Article Submitted to Journal of Symbolic Computation

Computing with quantized enveloping algebras: PBW-type bases, highest-weight modules, R-matrices

W. A. de Graaf

Mathematical Institute, University of Utrecht, The Netherlands

Abstract

Let $U_q(\mathfrak{g})$ be the quantized enveloping algebra corresponding to the semisimple Lie algebra $\mathfrak{g}$. We describe algorithms to obtain the multiplication table of a PBW-type basis of $U_q(\mathfrak{g})$. We use this to obtain an algorithm for calculating a Gröbner basis of an ideal in the subalgebra U^-, which leads to a general construction of irreducible highest-weight modules over $U_q(\mathfrak{g})$. We also indicate how to compute the corresponding R-matrices.

1. Introduction

Quantized enveloping algebras have been widely studied, almost exclusively by theoretical means (see, e.g., [2], [6], [8]). In this paper we consider the problem of computing with a quantized enveloping algebra. For this we need a basis of it, along with a method for computing the product of two basis elements. To this end we will use so-called PBW-type bases. The main subject of this paper will be an algorithm for computing the product of two elements of such a PBW-type basis. We use this to construct highest-weight modules over quantized enveloping algebras, and the corresponding R-matrices.

Below we recall some notation and definitions. Here, as in the rest of this paper, we borrow heavily from [6]. A citation as [6], 4.15(3) refers to the formula (3) in paragraph 4.15 of [6].

Let $\mathfrak{g}$ be a split semisimple Lie algebra over $\mathbb{Q}$, with root system Φ. We let V be the vector space over $\mathbb{R}$ spanned by Φ. By $W(\Phi)$ we denote the Weyl group of Φ. On V we fix a $W(\Phi)$-invariant inner product $(\ ,\ )$ such that $(\alpha, \alpha) = 2$ for all short roots α. This means that $(\alpha, \alpha) = 2, 4, 6$, where the last possibility only occurs if α comes from a component of type G_2.

We work over the field $\mathbb{Q}(q)$. For $\alpha \in \Phi$ we set

$$q_\alpha = q^{\frac{(\alpha,\alpha)}{2}},$$

and for a non-negative integer n, $[n]_\alpha = q_\alpha^{n-1} + q_\alpha^{n-3} + \cdots + q_\alpha^{-n+1}$. Furthermore, $[n]_\alpha! = [1]_\alpha [2]_\alpha \cdots [n]_\alpha$ and

$$\begin{bmatrix} n \\ k \end{bmatrix}_\alpha = \frac{[n]_\alpha!}{[k]_\alpha ! [n-k]_\alpha !}.$$

Let $\Delta = \{\alpha_1, \ldots, \alpha_l\}$ be a simple system of Φ. Then the quantized enveloping algebra $U_q(\mathfrak{g})$ is the associative algebra (with one) over $\mathbb{Q}(q)$ generated by F_α, K_α, K_α^{-1}, E_α for $\alpha \in \Delta$, subject to the following relations

$$\begin{aligned} K_\alpha K_\alpha^{-1} &= K_\alpha^{-1} K_\alpha = 1, \;\; K_\alpha K_\beta = K_\beta K_\alpha \\ E_\beta K_\alpha &= q^{-(\alpha,\beta)} K_\alpha E_\beta \\ K_\alpha F_\beta &= q^{-(\alpha,\beta)} F_\beta K_\alpha \\ E_\alpha F_\beta &= F_\beta E_\alpha + \delta_{\alpha,\beta} \frac{K_\alpha - K_\alpha^{-1}}{q_\alpha - q_\alpha^{-1}} \end{aligned}$$

together with, for $\alpha \neq \beta$,

$$\sum_{k=0}^{1-\langle \beta, \alpha^\vee \rangle} (-1)^k \begin{bmatrix} 1-\langle \beta, \alpha^\vee \rangle \\ k \end{bmatrix}_\alpha E_\alpha^{1-\langle \beta, \alpha^\vee \rangle - k} E_\beta E_\alpha^k \quad (R_E)$$

$$\sum_{k=0}^{1-\langle \beta, \alpha^\vee \rangle} (-1)^k \begin{bmatrix} 1-\langle \beta, \alpha^\vee \rangle \\ k \end{bmatrix}_\alpha F_\alpha^{1-\langle \beta, \alpha^\vee \rangle - k} F_\beta F_\alpha^k \quad (R_F).$$

Let U^-, U^0, U^+ denote the subalgebras of $U_q(\mathfrak{g})$ generated by respectively the F_α, the $K_\alpha^{\pm 1}$ and the E_α. Then [6], Theorem 4.21 states that the elements FKE, where F, K, E run through bases of U^-, U^0, U^+ respectively, form a basis of $U_q(\mathfrak{g})$. Furthermore, the elements $K_{\alpha_1}^{n_1} \cdots K_{\alpha_l}^{n_l}$ where the n_i are arbitrary elements of $\mathbb{Z}$, form a basis of U^0. So the problem of finding a basis of $U_q(\mathfrak{g})$ boils down to finding bases of U^- and U^+. Also, since these two algebras are isomorphic, the problem is the same in both cases.

A very crude way to find a basis of U^+ is to use the fact that U^+ is isomorphic to the free algebra generated by the E_α modulo the ideal generated by the relations (R_E). It is possible to show that this ideal has a finite Gröbner basis, which we can calculate (see, e.g., [9]). A basis of U^+ is then given by all monomials in the E_α that are not divisible by a leading monomial of an element of the Gröbner basis. However, a much more efficient way of dealing with the problem is via so-called PBW-type bases (cf., e.g., [6], [8], [10]). For $\alpha \in \Delta$ we use the automorphism T_α of $U_q(\mathfrak{g})$, given by the formulas of [6], §8.14. Let $w_0 = s_{\alpha_{i_1}} \cdots s_{\alpha_{i_t}}$ be a reduced expression for the longest element in $W(\Phi)$. For $1 \leq k \leq t$ set $E_k = T_{\alpha_{i_1}} \cdots T_{\alpha_{i_{t-k}}}(E_{\alpha_{t-k+1}})$. It can be shown that the E_k are elements of U^+, and that the elements $E_1^{n_1} \cdots E_t^{n_t}$ (where the $n_i \geq 0$) form a basis of U^+ (cf. [6]). We can define elements $F_k \in U^-$ similarly. Then the elements $F_1^{m_1} \cdots F_t^{m_t} K_{\alpha_1}^{n_1} \cdots K_{\alpha_l}^{n_l} E_1^{p_1} \cdots E_t^{p_t}$ (where $m_i, p_i \geq 0$, $n_i \in \mathbb{Z}$) form a basis of

$U_q(\mathfrak{g})$. This basis is called a PBW-type basis (where PBW stands for Poincaré-Birkhoff-Witt). The elements F_k, $K_{\alpha_j}^{\pm 1}$, E_k are called *PBW-generators* of $U_q(\mathfrak{g})$. It will be convenient to index the F_k, E_k by positive roots. For $1 \leq k \leq t$ set $\beta_k = s_{\alpha_{i_1}} \cdots s_{\alpha_{i_{t-k}}}(\alpha_{i_{t-k+1}})$. Then by E_{β_k} we also denote E_k, and similarly F_{β_k} will denote F_k.

Now we consider the problem of computing the product of two basis elements. For this it would be sufficient to have a set of relations of the form

$$A_j A_i = c_{ji} A_i A_j + \Omega_{ji},$$

where A_i, A_j are PBW-generators, and Ω_{ji} is a linear combination of basis elements. We call such a relation a *skew commutation relation*, or just *commutation relation* if it is clear what we mean. We call the set of all such relations a *multiplication table* of this PBW-type basis of $U_q(\mathfrak{g})$. Parts of the multiplication table follow easily from the defining relations of $U_q(\mathfrak{g})$. For all positive roots $\beta \in \Phi$ and $\alpha \in \Delta$,

$$E_\beta K_\alpha^{\pm 1} = q^{\mp(\alpha,\beta)} K_\alpha E_\beta, \text{ and } K_\alpha^{\pm 1} F_\beta = q^{\mp(\alpha,\beta)} F_\beta K_\alpha$$

(cf. [6], 4.7(1)). We still need the commutation relations of E_α, E_β of E_α, F_β, and of F_α, F_β for arbitrary positive roots $\alpha, \beta \in \Phi$. The main part of this paper deals with describing algorithms for finding these commutation relations (Section 2). Then in Section 3 we apply this to construct highest weight modules over quantized enveloping algebras, and the corresponding R-matrices. Finally in Section 4 we give an example and an account of practical experiences with an implementation of the algorithms in GAP4.

