.3.2 follows that for high enough N, $\Omega'_{\leq N}m$ and $\Omega''_{\leq N}m$ are independent of N for any $m \in M$. Hence the operators in (a) and (c) are well defined. To see that Ω' satisfies (b) first note that

$$\Omega'_{\leq N} = \sum_{\nu:lev(\nu)\leq N}(-1)^{lev(\nu)}v_\nu \sum_{b\in B_\nu}\overline{S}^2(\tilde{K}_{-\nu}b^-)b^{*+},$$

where $\overline{S}(u) = \overline{S(\overline{u})}$ for any $u \in \mathbf{U}$. Now we apply $v^{\nu.i}m(\overline{S}^2 \otimes 1)(\tilde{K}_{-\nu} \otimes 1)$ on 2.7.4 (a) and deduce

$$-(-1)^{lev(\nu)}v_\nu \sum_{b\in B_\nu}(v^{i.i}E_i\overline{S}^2(\tilde{K}_{-\nu}b^-)b^{*+} - v^{\nu.i+i.i}\overline{S}^2(\tilde{K}_{-\nu}b^-)E_ib^{*+}) =$$

$$(-1)^{lev(\nu-\alpha_i)}v_{\nu-\alpha_i}\sum_{b\in B_{\nu-\alpha_i}}(v^{\nu.i}\overline{S}^2(\tilde{K}_{-\nu+\alpha_i}b^-)E_ib^{*+} - v^{i.i}\overline{S}^2(\tilde{K}_{-\nu}b^-)b^{*+}\tilde{K}_{-2i}E_i).$$

where we have used the identities:

$$\overline{S}^2(E_i) = v^{i.i}E_i, \quad \overline{S}^2(F_i) = v^{-i.i}F_i.$$

Now we sum over $\nu : lev(\nu) \leq N+1$ and get

$$E_i\Omega'_{\leq N} \quad -\Omega'_{\leq N}\tilde{K}_{-2i}E_i = \sum_{\nu:lev(\nu)=N+1}(-1)^{lev(\nu)}v_\nu \sum_{b\in B_\nu}(E_i\overline{S}^2(\tilde{K}_{-\nu}b^-)b^{*+} - v^{\nu.i}\overline{S}^2(\tilde{K}_{-\nu}b^-)E_ib^{*+}).$$

From here and 2.3.2 follows the first statement in (b). The commutativity relation with F_i can be deduced in similar way by applying $m(\overline{S}^2\otimes 1)(\tilde{K}_{-\nu}\otimes 1)$ on both sides of 2.7.4(b), and then summing over $\nu : lev(\nu) \leq N+1$.

The proof that Ω'' satisfies (d) is very similar. First note that

$$\Omega''_{\leq N} = \sum_{\nu:lev(\nu)\leq N}(-1)^{lev(\nu)}v_\nu \sum_{b\in B_\nu} S^2(\tilde{K}_\nu b^{*+})b^-.$$

Then we apply $m(S^2\otimes 1)\tilde{s}(1\otimes\tilde{K}_\nu)$ to both sides of 2.7.4(a) , where $\tilde{s} : \mathbf{U}\otimes\mathbf{U} \xrightarrow{x\otimes y - y\otimes x} \mathbf{U}\otimes\mathbf{U}$. In this way we deduce

$$\begin{aligned}&(-1)^{lev(\nu)}v_\nu \sum_{b\in B_\nu}(S^2(\tilde{K}_\nu b^{*+})E_ib^- - S^2(\tilde{K}_\nu b^{*+})b^-E_i)\\ &= (-1)^{lev(\nu-\alpha_i)}v_{\nu-\alpha_i}\sum_{b\in B_{\nu-\alpha_i}}(S^2(\tilde{K}_{\nu-\alpha_i}b^{*+})E_ib^- - v^{-i.i}\tilde{K}_iE_iS^2(\tilde{K}_\nu b^{*+})b^-\tilde{K}_i),\end{aligned}$$

where we have used the identities:

$$S^2(E_i) = v^{-i.i}E_i, \quad S^2(F_i) = v^{i.i}F_i.$$

Now we sum over $\nu : lev(\nu) \leq N+1$ and get

$$\begin{aligned}&E_i\Omega''_{\leq N}\tilde{K}_{2i} - \Omega''_{\leq N}E_i\\ &= -\sum_{\nu:lev(\nu)=N+1}(-1)^{lev(\nu)}v_\nu\sum_{b\in B_\nu}(S^2(\tilde{K}_\nu b^{*+})E_ib^- - S^2(\tilde{K}_\nu b^{*+})b^-E_i)\\ &= -\sum_{\nu:lev(\nu)=N}(-1)^{lev(\nu)}v_\nu\sum_{b\in B_{\nu-\alpha_i}}(S^2(\tilde{K}_{\nu-i}b^{*+})E_ib^- - E_iS^2(\tilde{K}_\nu b^{*+})b^-\tilde{K}_{2i}).\end{aligned}$$

From here follows the first statement in (d). The commutativity relation with F_i can be deduced in similar way by applying $v^{\nu.i}m(S^2\otimes 1)s(1\otimes\tilde{K}_\nu)$ on both sides of 2.7.4(b), and then summing over $\nu : lev(\nu) \leq N+1$.

2.9 Lusztig's basis of ${}_{\mathcal{A}}\mathbf{U}^+$

2.9.1 For each $i \leq n$ Lusztig defines an algebra isomorphism $T_i : \mathbf{U} \to \mathbf{U}$ such that

$$\begin{aligned}
T_i(E_i) &= -\tilde{K}_{-i}F_i, \qquad\qquad T_i(F_i) = -E_i\tilde{K}_i,\\
T_i(E_j) &= \sum_{r+s=-\langle i,j\rangle} (-1)^r v_i^{-r} E_i^{(r)} E_j E_i^{(s)} \quad \text{for} \quad j \neq i,\\
T_i(F_j) &= \sum_{r+s=-\langle i,j\rangle} (-1)^r v_i^{r} F_i^{(s)} F_j F_i^{(r)} \quad \text{for} \quad j \neq i,\\
T_i(K_\mu) &= K_{\mu-\langle \mu,i\rangle h_i}.
\end{aligned}$$

The isomorphisms T_i arise as a representation of the braid group in $\mathbf{U}$, where the braid group is the group defined by the same generators as the Weyl group, and relations 4.4.1(a). We won't use this fact directly, so won't expand on it. Here is the result of Lusztig which establishes the existence of an orthogonal basis of $\mathbf{f}$ ([26, 41.1.4]).

2.9.2 Let l^0 be the length of the maximal element w^0 of $\mathbf{W}$. Fix $\mathbf{i} = (i_1, i_2, \ldots, i_{l^0})$, $i_r \leq n$ such that $s_{i_1} \ldots s_{i_{l^0}}$ is a reduced expression for w^0. Then

(a) $T_{i_1} T_{i_2} \ldots T_{i_{k-1}}(E_{i_k}^{(c_k)}) \in {}_{\mathcal{A}}\mathbf{U}^+$ for any $k \leq l^0$, and $c_k \geq 0$;

(b) Set $\beta_k = s_{i_1} \ldots s_{i_{k-1}}(\alpha_k)$, and $E_{\mathbf{i},\beta_k} = T_{i_1} \ldots T_{i_{k-1}}(E_{i_k})$. Then $\{E_{\mathbf{i},\beta_1}^{(c_1)} E_{\mathbf{i},\beta_2}^{(c_2)} \ldots E_{\mathbf{i},\beta_{l^0}}^{(c_{l^0})}, \quad c_i \geq 0\}$ is a basis for ${}_{\mathcal{A}}\mathbf{U}^+$;

(c) The basis above induces an orthogonal basis $B_{\mathbf{i}}$ of ${}_{\mathcal{A}}\mathbf{f}$. In particular, set $\mathbf{c} = \{c_1, \ldots, c_{l^0}\}$, and let $b_{\mathbf{i},\mathbf{c}}$ be the element in ${}_{\mathcal{A}}\mathbf{f}$, which maps to $E_{\mathbf{i},\beta_1}^{(c_1)} E_{\mathbf{i},\beta_2}^{(c_2)} \ldots E_{\mathbf{i},\beta_{l^0}}^{(c_{l^0})}$ in ${}_{\mathcal{A}}\mathbf{U}^+$. Then

$$\begin{aligned}
(b_{\mathbf{i},\mathbf{c}}, b_{\mathbf{i},\mathbf{c}'}) &= (\theta_{i_1}^{(c_1)}, \theta_{i_1'}^{(c_1')}) \ldots (\theta_{i_{l^0}}^{(c_{l^0})}, \theta_{i_{l^0}'}^{(c_{l^0}')})\\
&= \prod_{j=1}^{l^0} \delta_{c_j,c_j'} \frac{v_{i_j}^{c_j(c_j+1)/2}}{[c_j]_{i_j}^{!}(v_{i_j} - v_{i_j}^{-1})^{c_j}}
\end{aligned}$$

2.9.3 Obviously, the basis $B_{\mathfrak{i}}$ of ${}_{\mathcal{A}}\mathbf{f}$ satisfies the conditions in 2.7.2, and therefore can be used in constructing an expression for the quasi R-matrix, and the operators Ω' and Ω''. In particular, from (c) above follows that $\Omega'_{\leq N}$ and $\Omega''_{\leq N}$ are in ${}_{\mathcal{A}}\mathbf{U}$, and $\Theta_\nu \in {}_{\mathcal{A}}\mathbf{U}^- \otimes {}_{\mathcal{A}}\mathbf{U}^+$.

Chapter 3
Quantum algebras at prime root of unity

3.1 Change of ring.

In defining the algebras ${}_{\mathcal{R}'}\mathbf{U}$ and ${}_{\mathcal{R}'}\mathbf{u}$ we follow [25]. The algebra ${}_{\mathcal{R}'}\mathbf{U}$ is originally defined as the tensor product of ${}_{\mathcal{A}}\mathbf{U}$ with the corresponding ring, and ${}_{\mathcal{R}'}\mathbf{u}$ is a subalgebras of ${}_{\mathcal{R}'}\mathbf{U}$. But such kind of definition is difficult to work with. So, in [25], by putting some conditions on the ordering of the positive roots, Lusztig works out equivalent definitions in terms of generators and relations. Since the construction is long and elaborate, and this is a computationally oriented work, we will restrict ourselves only to listing the specific choices of ordering of the positive roots made here in agreement with the condition in [25], and presenting a limited number of relations which will be used in the programs.

3.1.1 Let p be an odd prime such that $p > 3$ if $\mathbf{C}$ is the Cartan matrix of the algebra $\mathbf{G}_2$. Let $\mathcal{R}$ be the quotient ring of $\mathcal{A}$ by the ideal generated by the p'th cyclotomic polynomial, and $\mathcal{R}'$ the quotient field of $\mathcal{R}$. Denote the image of v in $\mathcal{R}$ and $\mathcal{R}'$ with the same letter.

We say that $\alpha_{i_1}, \ldots, \alpha_{i_n}$ is a *good numbering* of the simple roots if the coefficient with which α_{i_j} appears in any element of $\mathbf{R}^+ \cap (\mathbf{N}\alpha_{i_1} + \ldots \mathbf{N}\alpha_{i_j})$ is at most one, except possibly for a single root where it is necessarily two. Given a good ordering of the simple roots, define a preorder on $\mathbf{R}^+$ in the following way: $\beta_1 \leq \beta_2$ if $m_1 \geq m_2$, and $\frac{\sum_{i=1}^{m_1} c_i^1}{c_{m_1}^1} \leq \frac{\sum_{i=1}^{m_2} c_i^2}{c_{m_2}^2}$, where $\beta_k = \sum_{i=1}^{m_k} c_i^k \alpha_i$, $k = 1, 2$, and $c_{m_1}^1 c_{m_2}^2 > 0$.

One can always choose a presentation of the maximal element of the Weyl group $w^0 = s_{i_1} \ldots s_{i_{l0}}$, such that the induced order of $\mathbf{R}^+$ satisfies the condition above. From now on we will consider such choice fixed, and the basis $B_{\mathbf{i}} = \{b_{\mathbf{i},\mathbf{c}},\ \mathbf{c} \in \mathbf{N}^n\}$ of ${}_{\mathcal{A}}\mathbf{f}$ (see 2.9.2) will be denoted just as $B = \{b_{\mathbf{c}},\ \mathbf{c} \in \mathbf{N}^n\}$.

3.1.2 If $\mathbf{k}$ is a ring such that there is a homomorphism $\mathcal{A} \to \mathbf{k}$, set ${}_{\mathbf{k}}\mathbf{U} = \mathbf{k} \otimes_{\mathcal{A}} ({}_{\mathcal{A}}\mathbf{U})$, and ${}_{\mathbf{k}}\mathbf{U}^{\pm} = \mathbf{k} \otimes_{\mathcal{A}} ({}_{\mathcal{A}}\mathbf{U}^{\pm})$ ${}_{\mathbf{k}}\mathbf{f} = \mathbf{k} \otimes_{\mathcal{A}} ({}_{\mathcal{A}}\mathbf{f})$. Then there is a direct sum decomposition ${}_{\mathbf{k}}\mathbf{f} = \oplus_{\nu \in X^+} {}_{\mathbf{k}}\mathbf{f}_\nu$, where ${}_{\mathbf{k}}\mathbf{f}_\nu = \mathbf{k} \otimes_{\mathcal{A}} ({}_{\mathcal{A}}\mathbf{f}_\nu)$. Note that through the quotient homomorphism $\Phi : \mathcal{A} \to \mathcal{R}$, $\mathcal{R}$ and $\mathcal{R}'$ can be

regarded as $\mathcal{A}$-algebras. Then we will denote the elements $1 \otimes E_{\beta_j}^{(N)}$, $1 \otimes F_{\beta_j}^{(N)}$, $1 \otimes b_{\mathbf{c}}^{+}$, and $1 \otimes b_{\mathbf{c}}^{-}$ in ${}_{\mathcal{R}}\mathbf{U}$ again with $E_{\beta_j}^{(N)}$, $F_{\beta_j}^{(N)}$, $b_{\mathbf{c}}^{+}$, and $b_{\mathbf{c}}^{-}$ respectively. Obviously, B is an $\mathcal{R}$-basis of ${}_{\mathcal{R}}\mathbf{U}$, and an $\mathcal{R}$'-basis of ${}_{\mathcal{R}'}\mathbf{U}$.

Define ${}_{\mathcal{R}}\mathbf{u}^{+}$ (${}_{\mathcal{R}}\mathbf{u}^{-}$) to be the subalgebra of ${}_{\mathcal{R}}\mathbf{U}^{+}$ (resp. ${}_{\mathcal{R}}\mathbf{U}^{-}$) generated by the elements $E_{\beta_j}^{(N)}$(resp.$F_{\beta_j}^{(N)}$),$i \leq n$, $0 \leq N \leq p-1$, and ${}_{\mathcal{R}}\mathbf{u}$ (${}_{\mathcal{R}'}\mathbf{u}$) to be the $\mathcal{R}$-subalgebra (resp. $\mathcal{R}'$-subalgebra) of ${}_{\mathcal{R}}\mathbf{U}$ (resp. ${}_{\mathcal{R}'}\mathbf{U}$) generated by the elements $E_{\beta_j}^{(N)}$, $F_{\beta_j}^{(N)}$, and $\tilde{K}_{\pm i}$, $i \leq n$, $j \leq l^0$, $0 \leq N \leq p-1$ modulo the ideal generated by the central elements $\tilde{K}_i^{\,p} - \mathbf{1}$, $i \leq n$.

3.1.3 In [25] Lusztig proves that the elements $b_{\mathbf{c}}^{+}$ ($b_{\mathbf{c}}^{-}$), with $0 \leq c_i < p$ form an $\mathcal{R}$-basis of ${}_{\mathcal{R}}\mathbf{u}^{+}$ (resp.${}_{\mathcal{R}}\mathbf{u}^{-}$). In particular, ${}_{\mathcal{R}}\mathbf{u}$ is finite dimensional with $dim({}_{\mathcal{R}}\mathbf{u}) = p^{2l^0+n}$.

3.1.4 ${}_{\mathcal{A}}\mathbf{U}^{\pm}$ can be defined as the $\mathcal{A}$- algebras with generators $E_{\beta}^{(N)}$, and $F_{\beta}^{(N)}$, satisfying the set of relations described in [25]. This leads to corresponding definitions for ${}_{\mathcal{R}}\mathbf{U}$, and ${}_{\mathcal{R}}\mathbf{u}$. For the algebras rank two these relations can be computed directly from the definitions of E_{β}. For higher rank algebras they correspond to two dimensional planes passing through the corresponding roots. We are going to use a limited number of these relations which are listed bellow together with the choices of a reduced expression of the maximal element of the Weyl group, and the corresponding order of the elements in $\mathbf{R}^{+}$. In ${}_{\mathcal{R}'}\mathbf{U}$ the relations, listed bellow, define an inductive procedure for expressing E_{β} ,$\beta \in \mathbf{R}^{+}$ in terms of E_i, $i \leq n$. In particular, ${}_{\mathcal{R}'}\mathbf{u}$ is generated by E_i, F_i, $\tilde{K}_{\pm i}$, $i \leq n$. It can be shown ([23]) that ${}_{\mathcal{R}'}\mathbf{U}$ is generated by E_i, $E_i^{(p)}$, F_i, $F_i^{(p)}$, $\tilde{K}_{\pm i}$, $i \leq n$.

(a) **Algebra type A_n**

$$\mathbf{C} = \begin{pmatrix} 2 & -1 & \dots & 0 & 0 \\ -1 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 2 & -1 \\ 0 & 0 & \dots & -1 & 2 \end{pmatrix}$$

$$w^0 = (s_n s_{n-1} \dots s_1)(s_n s_{n-1} \dots s_2) \dots (s_n s_{n-1}) s_n$$

$$\mathbf{R}^{+} = \{\alpha_n,\ \alpha_n + \alpha_{n-1}, \dots, \alpha_n + \dots + \alpha_1,\ \alpha_{n-1},\ \alpha_{n-1} + \alpha_{n-2}, \dots, \alpha_1\}$$

Relations: For any $k \geq l \geq 1$, and $\beta = \alpha_k + \alpha_{k-1} + \ldots + \alpha_{l-1}$, we have

$$E_{\beta+\alpha_l} = E_\beta E_l - v^{-1} E_l E_\beta.$$

(b) **Algebra type B_n**

$$\mathbf{C} = \begin{pmatrix} 2 & -1 & \ldots & 0 & 0 \\ -1 & 2 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 2 & -2 \\ 0 & 0 & \ldots & -1 & 2 \end{pmatrix}$$

$$\begin{aligned} w^0 = {} & s_1 s_2 \ldots s_{n-1} s_n s_{n-1} s_{n-2} \ldots s_2 s_1 \\ & s_2 s_3 \ldots s_{n-1} s_n s_{n-1} s_{n-2} \ldots s_2 \\ & \ldots \\ & s_{n-1} s_n s_{n-1} \\ & s_n \end{aligned}$$

$$\begin{aligned} \mathbf{R}^+ = {} & \{\alpha_1,\ \alpha_1 + \alpha_2, \ldots, \alpha_1 + \ldots + \alpha_n, \\ & \alpha_1 + \ldots + \alpha_{n-1} + 2\alpha_n,\ \ldots, \alpha_1 + 2\alpha_2 \ldots + 2\alpha_n, \\ & \alpha_2,\ \alpha_2 + \alpha_3, \ldots, \alpha_2 + \ldots + \alpha_n, \\ & \alpha_2 + \ldots + \alpha_{n-1} + 2\alpha_n,\ \ldots, \alpha_2 + 2\alpha_3 \ldots + 2\alpha_n, \\ & \ldots \\ & \alpha_{n-1},\ \alpha_{n-1} + \alpha_n,\ \alpha_{n-1} + 2\alpha_n, \\ & \alpha_n\} \end{aligned}$$

Relations:

Case1. For any $k < l \leq n-1$, and $\beta = \alpha_k + \alpha_{k+1} + \ldots + \alpha_{l-1}$, or $\beta = \alpha_k + \alpha_{k+1} + \ldots + \alpha_l + 2\alpha_{l+1} + \ldots 2\alpha_n$, we have

$$E_{\beta+\alpha_l} = E_\beta E_l - v^{-2} E_l E_\beta.$$

Case2. For any $k \leq n-1$, and $\beta = \alpha_k + \alpha_{k+1} + \ldots + \alpha_{n-1}$, we have

$$\begin{aligned} E_{\beta+\alpha_n} &= E_n E_\beta - v^{-2} E_\beta E_n, \\ E_{\beta+2\alpha_n} &= E_n^{(2)} E_\beta - v^{-1} E_n E_\beta E_n + v^{-2} E_\beta E_n^{(2)}. \end{aligned}$$

(c) **Algebra type C_n**

$$\mathbf{C} = \begin{pmatrix} 2 & -1 & \dots & 0 & 0 \\ -1 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 2 & -1 \\ 0 & 0 & \dots & -2 & 2 \end{pmatrix}$$

$$w^0 = (\prod_{k=1}^{[\frac{n}{2}]} s_n s_{n-2} \dots s_{n-2k+2} s_{n-1} s_{n-3} \dots s_{n-2k+1}) \times$$

$$s_n s_{n-2} \dots s_1 (\prod_{k=1}^{[\frac{n}{2}]} s_{n-1} s_{n-3} \dots s_{2k} s_n s_{n-2} \dots s_{2k+1}), \quad \text{if } n \text{ is odd,}$$

$$w^0 = (\prod_{k=1}^{[\frac{n}{2}]} s_n s_{n-2} \dots s_{n-2k+2} s_{n-1} s_{n-3} \dots s_{n-2k+1}) \times$$

$$(\prod_{k=1}^{[\frac{n}{2}]} s_{n-1} s_{n-3} \dots s_{2k} s_n s_{n-2} \dots s_{2k+1}) s_n, \quad \text{if } n \text{ is even.}$$

$$\begin{aligned} \mathbf{R}^+ = \{ & \alpha_n, \\ & \alpha_{n-1} + \alpha_n, \\ & 2\alpha_{n-1} + \alpha_n,\ \alpha_{n-2} + \alpha_{n-1} + \alpha_n, \\ & \alpha_{n-2} + 2\alpha_{n-1} + \alpha_n,\ \alpha_{n-3} + \alpha_{n-2} + \alpha_{n-1} + \alpha_n, \\ & \dots \\ & 2\alpha_1 + 2\alpha_2 + \dots + 2\alpha_{n-1} + \alpha_n, \\ & \alpha_{n-1},\ \alpha_{n-2} + \alpha_{n-1}, \dots,\ \alpha_1 + \dots \alpha_{n-1}, \\ & \alpha_{n-2},\ \alpha_{n-3} + \alpha_{n-2}, \dots,\ \alpha_1 + \dots \alpha_{n-2}, \\ & \dots \\ & \alpha_2,\ \alpha_1 + \alpha_2, \\ & \alpha_1 \} \end{aligned}$$

Relations

Case1. For any $k < l \leq n-1$, and $\beta = \alpha_{k+1} + \alpha_{k+2} + \dots + \alpha_l$, we have

$$E_{\alpha_k + \beta} = E_\beta E_k - v^{-2} E_k E_\beta.$$

Case2. For any $1 < k \leq n-1$, and $\beta = 2\alpha_k + 2\alpha_{k+1} + \ldots + 2\alpha_{n-1} + \alpha_n$, we have

$$E_{\beta+\alpha_{k-1}} = E_n E_\beta - v^{-2} E_\beta E_{k-1},$$
$$E_{\beta+2\alpha_{k-1}} = E_{k-1}^{(2)} E_\beta - v^{-1} E_{k-1} E_\beta E_{k-1} + v^{-2} E_\beta E_{k-1}^{(2)}.$$

(d) **Algebra type D_n**

$$\mathbf{C} = \begin{pmatrix} 2 & -1 & \ldots & 0 & 0 & 0 \\ -1 & 2 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 2 & -1 & -1 \\ 0 & 0 & \ldots & -1 & 2 & 0 \\ 0 & 0 & \ldots & -1 & 0 & 2 \end{pmatrix}$$

$$\begin{aligned} w^0 = {} & s_1 s_2 \ldots s_{n-1} s_n s_{n-2} \ldots s_2 s_1 \\ & s_2 s_3 \ldots s_{n-1} s_n s_{n-2} \ldots s_2 \\ & \ldots \\ & s_{n-2} s_{n-1} s_n s_{n-2} \\ & s_{n-1} s_n \end{aligned}$$

$$\begin{aligned} \mathbf{R}^+ = {} & \{\alpha_1,\ \alpha_1+\alpha_2, \ldots, \alpha_1+\ldots+\alpha_{n-1},\ \alpha_1+\ldots+\alpha_{n-2}+\alpha_n, \\ & \alpha_1+\ldots+\alpha_n,\ \alpha_1+\ldots+\alpha_{n-3}+2\alpha_{n-2}+\alpha_{n-1}+\alpha_n, \ldots, \\ & \alpha_1+2\alpha_2+\ldots+2\alpha_{n-2}+\alpha_{n-1}+\alpha_n, \\ & \alpha_2,\ \alpha_2+\alpha_3, \ldots, \alpha_2+\ldots+\alpha_{n-1},\ \alpha_2+\ldots+\alpha_{n-2}+\alpha_n, \\ & \alpha_2+\ldots+\alpha_n,\ \alpha_2+\ldots+\alpha_{n-3}+2\alpha_{n-2}+\alpha_{n-1}+\alpha_n, \ldots, \\ & \alpha_2+2\alpha_3\ldots+2\alpha_{n-2}+\alpha_{n-1}+\alpha_n, \\ & \ldots \\ & \alpha_{n-2},\ \alpha_{n-2}+\alpha_{n-1},\ \alpha_{n-2}+\alpha_n,\ \alpha_{n-2}+\alpha_{n-1}+\alpha_n, \\ & \alpha_{n-1}, \\ & \alpha_n\} \end{aligned}$$

Relations

Case1. For any $k < l \leq n$, and $\beta = \alpha_k + \alpha_{k+1} + \ldots + \alpha_{l-1}$, we have

$$E_{\beta+\alpha_l} = E_\beta E_l - v^{-1} E_l E_\beta.$$

Case2. For any $k \leq n-2$, and $\beta = \alpha_k + \alpha_{k+1} + \ldots + \alpha_{n-2}$, we have

$$E_{\beta+\alpha_n} = E_\beta E_n - v^{-1} E_n E_\beta.$$

Case2. For any $k < l-1 \leq n-2$, and $\beta = \alpha_k + \alpha_{k+1} + \ldots + \alpha_{l-1} + 2\alpha_l + \ldots 2\alpha_{n-2} + \alpha_{n-1} + \alpha_n$, we have

$$E_{\beta+\alpha_{l-1}} = E_\beta E_{l-1} - v^{-1} E_{l-1} E_\beta.$$

(f) **Algebra type F_4**

$$\mathbf{C} = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -2 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}$$

$$w^0 = s_1 s_2 s_3 (s_2 s_4)(s_1 s_3) s_2 (s_3 s_1)(s_4 s_2) s_3 s_2 s_1 s_2 s_3 (s_2 s_4) s_3 s_2 s_4 s_3 s_4$$

$$\begin{aligned} \mathbf{R}^+ = \{ & \alpha_1,\ \alpha_1+\alpha_2,\ \alpha_1+\alpha_2+\alpha_3,\ \alpha_1+\alpha_2+2\alpha_3,\ \alpha_1+\alpha_2+\alpha_3+\alpha_4, \\ & \alpha_1+2\alpha_2+2\alpha_3,\ \alpha_1+\alpha_2+2\alpha_3+\alpha_4,\ 2\alpha_1+3\alpha_2+4\alpha_3+2\alpha_4, \\ & \alpha_1+2\alpha_2+2\alpha_3+\alpha_4,\ \alpha_1+\alpha_2+2\alpha_3+2\alpha_4, \\ & \alpha_1+2\alpha_2+3\alpha_3+\alpha_4,\ \alpha_1+2\alpha_2+2\alpha_3+2\alpha_4, \\ & \alpha_1+2\alpha_2+3\alpha_3+2\alpha_4,\ \alpha_1+2\alpha_2+4\alpha_3+2\alpha_4, \\ & \alpha_1+3\alpha_2+4\alpha_3+2\alpha_4, \\ & \alpha_2,\ \alpha_2+\alpha_3,\ \alpha_2+2\alpha_3,\ \alpha_2+\alpha_3+\alpha_4, \\ & \alpha_2+2\alpha_3+\alpha_4,\ \alpha_2+2\alpha_3+2\alpha_4, \\ & \alpha_3,\ \alpha_3+\alpha_4, \\ & \alpha_4\} \end{aligned}$$

Relations:

Case1. If β is one of the roots α_1, $\alpha_1+\alpha_2+2\alpha_3$, or $\alpha_1+2\alpha_2+4\alpha_3+2\alpha_4$, we have:

$$E_{\beta+\alpha_2} = E_\beta E_2 - v^{-2} E_2 E_\beta.$$

Case2. If $\beta = \alpha_1 + 2\alpha_2 + 2\alpha_3$, and $\beta' = \alpha_1 + \alpha_2 + 2\alpha_3 + 2\alpha_4$, we have:

$$E_{\beta+\beta'} = E_\beta E_{\beta'} - v^{-2} E_{\beta'} E_\beta.$$

Case3. If β is one of the roots α_3, or $\alpha_2+\alpha_3$, and $j=4$, or if $\beta=\alpha_1+2\alpha_2+2\alpha_3+\alpha_4$, and $j=3$, we have

$$E_{\beta+\alpha_j}=E_\beta E_j-v^{-1}E_jE_\beta.$$

Case4. If β is one of the roots $\alpha_1+\alpha_2+\alpha_3$, $\alpha_1+2\alpha_2+2\alpha_3$, or $\alpha_2+2\alpha_3$, and $j=4$, or if β is one of the roots $\alpha_1+2\alpha_2+2\alpha_3+2\alpha_4$, or α_2, and $j=3$, we have

$$\begin{aligned}E_{\beta+\alpha_j}&=E_\beta E_j-v^{-2}E_jE_\beta,\\ E_{\beta+2\alpha_j}&=E_j^{(2)}E_\beta-v^{-1}E_jE_\beta E_j+v^{-2}E_\beta E_j^{(2)}\end{aligned}$$

(g) **Algebra type G_2**

$$\begin{aligned}\mathbf{C}&=\begin{pmatrix}2&-1\\-3&2\end{pmatrix}\\ w^0&=s_2s_1s_2s_1s_2\\ \mathbf{R}^+&=\{\alpha_2,\ \alpha_1+\alpha_2,\ 3\alpha_1+2\alpha_2,\ 2\alpha_1+\alpha_2,\ 3\alpha_1+\alpha_2,\ \alpha_1\}\end{aligned}$$

Relations:

$$\begin{aligned}E_{\alpha_1+\alpha_2}&=E_1E_2-v^3E_2E_1,\\ E_{3\alpha_1+2\alpha_2}&=E_1^{(3)}E_2^{(2)}+v^{-6}E_2^{(2)}E_1^{(3)}+v^{-2}E_1^{(2)}E_2^{(2)}E_1+v^{-4}E_1E_2^{(2)}E_1^{(2)}+\\ &\quad v^{-1}e_1E_2E_1^{(2)}E_2+v^{-3}(v+v^{-1})e_2E_1^{(3)}E_2+v^{-5}E_2E_1^{(2)}E_2E_1,\\ E_{2\alpha_1+2\alpha_2}&=E_1^{(2)}E_2-v^{-2}E_1E_2E_1+v^{-4}E_2E_1^{(2)},\\ E_{3\alpha_1+\alpha_2}&=-E_1^{(3)}E_2+v^{-1}E_1^{(2)}E_2E_1-v^{-2}E_1E_2E_1^{(2)}+v^{-3}E_2E_1^{(3)}.\end{aligned}$$

In principle, in the same way one can work out the relations for the algebra type E (see [24]), but algebras of so high rank are currently out of the reach of these algorithms since they require too much time and memory. For this reason we left the algebra type E out.

3.1.5 $\triangle:{}_{\mathcal{R}'}\mathbf{u}\to{}_{\mathcal{R}'}\mathbf{U}\otimes{}_{\mathcal{R}'}\mathbf{U}$, $S:{}_{\mathcal{R}'}\mathbf{U}\to{}_{\mathcal{R}'}\mathbf{U}^{opp}$, and $e:{}_{\mathcal{R}'}\mathbf{U}\to\mathcal{R}'$ can be defined as in 2.1.3 , and in this way ${}_{\mathcal{R}'}\mathbf{U}$ becomes a Hopf algebra.

3.2 Monoidal braided structure on the category of integrable ${}_{\mathcal{R}'}\mathbf{U}$ -modules

Most of the results in this section are proved in [26]. In particular, the expression for the commutativity morphism can be found in [26, 32.1] given in terms of the canonical basis. The same expression given in term of Lusztig's basis appears in [36], but there the value of the ribbon element is not given explicitly.

3.2.1 An ${}_{\mathcal{R}'}\mathbf{U}$-module M is integrable if M is a finite dimensional $\mathcal{R}'$-module and if there is a direct sum decomposition $M = \oplus_{\lambda \in X} M^\lambda$ such that, for any $i \leq n$ and $m \in M^\lambda$,

$$\tilde{K}_i m = v_i^{\langle h_i, \lambda \rangle} m, \text{ and } \begin{bmatrix} \tilde{K}_i; 0 \\ t \end{bmatrix} m = \Phi(\begin{bmatrix} \langle h_i, \lambda \rangle \\ t \end{bmatrix}_i) m.$$

As before the M^λ are called the weight spaces of M, and their dimensions are called multiplicities. Let $\mathcal{C}_p'$ be the category with integrable ${}_{\mathcal{R}'}\mathbf{U}$-modules as objects and morphisms ${}_{\mathcal{R}'}\mathbf{U}$-linear maps.

An object M in $\mathcal{C}_p'$ is a highest weight module with highest weight λ if there exists a vector $m \in M^\lambda$ such that m generates M^λ as $\mathcal{R}'$-module and

(a) $E_i^{(r)} m = 0$ for any $i \leq n$ and $r > 0$.

(b) $M = {}_{\mathcal{R}'}\mathbf{U}^- m$.

3.2.2 With the same argument as in 2.3.2, we can show that if $M \in \mathcal{C}_p'$, there exists $N \geq 1$ such that for any $m \in M$ and any $x \in \mathcal{R}' \otimes {}_{\mathcal{A}}\mathbf{f}$ with $lev(|x|) \geq N$, $x^+ m = 0$, and $x^- m = 0$.

3.2.3 Note that if $M \in \mathcal{C}$, and ${}_{\mathcal{A}}M$ is an ${}_{\mathcal{A}}\mathbf{U}$ invariant lattice of M, ${}_{\mathcal{R}'}M = \mathcal{R}' \otimes ({}_{\mathcal{A}}M) \in \mathcal{C}_p'$, and since $\mathcal{R}'$ is a field, different lattices yield isomorphic representations in $\mathcal{C}_p'$. Moreover, the modules in $\mathcal{C}_p'$ satisfy the following properties (see [23, 26, 1]):

(a) Each object in $\mathcal{C}_p'$ is sum of subobjects each isomorphic to a quotient object of ${}_{\mathcal{R}'}L_\lambda \otimes {}_{\mathcal{R}'}L_{\lambda'}$ for some $\lambda, \lambda' \in \mathcal{W}$.

(b) For any $\lambda \in \mathcal{W}$, there exists a simple object $\tilde{L}_\lambda \in \mathcal{C}_p'$, unique up to an isomorphism, which is a highest weigh module of highest weight λ. This object is quotient object of ${}_{\mathcal{R}'}L_\lambda$.

(c) If $\lambda \neq \lambda'$, $\tilde{L}_\lambda$ is not isomorphic to $\tilde{L}_{\lambda'}$.

(d) Let $\rho = 1/2 \sum_{\beta \in \mathbf{R}^+} \beta$, and

$$\triangle = \{\lambda \in \mathcal{W} \mid \langle h_\beta, \lambda + \rho \rangle < p \text{ for any } \beta \in \mathbf{R}^+\}.$$

Then if $\lambda \in \triangle$, $\tilde{L}_\lambda$ is isomorphic to ${}_{\mathcal{R}'}L_\lambda$.

3.2.4 Define the *affine Weyl group* $\mathbf{W}_p$ to be the group generated by the affine reflections: $\sigma_{\beta,k} : X \to X, \beta \in \mathbf{R}^+$, $k \in \mathbf{Z}$, given by

$$\sigma_{\beta,k}(\lambda) = \lambda - (\langle \beta, \lambda \rangle - kp)\beta.$$

Define also the dot action $w.\lambda$ of $\mathbf{W}_p$ on X to be

$$w.\lambda = w(\lambda + \rho) - \rho.$$

Then the different weights in $\triangle$ lie on different orbits of the dot action of $\mathbf{W}_p$ on X. Also, there is the following description of the objects in $\mathcal{C}_p'$ (see [1]). Let $\mathcal{O}$ denote the set of all orbits of $\mathbf{W}_p$ in $\mathcal{W}$. then any $M \in \mathcal{C}_p'$ is isomorphic to a direct sum $\oplus_{O \in \mathcal{O}} M_O$, where all simple subquotients of M_O are of the form $\tilde{L}_\lambda$, $\lambda \in O$.

3.2.5 An associative tensor product of two objects M' and M'' in $\mathcal{C}_p'$ is defined as in 2.3.4. The resulting module is in $\mathcal{C}_p'$, and

$$(M' \otimes M'')^\lambda = \sum_{\lambda' + \lambda'' = \lambda} M^{\lambda'} \otimes M''^{\lambda''}.$$

3.2.6 Given an ${}_{\mathcal{R}'}\mathbf{U}$-module M we define its dual M^* to be the ${}_{\mathcal{R}'}\mathbf{U}$-module with vector space $M^* = Hom(M, \mathcal{R}')$ and the following ${}_{\mathcal{R}'}\mathbf{U}$-structure: for any $u \in {}_{\mathcal{R}'}\mathbf{U}$, $\quad y \in M^*$, and $m \in M$

$$u(y)(m) = y(S(u)x).$$

Since the antipode is an algebra anti-homomorphism, this is well defined.

If M is an element in $\mathcal{C}_p'$, M^* is again in $\mathcal{C}_p'$, and

$$(M^*)^\lambda = Hom(M^{-\lambda}, \mathcal{R}').$$

3.2.7 From the invariance of the "."-form under the Weyl group (2.4.1) and the fact that $\mathbf{W}$ sends the set of positive roots into itself, we conclude that if $\lambda \in \triangle$ then $\hat{\lambda} \in \triangle$. Moreover, in this case, we have that $\mathcal{R}' \otimes_{\mathcal{A}} ({}_{\mathcal{A}}L_{\lambda}{}^{*}) = \mathcal{R}' \otimes_{\mathcal{A}} Hom({}_{\mathcal{A}}L_{\lambda}, \mathcal{A}) \simeq Hom(\mathcal{R}' \otimes_{\mathcal{A}} ({}_{\mathcal{A}}L_{\lambda}), \mathcal{R}' \otimes_{\mathcal{A}} \mathcal{A}) = Hom(\tilde{L}_{\lambda}, \mathcal{R}')$. Using this isomorphism, $1 \otimes_{\mathcal{A}} \vartheta_{\lambda} : \tilde{L}_{\lambda} \to \tilde{L}_{\hat{\lambda}}{}^{*}$ defines an isomorphism of $\mathcal{R}'$-modules.

3.2.8 Let $\varsigma \in \mathbf{Z}$ be such that $D = det(\mathbf{C})\varsigma = 1 \ (mod\ p)$, and let $f : X \times X \to \mathbf{Z}$ be given by $f(\lambda, \lambda') = -D(\lambda.\lambda')$. Then given any two objects M, M' in $\mathcal{C}_p{}'$ define an invertible linear operator $\Pi_f : M \otimes M' \to M \otimes M'$ given by $\Pi_f(m \otimes m') = v^{f(\lambda,\lambda')} m \otimes m'$, for $m \in M^{\lambda}$, and $m' \in M'^{\lambda'}$. Define also an $\mathcal{R}'$-linear map $\Theta : M \otimes M' \to M \otimes M'$ given by

$$\Theta(m \otimes m') = \sum_{\mathbf{c} \in \mathbf{Z}_p^{t0}} (-1)^{lev(|b_{\mathbf{c}}|)} \Phi(v_{|b_{\mathbf{c}}|} h_{\mathbf{c}}) b_{\mathbf{c}}^{-} m \otimes b_{\mathbf{c}}^{+} m'.$$

Here $h_{\mathbf{c}} = 1 \otimes_{\mathcal{A}} (b_{\mathbf{c}}, b_{\mathbf{c}})^{-1} \in \mathcal{R}$. From 2.7.2 follows that Θ is invertible with inverse the $\mathcal{R}'$-linear operator given by:

$$\overline{\Theta}(m \otimes m') = \sum_{\mathbf{c} \in \mathbf{Z}_p^{t0}} (-1)^{lev(|b_{\mathbf{c}}|)} \Phi(v_{-|b_{\mathbf{c}}|} \overline{h}_{\mathbf{c}}) \overline{b}_{\mathbf{c}}^{-} m \otimes \overline{b}_{\mathbf{c}}^{+} m',$$

where $\overline{b}_{\mathbf{c}}^{\pm}$ denote the elements $1 \otimes_{\mathcal{A}} \overline{b}_{\mathbf{c}} \in {}_{\mathcal{R}}\mathbf{U}$, and $\overline{h}_{\mathbf{c}} = 1 \otimes_{\mathcal{A}} \overline{(b_{\mathbf{c}}, b_{\mathbf{c}})^{-1}} \in \mathcal{R}$.

