Using computer algebra system Singular::Plural for computations in noncommutative polynomial algebras

V. Levandovskyy
Fachbereich Mathematik, Universität Kaiserslautern, Germany.
levandov@mathematik.uni-kl.de

H. Schönemann
Fachbereich Mathematik, Universität Kaiserslautern, Germany.
hannes@mathematik.uni-kl.de

http://www.singular.uni-kl.de/plural

1 Introduction

The aim of this poster is to demonstrate different aspects (from theoretical background to practical usefulness) of the algorithms, implemented in the computer algebra system SINGULAR::PLURAL (or just PLURAL for short). You can regard it as a one-page introduction to the system. More detailed information and packages for download you can find at our homepage `http://www.singular.uni-kl.de/plural`.

2 Computational Objects

2.1 Algebras

Let $\mathbb{K}$ be a field. Consider an algebra $A = \mathbb{K}\langle x_1, \dots, x_n \mid x_j x_i = c_{ij} x_i x_j + d_{ij} \ \forall\, i < j \,\rangle$ with $d_{ij} \in A, c_{ij} \in \mathbb{K}^*$. It is called **a** G**–algebra** (in n variables) if the following conditions hold:

1) There exists a monomial well–ordering $<_A$ such that $\forall\ i < j\ \ \mathrm{lm}(d_{ij}) <_A x_i x_j$,

2) *Nondegeneracy conditions* are fulfilled, that is $\forall\ 1 \le i < j < k \le n$

$$c_{ik}c_{jk} \cdot d_{ij}x_k - x_k d_{ij} + c_{jk} \cdot x_j d_{ik} - c_{ij} \cdot d_{ik}x_j + d_{jk}x_i - c_{ij}c_{ik} \cdot x_i d_{jk} = 0.$$

A GR**–algebra** is a factor of G–algebra in n variables by a proper two–sided ideal.

Commutative rings in SINGULAR provide us with the information on the ground field $\mathbb{K}$, variables $(x_1, \dots, x_n)$ and the monomial ordering $<$. As a set of data, G–algebra is represented in PLURAL as an extension of the data type `ring` by the two strictly upper-triangular $n \times n$ matrices $C = (c_{ij})$ and $D = (d_{ij})$ with entries as in the definition.

Theorem 1. *[3] Let A be a G–algebra in n variables. Then*

- *A has a Poincaré–Birkhoff–Witt (PBW) basis basis $\{x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} \mid \alpha_i \in \mathbb{N} \cup \{0\}\}$,*
- *A is noetherian,*
- *A is an integral domain,*
- *$\mathrm{gl.dim}\, A \le n$.*

Example 2. *GR–algebras*

- *commutative and quasi–commutative polynomial algebras*
- *algebras of solvable type, PBW algebras, some iterated Ore extensions*
- *universal enveloping algebras of finite dimensional Lie algebras*
- *positive (resp. negative) parts of quantized enveloping algebras*
- *many quantum algebras and nonstandard quantum deformations*
- *Weyl algebras and most of various flavors of their quantized versions*
- *Witten's deformation of $U(\mathfrak{sl}_2)$, Smith algebras and conformal $\mathfrak{sl}_2$–algebras*
- *Clifford algebras, exterior algebras; some diffusion algebras and many more*

Using the library `"center.lib"` (V.Levandovskyy, O.Motsak, 2003, to appear) we are able to compute all the central elements of G–algebras up to given degree. The next version of the library will feature an algorithm for computing a minimal generating set of the center up to given degree. The functionality of the library allowed us to compute the central element of the degree 6 in the algebra $U(\mathfrak{g}_2)$ (as far as we know, it has never been computed before). We give more details in the section 3.1.

2.2 Ideals and modules

The data type `ideal` corresponds to a left ideal in the GR–algebra. Some procedures like `twostd` interpret an ideal in the argument as a set of two–sided generators.

The data type `module` corresponds to the left submodule of a free module of finite rank over GR–algebra.

With the help of PLURAL we are able to compute Gröbner bases of modules with respect to a wide variety of monomial orderings (command `std`), what is especially useful for performing the elimination. The hardest problems, known to us, are elimination problems. We have the built-in command `eliminate`, which chooses an elimination ordering heuristically. It is also possible to construct an elimination ordering "by hands" while defining an algebra and use the command `std` for solving the sophisticated elimination problems.