We explain some more notation and terminology to be used in the sequel. Let $w = s_{\alpha_{i_1}} \cdots s_{\alpha_{i_m}}$ be a reduced expression in the Weyl group $W(\Phi)$. Then we set $T_w = T_{\alpha_{i_1}} \cdots T_{\alpha_{i_m}}$. It can be shown that this automorphism depends only on the element of the Weyl group represented by w, not on the particular reduced expression (cf. [6], 8.18). Let $u = s_{\alpha_{j_1}} \cdots s_{\alpha_{j_m}}$ be a second reduced expression in $W(\Phi)$. If the reduced expressions w and u are equal (i.e., $\alpha_{i_k} = \alpha_{j_k}$ for all k) then we write $u = w$. If u and w represent the same element of $W(\Phi)$, then we write $u \equiv w$. We denote the length function on $W(\Phi)$ by ℓ, e.g., $\ell(w) = m$.

In the sequel we will assign a *weight* to certain elements of U^+. Let $e = E_{\alpha_{i_1}} \cdots E_{\alpha_{i_r}}$ be a monomial in the generators of U^+. Then we say that e has *weight* $\sum_{k=1}^{r} \alpha_{i_k}$. Also, if a is a linear combination of monomials of weight μ, then we say that a has weight μ. We note that this means that the elements E_α have weight α for all $\alpha \in \Phi$ (cf. [6], 8.18(4)). Also if $\mu = \sum_{i=1}^{l} k_i \alpha_i$ is such a weight, then the *level* of μ is the number $\sum_{i=1}^{l} k_i$.

In the sequel we will use the term *straightening* to denote the process of rewriting elements of $U_q(\mathfrak{g})$ to normal form, using a set of skew-commutation relations.

2. Finding commutation relations

As before Φ is a root system, with simple system $\Delta = \{\alpha_1, \dots, \alpha_l\}$ and Weyl group $W(\Phi)$. We first describe an algorithm for finding the commutation relation of E_α, E_β, for positive roots α, β. For this we suppose that Φ does not contain components of type G_2. A set of commutation relations for this case is known in the literature. For the sake of completeness we have added such a set in Appendix A.

Let $w = s_{\alpha_{i_1}} \cdots s_{\alpha_{i_m}}$ be a reduced word in $W(\Phi)$. For $1 \leq k \leq m$ set $X_k = T_{\alpha_{i_1}} \cdots T_{\alpha_{i_{m-k}}}(E_{\alpha_{i_{m-k+1}}})$. Then we say that a monomial of the form $X_{i_1}^{n_1} \cdots X_{i_r}^{n_r}$ is a w-monomial. A linear combination of w-monomials is called a w-expression. Let p_w be a w-expression. If for all monomials $X_{i_1}^{n_1} \cdots X_{i_r}^{n_r}$ appearing in p_w we have that $i_1 < i_2 < \cdots < i_r$ then we say that p_w is in *normal form*. Now let u be a different reduced expression for the same element in $W(\Phi)$, i.e., $u \equiv w$. Let p_w be a w-expression; then p_w is equal to p_u, where p_u is a certain u-expression in normal form. (cf. [6], Proposition 8.22). Here we describe a straightforward algorithm for computing p_u given p_w.

First of all, we can transform w into u by a sequence of "elementary moves" (cf. [6], 8.18). The proof of this result in [1] translates to a straightforward algorithm for obtaining such a sequence of elementary moves. We may assume that u is obtained from w by one elementary move. This means that we may assume that we are dealing with the rank 2 case. The generalisation to the general case is straightforward. For the elementary move there are four possibilities, which we treat separately. We denote the simple roots by α, β.

First suppose that the elementary move consists of replacing $s_\alpha s_\beta$ by $s_\beta s_\alpha$, i.e., $\langle \alpha, \beta^\vee \rangle = 0$. In this case $X_1 = T_\alpha(E_\beta) = E_\beta$ and $X_2 = E_\alpha$ commute, and we get p_u simply by interchanging the exponents of X_1, X_2.

Secondly suppose that the elementary move consists of replacing $s_\alpha s_\beta s_\alpha$ by $s_\beta s_\alpha s_\beta$. In this case $\langle \alpha, \beta^\vee \rangle = \langle \beta, \alpha^\vee \rangle = -1$. We set $X_1 = T_\alpha T_\beta(E_\alpha) = E_\beta$, $X_2 = T_\alpha(E_\beta)$ and $X_3 = E_\alpha$. Also set $Y_1 = T_\beta T_\alpha(E_\beta) = E_\alpha$, $Y_2 = T_\beta(E_\alpha)$, $Y_3 = E_\beta$. The problem is to write the w-expression p_w as a linear combination of monomials (in normal form) in the Y's. We have $X_1 = Y_3$, $X_3 = Y_1$ and $X_2 = (1 - q_\alpha^{-2}) Y_1 Y_3 - q_\alpha^{-1} Y_2$. (This last equality is obtained by writing X_2 as an expression in the generators E_α, E_β, substituting $E_\alpha = Y_1$, $E_\beta = Y_2$ and straightening, using the table for A_2 of Appendix A.) We substitute these expressions in p_w, and we get a linear combination of monomials involving Y's. Then we straighten the monomials in this linear combination using the commutation rules for the A_2-case in Appendix A.

Thirly, suppose that $\langle \alpha, \beta^\vee \rangle = -1$ and $\langle \beta, \alpha^\vee \rangle = -2$. We set $X_1 = T_\alpha T_\beta T_\alpha(E_\beta) = E_\beta$, $X_2 = T_\alpha T_\beta(E_\alpha)$, $X_3 = T_\alpha(E_\beta)$, $X_4 = E_\alpha$, and $Y_1 = T_\beta T_\alpha T_\beta(E_\alpha) = E_\alpha$, $Y_2 = T_\beta T_\alpha(E_\beta)$, $Y_3 = T_\beta(E_\alpha)$, $Y_4 = E_\beta$. This case has two subcases. The first one occurs when the elementary move consists of replacing $s_\alpha s_\beta s_\alpha s_\beta$ by $s_\beta s_\alpha s_\beta s_\alpha$. For this case we have to transform an expression in X's into an expression involving Y's. We have that $X_1 = Y_4$ and $X_4 = Y_1$, and

$$X_2 = (1 - q^{-4})Y_1Y_4 - q^{-2}Y_3$$
$$X_3 = (q^{-1} - 2q^{-3} + q^{-5})Y_1^2Y_4 + (q^{-3} - q^{-1})Y_1Y_3 + q^{-2}Y_2$$

We substitute these expressions for the X's, and use the second table for type B_2 of Appendix A, to straighten the result.

The second subcase occurs when $s_\beta s_\alpha s_\beta s_\alpha$ is replaced by $s_\alpha s_\beta s_\alpha s_\beta$. In this case we have

$$Y_2 = (q^{-1} - 2q^{-3} + q^{-5})X_1X_4^2 + (q^{-3} - q^{-1})X_2X_4 + q^{-2}X_3$$
$$Y_3 = (1 - q^{-4})X_1X_4 - q^{-2}X_2$$

And we do the same as above, now substituting X's for Y's and using the first table for type B_2 of Appendix A.

The main part of the algorithm for computing the commutation relation of E_α, E_β consists of a recursive procedure that uses relations from the rank 2 case. This idea is taken from the proof of [2], Theorem 9.3. And essentially the algorithm below is an effective version of that proof. Roughly the algorithm looks as follows.

Algorithm CommutationRelation

Input: two reduced words w_1, w_2 in $W(\Phi)$, and two simple roots α, β such that

1. $w_2 = w_1 w_2'$, where $w_2' \neq 1$,
2. $w_2' = s_{\beta_1} \cdots s_{\beta_r}$, where $\beta_1 = \alpha$,
3. $\ell(w_2 s_\beta) > \ell(w_2)$.

Output: $X = T_{w_1}(E_\alpha)T_{w_2}(E_\beta) - q^{(w_1\alpha, w_2\beta)}T_{w_2}(E_\beta)T_{w_1}(E_\alpha)$, where X is a w_2-expression.

Step 1 If $\ell(w_1) \neq 0$, then by a recursive call compute $Y = E_\alpha T_{w_2'}(E_\beta) - q^{(\alpha, w_2'\beta)}T_{w_2'}(E_\beta)E_\alpha$. To Y apply T_{w_1} and return the result.

Step 2 If $\ell(w_1) = 0$, then we set $w = w_2' = s_{\beta_1} \cdots s_{\beta_r}$ and $u = s_{\beta_1} \cdots s_{\beta_{r-1}}$. If $\ell(w) = 1$, then return 0. Otherwise we use a recursive procedure to calculate X, distinguising a few cases according to the Dynkin diagram of the roots β and β_r.