3.2.9 Let $M, M' \in \mathcal{C}_p{}'$, and let $R_{M,M'} = \Theta \Pi_f s : M \otimes M' \to M' \otimes M$. Then

(a) $R_{M,M'}$ is an isomorphism in $\mathcal{C}_p{}'$ with inverse given by $s^{-1}(\Pi_f)^{-1}\overline{\Theta}$.

(b) $R_{(M \otimes M'),M''} = (R_{M,M''} \otimes id_{M'})(id_M \otimes R_{M',M''}) : M \otimes M' \otimes M'' \to M'' \otimes M' \otimes M'$, and $R_{M'',(M \otimes M')} = (id_M \otimes R_{M'',M'})(R_{M'',M} \otimes id_{M'}) : M'' \otimes M \otimes M' \to M \otimes M' \otimes M''$.

The properties in (b) are knows as the hexagon diagrams.

3.2.10 Let $M, M', M_1, M_1' \in \mathcal{C}_p{}'$, and $\phi : M \to M_1$ and $\phi' : M' \to M_1'$ be homomorphisms. Then from the explicit expression for $R_{M,M'}$ we conclude that

$$R_{M_1,M_1'}(\phi \otimes \phi') = (\phi' \otimes \phi) R_{M,M'}.$$

3.3 The quantum Casimir operator

For the algebra $\mathbf{U}$ the quantum Casimir operator was defined by Drinfeld ([5]). The exposition here is adaptation of the one in Lusztig ([26, 6.1]) to the case when v is a root of unity.

3.3.1 Given any function $G : X \to \mathbf{Z}$, and an object $M \in \mathcal{C}_p'$, define a linear operator $\hat{v}^G : M \to M$ given by $\hat{v}^G(m) = v^{G(\lambda)}$ for any $m \in M^\lambda$. Let D be as in 3.2.8. Then the functions on X $G'(\lambda) = -(\lambda.\lambda + \sum_{i=1}^n \lambda_i i.i)D$ and $G''(\lambda) = G'(-\lambda)$ have values in $\mathbf{Z}$, and therefore the operators $\hat{v}^{G'}$ and $\hat{v}^{G''}$ are well defined.

3.3.2 Given any object $M \in \mathcal{C}_p'$, define the linear operators Ω', $\Omega'' : M \to M$ in the following way. For any $m \in M$

$$\Omega'(m) = \sum_{\mathbf{c} \in \mathbf{N}^{t^0}} (-1)^{lev(|b_\mathbf{c}|)} \Phi(v_{-|b_\mathbf{c}|} h_\mathbf{c}) \tilde{K}_{-|b_\mathbf{c}|} b_\mathbf{c}^- b_\mathbf{c}^{*+} m,$$

$$\Omega''(m) = \sum_{\mathbf{c} \in \mathbf{N}^{t^0}} (-1)^{lev(|b_\mathbf{c}|)} \Phi(v_{-|b_\mathbf{c}|} h_\mathbf{c}) \tilde{K}_{|b_\mathbf{c}|} b_\mathbf{c}^{*+} b_\mathbf{c}^- m.$$

Note that $\Phi(h_\mathbf{c})$ is nonzero only for $\mathbf{c} \in \mathbf{Z}_p$, i.e. the sums above are finite.

Proposition 3.3.3 *Let $M \in \mathcal{C}_p'$. Assume that all weights of M are contained in the same coset C of X with respect to the $\mathbf{Z}$-lattice spanned by the vectors in $\mathbf{\Pi}$. Then, as maps on M, $\Omega'\hat{v}^{G'}$ and $\Omega''\hat{v}^{G''}$ are isomorphism in $\mathcal{C}_p'$. Moreover, if $M = \tilde{L}_\lambda$, $\Omega'\hat{v}^{G'} = \Omega''\hat{v}^{G''}$ acts as $v^{G'(\lambda)}$ times the identity.*

We will present here the proof only for $\Omega'\hat{v}^{G'}$, since the argument for $\Omega''\hat{v}^{G''}$ is analogous. Choose a $\lambda_0 \in C$, and set $G_0 : C \to \mathbf{Z}$ to be the function given by $G_0(\lambda) = -\lambda.\lambda + \lambda_0.\lambda_0 - \sum_{i=1}^n \lambda_i i.i$. Then $\hat{v}^{G'} = v^{-D\lambda_0.\lambda_0}\hat{v}^{G_0}$. Hence it is enough to prove the statement for the operator $\Omega'\hat{v}^{G_0}$. Because any object in $\mathcal{C}_p'$ is isomorphic to sum of quotient objects of ${}_{\mathcal{R}'}L_\lambda \otimes {}_{\mathcal{R}'}L_{\lambda'}$ for some $\lambda, \lambda' \in \mathcal{W}$, it is enough to prove that for $\lambda + \lambda' \in C$, $\Omega'\hat{v}^{G_0}$ (defined as in 2.8.1) are in the commutant of the ${}_{\mathcal{A}}\mathbf{U}$-modules ${}_{\mathcal{A}}L_\lambda \otimes {}_{\mathcal{A}}L_{\lambda'}$. Moreover, it is enough to prove this for the $\mathbf{U}$-modules $L_\lambda \otimes L_{\lambda'}$. Let m be in the μ weight space of the corresponding $\mathbf{U}$-module. Then using 2.8.1 (b) we

deduce:

$$\begin{aligned} E_i\Omega'\hat{v}^{G_0}m &= v^{G_0(\mu)}E_i\Omega'm \\ &= v^{G_0(\mu)}\tilde{K}_{-i}{}^2\Omega'E_im == v^{G_0(\mu+\alpha_i)}\Omega'E_im \\ &= \Omega'\hat{v}^{G_0}E_im, \text{ and} \\ F_i\Omega'\hat{v}^{G_0}m &= v^{G_0(\mu)}F_i\Omega'm \\ &= v^{G_0(\mu)}\tilde{K}_i{}^2\Omega'F_im == v^{G_0(\mu-\alpha_i)}\Omega'F_im \\ &= \Omega'\hat{v}^{G_0}F_im \end{aligned}$$

This proves that $\Omega'\hat{v}^{G'}$ defines an automorphism of M. When $M = \tilde{L}_\lambda$, $\Omega'\hat{v}^{G'}$ acts on the λ weight space as multiplication by $v^{G'(\lambda)}$. But $M \subset {}_{\mathcal{R}'}\mathbf{U}^-M^\lambda$. Hence the automorphism group of M is isomorphic to $\mathcal{R}'$, and $\Omega'\hat{v}^{G'}$ acts on M as $v^{G'(\lambda)}$ times the identity.

3.3.4 The operator $\Omega'\hat{v}^{G'} : M \to M$ is called the *quantum Casimir operator.*

3.4 Monoidal braided structure on the category of small representations

We introduce few definitions and the construction of the quotient category $\overline{\mathcal{C}}_p$ of $\mathcal{C}_p{}'$ from [10].

3.4.1 Let C be a category and $\diamond : \mathsf{C} \times \mathsf{C} \to \mathsf{C}$, $(a, b) \to a \diamond b$, be a functor. We say that C is monoidal braided category if the following conditions are satisfied:

(a) For any three object a, b, c there exists a functorial isomorphism

$$\phi_{a,b,c} : (a \diamond b) \diamond c \to a \diamond (b \diamond c)$$

such that, for all objects a, b, c, d the diagram

$$\begin{array}{ccccc} a \diamond (b \diamond (c \diamond d)) & \xleftarrow{\phi} & (a \diamond b) \diamond (c \diamond d) & \xleftarrow{\phi} & ((a \diamond b) \diamond c) \diamond d \\ \uparrow{\scriptstyle id_a \diamond \phi} & & & & \downarrow{\scriptstyle \phi \diamond id_d} \\ a \diamond ((b \diamond c) \diamond d) & & \xleftarrow{\phi} & & (a \diamond (b \diamond c)) \diamond d \end{array}$$

commutes. The ϕ's are called *associativity morphisms* and the diagram above is called the *pentagon diagram.*

(b) For any two objects a, b there is a functorial isomorphism $\psi_{a,b}$ compatible with the associativity morphisms, i.e. for all objects a, b, c the diagram

$$\begin{array}{ccccc} a \diamond (b \diamond c) & \xleftarrow{\phi} & (a \diamond b) \diamond c & \xleftarrow{\psi} & c \diamond (a \diamond b) \\ \uparrow{\scriptstyle id_a \diamond \psi} & & & & \uparrow{\scriptstyle \phi} \\ a \diamond (c \diamond b) & \xleftarrow{\phi} & (a \diamond c) \diamond b & \xleftarrow{\psi \diamond id_b} & (c \diamond a) \diamond b \end{array}$$

commutes. The ψ's are called *commutativity morphisms* and the diagram above is called the *hexagon diagram.*

From 3.2.9 follows that $\mathcal{C}_p{}'$ is a monoidal braided category.

3.4.2 Let $\Sigma = \{\check{L}_\lambda \mid \lambda \in \triangle\}$ (see 3.2.3). From the invariance of the "."-form under the Weyl group (2.4.1), and the fact that $\mathbf{W}$ sends the set of positive roots into itself we conclude that $\check{L}_{\hat{\lambda}} \in \Sigma$. Hence $\hat{}: \Sigma \to \Sigma$, $\check{L}_\lambda \to \check{L}_{\hat{\lambda}}$ defines an involution on Σ. To simplify the notation from now on we will denote the elements of Σ with a, b, $c, \ldots$, and their dual modules with a^*, b^*, $c^*, \ldots$. Then the involution above will be denoted by $a \to \hat{a}$, where $a = \check{L}_\lambda$ and $\hat{a} = \check{L}_{\hat{\lambda}}$. The trivial module in Σ will be denoted with $\mathbf{1}$.

3.4.3 Define $\mathcal{C}_p$ to be the full additive subcategory of $\mathcal{C}_p{}'$ generated by Σ, i.e. the objects in $\mathcal{C}_p$ are all finite direct sums of objects in Σ, and the set of homomorphisms between two such objects is the same as the set of homomorphisms between them in $\mathcal{C}_p{}'$.

The following results are due to Gelfand and Kazhdan ([10]).

3.4.4 Let p be greater than the Coxeter number of the corresponding Lie-algebra. Then there exists a full additive subcategory $\mathcal{C}_p{}^\perp$ of $\mathcal{C}_p{}'$ such that

(a) For any $M \in \mathcal{C}_p{}^\perp$ and $A, B \in \mathcal{C}_p$, any composite morphism $A \to M \to B$ is zero.

(b) $\mathcal{C}_p{}^\perp$ is invariant under tensor product, and if $A \in \mathcal{C}_p$ and $M \in \mathcal{C}_p{}^\perp$, $A \otimes M \in \mathcal{C}_p{}^\perp$.

(c) If $A, B \in \mathcal{C}_p$, $A \otimes B \simeq C \oplus M$, where $A \in \mathcal{C}_p$ and $M \in \mathcal{C}_p{}^\perp$.

3.4.5 Let $\tilde{\mathcal{C}}_p$ be the full additive subcategory of $\mathcal{C}_p'$ generated by all tensor products of elements in $\mathcal{C}_p$ and $\mathcal{C}_p{}^{\perp}$. The results above state that any object in $\tilde{\mathcal{C}}_p$ is isomorphic to a direct sum of objects in $\mathcal{C}_p$ and $\mathcal{C}_p{}^{\perp}$. Given any $W_1, W_2 \in \tilde{\mathcal{C}}_p$, let $hom^0(W_1, W_2) = \{f \in hom(W_1, W_2) \mid \exists M \in \mathcal{C}_p{}^{\perp}, f_1 : W_1 \to M$ and $f_2 : M \to W_2$ with $f_2 \circ f_1 = f\}$. Then define the quotient category $\overline{\mathcal{C}}_p$ of $\tilde{\mathcal{C}}_p$ by $\mathcal{C}_p$ in the following way. The objects in $\overline{\mathcal{C}}_p$ are the same as the objects in $\tilde{\mathcal{C}}_p$, and $hom_{\overline{\mathcal{C}}_p}(W_1, W_2) = hom(W_1, W_2)/hom^0(W_1, W_2)$. Since $hom^0(W_1, W_2)$ is an ideal of $hom(W_1, W_2)$ with respect to composition, the composition in $\overline{\mathcal{C}}_p$ is well defined. Note also that the identity morphisms in $\overline{\mathcal{C}}_p$ for $M \in \mathcal{C}_p{}^{\perp}$ are zero.

Let $P : \tilde{\mathcal{C}}_p \to \overline{\mathcal{C}}_p$ be the projection. Then define the associativity and commutativity morphisms in $\overline{\mathcal{C}}_p$ to be the images of the corresponding morphisms in $\tilde{\mathcal{C}}_p$ under P. Because of property (b) above this is well defined, and thus $\overline{\mathcal{C}}_p$ becomes a monoidal braided category.

3.4.6 Given any $W \in \overline{\mathcal{C}}_p$, we have that

(a) In $\overline{\mathcal{C}}_p$ W is isomorphic to $A \in \mathcal{C}_p$ if and only if there is a split injection $A \ominus M \xrightarrow{\imath} W \xrightarrow{\pi} A \oplus M$.

(b) For any $B \in \mathcal{C}_p$ and A as in (a), the map $hom(B, A) \to hom_{\overline{\mathcal{C}}_p}(B, W)$, given by $f \to \imath \circ f$, is an isomorphism.

Let $J : \mathcal{C}_p \to \tilde{\mathcal{C}}_p$ be the inclusion map, and let $F = P \circ J$. Then properties (a) and (b) imply that F is category equivalence. Then (see [34]) there exists a monoidal braided structure on $\mathcal{C}_p$. We now proceed with the explicit description of this structure.

3.4.7 Given any $W \in \Sigma$, let $A = \oplus_k a_k$, $a_k \in \Sigma$, be isomorphic to W in $\overline{\mathcal{C}}_p$. Then from (a) above it follows that there exist homomorphisms $a_k \xrightarrow{\imath_k} W \xrightarrow{\pi_k} a_k$ in $\mathcal{C}_p'$ such that $\pi_k \circ \imath_l = \delta_{k,l} id_{a_k}$. Given any $a \in \Sigma$, it is easy to see that $\{\imath_k\}_{a_k = a}$ form a basis for $hom_{\overline{\mathcal{C}}_p}(a, W)$. Hence the map $g : A \to \oplus_{a \in \Sigma} hom_{\overline{\mathcal{C}}_p}(a, W) \otimes_{\mathcal{R}'} a$, $a_k \to \imath_k \otimes a_k$ is an isomorphism, and the following diagram

$$\begin{array}{ccccc} A & \xrightarrow{\oplus_k \imath_k} & W & \xrightarrow{\sum_k \pi_k} & A \\ \downarrow g & & \downarrow id_W & & \downarrow g \\ \oplus_{a \in \Sigma} hom_{\overline{\mathcal{C}}_p}(a, W) \otimes_{\mathcal{R}'} a & \xrightarrow{\imath[W]} & W & \xrightarrow{\pi[W]} & \oplus_{a \in \Sigma} hom_{\overline{\mathcal{C}}_p}(a, W) \otimes_{\mathcal{R}'} a \end{array}.$$

commutes. Here $\imath[W](f\otimes x) = f(x)$, $f \in hom_{\overline{\mathcal{C}}_p}(a, W)$, $x \in a$, and $\pi[W](y) = \ominus(\imath_k \otimes \pi_k(y))$, $y \in W$. Obviously, $\pi[W] \circ \imath[W]$ is the identity, and the definition of these homomorphisms is independent of the choice of bases.

3.4.8 Now define a product $\diamond : \mathcal{C}_p \times \mathcal{C}_p \to \mathcal{C}_p$ in the following way. For $b, c, b', c' \in \Sigma$ and $f \in hom(b, c)$, $f' \in hom(b', c')$, $b \times c \to \oplus_{a\in\Sigma} hom_{\overline{\mathcal{C}}_p}(a, W) \otimes_{\mathcal{R}'} a$, and $f \times f' \to \pi[b' \otimes c'] \circ (f \otimes f') \circ \imath[b \otimes c]$. We then extend this product to any two objects in $\mathcal{C}_p$ by linearity (using fixed split injections to their simple components).

3.4.9 For any $b, c, d \in \Sigma$ the associativity morphisms $\phi_{b,c,d}$ are defined by the following commutative diagram:

$$\begin{array}{ccc}
(b \diamond c) \diamond d & \xrightarrow{\phi_{b,c,d}} & b \diamond (c \diamond d) \\
\downarrow{\scriptstyle \imath[(b\diamond c)\otimes d]} & & \uparrow{\scriptstyle \pi[b\otimes(c\diamond d)]} \\
(b \diamond c) \otimes d & & b \diamond (c \otimes d) \\
\downarrow{\scriptstyle \imath[b\otimes c]\otimes id_d} & & \uparrow{\scriptstyle id_b\otimes\pi[c\otimes d]} \\
(b \otimes c) \otimes d & \xrightarrow{\simeq} & b \otimes (c \otimes d)
\end{array}$$

Note that $\phi_{b,c,d} = \oplus_{a\in\Sigma} \mathtt{asoc}[a, b, c, d] \otimes id_a$, where $\mathtt{asoc}[a, b, c, d] : hom_{\overline{\mathcal{C}}_p}(a, (b \diamond c) \otimes d) \to hom_{\overline{\mathcal{C}}_p}(a, b \otimes (c \diamond d))$. For any other objects $B, C, D \in \mathcal{C}_p$, $\phi_{B,C,D}$ is defined by linearity.

3.4.10 For any $b, c \in \Sigma$ the commutativity morphisms $\psi_{b,c}$ are defined by the following commutative diagram:

$$\begin{array}{ccc}
b \diamond c & \xrightarrow{\psi_{b,c}} & c \diamond b \\
\downarrow{\scriptstyle \imath[b\otimes c]} & & \uparrow{\scriptstyle \pi[c\otimes b]} \\
b \otimes c & \xrightarrow{R_{b,c}} & c \otimes b
\end{array}$$

Note that $\psi_{b,c} = \oplus_{a\in\Sigma} \mathtt{com}[a, b, c] \otimes id_a$, where $\mathtt{com}[a, b, c] : hom_{\overline{\mathcal{C}}_p}(a, b \otimes c) \to hom_{\overline{\mathcal{C}}_p}(a, b \otimes c)$. For any other objects $B, C, D \in \mathcal{C}_p$, $\psi_{B,C}$ is defined by linearity. The ϕ's and ψ's define a monoidal braided structure on $\mathcal{C}_p$, i.e. the pentagon and hexagon diagrams as in 3.4.1 commute.

3.4.11 We call $\mathtt{asoc}[a, b, c, d]$ and $\mathtt{com}[a, b, c]$, $a, b, c, d \in \Sigma$, elementary associativity and commutativity morphisms. The goal of this work is to develop

programs which compute the values of these morphisms. The strategy is first, for any $a, b, c \in \Sigma$ to find bases $\{\imath_k(a, b \otimes c)\}_k$ of $hom_{\overline{\mathcal{C}_p}}(a, b \otimes c)$ and their splittings $\{\pi_k(b \otimes c, a)\}_k$ (see in 3.4.7); then to compute the matrices describing $\mathtt{asoc}[a, b, c, d]$ and $\mathtt{com}[a, b, c]$ with respect to these bases using the definitions above.

3.4.12 The category $\mathcal{C}_p$ is called *tortile*[2] if for any pair $(a, \hat{a})$ it is equipped with duality morphisms $\vartheta_a : a \to \hat{a}^*$ and $\vartheta_{\hat{a}} : \hat{a} \to a^*$ such that $\vartheta_a = \vartheta_{\hat{a}}^* \circ \zeta_a \circ \hat{v}^\rho$. Here, $\zeta_a : a \to a^{**}$ is the standard $\mathcal{R}'$-isomorphism and ρ is as in 2.6.3. From 2.6.4, it follows that such morphisms in $\mathcal{C}_p$ can be found if and only if all simple selfdual objects in it are symmetric. In case $\mathcal{C}_p$ doesn't satisfy this condition, there is at least two subcategories ${\mathcal{C}_p}^0 \subset {\mathcal{C}_p}^{tor} \subset \mathcal{C}_p$ which do. Let $\Sigma_0 = \{\check{L}_\lambda \in \Sigma \,|\, \lambda \in X^+\}$, and $\Sigma_{tor} = \{\check{L}_\lambda \in \Sigma \,|\, lev(\lambda) \in \mathbf{N}\}$. Then ${\mathcal{C}_p}^0$ (resp. ${\mathcal{C}_p}^{tor}$) is the full additive subcategory of $\mathcal{C}_p$ generated by the objects in Σ_0 (resp. Σ_{tor}). Note that each of these subcategories is invariant under the taking products, and hence, the monoidal braided structure on $\mathcal{C}_p$ induces one on ${\mathcal{C}_p}^0$ and ${\mathcal{C}_p}^{tor}$. These two categories can always be equipped with proper duality morphisms and so become tortile.

In the rest of the paper we assume that we work with a tortile category, and denote it again with $\mathcal{C}_p$, though by necessity it might be its to ${\mathcal{C}_p}^{tor}$ or ${\mathcal{C}_p}^0$.

[2]For the general definition of a tortile category see [37]

Chapter 4
Algebraic structures and link diagrams

We introduce some algebraic structures and identities which are the foundations of describing the bases for the spaces $hom_{\overline{\mathcal{C}}_p}(a, b \otimes c)$. Everything will be expressed in terms of labeled link diagrams with coupons. Note that we use these diagrams as a graphic notation for some algebraic operations. Behind those, there is a much deeper fact, i.e. the existence of a category equivalence between a free generated tortile category, and the category of labeled double tangles (see [37, 32, 33]) , but introducing it in all generality would be out of the scope of this work.

4.1 Labeled link diagrams with coupons

In this section we will denote with capital letters $A, B, \ldots$ tensor products of finite number of objects in Σ. As before, the objects in Σ will be denoted with $a, b, \ldots$.

4.1.1 A labeled link diagram with coupons is a composition of $\mathcal{C}_p'$-morphisms $\varphi_k \circ \varphi_{k-1} \ldots \circ \varphi_1$, where

$$\varphi_i : A_1^i \otimes \ldots \otimes A_{l_i}^i \to A_1^{i+1} \otimes \ldots \otimes A_{l_{i+1}}^{i+1}.$$

To each morphism $\varphi_i : A_1 \otimes \ldots \otimes A_l \to B_1 \otimes \ldots \otimes B_k$ we correspond a "coupon" with labeled "lines" coming out of it and to the composition of morphisms the composition of their coupons as shown on Figure 1. Then some algebraic identities can be paraphrased in terms of these diagrams in the following way. First, if the identity morphism appears somewhere, its coupon can obviously be removed, and the top and bottom ends connected with straight lines. Also, if $\varphi = \varphi^1 \otimes \varphi^2$, the identity: $\varphi = \varphi^1 \otimes \varphi^2 = (\varphi^1 \otimes id)(id \otimes \varphi^2) = (id \otimes \varphi^2)(\varphi^1 \otimes id)$ means that we can move two parallel coupons vertically up and down with respect to each other.

4.1.2 Let $R_{AB} : A \otimes B \to B \otimes A$, and $R_{AB}^{-1} : A \otimes B \to B \otimes A$ be the commutativity morphism and its inverse. The special notations for the coupons corresponding to these morphisms are shown in Figure 2.

(a) Single morphism. (b) Composition of morphisms.

Figure 1: Labeled link diagrams with coupons.

R_{AB} R_{AB}^{-1}

Figure 2: Coupons for the commutativity morphisms.

The frames of these coupons will be omitted. Then the identities: $R_{AB}R_{BA}^{-1} = id$, $R_{BA}^{-1}R_{AB} = id$, and the naturality of R can be expressed in terms of diagrams as shown in Figures 3.

The hexagon diagrams (3.2.9) then correspond to the diagram identities in Figure 4.

4.1.3 Given $A \in S$, denote with $\eta_A : \mathbf{1} \otimes A \to A$, and ${}_A\eta : A \otimes \mathbf{1} \to A$ the standard isomorphisms. Since ${}_B\eta \otimes id_A = id_B \otimes \eta_A : B \otimes \mathbf{1} \otimes A \to b \otimes a$, there is a standard isomorphism $\eta : a_1 \otimes \ldots \otimes a_k \to a_{i_1} \otimes \ldots \otimes a_{i_l}$ such that either $a_{i_j} \neq \mathbf{1}$, or $l = 1$ and $a_{i_1} = \mathbf{1}$. Then between any two coupons of the composition of 1 we can insert $\eta \circ \eta^{-1}$, and replace φ_i with $\eta^{-1} \circ \varphi_i \circ \eta$. Since

$$ {}_A\eta \circ R_{A\mathbf{1}} = id_A \circ \eta_A, \qquad \text{and } \eta_A \circ R_{\mathbf{1}A} = id_A \circ {}_A\eta, $$

in this process $R_{A\mathbf{1}}$ and $R_{\mathbf{1}A}$ can be replaced with straight lines. We also make the convention that if a coupon has only one bottom line, or only one top line labeled by the trivial representation, those will be omitted. In this way, to the original composition of morphisms we correspond a diagram in which the trivial representation is denoted by the empty line.

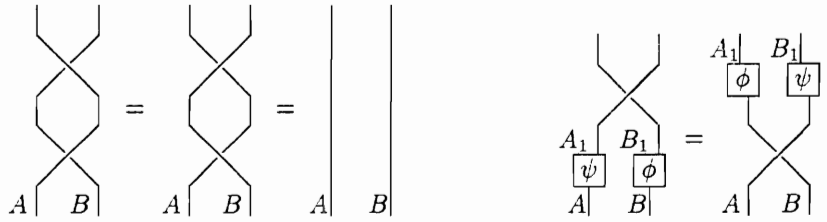


Figure 3: Identities satisfied by the commutativity.

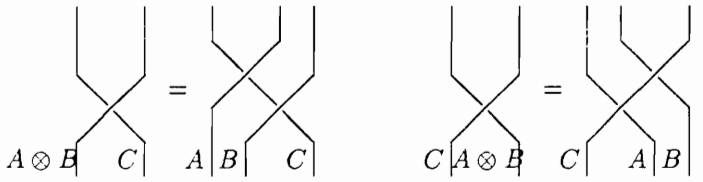


Figure 4: The hexagon identities

4.1.4 Let $\lambda_a : \hat{a} \otimes a \to \mathbf{1}$, be the standard form given by $\lambda_a(y \otimes x) = \vartheta_{\hat{a}}(y)(x)$, where $y \in \hat{a}$, and $x \in a$. Let $A = a_1 \otimes a_2 \ldots a_k$. Set $\hat{A} = \hat{a}_k \otimes \hat{a}_{k-1} \ldots \hat{a}_1$. Then define $\lambda_A : \hat{A} \otimes A \to \mathbf{1}$ by iterating the rule:

$$\lambda_{a\otimes b}((\hat{y} \otimes \hat{x}) \otimes (x \otimes y)) = \lambda_a(\hat{x} \otimes x)\lambda_b(\hat{y} \otimes y),$$

where $x \in a$, $\hat{x} \in \hat{a}$, $y \in b$, $\hat{y} \in \hat{b}$. The notation used for the coupon of the form λ_A. and the diagram equivalent of the identity above are presented in Figure 5.

4.1.5 Consider the two morphisms $\sigma_l, \sigma_r : B \otimes A \otimes \hat{B} \to A$, given by $\sigma_l = \eta_A(id_A \otimes \lambda_{\hat{B}})(R_{BA} \otimes id_{\hat{B}})$, $\sigma_r = (\lambda_{\hat{B}} \otimes id_A)(id_B \otimes R^{-1}_{A\hat{B}})$. Using the naturality of R one can easily see that $\sigma_l(id_b \otimes R_{A\hat{B}}) = \sigma_r(id_b \otimes R_{A\hat{B}}) : B \otimes \hat{B} \otimes A \to A$. Hence $\sigma_l = \sigma_r$. This is identity is expressed by the diagram in Figure 6.

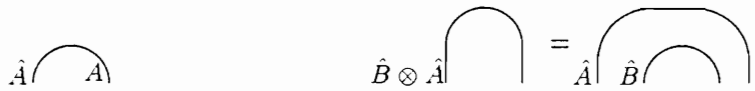


Figure 5: The form λ_A and $\lambda_{A\otimes B}$.

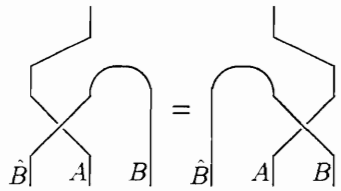


Figure 6: Identity involving the commutativity and the form.

4.1.6 For any $a \in \Sigma$ let's fix a basis $\mathcal{B}_a$ compatible with the weight-space decomposition of a. We denote with $\mathcal{B}_a^\mu$ the intersection of $\mathcal{B}_a$ with the μ-weight space of a and dual basis of a^* with $\mathcal{B}_a^*$. Then $(\mathcal{B}_a^*)^\mu = (\mathcal{B}_a^{-\mu})^*$.

Then there is a unique coform $\Lambda_a : \mathbf{1} \to a \otimes \hat{a}$ such that

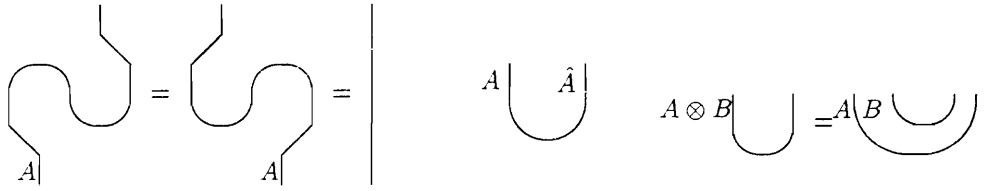
$$
\begin{aligned}
{}_a\eta(id \otimes \lambda_a)(\Lambda_a \otimes id)\eta_a^{-1} &= id \\
\eta_a(\lambda_{\hat{a}} \otimes id)(id \otimes \Lambda_{\hat{a}})_a\eta^{-1} &= id,
\end{aligned}
$$

given by $\Lambda_a = \sum_{e\in\mathcal{B}_a} e \otimes \vartheta_{\hat{a}}^{-1}(e^*)$. We extend Λ to the tensor product of any number of simple objects by iterating the rule:

$$
\Lambda_{a\otimes b}(1) = \sum_{e\in\mathcal{B}_a}\sum_{e'\in\mathcal{B}_b} e \otimes e' \otimes \vartheta_{\hat{b}}^{-1}(e'^*) \otimes \vartheta_{\hat{a}}^{-1}(e^*).
$$

The notation for the coupon corresponding to the coform and the diagrams corresponding to the above identities are shown in Figure 7.

4.1.7 By evaluating and using the symmetry of the category, one can deduce the diagram identity in Figure 8:



(a) The defining identities for Λ_A. (b) The coupon for $\Lambda_{\hat{A}}$ (c) Definition of $\Lambda_{A\otimes B}$

Figure 7: Some diagrams involving the coform.

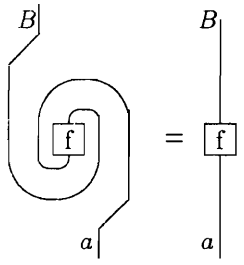


Figure 8: The "spiral".

4.2 The twist

4.2.1 Set $twist_l(A) = \eta_A(\lambda_A \otimes id_A)(id_{\hat{A}} \otimes R_{AA})(\Lambda_{\hat{A}} \otimes id_A)\eta_A^{-1} : A \to A$, and $twist_r(A) = {}_A\eta(id_A \otimes \lambda_{\hat{A}})(R_{AA} \otimes id_{\hat{A}})(id_A \otimes \Lambda_A){}_A\eta^{-1} : A \to A$. In terms of diagrams these morphisms are presented in Figure 9.

Note that because of the integrality properties of the commutativity morphism and the duality morphisms, for $a \in \Sigma$, $twist_l(a)$ and $twist_r(a)$ are just multiplication by an element in $\mathcal{R}$. Our goal is to compute these elements.

Proposition 4.2.2 *$twist_l(a)$ and $twist_r(a)$ are equal, and act on a as multiplication by $v^{-D(\lambda,\lambda-\sum_{i=1}^{n}\lambda_i i,i)}$, where λ is the highest weight of a.*

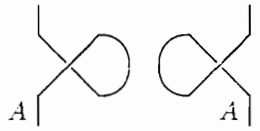


Figure 9: The twists.

Let x belong to the μ-weight space of a. Then by direct evaluating we get

$$twist_r(a)(x) = \sum_{c \in \mathbf{Z}_p^{l0}} \sum_{e \in \mathcal{B}_a^{\mu+|b_c|}} (-1)^{lev(|b_c|)} v_{|b_c|} h_{|b_c|} v^{-D(\mu.\mu+|b_c|.\mu)} \vartheta_a(b_c^+ x)(\vartheta_{\hat{a}}^{-1} e^*) b_c^- e.$$

Since the category is tortile (see 3.4.12), $\vartheta_a(b_c^+ x)(\vartheta_{\hat{a}}^{-1} e^*) b_c^- e = v^{\rho(\mu+|b_c|)} e^*(b_c^+ x) b_c^- e$. Using this fact we can perform the summation over e, and we get:

$$twist_r(a)(x) = \sum_{c \in \mathbf{Z}_p^{l0}} (-1)^{lev(|b_c|)} v_{-|b_c|} h_{|b_c|} v^{-D(\mu.\mu+|b_c|.\mu+\sum_{i=1}^n \mu_i i.i)} b_c^- b_c^+ x.$$

Comparing with 3.3.2 (a) we see that this is exactly equal to $\Omega' \hat{v}^{G'}(x)$.

The computation for $twist_l(a)$ goes in a similar way. By direct evaluation we get that:

$$twist_l(a)(x) = \sum_{c \in \mathbf{Z}_p^{l0}} \sum_{e \in \mathcal{B}_{\hat{a}}^{-\mu+|b_c|}} (-1)^{lev(|b_c|)} v_{|b_c|} h_{|b_c|} v^{-D(\mu.\mu-|b_c|.\mu)} \vartheta_{\hat{a}}(e)(b_c^- x) b_c^+ \vartheta_a^{-1}(e^*).$$

Using again the tortile property of the category, we deduce that

$$\vartheta_{\hat{a}}(e)(b_c^- x) b_c^+ \vartheta_a^{-1}(e^*) = v^{\rho(-\mu+|b_c|)} b_c^+ \vartheta_a^{-1}(\vartheta_a b_c^- x)(e) e^*.$$

Then, the summation over e can be performed and we get:

$$twist_l(a)(x) = \sum_{c \in \mathbf{Z}_p^{l0}} (-1)^{lev(|b_c|)} v_{-|b_c|} h_{|b_c|} v^{-D(\mu.\mu-|b_c|.\mu+\sum_{i=1}^n \mu_i i.i)} b_c^+ b_c^- x.$$

Comparing with 3.3.2 we see that this is exactly equal to $\Omega'' \hat{v}^{G''}(x)$. Then the statement follows from 3.3.3.

4.2.3 We denote $twist_r(a) = twist_l(a)$ with t_a. Note that from the invariance of the "."-form under the action of the Weyl group (2.4.1), and 2.4.3 we see that $t_a = t_{\hat{a}}$.

The identities in Figure 10a and 10c can be proved by applying few times the naturality of R, and the one in Figure 10b follows from the naturality of R and the identity in Figure 8.

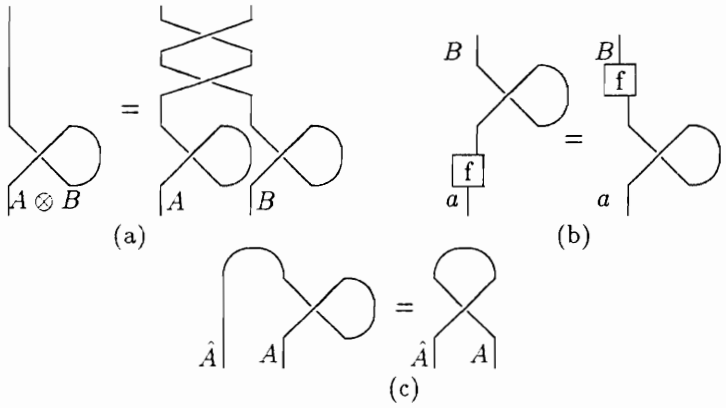


Figure 10: Some identities involving the twist.

4.2.4 By applying consecutively the naturality of R, and Figures 6 and 7, one can prove the identity in Figure 11.

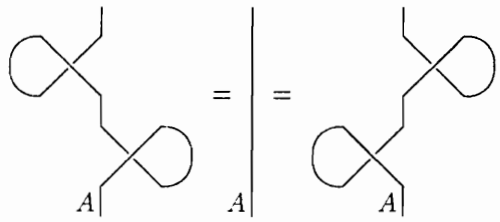


Figure 11: The inverse of the twist.

Therefore, we have that $\eta_a(id_{\hat{a}} \otimes R_{a\hat{a}}^{-1})(\lambda_a \otimes id_a)(\Lambda_{\hat{a}} \otimes id_a) = \eta_a(id_a \otimes \lambda_{\hat{a}})(id_{R_{a\hat{a}}^{-1} \otimes \hat{a}})(id_a \otimes \Lambda_a) == t_a^{-1}$.

4.3 Quantum dimension and traces

4.3.1 Given any morphism $f : A \to A$, the *trace* of f is defined to be $tr(f) = \lambda_A \circ (id_{\hat{A}} \otimes f)\Lambda_{\hat{A}}$. Then the quantum dimension of $a \in \Sigma$ is $\dim_q(a) = tr(id_a)$. By using the tortile property of the category we find:

$$dim_q(a)(1) = \sum_{e \in \mathcal{B}_{\hat{a}}} \vartheta_{\hat{a}}(e)(\vartheta_a(e^*)) = \sum_{e \in \mathcal{B}_{\hat{a}}} v^{\rho(\lambda_e)},$$

where λ_e is the weight of e and $\rho(\mu) = -\sum_{i=1}^{n} \mu_i i.i$ is as in 2.6.3.

The diagrams of $dim_q(a)$ and $tr(f)$ are presented on Figures 12a and 12b.

Figure 12: Quantum traces.

4.3.2 Let $f : A \to B$, $\varphi : B \to A$, and $\psi : A \to A$. Then by applying the naturality of R, and the identity in Figure 7 it can be shown that

(a) $\lambda_{\hat{A}}(\psi \otimes id_{\hat{A}})\Lambda_A = tr(\psi)$.

(b) $tr(\varphi \circ f) = \lambda_B(twist_l(B)^{-1} \otimes (f \circ twist_r(A) \circ \varphi))\Lambda_{\hat{B}}$.

The graphic notation for the statements above is given on Figure 13.

4.4 Some isomorphisms of spaces of homomorphisms

This section is a preparation for 5.4 where the choices of bases for the spaces $hom_{\overline{\mathcal{C}}_p}(a, b \otimes c)$ are described. Here we will denote with $\{\imath_k(a, B)\}$ and $\{\pi_k(B, a)\}$ a basis for $hom_{\overline{\mathcal{C}}_p}(a, B)$ and its splittings as described in 3.4.7. Often the tensor product of A and B will be denoted with AB.

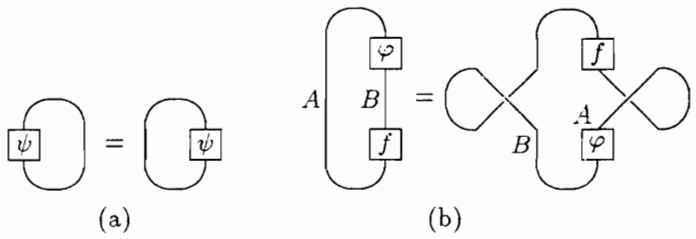


Figure 13: Some relations between traces.