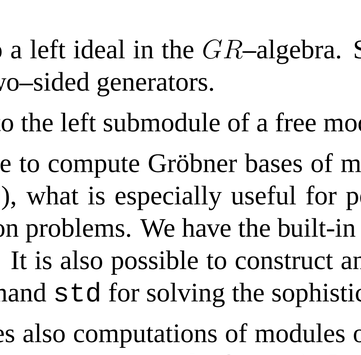
The functionality of PLURAL includes also computations of modules of syzygies (command `syz`) and several kinds of free resolutions (commands `nres`, `mres` and `minres`). In addition to the command `eliminate`, there will be more functions like`intersect` for basic operations with modules.

3 Applications

3.1 Magic numbers in $U(\mathfrak{g}_2)$

The universal enveloping algebra of the smallest exceptional Lie algebra $U(\mathfrak{g}_2)$ is generated by the 14 variables $x_1, x_2, \dots, x_6, y_1, y_2, \dots, y_6, h_\alpha, h_\beta$ (we denote $x_1 := x_\alpha, x_2 := x_\beta, y_1 := x_{-\alpha}, y_2 := x_{-\beta}$) subject to numerous relations which we do not list here.

Figure 1: The root system of $\mathfrak{g}_2$, generated by the simple roots α and β



The center $Z(U(\mathfrak{g}_2)) = \mathbb{K}[Z_2, Z_6]$, generated by the polynomials Z_2 and Z_6 of degree 2 and 6 respectively. The first one is easy to write down: $Z_2 = x_1y_1 + 3x_2y_2 + x_3y_3 + x_4y_4 + 3x_5y_5 + 3x_6y_6 + h_\alpha^2 + 3h_\alpha h_\beta + 3h_\beta^2 - 5h_\alpha - 9h_\beta$, but the second one, $Z_6 = 4x_1x_2^2y_1y_2^2 + \dots - 240h_\alpha - 480h_\beta$, consists of 754(!) monomials. We publish its explicit form on our homepage as well as other big objects appearing in our examples.

Using the standard PLURAL library `lieA.lib` you can set up $U(\mathfrak{g}_2)$ over $\mathbb{Q}$ in a fast and simple way:

```
LIB "lieA.lib";
def UG2=g2();
setring UG2;
```

```
==>
//   characteristic : 0
//   number of vars : 14
//   block 1 : ordering dp
//           : names    x(1) x(2) x(3) x(4) x(5) x(6) y(1) y(2) y(3) y(4) y(5) y(6) Ha Hb
//   block 2 : ordering C
//   noncommutative relations:
//    x(2)x(1)=x(1)*x(2)-x(3)
//    ......................
//    Hby(6)=y(6)*Hb-y(6)
```

Consider the two–sided ideal I, generated by the 3^{rd} power of the image of the shortest positive root α of the Lie algebra $\mathfrak{g}_2$ and compute the left Gröbner basis of I, using a proprietary algorithm for it.

```
ideal I=x(1)^3;
I=system("twostd",I);
size(I);
```

```
==> 106
```

As we see, it consists of 106 elements. Now we will check whether the module $M := U(\mathfrak{g}_2)/I$ is finite–dimensional and if it so, we compute its $\mathbb{K}$-base.

```
vdim(I);
```

```
==> 50  // so, M is finite--dimensional module
```

```
kbase(I);
```

==> $1, x_1, x_2, x_3, x_4, x_5, x_6, y_1, y_2, y_3, y_4, y_5, y_6, h_\alpha, h_\beta, x_1x_6, x_1y_6, x_1h_\alpha, x_1h_\beta, x_2h_\alpha, x_3h_\alpha, x_3h_\beta, x_3y_5, x_4h_\alpha, x_4h_\beta, x_4y_4,$
$x_5h_\beta, x_5y_3, x_6h_\beta, x_6y_1, y_1h_\alpha, y_1h_\beta, y_1y_6, y_2h_\alpha, y_3h_\alpha, y_3h_\beta, y_4h_\alpha, y_4h_\beta, y_5h_\beta, y_6h_\beta, h_\alpha h_\beta, h_\alpha^2, h_\beta^2,$
$x_1h_\alpha h_\beta, x_3h_\alpha h_\beta, x_4h_\beta^2, y_1h_\alpha h_\beta, y_3h_\alpha h_\beta, y_4h_\beta^2, h_\alpha h_\beta^2$ (We show the output in LaTeX form for better readability)