Comments: We will fill in the recursive procedure of Step 2 later. We will prove that it terminates by induction on the pair $(\ell(w_2) - \ell(w_1), \ell(w_1))$, where pairs of this form are lexicographically ordered. It is clear that the recursive call of Step 1 terminates by induction on this pair. Also if in Step 2, $\ell(w) = 1$ then the commutation relation is 0 by [6], 8.9(7).

In the recursive procedure of Step 2 we will need an algorithm Straighten. For a reduced word y it takes a y-expression that is not in normal form, and straightens it. This algorithm gets the commutation relations it needs from calls to CommutationRelation. If $\ell(y) \leq \ell(w)$ then those calls will terminate by induction. Indeed, write $y = s_{\gamma_1} \cdots s_{\gamma_m}$, then the algorithm CommutationRelation will be called with words y_1, y_2, where $y_1 = s_{\gamma_1} \cdots s_{\gamma_i}$, and $y_2 = s_{\gamma_1} \cdots s_{\gamma_j}$. But then $\ell(y_2) - \ell(y_1) \leq \ell(y) - 1 < \ell(w) = \ell(w_2) - \ell(w_1)$.

Now we describe the recursive procedure of Step 2. On several occasions we will need two more words in the Weyl group. If $\ell(us_\beta) < \ell(u)$ then we use the exchange condition to get a reduced word $v = s_{\beta_1} \cdots \hat{s}_{\beta_i} \cdots s_{\beta_{r-1}}$ such that $v \equiv us_\beta$. (Here the $\hat{}$ means that the corresponding generator is omitted.) Also, if $\ell(vs_{\beta_r}) < \ell(v)$, then we use the exchange condition to get a word $x = s_{\beta_1} \cdots \hat{s}_{\beta_j} \cdots \hat{s}_{\beta_i} \cdots s_{\beta_{r-1}}$ (or $x = s_{\beta_1} \cdots \hat{s}_{\beta_i} \cdots \hat{s}_{\beta_j} \cdots s_{\beta_{r-1}}$), such that $x \equiv vs_{\beta_r}$. Throughout the description of the algorithm the indices i and j will be fixed.

Since Φ does not contain components of type G_2, there are four possibilities for the Dynkin diagram of β, β_r. We treat each of these cases separately, and for each case we prove that the algorithm terminates with the correct output.

A) Suppose that $\langle \beta, \beta_r^\vee \rangle = 0$.

Here we return CommutationRelation$(1, u, \alpha, \beta)$.

Proof: Because $ws_\beta \equiv us_\beta s_{\beta_r}$ we have that us_β is a reduced word, so the recursive call is correct. By induction it will terminate, and return a u-expression, which is also a w-expression. Finally, $\langle \beta, \beta_r^\vee \rangle = 0$ implies that $T_{\beta_r}(E_\beta) = E_\beta$, so that $X = E_\alpha T_u(E_\beta) - q^{(\alpha, u\beta)} T_u(E_\beta) E_\alpha$. $\square$

B) Suppose that $\langle \beta, \beta_r^\vee \rangle = \langle \beta_r, \beta^\vee \rangle = -1$.

If $\ell(us_\beta) > \ell(u)$ then we take the following steps.

1. Set $\Sigma =$ CommutationRelation$(1, u, \alpha, \beta)$ and $\Omega =$ CommutationRelation$(1, u, \alpha, \beta_r)$.
2. Set $\Sigma' =$ Straighten$(q^{(\alpha, u\beta_r)} T_u(E_{\beta_r}) \Sigma - q_{\beta_r}^{-1} \Sigma T_u(E_{\beta_r}))$.
3. Set $\Omega' =$ Straighten$(\Omega T_u(E_\beta) - q_{\beta_r}^{-1} q^{(\alpha, u\beta)} T_u(E_\beta) \Omega)$.
4. Return $\Sigma' + \Omega'$.

Proof: The recursive calls of 1. terminate by induction. We have that Σ, Ω are u-expressions. Therefore the input to the straightening algorithm in 2. is a w-expression. By what we have seen above, this call will terminate, and the output will be a w-expression. The input to the same algorithm in 3. is a us_β-expression. Therefore this call will terminate. Also, because Ω has weight $\alpha + u\beta_r$ we have that $\Omega T_u(E_\beta) = q^{(\alpha + u\beta_r, u\beta)} T_u(E_\beta) \Omega + \Omega'$, where Ω' is a u-expression. Therefore the terms with $T_u(E_\beta)$ cancel, and the output at 4. will be a w-expression.

We have

$$E_\alpha T_u(E_\beta) - q^{(\alpha, u\beta)} T_u(E_\beta) E_\alpha = \Sigma, \tag{1}$$

and

$$E_\alpha T_u(E_{\beta_r}) - q^{(\alpha, u\beta_r)} T_u(E_{\beta_r}) E_\alpha = \Omega. \tag{2}$$

We also have $T_{\beta_r}(E_\beta) = E_{\beta_r} E_\beta - q_{\beta_r}^{-1} E_\beta E_{\beta_r}$, so that

$$E_\alpha T_w(E_\beta) = E_\alpha T_u(E_{\beta_r}) T_u(E_\beta) - q_{\beta_r}^{-1} E_\alpha T_u(E_\beta) T_u(E_{\beta_r}).$$

Now by using (1) and (2) this can be rewritten as

$$q^{(\alpha,w\beta)} T_w(E_\beta) E_\alpha + q^{(\alpha,u\beta_r)} T_u(E_{\beta_r}) \Sigma - q_{\beta_r}^{-1} \Sigma T_u(E_{\beta_r}) + \Omega T_u(E_\beta) - q_{\beta_r}^{-1} q^{(\alpha,u\beta)} T_u(E_\beta)\Omega.$$

This shows that the algorithm gives the correct output. □

If $\ell(us_\beta) < \ell(u)$ then we take the following steps.

1. If $i = 1$, then return $T_u(E_{\beta_r})$.
2. If $i \neq 1$ then calculate $Y = \mathsf{CommutationRelation}(1, v, \alpha, \beta_r)$, which gives Y as a v-expression. We transform it into a u-expression, and return that.

Proof: We recall that $v = s_{\beta_1} \cdots \hat{s}_{\beta_i} \cdots s_{\beta_{r-1}}$. If $i = 1$ then $s_\alpha v(\beta) = u\beta < 0$ (here we use that $\ell(zs_\gamma) = \ell(z) + 1$ if and only if $z\gamma > 0$, [5], Proposition 5.7). Therefore $v\beta$ is a positive root (because $\ell(vs_\beta) = \ell(v) + 1$) sent to a negative one by s_α, so that $v\beta = \alpha$. By [6], Proposition 8.20, $T_v(E_\beta) = E_\alpha$. Furthermore, using [6], 8.16(6), we see that $T_w(E_\beta) = T_v T_\beta T_{\beta_r}(E_\beta) = T_v(E_{\beta_r})$. Therefore $X = T_v(E_\beta) T_v(E_{\beta_r}) - q^{(v\beta, v\beta_r)} T_v(E_{\beta_r}) T_v(E_\beta) = T_v(E_\beta E_{\beta_r} - q_\beta^{-1} E_{\beta_r} E_\beta) = T_v T_\beta(E_{\beta_r}) = T_u(E_{\beta_r})$.

If $i \neq 1$ then $X = E_\alpha T_v(E_{\beta_r}) - q^{(\alpha, v\beta_r)} T_v(E_{\beta_r}) E_\alpha$ can by recursion be written as a v-expression. (Note that $\ell(vs_{\beta_r}) > \ell(v)$; otherwise $\ell(u) + 2 = \ell(us_{\beta_r} s_\beta) = \ell(vs_\beta s_{\beta_r} s_\beta) = \ell(vs_{\beta_r} s_\beta s_{\beta_r}) < \ell(v) + 2 = \ell(u) + 1$). We have that $u \equiv vs_\beta$ so we can transform this v-expression into a u-expression, using the algorithm described at the beginning of this section. □

C) Suppose that $\langle \beta, \beta_r^\vee \rangle = -2$ and $\langle \beta_r, \beta^\vee \rangle = -1$.

In this case we have $(\beta, \beta_r) = -2$, $(\beta, \beta) = 4$ and $(\beta_r, \beta_r) = 2$. If $\ell(us_\beta) > \ell(u)$ then we take the following steps.