4.4.1 Since $hom(a,a) \simeq \mathcal{R}'$ one can define a pairing $F_{a,B} : hom(a,B) \otimes hom(\hat{a},\hat{B}) \to \mathcal{R}'$ as shown in Figure 14a. This defines a pairing $F_{a,B} : hom_{\overline{\mathcal{C}}_p}(a,B) \otimes hom_{\overline{\mathcal{C}}_p}(\hat{a},\hat{B}) \to \mathcal{R}'$. Using the identities in Figure 7 is easy to see that if $\sigma : hom(B,a) \to hom(\hat{a},\hat{B})$ and $\hat{\sigma} : hom(\hat{B},\hat{a}) \to hom(a,B)$ are as in Figure 14b and 14c, $F_{a,B}(\iota_k(a,B), \sigma(\pi_k(B,a))) = 1$, and $F_{a,B}(\hat{\sigma}(\pi_k(\hat{B},\hat{a})), \iota_k(\hat{B},\hat{a})) = 1$. Hence the pairing $F_{a,B}$ is nondegenerate, and its adjoint $\partial : hom_{\overline{\mathcal{C}}_p}(\hat{a},\hat{B}) \to hom_{\overline{\mathcal{C}}_p}(a,B)^*$ is an isomorphism.

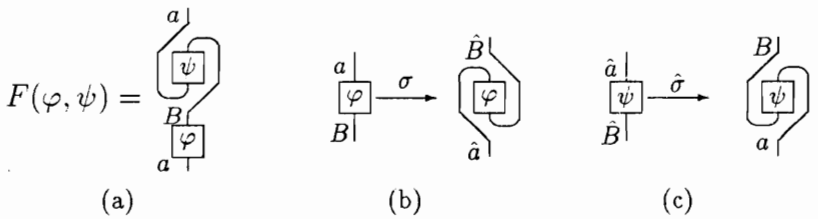


Figure 14: The pairing $F : hom(a,B) \otimes hom(\hat{a},\hat{B}) \to \mathcal{R}'$.

4.4.2 Define the maps $\tau : hom(a,B) \to hom(\mathbf{1}, a \otimes B)$, and $f_{cyc} : hom(\mathbf{1}, a \otimes B) \to hom(\mathbf{1}, B \otimes a)$, as presented in Figures 15a and 16a.

These maps are isomorphisms, and their inverses are presented in Figures 15b and 16b. Moreover they define isomorphisms $\tau : hom_{\overline{\mathcal{C}}_p}(a,B) \to$

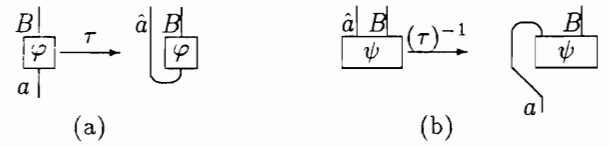


Figure 15: The isomorphism τ and its inverse.

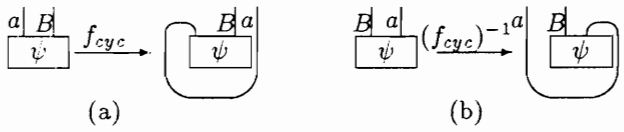


Figure 16: The isomorphism f_{cyc} and its inverse.

$hom_{\overline{\mathcal{C}}_p}(\mathbf{1}, a \otimes B)$ and $f_{cyc} : hom_{\overline{\mathcal{C}}_p}(\mathbf{1}, a \otimes B) \to hom_{\overline{\mathcal{C}}_p}(\mathbf{1}, B \otimes a)$. We will refer to the map f_{cyc} as the *cyclic permutation.*

Proposition 4.4.3 *Let $f : \hat{a} \to B$ be a morphism in $\mathcal{C}_p'$, and let $s_{aB} : a \otimes B \to B \otimes a$ be the $\mathcal{R}'$-linear map given by $x \otimes y \to y \otimes x$, $x \in a$, $y \in B$. Then*

$$(f \otimes id_a)\Lambda_{\hat{a}} = (id_b \otimes \hat{v}^{-\rho})s_{aB}(id_a \otimes f)\Lambda_a,$$

i.e. $f_{cyc} = (id_B \otimes \hat{v}^{-\rho})s_{aB}$.

First note that the set of vectors $\mathcal{B}'_{\hat{a}} = \{\vartheta_{\hat{a}}^{-1}(e^*)\}_{e \in \mathcal{B}_a}$ forms a basis of $\hat{a}$ and $(\vartheta_{\hat{a}}^{-1}(e^*))^* = \vartheta_{\hat{a}}^* \circ \Phi_a(e)$. Then, by using the fact that the coform is unique and the tortile property of the category we get

$$\begin{aligned}(f \otimes id_a)\Lambda_{\hat{a}} &= \sum_{e' \in \mathcal{B}'_{\hat{a}}} f(e') \otimes \vartheta_a^{-1}(e'^*) \\ &= \sum_{e \in \mathcal{B}_a} f \circ \vartheta_{\hat{a}}^{-1}(e^*) \otimes \vartheta_a^{-1} \circ \vartheta_{\hat{a}}^* \circ \Phi_a(e) \\ &= \sum_{e \in \mathcal{B}_a} f \circ \vartheta_{\hat{a}}^{-1}(e^*) \otimes \hat{v}^{-\rho}(e).\end{aligned}$$

The statement follows.

Proposition 4.4.4 *Let* $a, b, c \in \Sigma$, *and* $F'_{a,bc} = F_{b,ca}(\tau^{-1} f_{cyc}\tau \otimes \tau^{-1} f_{cyc}^{-1}\tau)$. *Then* $F'_{a,bc} = \frac{dim_q(a)}{dim_q(b)} F_{a,bc}$.

It is equivalent to show that for any $\phi \in hom_{\overline{\mathcal{C}}_p}(a, bc)$ and $\psi \in hom_{\overline{\mathcal{C}}_p}(\hat{a}, \hat{c}\hat{b})$, $tr(F'_{a,bc}(\phi, \psi)) = tr(F_{a,bc}(\phi, \psi))$. This is proved on Figure 17 where for the first step we have used twice the identity in Figure 7a and then Figure 13a.

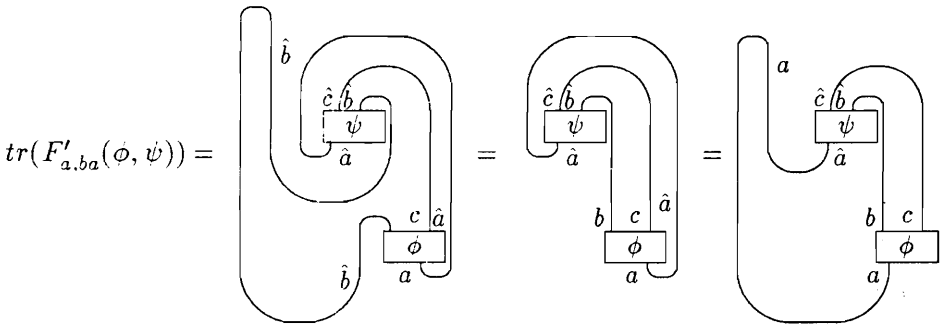


Figure 17: Proof of 4.4.4

4.4.5 From 4.1.5 and the naturality of R it can be easily deduced that $F_{a,cb}(R_{b,c} \otimes R^{-1}_{\hat{b},\hat{c}}) = F_{a,bc} : hom_{\overline{\mathcal{C}}_p}(a, bc) \otimes hom_{\overline{\mathcal{C}}_p}(\hat{a}, \hat{c}\hat{b}) \to \mathcal{R}'$.

Chapter 5
Algorithms

5.1 Linear algebra

5.1.1 In general, the computations which we discuss, are done over the field $\mathcal{R}'$, but because of the integrality properties of the L-S bases, and the duality morphisms, many of them actually go on in $\mathcal{R}$. Any element r in $\mathcal{R}'$ has a unique presentation of the form $r = a_1 + a_2 v + \ldots + a_{p-1}v^{p-2}$, $a_i \in \mathbf{Q}$, $1 \leq a \leq p-1$. Such an element is presented by the list $(a_1, a_2, \ldots, a_{p-1})$. Finding the inverse of r is reduced to solving the following linear system for $(x_1, x_2, \ldots, x_{p-1})$.

$$\begin{array}{lllll}
a_1x_1- & a_{p-1}x_2- & (a_{p-2}-a_{p-1})x_3- & \ldots- & (a_2-a_3)x_{p-1} = 1 \\
a_2x_1+ & (a_1-a_{p-1})x_2- & a_{p-2}x_3- & \ldots- & (a_2-a_4)x_{p-1} = 0 \\
a_3x_1+ & (a_2-a_{p-1})x_2- & (a_1-a_{p-2})x_3- & \ldots- & (a_2-a_5)x_{p-1} = 0 \\
\vdots & \vdots & \vdots & & \vdots \quad \vdots \\
a_{p-1}x_1+ & (a_{p-2}-a_{p-1})x_2- & (a_{p-3}-a_{p-2})x_3- & \ldots+ & (a_1-a_2)x_{p-1} = 0
\end{array}$$

The solution of this system is found by putting it in row-echelon form.

5.1.2 If A is an $l \times m$ matrix over $\mathcal{R}'$, all linear algebra in the programs rely on a function which puts A in standard row-echelon form by row operations. To be more precise, if A' denotes the row-echelon form of A and $A'_{i,j(i)}$ is the first nonzero element in the I'th row of A', we have:

(a) $j(i) < j(i')$ if $i < i'$,

(b) $A'_{k,j(i)} = \delta_{k,i}$, $1 \leq k \leq l$.

The left inverse of A is found by performing on the $l \times l$ identity matrix the same row operations needed to put A in standard form.

5.2 Generating the necessary representation data

In this section, if not said otherwise, we assume that $\tilde{L}_\lambda \in \Sigma$.

5.2.1 There are three kind of data about the simple objects in Σ, which we need to generate in order to be able to compute the split injections involved in the definition of the commutativity and associativity morphisms (see 3.4.9 and 3.2.9):

(a) an explicit expression for the monomial bases of the objects in Σ;

(b) the action of the algebra generators $E_i, F_i, i \leq n$ on these bases;

(c) the duality morphisms ϑ_a, $a \in \Sigma$.

Though most of the necessary statements were already given in 2.4 and 2.5, in this section we will summarize and outline the algorithms for generating this data.

5.2.2 According to 2.5.8, in order to be able to generate the L-S basis for ${}_{\mathcal{A}}L_\lambda$, we need to start with a reduced expression for the maximal element $w^0_\lambda \in \check{\mathbf{W}}_\lambda$. An inductive procedure for generating a reduced expression for any $w \in \check{\mathbf{W}}_\lambda$ relies on the proposition bellow.

Proposition 5.2.3 *Let $w = s_{i_1}s_{i_2}\dots s_{i_k} \in \check{\mathbf{W}}_\lambda$ be a reduced expression. Then, if for some $j \leq n$, $\langle h_j, s_{i_1}s_{i_2}\dots s_{i_k}(\lambda)\rangle > 0$, $s_j s_{i_1}s_{i_2}\dots s_{i_k}$ is also a reduced expression in $\check{\mathbf{W}}_\lambda$.*

This is a corollary of 2.4.4. In particular, from 2.4.1 it follows that $\langle s_{i_k}s_{i_{k-1}}\dots s_{i_1}(h_j), \lambda\rangle > 0$. Since $\lambda \in \mathcal{W}$, from 2.4.4 we have that $s_{i_k}\dots s_{i_1}(h_j)$ is a positive coroot, and therefore $s_{i_k}\dots s_{i_1}(h_j)s_j$ is reduced in $\mathbf{W}$. Now suppose that $s_j s_{i_1}s_{i_2}\dots s_{i_k}$ is not reduced in $\check{\mathbf{W}}_\lambda$. Then there exists $s_l \in \mathbf{W}_\lambda$, such that

$$s_j s_{i_1}s_{i_2}\dots s_{i_k} = s_{m_1}\dots s_{m_k}s_l,$$

i.e. $s_j s_{i_1}\dots s_{i_k}s_l$ is not reduced in $\mathbf{W}$, and hence $s_l s_{i_k}s_{i_{k-1}}\dots s_{i_1}(h_j)$ is a negative coroot. Then $\langle s_l s_{i_k}s_{i_{k-1}}\dots s_{i_1}(h_j), \lambda\rangle$ must be negative, but

$$\langle s_l s_{i_k}\dots s_{i_1}(h_j), \lambda\rangle = \langle h_j, s_{i_1}s_{i_2}\dots s_{i_k}s_l(\lambda)\rangle = \langle h_j, s_{i_1}s_{i_2}\dots s_{i_k}(\lambda)\rangle.$$

The last expression is positive. This leads to contradiction, and the statement follows.

5.2.4 By using this statement, we can find a reduced expression for any $w \in \tilde{\mathbf{W}}_\lambda$ by induction over the length of the elements starting with 1. Let $w^0 = s_{i_1}s_{i_2}\dots s_{i_{l^0}} \in \mathbf{W}$ be as in 3.1.4. Then for any element $w = s_{i_{g_1}}\dots s_{i_{g_k}} \in \tilde{\mathbf{W}}_\lambda$ of length k, and for any $j > g_1$ we check if

$$\langle h_{i_j}, s_{i_{g_1}}\dots s_{i_{g_k}}(\lambda)\rangle > 0.$$

If this is so, then $s_{i_j}s_{i_{g_1}}\dots s_{i_{g_k}}$ is element in $\tilde{\mathbf{W}}_\lambda$ of length $k+1$. The fact that in this way we generate all elements of length $k+1$ follows from 2.4.3 (a).

Note that if $w^0_\lambda = s_{j_1}\dots s_{j_r}$ is a reduced expression for the maximal element in $\tilde{\mathbf{W}}_\lambda$, then $w^0_{\hat{\lambda}} = s_{j_r}\dots s_{j_1}$ is a reduced expression for the maximal element in $\mathbf{W}_{\hat{\lambda}}$.

5.2.5 Having found a reduced expression $w^0_\lambda = s_{j_1}\dots s_{j_r}$ for the maximal element in $\tilde{\mathbf{W}}_\lambda$, we can generate the L-S basis of $\tilde{L}_\lambda$. According to 2.5.8 this is reduced to finding the canonical reduced expression for any $\pi \in \mathcal{P}_\lambda$ with respect to w^0_λ. Let $w_{r-k+1} = s_{i_k}s_{i_{k+1}}\dots s_{i_r}$, $1 \le k \le r$ be as in 2.5.8. Set $w_0 = 1$. Then we generate $\mathcal{P}_{\lambda,w_{j_k}}$ by induction over increasing k starting with $\mathcal{P}_{\lambda,w_0} = \{\pi_\lambda\}$. Suppose now that we know $\mathcal{P}_{\lambda,w_{j_{k-1}}}$. We then generate $\mathcal{P}_{\lambda,w_{j_k}} \backslash \mathcal{P}_{\lambda,w_{j_{k-1}}}$ in the following way. For any $\pi \in \mathcal{P}_{\lambda,w_{j_{k-1}}}$ we check if $min_t\langle h_{j_k}, \pi(t)\rangle = 0$. If not, we go to another path in $\mathcal{P}_{\lambda,w_{j_{k-1}}}$, otherwise we find $f_{j_k}\pi$, and check if it is in $\mathcal{P}_{\lambda,w_{j_{k-1}}}$. If it is so, we go to another path in $\mathcal{P}_{\lambda,w_{j_{k-1}}}$, if not, we add the set of paths $\{f^l_{j_k}\pi, 1 \le l \le \langle h_{j_k}, \pi(t)\rangle\}$ to $\mathcal{P}_{\lambda,w_{j_k}} \backslash \mathcal{P}_{\lambda,w_{j_{k-1}}}$. Here we used the fact that if for some $i \le n$, $f^l_i\pi = f^k_i\pi'$, with $min_t\langle h_i, \pi(t)\rangle = 0$ and $min_t\langle h_i, \pi'(t)\rangle = 0$, then $k = l$ and $\pi = \pi'$. This statement easily follows from the uniqueness in 2.5.7.

5.2.6 From the explicit knowledge of the monomial basis of the simple representation $\tilde{L}_\lambda$, it is easy to find the action of the algebra generators $E_i, F_i, i \le n$. To this goal, use induction over the decreasing level of the weight spaces of $\tilde{L}_\lambda$. We start the induction with the knowledge that the action of E_i on any weight space of level bigger or equal to $lev(\lambda)$, as well as the action of F_i on any weight space of level bigger then $lev(\lambda)$, is zero. Suppose that we know $E_i : \tilde{L}_\lambda{}^\mu \to \tilde{L}_\lambda{}^\nu$ and $F_i : \tilde{L}_\lambda{}^\nu \to \tilde{L}_\lambda{}^\mu$ for any μ and ν of level bigger or equal k. The we follow the steps below to compute the action of the generators mapping the weight spaces of level k and $k-1$ into each other.

(i) First we find the action of any E_j, $j \leq n$ on the weight spaces of level $k-1$ in the following way. Let $b = F_{i_1}^{(n_1)} F_{i_2}^{(n_2)} \dots F_{i_r}^{(n_r)} e_1$ be a basis vector in the weight space $\tilde{L}_\lambda{}^\mu$ where $lev(\mu) = k-1$. Then from 2.1.2 (c) we have

$$\begin{aligned} E_j b = \; & ([n_1]_{i_1})^{-1} F_{i_1} E_j F_{i_1}^{(n_1-1)} F_{i_2}^{(n_2)} \dots F_{i_r}^{(n_r)} e_1 \\ & + \delta_{i_1,j} ([n_1]_{i_1})^{-1} [\langle h_{i_1}, \mu - \alpha_{i_1} \rangle]_{i_1} F_{i_1}^{(n_1-1)} F_{i_2}^{(n_2)} \dots F_{i_r}^{(n_r)} e_1. \end{aligned}$$

Since $F_{i_1}^{(n_1-1)} F_{i_2}^{(n_2)} \dots F_{i_r}^{(n_r)} e_1$ is a basis vector of level k, all terms on the right are known.

(ii) Now we can find the action of any F_j, $j \leq n$ on the weight spaces of level k. Let b be a basic vector in the weight space $\tilde{L}_\lambda{}^\mu$ where $lev(\mu) = k$. Let $\{b_k\}_{k=1}^N$ be the L-S basis of $\tilde{L}_\lambda{}^{\mu-\alpha_j}$. Then we want to find $x_k \in \mathcal{R}$, $1 \leq k \leq N$ such that $F_j b = \sum_{k=1}^N x_k b_k$. Given any $l \leq n$, let $\{b'_s\}_{s=1}^M$ be the L-S basis of $\tilde{L}_\lambda{}^{\mu-\alpha_j+\alpha_l}$. Then the solution of the problem is given by the following proposition.

Proposition 5.2.7 *Let* $E_l b_k = \sum_{s=1}^M c_{l,k}^s b'_s$, *and* $E_l F_j b = \delta_{l,j}[\langle h_j, \mu \rangle]_j + F_j E_l b = \sum_{s=1}^M d_l^s b'_s$ [3]. *Then the linear system for* $x_1, x_2, \dots, x_N$

$$\sum_{k=1}^{N} c_{l,k}^s x_k = d_l^s, \; 1 \leq l \leq n, \; 1 \leq s \leq M,$$

has a unique solution.

The existence of a solution follows from the existence of a $\mathcal{R}'\mathfrak{u}$-module with the L-S basis constructed above. Define a map $\Gamma_\lambda^\nu : \tilde{L}_\lambda{}^\nu \to \oplus_{l=1}^n \tilde{L}_\lambda{}^{\nu+\alpha_l}$, given by $\Gamma_\lambda^\nu(m) = \oplus_{l=1}^n E_l(m)$, $m \in \tilde{L}_\lambda{}^\nu$. Note that if $lev(\nu) < lev(\lambda)$ this map is injective. Hence, the solution of the system above is unique.

5.2.8 Finally, we need to generate the duality morphisms $\vartheta_a : a \to \hat{a}^*$. Since the category is tortile, if an order is introduced on Σ, the ϑ_a's need to be computed only for $a \geq \hat{a}$. Then $\vartheta_{\hat{a}}$ is reconstructed from the tortile property. For $a \geq \hat{a}$, ϑ_a is computed inductively on the level of the weight

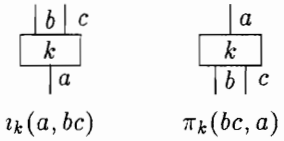
[3] note that in (i) we found $c_{l,k}^s$, and from the induction hypothesis we know d_l^s.

spaces, defining it to be the identity map on the highest weight space of a, and using the fact that any basic vector b of level $k-1$ is of the form $b = ([m]_l)^{-1}F_l b'$ for some $m \in \mathbf{N}$, $l \leq n$, and b' - a basis vector of level k. Hence, $\theta_a(b) = ([m]_l)^{-1}S(F_l)^*\theta_a(b')$ (by definition F_l acts on $\hat{a}$ as $S(F_l)^*$.

5.3 Finding a basis for $hom_{\overline{\mathcal{C}}_p}(a, b\otimes c)$ by direct computation

In the next sections all multiples $(a, b, c, \ldots)$ will be thought of as ordered ones. The subset of the same elements of Σ will be denoted with $\{a, b, c, \ldots\}$. Also, given $a \in \Sigma$, and $A \in \tilde{\mathcal{C}}_p$ we set $V(a, A) = hom_{\overline{\mathcal{C}}_p}(a, A)$. Often $b \otimes c$ will be written as bc to simplify the notations.

5.3.1 As said in 3.4.11, direct computation of the associativity morphisms from their definition requires for any triple (a, b, c) to find a basis $\{\imath_k(a, bc) : a \to bc\}_k$ of $V(a, b\otimes c)$, and splittings $\{\pi_k(bc, a) : bc \to a\}_k$. Often, we will refer to $\imath_k$ as injections, and π_k as projections. Through the isomorphism $hom(a, a) \simeq \mathcal{R}'$, the space, spanned by the projections, is identified with the dual $V(a, bc)^*$ of $V(a, bc)$, and with respect to this identification the π_k's represent the dual elements of the $\imath_k$'s. The graphic notation for the injections and projections is introduced in Figure 18.



$\imath_k(a, bc)$ $\quad$ $\pi_k(bc, a)$

Figure 18: Diagram notation for the basic elements of $V(a, W)$and their duals.

Due to the size of the ring elements and of the representations, the storage of all the basic elements and their duals for all triples is highly problematic, if not impossible, even for small p. In [31], an algorithm is described by which the storage time, and the computational time are vastly reduced. In the next few sections we summarize the results in [31] which relate to choices of

bases for the spaces $hom_{\overline{\mathcal{C}}_p}(a, bc)$, by using the graphic notations introduced in Chapter 3. In general terms the idea is to find a relatively small number of triples such that if a basis, and its dual, for each of them is known, there exists a basis for any other triple which, when needed, can be reconstructed relatively simply and quiqly from the data already known. The bases for this small number of triples are found by "direct computation", i.e. solving the corresponding linear systems, and now we proceed with describing how this is done.

Proposition 5.3.2 *Let $a, b, c \in \Sigma$, and let λ be the highest weight of a, and $g : a^\lambda \to (bc)^\lambda$ be a nonzero $\mathcal{R}'$-linear map such that $E_i \circ g : a^\lambda \to bc$ and $E_i^{(p)} \circ g : a^\lambda \to bc$ are zero for any $i \leq n$. Then g can be uniquely extended to a homomorphism $G : a \to bc$.*

From 3.4.4 we have that in $\mathcal{C}_p'$, bc is isomorphic to $A \oplus M$, $A \in \mathcal{C}_p$, $M \in \mathcal{C}_p{}^\perp$. Since the statement is obviously true for A, we only need to prove it for bc replaced with M. Let $e \in a$ be a highest weight vector. Then from the assumptions above follows that $g(e)$ is a highest weight vector in M (since ${}_{\mathcal{R}'}\mathbf{U}$ is generated by E_i and $E_i^{(p)}$), and $I = {}_{\mathcal{R}'}\mathbf{U}^- g(e)$ is a submodule in M. Then from 3.2.4 follows that $I = \oplus_{\mathcal{O}} I_{\mathcal{O}}$, where the sum is over all orbits of the dot action of $\mathbf{W}_p$ in $\mathcal{W}$ and all simple subquotients of $I_{\mathcal{O}}$ have the highest weights lying in $\mathcal{O}$. But since I is a highest weight module, and since there is an unique element of each orbit which belongs to $\triangle$, we deduce that all simple subquotients of I are isomorphic to a, i.e. I is isomorphic to a.

5.3.3 Using duality, a similar argument would show the following. Let $a, b, c \in \Sigma$, and let λ be the highest weight of a, and $B = bc$. Let also $g : (a^\lambda)^* \to (B^\lambda)^*$ be a nonzero $\mathcal{R}'$-linear map such that $F_i^* \circ g : (a^\lambda)^* \to B^*$ and $F_i^{(p)^*} \circ g : (a^\lambda)^* \to B^*$ are zero for any $i \leq n$. Then there exists a homomorphism $G : bc \to a$, such that $G|_{B^\lambda} = g^*$.

5.3.4 The triples (a, b, c) for which we want to find the basis by direct computation have that the dimension of a is greater or equal then the dimensions of b and c (see the definition of special triple in the next section). As a result, for the examples we have studied, $E_i^{(p)}$, and $F_i^{(p)}$ act as zero for dimension reasons. Then the propositions above suggest the following algorithm for finding a basis for $V(a, bc)$. As before we denote with λ the highest weight space of a and with e_a the highest weight vector in a.

(i) Find a basis $z_1, z_2, \ldots, z_N$ for the intersection of the kernels of all E_i, $i \leq n$, on $(bc)^\lambda$. From 5.3.2 this produces a basis $I_1, I_2, \ldots, I_N$ of $hom(a, bc)$ and a map $Inj : hom(a, bc) \otimes e \to bc$, $I_k \otimes e \to I_k(e) = z_k$.

(ii) Find a basis $y_1, y_2, \ldots, y_M$ for the intersection of the kernels of all F_i^*, $i \leq n$, on $((bc)^\lambda)^*$. From 5.3.3 this produces a basis $\pi_1, \pi_2, \ldots, \pi_M$ of $hom(a, bc)$ and defines a map $Proj : (bc) \to hom(bc, a) \otimes e$, $m \to \sum_{k=1}^{M} y_k(e^*)(m)\pi_k \otimes e$.

(iii) Change the basis of $hom(a, bc)$ to $\imath_1, \imath_2, \ldots, \imath_N$, so that $y_k(e^*)(\imath_l(e)) = \delta_{k,l}$ if $k \leq dim(a, bc) \leq N$, and zero otherwise. Then $\imath_k$, $1 \leq k \leq dim(a, bc)$, which we denote with $\imath_k(a, bc)$, form a basis for $V(a, bc)$, and π_k, $1 \leq k \leq dim(a, bc)$, denoted with $\pi_k(bc, a))$, form the corresponding dual basis.

5.3.5 The procedure, described above, determines a basis for $V(a, bc)$ and its dual, but only the restrictions of these morphisms to the λ-weight spaces of a and bc are explicitly known. We introduce special names for those. The map $\imath_k(a, bc) : a^\lambda \to (bc)^\lambda$ is called the *goodvector* of the triples (a, b, c), and $\pi_k(bc, a) : (bc)^\lambda \to a^\lambda$ is called the *goodcovector* of this triple. Knowing the goodvectors, $\imath_k(a, bc) : a^\mu \to (bc)^\mu$ can be computed inductively on decreasing level μ. In particular, suppose, we know the value of $\imath_k(a, bc)$ on any $x \in a$ of level greater or equal to k. By the definition of the L-S basis of a, any basic vector $b \in a$ of level $k - 1$ is of the form $b = ([m]_l)^{-1} F_l b'$ for some $m \in \mathbf{N}$, $l \leq n$, and some basis vector b' of level k. Hence, $\imath_k(a, bc)(b) = ([m]_l)^{-1} \triangle(F_l)\imath_k(a, bc)(b')$

5.3.6 The rest of $\pi_k(bc, a)$ can not be reconstructed from the goodcovectors in the same manner. But note that in 4.4.1 we defined an isomorphism $\partial : V(\hat{a}, \hat{cb}) \to V(a, bc)^*$. Hence $\imath_k(\hat{a}, \hat{cb}) = \partial^{-1}(\pi_k(bc, a))$, $1 \leq k \leq dim(a, bc)$ form a basis for $V(\hat{a}, \hat{cb})$, and the goodcovectors correspond to the maps $\imath_k(\hat{a}, \hat{cb}) : \hat{a}^{-\lambda} \to (\hat{cb})^{-\lambda}$. We will show how from this data to find the goodvectors for the triple $(\hat{a}, \hat{cb})$, and then one can use the same algorithm as in 5.3.5 to find the rest of $\imath_k(\hat{a}, \hat{cb})$.

Proposition 5.3.7 *Let* $\epsilon_{\hat{a}} = F_{i_1}^{(l_1)} F_{i_2}^{(l_2)} \ldots F_{i_s}^{(l_s)} e_{\hat{a}}$ *be the basic vector in the lowest weight space of* $\hat{a}$. *Then* $e_{\hat{a}} = E_{i_s}^{(l_s)} E_{i_{s-1}}^{(l_{s-1})} \ldots E_{i_1}^{(l_1)} \epsilon_{\hat{a}}$.

Let $y_k = F_{i_k}^{(l_k)} \dots F_{i_s}^{(l_s)} e_{\hat{a}}$, $1 \leq k \leq s$. From the definition of L-S basis, it is easy to see that $\langle h_{i_k}, \lambda_{k-1} \rangle = l_k$. Then, by using the commutation relations in 2.1.6 we deduce that $E_{i_k}^{(l_k)} y_k = y_{k-1}$. The statement follows.

From the observation above follows that the goodvectors for the triple $\hat{a}, \hat{c}, \hat{b}$ can be obtained by applying the above sequence of increasing operators, i.e.

$$\imath_k(\hat{a}, \hat{cb})(e_{\hat{a}}) = \triangle(E_{i_s}^{(l_s)} E_{i_{s-1}}^{(l_{s-1})} \dots E_{i_1}^{(l_1)}) \imath_k(\hat{a}, \hat{cb})(\epsilon_{\hat{a}}).$$

5.3.8 Now, given $b, c \in \Sigma$, and $\lambda \in \mathcal{W}$, we describe the algorithms for solving the linear system

$$E_z(b \otimes c)^\lambda = 0, \ 1 \leq z \leq n.$$

This is a homogeneous linear system with a very particular block structure, and we want to take advantage of it. The algorithm has two main steps:

(a) Solve the system $E_z(b \otimes c)^\lambda = 0$ for each z separately.

(b) Find the intersection of the solutions above.

5.3.9 A basis for $(b \otimes c)^\lambda$ is constructed in the following way. $(b \otimes c)^\lambda = \ominus_{\mu+\nu=\lambda}[\mu, \nu]$, where $[\mu, \nu] = b^\mu \oplus c^\nu$ are called pieces. Then if $\{e_i^\mu\}_i$ is the basis of b^μ, and $\{f_j^\mu\}_j$ is the basis of c^ν, the set $\{e_i^\mu \otimes f_j^\mu\}_{i,j}$, ordered lexicographical, forms a basis for $[\mu, \nu]$. Then fixing an order $p_1, p_2, \dots, p_N$ of the set of pieces, fixes a basis of $(b \otimes c)^\lambda$.

We need to introduce some conventions. If $V = (V_1, V_2, \dots, V_k)$ is a list of data, we refer to the m-th element in the list as $V[[m]]$. A vector in a vector space L will be described as a horizontal row so that the operators act by multiplication on the right. Then a collection of vectors $v^{(m)} \in L$, $1 \leq m \leq M$ will be described by a matrix v, where the m-th row of v corresponds to $v^{(m)}$. Moreover, if $L = \oplus_{i=1}^N L_i$, a collection of vectors $v^{(m)}$, $1 \leq m \leq M$ in L will be described as a list of N matrices V such that the m-th row in $V[[i]]$ is equal to $v_i^{(m)}$, where $v^{(m)} = \oplus_{i=1}^N v_i^{(m)}$, $v_i^{(m)} \in L_i$.

5.3.10 In this paragraph we consider $z \leq n$ fixed, and describe how to find a basis for the kernel of E_z on $(b \otimes c)^\lambda$. There are the following decompositions:

$$(b \otimes c)^\lambda = \oplus_{k=1}^{l(z)} s(z,k),$$
$$(b \otimes c)^{\lambda+\alpha_z} = \oplus_{k=1}^{l(z)} s'(z,k),$$

such that

(i) $s(z,k)$ and $s'(z,k)$ are each direct sum of pieces;

(ii) If $[\mu,\nu],[\mu',\nu'] \in s(z,k)$ (or $s'(z,k)$), $\mu' = \mu + m\alpha_z$, $\nu' = \mu - m\alpha_z$ for some $m \in \mathbf{Z}$;

(iii) $E_z : s(z,k) \to s'(z,k)$, and for each piece q in $s'(z,k)$, there is no more then two pieces in $s(z,k)$ which are mapped into q;

(iv) $s(k,z)$ are nonempty (some of $s'(k,z)$ may be empty).

We refer to $s(z,k)$ and $s'(z,k)$ as *strings* and *upper strings*. If $E_{z,k}$ denotes the restriction of E_z to $s(z,k)$, a basis for the kernel of E_z will be a union of bases for the kernels of each $E_{z,k}$. So, we fix k and describe how to find the kernel of $E_{z,k}$. To simplify the notation, set $s = s(z,k)$ and $s' = s'(z,k)$. Order the pieces $(q_1, q_2, \ldots, q_L)$ of s and $(q_0, q_1, \ldots, q'_{L+1})$ of s' in such a way, that

(i) If $q_i, q_{i+1} \in s$, and $q_i = [\mu,\nu]$, then $q_{i+1} = [\mu - \alpha_z, \nu + \alpha_z]$ (and the same for the pieces in s');

(b) $\triangle(E_z)q_i \xrightarrow{D_{i,i-1}+D_{i,i}} q'_{i-1} \oplus q'_i$, where $D_{i,i-1} = E_z \otimes 1$ and $D_{i,i} = \tilde{K}_z \otimes E_z$.

These properties are illustrated in Figure 19. Note that some of q'_i can be zero.

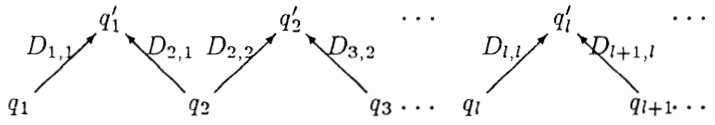


Figure 19: Action of $E_{z,k}$ on the ordered string.

By the convention in 5.3.9, a basis V for the kernel of $E_{z,k}$ is a list of N matrices where $V[[l]]$ is the zero matrix if p_l is not in s. So, we need to solve for $(V[[i_1]], V[[i_2]], \ldots, V[[i_L]])$, where $p_{i_l} = q_l, 1 \leq l \leq L$. Set $Q^j = \sum_{i=1}^{j}(D_{i,i} + D_{i+1,i}) : s \to s'$, $0 \leq j \leq L$. Define $Q^{-1} : s \to s'$ to be the zero map. We find the kernel of Q^j by induction over j.

(i) Start with $j = -1$. Then the basis of the kernel of Q^{-1} is $(V^{-1}[[i_1]], V^{-1}[[i_2]], \ldots, V^{-1}[[i_L]])$, where each $V^{-1}[[i_l]]$ is the corresponding identity matrix.

(ii) Suppose $(V^{j-1}[[i_1]], V^{j-1}[[i_2]], \ldots, V^{j-1}[[i_L]])$ is basis for the kernel of Q^{j-1}. Solve for (x_1, x_2) the linear system $x_1 V_{i_j}^{j-1} D_{j,j} + x_2 D_{j=1,j} = 0$. Then

$$(x_1 V^{j-1}[[i_1]], x_1 V^{j-1}[[i_2]], \ldots, x_1 V^{j-1}[[j]], x_2, V^{j-1}[[j+2]], \ldots, V^{j-1}[[i_L]])$$

is a basis for the kernel of Q^j.

5.3.11 Summarizing, the basis for the kernel of E_z , found in this way, can be thought of as a matrix A_z, divided in $l(z)$ rows and N columns of blocks, such that the k'th row of blocks is exactly the basis for the kernel of $E_{z,k}$. Moreover, in each column of blocks there is only one nonzero block. We then introduce a list of N nonzero matrices A_z^{flat} such that $A_z^{flat}[[m]]$ is equal to the nonzero block in the m-th column of blocks in A_z, and a list $\texttt{followup}_z$ of length N, such that $\texttt{followup}_z[[m]] = l$ where $p_m \in s(z, l)$. In this way, all the information about A_z is contained in these two lists; in particular, the only nonzero block in the m'th column of blocks in A_z is $A_z^{flat}[[m]]$, and it is located in the $\texttt{followup}_z[[m]]$ row of blocks.

5.3.12 Now we want to find a basis for the intersection of the kernels of all E_z, $1 \leq z \leq n$. The algorithm goes in the following way. First, we modify the values of A_1^{flat}, A_2^{flat}, $\texttt{followup}_1$, and $\texttt{followup}_2$, in such a way that, at the end, A_1^{flat} and $\texttt{followup}_1$ describe a basis for the intersection of the kernels of E_1 and E_2. Then apply the same procedure to A_1^{flat} and A_3^{flat}, and so on.

5.3.13 To be concrete, we show how to find a basis for the intersection of the kernels of E_1 and E_2. The algorithm relies on the following trivial observation. Let L_1, L_2, L be vector spaces over a field, and $P_i : L_i \to L, i = 1, 2$ linear operators. Let each $b^r, 1 \leq r \leq k$ be a set of vectors in $L_1 \oplus L_2$ such that

(i) The union of all $b^r, 1 \leq r \leq k$ forms a basis for a subspace $W \subset L_1 \oplus L_2$.

(ii) There is unique $m \leq k$ and unique $l \leq k$ such that $b^m[[1]]P_1 \neq 0$ and $b^l[[2]]P_2 \neq 0$.

If $l = m$ we denote with y the solution of the system $y(b^m[[1]]P_1 - b^l[[2]]P_2) = 0$, and if $l \neq m$ denote with (y_1, y_2) the solution of the system $y_1 b^m[[1]]P_1 - y_2 b^l[[2]]P_2 = 0$. Then a basis for the intersection of the kernel of the operator $L_1 \oplus L_2 \to L, (x_1, x_2) \to P_1(x_1) - P_2(x_2)$ with W, is given by the union of the vectors in $b^r, r \neq l, m$ plus, if $l = m$, the vectors $(yb^m[[1]], yb^l[[2]])$, and if $l \neq m$, the vectors $(y_2 b^l[[1]] + y_1 b^m[[1]], y_2 b^l[[2]] + y_1 b^m[[2]])$.

5.3.14 Let Π_j be the projection $(b \otimes c)^\lambda \to p_j$, and $\hat{\Pi}_j = \sum_{r=1}^{j} \Pi_j$. Set $\Upsilon_j : Ker(E_1) \oplus Ker(E_2) \to \oplus_{r=1}^{j} p_r, (x, y) \to \hat{\Pi}_j(x - y)$. The idea is to find inductively the kernel of Υ_j using the observation above. To start the induction, take the lists A_z^{flat}, $\texttt{followup}_z$, $z = 1, 2$ as described above, and add N to each element of $\texttt{followup}_2$. Then the statements of the induction is the following:

(i) If $\texttt{followup}_z[[m]] = \texttt{followup}_{z'}[[k]]$, $A_z^{flat}[[m]]$, and $A_{z'}^{flat}[[l]]$, $z, z' = 1, 2$, have the same number of rows.

(ii) Let S_z denote the set of different values of $\texttt{followup}_z$. Then for any $s \in S_z$, denote with $B_{z,s}$ the set of vectors in $\oplus_{i=1}^{N} p_i$, where $B_{z,s}[[m]] = A_z^{flat}[[m]]$ if $\texttt{followup}[[m]] = s$, and is the zero block otherwise. Then for any $k \leq N$ there exists unique $s \in S_z$ with the k-th block of $B_{z,s}$ different from zero.

(iii) A basis for Υ_j is given by $(B_{1,s}, B_{2,s})$, $s \in S_1 \cup S_2$, where we make the convention that if s is not in S_z, $B_{s,z}$ is the corresponding zero matrix.

The statements are obviously true for $j = 0$. Assume we are at the step j. Let $m_z = \texttt{followup}_z[[j+1]], z = 1, 2$. If $m_1 \neq m_2$, solve for y_1, y_2 the system $y_1 A_1^{flat}[[j+1]] - y_2 A_2^{flat}[[j+1]] = 0$. Then for any $z = 1, 2$, multiply by y_z all blocks in A_1^{flat} and A_2^{flat} for which the value of the $\texttt{followup}$ list is equal to m_z. Then, for any $k \leq N$ and $z = 1, 2$ such that $\texttt{followup}_z[[k]] = m_1$ or m_2, put $\texttt{followup}_z[[k]] = m$, where m is the smaller of m_1 and m_2.

If $m_1 = m_2$, solve for y the system $y(A_1^{flat}[[j+1]] - A_2^{flat}[[j+1]]) = 0$. Then for any $z = 1, 2$, multiply by y all blocks in A_1^{flat} and A_2^{flat} for which the value of the $\texttt{followup}$ list is equal to m_1.

From the observation in 5.3.13, it follows that these number of steps produce lists A_z^{flat}, $\texttt{followup}_z$, $z = 1, 2$ which satisfy (i)–(iii) above with j replaced by $j+1$. This completes the outline of the algorithm for finding the

intersection of the kernels of the E_i's. Exactly the same algorithm is applied to find the intersection of the kernels of F_i^*'s.

5.4 Recovering bases out of minimal amount of data.

Following [31], for any triple (a, b, c), we describe a choice of basis for $hom_{\overline{\mathcal{C}}_p}(a, bc)$, and an algorithm how to recover those bases out of minimal amount of data. The convention for the notations is the same as in the previous section.

5.4.1 Choose an order of the elements in Σ in such a way that if the dimension of a is smaller or equal to the dimension of b, $a \leq b$, and also a and $\hat{a}$ are adjacent [4]. The triple (a_1, a_2, a_3) is called *special* if $a_1 \geq a_2 \geq a_3$, and $(a_1, a_2, a_3) \geq (\hat{a}_{i_1}, \hat{a}_{i_2}, \hat{a}_{i_3})$, where $\hat{a}_{i_1} \geq \hat{a}_{i_2} \geq \hat{a}_{i_3}$. The triple $(\hat{a}, \hat{c}, \hat{b})$ is called the *dual* of (a, b, c), and a triple which is special or its dual is special is called *good*. A triple (a, b, c) such that $\{a, b, c\} = \{\hat{a}, \hat{b}, \hat{c}\}$ is called *selfdual*.

5.4.2 First, we describe how to get the minimal amount of data out of which all bases are being reconstructed. To any triple $(\hat{a}, b, c)$ such that (a, b, c) is special, we apply the algorithm for finding a basis by direct computation, as described in the previous section. The result is a basis $\{\imath_k(\hat{a}, bc)\}_k$ of $V(\hat{a}, bc)$ and a basis $\{\imath_k(a, \hat{c}\hat{b})\}_k$ of $V(a, \hat{c}\hat{b})$ such that $\partial(\imath_k(a, \hat{c}\hat{b}) = \pi_k(\hat{a}, bc)$. If the triple is selfdual, we do not generate the basis for the dual triple, but instead compute the matrix $M_{\hat{a},b,c}$ whose (k, l) element is defined as here below.

(a) If $a > b > c$, and $a = \hat{a}$, $b = \hat{b}$ and $c = \hat{c}$,

$$M(\hat{a}, b, c)_{k,l} = F_{\hat{a},bc}(\imath_k(\hat{a}, bc), R_{b,c}\imath_l(\hat{a}, bc));$$

(b) If $a = \hat{a}$, and $b = \hat{c}$,

$$M(\hat{a}, b, c)_{k,l} = F_{\hat{a},bc}(\imath_k(\hat{a}, bc), \imath_l(\hat{a}, bc));$$

[4] The described algorithms are general, and do not depend on this particular choice of order. The advantage of it is to minimize the size of $b \otimes c$ for the triples for which the injections $\imath_k(a, b \otimes c)$ are directly computed. Note though that in [31] further considerations impose requirements for different order.

(c) If $b > c$ and $a = \hat{b}$, and $c = \hat{c}$,

$$M(\hat{a}, b, c)_{k,l} = F_{\hat{a},bc}(\imath_k(\hat{a}, bc), \tau^{-1} f_{cyc} \tau \imath_l(\hat{a}, bc)).$$

In terms of diagrams, the elements of these matrices are presented in Figure 20.

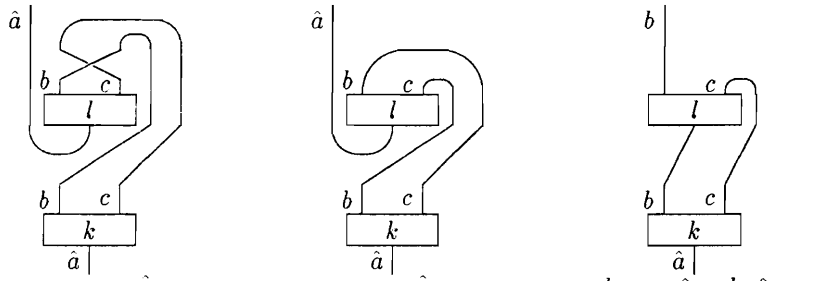


Figure 20: The matrices $M(\hat{a}, b, c)$.

The data described above is the primary data. It consists of the bases for $V(\hat{a}, bc)$, where (a, b, c) is a good triple, and of the matrices $M(\hat{a}, b, c)$ for (a, b, c) special. The goodvectors for these bases and the elements of the matrices are being computed and stored. When needed, this data is loaded and then the full values of the injections are being generated by the algorithm in 5.3.5. Now, we proceed with defining a basis for $V(a_1, b_1 c_1)$ for any other triple (a_1, b_1, c_1), and explaining how those bases and their duals, are expressed in terms of the primary data.

5.4.3 For any $a, b, c \in \Sigma$, set $\hat{f}_{cyc} = \tau^{-1} f_{cyc} \tau : V(a, bc) \to V(\hat{b}, c\hat{a})$ and $R : V(a, bc) \to V(a, cb), R(\phi) = R_{b,c} \circ \phi$. Let $(\hat{a}, b, c)$ be any triple such that (a, b, c) is not good. Then (a, b, c) is a permutation $\wp$ of a good triple (a', b', c') , and we define a basis for $V(\hat{a}, bc)$ as follows.

(a) If (a', b', c') is special, define a map $\chi_\wp : V(\hat{a}', b'c') \to V(\hat{a}, bc)$ as $\chi_{(3,1,2)} = \hat{f}_{cyc}$, and $\chi_{(2,3,1)} = \hat{f}^2_{cyc}$. Note that if some of a, b, c are the same, this covers all possibilities. In the case when they are all different, define $\chi_{(1,3,2)} = R$, $\chi_{(3,1,2)} = \hat{f}_{cyc} R$, and $\chi_{(2,3,1)} = \hat{f}^2_{cyc} R$. Then

set

$$\imath(\hat{a}, bc) = \chi_{\wp}(\imath_k(\hat{a}', b'c')).$$

The choices of bases above are described in terms of diagrams on Figure 21.

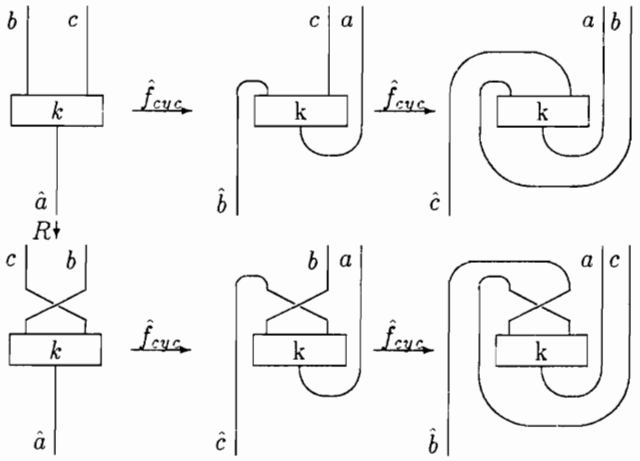


Figure 21: Bases associated with permutations of the special triple (a, b, c).

(b) If (a', b', c') is not special, define a map $\hat{\chi}_{\wp} : V(\hat{a}', b'c') \to V(\hat{a}, bc)$ as $\hat{\chi}_{(3,1,2)} = \hat{f}_{cyc}^{-1}$, and $\hat{\chi}_{(2,3,1)} = \hat{f}_{cyc}^{-2}$. Note that if some of a, b, c are the same, this covers all possibilities. In the case when they are all different, define $\hat{\chi}_{(1,3,2)} = R^{-1}$, $\hat{\chi}_{(3,1,2)} = \hat{f}_{cyc}^{-1}R^{-1}$, and $\hat{\chi}_{(2,3,1)} = \hat{f}_{cyc}^{-2}R^{-1}$. Then set

$$\imath(\hat{a}, bc) = \hat{\chi}_{\wp}(\imath_k(\hat{a}', b'c')).$$

The choices of bases above are described in terms of diagrams on Figure 22.

Therefore, a basis for any triple is computed by applying $\hat{f}_{cyc}$ or $\hat{f}_{cyc}^{-1}$ on the bases for $V(\hat{a}', b'c')$ and for $V(\hat{a}', c'b')$, where (a', b', c') is a good triple. From the definition of τ, and 4.4.3 follows that this is reduced to applying duality morphisms and multiplication by a constant, and is very fast. It is

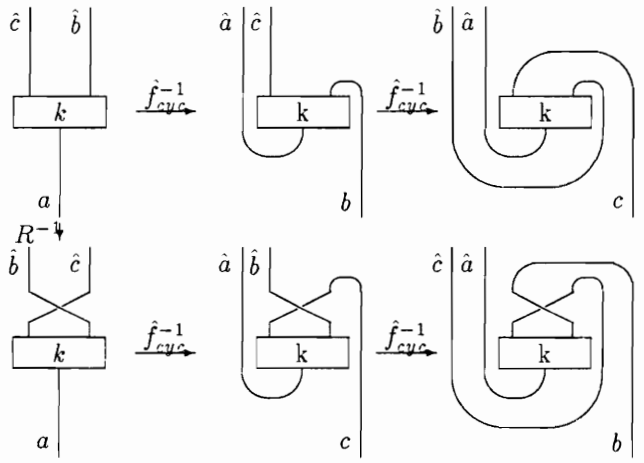


Figure 22: Bases associated with permutations of the dual of the special triple (a, b, c).

more problematic to get the bases for $V(\hat{a}', c'b')$ with (a', b', c') being good. By the definition above, this requires applying the commutativity morphism in 3.2.9. To be able to do so, first, we express $E_\beta, F_\beta, \beta \in \mathbf{R}^+$ from 2.9.2 in terms of the generators $E_i, F_i, i \leq n$ using the algebra relations described in 3.1.4, and then apply the formula in 3.2.9. The problem is that the expression for the quasi R-matrix (see 3.2.8) contains a very high number of terms. For example, for G_2, $p = 11$, the number is 11^6, and this is the smallest category associated with G_2. What has been possible for the examples computed so far is to apply the commutativity morphism on the goodvectors for $V(\hat{a}, bc)$, getting in this way the goodvectors for the $V(\hat{a}, cb)$, and then use the algorithm in 5.3.5 to compute the rest of the injections.

Proposition 5.4.4 *For any triple (a, b, c), where at least two elements are different, and for any element $\imath_k(\hat{a}, bc)$ in the basis of $V(\hat{a}, bc)$,*

$$f_{cyc}\tau\imath_k(\hat{a}, bc) = \tau\imath_k(\hat{b}, ca).$$

Given a good triple (a, b, c), it is enough to show that $f_{cyc}^3\tau\imath_k(\hat{a}', b'c') = \tau\imath_k(\hat{a}', b'c')$, where $(a', b', c') = (a, b, c)$ or (a, c, b) if (a, b, c) is special, and

$(a', b', c') = (b, c, a)$ or (c, b, a) if (a, b, c) is the dual of a special. But actually for any $\varphi \in V(\mathbf{1}, a'b'c')$, we have $f^3_{cyc}\varphi = \eta_A(\lambda_{\hat{A}} \otimes id_A)(id_{\hat{A}} \otimes \varphi \otimes id_A)({}_{\hat{A}}\eta^{-1} \otimes id_A)\Lambda_{\hat{A}} : \mathbf{1} \to A$, where $A = a' \otimes b' \otimes c'$. Then from the naturality of R and the identity on Figure 10b follows that $f^3_{cyc}\varphi = \varphi$ as it is shown in Figure 23.

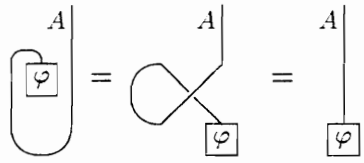


Figure 23: Invariance of the bases under cyclic permutations.

5.4.5 Finally, we now describe how to compute the dual bases. Using the isomorphism ∂ in 4.4.1 they can be obtained from the basis for the dual triple if the value of the form F with respect to these bases is known. Set $\mathtt{dual}[\hat{a}, b, c]_{k,l} = F_{\hat{a},bc}(\imath_k(\hat{a}, bc), \imath_l(a, \hat{cb}))$. Note that if (a, b, c) is special we already know the value of these matrices. In particular, if the triple is not selfdual, $\mathtt{dual}[\hat{a}, b, c]_{k,l} = \delta_{k,l}$, otherwise, $\mathtt{dual}[\hat{a}, b, c]_{k,l} = M(\hat{a}, b, c)_{k,l}$, which is part of the primary data. Our goal is to evaluate these matrices for any other triple.

Proposition 5.4.6 *Let (a, b, c) be a permutation $\wp$ of the special triple (a', b', c'). Then if the triple (a, b, c) is not selfdual, or if $\wp$ is even,*

$$\mathtt{dual}[\hat{a}, b, c]_{k,l} = \frac{dim_q(a')}{dim_q(a)}\mathtt{dual}[\hat{a'}, b', c']_{k,l}.$$

Otherwise,

$$\mathtt{dual}[\hat{a}, b, c]_{k,l} = \frac{dim_q(a')}{dim_q(a)}const(a', b', c')\mathtt{dual}[\hat{a'}, b', c']_{k,l},$$

where if $c = \hat{b}$, $const(a', b', c') = t_{a'}t_{b'}^{-2}$, and if $a = \hat{b}$, $const(a', b', c') = t_c^{-1}$. Otherwise, $const(a', b', c') = 1$.

Note that the bases were defined in such a way that if the basis for $triple_1$ is obtained from the basis of $triple_2$ by applying $\hat{f}_{cyc}$, then the basis for the dual of $triple_1$ is obtained by applying $\hat{f}_{cyc}^{-1}$ on the dual of $triple_2$. Hence, from 4.4.4 we deduce that, if φ is even, then

$$\texttt{dual}[\hat{a}, b, c]_{k,l} = \frac{dim_q(a')}{dim_q(a)}\texttt{dual}[\hat{a'}, b', c']_{k,l},$$

and if φ is odd, then

$$\texttt{dual}[\hat{a}, b, c]_{k,l} = \frac{dim_q(a')}{dim_q(a)}\texttt{dual}[\hat{a'}, c', b']_{k,l},$$

So, the only case we need to study is the one when $(a, b, c) = (a', c', b')$. Then the basis for $V(\hat{a}', c'b')$ is obtained by applying R on the basis for $V(\hat{a}', b'c')$. In the case when (a', b', c') is not selfdual, the basis for $V(a', \hat{b'c'})$ is obtained by applying R^{-1} on the basis for $V(a', \hat{c'b'})$. Hence, in this case, the statement of the proposition follows from 4.1.5. We still have to consider the case when (a', b', c') is selfdual, and $(a, b, c) = (a', c', b')$. Then there are three separate subcases, and the proofs are presented in Figure 24, Figure 25, and Figure 26.

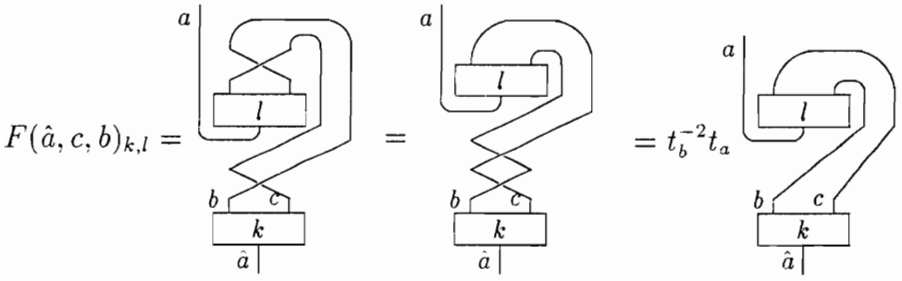


Figure 24: Computing the form in the case $a = \hat{a}$, $b = \hat{c}$.

This completes the evaluation of the pairing for triples which are permutations of special ones.

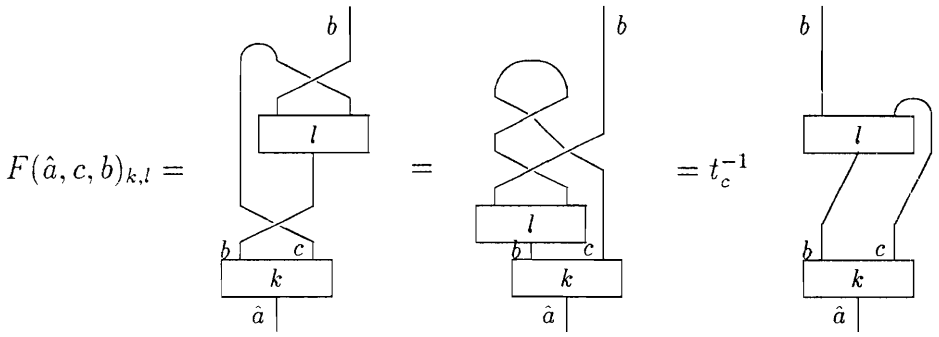


Figure 25: Computing the form in the case $b > c$, $a = \hat{b}$, $c = \hat{c}$.

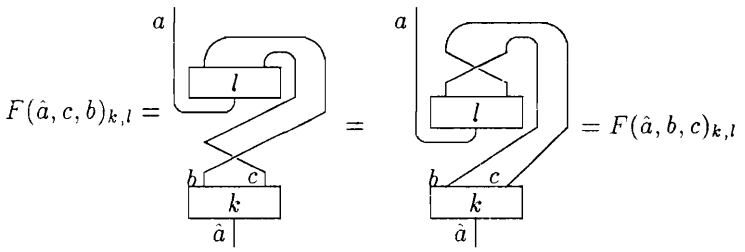


Figure 26: Computing the form in the case $a > b > c$, $a = \hat{a}$, $b = \hat{b}$, $c = \hat{c}$.

5.4.7 Note that if (a, b, c) is not a permutation of a special triple, then its dual is. But from 4.3.2b and Figure 10b it follows that $tr(\mathtt{dual}[\hat{a}, b, c]_{k,l}) = tr(\mathtt{dual}[a, \hat{c}\hat{b}]_{l,k})$. Since $dim_q(a) = dim_q(\hat{a})$, this implies that $\mathtt{dual}[\hat{a}, b, c]_{k,l} = \mathtt{dual}[a, \hat{c}\hat{b}]_{l,k}$. This completes the evaluation of the pairing for any triple.

5.4.8 We want to comment on the special case when one of the elements in a triple is $\mathbf{1}$. From the way the ordering was chosen in 5.4.1, it follows that all special triples which contain $\mathbf{1}$ are $(\hat{a}, a, \mathbf{1})$, $a \in \Sigma, \hat{a} > a$, and these triples are selfdual. The space $V(a, a\mathbf{1})$ is one dimensional with basis ${}_a\eta$. Then, by applying the rules above for finding a basis for any permutation of this triple, we get that the bases for $V(\hat{a}, \mathbf{1}\hat{a})$, $V(a, \mathbf{1}a)$, $V(\hat{a}, \hat{a}\mathbf{1})$ are $\eta_{\hat{a}}$, η_a and ${}_{\hat{a}}\eta$, respectively. Furthermore, the bases for $V(\mathbf{1}, a\hat{a})$ and $V(\mathbf{1}, \hat{a}a)$ are Λ_a and $\Lambda_{\hat{a}}$ respectively. The last piece of the primary data follows from

Figure 7a to be $M(\hat{a}, \hat{a}, \mathbf{1}) = 1$. Using this, or just evaluating explicitly, we deduce that the dual of Λ_a is $\frac{1}{dim_q(a)}\lambda_{\hat{a}}$, and the dual of $\Lambda_{\hat{a}}$ is $\frac{1}{dim_q(a)}\lambda_a$.

5.5 Computing the elementary commutativity and associativity morphisms

5.5.1 With respect to the bases described in the previous section, $\mathtt{com}[a, b, c] : V(a, bc) \to V(a, cb)$ is the matrix with k, l entry

$$\mathtt{com}[a, b, c]_{k,l} = \pi_l(cb, a) \circ R_{b,c} \circ \imath_k(a, bc),$$

Because of the way in which the bases were chosen, in the case when a, b, c are all different, $\mathtt{com}[\hat{a}, b, c]$ can be evaluated analytically. Moreover, if at least two are the same, for any permutation (a', b', c') of (a, b, b), $\mathtt{com}[\hat{a'}, b', c']$ can be expressed in terms of $\mathtt{com}[\hat{a}, b, b]$. We now proceed with describing how this is done.

Proposition 5.5.2 *For any* $a, b, c \in \Sigma$,

$$\mathtt{com}[a, b, c].\mathtt{dual}[a, c, b] = \mathtt{dual}[a, b, c]\mathtt{Transpose}[\mathtt{com}[\hat{a}, \hat{b}, \hat{c}]].$$

We set $b_k = \imath_k(a, bc)$ and $\hat{b}_k = \imath_l(\hat{a}, \hat{b}\hat{c})$. Then using 4.3.2 and 4.4.5 we have

$$\begin{aligned} F_{a.cb}(R_{b,c}b_k, \hat{b}_l) &= \frac{1}{dim_q(a)} tr(F_{a,cb}(R_{b,c}b_k, \hat{b}_l)) \\ &= \frac{1}{dim_q(a)} tr(F_{a,cb}(b_k, R_{\hat{b},\hat{c}}\hat{b}_l) \\ &= \frac{1}{dim_q(a)} tr(F_{\hat{a},\hat{c}\hat{b}}(R_{\hat{b},\hat{c}}\hat{b}_l, b_k) \\ &= F_{\hat{a},\hat{c}\hat{b}}(R_{\hat{b},\hat{c}}\hat{b}_l, b_k). \end{aligned}$$

The left hand side is actually equal to

$$\begin{aligned} \partial(\hat{b}_l) \circ R_{b,c} \circ b_k &= \sum_{l'} \partial(\hat{b}_l) \circ \imath_{l'}(a, cb) \circ \pi_{l'}(cb, a) \circ R_{b,c} \circ b_k(a, bc) \\ &= (\mathtt{com}[a, b, c].\mathtt{dual}[a, c, b])_{k,l}. \end{aligned}$$

In the same way can be shown that the right hand side is equal to $(\mathtt{com}[\hat{a}, \hat{b}, \hat{c}].\mathtt{dual}[\hat{a}, \hat{c}, \hat{b}])_{l,k}$. Then the proposition follows from 5.4.7.

5.5.3 The proposition above implies that it is enough to evaluate the commutativity matrices for triples which are permutations of a special ones. Furthermore, from Figure 10a we have that $\mathtt{com}[a,c,b] = t_a t_b^{-1} t_c^{-1} \mathtt{com}[a,b,c]^{-1}$. Hence all other cases can be deduce from the even permutations of the special triples.

Proposition 5.5.4 *Let (a',b',c') be an even permutation $\wp$ of the special triple (a,b,c). Then*

(a) If $a > b > c$

$$\mathtt{com}[\hat{a'},b',c'] = \mathtt{id} \begin{cases} 1 & \text{if } \wp = (1,2,3), \\ t_b t_a^{-1} & \text{if } \wp = (3,1,2), \\ t_c t_a^{-1} & \text{if } \wp = (2,3,1). \end{cases}$$

(b) If $a > b = c$,

$$\mathtt{com}[\hat{a'},b',c'] = \begin{cases} t_b^{-1}\mathtt{com}[\hat{a},b,b]^{-1} & \text{if } \wp = (3,1,2), \\ t_a^{-1} t_b \mathtt{com}[\hat{a},b,b] & \text{if } \wp = (2,3,1). \end{cases}$$

(b) If $a = b > c$,

$$\mathtt{com}[\hat{a'},b',c'] = \begin{cases} t_a^{-1}\mathtt{com}[\hat{c},a,a]^{-1} & \text{if } \wp = (1,2,3), \\ t_c^{-1} t_a \mathtt{com}[\hat{c},a,a] & \text{if } \wp = (3,1,2). \end{cases}$$

The statement in case (a) follows from the identities in Figures 27 and 28, which show that $\hat{f}_{cyc} R\imath_k(\hat{b},ca) = t_a^{-1} t_b \hat{f}_{cyc} \imath_k(\hat{b},ac)$ and $\tau R\imath_k(\hat{c},ab) = t_a^{-1} t_c \imath_k(\hat{c},ba)$, correspondingly.

To prove the statement in (b) , we show in Figure 29 that

$$\tau R \hat{f}_{cyc} R\imath_k(\hat{a},bb) = t_b^{-1} \tau \imath_k(\hat{b},ab).$$

From here, proposition 5.4.4 implies that $\mathtt{com}[\hat{a},b,b]\mathtt{com}[\hat{b},b,a] = t_b^{-1}\mathrm{Id}$. This proves the statement for the permutation $(3,1,2)$. Then for the permutation $(2,3,1)$ follows from 5.5.3.

The statement in (c) is done analogously. First, in Figure 30 we show that

$$\tau R \hat{f}_{cyc} R\imath_k(\hat{c},aa) = t_a^{-1} \tau \imath_k(\hat{a},ca).$$

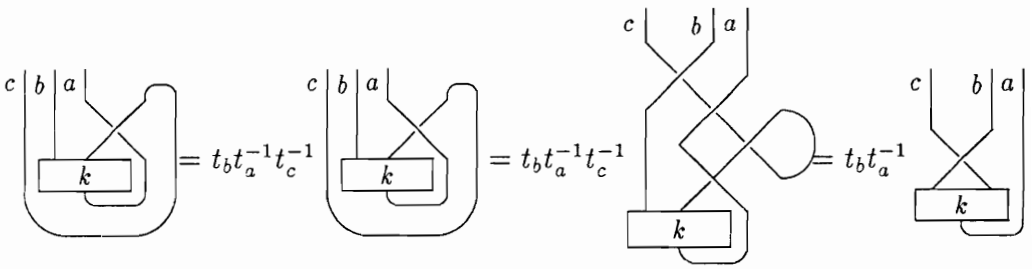


Figure 27: Computing $\mathtt{com}[\hat{b}, c, a]$ in the case $a > b > c$

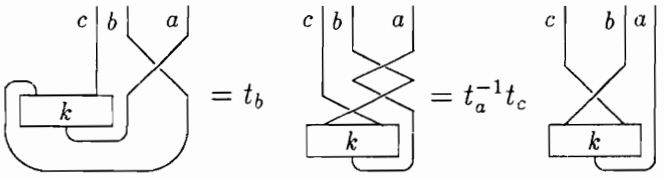


Figure 28: Computing $\mathtt{com}[\hat{c}, b, a]$ in the case $a > b > c$

From here, proposition 5.4.4 implies that $\mathtt{com}[\hat{c}, a, a]\mathtt{com}[\hat{b}, a, c] = t_a^{-1}\mathrm{Id}$. This proves the statement for the permutation $(1, 2, 3)$. Then for the permutation $(3, 1, 2)$ follows from 5.5.3.

In conclusion, $\mathtt{com}[a, b, c]$ need to be directly computed by evaluating the definition in 5.5.1 only when $b = c$. Then the corresponding composition of morphisms are evaluated on the highest weight vector of a to reduce the size of the computation.

5.5.5 To compute the associativity morphisms we first need to choose bases for the spaces $V(a, b \otimes (c \diamond d))$ and $V(a, (b \diamond c) \otimes d)$ (see 3.4.9). By definition, $b \diamond c = \oplus_{x \in \Sigma} V(x, bc) \otimes x$ and $c \diamond d = \oplus_{y \in \Sigma} V(y, cd) \otimes y$. Then, there are isomorphisms

$$V(a, (b \diamond c) \otimes d) \simeq \oplus_{x \in \Sigma} V(a, xd) \otimes V(x, bc),$$
$$V(a, b \otimes (c \diamond d)) \simeq \oplus_{y \in \Sigma} V(a, by) \otimes V(y, cd).$$

We define the basis for each piece $V(a, xd) \otimes V(x, bc)$ to be the one given by the lexicographical order of the product of the basic vectors of the two

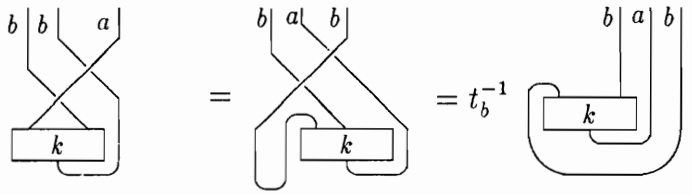


Figure 29: Computing $\mathtt{com}[\hat{b}, b, a]$ in the case $a > b$

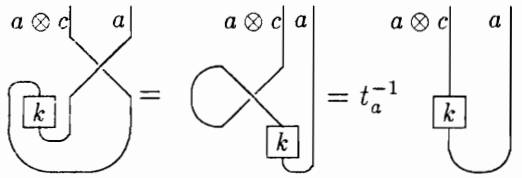


Figure 30: Computing $\mathtt{com}[\hat{a}, a, c]$ in the case $a > c$

spaces. This extends to a basis for $V(a, (b \diamond c) \otimes d)$ by fixing an order on the set $\{x \in \Sigma \mid V(a, xd)$ and $V(x, bc)$ are nonzero$\}$. In an anologous way, we define a basis for $V(a, b \otimes (c \diamond d))$. With respect to these bases asoc are represented by matrices with coefficients

$$\mathtt{asoc}[a, b, c, d]_{x,y,k,l,k',l'} = \pi_{l'}(by, a)(id_b \otimes \pi_k(cd, y))(\imath_l(x, bc) \otimes id_d)\imath_k(a, xd).$$

5.5.6 The programs, presented in this work, compute almost all associativity matrices by direct evaluation of the definition above [5]. The only exception are $\mathtt{asoc}[a, b, c, d]$ where some of a, b, c, d are equal to $\mathbf{1}$ and if $b, c, d > \mathbf{1}$, then they are not all the same. It is easy to see that if (b, c, d) is of the form $(\mathbf{1}, b', c')$, $(b', \mathbf{1}, c')$ or $(b', c', \mathbf{1})$, then $V(a, (b \diamond c)d) \simeq V(a, b'c')$, and $V(a, b(c \diamond d)) \simeq V(a, b'c')$, and with respect to these identifications, $\mathtt{asoc}[a, b, c, d] = \mathtt{id}$. Now, we want to evaluate $\mathtt{asoc}[\mathbf{1}, a, b, c]$ with $a, b, c \in \Sigma$ such that at least two of them are not the same. In this case, we have the isomorphisms

$$V(\mathbf{1}, (a \diamond a)c) = V(\mathbf{1}, \hat{c}c) \otimes V(\hat{c}, ab) \simeq V(\hat{c}, ab),$$

[5] For further developments see [31]

$$V(\mathbf{1}, a(b \diamond c)) = V(\mathbf{1}, a\hat{a}) \otimes V(\hat{a}, bc) \simeq V(\hat{a}, bc)$$

With respect to these identifications, the associativity morphisms are given exactly by $\hat{f}_{cyc}$. Then proposition 5.4.4 implies that when at least two of a, b, c are different, $\mathtt{asoc}[1, a, b, c] = \mathtt{id}$.

Chapter 6
Numerical aspects and conclusions

The work we present specifies a general class of tortile subcategories of the monoidal braided categories studied in [10], and describes algorithms for evaluating the associativity, commutativity and duality morphisms in these categories. The input data in these algorithms consists just of the name of a simple Lie algebra, a prime number greater than the Coxeter number of this Lie algebra, and possibly specifying if we want to restrict to the class0 category. The output consists of the following files:

Representation Data. File which contains data lists specifying the basis of each simple module, the action of the algebra generators on them, and the duality morphisms.

Summands Data. For any good triple (a, b, c), this file contains the dimension of the space $\hom_{\overline{\mathcal{C}}_p}(\hat{a}, bc)$, and the goodvectors for this spaces.

Symmetry Data. File which contains the matrices $\mathtt{com}[a, b, b]$ for any a, b different from $\mathbf{1}$.

Category Data. File which contains the matrices $\mathtt{asoc}[a, b, c, d]$ for any a, b, c, d different from $\mathbf{1}$.

To our knowledge, this is the first time such general exact computations have been attempted.

The algorithms have been implemented in Mathematica and tested for the categories $A_2, p = 5$, $C_2, p = 7$, $G_2, p = 11$, $A_3, p = 7$, and $A_4, p = 7$. Some data about those categories is listed bellow.

algebra	class0	prime	$\lvert\Sigma\rvert$	maxdim
A_2	No	5	6	8
A_3	Yes	7	5	45
A_4	No	7	15	75
C_2	Yes	7	3	10
G_2	Yes	11	5	64

Here "class0" indicates if we have restricted to ${\mathcal{C}_p}^0$ or not, $|\Sigma|$ is the number of simple modules on the category, and "maxdim" is the maximum dimension of these simple modules. We can say that generating the representation data, the goodvectors, and computing the necessary commutativity morphisms for these categories did't present any time or memory problem, but evaluating the associativity morphisms is very time consuming. Yet, as a result of the computations performed, we may conclude that, due to the careful choice of the bases of the modules and the spaces of homomorphisms, exact evaluation of the categorical data in question is possible[6].

There is two questions we need to address. The first one is why we consider these examples as a good enough test for the algorithms and the programs, and the second is what, in addition to going to a faster machine, can be done to make bigger categories accessible. With respect to the first question, we point out that the computations we discuss are exact, and their goal is the evaluation of algebraic objects with very unique properties. Each step of the computation can be, and is, carefully tested. In particular, we test if the action of the algebra generators on the simple modules satisfies the algebra relations, if the basic elements we find for $hom_{\overline{\mathcal{C}}_p}(a, bc)$ are actually homomorphisms of representations, and if $\mathtt{com}[a, b, b]^2 = t_a t_b^{-2}$. So, we consider all these tests on the generated examples as good enough for the algorithms. This doesn't imply that the programs are correct. But we claim that, because our computations are exact, any category generated with these programs can be tested extensively enough to be considered reliable.

We now would like to address the question of how to access bigger categories. A very serious step in this direction is done in [31]. There an algorithm is described, which uses the system of equations corresponding to the pentagon and hexagon diagrams to express all elementary associativity morphisms in terms of a minimum data. This minimum data consists of the elementary commutativity morphisms and a small number of associativities. In the context of [31], the programs described here can be used only to evaluate the minimum data, then the rest of the associativity morphisms would be found by solving the pentagon-hexagon system. This should enlarge significantly the size of the categories which can be computed. Another necessary

[6]Note that because the computation are exact, there is no insurance against huge numbers appearing in the process. Luckily, with respect to the chosen bases this is not the case

step is to rewrite big parts of the programs in C. Some of this has already been done, but not yet tested.

In conclusion, we would like to list two facts which have been observed in the calculations, and need to be studied further:

(i) If $a = \tilde{L}_\lambda$, $b = \tilde{L}_\nu$, $c = \tilde{L}_\mu$ then $\imath_k(a, bc) : {}_{\mathcal{R}}L_\lambda \to \frac{1}{p}{}_{\mathcal{R}}L_\nu \otimes {}_{\mathcal{R}}L_\mu$;

(ii) The entries of the matrices describing the associativity and commutativity morphisms are actually in $\mathcal{R}$.

References

[1] H.Andersen, P.Polo and W.Kexin, *Representations of quantum algebras*, Invent.Math. **104** (1991),1-59.

[2] M.F.Atiyah, *Topological quantum field theories*, Publ. Math. Inst. Hautes Etudes Sci. (Paris) **68** (1989),175-186.

[3] V.G.Drinfeld, *Hopf algebras and the quantum Yang-Baxter equation*, Soviet Math.Dokl. **32** (1985),254-258.

[4] V.G.Drinfeld, *Quantum groups*, Proc.Int.Congr.Math. Berkeley (1986),vol.1254-258.

[5] V.G.Drinfeld, *On almost cocomutative Hopf algebras*, Algebra and Analysis **1** (1989),30-46.

[6] D. Freed and R.Gompf, *Computer calculations of Witten's 3-manifold invariant*, Comm. Math. Physics **141** (1991),79-117.

[7] D. Freed and F.Quinn, *Chern-Simons theory with finite gauge group*, Comm. Math. Physics **156** (1993),435-472.

[8] W.Fulton and J.Harris, *Representation theory, A first course*, 1991 Springer-Verlag New York Inc.

[9] S.Garoufalidis, *On some aspects of Chern-Simons gauge theory*, PhD thesis, The University of Chicago (1992).

[10] S.Gelfand and D.Kazhdan, *Examples of tensor categories*, Invent.Math. **109** (1992),595-617.

[11] S.R.Hansen, *A q-analogue of the Kempf's vanishing theorem*, preprint, 1994.

[12] M.Jimbo, *The q-difference analogue of $U(g)$ and the Yang-Baxter equation*, Lett. Math. Phys. **11** (1985),247-252.

[13] L.C.Jeffrey, *On some aspects of Chern-Simons gauge theory*, PhD thesis, University of Oxford (1991).

[14] L.C.Jeffrey, *Chern-Simons-Wittens invariants of Lens spaces and torus bundles, and the semiclassical approximation*, preprint (1991).

[15] M.Kashiwara, *Crystallizing the q-analogue of universal enveloping algebras*, Comm.Math.Phys. **133** (1990),249-260.

[16] M.Kashiwara, *On crystal bases of the q-analogue of universal enveloping algebras*, Duke Math.J. **63** (1990),465-516.

[17] M.Kashiwara, *The crystal base and Littelmann's refined Demazure character formula*, Duke Math.J. **71** (1990),839-858.

[18] G.Kuiperberg, *Involutory Hopf algebras and 3-manifold invariants*, Internat.J.Math. **2** (1991),41-66.

[19] V.Lakshmibai, *Bases for quantum enveloping algebras II*, Proc. of Symp. Pure Math. **56** (1994),149-168.

[20] V.Lakshmibai, *Bases for quantum Demazure modules*, preprint (1995).

[21] V.Lakshmibai, C.Musili and C.S.Seshadri, *Cohomology of line bundles on G/B*, Ann.scient.Èc.Norm.Sup.4^e sèrie,t.7 (1974),89-138

[22] P.Littelmann, *Litlewood-Richardson rule for symmetrizable Kac-Moody algebras*, Inv.Math. **116** (1993),329.

[23] G.Lusztig, *Modular representations and quantum groups*, Contemp. Math. **82** (1989),59-77, Amer.Math.Soc., Providence, R.I..

[24] G.Lusztig, *Finite dimensional algebras arising from quantized universal enveloping algebras*, J.Amer.Math.Soc. **3** (1990),257-296.

[25] G.Lusztig, *Quantum groups at roots of 1*, Geom.Dedicata. **35** (1990),89-114.

[26] G.Lusztig, *Introduction to quantum groups*, Birkhäuser Boston, 1993.

[27] G.Lusztig, *Quantum deformations of certain simple modules over enveloping algebras*, Adv.Math. **70** (1988),237-249.

[28] G.Lusztig, *Canonical bases arising from quantized enveloping algebras*, J.Amer.Math.Soc. **3**(1990),447-498.

[29] G.Lusztig, *Canonical bases in tensor products*, Proc.Nat.Acad.Sci. **89** (1992),8177-8179.

[30] F.Quinn, *Lectures on Axiomatic Topological Quantum Field Theory*, LAS/Park City Mathematical Series, vol.**1**, 1995.

[31] F.Quinn and I.Bobtcheva, *Numerical presentations of tortile categories*, internal report.

[32] N.Yu.Reshetikhin and V.G.Turaev, *Ribon graphs and their invariants derived from quantum groups*, Comm.Math.Phys. **127** (1990),1-26.

[33] N.Yu.Reshetikhin and V.G.Turaev, *Invariants of 3-manifold via link polynomials and quantum groups*, Invent.Math. **127** (1991),547-597.

[34] Saavedra Rivano, *Categories Tannakiennes*, Lecture Notes in Math. **265**, Springer, Berlin Heidelberg New York (1972).

[35] M.Rosso, *Finite dimensional representations of the quantum analogue of a complex semisimple Lie algebra*, Comm.Math.Phys. **117** (1988),581-593.

[36] M.Rosso, *Quantum groups at roots of 1 and tangle invariants*, Topological and Geometrical Methods in Fied Theory, J.Mickelson and D.Pekonen, 347-58.

[37] Mei Chee Shum, *Tortile tensor categories*, Journal of Pure and Applied Algebra **93** Berlin Heidelberg New York (1994), 57-110.

[38] V.G.Turaev, *Quantum invariants of 3-manifolds*, Walter de Gryter **31** (1994).