1. Set $\Sigma = \mathsf{CommutationRelation}(1, u, \alpha, \beta)$ and $\Omega = \mathsf{CommutationRelation}(1, u, \alpha, \beta_r)$.
2. Set $\Sigma' = \mathsf{Straighten}(q^{(\alpha,u\beta_r)} T_u(E_{\beta_r})\Sigma - q^{-2} \Sigma T_u(E_{\beta_r}))$.
3. Set $\Omega' = \mathsf{Straighten}(\Omega T_u(E_\beta) - q^{-2+(\alpha,u\beta)} T_u(E_\beta)\Omega)$.
4. Set $\Pi = \Sigma' + \Omega'$.
5. Set $U = \mathsf{Straighten}(q^{(\alpha,u\beta_r)} T_u(E_{\beta_r})\Pi - \Pi T_u(E_{\beta_r}))$.
6. Set $\Omega'' = \mathsf{Straighten}(\Omega T_u(E_{\beta_r}) - q^{2+(\alpha,u\beta_r)} T_u(E_{\beta_r})\Omega)$.
7. Set $V = \mathsf{Straighten}(q^{2+(\alpha,u\beta_r)} T_u(E_{\beta_r})\Omega' - q^{-2} \Omega' T_u(E_{\beta_r}))$.
8. Set $\Omega''' = \mathsf{Straighten}(\Omega'' T_u(E_\beta) - q^{-4+(\alpha,u\beta)} T_u(E_\beta)\Omega'')$.
9. Return $\frac{U+V+\Omega'''}{q+q^{-1}}$.

Proof: The fact that all calls terminate follows easily by induction. Again Ω' is a u-expression. So in 4., Π is a w-expression. Also U is a w-expression. As Ω'' has weight $\alpha + 2u\beta_r$ we have that $\Omega'' T_u(E_\beta) = q^{-4+(\alpha,u\beta)} T_u(E_\beta)\Omega'' + \Omega'''$, where Ω''' is a u-expression. Therefore the output of the algorithm is a w-expression.

Using [6], 8.17(5) together with (1) and (2) (which also hold in this case) we infer that

$$\begin{aligned} E_\alpha T_u T_{\beta_r} T_\beta(E_{\beta_r}) =& E_\alpha T_u(E_{\beta_r}) T_u(E_\beta) - q^{-2} E_\alpha T_u(E_\beta) T_u(E_{\beta_r}) \\ =& q^{(\alpha, w s_\beta(\beta_r))} T_u T_{\beta_r} T_\beta(E_{\beta_r}) E_\alpha + q^{(\alpha,u\beta_r)} T_u(E_{\beta_r})\Sigma \\ & - q^{-2}\Sigma T_u(E_{\beta_r}) + \Omega T_u(E_\beta) - q^{-2+(\alpha,u\beta)} T_u(E_\beta)\Omega. \end{aligned}$$

Here the terms with Ω reduce to Ω', and it follows that

$$E_\alpha T_u T_{\beta_r} T_\beta(E_{\beta_r}) - q^{(\alpha, w s_\beta(\beta_r))} T_u T_{\beta_r} T_\beta(E_{\beta_r}) E_\alpha = \Pi. \tag{3}$$

Now the formula following [6], 8.17(8) implies that

$$(q+q^{-1}) E_\alpha T_u T_{\beta_r}(E_\beta) = E_\alpha T_u(E_{\beta_r}) T_u T_{\beta_r} T_\beta(E_{\beta_r}) - E_\alpha T_u T_{\beta_r} T_\beta(E_{\beta_r}) T_u(E_{\beta_r}).$$

Rewriting this, using (2) and (3) we get

$$\begin{aligned} (q+q^{-1}) E_\alpha T_u T_{\beta_r}(E_\beta) =& (q+q^{-1}) q^{(\alpha,w\beta)} T_u T_{\beta_r}(E_\beta) E_\alpha \\ & + q^{(\alpha,u\beta_r)} T_u(E_{\beta_r})\Pi - \Pi T_u(E_{\beta_r}) \\ & + \Omega T_u T_{\beta_r} T_\beta(E_{\beta_r}) - q^{(\alpha, w s_\beta(\beta_r))} T_u T_{\beta_r} T_\beta(E_{\beta_r})\Omega. \end{aligned}$$

The terms with Π straighten without problems. For the terms with Ω we note that we have

$$\Omega T_u(E_\beta) = q^{-2+(\alpha,u\beta)} T_u(E_\beta)\Omega + \Omega', \tag{4}$$

$$\Omega T_u(E_{\beta_r}) = q^{2+(\alpha,u\beta_r)} T_u(E_{\beta_r})\Omega + \Omega''. \tag{5}$$

Using (4) and (5), along with [6] 8.17(5) we calculate

$$\begin{aligned} \Omega T_u T_{\beta_r} T_\beta(E_{\beta_r}) =& \Omega T_u(E_{\beta_r}) T_u(E_\beta) - q^{-2}\Omega T_u(E_\beta) T_u(E_{\beta_r}) \\ =& q^{(\alpha, u s_\beta \beta_r)} T_u T_{\beta_r} T_\beta(E_{\beta_r})\Omega + q^{2+(\alpha,u\beta_r)} T_u(E_{\beta_r})\Omega' - q^{-2}\Omega' T_u(E_{\beta_r}) \\ & + \Omega'' T_u(E_\beta) - q^{-4+(\alpha,u\beta)} T_u(E_\beta)\Omega''. \end{aligned}$$

Again the terms with Ω' straighten without problems to a u-expression. Furthermore, the terms with Ω'' rewrite to Ω''' which is also a u-expression. So the output of the algorithm is correct. □

If $\ell(us_\beta) < \ell(u)$ then we take the following steps.

1. If $i = 1$ then return $\frac{q^2-1}{q+q^{-1}} T_u(E_{\beta_r})^2$.

2. If $i \neq 1$ and $\ell(vs_{\beta_r}) > \ell(v)$ then execute the following
   (a) Set $\Sigma = \mathsf{CommutationRelation}(1, v, \alpha, \beta_r)$, and $\Omega = \mathsf{CommutationRelation}(1, u, \alpha, \beta_r)$.
   (b) Transform Σ to a u-expression and set $U = \mathsf{Straighten}(q^{(\alpha, u\beta_r)} T_u(E_{\beta_r})\Sigma - \Sigma T_u(E_{\beta_r}))$.
   (c) Transform Ω into a vs_β-expression. Then Ω only involves v-monomials, along with products of powers of $T_v(E_\beta)$ and v-monomials. We have that

$$T_v(E_\beta)^n T_v(E_{\beta_r}) = \\ q^{-2n} T_v(E_{\beta_r}) T_v(E_\beta)^n + (\sum_{i=1}^{n} q^{2n+2-4i}) T_u(E_{\beta_r}) T_v(E_\beta)^{n-1}. \quad (6)$$

   Using this and commutation relations obtained by recursion we straighten $\Omega T_v(E_{\beta_r})$. In the straightening process we may encounter expressions of the form $mT_v(E_{\beta_r})$, where m is a v-monomial. Using relations obtained by recursion we can rewrite this as $q^{(\mu, v\beta_r)} T_v(E_{\beta_r})m + a$, where μ is the weight of m, and a is a v-expression, which we transform to a u-expression. Also we may encounter expressions of the form $T_v(E_\beta)^n \cdot m \cdot T_v(E_{\beta_r})$, where m is a v-monomial. We can rewrite this as $q^{(\mu, v\beta_r)} T_v(E_\beta)^n T_v(E_{\beta_r}) m + T_v(E_\beta)^n a$. Here $T_v(E_\beta)^n a$ is a vs_β-expression, which we can transform into a u-expression. Using (6) we can deal with the first summand. This yields an expression of the form $T_u(E_{\beta_r}) T_v(E_\beta)^{n-1} m$. After transforming $T_v(E_\beta)^{n-1} m$ to a u-expression this becomes a w-expression. So we can rewrite $\Omega T_v(E_{\beta_r})$ as $q^{(\alpha, v\beta_r)} T_v(E_{\beta_r})\Omega + \Omega'$, where Ω' is a w-expression.
   (d) Return $\frac{U+\Omega'}{q+q^{-1}}$.
3. If $i \neq 1$, $\ell(vs_{\beta_r}) < \ell(v)$, and $j = 1$ then return $T_u(E_{\beta_r})$.
4. If $i \neq 1$, $\ell(vs_{\beta_r}) < \ell(v)$, and $j \neq 1$ then set $Y = \mathsf{CommutationRelation}(1, x, \alpha, \beta)$. Transform Y into a u-expression, and return the result.