[39] V.G.Turaev, O.Y.Viro, *State sum invariants of 3-manifolds and quantum 6j-symbols*, Topology **31** (1992),865-902.

[40] V.G.Turaev, H.Wenzl, *Quantum invariants of 3-manifolds associated with classical simple Lie algebras*, Internat.J.Math. **4** (1993),323-358.

[41] K.Walker, *On Witten's 3-manifold invariants*, preprint, (1991).

[42] E.H.Witten, *Quantum field theory and the Jones polynomial*, Comm. Math. Physics **121** (1989),351-399.

[43] D.N.Yetter, *State-sum invariants of 3-manifolds associated to artinian semisimple tortile categories*, Topology and Applications **58** (1994),47-80.

A AlgSetup

```
alg="A";
rank=2;
reports=True;

algcategory=alg<>ToString[rank]<>"q";
outputfile=algcategory<>".alg";
Off[General::spell1];
Off[General::spell];

announce[string_]:= If[reports, Print[string<>" at",
                        Date[][[4]],":",Date[][[5]]]];

(*generating the Cartan Matrix and the the
  Killing form on h* with normalization
  (alpha_1,alpha_1)=1,
  where x znd y are given in alpha-basis*)

CC=Table[If[(i==j+1)||(i==j-1),-1,
               If[i==j,2,0]],{i,rank},{j,rank}];

If[alg==="B", CC[[rank-1,rank]]=-2,

If[alg==="C", CC[[rank,rank-1]]=-2,

If[alg==="F"&&rank==4, CC[[2,3]]=-2,

If[alg==="G"&&rank==2, CC[[2,1]]=-3,

If[(alg==="D")&&(rank>=4), CC=ReplacePart[
       ReplacePart[CC,0,{{rank-1,rank},{rank,rank-1}}],
       -1,{{rank-2,rank},{rank,rank-2}}];
       rat[rank-1]=CC[[rank-2,rank]]/CC[[rank,rank-2]],

If[(alg==="E")&&(rank==6||rank==7), CC=ReplacePart[
```

```
       ReplacePart[CC,0,{{rank-1,rank},{rank,rank-1}}],
       -1,{{3,rank},{rank,3}}];
       rat[rank-1]=CC[[3,rank]]/CC[[rank,3]];
       form[rank-1]=1/(rat[1]*rat[2]*rat[rank-1]),

If[(alg==="E")&&(rank==8), CC=ReplacePart[
       ReplacePart[CC,0,{{rank-1,rank},{rank,rank-1}}],
       -1,{{5,rank},{rank,5}}];
       rat[rank-1]=CC[[5,rank]]/CC[[rank,5]];
       form[rank-1]=
           1/(rat[1]*rat[2]*rat[3]*rat[4]*rat[rank-1])

 ,If[alg=!="A",Print["Check the algebra."];
     Abort[]]]]]]]]];


rat[k_]:=CC[[k,k+1]]/CC[[k+1,k]];
form[al_]:=1/Product[rat[k],{k,1,al-1}];
Pairing=If[alg==="B"||alg==="F",4,2]*
        Table[form[j]*CC[[i,j]]/2,{i,rank},{j,rank}];
Kf[x_,y_]:=Sum[x[[i]]*y[[j]]*Pairing[[i,j]]
              ,{i,Length[x]},{j,Length[y]}];

(* Finding all positive roots by induction on
   increasing level starting with level one,
   which are the simple roots.*)

Posroots={IdentityMatrix[rank]};
Preprtable={Table[{i},{i,rank}]};
level=1;
While[Posroots[[-1]]=!={},
      curr=Posroots[[-1]]; curl={}; curt={};
      Do[beta=curr[[i]];
         Do[If[Not[MemberQ[curl,beta+Posroots[[1,j]]]],
            q=Sum[beta[[k]]*CC[[k,j]],{k,rank}]+1;
            If[(q<=0)||(((level-q)>0)&&
               MemberQ[Posroots[[level-q]],
```

```
                        beta-q*Posroots[[1,j]]])
                ,AppendTo[curl,beta+Posroots[[1,j]]];
                 AppendTo[curt,
                          Append[Preprtable[[-1,i]],j]] ]];
           ,{j,rank}];
         ,{i,Length[curr]}];
      AppendTo[Posroots,curl];
      AppendTo[Preprtable,curt]; level++];
Posroots=Flatten[Delete[Posroots,-1],1];
Preprtable=Flatten[Delete[Preprtable,-1],1];
Allroots=Join[Posroots,-Posroots];
lpr=Length[Posroots];
Do[numroot[Allroots[[i]]]=i,{i,2*lpr}];
numroot[x_]:=0;

Posrtable={};
Do[
   curroot=Preprtable[[k]];
   term={{{curroot[[1]]},1}};
   Do[s=1;
      While[s<=Length[term],
            tt=term[[s]];
            term[[s]]=
              {{Append[tt[[1]],curroot[[1]]],tt[[2]]},
               {Prepend[tt[[1]],curroot[[1]]],-tt[[2]]}};
            s++];
            term=Flatten[term,1]
     ,{1,2,Length[curroot]}];
   term=Sort[term];
   lt =Length[term];
   r=1; {pr,coef}=term[[1]]; newterm=If[lt>1,{},term];
   While[r<lt,
         r++;
         If[term[[r,1]]=!=pr ,
            If[coef=!=0,AppendTo[newterm,{pr,coef}]];
            {pr,coef}=term[[r]], coef+=term[[r,2]]];
         If[r==lt,AppendTo[newterm,{pr,coef}]]
```

```
      ];
   AppendTo[Posrtable,newterm]
  ,{k,Length[Preprtable]}];

rho=1/2*(Sum[Posroots[[i]],{i,Length[Posroots]}]);

(*Finding the fundamental weights*)

Fweights=Inverse[CC];

(*Finding gamma*)

smlength=2;

Do[root=Posroots[[i]]; test=1;
   Do[If[Kf[root,Fweights[[l]]]<0,test=0] ,{l,rank}];
   If[Kf[root,root]==smlength && test==1,gamma=root]
  ,{i,Length[Posroots]}];

Wmaxelem=
If[alg=="A",
   Flatten[Table[j,{i,rank},{j,rank-i+1,rank}]],

If[alg=="B",
   Flatten[Table[Join[Table[j,{j,rank-i+1,rank}]
                     ,Table[rank-j,{j,i-1}]]
                ,{i,rank}]],

If[alg=="C",Flatten[Join[
    Table[j,{i,rank-1},{j,rank-i,rank-1}]
   ,Reverse[Flatten[Join[Table[
     Table[If[OddQ[i],rank-j,rank-1-j],{j,0,i-1,2}],{i,rank}]
    ,Table[
       Table[If[OddQ[rank+i],rank-j,rank-1-j]
          ,{j,0,If[OddQ[rank+i],rank-i-1,rank-i-2],2}]
      ,{i,rank-1}]]]]]],
```

```
If[alg=="D", Flatten[
   Table[Join[Table[j,{j,rank-i,rank}]
             ,Table[rank-1-j,{j,i-1}]]
        ,{i,rank-1}]],

If[alg=="G",
   {2,1,2,1,2,1},

If[alg=="F",
   {4,3,4,2,3,4,2,3,2,1,2,3,2,4,1,3,2,3,1,4,2,3,2,1}]]]]]];


sgen[i_,j_]:=Posroots[[j]]-2*Pairing[[i,j]]/Pairing[[i,i]]*
             Posroots[[i]];

applyseq[seq_,j_]:=Module[{ls,curelem,i,newelem,coef},

ls=Length[seq];
curelem=Posroots[[j]];
Do[i=seq[[-k]];
   newelem=Table[0,{rank}];
   Do[coef=curelem[[s]];
      If[coef=!=0, newelem+=coef*sgen[i,s]]
     ,{s,rank}];
   curelem=newelem
  ,{k,ls}];
curelem];

W=Table[0,{rank}];
Do[W[[numroot[-applyseq[Wmaxelem,i]]]]=i, {i,rank}];

Save[outputfile, algcategory,CC,Fweights,rho,gamma,
     Pairing, Posroots, Wmaxelem,W];
```

B RmatrixSetup

```
alg="A";
rank=3;
algcategory=alg<>ToString[rank]<>"q";
inputfile=alg<>ToString[rank]<>"q"<>".alg";
outputfile=algcategory<>".roots";
reports=True;
Off[General::spell1];
SetDirectory[
"Macintosh HD:Mathematica 2.2.2:QuantumGroups:"<>algcategory];

announce[string_]:= If[reports, Print[string<>" at",
                        Date[][[4]],":",Date[][[5]]]];
announce["Started"];

If[FileNames[inputfile]=={},
   Print["Can't find the inputfile"];
   Abort[], Get[inputfile]];

(*functions which define ring operations on
  Q(v) with v-indeterminant*)

prod[poly1_,poly2_]:=(prepr={};

Do[AppendTo[prepr,{poly1[[io,1]]+poly2[[jo,1]],
                   poly1[[io,2]]*poly2[[jo,2]]}]
  ,{io,Length[poly1]},{jo,Length[poly2]}];

prepr=Sort[prepr]; lppr=Length[prepr];
result={}; pr=prepr[[1]]; term=pr[[1]];
io=1;
While[io<lppr, io++;
      If[prepr[[io,1]]==term, pr[[2]]+=prepr[[io,2]]
        , AppendTo[result,pr]; pr=prepr[[io]];
         term=pr[[1]]]];
```

```
AppendTo[result,pr];
result);

If[alg == "A",

numposr=rank*(rank+1)/2;
Posrlist={};
Do[AppendTo[Posrlist,
     Table[If[i<j||i>k,0,1],{i,rank}]]
  ,{k,rank},{j,k}];

PreRlist={}; i=rank; j=rank; Const={};
While[i>0,
      If[j==i,PrependTo[PreRlist,{{{i},{{0,1}}}}];
         PrependTo[Const,1],
         prevelem=PreRlist[[1]];
         PrependTo[PreRlist,
          Flatten[Table[{seq,{{exp,f}}}=prevelem[[s]];
           {{Append[seq,j],{{exp,f}}},
            {Prepend[seq,j],{{exp-1,-f}}}}
            ,{s,Length[prevelem]}],1]];
         AppendTo[Const,1]];
      If[j>1,j--,i--;j=i]]]000;

If[alg == "C",

Posrlist={Table[If[j==rank,1,0],{j,rank}]};
startl=Posrlist[[1]];
s=1;
While[s<=2*(rank-1), k=0;
      While[k<=If[EvenQ[s],s/2,(s-1)/2],
            If[rank-s+k>0, curel=startl;
               Do[curel[[l]]=If[l>=rank-k,2,1]
                 ,{l,rank-s+k,rank-1}];
               AppendTo[Posrlist, curel]]; k++];
     s++];
m=k=rank-1; zerol=Table[0,{rank}];
```

```
While[m>0, curel=zerol;
      Do[curel[[l]]=1,{l,k,m}];
      AppendTo[Posrlist,curel];
      If[k>1,k--,m--;k=m]];
Do[numpl[Posrlist[[k]]]=k, {k,Length[Posrlist]}];
PreRlist={{{{rank},{{0,1}}}}}; Const={1};
Do[               (*k over Posrlist*)
   curroot=Posrlist[[k]]; l=1; curconst={};
   While[curroot[[l]]==0,l++];
   If[curroot[[l]]==2,
     AppendTo[curconst,{2,1}];
     prevelem=
       PreRlist[[numpl[ReplacePart[curroot,0,l]]]];
     AppendTo[PreRlist, Flatten[
      Table[{seq,poly}=prevelem[[s]];
            {{Join[{l,l},seq],poly},
            {Join[{l},seq,{l}],
             prod[{{0,-1},{-2,-1}},poly]},
            {Join[seq,{l,l}],
             prod[{{-2,1}},poly]}}
            ,{s,Length[prevelem]}],1]]
    ,
     prevroot=ReplacePart[curroot,0,l];
     If[prevroot==zerol,
        AppendTo[PreRlist,{{{l},{{0,1}}}}]
      ,
        prevelem=PreRlist[[numpl[prevroot]]];
        If[curroot[[l+1]]==2 || l==rank-1,
           AppendTo[PreRlist, Flatten[
            Table[{seq,poly}=prevelem[[s]];
              {{Join[{l},seq],poly},
              {Join[seq,{l}],
               prod[{{-2,-1}},poly]}}
             ,{s,Length[prevelem]}],1]]
       ,
         AppendTo[PreRlist, Flatten[
           Table[{seq,poly}=prevelem[[s]];
```

```
                {{Join[{1},seq],poly},
                {Join[seq,{1}],
                 prod[{{-1,-1}},poly]}}
               ,{s,Length[prevelem]}],1]]]
]];
   If[curconst=={},AppendTo[Const,1],
                   AppendTo[Const,curconst]]
,{k,2,Length[Posrlist]}];

PreRlist=Reverse[Map[Map[MapAt[Reverse,#,{1}]&,#]&,PreRlist]];
Const=Reverse[Const];
Posrlist=Reverse[Posrlist];
];

If[alg=="D",
Posrlist={}; PreRlist={}; idmatrix=IdentityMatrix[rank];
Do[                    (* {k,1,rank} *)
   AppendTo[PreRlist,{{{k},{{0,1}}}}];
   AppendTo[Posrlist, idmatrix[[k]]];

   Do[AppendTo[PreRlist,commutator[PreRlist[[-1]],-1,i]];
      AppendTo[Posrlist, Posrlist[[-1]]+idmatrix[[i]]];
     ,{i,k+1,rank-1}];
If[k<rank-1,AppendTo[PreRlist,commutator[PreRlist[[-2]],-1,rank]];
     AppendTo[Posrlist, Posrlist[[-2]]+idmatrix[[rank]]];
    AppendTo[PreRlist,commutator[PreRlist[[-2]],-1,rank]];
    AppendTo[Posrlist, Posrlist[[-2]]+idmatrix[[rank]]]];
i=rank-2;
While[i>=k+1,
      AppendTo[PreRlist,commutator[PreRlist[[-1]],-1,i]];
      AppendTo[Posrlist, Posrlist[[-1]]+idmatrix[[i]]];
      i--];
,{k,rank}];

Const=Table[1,{Length[PreRlist]}];
PreRlist=Reverse[PreRlist];
Posrlist=Reverse[Posrlist];
```

```
];

If[alg=="G",

Posrlist=Reverse[{{0,1},{1,1},{3,2},{2,1},{3,1},{1,0}}];

Const=Reverse[{1,1,{{2,1},{2,2},{3,1}},{{2,1}},
       {{2,1},{3,1}},1}];

PreRlist={{{{2},{{0,1}}}},

       {{{1,2},{{0,1}}},{{2,1},{{-3,-1}}}},

       {{{1,1,1,2,2},{{-1,1},{1,1}}},
        {{2,2,1,1,1},{{-7,1},{-5,1}}},
        {{1,1,2,2,1},{{-4,1},{-2,1},{0,1}}},
        {{1,2,2,1,1},{{-6,1},{-4,1},{-2,1}}},
        {{1,2,1,1,2},
         {{-6,-1},{-4,-1},{-2,-1},{0,-1},{2,-1},{4,-1}}},
        {{2,1,1,2,1},{{-10,-1},{-8,-1},{-6,-1},{-4,-1},
          {-2,-1},{0,-1}}},
        {{2,1,1,1,2},{{-7,1},{-5,1},{-1,1},{1,1}}}},

       {{{1,1,2},{{0,1}}},{{1,2,1},{{-3,-1},{-1,-1}}},
        {{2,1,1},{{-4,1}}}},

       {{{1,1,1,2},{{0,-1}}},
        {{1,1,2,1},{{-3,1},{-1,1},{1,1}}},
        {{1,2,1,1},{{-4,-1},{-2,-1},{0,-1}}},
        {{2,1,1,1},{{-3,1}}}},

       {{{1},{{0,1}}}}};

PreRlist=
  Reverse[Map[Map[MapAt[Reverse,#,{1}]&,#]&,PreRlist]]];

Save[outputfile,Posrlist,PreRlist,Const];
```

C RepSetup

```
p=7;
alg="A";
rank=3;
class0=False;
algcategory=alg<>ToString[rank]<>"q";
category=alg<>ToString[rank]<>"q"<>"mod"<>ToString[p];
inputfile=alg<>ToString[rank]<>"q"<>".alg";
outputfile=category<>".rep";
statusfile=category<>".status";
class0rep=False;
allclassesrep=False;
reports=True;
computedimF=True;
dotests=False;


Off[General::spell1];
SetDirectory[
"Macintosh HD:Mathematica 2.2.2:QuantumGroups:"<>
        algcategory];
announce[string_]:= If[reports, Print[string<>" at",
                        Date[][[4]],":",Date[][[5]]]];
announce["Started"];

coxn=If[alg==="A",rank+1,

     If[alg==="B",2*rank,

     If[alg==="C",2*rank,

     If[alg==="F"&&rank==4,12,

     If[alg==="G"&&rank==2,6,
```

```
     If[alg==="D",2*rank-2,

     If[(alg==="E")&&(rank==6||rank==7||rank==8),
       If[rank==6,12,If[rank==7,18,30]]]]]]]]];

If[p<coxn,
   Print["The Coxeter number for this algebra
          is ",coxn,".", "Your prime should be
          bigger then it."]; Abort[]];

If[FileNames[inputfile]=={},
   Print["Can't find the inputfile"];
   Abort[], Get[inputfile]];

(*Killing form*)
Kf[x_,y_]:=Sum[x[[i]]*y[[j]]*Pairing[[i,j]]
              ,{i,Length[x]},{j,Length[y]}];

Sroots=IdentityMatrix[rank];
zero=Table[0,{p-1}];
one=Table[1,{p-1}];
id=Table[If[i==1,1,0],{i,1,p-1}];

prod[a_,b_]:= If[a==zero||b==zero,zero,
 If[a==id,b,If[b==id,a,
    Table[a[[1]]*b[[ke+1]] +
    Sum[a[[ie+1]]*(b[[ke-ie+1]]-b[[p-ie]]),{ie,1,ke}]-
    If[ke<p-2,a[[ke+2]]*b[[p-ke-1]],0] +
    Sum[a[[ie+1]]*(b[[p-ie+ke+1]]-b[[p-ie]])
       ,{ie,ke+2,p-2}]
   ,{ke,0,p-2}]]]];

inv[a_]:= Module[{GG}, If[a==id,id, If[a==zero,
  Print["inv has been called with zero argument."],
  GG= Table[ If[ia==1,a[[ka+1]],
       If[ia>1 && ia<=ka+1,a[[ka-ia+2]]-a[[p-ia+1]],
          If[ia==ka+2,-a[[p-ka-1]],
```

```
            If[ia>ka+2 && ia<=p-1
              ,a[[p-ia+ka+2]]-a[[p-ia+1]]]]]]
               ,{ka,0,p-2}, {ia,p-1}];
             Qtriang[Transpose[GG],True][[-1,1]]]] ];


vexp[x_]:=If[Mod[x,p]=!=p-1,
           ReplacePart[zero,1,Mod[x,p]+1],
           Table[-1,{p-1}]];

starp[scal_,matrix_]:=If[scal==id,matrix
                 ,Map[prod[scal,#]&,matrix,{-2}]];

dotp[A_,B_]:= Module[{do},

If[A===1,B,If[B===1,A, If[A==={}||B==={},{},
   do=Dimensions[A][[-2]];
   Map[Sum[starp[#[[i10]],B[[i10]]]
        ,{i10,do}]&,A,{-3}]]]]];

simplpoly[poly_]:=Module[{prep},

prep=Table[0,{p}];

Do[     (*over the length of poly*)
   prep[[Mod[poly[[ji,1]],p]+1]]+=poly[[ji,2]];
  ,{ji,Length[poly]}];
If[prep[[p]]==0, Drop[prep,-1],
   Drop[prep,-1]-prep[[-1]]*one]
];


braket[x_,i_]:=If[x==0,zero,If[x==1,id,

simplpoly[Table[{-Pairing[[i,i]]/2*(x-1-2*ze),1}
               ,{ze,0,Mod[x,p]-1}]]]];
```

```
Qtriang[qA_,doparam_]:=Module[
      {qnrows,qncols,qAA,qrow,qcol,qrr,
            qrang,qpointer,qra,qc0,qc1,qparam,
            qllo,qlle,qru,qmul,qde},
      {qnrows,qncols}={Length[qA],Length[qA[[1]]]};
      qAA=qA;
      If[doparam,qparam=IdentityMatrix[qnrows]];
      qpointer=Table[qi,{qi,qnrows}];
      qcol=0; qrow=0; qrang={};
      While[qcol<qncols && qrow<qnrows,
            qrow++; qcol++; qrr=qrow;
            While[qcol<=qncols &&
                  qAA[[qpointer[[qrr]],qcol]]==0,
                  If[qrr<qnrows,qrr++,qcol++;
                      qrr=qrow]];
            AppendTo[qrang,qcol];
            qra=qpointer[[qrr]];
            qpointer=Delete[Insert[qpointer,qra,qrow]
                        ,{qrr+1}];
            If[qcol<=qncols,
               If[qAA[[qra,qcol]]=!=1,
                  qc0=1/qAA[[qra,qcol]];
                  qAA[[qra]]=qc0*qAA[[qra]];
                  If[doparam,
                     qparam[[qra]]=qc0*qparam[[qra]]]];
               Do[qllo=qpointer[[qlle]];
                  qc1=qAA[[qllo,qcol]];
                  If[qc1=!=0,
                    qAA[[qllo]]=
                        qAA[[qllo]]-qAA[[qra]]*qc1;
                    If[doparam,qparam[[qllo]]=
                     qparam[[qllo]]-qparam[[qra]]*qc1]]
                 ,{qlle,qrow+1,qnrows}]];
     ];

    If[qcol<=qncols,qmul=qrow,qmul=qrow-1;
```

```
                                qrang=Drop[qrang,-1]];
     If[qmul>0, qAA=Table[qAA[[qpointer[[qi]]]]
       ,{qi,qmul}];
        If[doparam,
           qparam=Table[qparam[[qpointer[[qi]]]]
                       ,{qi,qmul}]];
        qru=qmul;
        While[qru>1,
           Do[qde=qAA[[qllo,qrang[[qru]]]];
              If[qde=!=0,
                 qAA[[qllo]]=
                   qAA[[qllo]]-qde*qAA[[qru]];
                 If[doparam,qparam[[qllo]]=
                   qparam[[qllo]]-qde*qparam[[qru]]]]
             ,{qllo,qru-1}]; qru--];
        If[doparam,{qmul,qAA,qrang,qparam}
                  ,{qmul,qAA,qrang}]
     ,{0}]
];


Triang[A_,doparam_]:=Module[

{nrows,ncols,AA,param,pointer,col,row,rang,rr,
 ra,c0,llo,c1,lle,mul,ru,de},

      {nrows,ncols}={Length[A],Length[A[[1]]]};
      AA=A;
      If[doparam,param=Table[If[i==j,id,zero]
                            ,{i,nrows},{j,nrows}]];
      pointer=Table[i,{i,nrows}];
      col=0; row=0; rang={};
      While[col<ncols && row<nrows,
            row++; col++; rr=row;
            While[col<=ncols &&
                  AA[[pointer[[rr]],col]]==zero,
                  If[rr<nrows,rr++,col++; rr=row]];
```

```
        AppendTo[rang,col];
        ra=pointer[[rr]];
        pointer=Delete[Insert[pointer,ra,row]
                       ,{rr+1}];
        If[col<=ncols,
           If[AA[[ra,col]]=!=id,
              c0=inv[AA[[ra,col]]];
              AA[[ra]]=starp[c0,AA[[ra]]];
              If[doparam,
                 param[[ra]]=
                  starp[c0,param[[ra]]]];
           Do[llo=pointer[[lle]];c1=AA[[llo,col]];
              If[c1=!=zero,
               AA[[llo]]=
                 AA[[llo]]-starp[c1,AA[[ra]]];
               If[doparam,param[[llo]]=
               param[[llo]]-starp[c1,param[[ra]]]]]
             ,{lle,row+1,nrows}]];
 ];

If[col<=ncols,mul=row,mul=row-1;
   rang=Drop[rang,-1]];
If[mul>0, AA=Table[AA[[pointer[[i]]]],{i,mul}];
   If[doparam,
      param=Table[param[[pointer[[i]]]]
                  ,{i,nrows}]];
   ru=mul;
   While[ru>1,
      Do[de=AA[[llo,rang[[ru]]]];
         If[de=!=zero,
           AA[[llo]]=AA[[llo]]-starp[de,AA[[ru]]];
            If[doparam,param[[llo]]=
             param[[llo]]-starp[de,param[[ru]]]]]
        ,{llo,ru-1}]; ru--];
      If[doparam,{mul,nrows-mul,AA,rang,param}
        ,{mul,nrows-mul,AA,rang}]
,{0}]
```

```
];


(*generating the representations in the first
  Weyl chamber. The hweights are given in omega basis*)

AllRep={};
up=Table[
   Kf[p*gamma-2*rho,gamma]/(2*Kf[Fweights[[i]],gamma])
        ,{i,rank}];
gg[0,l_]=0;
gg[n_,l_]:=(bry=up[[n]]*(1-Sum[l[[j]]/up[[j]]
                              ,{j,1,n-1}]);
   If[l[[n]]+1<bry,
       ReplacePart[ReplacePart[l,l[[n]]+1,n],0,
                   Table[{ff},{ff,n+1,Length[l]}]]
     ,gg[n-1,l]]
);

hw=Table[0,{rank}];
While[gg[rank,hw]=!=0,
      hw=gg[rank,hw];
      alhw=Sum[hw[[kk]]*Fweights[[kk]],{kk,rank}];
      lev=Sum[alhw[[k]],{k,rank}];
      If[IntegerQ[lev],
        AppendTo[AllRep,Join[{lev},hw]]]
     ];
AllRep=Sort[AllRep];
repno=Length[AllRep];

AllRep=Map[Rest,AllRep];

If[AllRep=!={},
   RepA=AllRep.Fweights;
   nRep0={}; nRep1={};
   Do[If[VectorQ[RepA[[i]],IntegerQ],
```

```
          AppendTo[nRep0,AllRep[[i]]],
          AppendTo[nRep1,AllRep[[i]]]]
       ,{i,repno}]
  , nRep0={}; nRep1={}];

AllRep=Join[nRep0,nRep1];
numrep0=Length[nRep0];
repno=Length[AllRep];
Do[num[AllRep[[n]]]=n
   ,{n,repno}];
num[x_]:=0;
Dual=Table[hwt=Table[AllRep[[n,W[[i]]]],{i,rank}];
           num[hwt], {n,repno}];

(* Check status file *)

If[FileNames[statusfile]=={},repdone=0;
        Save[statusfile,repno, numrep0,
             repdone, class0rep, allclassesrep]];
Get[statusfile];

If[(class0 && class0rep) || allclassesrep
  ,Print["According to the status file the data
          is already complete"]];

(*save Dual and Rep if the datafile is new *)

If[repdone==0,Save[outputfile,category,p,Pairing,
                   CC,Fweights,AllRep,repno,numrep0,
                   nRep0,nRep1, Dual]];

announce["Finished setup"];

Print["The category has ",
       If[class0,numrep0,repno],
          " irreducibles. There is ",If[class0,
        If[class0rep,0,numrep0-repdone],repno-repdone],
```

```
        " left to compute."];


(* function called to update data file *)

RepUpdate[]:=(

        Save[outputfile,Basis,maxelem,Wts,Mult,Action,
             Dmorph];
        If[computedimF, Save[outputfile,summands]];
        repdone++;
        If[reports,
           Print["Representation ",AllRep[[repdone]],
                         " finished at ",
                Date[][[4]],":",Date[][[5]]]];
        If[repdone==numrep0, class0rep=True];
        If[repdone==repno, allclassesrep=True];
        DeleteFile[statusfile];
        Save[statusfile,category,repno,numrep0,repdone,
            class0rep,
            allclassesrep, PermRep,sumdone,
            class0sum,allclassessum,goup,
            piecesaved, class0cat,
            allclassescat,statustable];
        Clear[Wts, Mult];Clear[Action];Clear[Dmorph];
        Clear[Basis,maxelem,summands];
        ); (*end RepUpdate *)


ordertest[x_,y_]:=If[x>=y,Border[[x,y]],
                     -Border[[y,x]]];

refl[i0_,wt_]:=wt-wt[[i0]]*CC[[i0]];

FindqW[n_]:=Module[

{qW,lme,i0,l,curelem,pres,newelem,
```

```
 elemno,elemno0,lquotW},

Clear[quotW,qWorder, maxelem,posqW,dn,numqW];
hwt=AllRep[[n]];dn=Dual[[n]];
qW={{0,hwt,{}}};
lme=Length[Wmaxelem];
numqW[hwt]=1; elemno=1; numqW[x_]:=0;

Do[
  i0=Wmaxelem[[i]]; elemno0=elemno;
  Do[
    {l,curelem,pres}=qW[[j]];
    If[curelem[[i0]]>0,
      newelem=refl[i0,curelem];
      If[numqW[newelem]==0,
        AppendTo[qW,{l+1,newelem,Append[pres,i0]}];
        elemno++;
        numqW[newelem]=elemno]
]
    ,{j,elemno0}]
  ,{i,lme}];

qW=Sort[qW]; lquotW=Length[qW];
maxelem[n]=qW[[-1,3]];
quotW[n]=Table[qW[[i,2]],{i,lquotW}];

If[n=!=dn, quotW[dn]={};
   Do[AppendTo[quotW[dn],
       Table[quotW[n][[i,W[[j]]]],{j,rank}]]
      ,{i,lquotW}];
   maxelem[dn]=Reverse[maxelem[n]]];

Do[posqW[n][quotW[n][[i]]]=i;
   If[n=!=dn,posqW[dn][quotW[n][[i]]]=i],{i,lquotW}];
];

posqW[n_][x_]:=0;
```

```
FindParam[path_,i0_]:=Module[

{values,valueat1,ordervalues,Q,P},

values=Table[{path[[j,i0]] ,j}
           ,{j,Length[path]}];
valueat1=values[[-1,1]];
ordervalues=Sort[values];
Q=ordervalues[[1,1]];
P=valueat1-Q; pp=1;
While[ordervalues[[pp,1]]==Q,pp++];
If[P<1,xx={},xx=ordervalues[[pp-1,2]];
   While[values[[xx,1]]<Q+1,xx++]];
{Q,P,ordervalues[[pp-1,2]]
     ,If[xx===={},{},xx],If[xx==={},{},values[[xx,1]]]}
];



Fpath[path_,i0_,Q_,P_,pp_,x_,xi0_]:=Module[

{newpath,a},

If[P<=0,newpath={},
   If[xi0==Q+1,
      If[pp>1 && refl[i0,path[[1,pp]]]==path[[1,pp-1]],
         newpath=MapAt[refl[i0,#]&,
                   Delete[path,{{1,pp},{2,pp}}]
                  ,Table[{1,k},{k,pp,x-2}]]
        ,    (*pp=0 ||
               refl[i0,path[[1,pp]]]<path[[1,pp-1]]*)
         newpath=MapAt[refl[i0,#]&,path,
                    Table[{1,k},{k,pp,x-1}]]]
      ,        (*if xi0>Q+1*)
       a=(Q+1-
```

```
          Sum[(path[[2,i+1]]-path[[2,i]])*path[[1,i,i0]]
                ,{i,x-2}])/path[[1,x-1,i0]] +
          path[[2,x-1]];

        If[pp>1 &&
           refl[i0,path[[1,pp]]]==path[[1,pp-1]],
           newpath=MapAt[refl[i0,#]&,
            Insert[Delete[path,{{1,pp},{2,pp}}]
                  ,path[[1,x-1]],{1,x-1}]
            ,Table[{1,k},{k,pp,x-2}]];
           newpath=Insert[newpath,a,{2,x-1}]
          ,     (*pp=0 ||
                refl[i0,path[[1,pp]]]<path[[1,pp-1]]*)
           newpath=MapAt[refl[i0,#]&,
            Insert[path,path[[1,x-1]],{1,x-1}]
           ,Table[{1,k},{k,pp,x-1}]];
           newpath=Insert[newpath,a,{2,x}]]]];
newpath];

(*
 checks if the path of shape AllRep[[l]]
 defines a summand of k.l using the Littlewood
 decomposition rule
*)


CheckSummand[path_,k_,l_]:=Module[

{lph,shiftpath,test},

lph=Length[path]; kwt=AllRep[[k]];
shiftpath=path+Table[kwt,{lph}];
test=True;
Do[If[shiftpath[[i,j]]<0,test=False]
  ,{i,lph},{j,rank}];
{test,num[shiftpath[[-1]]]}];
```

```
FindBasis[n_]:=Module[

{hwt,lme,Prebasis,curelem,i0,lpb,lev,wt,vct,newpath,
 Q,P,pp,xx,xi0,test,nwt,nwtno,checkno,newvct},
Clear[numprewts];

lme=Length[maxelem[n]];
hwt=AllRep[[n]];
Prebasis={{-Apply[Plus,hwt.Fweights],hwt,{{-1
          ,Table[0,{lme}],{{hwt},{0,1}}
          ,{Table[0,{rank}],hwt},0}}}};
numprewts[hwt]=1; numprewts[x_]:=0;
curelem=hwt;
If[computedimF,Do[summands[m,n]={},{m,n,repno}];
  Do[{test,smd}=
  CheckSummand[{Table[0,{rank}],hwt},m,n];
     If[test && smd=!=0,
        AppendTo[summands[m,n],smd]]
   ,{m,n,repno}]];
Do[          (*kk over the length of maxelem*)
   i0=maxelem[n][[kk]];
   curelem=refl[i0,curelem];
   curlpb=lpb=Length[Prebasis];
   curmlt=Map[Length[#[[3]]]&,Prebasis];
   Do[
      {lev,wt}=Prebasis[[mm,{1,2}]];
    If[wt[[i0]]>0,
     Do[
       vct=Prebasis[[mm,3,ml]];
       {Q,P,pp,xx,xi0}=FindParam[vct[[4]],i0];
       If[Q>=0 && P>0,
        newpath=Fpath[vct[[3]],i0,Q,P,pp,xx,xi0];
        test=True;
        nwt=wt-CC[[i0]];
        nwtno=numprewts[nwt];
        If[nwtno=!=0, jj=1;
```

```
          checkno=Length[Prebasis[[nwtno,3]]];
          While[test && jj<=checkno,
             If[Prebasis[[nwtno,3,jj,3]]==newpath,
                test=False];jj++]];
       If[test, Do[
         corners={Table[0,{rank}]};
         Do[ (*over the corners of the newpath*)
           AppendTo[corners,
            Sum[(newpath[[2,i+1]]-newpath[[2,i]])*
                  newpath[[1,i]],{i,j-1}]+
            (newpath[[2,j+1]]-newpath[[2,j]])*
             newpath[[1,j]] ]
         ,{j,Length[newpath[[1]]]}];
         newvct={-posqW[n][newpath[[1,1]]]
                ,ReplacePart[vct[[2]],s,lme-kk+1]
                ,newpath,corners,-kk};
         If[nwtno==0,
            AppendTo[Prebasis,
                     {lev+s,nwt,{newvct}}];
            curlpb++; numprewts[nwt]=curlpb
          ,
            Prebasis[[nwtno,3]]=
               AppendTo[Prebasis[[nwtno,3]],newvct]];
         If[computedimF,
           Do[{test,smd}=CheckSummand[corners,m,n];
              If[test && smd=!=0,
                 AppendTo[summands[m,n],smd]]
             ,{m,n,repno}]];
         {Q,P,pp,xx,xi0}=FindParam[corners,i0];
         newpath=Fpath[newpath,i0,Q,P,pp,xx,xi0];
         nwt=nwt-CC[[i0]];
         nwtno=numprewts[nwt]
        ,{s,wt[[i0]]}] ]]
     ,{ml,curmlt[[mm]]}]],{mm,lpb}]
  ,{kk,lme}];

Prebasis=Sort[Map[MapAt[Sort,#,{3}]&,Prebasis]];
```

```
Wts[n]=Map[#[[2]]&,Prebasis];
Basis[n]=Table[Map[Reverse[
           Take[#[[2]],#[[5]]]]&,Prebasis[[i,3]]]
          ,{i,Length[Prebasis]}];
Mult[n]=Map[Length[#]&,Basis[n]];
Clear[Prebasis];
Do[numwts[n][Wts[n][[k]]]=k;
   Do[numbase[n][Basis[n][[k,i]]]={k,i}
    ,{i,Mult[n][[k]]}]
  ,{k,Length[Wts[n]]}]
];

numwts[n_][x_]:=0; numbase[n_][x_]:=0;


FindAction[n_]:=Module[

{lme,Preaction,wnum1,wnum2,dim,nr,mult,Eact,vct,prf,
 eo,k1,j1,f,k2,curact,mult1,y,syst,mul,dimker,AA,rang},

   Clear[Action];

   lme=Length[maxelem[n]];
   Preaction={Table[{},{i,2*rank}]};

   Do[             (*fix the k'th weight space*)

   (*wnum1[[i]] gives the number of the weight's
     space where the i'th positive simple root
     applied to the k'th weight space goes;

     wnum2[[i,j]] gives the number of the weight's
     space where the sum of the i'th and j'th
     positive simple roots
     applied to the k'th weight space goes;

     dim[[i]] is the dimention of the wnum1[[i]]'th
```

```
  weight space if wnum[[i]]=!=0, and 0 otherwise*)

wnum1=Table[numwts[n][Wts[n][[k]]+CC[[i]]]
            ,{i,rank}];

wnum2=Table[numwts[n][Wts[n][[k]]+CC[[i]]+CC[[j]]]
            ,{i,rank},{j,rank}];

dim=Table[If[wnum1[[i]]>0,Mult[n][[wnum1[[i]]]],0]
         ,{i,rank}];
nr=Apply[Plus,dim]; mult=Mult[n][[k]];

Eact=Table[{},{2*rank}];

Do[        (*j over the elements in Bases[n][[k]]*)
   vct=Basis[n][[k,j]];
   prf=Length[vct]; mm=vct[[-1]];
   If[mm>1,vct[[-1]]--,
      eo=-1; While[-eo<prf && vct[[eo-1]]==0,eo--];
      vct=Drop[vct,eo]];
   {k1,j1}=numbase[n][vct];
   f=maxelem[n][[prf]];

   Do[     (*ii over the simple roots*)

     If[wnum1[[ii]]=!=0,
     k2=numwts[n][Wts[n][[k1]]+CC[[ii]]];
     AppendTo[Eact[[ii]],starp[inv[braket[mm,f]],
      If[ii=!=f,
        If[k2=!=0,dotp[Preaction[[k1,ii]]
                      ,Preaction[[k2,rank+f]]][[j1]]
          ,Table[zero,{dim[[ii]]}]]
       ,curact=If[k2=!=0,dotp[Preaction[[k1,ii]]
                    ,Preaction[[k2,rank+f]]][[j1]]
                  ,Table[zero,{dim[[ii]]}]];
        curact[[j1]]+=braket[Wts[n][[k1,f]],f];
        curact]]]]
```

```
      ,{ii,rank}]
    ,{j,mult}];

Do[k1=wnum1[[ii]];          (*ii simple roots*)

   If[k1=!=0,
      mult1=Mult[n][[k1]];

    y=Table[     (*ll over simple roots*)
      k2=wnum2[[ii,ll]]; kk1=wnum1[[ll]];
      If[ll=!=ii,If[kk1==0,{}, If[k2=!=0,
      dotp[Preaction[[k1,ll]],
           Preaction[[k2,rank+ii]]]
      ,Table[zero,{mult1},{dim[[ll]]}]]]
     ,
      If[k2=!=0,
      dotp[Preaction[[k1,ll]],
           Preaction[[k2,rank+ii]]]]+
      starp[braket[Wts[n][[k1]][[ii]],ii],
             Table[If[i==j,id,zero],{i,dim[[ii]]}
                   ,{j,dim[[ii]]}]]
     ,
      starp[braket[Wts[n][[k1]][[ii]],ii],
             Table[If[i==j,id,zero],{i,dim[[ii]]}
                   ,{j,dim[[ii]]}]]]]
      ,{ll,rank}];
     syst={};
     Do[
      (*ll over simple roots*)
        If[wnum1[[ll]]=!=0,
          syst=Join[syst,Transpose[
             Join[Eact[[ll]],y[[ll]]]]]]
        ,{ll,rank}];
     {mul,dimker,AA,rang}=
                Triang[syst,False];
     If[rang=!=Table[i,{i,mult}],
        Print["Problem with F on",{k,ii}];Abort[]
```

```
          ,
           Preaction[[k1,ii+rank]]=
             Take[Transpose[AA],-mult1]]
]
       ,{ii,rank}];


  AppendTo[Preaction,Eact];
  ,{k,2,Length[Wts[n]]}];

 Action[n]=Preaction; Clear[Preaction] ];


FindDmorph[n_]:=Module[

{dn,lme,lw,curmorph,vct,prf,eo,k1,j1,f,targetk1,
 targetk,symtest},

Clear[dn,Dmorph];
dn=Dual[[n]]; hwt=AllRep[[n]];
lme=Length[maxelem[n]];
Dmorph[n]={{{id}}};
lw=Length[Wts[n]];

Do[             (*fix the k'th weight space*)
     curmorph={};
     Do[   (*j over the elements in Bases[n][[k]]*)
        vct=Basis[n][[k,j]];
        prf=Length[vct]; mm=vct[[-1]];
        If[mm>1,vct[[-1]]--,
          eo=-1; While[-eo<prf && vct[[eo-1]]==0,eo--];
          vct=Drop[vct,eo]];
        {k1,j1}=numbase[n][vct];
        f=maxelem[n][[prf]];
        targetk1=numwts[dn][-Wts[n][[k1]]];
        targetk=numwts[dn][-Wts[n][[k]]];
        AppendTo[curmorph,
```

```
          -starp[prod[vexp[(Wts[dn][[targetk1,f]]+2)*
                  Pairing[[f,f]]/2],inv[braket[mm,f]]],
           dotp[Dmorph[n][[k1,j1]],
             Transpose[Action[dn][[targetk,rank+f]]]]]]
        ,{j,Length[Basis[n][[k]]]}];
      AppendTo[Dmorph[n],curmorph]
     ,{k,2,lw}];

If[n==dn, symtest=False;
   Do[If[Dmorph[n][[k]]=!=
         starp[vexp[-Sum[(Wts[n][[k]].Fweights)[[i]]
                     *Pairing[[i,i]],{i,rank}]],
            Transpose[Dmorph[n][[lw-k+1]]]],
       symtest=True ],{k,lw}];
   If[symtest,
Print["The ",n,"'th representation is not symmetric."]
]]];



(* ** MAIN LOOP ** *)
Clear[Wts,Mult,Basis,summands,maxelem,
      Dmorph,quotW,qWorder];
Clear[Action];
While[repdone<If[class0,numrep0,repno], n=repdone+1;
      If[Dual[[n]]>=n,
         announce["Started n="<>ToString[n]];
         FindqW[n];
         announce["Finished qW"];
         FindBasis[n]; announce["Finished Basis"];
         If[n=!=Dual[[n]],FindBasis[Dual[[n]]]];
         announce["Finished dual Basis"];
         FindAction[n];
         announce["Finished Action"];
         FindDmorph[Dual[[n]]];
         announce["Finished Dmorph"]];
         RepUpdate[]];
```

```
(* test programs *)

If[dotests,
  Get[outputfile];
  Clear[numwts];
  Do[Do[numwts[n][Wts[n][[k]]]=k, {k,Length[Wts[n]]}]
   ,{n, Length[AllRep]}];
  numwts[n_][wt_]:=0;

 ran[n_][k_,j_]:=numwts[n][Wts[n][[k]]+
         If[j>rank,-CC[[j-rank]],CC[[j]]] ]];


TestAction[n_]:=(problem=False;

Do[              (*k over the wtspaces *)
   ewnum=Table[numwts[n][Wts[n][[k]]+CC[[i]]]
               ,{i,rank}];

   efwnum=Table[numwts[n][Wts[n][[k]]+CC[[i]]-CC[[j]]]
               ,{i,rank},{j,rank}];

   fwnum=Table[numwts[n][Wts[n][[k]]-CC[[i]]]
               ,{i,rank}];

   dim=Table[If[efwnum[[i,j]]>0,
             Mult[n][[efwnum[[i,j]]]]
               ,0],{i,rank},{j,rank}];

  Do[dd=dim[[ii,jj]];
     If[dd>0,
        mzero=Table[zero,{Mult[n][[k]]},{dd}];
```

```
        test = If[ewnum[[ii]]>0,
                dotp[Action[n][[k,ii]],
                     Action[n][[ewnum[[ii]],jj+rank]]
                      ,mzero]-
            If[fwnum[[jj]]>0,
              dotp[Action[n][[k,jj+rank]],
                  Action[n][[fwnum[[jj]],ii]]],mzero]+
            If[ii==jj,starp[
               braket[Wts[n][[efwnum[[ii,jj]],ii]],ii],
                Table[If[i==j,id,zero],{i,dd},{j,dd}]]
               ,mzero];
    If[test=!=mzero,
       Print["Problem with the ",{ii,jj},
             "commutator on the ",k,"'th wtspace."];
       problem=True]
];
  ,{ii,rank},{jj,rank}]
,{k,Length[Mult[n]]}];
If[!problem, Print["Action[",n,"] is fine."]]);



Testsdual[n_]:=(problem=False;

lw=Length[Wts[n]];
     If[Dual[[n]]==n,
        Do[If[Dmorph[n][[k]]=!=
              starp[vexp[
               -Sum[(Wts[n][[k]].Fweights)[[i]]
                    *Pairing[[i,i]]
                   ,{i,rank}]],
                      Transpose[Dmorph[n][[lw-k+1]]]],
              problem=True;
              Print["The normalization at ",k,
                    "'th wtspace fails."]];
           lolev={};
           Do[l=ran[n][k,j];
```

```
              If[l>0,AppendTo[lolev,{j,l}]]
             ,{j,rank+1,2*rank}];
           Do[
            {j,l}=lolev[[s]];
            FS=dotp[Action[n][[k,j]],Dmorph[n][[l]]];
            SF=starp[-vexp[(Wts[n][[lw-k+1,j-rank]]+2)*
                           Pairing[[j-rank,j-rank]]/2],
                  dotp[Dmorph[n][[k]],
                  Transpose[Action[n][[lw-l+1,j]]]]];
            If[FS=!=SF,
               Print["Problem with the ",k,
                    "th wtspace of the ",n,"th rep."];
                 problem=True ]
             ,{s,Length[lolev]}]
           ,{k,lw}] ];
If[!problem, Print["Dmorph[",n,"] is fine."]];
);


(*main loop for the test functions*)

If[dotests,
  Do[If[n<=Dual[[n]],
     TestAction[n];Testsdual[Dual[[n]]]]
 ,{n,If[class0,numrep0,repno]}]];
```