Proof: Here if $i = 1$ we get

$$X = T_v(E_\beta T_\beta T_{\beta_r}(E_\beta) - T_\beta T_{\beta_r}(E_\beta) E_\beta) \\ = \frac{q^2-1}{q+q^{-1}} T_v(T_\beta(E_{\beta_r}))^2 = \frac{q^2-1}{q+q^{-1}} T_u(E_{\beta_r})^2.$$

(In the second to last equality we used [6], Appendix 8(9).) If $i \neq 1$, then we distinguish two cases. If $\ell(vs_{\beta_r}) > \ell(v)$, then we have by recursion

$$E_\alpha T_v(E_{\beta_r}) = q^{(\alpha, v\beta_r)} T_v(E_{\beta_r}) E_\alpha + \Sigma, \quad (7)$$

$$E_\alpha T_u(E_{\beta_r}) = q^{(\alpha,u\beta_r)}T_u(E_{\beta_r})E_\alpha + \Omega, \tag{8}$$

where Σ is a v-expression, and Ω is a u-expression. Now, using [6], 8.17 we see that $(q+q^{-1})E_\alpha T_v T_\beta T_{\beta_r}(E_\beta) = E_\alpha T_u(E_{\beta_r})T_v(E_{\beta_r}) - E_\alpha T_v(E_{\beta_r})T_u(E_{\beta_r})$. Rewriting this, using (7) and (8) we get

$$(q+q^{-1})X = q^{(\alpha,u\beta_r)}T_u(E_{\beta_r})\Sigma - \Sigma T_u(E_{\beta_r}) + \Omega T_v(E_{\beta_r}) - q^{(\alpha,v\beta_r)}T_v(E_{\beta_r})\Omega.$$

And this last expression is equal to $U+\Omega'$. We note that (6) follows from [6], Appendix 8(3), (7).

If $\ell(v\beta_r) < \ell(v)$ then we note that $T_w(E_\beta) = T_x T_{\beta_r} T_\beta T_{\beta_r}(E_\beta) = T_x(E_\beta)$ (cf. [6], 8.17(9)). If $j=1$ then we have that $x\beta_r = \alpha$ and therefore $T_x(E_{\beta_r}) = E_\alpha$. This means that $X = T_x(E_{\beta_r}E_\beta - q^{-2}E_\beta E_{\beta_r}) = T_x T_{\beta_r} T_\beta(E_{\beta_r}) = T_u(E_{\beta_r})$. (In the second to last equality we used [6], 8.17(5).) If $j \neq 1$ then we use that $\ell(xs_\beta) > \ell(x)$ (indeed, otherwise $\ell(u)+2 = \ell(xs_{\beta_r}s_\beta s_{\beta_r}s_\beta) = \ell(xs_\beta s_{\beta_r}s_\beta s_{\beta_r}) < \ell(x)+3 = \ell(u)+1$). So the recursive call in 4. is justified. By induction we have that $X = E_\alpha T_x(E_\beta) - q^{(\alpha,x\beta)}T_x(E_\beta)E_\alpha$ is an x-expression, which we can transform to a u-expression. □

D) Suppose that $\langle \beta, \beta_r^\vee \rangle = -1$ and $\langle \beta_r, \beta^\vee \rangle = -2$.

In this case we have $(\beta,\beta_r) = -2$, $(\beta,\beta)=2$ and $(\beta_r,\beta_r)=4$. If $\ell(us_\beta) > \ell(u)$ then we take the following steps.

1. Set $\Sigma = \mathsf{CommutationRelation}(1,u,\alpha,\beta)$ and $\Omega = \mathsf{CommutationRelation}(1,u,\alpha,\beta_r)$.
2. Set $\Sigma' = \mathsf{Straighten}(q^{(\alpha,u\beta_r)}T_u(E_{\beta_r})\Sigma - q^{-2}\Sigma T_u(E_{\beta_r}))$.
3. Set $\Omega' = \mathsf{Straighten}(\Omega T_u(E_\beta) - q^{-2+(\alpha,u\beta)}T_u(E_\beta)\Omega)$.
4. Return $\Sigma' + \Omega'$.

Proof: Again all recursive calls terminate by induction.

Using [6], 8.17(3) we have

$$E_\alpha T_u T_{\beta_r}(E_\beta) = E_\alpha T_u(E_{\beta_r})T_u(E_\beta) - q^{-2}E_\alpha T_u(E_\beta)T_u(E_{\beta_r}).$$

Rewriting this using (1) and (2) (which also hold in this case) we get

$$\begin{aligned} E_\alpha T_u T_{\beta_r}(E_\beta) =& q^{(\alpha,us_{\beta_r}\beta)}T_u T_{\beta_r}(E_\beta)E_\alpha + q^{(\alpha,u\beta_r)}T_u(E_{\beta_r})\Sigma - q^{-2}\Sigma T_u(E_{\beta_r}) \\ &+ \Omega T_u(E_\beta) - q^{-2+(\alpha,u\beta)}T_u(E_\beta)\Omega. \end{aligned}$$

Here the terms with Σ do not pose any difficulties. Furthermore, the terms with Ω rewrite to Ω'. We have that the Σ' of 2. is a w-expression, so the output is a w-expression as well. □

If $\ell(us_\beta) < \ell(u)$ then we take the following steps.

1. If $i=1$ then return $(q+q^{-1})T_u(E_{\beta_r})$.

2. If $i \neq 1$ and $\ell(vs_{\beta_r}) > \ell(v)$ then execute the following steps.
   (a) Set $\Sigma = \mathsf{CommutationRelation}(1, v, \alpha, \beta_r)$, and $\Omega = \mathsf{CommutationRelation}(1, v, \alpha, \beta)$.
   (b) Set $\Sigma' = \mathsf{Straighten}(q^{(\alpha, v\beta)} T_v(E_\beta)\Sigma - q^{-2}\Sigma T_v(E_\beta))$.
   (c) Set $\Omega' = \mathsf{Straighten}(\Omega T_v(E_{\beta_r}) - q^{-2+(\alpha, v\beta_r)} T_v(E_{\beta_r})\Omega)$.
   (d) Set $U = \Sigma' + \Omega'$. Transform U to a u-expression and return the result.
3. If $i \neq 1$, $\ell(vs_{\beta_r}) < \ell(v)$, and $j = 1$ then set $U = T_v(E_\beta)$. Transform U to a u-expression and return the result.
4. f $i \neq 1, \ell(vs_{\beta_r}) < \ell(v)$, and $j \neq 1$ then set $U = \mathsf{CommutationRelation}(1, x, \alpha, \beta)$. Transform U into a u-expression, and return the result.

Proof: If $i = 1$ then

$$X = T_v(E_\beta T_\beta T_{\beta_r}(E_\beta) - T_\beta T_{\beta_r}(E_\beta)E_\beta) = (q + q^{-1})T_v T_\beta(E_{\beta_r}) = (q + q^{-1})T_u(E_{\beta_r}).$$

(In the second to last equality we used [6], Appendix 8(6).) If $i \neq 1$ then we first suppose that $\ell(vs_{\beta_r}) > \ell(v)$. After Step (a) we have

$$E_\alpha T_v(E_{\beta_r}) = q^{(\alpha, v\beta_r)} T_v(E_{\beta_r})E_\alpha + \Sigma, \tag{9}$$

$$E_\alpha T_v(E_\beta) = q^{(\alpha, v\beta)} T_v(E_\beta)E_\alpha + \Omega, \tag{10}$$

where Ω and Σ are v-expressions. Also by [6], 8.17(5) we have

$$E_\alpha T_v T_\beta T_{\beta_r}(E_\beta) = E_\alpha T_v(E_\beta)T_v(E_{\beta_r}) - q^{-2} E_\alpha T_v(E_{\beta_r})T_v(E_\beta).$$

Rewriting this using (9) and (10) we get

$$X = \Omega T_v(E_{\beta_r}) - q^{-2+(\alpha, v\beta_r)} T_v(E_{\beta_r})\Omega + q^{(\alpha, v\beta)} T_v(E_\beta)\Sigma - q^{-2}\Sigma T_v(E_\beta).$$

We rewrite the terms with Σ, which results in a vs_β-expression. As Ω has weight $\alpha + v(\beta)$ we have that $\Omega T_v(E_{\beta_r}) = q^{-2+(\alpha, v\beta_r)} T_v(E_{\beta_r})\Omega + \Omega'$, where Ω' is a v-expression. This means that the terms with Ω rewrite to Ω'. So the U in (d) is a vs_β-expression, which we transform into a u-expression. If $\ell(vs_{\beta_r}) < \ell(v)$, then we have $T_w(E_\beta) = T_x T_{\beta_r} T_\beta T_{\beta_r}(E_\beta) = T_x(E_\beta)$ ([6], 8.17(6)). If $j = 1$ then, using [6], Appendix 8(7), we get $X = T_x(E_{\beta_r} E_\beta - q^{-2} E_\beta E_{\beta_r}) = T_x T_{\beta_r}(E_\beta) = T_v(E_\beta)$, which we transform to a u-expression. If $j \neq 1$, then we have the same as in case C). □

Now we have an algorithm for finding the skew-commutator of E_α and E_β for all positive roots $\alpha, \beta \in \Phi$. In order to find the skew-commutator of F_α, F_β we use the automorphism ω of $U_q(\mathfrak{g})$ given by $\omega(F_\alpha) = E_\alpha$, $\omega(E_\alpha) = F_\alpha$ and $\omega(K_\alpha) = K_\alpha^{-1}$ (cf. [6], 4.6). By [6], 8.18(5) we have that

$$\omega(T_w(E_\alpha)) = \left(\prod_{\gamma \in \Delta} (-q_\gamma)^{-m_\gamma} \right) T_w(F_\alpha),$$

where $w\alpha - \alpha = \sum_{\gamma \in \Delta} m_\gamma \gamma$. Using this, and the commutation relation of E_α, E_β we easily get the commutation relation of F_α, F_β.