D SummandsSetup

```
p=7;
alg="A";
rank=3;
class0=False;
algcategory=alg<>ToString[rank]<>"q";
category=alg<>ToString[rank]<>"q"<>"mod"<>ToString[p];
inputfile=category<>".rep";
rootsfile=alg<>ToString[rank]<>"q.roots";
predimfile=category<>".predim";
outputfile=category<>".sum";
symmfile=category<>".symm";
statusfile=category<>".status";
class0sum=False;
allclassessum=False;
reports=True;
dimFdone=False;
dotests=False;


Off[General::spell1];
Off[General::spell];
SetDirectory[
"Macintosh HD:Mathematica 2.2.2:QuantumGroups:"<>algcategory];

     (* setup ring operations and linear algebra *)

zero=Table[0,{p-1}];
one=Table[1,{p-1}];
id=Table[If[i==1,1,0],{i,1,p-1}];

prod[a_,b_]:=

If[a==zero||b==zero,zero,
 If[a==id,b,If[b==id,a,
```

```
    Table[a[[1]]*b[[ke+1]] +
    Sum[a[[ie+1]]*(b[[ke-ie+1]]-b[[p-ie]]),{ie,1,ke}] -
    If[ke<p-2,a[[ke+2]]*b[[p-ke-1]],0] +
    Sum[a[[ie+1]]*(b[[p-ie+ke+1]]-b[[p-ie]])
       ,{ie,ke+2,p-2}]
   ,{ke,0,p-2}]]]];

inv[a_]:= Module[{GG}, If[a==id,id, If[a==zero,
  Print["inv has been called with zero argument."],
  GG= Table[ If[ia==1,a[[ka+1]],
       If[ia>1 && ia<=ka+1,a[[ka-ia+2]]-a[[p-ia+1]],
          If[ia==ka+2,-a[[p-ka-1]],
           If[ia>ka+2 && ia<=p-1
             ,a[[p-ia+ka+2]]-a[[p-ia+1]]]]]]
              ,{ka,0,p-2}, {ia,p-1}];
            Qtriang[Transpose[GG],True][[-1,1]]] ];

vexp[x_]:=If[Mod[x,p]=!=p-1,
           ReplacePart[zero,1,Mod[x,p]+1],
           Table[-1,{p-1}]];

starp[scal_,matrix_]:=If[scal==id,matrix
                        ,Map[prod[scal,#]&,matrix,{-2}]];

dotp[A_,B_]:= Module[{do},

If[A===1,B,If[B===1,A, If[A==={}||B==={},{},
   do=Dimensions[A][[-2]];
   Map[Sum[starp[#[[i10]],B[[i10]]],{i10,do}]&,A,{-3}]]]]];

simplpoly[poly_]:=Module[{prep},

prep=Table[0,{p}];

Do[     (*over the length of poly*)
   prep[[Mod[poly[[ji,1]],p]+1]]+=poly[[ji,2]];
  ,{ji,Length[poly]}];
```

```
If[prep[[p]]==0, Drop[prep,-1],
   Drop[prep,-1]-prep[[-1]]*one]
];


braket[x_,ip_]:=If[x==0,zero,If[x==1,id,

simplpoly[Table[{-Pairing[[ip,ip]]/2*(x-1-2*ze),1}
               ,{ze,0,Mod[x,p]-1}]]]];

Qtriang[qA_,doparam_]:=Module[
      {qnrows,qncols,qAA,qrow,qcol,qrr,
         qrang,qpointer,qra,qc0,qc1,qparam,qllo,qlle,
         qru,qmul,qde},
      {qnrows,qncols}={Length[qA],Length[qA[[1]]]};
      qAA=qA;
      If[doparam,qparam=IdentityMatrix[qnrows]];
      qpointer=Table[qi,{qi,qnrows}];
      qcol=0; qrow=0; qrang={};
      While[qcol<qncols && qrow<qnrows,
            qrow++; qcol++; qrr=qrow;
            While[qcol<=qncols &&
                  qAA[[qpointer[[qrr]],qcol]]==0,
                  If[qrr<qnrows,qrr++,qcol++; qrr=qrow]];
            AppendTo[qrang,qcol];
            qra=qpointer[[qrr]];
            qpointer=Delete[Insert[qpointer,qra,qrow]
                           ,{qrr+1}];
            If[qcol<=qncols,
               If[qAA[[qra,qcol]]=!=1,
                  qc0=1/qAA[[qra,qcol]];
                  qAA[[qra]]=qc0*qAA[[qra]];
                  If[doparam,
                     qparam[[qra]]=qc0*qparam[[qra]]]];
               Do[qllo=qpointer[[qlle]];
                  qc1=qAA[[qllo,qcol]];
                  If[qc1=!=0,
```

```
                    qAA[[qllo]]=
                       qAA[[qllo]]-qAA[[qra]]*qc1;
                    If[doparam,qparam[[qllo]]=
                       qparam[[qllo]]-qparam[[qra]]*qc1]]
                ,{qlle,qrow+1,qnrows}]];
    ];

    If[qcol<=qncols,qmul=qrow,qmul=qrow-1;
                              qrang=Drop[qrang,-1]];
    If[qmul>0, qAA=Table[qAA[[qpointer[[qi]]]],{qi,qmul}];
       If[doparam,
          qparam=Table[qparam[[qpointer[[qi]]]]
                       ,{qi,qmul}]];
       qru=qmul;
       While[qru>1,
          Do[qde=qAA[[qllo,qrang[[qru]]]];
             If[qde=!=0,
                qAA[[qllo]]=qAA[[qllo]]-qde*qAA[[qru]];
                If[doparam,qparam[[qllo]]=
                     qparam[[qllo]]-qde*qparam[[qru]]]]
            ,{qllo,qru-1}]; qru--];
          If[doparam,{qmul,qAA,qrang,qparam}
                     ,{qmul,qAA,qrang}]
    ,{0}]
];


Triang[A_,doparam_]:=Module[

{nrows,ncols,AA,param,pointer,col,row,rang,rr,
 ra,c0,llo,c1,lle,mul,ru,de},

      {nrows,ncols}={Length[A],Length[A[[1]]]};
      AA=A;
      If[doparam,param=Table[If[i==j,id,zero]
                             ,{i,nrows},{j,nrows}]];
      pointer=Table[i,{i,nrows}];
```

```
col=0; row=0; rang={};
While[col<ncols && row<nrows,
      row++; col++; rr=row;
      While[col<=ncols &&
            AA[[pointer[[rr]],col]]==zero,
            If[rr<nrows,rr++,col++; rr=row]];
      AppendTo[rang,col];
      ra=pointer[[rr]];
      pointer=Delete[Insert[pointer,ra,row],{rr+1}];
      If[col<=ncols,
         If[AA[[ra,col]]=!=id,
            c0=inv[AA[[ra,col]]];
            AA[[ra]]=starp[c0,AA[[ra]]];
            If[doparam,
               param[[ra]]=starp[c0,param[[ra]]]]];
         Do[llo=pointer[[lle]];c1=AA[[llo,col]];
            If[c1=!=zero,
             AA[[llo]]=AA[[llo]]-starp[c1,AA[[ra]]];
             If[doparam,param[[llo]]=
                param[[llo]]-starp[c1,param[[ra]]]]]
           ,{lle,row+1,nrows}]];
];

If[col<=ncols,mul=row,mul=row-1;rang=Drop[rang,-1]];
If[mul>0, AA=Table[AA[[pointer[[i]]]],{i,mul}];
   If[doparam,
      param=Table[param[[pointer[[i]]]],{i,nrows}]];
   ru=mul;
   While[ru>1,
      Do[de=AA[[llo,rang[[ru]]]];
         If[de=!=zero,
            AA[[llo]]=AA[[llo]]-starp[de,AA[[ru]]];
            If[doparam,param[[llo]]=
                 param[[llo]]-starp[de,param[[ru]]]]]
        ,{llo,ru-1}]; ru--];
      If[doparam,{mul,nrows-mul,AA,rang,param}
        ,{mul,numrows-mul,AA,rang}]
```

```
     ,{0}]
];


kerp[A_]:=Module[{trdata,dimker,param},

trdata=Triang[A,True];
If[trdata=!={0},{dimker,param}=trdata[[{2,5}]];
                  Take[param,-dimker]
  ,1]
];


(*right inverse of a matrix*)

rightinv[T_]:=Module[{trdata,randim,rnm},

trdata=Triang[T,True];
If[trdata=!={0},
    {randim,inm}=trdata[[{1,-1}]], randim=0];
    If[randim<Length[T],
       Print["rightinv called with degenerate matrix"]
      ,inm] ];


(* identitymatrix of dim d, tensor A *)

zerow[dim_]:=Table[zero,{dim}];

Lid[A_,scal_,d_]:=Module[{nc},nc=Length[A[[1]]];
    Flatten[Table[
      Flatten[{zerow[(r-1)nc],starp[scal,A[[s]]]
               ,zerow[(d-r)nc]}
                ,1],{r,d},{s,Length[A]}],1]];


(* A tensor identitymatrix of dim d *)
```

```
Rid[A_,scal_,d_]:=Module[{B,nc,R},

B={}; nc=Length[A[[1]]];
Do[R=Flatten[Table[Join[{starp[scal,A[[r,i]]]}
               ,zerow[d-1]],{i,nc}],1];
   Do[AppendTo[B,R];
      R=Flatten[{{zero},Drop[R,-1]},1],{s,d}]
  ,{r,Length[A]}]; B];


(* check status, get data *)

announce[string_]:= If[reports, Print[string<>" at",
                       Date[][[4]],":",Date[][[5]]]];
announce["Started"];
sumdone={}; PermRep={};
Get[statusfile];

If[If[class0, !class0rep, !allclassesrep] ,Print[
 "According to the status file, the ",inputfile,
 " file is not complete.
 Run the RepSetup program again with p = ",p];
 Abort[]];

If[(class0&&class0sum) || (Not[class0]&&allclassessum),
   Print["According to the statusfile",
         " the data is already complete."]];

Get[rootsfile];
Get[inputfile];
If[dimFdone,Get[predimfile]];

(* determine repno and space no from weights *)

Do[num[AllRep[[n]]]=n,{n,repno}];
repnum[x_]:=0;
```

```
Do[ numwts[n][Wts[n][[k]]]= k,
        {n,repno},{k,Length[Wts[n]]}];
numwts[n_][wt_]:=0;

ran[n_][k_,j_]:=numwts[n][Wts[n][[k]]+
                  If[j>rank,-CC[[j-rank]],CC[[j]]]];


(* Fill in invDmorph, and Action on Duals *)

Do[If[Dual[[n]]<=n,
   invDmorph[n]=Map[rightinv[#]&,Dmorph[n]]]
  ,{n,If[class0,numrep0,repno]}];


dualm[n_,k_]:=Module[{alwt,lwn},

If[Dual[[n]]<=n,Dmorph[n][[k]],
   alwt=Wts[n][[k]].Fweights;
   lwn=Length[Wts[n]];
   starp[vexp[-Sum[alwt[[i]]*Pairing[[i,i]],{i,rank}]]
     ,Transpose[Dmorph[Dual[[n]]][[lwn-k+1]]] ]
]];

invdualm[n_,k_]:=Module[{alwt,lwn},

If[Dual[[n]]<=n,invDmorph[n][[k]],
   alwt=Wts[n][[k]].Fweights;
   lwn=Length[Wts[n]];
   starp[vexp[Sum[alwt[[i]]*Pairing[[i,i]],{i,rank}]]
     ,Transpose[invDmorph[Dual[[n]]][[lwn-k+1]]] ]
]];


Do[dn=Dual[[n]]; lw=Length[Wts[n]];
   If[n>dn, Preaction=Table[Table[{},{2*rank}]
```

```
                              ,{lw}];
   Do[dmr=Dmorph[n][[k]]; curwt=Wts[n][[k]];
      l=lw-k+1;
      Do[l1=numwts[dn][-curwt+CC[[z]]];
        If[l1>0,
         k1=lw-l1+1;
         Preaction[[k,z+rank]]=starp[
          vexp[Wts[dn][[l1,z]]],-dotp[dotp[Dmorph[n][[k]],
          Transpose[Action[dn][[l1,z+rank]]]],
          invDmorph[n][[k1]]]];

         Preaction[[k1,z]]=starp[
          vexp[-Wts[dn][[l1,z]]],-dotp[
          dotp[Dmorph[n][[k1]],
          Transpose[Action[dn][[l,z]]]],
          invDmorph[n][[k]]]]]
        ,{z,rank}],{k,lw-1}];
    Action[n]=Preaction; Clear[Preaction]]
      ,{n,If[class0,numrep0,repno]}];


If[!dimFdone,
   dimFlist=Table[0,{j,repno},{i,j},{k,repno}];
   Do[Do[dimFlist[[j,i,Dual[[summands[j,i][[k]]]]]]]++
        ,{k,Length[summands[j,i]]}];
     ,{i,If[class0,numrep0,repno]},{j,i,repno}];

   dimFlist=Table[{i1,j1,k1}=Sort[{i,j,k}];
                  dimFlist[[j1,i1,k1]]
                ,{i,repno},{j,repno},{k,repno}]];


lpr=Length[Posrlist];


FindRlist[]:=(
```

```
Rlist=Table[
        div=If[Const[[k]]===1,id,inv[Fold[prod,id,
               Map[braket[#[[1]],#[[2]]]&,Const[[k]]]]]];
        {PreRlist[[k,1,1]],
         prod[simplpoly[PreRlist[[k,1,2]]],div]}
      ,{k,lpr},{l,Length[PreRlist[[k]]]}];
);


deltaexp[genlist_,nlist_,tlist_]:=Module[{tlist1,gen,lgl},

tlist1=nlist-tlist; lgl=Length[genlist];
Mod[Sum[gen=genlist[[i]]; tlist[[i]]*(
        Pairing[[gen,gen]]*tlist1[[i]]/2 +
        Sum[tlist1[[j]]*Pairing[[genlist[[j]],gen]]
           ,{j,i+1,lgl}])
      ,{i,lgl}],p]
];


factorial[n_,i_]:=If[n==0,id,If[n==1,id,
             Fold[prod,id,Table[braket[h,i],{h,2,n}]]]
];


seqfactorial[path_,genlist_]:=Module[{termsno},

termsno=Length[genlist];
If[termsno==1,factorial[path[[1]],genlist[[1]]],
   Fold[prod,id,Table[factorial[path[[h]],genlist[[h]]]
                     ,{h,termsno}]]]
];


incr[a_,wtno_,path_,genlist_]:=Module[

{ni,lpth,Ag,nwtno,ms,ss,gen},
```

```
ni=1; lpth=Length[path];
While[ni<=lpth && path[[ni]]==0,ni++];
If[ni<=lpth,
   Ag=Action[a][[wtno,genlist[[ni]]]];
   nwtno=numwts[a][Wts[a][[wtno]]+CC[[genlist[[ni]]]]];
   Do[            (*rr over length of path*)
      ms=path[[rr]]; ss=If[rr==ni,2,1];gen=genlist[[rr]];
      While[ss<=ms,
               Ag=dotp[Ag,Action[a][[nwtno,gen]]];
               ss++;
               nwtno=numwts[a][Wts[a][[nwtno]]+CC[[gen]]]];
     ,{rr,ni,lpth}];
   {nwtno,starp[inv[seqfactorial[path,genlist]],Ag]}
  ,{wtno,Table[If[i==j,id,zero]
          ,{i,Mult[a][[wtno]]},{j,Mult[a][[wtno]]}]}]];


Findgoup[]:=Module[{A}, goup={};

Do[           (*over the representations*)
    If[(class0 && a>numrep0)||Dual[[a]]>a, A={{0}}
      ,A=incr[a,Length[Mult[a]],Reverse[Basis[a][[-1,1]]],
             Reverse[maxelem[a]]][[2]]];
      AppendTo[goup,A[[1,1]]]
   ,{a,repno}]
];


FindRlist[];

(*reordering the representations acording to their
  dimention s.t. Dual[[n]] and n are adjacent*)

newPermRep={};
ActiveRep=Table[i,{i,If[allclassesrep,repno,numrep0]}];
PassiveRep=If[allclassesrep,{},
```

```
            Table[i,{i,numrep0+1,repno}]];
m=1;
While[m<=Length[ActiveRep],
  n=ActiveRep[[m]];
  If[Dual[[n]]<n,m++,
    dim=Sum[Mult[n][[k]],{k,Length[Mult[n]]}];
    If[Dual[[n]]>n,
      AppendTo[newPermRep,{dim,{n,Dual[[n]]}}],
      AppendTo[newPermRep,{dim,{n}}]];
        m++]];
newPermRep=Flatten[Map[Rest,Sort[newPermRep]]];
newPermRep=Join[newPermRep,PassiveRep];

Do[posrep[newPermRep[[i]]]=i, {i,repno}];
If[PermRep=={}, PermRep=newPermRep;
   If[PermRep=!=newPermRep,
      sumdone=Map[posrep[PermRep[[#]]]&,sumdone];
      PermRep=newPermRep;
      Findgoup[];
      DeleteFile[statusfile];
      Save[statusfile,repno,numrep0,repdone,class0rep,
            allclassesrep, PermRep,sumdone,
            class0sum,allclassessum,goup,
            class0cat,
            allclassescat,statustable]]];

Pos0=Table[If[PermRep[[n]]<=numrep0,True,False]
           ,{n,repno}];

first0=1;
While[!Pos0[[first0]] && first0<repno,first0++];
If[first0==repno&&!Pos0[[first0]],
   first0=1;class0sum=True];

Cdet=Det[CC];
droot=1;
While[Mod[Cdet*droot-1,p]=!=0,droot++];
```

```
halflpr=Table[Sum[Posrlist[[k,i]]*Posrlist[[k,j]]*
                  Pairing[[i,j]]/2,{i,rank},{j,rank}]
             ,{k,lpr}];



(* if starting fresh, save stuff to outputfile *)

If[sumdone=={},
   Findgoup[];
   FindRlist[];
   Save[outputfile,category,p,CC,Fweights,
        Pairing,AllRep,repno,numrep0,Dual,Rlist, Posrlist];
   Save[symmfile,category,p,AllRep,repno,numrep0,Dual];
   DeleteFile[statusfile];
   sumdone={0,first0,first0};
   Save[statusfile,repno,numrep0,repdone,class0rep,
        allclassesrep,PermRep,goup,sumdone]];


announce["Finished setup"];


(* pieces of the wt wtspace of ab *)

Pieces[wt_,b_,c_]:=Pieces[wt,b,c]=Module[{stpc},

stpc={};
Do[If[Wts[b][[k]]+Wts[c][[l]]==wt,
      AppendTo[stpc,{k,l}]]
  ,{k,Length[Wts[b]]},{l,Length[Wts[c]]}];

stpc];
```

```
Findpieceno[wt_,b_,c_]:=(

Do[pieceno[wt,b,c][Pieces[wt,b,c][[i]]]=i
  ,{i,Length[Pieces[wt,b,c]]}]
);

pieceno[x_,y_,z_][f_]:=0;


SortPieces[wt_,b_,c_]:=Module[

{m,prepc,found,k,l,kwt,mn,k1wt,ff},

Do[                (*z up to rank*)
  m=1; prepc={};
  While[m<=Length[Pieces[wt,b,c]],
  found=False;
      {k,l}=Pieces[wt,b,c][[m]];
      kwt=Wts[b][[k]]; mn=1;
      While[mn<=Length[prepc] && !found,
            k1wt=Wts[b][[prepc[[mn,1,1]]]];
            If[Delete[(k1wt-kwt).Fweights,{z}]==
               Table[0,{rank-1}],
               AppendTo[prepc[[mn]],{k,l,m}];
               found=True]; mn++];
     If[!found, AppendTo[prepc,{{k,l,m}}]]; m++
];

spieces[z]=Map[Part[#,-1]&,
             Map[Sort[#,(#2[[1]]<#1[[1]])&]&,prepc],{2}];
Do[ff=spieces[z][[l]];
   Do[spieceno[ff[[j]],z]=l,{j,Length[ff]}]
  ,{l,Length[spieces[z]]}];
,{z,rank}];

];
```

```
checkpath[a_,wtno_,path_,genlist_,decr_]:=Module[

{test,ss,lph,nwtno},

test=True;
ss=1; lph=Length[path]; nwtno=wtno;
While[ss<=lph && test,
      nwtno=numwts[a][Wts[a][[nwtno]]-If[decr,1,-1]*
            path[[ss]]*CC[[genlist[[ss]]]]];
      If[nwtno==0,test=False];
      ss++];
test];


checkpath1[a_,wtno_,path_,genlist_,decr_]:=Module[

{lph,nwtno,ss,test},

lph=Length[path];
nwtno=numwts[a][Wts[a][[wtno]]-If[decr,1,-1]*Sum[
            path[[jj]]*genlist[[jj]].CC,{jj,lph}]];

test=If[nwtno==0,False,True];
ss=lph;
While[ss>1 && test,
      nwtno=numwts[a][Wts[a][[nwtno]]+If[decr,1,-1]*
            path[[ss]]*genlist[[ss]].CC];
      If[nwtno==0,test=False];
      ss--];
test];


prgoup[a1_,b1_,c1_][lwvect_]:=Module[

{dda,ddb,ddc,wta1,dimparam,genlist,path0,hwvect,
 lwb,lwc,lpc,lwvpc,k,l,bpath,j,cpath,normcpath},
```

```
{dda,ddb,ddc}=Map[Dual[[#]]&,{a1,b1,c1}];
wta1=Wts[a1][[1]]; wtdda=Wts[dda][[1]];
dimparam=Length[lwvect[[1,1]]];
genlist=Reverse[maxelem[a1]]; lgl=Length[maxelem[a1]];
path0=Reverse[Basis[a1][[-1,1]]];
hwvect=Table[Table[zero,{1},{dimparam}
              ,{Mult[b1][[Pieces[wta1,b1,c1][[i,1]]]]}
              ,{Mult[c1][[Pieces[wta1,b1,c1][[i,2]]]]}]
            ,{i,Length[Pieces[wta1,b1,c1]]}];
lwb=Length[Wts[b1]];
lwc=Length[Wts[c1]];
lpc=Length[lwvect];


Do[        (*qq over Length[lwvect]*)
   lwvpc=lwvect[[qq]];
   {k,l}={lwb-Pieces[wtdda,ddc,ddb][[qq,2]]+1,
          lwc-Pieces[wtdda,ddc,ddb][[qq,1]]+1};
   bpath=Table[If[i<lgl,0,-1],{i,lgl}];
   While[bpath=!=path0,
     j=-1;
     While[bpath[[j]]==path0[[j]],bpath[[j]]=0;j--];
     bpath[[j]]++;
     cpath=path0-bpath;
     If[checkpath[b1,k,bpath,genlist,False] &&
        checkpath[c1,l,cpath,genlist,False],
        {k1,B1}=incr[b1,k,bpath,genlist];
        {l1,C1}=incr[c1,l,cpath,genlist];
        normcpath=Table[0,{rank}];
        Do[normcpath[[genlist[[i]]]]+=cpath[[i]]
          ,{i,lgl}];
        hwvect[[pieceno[wta1,b1,c1][{k1,l1}]]]+=
         starp[vexp[Sum[Wts[b1][[k,i]]*Pairing[[i,i]]/2*
                        normcpath[[i]], {i,rank}]
                   +deltaexp[genlist,path0,bpath]],
         dotp[Transpose[dotp[Transpose[lwvpc,{1,2,4,3}]
                             ,B1],{1,2,4,3}],C1]]
```

```
]
]
   ,{qq,lpc}];
Map[starp[inv[goup[[a1]]],#]&,hwvect]
];

     (* starting the functions for findgoodvct *)


ztable[d1_,d2_]:=ztable[d1,d2]=Table[zero,{d1},{d2}];
idmatrix[d_]:=Table[If[i=!=j,zero,id],{i,d},{j,d}];


kerprm1[pc1_,pc2_,d1_,d2_]:=(

If[pc1==={},If[pc2===1,
   prm1=If[d1>0,1,{}]; prm2={}

  , (* pc1==={} && pc2=!=1 *)

   preprm2=If[pc2==={},idmatrix[d2]
            ,ppr=kerp[pc2];If[ppr===1,idmatrix[d2],ppr]];
   dd2=Length[preprm2];
   prm1=If[d1>0,If[dd2>0,Join[idmatrix[d1]
                              ,ztable[dd2,d1]]
                  ,idmatrix[d1]]
         ,{}];
   prm2=If[d1>0,Join[ztable[d1,d2],preprm2],preprm2]]

  ,  (* pc1=!={}  *)

   If[pc2==={},If[pc1===1,
      prm1={};prm2=If[d2>0,1,{}]

        , (* pc2 ==={} && pc1=!=1 && pc1=!={} *)

      preprm1=kerp[pc1];
```

```
If[preprm1==1,preprm1=idmatrix[d1]];
dd1=Length[preprm1];
prm1=Join[preprm1,ztable[d2,d1]];
prm2=If[dd1>0,Join[ztable[dd1,d2],idmatrix[d2]]
       ,idmatrix[d2]]]

,  (* pc2=!={} && pc1=!={} *)

If[pc2===1,If[pc1===1,
   prm1=1;
   prm2=-idmatrix[d2]

    0, (* pc2 ===1 && pc1=!=1 && pc1 =!={} *)

   joinpr=Join[pc1,idmatrix[d2]];
   pcsol=kerp[joinpr];
   If[pcsol==1,pcsol=idmatrix[d1+d2]];
   prm1=If[pcsol=!={},
        Transpose[Take[Transpose[pcsol],d1]],{}];
   prm2=If[pcsol=!={},
        Transpose[Take[Transpose[pcsol],-d2]],{}]]
, (* pc2=!=1 *)

   If[pc1===1,
      joinpr=Join[idmatrix[d1],pc2];
       pcsol=kerp[joinpr];
       If[pcsol==1,pcsol=idmatrix[d1+d2]];
       prm1=If[pcsol=!={},
              Transpose[Take[Transpose[pcsol],d1]],{}];
       prm2=If[pcsol=!={},
             Transpose[Take[Transpose[pcsol],-d2]],{}]
     ,

      joinpr=Join[pc1,pc2];
      pcsol=kerp[joinpr];
      If[pcsol==1,pcsol=idmatrix[d1+d2]];
      prm1=If[pcsol=!={},
```

```
                Transpose[Take[Transpose[pcsol],d1]],{}];
           prm2=If[pcsol=!={},
                Transpose[Take[Transpose[pcsol],-d2]],{}]
]]]];

{prm1,prm2});


kerprm2[pc1_,pc2_,d1_,d2_]:=(

If[pc1==={} && pc2==={},prm=1,
   If[(pc1==={}&&pc2===1) || (pc2==={}&&pc1===1) ||
      (pc1===1 && pc2===1),prm={},
   If[pc1==={},prm=kerp[pc2], If[pc2==={},
      prm=kerp[pc2]
     ,If[pc1===1 || pc2===1, syst=If[pc1===1,pc2,pc1];
         Do[syst[[i,i]]+=id,{i,d1}]
        ,syst=pc1+pc2];
      prm=kerp[syst]]]]];
prm);


check[z_]:=Table[True,{lsp[z]}];


corrprm1[prm1_,prm2_,comp1_,comp2_,s_,z_]:=(

If[comp1==0,
 If[comp2==0,
    change1=change2=True; newcomp=h0+1;h0++
   ,change2=False; change1=True; newcomp=comp2]
 ,
If[comp2==0 || comp2>comp1,
     change2=True; change1=False; newcomp=comp1
    ,change2=False; change1=True; newcomp=comp2]
];
```

```
{t1,t2}={check[1],check[z]};
Do[       (*s1 up to s*)
   {ff1,gg1}={spieceno[s1,1],spieceno[s1,z]};
   {fcomp,gcomp}={followup[[1,ff1]],followup[[z,gg1]]};

   If[(fcomp==comp1||fcomp==comp2) && t1[[ff1]],
     If[fcomp==comp1, newprm=prm1;
       If[change1,followup[[1,ff1]]=newcomp]
      ,newprm=prm2;
       If[change2,followup[[1,ff1]]=newcomp]];
     If[newprm=!=1,
        Do[cpc=spieces[1][[ff1,k]];
           listprm[[1,cpc]]=dotp[newprm,listprm[[1,cpc]]]
          ,{k,Length[spieces[1][[ff1]]]}];
        listdim[[1,ff1]]=Length[newprm]];
      t1[[ff1]]=False];
  If[s1<=s,
   If[(gcomp==comp2 || gcomp==comp1) && t2[[gg1]],
     If[gcomp==comp1 && gcomp=!=0, newprm=prm1;
       If[change1,followup[[z,gg1]]=newcomp]
      ,newprm=prm2;
       If[change2,followup[[z,gg1]]=newcomp]];
     If[newprm=!=1,
        Do[cpc=spieces[z][[gg1,k]]; If[cpc>s,
           listprm[[z,cpc]]=
                   dotp[newprm,listprm[[z,cpc]]]]
          ,{k,Length[spieces[z][[gg1]]]}];
        listdim[[z,gg1]]=Length[newprm]];
      t2[[gg1]]=False]]
,{s1,If[z==2,s,lpc]}]
);


corrprm2[newprm_,comp_,s_,z_]:=(

If[newprm=!=1,
  {t1,t2}={check[1],check[z]};
```

```
  Do[       (*s1 up to s*)
   {ff1,gg1}={spieceno[s1,1],spieceno[s1,z]};
   {fcomp,gcomp}={followup[[1,ff1]],followup[[z,gg1]]};
   If[fcomp==comp && t1[[ff1]],
    Do[cpc=spieces[1][[ff1,k]];
      listprm[[1,cpc]]=dotp[newprm,listprm[[1,cpc]]]
    ,{k,Length[spieces[1][[ff1]]]}];
    listdim[[1,ff1]]=Length[newprm];
    t1[[ff1]]=False];
  If[s1<=s,
   If[gcomp==comp && t2[[gg1]],
    Do[cpc=spieces[z][[gg1,k]]; If[cpc>s,
      listprm[[z,cpc]]=dotp[newprm,listprm[[z,cpc]]]]
      ,{k,Length[spieces[z][[gg1]]]}];
    listdim[[z,gg1]]=Length[newprm];
    t2[[gg1]]=False]]
,{s1,If[z==2,s,lpc]}]]
);


intersc[]:=(

followup=Table[Table[0,{lsp[z]}],{z,rank}];
h0=0; testker=True;

z=2;
While[z<=rank,
 s=1;       (* z from 2 up to rank *)
  While[s<=lpc,        (*s over lpc*)

   {ff,gg}={spieceno[s,1],spieceno[s,z]};
   {d1,d2}={listdim[[1,ff]],listdim[[z,gg]]};
   {pc1,pc2}={listprm[[1,s]],listprm[[z,s]]};
   {comp1,comp2}={followup[[1,ff]],followup[[z,gg]]};

   If[(comp1=!=comp2) || (comp1==comp2==0) ,
      {prm1,prm2}=kerprm1[pc1,pc2,d1,d2];
```

```
      If[prm1=!={} || prm2=!={},
            corrprm1[prm1,prm2,comp1,comp2,s,z]
        ,s=lpc;z=rank; testker=False];
     ,    (* if comp1==comp2 *)
      prm=kerprm2[pc1,pc2,d1,d2];
      If[prm=!={},corrprm2[prm,comp1,s,z]
        ,s=lpc;z=rank; testker=False]
];
  s++];
z++];
If[testker,

  complist=Union[followup[[1]]];
  lcp=Length[complist];
  Do[compno[complist[[n]]]=n,{n,lcp}];
  finaldims=Table[0,{lcp}];
  Do[finaldims[[compno[followup[[1,s]]]]]=
      listdim[[1,s]]
    ,{s,lsp[1]}];
  d1=0;
  Do[d1+=finaldims[[i]],{i,lcp}];
  If[lcp==1,intersvect=listprm[[1]],
   intersvect={};
   Do[          (* s over lpc *)
      n=compno[followup[[1,spieceno[s,1]]]];
      prevdim=Map[Plus,Take[finaldims,n-1]];
      afterdim=Map[Plus,Drop[finaldims,n]];
      colno=dpc[[1]]*dpc[[2]];
      AppendTo[intersvect,
       Join[ztable[prevdim,colno],listprm[[1,s]],
            ztable[afterdim,colno]]]
    ,{s,lpc}]]
,d1=0];
Clear[listprm]; Clear[listdim];
);
```

```
Findgoodvct[a_,b_,c_]:=(

Clear[goodvct];

{duala,dualb,dualc}=Map[Dual[[#]]&,{a,b,c}];
dualtest=(Sort[{a,dualb,dualc}]==