We turn our attention towards finding the skew-commutator of E_α and F_β, where α, β are arbitrary positive roots. The following lemma is useful. Here the weight of an element of U^- is defined in the same way as the weight of an element of U^+.

Lemma 1 *Let $\alpha, \beta \in \Phi$ be two positive roots, then $E_\alpha F_\beta = F_\beta E_\alpha + \Sigma$, where Σ is a linear combination of monomials of the form FKE, where $K \in U^0$, and the weight of $F \in U^-$ is of lower level than β and the weight of $E \in U^+$ is of lower level than α.*

Proof: This can be seen by writing E_α and F_β as expressions in the generators (i.e., E_γ, F_γ for $\gamma \in \Delta$). Then we straighten the product $E_\alpha F_\beta$ using the defining relations of $U_q(\mathfrak{g})$. Each time we use a relation of the form $E_\gamma F_\eta = F_\eta E_\gamma + \delta_{\gamma,\eta}(K_\gamma - K_\gamma^{-1})/(q_\gamma - q_\gamma^{-1})$, we replace one monomial by at most two. However, in the one arising from the second term, the total level of the E's and the total level of the F's has dropped. Hence the result is $F_\beta E_\alpha$ plus a linear combination of monomials FKE, where F is an expression in the generators of lower level than β, and E is an expression in the generators of lower level than α. We finally rewrite these as linear combinations of monomials in the PBW-generators, which does not change their weight. □

In principle the proof of Lemma 1 gives an algorithm for computing the commutation relation of E_α and F_β. However, because the expression of E_α, F_β in the generators can be rather large, a more efficient algorithm works as follows.

First we suppose that $\beta \in \Delta$ is a simple root. If α is also a simple root, then we know the commutation relation from the defining relations of $U_q(\mathfrak{g})$. If α is not a simple root, then, because E_α lies in U^+, there is at least one commutation relation of the form $E_\gamma E_\delta - q^{(\gamma,\delta)} E_\delta E_\gamma = \lambda E_\alpha + \Pi$, where Π involves E_ϵ for ϵ of lower level than α. This allows us to write E_α as a linear combination e of monomials involving only E_η where η is of lower level than α. By recursion we already know the commutation relations of F_β with those elements. Finally we straighten eF_β. Lemma 1 ensures that all commutation relations needed for this are already known.

If β is not a simple root, then we do the same as above, this time writing F_β as a linear combination f, involving only F_γ, where γ is of lower level than β. We straighten $E_\alpha f$ using the relations that we already know. Again Lemma 1 ensures that all commutation relations we need are already known.

We conclude that we have algorithms for determining a multiplication table of a PBW-type basis of $U_q(\mathfrak{g})$.

Remark. It is possible to show that the output X of the algorithm CommutationRelation only involves monomials in $T_{w_1} T_{\beta_1} \cdots T_{\beta_{r-1}}(E_{\beta_r}), \ldots, T_{w_1} T_{\beta_1}(E_{\beta_2})$, cf. [2], Theorem 9.1. Roughly the argument given there runs as follows. First we note that we only have to prove this for the case where $\ell(w_1) = 0$. For $1 \leq i \leq r$ we set $\gamma_i = s_{\beta_1} \cdots s_{\beta_{r-i}}(\beta_{r-i+1})$. (We use the same notation as in the statement of the algorithm.) Suppose that in X there occurs a monomial of the form $E_{\gamma_1}^{m_1} \cdots E_{\gamma_{r-1}}^{m_{r-1}} E_\alpha^m$, where $m > 0$ (we recall that $\gamma_r = \beta_1 = \alpha$). Then, because the weight of this monomial must be $\alpha + w\beta$ we have that $w\beta = (m-1)\alpha + \sum_{i=1}^{r-1} m_i\gamma_i$. Now we have $w^{-1}\alpha < 0$ as $\ell(w^{-1}s_\alpha) < \ell(w^{-1})$. Also we have $w^{-1}\gamma_i < 0$. Indeed, write $\gamma_i = w_i(\beta_{r-i+1})$, then $s_{\gamma_i} = w_i s_{\beta_i} w_i^{-1}$, and this implies that $\ell(w^{-1}s_{\gamma_i}) < \ell(w^{-1})$. Using the above, we infer that $\beta < 0$, which is a contradiction.

3. Highest-weight modules and R-matrices

In this section we denote the weight lattice of Φ by P. Furthermore, the fundamental weights are denoted by $\lambda_1, \ldots, \lambda_l$, so that $P = \mathbb{Z}\lambda_1 + \cdots + \mathbb{Z}\lambda_l$.

Let V be an irreducible finite-dimensional module over $U_q(\mathfrak{g})$. Then there is a dominant weight $\lambda \in P$ such that V is isomorphic to the finite-dimensional highest weight module $L(\lambda)$ with highest weight λ (cf. [6], Theorem 5.10). So the problem of constructing the irreducible modules over $U_q(\mathfrak{g})$ boils down to constructing the finite-dimensional highest-weight modules. These modules are constructed in the following way. Let $\lambda = r_1\lambda_1 + \cdots + r_l\lambda_l$ be a dominant weight. We let $J(\lambda)$ be the left ideal of $U_q(\mathfrak{g})$ generated by E_α for $\alpha \in \Phi$, along with $K_\alpha^{\pm 1} - q^{\pm(\lambda,\alpha)}$ for $\alpha \in \Delta$. Then we construct the Verma module $M(\lambda) = U_q(\mathfrak{g})/J(\lambda)$. This is a $U_q(\mathfrak{g})$-module. But also $M(\lambda) \cong U^-$ (as vector spaces), where the isomorphism respects the left action of U^-. Let $I(\lambda)$ be the left ideal of U^- generated by $F_{\alpha_1}^{r_1+1}, \ldots, F_{\alpha_l}^{r_l+1}$. This left ideal is also a $U_q(\mathfrak{g})$-submodule and by [6], 5.9, Theorem 5.15 we have that $L(\lambda) = U^-/I(\lambda)$ is the irreducible highest-weight module over $U_q(\mathfrak{g})$ with highest weight λ. In order to construct a basis of the quotient $U^-/I(\lambda)$, we calculate a Gröbner basis of $I(\lambda)$.

We consider the problem of calculating Gröbner bases of ideals in U^-. As before we fix a reduced expression $s_{\alpha_{i_1}} \cdots s_{\alpha_{i_t}}$ of the longest element in the Weyl group. And for $1 \leq k \leq t$ we set $\beta_k = s_{\alpha_{i_1}} \cdots s_{\alpha_{i_{t-k}}}(\alpha_{i_{t-k+1}})$, and $F_k = F_{\beta_k} = T_{\alpha_{i_1}} \cdots T_{\alpha_{i_{t-k}}}(F_{\alpha_{i_{t-k+1}}})$. Then U^- is spanned by the monomials $F_1^{n_1} \cdots F_t^{n_t}$. Furthermore, from the properties of the output of the algorithm CommutationRelation we see that for $j > i$

$$F_jF_i = q^{(\beta_i,\beta_j)}F_iF_j + \Sigma_{ij},$$

where Σ_{ij} involves only $F_{i+1}, \ldots, F_{j-1}$. Now let $<$ be the ordering of the set of monomials defined in the following way. Let $a_1 = F_1^{m_1} \cdots F_t^{m_t}$ and $a_2 = F_1^{n_1} \cdots F_t^{n_t}$. Then $a_1 < a_2$ if and only if the last non-zero coefficient of $(m_1 - n_1, \ldots, m_t - n_t)$ is negative. Then for any monomial a appearing in Σ_{ij}, we have $a < F_iF_j$. This shows that U^- is a solvable polynomial ring in the sense of [7]. This means that there are algorithms to compute Gröbner bases of left and twosided ideals of U^- (cf. [7]).