          Sort[{duala,b,c}]);
hwt=AllRep[[a]];
lwb=Length[Mult[b]]; lwc=Length[Wts[c]];
lwa=Length[Wts[a]];
Findpieceno[hwt,b,c];
Findpieceno[AllRep[[duala]],dualc,dualb];
lpc=Length[Pieces[hwt,b,c]];
If[lpc<=0,dimFsum[duala,b,c]=0,

 If[hwt==AllRep[[b]]+AllRep[[c]],
   goodvct[a,b,c]={{{{{id}}}}}; goodcovect={{{{{id}}}}};
   dimFsum[duala,b,c]=mult=1
  ,
   SortPieces[hwt,b,c];
   dpc=Table[{Mult[b][[Pieces[hwt,b,c][[k,1]]]]
               ,Mult[c][[Pieces[hwt,b,c][[k,2]]]]}
             ,{k,lpc}];
   listprm={};
   listdim={};

   Do[            (* z over rank *)
      Print["Started z=",z];
      lsp[z]=Length[spieces[z]];
      epr=Table[1,{lpc}];
      dimepr=Table[dpc[[spieces[z][[s,1]],1]]*
                   dpc[[spieces[z][[s,1]],2]]
                         ,{s,lsp[z]}];

      Do[           (* s over the length of spieces *)
         (*Print["started s=",s];*)
```

```
curseq=spieces[z][[s]];
curpc0=curseq[[1]];
{k0,l0}=Pieces[hwt,b,c][[curpc0]]; jj=1;
{mlb0,mlc0}={Mult[b][[k0]],Mult[c][[l0]]};
l01=numwts[c][Wts[c][[l0]]+CC[[z]]];
{epr0}={1};
If[l01=!=0,
  epr[[curpc0]]=epr0=
     kerp[Lid[Action[c][[l0,z]]
      ,vexp[Wts[b][[k0,z]]*Pairing[[z,z]]/2]
      ,mlb0]];
  If[epr0=!=1,dimepr[[s]]=Length[epr0]];
  ];
While[jj<=Length[curseq]-1,
      curpc1=curseq[[jj+1]];
      {k1,l1}=Pieces[hwt,b,c][[curpc1]];
      {mlb1,mlc1}={Mult[b][[k1]],Mult[c][[l1]]};
      {epr1}={1};
      EE=Join[dotp[epr0,
           Rid[Action[b][[k0,z]],id,mlc0]]
          ,Lid[Action[c][[l1,z]],
            vexp[Wts[b][[k1,z]]*Pairing[[z,z]]/2]
            ,mlb1]];
      pr=kerp[EE];
      If[pr=!=1,curdim=Length[pr];
         pr=Transpose[pr];
         pr0=Transpose[Take[pr,dimepr[[s]]]];
         pr1=Transpose[Take[pr,-mlb1*mlc1]];
         Do[epr[[curseq[[ii]]]]=
                dotp[pr0,epr[[curseq[[ii]]]]]
           ,{ii,jj}];
         epr[[curpc1]]=epr0=pr1;
         dimepr[[s]]=curdim
        ,    (* if pr==1*)
         curdim=Length[EE];
         Do[epr[[curseq[[ii]]]]=
            Join[epr[[curseq[[ii]]]],
```

```
                    Table[zero,{curdim-dimepr[[s]]}
                      ,{Length[epr[[curseq[[ii]],1]]]}]]
                  ,{ii,jj}];
               epr[[curpc1]]=epr0=
                  Table[If[i<=dimepr[[s]],zero,
                    If[i==1,id,zero]]
                   ,{i,curdim},{l,mlb1*mlc1}];
               dimepr[[s]]=curdim];

            jj++;
            {k0,l0,mlb0,mlc0,curpc0}=
                    {k1,l1,mlb1,mlc1,curpc1}
];

      k01=numwts[b][Wts[b][[k0]]+CC[[z]]];
      If[k01=!=0,
         pr0=kerp[dotp[epr0,Rid[Action[b][[k0,z]]
                          ,id,mlc0]]];
         If[pr0=!=1,
            Do[epr[[curseq[[ii]]]]=
                      dotp[pr0,epr[[curseq[[ii]]]]];
               dimepr[[s]]=Length[pr0]
              ,{ii,jj}]];
         ]
     ,{s,lsp[z]}];

  AppendTo[listprm,epr]; Clear[epr];
  AppendTo[listdim,dimepr];

  ,{z,rank}];

Print["Finished epr"];
intersc[];
listepr=Table[qw=intersvect[[k]];
            If[qw===={},ztable[d1,dpc[[k,1]]*dpc[[k,2]]],
               If[qw===1,idmatrix[d1],qw]],{k,lpc}];
edim=d1;
```

```
If[edim>0,


listprm={}; listdim={};
   Do[            (* z over rank *)
     Print["Started z=",z];
     fpr=Table[1,{lpc}];
     dimfpr=Table[dpc[[spieces[z][[s,1]],1]]*
                  dpc[[spieces[z][[s,1]],2]]
                        ,{s,lsp[z]}];
      Do[           (* s over the length of spieces *)
         (*Print["Started s=",s];*)
         curseq=spieces[z][[s]];
         curpc0=curseq[[1]];
         {k0,l0}=Pieces[hwt,b,c][[curpc0]]; jj=1;
         {mlb0,mlc0}={Mult[b][[k0]],Mult[c][[l0]]};
         l01=numwts[c][Wts[c][[l0]]+CC[[z]]];
         fpr0=1;
         If[l01=!=0,
            fpr[[curpc0]]=fpr0=
              kerp[Transpose[Lid[Action[c][[l01,z+rank]]
                              ,id,mlb0]]];
           If[fpr0=!=1,dimfpr[[s]]=Length[fpr0]];
];
         While[jj<=Length[curseq]-1,
               curpc1=curseq[[jj+1]];
               {k1,l1}=Pieces[hwt,b,c][[curpc1]];
               {mlb1,mlc1}={Mult[b][[k1]],Mult[c][[l1]]};
               fpr1=1;
               F=Join[dotp[fpr0,
                  Transpose[Rid[Action[b][[k1,z+rank]],
                    vexp[-Wts[c][[l0,z]]*Pairing[[z,z]]/2]
                    ,mlc0]]]
                 ,Transpose[Lid[Action[c][[l0,z+rank]],
                      id,mlb1]]];
               pr=kerp[F];
               If[pr=!=1,
```

```
                curdim=Length[pr];
                pr=Transpose[pr];
                pr0=Transpose[Take[pr,dimfpr[[s]]]];
                pr1=Transpose[Take[pr,-mlb1*mlc1]];
                Do[fpr[[curseq[[ii]]]]=
                       dotp[pr0,fpr[[curseq[[ii]]]]];
                 ,{ii,jj}];
                fpr[[curpc1]]=fpr0=pr1;
                dimfpr[[s]]=curdim
                ,    (* if pr==1*)
                curdim=Length[F];
                Do[fpr[[curseq[[ii]]]]=
                   Join[fpr[[curseq[[ii]]]],
                    Table[zero,{curdim-dimfpr[[s]]}
                      ,{Length[fpr[[curseq[[ii]],1]]]}]]
                  ,{ii,jj}];
                fpr[[curpc1]]=fpr0=
                   Table[If[i<=dimfpr[[s]],zero,
                     If[i==l,id,zero]]
                    ,{i,curdim},{l,mlb1*mlc1}];
                dimfpr[[s]]=curdim];
             jj++;
             {k0,l0,mlb0,mlc0,curpc0}=
                     {k1,l1,mlb1,mlc1,curpc1}
];

          k01=numwts[b][Wts[b][[k0]]+CC[[z]]];
          If[k01=!=0,
             pr0=kerp[dotp[fpr0,
                  Transpose[Rid[Action[b][[k01,z+rank]],
                      vexp[-Wts[c][[l0,z]]*Pairing[[z,z]]/2]
                    ,mlc0]]]];
             If[pr0=!=1,
                Do[fpr[[curseq[[ii]]]]=
                          dotp[pr0,fpr[[curseq[[ii]]]]];
                 dimfpr[[s]]=Length[pr0]
                ,{ii,jj}]]]
```

```
       ,{s,lsp[z]}];

 AppendTo[listprm,fpr]; Clear[fpr];
 AppendTo[listdim,dimfpr]; Clear[dimfpr];

 ,{z,rank}];

announce["Finished fpr"];
intersc[];
announce["Finished intersection"];
listfpr=Table[qw=intersvect[[k]];
              If[qw===={},ztable[d1,dpc[[k,1]]*dpc[[k,2]]],
                 If[qw===1,idmatrix[d1],qw]],{k,lpc}];
fdim=d1;

listfpr=Map[Transpose[#]&,listfpr];
norm=Sum[dotp[listepr[[k]],listfpr[[k]]],{k,lpc}];
Print["norm=",norm];
triang=Triang[norm,True];
If[triang===={0},mult=0,
  {mult,rang,param}=triang[[{1,-2,-1}]]];
Print["mult=",mult];

If[dimFdone && (dimFlist[[duala,b,c]]=!=mult)
    || dimFlist[[duala,b,c]]<mult,
    Print["Problem with goodvct[",a,",",b,",",c,"]"];
          Abort[], dimFsum[duala,b,c]=mult];

If[mult=!=0,

(*partitioning the hwgoodvect and hwgoodcovect
     into 4-tensors*)
    hwcovect=Map[Transpose[Take[Transpose[#],rang]]&
                ,listfpr];
    hwvect=Map[dotp[Take[param,mult],#]&,listepr];
    goodvct[a,b,c]=Table[{Map[Partition[#,dpc[[k,2]]]&,
                        hwvect[[k]]]},{k,lpc}];
```

```
    If[!dualtest,
       goodcovect=Table[Partition[Map[Partition[#,1]&,
                    hwcovect[[k]]],dpc[[k,2]]],{k,lpc}]]
,dimFsum[duala,b,c]=mult=0]
];

If[mult=!=0 && !dualtest,
    lwvect=Map[Transpose[#,{3,4,2,1,5}]&,goodcovect];
    lwvect=Table[{k,l}=Pieces[hwt,b,c][[f]];
      dotp[dualm[duala,lwa],
         dotp[Transpose[dotp[lwvect[[f]],
                 invdualm[dualc,lwc-l+1]],{1,2,4,3,5}],
              invdualm[dualb,lwb-k+1]]]
      ,{f,lpc}];
    goodvct[duala,dualc,dualb]=
        prgoup[duala,dualc,dualb][lwvect];
]]
 );


decr[a_,wtno_,path_,genlist_]:=Module[

{ni,lpth,Ad,nwtno,ms,ss,gen},

ni=1;lpth=Length[path];
While[ni<=lpth && path[[ni]]==0,ni++];
If[ni<=lpth,
   Ad=Action[a][[wtno,rank+genlist[[ni]]]];
   nwtno=numwts[a][Wts[a][[wtno]]-CC[[genlist[[ni]]]]];
   Do[            (*rr over length of path*)
      ms=path[[rr]]; ss=If[rr==ni,2,1];gen=genlist[[rr]];
      While[ss<=ms,
            Ad=dotp[Ad,Action[a][[nwtno,gen+rank]]];
            ss++;
            nwtno=numwts[a][Wts[a][[nwtno]]-CC[[gen]]]];
     ,{rr,ni,lpth}];
```

```
    {nwtno,starp[inv[seqfactorial[path,genlist]],Ad]}
   ,{wtno,Table[If[i==j,id,zero]
           ,{i,Mult[a][[wtno]]},{j,Mult[a][[wtno]]}]}]];



prgodown[b1_,c1_][genlist_,path_,wt_,vect_]:=Module[

{pieces,dimparam,dimvect,lgl,newwt,newpc,lnpc,newvect,
 lwb,lwc,lpc,vpc,k,l,bpath,cpath,bnorm},

If[path=!={},

pieces=Pieces[wt,b1,c1];
dimparam=Length[vect[[1,1]]];
dimvect=Length[vect[[1]]];
lgl=Length[path];
newwt=wt-Sum[path[[i]]*CC[[genlist[[i]]]],{i,lgl}];
Findpieceno[newwt,b1,c1];
newpc=Pieces[newwt,b1,c1];
lnpc=Length[newpc];
newvect=Table[Table[zero,{dimvect},{dimparam}
              ,{Mult[b1][[newpc[[i,1]]]]}
              ,{Mult[c1][[newpc[[i,2]]]]}]
             ,{i,Length[newpc]}];
lwb=Length[Wts[b1]];
lwc=Length[Wts[c1]];
lpc=Length[vect];

Do[        (*qq over Length[vect]*)
   vpc=vect[[qq]];
   {k,l}=pieces[[qq]];
   bpath=Table[If[i<lgl,0,-1],{i,lgl}];
   While[bpath=!=path,
     j=-1;
     While[bpath[[j]]==path[[j]],bpath[[j]]=0;j--];
     bpath[[j]]++;
```

```
      cpath=path-bpath;
      If[checkpath[b1,k,bpath,genlist,True] &&
         checkpath[c1,l,cpath,genlist,True],
         {k1,B1}=decr[b1,k,bpath,genlist];
         {l1,C1}=decr[c1,l,cpath,genlist];
         bnorm=Table[0,{rank}];
         Do[bnorm[[genlist[[i]]]]+=bpath[[i]]
           ,{i,lgl}];
         newvect[[pieceno[newwt,b1,c1][{k1,l1}]]]+=
          starp[vexp[-Sum[Wts[c1][[l1,i]]*Pairing[[i,i]]/2*
                         bnorm[[i]], {i,rank}]
                    -deltaexp[genlist,path,bpath]],
          dotp[Transpose[dotp[Transpose[vpc,{1,2,4,3}]
                              ,B1],{1,2,4,3}],C1]]
]
]
   ,{qq,lpc}];
newvect, vect]
];



pcInj[a_,b_,c_][k_]:=Module[

{hwt,kwt,lkpc,inj,path,genlist,vct},

hwt=AllRep[[a]];

If[k==1,goodvct[a,b,c],
   kwt=Wts[a][[k]]; lkpc=Length[Pieces[kwt,b,c]];
   inj=Table[{},{lkpc}];

   Do[path=Basis[a][[k,s]];
      genlist=Take[maxelem[a],Length[path]];
      vct=Map[Flatten[#,1]&,
           prgodown[b,c][genlist,path,hwt,goodvct[a,b,c]]
          ];
```

```
      Do[inj[[f]]=AppendTo[inj[[f]],vct[[f]]]
        ,{f,lkpc}]
      ,{s,Length[Basis[a][[k]]]}];
   inj] ];



FindQdim[]:=(

Qdim=Table[alphawts=Map[#.Fweights&,Wts[a]];
           Sum[Mult[a][[k]]*
               vexp[-Sum[alphawts[[k,l]]*Pairing[[l,l]]
                         ,{l,rank}]]
              ,{k,Length[Wts[a]]}]
         ,{a,If[class0,numrep0,repno]}];
);



Findtwist[]:=(

twist=Table[wt1=AllRep[[n]].Fweights;
         vexp[Cdet*droot*
              (-Sum[wt1[[ii]]*Pairing[[ii,ii]],{ii,rank}]
               -Sum[wt1[[i]]*wt1[[j]]*Pairing[[i,j]]
                    ,{i,rank},{j,rank}])]
           ,{n,If[allclassesrep,repno,numrep0]}]);


power[elem_,n_]:=Module[{n1,ff,result},
   If[n==0,id,n1=If[n>0,n,-n];
      ff=2;result=elem;
      While[ff<=n1,result=prod[result,elem];ff++];
      If[n>0,result,inv[result]]]];
```

```
Rnorm[explist_,bar_]:=Module[

{niu,barniu,sumexp,ss,vterm},

niu=Sum[explist[[i]]*Posrlist[[i]],{i,lpr}];
barniu=Sum[niu[[i]],{i,rank}];
sumexp=Sum[explist[[i]],{i,lpr}];
ss=1; vterm=id;
While[ss<=lpr,If[explist[[ss]]=!=0,
      vterm=prod[
        power[vexp[halflpr[[ss]]]-vexp[-halflpr[[ss]]]
              ,2*explist[[ss]]],vterm]];
      ss++];
(-1)^(Mod[barniu+If[bar,sumexp,0],2])*prod[
   vexp[If[bar,-1,1]*(
        Sum[niu[[i]]*Pairing[[i,i]]/2,{i,rank}]-
        Sum[explist[[i]]*(explist[[i]]+1)*halflpr[[i]]/2
            ,{i,lpr}])],
   prod[vterm,inv[Fold[prod,id,Flatten[
       Table[(vexp[l*halflpr[[k]]]-vexp[-l*halflpr[[k]]])
            ,{k,lpr},{l,explist[[k]]}],1]]]]
];


Bar[elem_]:=Module[{newelem},

newelem=zero;
Do[newelem+=elem[[i]]*vexp[-i+1],{i,p-1}];
newelem];


applyR[inverse_,b_,c_][wt_,vect_]:=Module[

{pieces,lpc,vect1,path0,corr,newvect,k,l,curpath,curj,
 curtest,bwtslist,boperlist,newwtslist,newoperlist,
 oper,coef,curwts,k1,u,r,curoper,curcoef,cwtslist,
 coperlist,l1,npcno,pc,newpc,act1,act2,r1},
```

```
announce["Started applying R."];
pieces=Pieces[wt,b,c];
Findpieceno[wt,b,c];
lpc=Length[pieces];

vect1=If[inverse,vect,
   Table[{k,l}=pieces[[s]];
     {alpb,alpc}=
            {Wts[b][[k]].Fweights,Wts[c][[l]].Fweights};
     starp[vexp[-Cdet*droot*Sum[alpb[[i]]*alpc[[j]]*
                   Pairing[[i,j]],{i,rank},{j,rank}]]
          ,vect[[s]]]
     ,{s,lpc}]];

path0=Table[p-1,{lpr}];
corr=Table[1,{lpr}];
newvect=vect1;

Do[          (* s over lpc*)
   {k,l}=pieces[[s]];
   curpath=Table[0,{i,lpr}];
   curj=0; curtest=True;
   While[curj>-lpr || (curtest && curpath[[-lpr]]<p-1),
    j=-1;
    While[If[curtest,curpath[[j]]==path0[[j]],j>=curj]
         ,curpath[[j]]=0;j--];
     curpath[[j]]++;
     curtest=False;
     If[checkpath1[b,k,curpath,Posrlist,True]
        && checkpath1[c,l,curpath,Posrlist,False],
        bwtslist={{k}}; boperlist={{{},id}};
        Do[r=1;              (*t over lpr*)
           While[r<=curpath[[t]] && bwtslist=!={},
              newwtslist={}; newoperlist={};
              Do[        (* n over Rlist[[t]]*)
                {oper,coef}=Rlist[[t,n]];
```

```
        Do[     (*m over the length of bwtslist*)
           curwts=bwtslist[[m]];
           {curoper,curcoef}=boperlist[[m]];
           k1=curwts[[-1]];u=1;
           While[k1=!=0 && u<=Length[oper],
                 k1=ran[b][k1,oper[[u]]+rank];
                 AppendTo[curwts,k1];
                 AppendTo[curoper,oper[[u]]+rank];
                 u++];
           If[k1=!=0,AppendTo[newwtslist,curwts];
             AppendTo[newoperlist
               ,{curoper,prod[curcoef,coef]}]];
          ,{m,Length[bwtslist]}];
         ,{n,Length[Rlist[[t]]]}];
        bwtslist=newwtslist;
        boperlist=newoperlist;r++]
    ,{t,lpr}];

If[bwtslist=!={},
  cwtslist={{1}}; coperlist={{{},id}};
  Do[r=1;        (*t over lpr*)
     While[r<=curpath[[t]] && cwtslist=!={},
        newwtslist={}; newoperlist={};
       Do[        (* n over Rlist[[t]]*)
          {oper,coef}=Rlist[[t,n]];
        Do[     (*m over the length of cwtslist*)
           curwts=cwtslist[[m]];
           {curoper,curcoef}=coperlist[[m]];
           k1=curwts[[-1]];u=1;
           While[k1=!=0 && u<=Length[oper],
                 k1=ran[c][k1,oper[[u]]];
                 AppendTo[curwts,k1];
                 AppendTo[curoper,oper[[u]]];
                 u++];
           If[k1=!=0,AppendTo[newwtslist,curwts];
             AppendTo[newoperlist
               ,{curoper,prod[curcoef,coef]}]];
```

```
            ,{m,Length[cwtslist]}];
           ,{n,Length[Rlist[[t]]]}];
         cwtslist=newwtslist;
         coperlist=newoperlist;r++]
      ,{t,lpr}];

   If[bwtslist=!={} && cwtslist=!={},
      curtest=True;
      {k1,l1}={bwtslist[[-1,-1]],cwtslist[[-1,-1]]};
      npcno=pieceno[wt,b,c][{k1,l1}];
      pc=Transpose[vect1[[s]],{1,2,4,3,5}];
      newpc=Sum[ (*t up to Length[boperlist]*)
                 (*s1 up to Length[coperlist]*)
       act1=Action[b][[bwtslist[[t,1]],
                      boperlist[[t,1,1]] ]];
       r=2;
       While[r<=Length[boperlist[[t,1]]],
          act1=dotp[act1,Action[b][[bwtslist[[t,r]],
                      boperlist[[t,1,r]] ]]];
          r++];
       curpc=Transpose[ dotp[pc,act1],{1,2,4,3,5}];
       act2=Action[c][[cwtslist[[s1,1]],
                      coperlist[[s1,1,1]] ]];
       r1=2;
       While[r1<=Length[coperlist[[s1,1]]],
          act2=dotp[act2,Action[c][[cwtslist[[s1,r1]],
                         coperlist[[s1,1,r1]] ]]];
          r1++];
       starp[If[!inverse,
        prod[boperlist[[t,2]],coperlist[[s1,2]]]
       ,Bar[prod[boperlist[[t,2]],coperlist[[s1,2]]]]]
           ,dotp[curpc,act2]]
       ,{t,Length[boperlist]}
       ,{s1,Length[coperlist]}];

      newvect[[npcno]]+=
              starp[Rnorm[curpath,inverse],newpc]
```

```
          ]]];
curj=j],{s,lpc}];

If[!inverse,newvect,
   Table[{k,l}=pieces[[s]];
     {alpb,alpc}=
            {Wts[b][[k]].Fweights,Wts[c][[l]].Fweights};
      starp[vexp[Cdet*droot*Sum[alpb[[i]]*alpc[[j]]*
                   Pairing[[i,j]],{i,rank},{j,rank}]]
                  ,newvect[[s]]]
     ,{s,lpc}]]
 ];


Comm[inverse_,b_,c_][wt_,vect_]:=

If[!inverse,applyR[False,c,b][wt,
          Reverse[Map[Transpose[#,{1,2,4,3,5}]&,vect]]]
  ,
   Reverse[Map[Transpose[#,{1,2,4,3,5}]&,
               applyR[True,b,c][wt,vect]]]];



testform1[a_,b_,c_]:=(a==dualrep[a] && b==dualrep[b]
                      && c==dualrep[c]);

testform2[a_,b_,c_]:=(a==dualrep[b] && c==dualrep[c]);

testform3[a_,b_,c_]:=(a==dualrep[a] && b==dualrep[c]);


Pairing1[inverse1_,a_,b_,c_]:=(

lowinj=pcInj[dla,dlb,dlc][lwtsa];
commvect=Comm[inverse1,dlb,dlc][-awt,lowinj];
```

```
  Sum[{k,l}=Pieces[awt,b,c][[n]];
   pcproj=starp[invdualm[dla,lwtsa][[1,1]],
          dotp[Transpose[
          dotp[commvect[[n]],dualm[dlb,lwtsb-k+1]]
            ,{1,2,4,3,5}],dualm[dlc,lwtsc-l+1]]];
   dotp[Map[Flatten[#,1]&,Flatten[goodvct[a,b,c][[n]],1]],
        Transpose[Map[Flatten[#,1]&,Flatten[pcproj,1]]]]

,{n,lpc}]
)

cyclperm[a_,k_]:=Module[{lambda},
             Sum[lambda=Wts[a][[k]].Fweights;
                  lambda[[i]]*Pairing[[i,i]] ,{i,rank}]];



FindSymm[a_,b_,c_]:=(

{dla,dlb,dlc}=Map[Dual[[#]]&,{a,b,c}];

awt=AllRep[[a]];
lpc=Length[Pieces[awt,b,c]];
lwtsb=Length[Wts[b]];
lwtsc=Length[Wts[c]];
lwtsa=Length[Wts[a]];
testform=Sort[{a,dlb,dlc}]===Sort[{dla,b,c}];
If[testform,

  If[dla=!=b && b=!=c && b==dlb,
     Form[dla,b,c]=starp[Qdim[[a]],Pairing1[False,a,b,c]]
    ,
     If[b==c || b==dlc,
       lowinj=pcInj[a,b,c][lwtsa];
       Form[dla,b,c]=starp[Qdim[[a]],Sum[
         {k,l}=Pieces[awt,b,c][[n]];
```

```
        pcproj=dotp[invdualm[a,lwtsa],dotp[Transpose[
           dotp[lowinj[[n]],dualm[b,lwtsb-k+1]]
           ,{1,2,4,3,5}],dualm[c,lwtsc-l+1]]];
        dotp[Map[Flatten[#,1]&
                ,Flatten[goodvct[a,b,c][[n]],1]],
             Transpose[Map[Flatten[#,1]&
                           ,Flatten[pcproj,1]]]]
      ,{n,lpc}]]

  ,If[a==b,
        upvect={};
        Do[{k,l}=Pieces[awt,b,c][[n]];
           AppendTo[upvect,pcInj[b,a,c][k][[1]]]
         ,{n,lpc}];
        Form[dla,b,c]=starp[Qdim[[a]],Sum[
          {k,l}=Pieces[awt,b,c][[n]];
          dotp[Map[Flatten[#,1]&
                   ,Flatten[goodvct[a,b,c][[n]],1]],
            Map[Flatten[#,1]&,Flatten[
            Transpose[dotp[upvect[[n]],
              dualm[c,lwtsc-l+1]],{1,4,3,2,5}],1]]]
         ,{n,lpc}]]
]]]];

  If[b==c,
     preT2=Pairing1[False,a,b,c];
     T2[dla,b,c]=If[!testform,preT2,
         dotp[starp[Qdim[[a]],preT2]
             ,rightinv[Form[dla,b,c]]]]];

  If[dla==b, If[b==c || !testform,
     cpc=Pieces[Wts[a][[1]],c,dla];
        proj={};
        Do[{k,l}=cpc[[n]];
           AppendTo[proj,starp[
            vexp[-cyclperm[a,1]+cyclperm[dla,l]]
           ,Transpose[dotp[
```

```
                Transpose[pcInj[dla,dlc,a][l][[-1]]
                          ,{1,2,4,3}]
                ,dualm[dlc,lwtsc-k+1]]
             ,{2,3,4,1}]]]
          ,{n,Length[cpc]}];

         inj=Comm[True,dla,c][Wts[a][[1]],goodvct[a,b,c]];

         T1[dla,b,c]=starp[inv[twist[[a]]],Sum[
         dotp[Map[Flatten[#,1]&,Flatten[inj[[n]],1]],
              Map[Flatten[#,1]&,Flatten[proj[[n]],1]]]
          ,{n,Length[inj]}]];

        If[testform, T1[dla,b,c]=starp[Qdim[[dlc]],
              dotp[T1[dla,b,c],rightinv[Form[dla,b,c]]]];]

     ,  (*if dla==b && b=!=c && testform *)

      T1[dla,b,c]=starp[Qdim[[a]],dotp[Pairing1[True,a,b,c],
              rightinv[starp[twist[[b]],Form[dla,b,c]]]]];
]]
);


(*Main Loop*)

FindQdim[]; Findtwist[];
{xa,xb,xc}=sumdone; start=True;
Clear[goodvct,dimFsum,Form,T1,T2];


While[(!class0&&!allclassessum) || (class0&&!class0sum),
  If[xa>=repno,
    If[xb>=repno,
     If[xc>=repno, xc=1,If[!class0sum, xc++;
        While[xc<=repno && !Pos0[[xc]],xc++],xc++]];
     xb=xc;
```

```
    If[!class0sum,
      While[xb<=repno && !Pos0[[xb]],xb++]
     ,If[Pos0[[xc]],While[xb<=repno && Pos0[[xb]],xb++]]]
   ,xb++;
    If[!class0sum, While[xb<=repno && !Pos0[[xb]],xb++]
     ,If[Pos0[[xc]],While[xb<=repno && Pos0[[xb]],xb++]]
    ]];
   xa=xb
  ,       (* if xa<=repno *)
   If[xa==0, xa=first0, xa++;
    If[!class0sum,While[xa<=repno && !Pos0[[xa]],xa++]]]
];

If[xa<=repno && xb<=repno && xc<=repno,

  {a,b,c}=Map[PermRep[[#]]&,{xa,xb,xc}];
  {da,dlb,dlc}=Map[Dual[[#]]&,{a,b,c}];

  If[dimFlist[[a,b,c]]==0, dimFsum[a,b,c]=0
    ,If[OrderedQ[{Map[posrep[#]&,{da,dlb,dlc}]
                                  ,{xa,xb,xc}}],
       announce[
        "Starting "<>ToString[goodvectors[a,b,c]]];
       Findgoodvct[da,b,c];
       announce[
          "Finished "<>ToString[goodvectors[a,b,c]]];
       If[dimFsum[a,b,c]>0,
          announce["Starting "<>
                 ToString[symmetries[a,b,c]]];
          FindSymm[da,b,c];
          announce["Finished "<>
                 ToString[symmetries[a,b,c]]];
          Save[outputfile,goodvct];
          Save[symmfile,Form,T1,T2]]]];
   Save[symmfile,dimFsum];
   sumdone={xa,xb,xc}];
   If[xc>=repno,
```

```
      If[!class0sum,class0sum=True;
                If[numrep0==repno,allclassessum=True]
        ,allclassessum=True]];
    DeleteFile[statusfile];
    Save[statusfile,repno,numrep0,repdone,class0rep,
         allclassesrep, PermRep,sumdone,
         class0sum,allclassessum,goup,class0cat,
         allclassescat,statustable];
    Clear[goodvct,dimFsum,Form,T1,T2]
];

If[dotests,
 Get[outputfile];
 Get[symmfile];
 FindRlist[];
 Findtwist[]; FindQdim[]];



TestT1[a_,b_,c_]:=Module[{a1,c1,c2,tw},

a1=dotp[T1[a,b,c],T1[a,b,c]];
Print["T1.T1 = ",a1];
c1=inv[twist[[a]]];
c2=twist[[c]];
tw=prod[prod[c1,c1],c2];
Print["twists = ",tw];
If[starp[inv[tw],a1]=!=idmatrix[dimFsum[a,b,c]]
  ,Print["***Problem with T1 for ",{a,b,c}]
  ,Print["No problem with T1 for ",{a,b,c}]]];


TestT2[a_,b_,c_]:=Module[{a1,c1,c2,tw},

a1=dotp[T2[a,b,c],T2[a,b,c]];;
Print["T2.T2 = ",a1];
c1=inv[twist[[c]]];
```

```
c2=twist[[a]];
tw=prod[prod[c1,c1],c2];
Print["twists = ",tw];
If[starp[inv[tw],a1]=!=idmatrix[dimFsum[a,b,c]]
  ,Print["***Problem with T2 for ",{a,b,c}]
  ,Print["No problem with T2 for ",{a,b,c}]]];

If[dotests,
  dimFsum[a_,b_,c_]:=0;
  Do[If[dimFsum[a,b,c]>0,If[a==b,TestT1[a,b,c]];
      If[b==c,TestT2[a,b,c]]]
  ,{a,repno},{b,repno},{c,repno}]];
```

E CategorySetup

```
p=7;
alg="A";
rank=2;
class0=True;
algcategory=alg<>ToString[rank]<>"q";
category=alg<>ToString[rank]<>"q"<>"mod"<>ToString[p];
repfile=category<>".rep";
sumfile=category<>".sum";
symmfile=category<>".symm";
outputfile=category<>".category";
statusfile=category<>".status";
class0cat=False;
allclassescat=False;
reports=True;
dotests=False;

Off[General::spell1];
Off[General::spell];
SetDirectory[
"Macintosh HD:Mathematica 2.2.2:QuantumGroups:"<>algcategory];

    (* setup ring operations and linear algebra *)

zero=Table[0,{p-1}];
one=Table[1,{p-1}];
id=Table[If[i==1,1,0],{i,1,p-1}];

prod[a_,b_]:=
 If[a==zero||b==zero,zero,
  If[a==id,b,If[b==id,a,
    Table[a[[1]]*b[[ke+1]] +
    Sum[a[[ie+1]]*(b[[ke-ie+1]]-b[[p-ie]]),{ie,1,ke}] -
    If[ke<p-2,a[[ke+2]]*b[[p-ke-1]],0] +
    Sum[a[[ie+1]]*(b[[p-ie+ke+1]]-b[[p-ie]])
```

```
        ,{ie,ke+2,p-2}]
    ,{ke,0,p-2}]]]];

inv[a_]:= Module[{GG}, If[a==id,id, If[a==zero,
  Print["inv has been called with zero argument."],
  GG= Table[ If[ia==1,a[[ka+1]],
       If[ia>1 && ia<=ka+1,a[[ka-ia+2]]-a[[p-ia+1]],
          If[ia==ka+2,-a[[p-ka-1]],
           If[ia>ka+2 && ia<=p-1
             ,a[[p-ia+ka+2]]-a[[p-ia+1]]]]]]
              ,{ka,0,p-2}, {ia,p-1}];
            Qtriang[Transpose[GG],True][[-1,1]]]] ];

vexp[x_]:=If[Mod[x,p]=!=p-1,
           ReplacePart[zero,1,Mod[x,p]+1],
           Table[-1,{p-1}]];

starp[scal_,matrix_]:=If[scal==id,matrix
                        ,Map[prod[scal,#]&,matrix,{-2}]];

dotp[A_,B_]:= Module[{do},
  If[A===1,B,If[B===1,A, If[A==={}||B==={},{},
   do=Dimensions[A][[-2]];
   Map[Sum[starp[#[[i10]],B[[i10]]],{i10,do}]&,A,{-3}]]]]];

simplpoly[poly_]:=Module[{prep},
  prep=Table[0,{p}];
  Do[     (*over the length of poly*)
     prep[[Mod[poly[[ji,1]],p]+1]]+=poly[[ji,2]];
    ,{ji,Length[poly]}];
  If[prep[[p]]==0, Drop[prep,-1],
   Drop[prep,-1]-prep[[-1]]*one]
];

braket[x_,ip_]:=If[x==0,zero,If[x==1,id,
   simplpoly[Table[{-Pairing[[ip,ip]]/2*(x-1-2*ze),1}
                  ,{ze,0,Mod[x,p]-1}]]]];
```

```
Qtriang[qA_,doparam_]:=Module[
      {qnrows,qncols,qAA,qrow,qcol,qrr,
            qrang,qpointer,qra,qc0,qc1,qparam,qllo,qlle,
            qru,qmul,qde},
      {qnrows,qncols}={Length[qA],Length[qA[[1]]]};
      qAA=qA;
      If[doparam,qparam=IdentityMatrix[qnrows]];
      qpointer=Table[qi,{qi,qnrows}];
      qcol=0; qrow=0; qrang={};
      While[qcol<qncols && qrow<qnrows,
            qrow++; qcol++; qrr=qrow;
            While[qcol<=qncols &&
                  qAA[[qpointer[[qrr]],qcol]]==0,
                  If[qrr<qnrows,qrr++,qcol++; qrr=qrow]];
            AppendTo[qrang,qcol];
            qra=qpointer[[qrr]];
            qpointer=Delete[Insert[qpointer,qra,qrow]
                        ,{qrr+1}];
            If[qcol<=qncols,
               If[qAA[[qra,qcol]]=!=1,
                  qc0=1/qAA[[qra,qcol]];
                  qAA[[qra]]=qc0*qAA[[qra]];
                  If[doparam,
                     qparam[[qra]]=qc0*qparam[[qra]]]];
               Do[qllo=qpointer[[qlle]];
                  qc1=qAA[[qllo,qcol]];
                  If[qc1=!=0,
                     qAA[[qllo]]=
                        qAA[[qllo]]-qAA[[qra]]*qc1;
                     If[doparam,qparam[[qllo]]=
                        qparam[[qllo]]-qparam[[qra]]*qc1]]
                 ,{qlle,qrow+1,qnrows}]];
      ];

     If[qcol<=qncols,qmul=qrow,qmul=qrow-1;
                              qrang=Drop[qrang,-1]];
```

```
    If[qmul>0, qAA=Table[qAA[[qpointer[[qi]]]],{qi,qmul}];
       If[doparam,
          qparam=Table[qparam[[qpointer[[qi]]]]
                       ,{qi,qmul}]];
       qru=qmul;
       While[qru>1,
          Do[qde=qAA[[qllo,qrang[[qru]]]];
             If[qde=!=0,
                qAA[[qllo]]=qAA[[qllo]]-qde*qAA[[qru]];
                If[doparam,qparam[[qllo]]=
                     qparam[[qllo]]-qde*qparam[[qru]]]]
            ,{qllo,qru-1}]; qru--];
          If[doparam,{qmul,qAA,qrang,qparam}
                     ,{qmul,qAA,qrang}]
    ,{0}]
];

Triang[A_,doparam_]:=Module[
   {nrows,ncols,AA,param,pointer,col,row,rang,rr,
    ra,c0,llo,c1,lle,mul,ru,de},

      {nrows,ncols}={Length[A],Length[A[[1]]]};
      AA=A;
      If[doparam,param=Table[If[i==j,id,zero]
                             ,{i,nrows},{j,nrows}]];
      pointer=Table[i,{i,nrows}];
      col=0; row=0; rang={};
      While[col<ncols && row<nrows,
            row++; col++; rr=row;
            While[col<=ncols &&
                  AA[[pointer[[rr]],col]]==zero,
                  If[rr<nrows,rr++,col++; rr=row]];
            AppendTo[rang,col];
            ra=pointer[[rr]];
            pointer=Delete[Insert[pointer,ra,row],{rr+1}];
            If[col<=ncols,
               If[AA[[ra,col]]=!=id,
```

```
                    c0=inv[AA[[ra,col]]];
                    AA[[ra]]=starp[c0,AA[[ra]]];
                    If[doparam,
                       param[[ra]]=starp[c0,param[[ra]]]];
                 Do[llo=pointer[[lle]];c1=AA[[llo,col]];
                    If[c1=!=zero,
                     AA[[llo]]=AA[[llo]]-starp[c1,AA[[ra]]];
                     If[doparam,param[[llo]]=
                        param[[llo]]-starp[c1,param[[ra]]]]]
                   ,{lle,row+1,nrows}]];
      ];

     If[col<=ncols,mul=row,mul=row-1;rang=Drop[rang,-1]];
     If[mul>0, AA=Table[AA[[pointer[[i]]]],{i,mul}];
        If[doparam,
           param=Table[param[[pointer[[i]]]],{i,nrows}]];
        ru=mul;
        While[ru>1,
           Do[de=AA[[llo,rang[[ru]]]];
              If[de=!=zero,
                 AA[[llo]]=AA[[llo]]-starp[de,AA[[ru]]];
                 If[doparam,param[[llo]]=
                      param[[llo]]-starp[de,param[[ru]]]]]
             ,{llo,ru-1}]; ru--];
           If[doparam,{mul,nrows-mul,AA,rang,param}
             ,{mul,numrows-mul,AA,rang}]
     ,{0}]
];

kerp[A_]:=Module[{trdata,dimker,param},
  trdata=Triang[A,True];
  If[trdata=!={0},{dimker,param}=trdata[[{2,5}]];
                  Take[param,-dimker]
  ,1]
];

(*right inverse of a matrix*)
```

```
rightinv[T_]:=Module[{trdata,randim,rnm},
  trdata=Triang[T,True];
  If[trdata=!={0},
    {randim,inm}=trdata[[{1,-1}]], randim=0];
    If[randim<Length[T],
       Print["rightinv called with degenerate matrix"]
      ,inm] ];

(* identitymatrix of dim d, tensor A *)

zerow[dim_]:=Table[zero,{dim}];

Lid[A_,scal_,d_]:=Module[{nc},nc=Length[A[[1]]];
    Flatten[Table[
      Flatten[{zerow[(r-1)nc],starp[scal,A[[s]]]
               ,zerow[(d-r)nc]}
                ,1],{r,d},{s,Length[A]}],1]];

(* A tensor identitymatrix of dim d *)

Rid[A_,scal_,d_]:=Module[{B,nc,R},
  B={}; nc=Length[A[[1]]];
  Do[R=Flatten[Table[Join[{starp[scal,A[[r,i]]]}
               ,zerow[d-1]],{i,nc}],1];
   Do[AppendTo[B,R];
      R=Flatten[{{zero},Drop[R,-1]},1],{s,d}]
  ,{r,Length[A]}]; B];

(* check status, get data *)

announce[string_]:= If[reports, Print[string<>" at",
                       Date[][[4]],":",Date[][[5]]]];
announce["Started"];

asocdone={1,1,1,-1};
Get[statusfile];
```

```
If[If[class0, !class0sum, !allclassessum] ,Print[
        "According to the status file, the ",sumfile,
        " file is not complete.
        Run the SummandsSetup program again with p = ",p];
        Abort[]];

If[(class0&&class0cat) || (Not[class0]&&allclassescat),
        Print["According to the statusfile",
        " the data is already complete."]];

Get[repfile];
Get[sumfile];
Get[symmfile];

Do[posrep[PermRep[[i]]]=i, {i,repno}];
Pos0=Table[If[PermRep[[n]]<=numrep0,True,False]
           ,{n,repno}];

dualrep[a_]:=If[a==0,0,Dual[[a]] ];

Wts[0]={Table[0,{rank}]};
Mult[0]={1};

dimFsum[a_,b_,c_]:=dimFsum[Dual[[a]],Dual[[c]],Dual[[b]]];

sortrep[expr_]:=Sort[expr,(#2<#1)&];

testorder[expr1_,expr2_]:=
    If[sortrep[expr1]===sortrep[expr2],
       True,
       !OrderedQ[{sortrep[expr1],sortrep[expr2]}]];


Orbit[{a_,b_,c_}]:=Module[
 {yya,yyb,yyc,dya,dyb,dyc,ya1,yb1,yc1,newa,newb,newc,
  permu,signat},
```

```
   {yya,yyb,yyc}=Map[posrep[#]&,{a,b,c}];
   {dya,dyb,dyc}=Map[posrep[dualrep[#]]&,{a,b,c}];
   If[testorder[{yya,yyb,yyc},{dya,dyb,dyc}],
      {ya1,yb1,yc1}=sortrep[{yya,yyb,yyc}];
      {newa,newb,newc}=Map[PermRep[[#]]&
                           ,{ya1,yb1,yc1}];
     ,
      {ya1,yb1,yc1}=sortrep[{dya,dyb,dyc}];
      {newa,newc,newb}=Map[dualrep[PermRep[[#]]]&
                           ,{ya1,yb1,yc1}]
];
     permu=If[{newa,newb}=={a,b},{1,2,3},
          If[{newa,newb}=={c,a},{3,1,2},
           If[{newa,newb}=={b,c},{2,3,1},
             If[{newa,newb}=={a,c},{1,3,2},
              If[{newa,newb}=={b,a},{2,1,3},
                 {3,2,1}]]]]];
    If[a==b || b==c || a==c, signat=1,
       signat=Signature[permu]];
    {{newa,newb,newc},permu,signat}
];

dimF[a_,b_,c_]:=
  If[a==0,If[b==dualrep[c],1,0],
   If[b==0, If[a==dualrep[c],1,0],
      If[c==0, If[a==dualrep[b],1,0],
         {a1,b1,c1}=Orbit[{a,b,c}][[1]];
         If[Pos0[[posrep[b1]]]&&Pos0[[posrep[c1]]],
            If[Pos0[[posrep[a1]]],dimFsum[a1,b1,c1],0]
           ,
            If[allclassessum,dimFsum[a1,b1,c1],-1]]
]]];

Do[num[AllRep[[i]]]=i,{i,repno}];
num[Table[0,{rank}]]=0;
Do[numwts[n][Wts[n][[k]]]=k;
```

```
   Do[numbase[n][Basis[n][[k,i]]]={k,i}
     ,{i,Mult[n][[k]]}]
   ,{n,If[class0,numrep0,repno]},{k,Length[Wts[n]]}];

numwts[0][Table[0,{rank}]]=1;
numwts[n_][wt_]:=0;

lengthwts= Table[Length[Wts[a]],{a,repno}];
lwts[a_]:=If[a==0,1,lengthwts[[a]]];

dimrep[a_]:=dimrep[a]=If[a==0,1,
         Sum[Mult[a][[i]],{i,lwts[a]}]];

ran[n_][k_,j_]:=numwts[n][Wts[n][[k]]+
                 If[j>rank,-CC[[j-rank]],CC[[j]]]];

(* Fill in invDmorph, and Action on Duals *)

Do[If[Dual[[n]]<=n,
   invDmorph[n]=Map[rightinv[#]&,Dmorph[n]]
  ,{n,If[class0,numrep0,repno]}];

dualm[n_,k_]:=Module[{alwt,lwn},
  If[Dual[[n]]<=n,Dmorph[n][[k]],
   alwt=Wts[n][[k]].Fweights;
   lwn=Length[Wts[n]];
   starp[vexp[-Sum[alwt[[i]]*Pairing[[i,i]],{i,rank}]]
     ,Transpose[Dmorph[Dual[[n]]][[lwn-k+1]]] ]
]];

dualm[0,1]={{id}};

invdualm[n_,k_]:=Module[{alwt,lwn},
  If[Dual[[n]]<=n,invDmorph[n][[k]],
   alwt=Wts[n][[k]].Fweights;
   lwn=Length[Wts[n]];
   starp[vexp[Sum[alwt[[i]]*Pairing[[i,i]],{i,rank}]]
```

```
      ,Transpose[invDmorph[Dual[[n]]][[lwn-k+1]]] ]
]];

invdualm[0,1]={{id}};

Do[dn=Dual[[n]]; lw=Length[Wts[n]];
   If[n>dn, Preaction=Table[Table[{},{2*rank}]
                            ,{lw}];
   Do[dmr=Dmorph[n][[k]]; curwt=Wts[n][[k]];
      l=lw-k+1;
      Do[l1=numwts[dn][-curwt+CC[[z]]];
        If[l1>0,
         k1=lw-l1+1;
         Preaction[[k,z+rank]]=starp[
          vexp[Wts[dn][[l1,z]]],-dotp[dotp[Dmorph[n][[k]],
          Transpose[Action[dn][[l1,z+rank]]]],
          invDmorph[n][[k1]]]];

         Preaction[[k1,z]]=starp[
          vexp[-Wts[dn][[l1,z]]],-dotp[
          dotp[Dmorph[n][[k1]],
          Transpose[Action[dn][[l,z]]]],
          invDmorph[n][[k]]]]]
        ,{z,rank}],{k,lw-1}];
    Action[n]=Preaction; Clear[Preaction]]
      ,{n,If[class0,numrep0,repno]}];

lpr=Length[Posrlist];

deltaexp[genlist_,nlist_,tlist_]:=Module[{tlist1,gen,lgl},
  tlist1=nlist-tlist; lgl=Length[genlist];
  Mod[Sum[gen=genlist[[i]]; tlist[[i]]*(
        Pairing[[gen,gen]]*tlist1[[i]]/2 +
        Sum[tlist1[[j]]*Pairing[[genlist[[j]],gen]]
           ,{j,i+1,lgl}])
      ,{i,lgl}],p]
];
```

```
factorial[n_,i_]:=If[n==0,id,If[n==1,id,
              Fold[prod,id,Table[braket[h,i],{h,2,n}]]]
];

seqfactorial[path_,genlist_]:=Module[{termsno},
  termsno=Length[genlist];
  If[termsno==1,factorial[path[[1]],genlist[[1]]],
     Fold[prod,id,Table[factorial[path[[h]],genlist[[h]]]
                    ,{h,termsno}]]]
];

checkpath[a_,wtno_,path_,genlist_,decr_]:=Module[
   {test,ss,lph,nwtno},
   test=True;
   ss=1; lph=Length[path]; nwtno=wtno;
   While[ss<=lph && test,
      nwtno=numwts[a][Wts[a][[nwtno]]-If[decr,1,-1]*
            path[[ss]]*CC[[genlist[[ss]]]]];
      If[nwtno==0,test=False];
      ss++];
test];

checkpath1[a_,wtno_,path_,genlist_,decr_]:=Module[
  {lph,nwtno,ss,test},
  lph=Length[path];
  nwtno=numwts[a][Wts[a][[wtno]]-If[decr,1,-1]*Sum[
            path[[jj]]*genlist[[jj]].CC,{jj,lph}]];

  test=If[nwtno==0,False,True];
  ss=lph;
  While[ss>1 && test,
      nwtno=numwts[a][Wts[a][[nwtno]]+If[decr,1,-1]*
            path[[ss]]*genlist[[ss]].CC];
      If[nwtno==0,test=False];
      ss--];
test];
```

```
Cdet=Det[CC];
droot=1;
While[Mod[Cdet*droot-1,p]=!=0,droot++];

(* find components of the weight spaces in product ab *)

Pieces[wt_,b_,c_]:=Pieces[wt,b,c]=Module[{stpc},
   stpc={};
   Do[If[Wts[b][[k]]+Wts[c][[l]]==wt,
      AppendTo[stpc,{k,l}]]
  ,{k,Length[Wts[b]]},{l,Length[Wts[c]]}];
stpc];

Findpieceno[wt_,b_,c_]:=(
   Do[pieceno[wt,b,c][Pieces[wt,b,c][[i]]]=i
      ,{i,Length[Pieces[wt,b,c]]}]
);

pieceno[wt_,b_,c_][x_]:=0;

If[asocdone=={1,1,1,-1},
   asocdone={1,1,1,0};
   Save[outputfile,category,p,AllRep,nRep0,
          nRep1,Dual];
   DeleteFile[statusfile];
   Save[statusfile,repno,Repdone,class0rep,
            allclassesrep, PermRep,sumdone,
            class0sum,allclassessum,goup,
             class0cat,
            allclassescat,asocdone]];

(* renewing status and dimension data if the category
   is changed from class0 to allclasses                *)

FindQdim[]:=(
  Qdim=Table[alphawts=Map[#.Fweights&,Wts[a]];
```

```
            Sum[Mult[a][[k]]*
                vexp[-Sum[alphawts[[k,l]]*Pairing[[l,l]]
                          ,{l,rank}]]
               ,{k,Length[Wts[a]]}]
           ,{a,If[class0,numrep0,repno]}];
);

Findtwist[]:=(
  twist=exptwist={};
  Do[wt1=AllRep[[n]].Fweights;
     AppendTo[exptwist,Mod[Cdet*droot*
              (-Sum[wt1[[ii]]*Pairing[[ii,ii]],{ii,rank}]
               -Sum[wt1[[i]]*wt1[[j]]*Pairing[[i,j]]
                    ,{i,rank},{j,rank}]),p]];
   AppendTo[twist,vexp[exptwist[[-1]]]]
  ,{n,If[allclassesrep,repno,numrep0]}]);

FindQdim[]; Findtwist[];

announce["Finished setup"];
             (** functions **)

decr[a_,wtno_,path_,genlist_]:=Module[
   {ni,lpth,Ad,nwtno,ms,ss,gen},
   ni=1;lpth=Length[path];
   While[ni<=lpth && path[[ni]]==0,ni++];
  If[ni<=lpth,
     Ad=Action[a][[wtno,rank+genlist[[ni]]]];
     nwtno=numwts[a][Wts[a][[wtno]]-CC[[genlist[[ni]]]]];
     Do[            (*rr over length of path*)
      ms=path[[rr]]; ss=If[rr==ni,2,1];gen=genlist[[rr]];
      While[ss<=ms,
            Ad=dotp[Ad,Action[a][[nwtno,gen+rank]]];
            ss++;
            nwtno=numwts[a][Wts[a][[nwtno]]-CC[[gen]]]];
     ,{rr,ni,lpth}];
    {nwtno,starp[inv[seqfactorial[path,genlist]],Ad]}
```

```
   ,{wtno,Table[If[i==j,id,zero]
           ,{i,Mult[a][[wtno]]},{j,Mult[a][[wtno]]}]}]];

prgodown[b1_,c1_][genlist_,path_,wt_,vect_]:=Module[
  {pieces,dimparam,dimvect,lgl,newwt,newpc,lnpc,newvect,
   lwb,lwc,lpc,vpc,k,l,bpath,cpath,bnorm},
  If[path=!={},
    pieces=Pieces[wt,b1,c1];
    dimparam=Length[vect[[1,1]]];
    dimvect=Length[vect[[1]]];
    lgl=Length[path];
    newwt=wt-Sum[path[[i]]*CC[[genlist[[i]]]],{i,lgl}];
    Findpieceno[newwt,b1,c1];
    newpc=Pieces[newwt,b1,c1];
    lnpc=Length[newpc];
    newvect=Table[Table[zero,{dimvect}],{dimparam}
              ,{Mult[b1][[newpc[[i,1]]]]}
              ,{Mult[c1][[newpc[[i,2]]]]}]
             ,{i,Length[newpc]}];
    lwb=Length[Wts[b1]];
    lwc=Length[Wts[c1]];
    lpc=Length[vect];

  Do[        (*qq over Length[vect]*)
   vpc=vect[[qq]];
   {k,l}=pieces[[qq]];
   bpath=Table[If[i<lgl,0,-1],{i,lgl}];
   While[bpath=!=path,
     j=-1;
     While[bpath[[j]]==path[[j]],bpath[[j]]=0;j--];
     bpath[[j]]++;
     cpath=path-bpath;
     If[checkpath[b1,k,bpath,genlist,True] &&
        checkpath[c1,l,cpath,genlist,True],
        {k1,B1}=decr[b1,k,bpath,genlist];
        {l1,C1}=decr[c1,l,cpath,genlist];
        bnorm=Table[0,{rank}];
```

```
        Do[bnorm[[genlist[[i]]]]+=bpath[[i]]
          ,{i,lgl}];
        newvect[[pieceno[newwt,b1,c1][{k1,l1}]]]+=
         starp[vexp[-Sum[Wts[c1][[l1,i]]*Pairing[[i,i]]/2*
                        bnorm[[i]], {i,rank}]
                   -deltaexp[genlist,path,bpath]],
         dotp[Transpose[dotp[Transpose[vpc,{1,2,4,3}]
                             ,B1],{1,2,4,3}],C1]]
]
]
   ,{qq,lpc}];
newvect, vect]
];

halflpr=Table[Sum[Posrlist[[k,i]]*Posrlist[[k,j]]*
                  Pairing[[i,j]]/2,{i,rank},{j,rank}]
             ,{k,lpr}];
power[elem_,n_]:=Module[{n1,ff,result},
   If[n==0,id,n1=If[n>0,n,-n];
      ff=2;result=elem;
      While[ff<=n1,result=prod[result,elem];ff++];
      If[n>0,result,inv[result]]]];

Rnorm[explist_,bar_]:=Module[
  {niu,barniu,sumexp,ss,vterm},

niu=Sum[explist[[i]]*Posrlist[[i]],{i,lpr}];
barniu=Sum[niu[[i]],{i,rank}];
sumexp=Sum[explist[[i]],{i,lpr}];
ss=1; vterm=id;
While[ss<=lpr,If[explist[[ss]]=!=0,
      vterm=prod[
        power[vexp[halflpr[[ss]]]-vexp[-halflpr[[ss]]]
              ,2*explist[[ss]]],vterm]];
      ss++];
(-1)^(Mod[barniu+If[bar,sumexp,0],2])*prod[
   vexp[If[bar,-1,1]*(
```

```
        Sum[niu[[i]]*Pairing[[i,i]]/2,{i,rank}]-
        Sum[explist[[i]]*(explist[[i]]+1)*halflpr[[i]]/2
            ,{i,lpr}])],
   prod[vterm,inv[Fold[prod,id,Flatten[
       Table[(vexp[l*halflpr[[k]]]-vexp[-l*halflpr[[k]]])
            ,{k,lpr},{l,explist[[k]]}],1]]]]
];

Bar[elem_]:=Module[{newelem},
   newelem=zero;
   Do[newelem+=elem[[i]]*vexp[-i+1],{i,p-1}];
   newelem];

applyR[inverse_,b_,c_][wt_,vect_]:=Module[

{pieces,lpc,vect1,path0,corr,newvect,k,l,curpath,curj,
 curtest,bwtslist,boperlist,newwtslist,newoperlist,
 oper,coef,curwts,k1,u,r,curoper,curcoef,cwtslist,
 coperlist,l1,npcno,pc,newpc,act1,act2,r1},

pieces=Pieces[wt,b,c];
Findpieceno[wt,b,c];
lpc=Length[pieces];

vect1=If[inverse,vect,
   Table[{k,l}=pieces[[s]];
     {alpb,alpc}=
            {Wts[b][[k]].Fweights,Wts[c][[l]].Fweights};
     starp[vexp[-Cdet*droot*Sum[alpb[[i]]*alpc[[j]]*
                    Pairing[[i,j]],{i,rank},{j,rank}]]
          ,vect[[s]]]
     ,{s,lpc}]];

path0=Table[p-1,{lpr}];
corr=Table[1,{lpr}];
newvect=vect1;
```

```
Do[          (* s over lpc*)
   {k,l}=pieces[[s]];
   curpath=Table[0,{i,lpr}];
   curj=0; curtest=True;
   While[curj>-lpr || (curtest && curpath[[-lpr]]<p-1),
    j=-1;
    While[If[curtest,curpath[[j]]==path0[[j]],j>=curj]
         ,curpath[[j]]=0;j--];
     curpath[[j]]++;
     curtest=False;
     If[checkpath1[b,k,curpath,Posrlist,True]
        && checkpath1[c,l,curpath,Posrlist,False],
        bwtslist={{k}}; boperlist={{{},id}};
        Do[r=1;              (*t over lpr*)
           While[r<=curpath[[t]] && bwtslist=!={},
              newwtslist={}; newoperlist={};
              Do[        (* n over Rlist[[t]]*)
                {oper,coef}=Rlist[[t,n]];
              Do[     (*m over the length of bwtslist*)
                 curwts=bwtslist[[m]];
                 {curoper,curcoef}=boperlist[[m]];
                 k1=curwts[[-1]];u=1;
                 While[k1=!=0 && u<=Length[oper],
                       k1=ran[b][k1,oper[[u]]+rank];
                       AppendTo[curwts,k1];
                       AppendTo[curoper,oper[[u]]+rank];
                       u++];
                 If[k1=!=0,AppendTo[newwtslist,curwts];
                   AppendTo[newoperlist
                     ,{curoper,prod[curcoef,coef]}]];
               ,{m,Length[bwtslist]}];
              ,{n,Length[Rlist[[t]]]}];
              bwtslist=newwtslist;
              boperlist=newoperlist;r++]
         ,{t,lpr}];

      If[bwtslist=!={},
```

```
cwtslist={{1}}; coperlist={{{},id}};
Do[r=1;        (*t over lpr*)
   While[r<=curpath[[t]] && cwtslist=!={},
      newwtslist={}; newoperlist={};
     Do[        (* n over Rlist[[t]]*)
        {oper,coef}=Rlist[[t,n]];
       Do[     (*m over the length of cwtslist*)
          curwts=cwtslist[[m]];
          {curoper,curcoef}=coperlist[[m]];
          k1=curwts[[-1]];u=1;
          While[k1=!=0 && u<=Length[oper],
                k1=ran[c][k1,oper[[u]]];
                AppendTo[curwts,k1];
                AppendTo[curoper,oper[[u]]];
                u++];
          If[k1=!=0,AppendTo[newwtslist,curwts];
            AppendTo[newoperlist
              ,{curoper,prod[curcoef,coef]}]];
         ,{m,Length[cwtslist]}];
        ,{n,Length[Rlist[[t]]]}];
      cwtslist=newwtslist;
      coperlist=newoperlist;r++]
  ,{t,lpr}];

If[bwtslist=!={} && cwtslist=!={},
   curtest=True;
   {k1,l1}={bwtslist[[-1,-1]],cwtslist[[-1,-1]]};
   npcno=pieceno[wt,b,c][{k1,l1}];
   pc=Transpose[vect1[[s]],{1,2,4,3,5}];
   newpc=Sum[ (*t up to Length[boperlist]*)
              (*s1 up to Length[coperlist]*)
    act1=Action[b][[bwtslist[[t,1]],
                   boperlist[[t,1,1]] ]];
    r=2;
    While[r<=Length[boperlist[[t,1]]],
       act1=dotp[act1,Action[b][[bwtslist[[t,r]],
                   boperlist[[t,1,r]] ]]];
```

```
            r++];
          curpc=Transpose[ dotp[pc,act1],{1,2,4,3,5}];
          act2=Action[c][[cwtslist[[s1,1]],
                         coperlist[[s1,1,1]] ]];
          r1=2;
          While[r1<=Length[coperlist[[s1,1]]],
            act2=dotp[act2,Action[c][[cwtslist[[s1,r1]],
                           coperlist[[s1,1,r1]] ]]];
            r1++];
          starp[If[!inverse,
           prod[boperlist[[t,2]],coperlist[[s1,2]]]
          ,Bar[prod[boperlist[[t,2]],coperlist[[s1,2]]]]]
              ,dotp[curpc,act2]]
          ,{t,Length[boperlist]}
          ,{s1,Length[coperlist]}];

        newvect[[npcno]]+=
                starp[Rnorm[curpath,inverse],newpc]
        ]]];
curj=j],{s,lpc}];

If[!inverse,newvect,
   Table[{k,l}=pieces[[s]];
     {alpb,alpc}=
            {Wts[b][[k]].Fweights,Wts[c][[l]].Fweights};
      starp[vexp[Cdet*droot*Sum[alpb[[i]]*alpc[[j]]*
                  Pairing[[i,j]],{i,rank},{j,rank}]]
                 ,newvect[[s]]]
     ,{s,lpc}]]
 ];

Comm[inverse_,b_,c_][wt_,vect_]:=

If[!inverse,applyR[False,c,b][wt,
          Reverse[Map[Transpose[#,{1,2,4,3,5}]&,vect]]]
  ,
   Reverse[Map[Transpose[#,{1,2,4,3,5}]&,
```

```
                applyR[True,b,c][wt,vect]]]];

FindInj[a_,b_,c_]:=Module[

{inj,curinj,alwts,vct,prf,mm,eo,k1,j1,curpc,upperpc,
 us1,us2,lme,curwt,lcpc,upperwt,l,m,l1,m1,part1,part2},

announce["Starting "<>ToString[findinj[a,b,c]]];
inj={goodvct[a,b,c]}; lgv=Length[goodvct[a,b,c][[1,1]]];
alwts=Length[Wts[a]]; lme=Length[maxelem[a]];
Do[(*Print[k]*)
   curwt=Wts[a][[k]];
   curpc=Pieces[curwt,b,c]; lcpc=Length[curpc];
   Findpieceno[curwt,b,c];
   curinj=Table[{},{lcpc}];
   Do[
      vct=Basis[a][[k,j]];
      prf=Length[vct];  mm=vct[[-1]];
      If[mm>1, vct[[-1]]--,
         eo=-1; While[-eo<prf && vct[[eo-1]]==0,eo--];
         vct=Drop[vct,eo]];
      {k1,j1}=numbase[a][vct];
      f=maxelem[a][[prf]];
      upperwt=Wts[a][[k1]];
      upperpc=Pieces[upperwt,b,c];
      Findpieceno[upperwt,b,c];
      Do[
         {l,m}=curpc[[s]];
         l1=numwts[b][Wts[b][[l]]+CC[[f]]];
         us1=pieceno[upperwt,b,c][{l1,m}];
         m1=numwts[c][Wts[c][[m]]+CC[[f]]];
         us2=pieceno[upperwt,b,c][{l,m1}];
         part2=If[m1>0,dotp[inj[[k1,us2,j1]],
                 Action[c][[m1,rank+f]]],0];
         part1=If[l1>0,Transpose[dotp[Transpose[
            inj[[k1,us1,j1]],{1,3,2}],
            starp[vexp[-Wts[c][[m,f]]*Pairing[[f,f]]/2],
```

```
                    Action[b][[l1,rank+f]]]],{1,3,2}],0];
          curinj[[s]]=Append[curinj[[s]],If[mm>1,starp[
                 inv[braket[mm,f]],
                 If[part1===0,If[part2===0,
                    Table[zero,{lgv}
                              ,{Mult[b][[l]]},{Mult[c][[m]]}]
                  ,part2],If[part2===0,part1,part1+part2]]]
                ,If[part1===0,If[part2===0,
                    Table[zero,{lgv}
                              ,{Mult[b][[l]]},{Mult[c][[m]]}]
                  ,part2],If[part2===0,part1,part1+part2]]]]
         ,{s,lcpc}]
     ,{j,Length[Basis[a][[k]]]}];

 AppendTo[inj,curinj]

 ,{k,2,alwts}];

 Inj[a,b,c]=inj;
 announce["Finished "<>ToString[findinj[a,b,c]]]];


test[trpl_]:= OrderedQ[{Map[posrep[dualrep[#]]&,trpl],
                        Map[posrep[#]&,trpl]}];
testform[trpl_]:=(Sort[trpl]===Sort[Map[dualrep[#]&,trpl]]);
testuneq[a_,b_,c_]:=(a=!=b && b=!=c && a=!=c);
ss[a_,k_]:=Module[{lambda},
             Sum[lambda=Wts[a][[k]].Fweights;
                  lambda[[i]]*Pairing[[i,i]] ,{i,rank}]];

zerow[d_]:=zerow[d]=Table[0,{d}];

memorizedFinj[a_,b_,c_]:=False;

FindFinj[{a_,b_,c_},sign_]:=(

 If[!memorizedFinj[a,b,c],
```

```
    FindInj[dualrep[a],b,c];
    Finj[a,b,c]=
      Table[Map[Transpose[
            dotp[invdualm[dualrep[a],k],#],{2,1,3,4}]&,
                 Inj[dualrep[a],b,c][[k]]],{k,lwts[a]}];
    memorizedFinj[a,b,c]=True;
    (*Clear[Inj]*)];
 If[sign==-1 && !memorizedFinj[a,c,b],
   announce["Starting applying Comm on"<>ToString[{a,b,c}]];
   goodvct[dualrep[a],c,b]=Comm[Not[test[{a,b,c}]],b,c][
           Wts[dualrep[a]][[1]],goodvct[dualrep[a],b,c]];
   announce["Finished applying Comm on"<>
       ToString[{dualrep[a],b,c}]];
   FindInj[dualrep[a],c,b];
   Finj[a,c,b]=
      Table[Map[Transpose[
            dotp[invdualm[dualrep[a],k],#],{2,1,3,4}]&,
                 Inj[dualrep[a],c,b][[k]]],{k,lwts[a]}];
    (*Clear[Inj]*)
   memorizedFinj[a,c,b]=True];
);

pcFinj[a_,b_,c_][k_,l_,m_]:=Module[

{a1,b1,c1,ppp,sign,k1,l1,m1,piece,bark1,awt,pcno},

If[Wts[a][[k]]+Wts[b][[l]]+Wts[c][[m]]=!=
   Table[0,{rank}],
   Print["pcFinj has been called with improper
          arguments"];{},
If[a==0,If[b==dualrep[c],
      {{invdualm[c,m]}},{}]
,If[b==0,If[a==dualrep[c],
      Table[invdualm[c,m][[i,j]],{i}
           ,{i,Mult[a][[k]]},{i},{j,Mult[c][[m]]}],{}]
,If[c==0,If[b==dualrep[a],
      Table[invdualm[b,l][[i,j]],{i}
```

```
          ,{i,Mult[a][[k]]},{j,Mult[b][[l]]},{1}],{}]
,
piece={k,l,m};
{{a1,b1,c1},ppp,sign}=Orbit[{a,b,c}];
FindFinj[{a1,b1,c1},sign];
If[sign==-1,ppp={ppp[[1]],ppp[[3]],ppp[[2]]};
   {b1,c1}={c1,b1}];
{k1,l1,m1}=Map[piece[[#]]&,ppp];
bark1=lwts[a1]-k1+1;
awt=Wts[dualrep[a1]][[bark1]];
Findpieceno[awt,b1,c1];
pcno=pieceno[awt,b1,c1][{l1,m1}];
If[ppp==={1,2,3},Finj[a,b,c][[bark1,pcno]]
 ,
 If[testuneq[a,b,c],
  If[ppp==={1,3,2},Finj[a,b,c][[bark1,pcno]],
   scal=vexp[If[test[If[sign==1,{a1,b1,c1},{a1,c1,b1}]],
    If[ppp==={3,1,2},ss[a1,k1],
    If[ppp==={2,3,1},ss[a1,k1]+ss[b1,l1],
    ]]
   ,    (* if !test[{a1,b1,c1}] *)
    If[ppp==={3,1,2},-ss[c1,m1]-ss[b1,l1],
    If[ppp==={2,3,1},-ss[c1,m1],
    ]]]];
   starp[scal,Transpose[Finj[a1,b1,c1][[bark1,pcno]]
        ,{1,ppp[[1]]+1,ppp[[2]]+1,ppp[[3]]+1}]]]

 ,     (*if !testuneq *)

  If[test[{a1,b1,c1}],
   If[ppp==={3,1,2},
     starp[vexp[ss[a1,k1]],
        Transpose[Finj[a1,b1,c1][[bark1,pcno]],{1,4,2,3}]],
   If[ppp==={2,3,1},
      Transpose[starp[vexp[ss[a1,k1]+ss[b1,l1]],
         Finj[a1,b1,c1][[bark1,pcno]]],{1,3,4,2}]]]
   ,                (*if !test[{a1,b1,c1}] *)
```

```
    If[ppp===={3,1,2},
       Transpose[starp[vexp[-ss[b1,l1]-ss[c1,m1]],
           Finj[a1,b1,c1][[bark1,pcno]]],{1,4,2,3}],
    If[ppp===={2,3,1},
       starp[vexp[-ss[c1,m1]],
        Transpose[Finj[a1,b1,c1][[bark1,pcno]],{1,3,4,2}]]]]]

]]]
]]]];

pcInj[a_,b_,c_][k_,l_,m_]:=

If[Wts[a][[k]]=!=Wts[b][[l]]+Wts[c][[m]],
   Print["pcInj has been called with improper
          arguments"];{},
   dotp[dualm[a,k],Transpose[
        pcFinj[dualrep[a],b,c][lwts[a]-k+1,l,m],{2,1,3,4}]]

];

testform1[a_,b_,c_]:=(a==dualrep[a] && b==dualrep[b]
                      && c==dualrep[c]);

testform2[a_,b_,c_]:=(a==dualrep[b] && c==dualrep[c]);

testform3[a_,b_,c_]:=(a==dualrep[a] && b==dualrep[c]);

pcProj[b_,c_,a_][l_,m_,k_]:=Module[

{barm,barl,dc,db,aa1,bb1,cc1,perm,sign,prepcproj},

If[Wts[a][[k]]=!=Wts[b][[l]]+Wts[c][[m]],
   Print["pcProj has been called with improper
          arguments"];{},

   {barm,barl}={lwts[c]-m+1,lwts[b]-l+1};
   {dc,db}={dualrep[c],dualrep[b]};
```

```
   prepcproj=Transpose[dotp[Transpose[dotp[
    pcFinj[a,dc,db][k,barm,barl],dualm[db,barl]],{1,2,4,3}],
    dualm[dc,barm]],{3,4,1,2}];

 If[a==0,If[b==0||c==0, prepcproj,
         starp[inv[Qdim[[b]]],prepcproj]],
 If[b==0||c==0, prepcproj,
 {{aa1,bb1,cc1},perm,sign}=Orbit[{a,dc,db}];
 If[!testform[{a,dc,db}],
   starp[prod[inv[Qdim[[aa1]]],Qdim[[a]]],prepcproj]
  ,
   prepcproj=Transpose[dotp[Transpose[prepcproj,{1,2,4,3}]
    ,starp[Qdim[[a]],rightinv[Form[aa1,bb1,cc1]]]],{1,2,4,3}];
   If[!testuneq[aa1,bb1,cc1] || sign==1,prepcproj,
    If[testform2[aa1,bb1,cc1],
       starp[vexp[exptwist[[cc1]]],prepcproj],
       starp[vexp[2*exptwist[[bb1]]-exptwist[[aa1]]],
            prepcproj]]]
]]
]];


FindAssoc[a_,b_,c_,e_]:=Module[

{da,db,dc,de,xx,yy,tempassoc,x,y,zerom,axepc,
 currow,xn,en,xbcpc,pctable,bn,cn,yn},
announce["Starting "<>ToString[assoc[a,b,c,e]]];
{da,db,dc,de}=Map[dualrep[#]&,{a,b,c,e}];
(*find intermidiate objects*)
xx={}; yy={};
Do[If[dimF[a,u,e]>0 && dimF[u,db,dc]>0,
      AppendTo[xx,u]];
   If[dimF[a,b,u]>0 && dimF[u,dc,de]>0,
      AppendTo[yy,u]]
  ,{u,0,repno}];
```

```
Assoc[a,b,c,e]=

If[xx=={},{{}},
   tempassoc={};
   Do[
     x=xx[[r]];
     currow={};
    Do[
       y=yy[[s]];
       axepc=Pieces[Wts[da][[1]],x,e];
       (*If[x=!=0,FindFinj[Orbit[{a,x,e}][[1]]]];
       If[x=!=0,
          FindFinj[Orbit[{dualrep[x],b,c}][[1]]]];
       If[y=!=0,FindFinj[Orbit[{y,dc,de}][[1]]]];
       If[y=!=0,
          FindFinj[Orbit[{da,db,dualrep[y]}][[1]]]];*)

       zerom=Table[zero,
              {dimF[a,x,e]*dimF[dualrep[x],b,c]}
             ,{dimF[a,b,y]*dimF[dualrep[y],c,e]}];

       currow=Join[currow, Transpose[
       Sum[          (*over axepc*)
           {xn,en}=axepc[[i]];
           xbcpc=Pieces[Wts[x][[xn]],b,c];
           pctable={};
           Do[{bn,cn}=xbcpc[[j]];
              yn=numwts[y][Wts[c][[cn]]+Wts[e][[en]]];
              If[yn>0,AppendTo[pctable,{bn,cn,yn}]]
              ,{j,Length[xbcpc]}];
           If[pctable==={},zerom,
             Sum[      (*over pctable*)
                 {bn,cn,yn}=pctable[[j]];
                  Flatten[Map[Flatten[#,1]&,
                  dotp[Map[Flatten[#,1]&,Transpose[
                  dotp[Map[Flatten[#,1]&,Transpose[
                  dotp[Transpose[
```

```
                    pcInj[da,x,e][1,xn,en][[1]],{1,3,2}]
                   ,pcInj[x,b,c][xn,bn,cn]],{1,5,2,3,4}]
                   ,{3}]
                   ,Flatten[pcProj[c,e,y][cn,en,yn],1]]
                   ,{1,2,4,3,5}],{3}]
                   ,Map[Flatten[#,1]&,Flatten[
                            pcProj[b,y,da][bn,yn,1],1]]]
                   ,{2}],1]

                 ,{j,Length[pctable]}]]
               ,{i,Length[axepc]}]]]
        ,{s,Length[yy]}];
        tempassoc=Join[tempassoc,Transpose[currow]]
       ,{r,Length[xx]}];

     tempassoc];
announce["Finished "<>ToString[assoc[a,b,c,e]]];
];


(*** main loop for .fullassoc ***)

{a,b,c,e}=asocdone;
While[If[class0,!class0cat,!allclassescat],
      If[e<If[!class0cat,numrep0,repno],
         If[class0cat && a<=numrep0 && b<=numrep0 &&
             c<=numrep0 && e<numrep0,e=numrep0+1, e++],
       If[c<If[!class0cat,numrep0,repno],c++; e=1,
         If[b<If[!class0cat,numrep0,repno],b++; c=e=1,
          If[a<If[!class0cat,numrep0,repno],a++; b=c=e=1]]]];

      FindAssoc[a,b,c,e];
      If[Assoc[a,b,c,e]=!={{}},
         Save[outputfile,Assoc]];
      asocdone={a,b,c,e};
      If[asocdone=={numrep0,numrep0,numrep0,numrep0},
         class0cat=True];
```

```
      If[asocdone=={repno,repno,repno,repno},
         allclassescat=True];
      Clear[Assoc];
      DeleteFile[statusfile];
      Save[statusfile,repno,Repdone,class0rep,
            allclassesrep, PermRep,sumdone,
            class0sum,allclassessum,goup,
             class0cat,
            allclassescat,asocdone]];

(*test functions*)

TestComp[a_,b_,c_,wt_]:=Module[

{tpc,Comp,k,l},

announce["Starting "<>ToString[testcomp[a,b,c,wt]]];

tpc=Pieces[Wts[a][[wt]],b,c];
Comp=Sum[{k,l}=tpc[[ii]];
             dotp[Map[Flatten[#,1]&,Flatten[
                           pcInj[a,b,c][wt,k,l],1]],
                  Map[Flatten[#,1]&,Flatten[
                           pcProj[b,c,a][k,l,wt],1]]]
            ,{ii,Length[tpc]}];

Print["The composition for ",{a,b,c,wt}," is ",Comp,"."];
];


TestpcFunct[a_,b_,c_]:=Module[

{lwa,lgv,problem,pc,z,opz,q,pc1,mm,upinj,upproj,
 actinj,projact,r,s,r1,s1,injzero,projzero,injactpc,
 actprojpc},

announce["Starting "<>ToString[testpcfunct[a,b,c]]];
```

```
lwa= Length[Wts[a]];
lgv=dimF[dualrep[a],b,c];
problem=False; k=0;
While[k<lwa && !problem,
      k++;
      announce["Starting k="<>ToString[k]];
      pc=Pieces[Wts[a][[k]],b,c];
      z=0;
      While[z<1 && !problem,
        z++;
        announce["Starting z="<>ToString[z]];
        opz=If[z<=rank,z+rank,z-rank];
        q=ran[a][k,z];
        pc1={};
        Do[mm=numwts[c][Wts[a][[k]]+
             If[z<=rank,CC[[z]],-CC[[z-rank]]]-
             Wts[b][[ll]]];
          If[mm=!=0,AppendTo[pc1,{ll,mm}]]
         ,{ll,lwts[b]}];

        If[q>0,
           upinj=Table[{r,s}=pc1[[i]];
              pcInj[a,b,c][q,r,s],{i,Length[pc1]}];
           upproj=Table[{r,s}=pc1[[i]];
              pcProj[b,c,a][r,s,q],{i,Length[pc1]}];
           actinj=Map[dotp[Action[a][[k,z]],#]&,
                           upinj ];
           projact=Map[dotp[#,Action[a][[q,opz]]]&,
                           upproj]];
        If[pc1=!={},
           Do[
             {r,s}=pc1[[v]];
             r1=ran[b][r,opz];
             s1=ran[c][s,opz];
             injzero=Table[zero,{Mult[a][[k]]},{lgv}
                      ,{Mult[b][[r]]},{Mult[c][[s]]}];
             projzero=Transpose[injzero,{4,3,1,2}];
```

```
injactpc=
  If[r1=!=0,
   If[z<=rank,
       Transpose[dotp[Transpose[
        pcInj[a,b,c][k,r1,s]
        ,{1,2,4,3}],Action[b][[r1,z]]]
        ,{1,2,4,3}]
     ,starp[vexp[-Wts[c][[s,opz]]*
                Pairing[[opz,opz]]/2],
       Transpose[dotp[Transpose[
         pcInj[a,b,c][k,r1,s]
        ,{1,2,4,3}],Action[b][[r1,z]]]
        ,{1,2,4,3}]]]
  ,injzero]
+If[s1=!=0,
  If[z<=rank,
  starp[vexp[Wts[b][[r,z]]*Pairing[[z,z]]/2],
     dotp[pcInj[a,b,c][k,r,s1]
          ,Action[c][[s1,z]]]]
   ,
     dotp[pcInj[a,b,c][k,r,s1]
          ,Action[c][[s1,z]]]]
 ,injzero];


actprojpc=
   If[r1=!=0,If[z<=rank,
      starp[vexp[-Wts[c][[s,z]]*
            Pairing[[z,z]]/2]
       ,dotp[Action[b][[r,opz]]
           ,pcProj[b,c,a][r1,s,k]]]
     ,dotp[Action[b][[r,opz]]
          ,pcProj[b,c,a][r1,s,k]]]
     ,projzero] +
   If[s1=!=0,If[z<=rank,
      Transpose[dotp[Action[c][[s,opz]]
         ,Transpose[pcProj[b,c,a][r,s1,k]
```

```
                        ,{2,1,3,4}]],{2,1,3,4}]
          ,starp[vexp[
           Wts[b][[r,opz]]*Pairing[[opz,opz]]/2],
            Transpose[dotp[Action[c][[s,opz]]
              ,Transpose[pcProj[b,c,a][r,s1,k]
                        ,{2,1,3,4}]],{2,1,3,4}]]]
       ,projzero];

     If[(q>0 && injactpc=!=actinj[[v]]) ||
        (q==0 && injactpc=!=injzero)
        ,problem=True;
         Print["injactpc=",injactpc];
         Print["actinjpc=",If[q>0,actinj[[v]],
                injzero]];
         Print["Problem with "<>
             ToString[injection[a,b,c]]<>
             "and z=",z," on the ",
             k,"th wtspace of ",a,"."]];

      If[(q>0 && actprojpc=!=projact[[v]]) ||
        (q==0 && actprojpc=!=projzero)
        ,problem=True;
         Print["actprojpc=",actprojpc];
         Print["projactpc=",If[q>0,projact[[v]],
                projzero]];
         Print["Problem with "<>
             ToString[projection[a,b,c]]<>
             "and z=",opz,",v=",v," into the ",
             k,"th wtspace of ",a,"."]];

       ,{v,Length[pc1]}];
     , (*if pc1=={} *)
      If[q=!=0, problem=True;
       Print["Problem with "<>
            ToString[injection[a,b,c]]<>
            "on the ",k,"th wtspace of ",
            a,"."] ]];
```

```
      (*announce["Finished z="<>ToString[z]]*)
]];
If[!problem,Print[ToString[injection[a,b,c]]<>" and "<>
              ToString[projection[b,c,a]]<>" are fine."]]
];

        (*main loop for the test functions*)

If[dotests,
  problem=False;
  Do[If[dimF[a,b,c]>0 && OrderedQ[{{1,3,3},{a,b,c}}],
    da=dualrep[a];
    If[!problem, TestpcFunct[da,b,c]];
    If[!problem,TestComp[da,b,c,1]]]
   ,{a,repno},{b,repno},{c,repno}]];
```

Vita

I was born in Shoumen, Bulgaria, on July the 20th. 1964. I obtained the Master's degree in Physics in October 1987 at the University of Sofia, Bulgaria, with a thesis in Suppersymmetric Quantum Field Theory under the lead of Dr. Bonka Ivanova.

In the Spring of 1988, I was on specialization fellowship at the Department of Physics at the University of Sofia, where I worked as assistant to the course in Mathematical Methods in Physics.

Since August 1988 I have been a Ph.D student in Mathematics at Virginia Polytechnic Institute and State University. For the period 1989-1993 I was employed as a teaching assistant at the Department of Mathematics, and in Fall 1992 and Spring 1993 I taught courses of elementary calculus. In July, 1991 I attended the Regional Geometry Institute in Park City, Utah.

August 29,1996

I. Bobtcheva

\Ivelina Bobtcheva \