Now we can calculate a Gröbner basis of $I(\lambda)$, and construct a basis of $L(\lambda)$. Furthermore, the weights of $L(\lambda)$ and their multiplicities are equal to the weights and multiplicities of the irreducible highest-weight module with highest weight λ over $\mathfrak{g}$ (cf. [6], Theorem 5.15). This implies immediately that the algorithm for finding a Gröbner basis described in [4] generalizes to this setting.

Let V be a highest-weight module over $U_q(\Phi)$. We now have all ingredients for constructing an R-matrix corresponding to V. For this we follow the construction in Chapter 7 of [6]. By [6], Proposition 8.29 we see that a PBW-type basis immediately yields dual bases as in [6], §7.1. Using this, the operator Θ from [6], §7.2 can easily be constructed. For the construction of the map f from [6], §7.3, we need a set of representatives $\nu_1, \dots, \nu_r$ of $P/\mathbb{Z}\Phi$, along with an algorithm that for given $\mu \in P$ finds a ν_i and a $\gamma \in \mathbb{Z}\Phi$ such that $\mu = \nu_i + \gamma$. This can be done by calculating the Smith normal form of the Cartan matrix of Φ, and using Proposition 3.3 of Chapter 8 of [11]. Now by composing Θ and the map $\tilde{f}$ from [6], §7.3, we find an R-matrix corresponding to V.

4. Examples and practical experiences

Let Φ be the root system of type B_3. There are three simple roots, which we denote by α, β, γ. We have that $(\alpha,\alpha) = (\beta,\beta) = 4$, $(\gamma,\gamma) = 2$, $(\alpha,\beta) = -2$, $(\alpha,\gamma) = 0$ and $(\beta,\gamma) = -2$. We compute the commutation relation of E_β and $T_\beta T_\alpha T_\gamma T_\beta T_\alpha T_\gamma(E_\beta)$. We have $\beta_r = \gamma$ and the β of the algorithm is also β here. So we are in case C). Also $u = s_\beta s_\alpha s_\gamma s_\beta s_\alpha$, and $\ell(us_\beta) = 4$. Furthermore $v = s_\beta s_\gamma s_\beta s_\alpha$ is a reduced expression such that $v \equiv us_\beta$ in the Weyl group. Therefore we have that $i = 2$. Also $\ell(vs_\gamma) = 5 > \ell(v)$. Hence we execute Step 2 of the second piece of algorithm of case C). By inductive calls (which we leave to the reader) we have $\Sigma = T_\beta(E_\gamma)$ and $\Omega = (q^2 - q^{-2})T_\beta T_\alpha(E_\gamma)T_\beta(E_\alpha)$. In this case it is straightforward to transform Σ into a u-expression, as $\Sigma = T_\beta T_\alpha(E_\gamma)$. Then the straightening operation of Step b. gives $U = -(q+q^{-1})T_\beta T_\alpha T_\gamma(E_\beta)$. We perform Step c. After transforming Ω to a vs_β-expression we get $\Omega = (q^2 - q^{-2})T_\beta T_\gamma T_\beta T_\alpha(E_\beta)T_\beta(E_\gamma) = (q^2-q^{-2})T_v(E_\beta)T_\beta(E_\gamma)$. We straighten $\Omega T_v(E_\gamma)$. Using a commutation relation that we get from a recursive call we get

$$T_v(E_\beta)T_\beta(E_\gamma)T_v(E_\gamma) = T_v(E_\beta)T_v(E_\gamma)T_\beta(E_\gamma) + (q+q^{-1})T_v(E_\beta)T_\beta T_\gamma(E_\beta).$$

Transforming the second term into a u-expression we get that it is equal to

$$(q^{-3}+q^{-1})T_\beta T_\alpha T_\gamma T_\beta(E_\alpha)T_\beta(E_\alpha) + (q+q^{-1})T_\beta T_\alpha T_\gamma(E_\beta).$$

By relation (6) we have that the first term is equal to

$$q^{-2}T_v(E_\gamma)T_v(E_\gamma)T_v(E_\beta) + T_u(E_\gamma)T_\beta(E_\gamma).$$

Using $T_\beta(E_\gamma) = T_\beta T_\alpha(E_\gamma)$ and summing we get that

$$\Omega' = (q^2-q^{-2})(T_u(E_\gamma)T_\beta T_\alpha(E_\gamma) + (q^{-3}+q^{-1})T_\beta T_\alpha T_\gamma T_\beta(E_\alpha)T_\beta(E_\alpha) + (q+q^{-1})T_\beta T_\alpha T_\gamma(E_\beta)).$$

Then in Step d. we return

$$(q^2 - 1 - q^{-2})T_\beta T_\alpha T_\gamma(E_\beta) + (q - q^{-1})T_\beta T_\alpha T_\gamma T_\beta T_\alpha(E_\gamma)T_\beta T_\alpha(E_\gamma) \\ + (1 - q^{-4})T_\beta T_\alpha T_\gamma T_\beta(E_\alpha)T_\beta(E_\alpha),$$

which is equal to $E_\beta \cdot T_\beta T_\alpha T_\gamma T_\beta T_\alpha T_\gamma(E_\beta) - q^{-2}T_\beta T_\alpha T_\gamma T_\beta T_\alpha T_\gamma(E_\beta) \cdot E_\beta$.

We have implemented the algorithms described in this paper in the computer algebra system GAP4 ([3]). By E we denote the algorithm for finding the commutation relations of the E_α, E_β. By FE we denote the algorithm for finding the commutation relations of the E_α, F_β. In Table 1 we display some computation times* of these algorithms for several root systems.

Table 1: Running times (in seconds) of the algorithms E and FE, for several root systems.

type	A_6	B_6	C_6	D_6	D_7	D_8	E_6	E_7	E_8	F_4
E	8	69	66	34	142	519	58	704	11129	12
FE	5	41	39	19	60	175	53	665	∞	26

We remark the following

- The algorithms are efficient enough to be able to deal with most root systems of rank ≤ 8.
- However, for E_8 the algorithm FE did not terminate. Some elements computed by the algorithm turned out to be so large that the program was not able to do the computations in 128M of RAM.
- From the computation times for $D_{6,7,8}$, and by comparing D_8 and E_8 we see that the computation time increases rapidly if the number of roots increases.

In Table 2 we list some computation times for the algorithm for constructing a highest-weight module. Table 3 contains some computation times for the algorithm for computing an R-matrix. On these tables we remark the following

- The algorithm for constructing a highest-weight module is efficient enough to be able to construct rather high-dimensional modules.
- For F_4 we constructed a 273-dimensional module, and for E_6 a 351-dimensional module. However, the first construction ran markedly longer than the second. This is caused by the fact that the multiplication of PBW elements is much more time consuming in the F_4 case, as the multiplication table is denser (i.e., contains elements with, on the average, more monomials than the table in the E_6 case). In the F_4 case a multiplication took on the average 0.203 seconds, while this was 0.036 seconds in the E_6 case.

*The computations were done on a Pentium 500

- The construction of R-matrices is only feasible for rather low-dimensional modules. This is caused by the fact that the endomorphism Θ is the sum of all Θ_μ, where μ runs over all weights that are differences of weights of $L(\lambda)$. For $L(\lambda)$ of higher dimension there are many more such μ. Furthermore, the dimension of the spaces U_μ^+ increases rapidly. Therefore the calculation of Θ becomes rather cumbersome.

Table 2: Running times (in seconds) of the algorithm for constructing a highest-weight module, for two root systems, and several highest-weights.

type	λ	$\dim L(\lambda)$	time
F_4	(0,0,0,1)	26	10
F_4	(1,0,0,0)	52	15
F_4	(0,0,1,0)	273	1891
E_6	(1,0,0,0,0,0)	27	3
E_6	(0,0,1,0,0,0)	351	160
E_6	(0,0,0,1,0,0)	2925	13725

Table 3: Running times (in seconds) of the algorithm for computing an R-matrix, for several root systems, and several highest-weight modules.

type	λ	$\dim L(\lambda)$	time
G_2	(0,1)	14	141
G_2	(2,0)	27	2079
C_3	(0,1,0)	14	113
C_3	(2,0,0)	21	628
D_4	(1,0,0,0)	8	15
D_4	(0,1,0,0)	28	2227

Appendix A

Here we list multiplication tables for the rank 2 cases. In all cases α, β will be the two simple roots.

We start with A_2, where $\langle \alpha, \beta^\vee \rangle = \langle \beta, \alpha^\vee \rangle = -1$. Here we use the reduced expression $s_\alpha s_\beta s_\alpha$ for the longest element of the Weyl group. (By symmetry the other reduced expression gives exactly the same table.) We have the elements $T_\alpha T_\beta(E_\alpha) = E_\beta$, $E_{\alpha+\beta} = T_\alpha(E_\beta)$ and E_α. They satisfy the following relations (cf. [6], 8.23):

$$\begin{aligned} E_{\alpha+\beta} E_\beta &= q_\alpha E_\beta E_{\alpha+\beta} \\ E_\alpha E_\beta &= q_\alpha^{-1} E_\beta E_\alpha + E_{\alpha+\beta} \\ E_\alpha E_{\alpha+\beta} &= q_\alpha E_{\alpha+\beta} E_\alpha. \end{aligned}$$

In the case of B_2 we assume that $\langle \alpha, \beta^\vee \rangle = -1$ and $\langle \beta, \alpha^\vee \rangle = -2$, which means that $(\alpha,\beta) = -2$, $(\alpha,\alpha) = 2$ and $(\beta,\beta) = 4$. In this case we have two reduced expressions for the longest element in the Weyl group, which lead to different multiplication tables. If we use $s_\alpha s_\beta s_\alpha s_\beta$, then we set $T_\alpha T_\beta T_\alpha(E_\beta) = E_\beta$, $T_\alpha T_\beta(E_\alpha) = E_{\alpha+\beta}$, $T_\alpha(E_\beta) = E_{2\alpha+\beta}$ and E_α. They satisfy the following relations (cf. [6], Appendix 8):

$$\begin{aligned}
E_{\alpha+\beta}E_\beta &= q^2 E_\beta E_{\alpha+\beta}\\
E_{2\alpha+\beta}E_\beta &= E_\beta E_{2\alpha+\beta} + \frac{q^2-1}{q+q^{-1}}E_{\alpha+\beta}^2\\
E_\alpha E_\beta &= q^{-2}E_\beta E_\alpha + E_{\alpha+\beta}\\
E_{2\alpha+\beta}E_{\alpha+\beta} &= q^2 E_{\alpha+\beta}E_{2\alpha+\beta}\\
E_\alpha E_{\alpha+\beta} &= E_{\alpha+\beta}E_\alpha + (q+q^{-1})E_{2\alpha+\beta}\\
E_\alpha E_{2\alpha+\beta} &= q^2 E_{2\alpha+\beta}E_\alpha.
\end{aligned}$$

Taking $s_\beta s_\alpha s_\beta s_\alpha$ as reduced expression, we set $T_\beta T_\alpha T_\beta(E_\alpha) = E_\alpha$, $T_\beta T_\alpha(E_\beta) = E_{2\alpha+\beta}$, $T_\beta(E_\alpha) = E_{\alpha+\beta}$ and E_β. In this case we have the following relations:

$$\begin{aligned}
E_{2\alpha+\beta}E_\alpha &= q^2 E_\alpha E_{2\alpha+\beta}\\
E_{\alpha+\beta}E_\alpha &= E_\alpha E_{\alpha+\beta} + (q+q^{-1})E_{2\alpha+\beta}\\
E_\beta E_\alpha &= q^{-2}E_\alpha E_\beta + E_{\alpha+\beta}\\
E_{\alpha+\beta}E_{2\alpha+\beta} &= q^2 E_{2\alpha+\beta}E_{\alpha+\beta}\\
E_\beta E_{2\alpha+\beta} &= E_{2\alpha+\beta}E_\beta + \frac{q^2-1}{q+q^{-1}}E_{\alpha+\beta}^2\\
E_\beta E_{\alpha+\beta} &= q^2 E_{\alpha+\beta}E_\beta.
\end{aligned}$$

Finally we deal with the case of G_2. In this case we assume that $\langle\alpha,\beta^\vee\rangle = -1$, $\langle\beta,\alpha^\vee\rangle = -3$. This means that $(\alpha,\beta) = -3$, $(\alpha,\alpha) = 2$ and $(\beta,\beta) = 6$. The reduced form of the longest element in the Weyl group that we use is $w_0 = s_\alpha s_\beta s_\alpha s_\beta s_\alpha s_\beta$. This leads to six PBW-generators in U^+, namely E_β, $E_{\alpha+\beta}$, $E_{3\alpha+2\beta}$, $E_{2\alpha+\beta}$, $E_{3\alpha+\beta}$, E_α. (Here $E_{\alpha+\beta} = T_\alpha T_\beta T_\alpha T_\beta(E_\alpha)$, etc.) We computed the multiplication table of the elements E_γ by computing a Gröbner basis of the ideal (of the free algebra generated by E_α, E_β) generated by the elements (R_E). Below we list the result:

$$\begin{aligned}
E_{\alpha+\beta}E_\beta &= q^3 E_\beta E_{\alpha+\beta}\\
E_{3\alpha+2\beta}E_\beta &= q^3 E_\beta E_{3\alpha+2\beta} + (q^6-q^4-q^2+1)E_{\alpha+\beta}^{(3)}\\
E_{2\alpha+\beta}E_\beta &= E_\beta E_{2\alpha+\beta} + (q^3-q^{-1})E_{\alpha+\beta}^{(2)}\\
E_{3\alpha+\beta}E_\beta &= q^{-3}E_\beta E_{3\alpha+\beta} + (q^3-q^{-1}-q)E_{3\alpha+2\beta} + (q^2-1)E_{\alpha+\beta}E_{2\alpha+\beta}\\
E_\alpha E_\beta &= q^{-3}E_\beta E_\alpha + E_{\alpha+\beta}\\
E_{3\alpha+2\beta}E_{\alpha+\beta} &= q^3 E_{\alpha+\beta}E_{3\alpha+2\beta}
\end{aligned}$$

$$\begin{aligned}
E_{2\alpha+\beta}E_{\alpha+\beta} &= qE_{\alpha+\beta}E_{2\alpha+2\beta} + (q^2+1+q^{-2})E_{3\alpha+2\beta}\\
E_{3\alpha+\beta}E_{\alpha+\beta} &= E_{\alpha+\beta}E_{3\alpha+\beta} + (q^3-q^{-1})E^{(2)}_{2\alpha+\beta}\\
E_{\alpha}E_{\alpha+\beta} &= q^{-1}E_{\alpha+\beta}E_{\alpha} + (q+q^{-1})E_{2\alpha+\beta}\\
E_{2\alpha+\beta}E_{3\alpha+2\beta} &= q^3E_{3\alpha+2\beta}E_{2\alpha+\beta}\\
E_{3\alpha+\beta}E_{3\alpha+2\beta} &= q^3E_{3\alpha+2\beta}E_{3\alpha+\beta} + (q^6-q^4-q^2+1)E^{(3)}_{2\alpha+\beta}\\
E_{\alpha}E_{3\alpha+2\beta} &= E_{3\alpha+2\beta}E_{\alpha} + (q^3-q^{-1})E^{(2)}_{2\alpha+\beta}\\
E_{3\alpha+\beta}E_{2\alpha+\beta} &= q^3E_{2\alpha+\beta}E_{3\alpha+\beta}\\
E_{\alpha}E_{2\alpha+\beta} &= qE_{2\alpha+\beta}E_{\alpha} + (q^2+1+q^{-2})E_{3\alpha+\beta}\\
E_{\alpha}E_{3\alpha+\beta} &= q^3E_{3\alpha+\beta}E_{\alpha}.
\end{aligned}$$

References

[1] K. S. Brown. *Buildings.* Springer-Verlag, New York, 1989.

[2] C. De Concini and C. Procesi. Quantum groups. In *D-modules, representation theory, and quantum groups (Venice, 1992)*, pages 31–140. Springer, Berlin, 1993.

[3] The GAP Group, Aachen, St Andrews. *GAP – Groups, Algorithms, and Programming, Version 4.2*, 2000. (`http://www-gap.dcs.st-and.ac.uk/~gap`).

[4] W. A. de Graaf. Constructing representations of split semisimple Lie algebras. *J. Pure and Appl. Algebra.* to appear.

[5] J. E. Humphreys. *Reflection groups and Coxeter groups.* Cambridge University Press, Cambridge, 1990.

[6] J. C. Jantzen. *Lectures on Quantum Groups*, volume 6 of *Graduate Studies in Mathematics.* American Mathematical Society, 1996.

[7] A. Kandri-Rody and V. Weispfenning. Noncommutative Gröbner bases in algebras of solvable type. *J. Symbolic Comput.*, 9(1):1–26, 1990.

[8] G. Lusztig. *Introduction to quantum groups.* Birkhäuser Boston Inc., Boston, MA, 1993.

[9] T. Mora. An introduction to commutative and noncommutative Gröbner bases. *Theoretical Computer Science*, 134(1):131–173, 1994.

[10] C. M. Ringel. PBW-bases of quantum groups. *J. Reine Angew. Math.*, 470:51–88, 1996.

[11] C. C. Sims. *Computation with Finitely Presented Groups.* Cambridge University Press, Cambridge, 1994.