It is time to compute the first syzygy module $\mathrm{syz}(I)$ of I and look on its size

```
module S=syz(I);
size(S);   // size() returns the number of generators
```

```
==> 3244
```

We would like to check what kind of connection exists between central elements and the ideal I. Let us compute the normal forms of Z_2 and Z_6 with respect to I.

```
NF(Z2,I);
```

==> $2x_4y_4 + 2h_\alpha^2 + 6h_\alpha h_\beta + 7h_\beta^2 - 2h_\alpha - 3h_\beta$

```
NF(Z6,I);
```

```
==> 0  // Z6 lies in I!
```

We have designed an algorithm for computing the intersection of an ideal with the subalgebra. Using it, we obtain that $I \cap Z(U(\mathfrak{g}_2))$ is equal to the ideal $\langle Z_2^2 - 6Z_2, Z_6 \rangle$. In particular, the center $Z(U(\mathfrak{g}_2)/I)$ of the factor–algebra $U(\mathfrak{g}_2)/I$ equals $\mathbb{K}[Z_2]/\langle Z_2^2 - 6Z_2 \rangle \cong \mathbb{K} \oplus \mathbb{K} \cdot \mathrm{NF}(Z_2, I)$.

3.2 Combined computations

While working in mixed commutative and noncommutative setting, it is quite comfortable to use all the commutative functionality of SINGULAR ([2]). It is in particular useful in problems, involving sequences of noncommutative preprocessings and commutative postprocessings. We illustrate this by computing one–dimensional representations of the algebras $U_q'(\mathfrak{so}_3)$.

The Fairlie–Odesskii algebra $U_q'(\mathfrak{so}_3)$ ([1]) is an associative unital algebra with generating elements I_1, I_2, I_3 and defining relations $q^{1/2}I_1I_2 - q^{-1/2}I_2I_1 = I_3, \quad q^{1/2}I_2I_3 - q^{-1/2}I_3I_2 = I_1, \quad q^{1/2}I_3I_1 - q^{-1/2}I_1I_3 = I_2,$ where $q \neq 0, \pm 1$, is a complex number, called *deformation parameter*. In the limit $q \to 1$, the algebra $U_q'(\mathfrak{so}_3)$ reduces to the enveloping algebra $U(\mathfrak{so}_3)$. All of these algebras are, of course, G–algebras.

Lemma 3. *Let A be a G–algebra over $\mathbb{K}$, generated by $x_1, \dots, x_n$. Then $\overline{a} := (a_1, \dots, a_n) \in \mathbb{K}^n$ is a one–dimensional representation of A if and only if the ideal $\mathfrak{m}_a := \langle x_1 - a_1, \dots, x_n - a_n \rangle$ is proper in A.*

From the lemma it becomes clear how to compute all the one–dimensional representations of a given algebra. Below is the PLURAL code for computing such representations of $U_q'(\mathfrak{so}_3)$ in three cases. We equip every case with the string, setting value of minimal polynomial for $q := Q^2$ in the PLURAL language.

A) $U_q'(\mathfrak{so}_3)$; there is no string with `minpoly` since q is a free parameter;

B) $U_q'(\mathfrak{so}_3)$; `minpoly=Q^4+Q^2+1;` that is q is a 3^{rd} primitive root of unity;

C) $U_q(\mathfrak{so}_3)$; `minpoly=Q-1;` simulates the limit $q \to 1$.

```
ring r=(0,Q),(I1,I2,I3,a,b,c),dp; // here Q^2=q
minpoly=...; // here goes the string from one of the above cases
matrix C[6][6]; matrix D[6][6]; int i,j;
for(i=1;i<6;i++)
{
  for(j=i;j<=6;j++)  {C[i,j]=1;}
}
C[1,2]=Q2; C[1,3]=1/Q2; C[2,3]=Q2;
D[1,2]=-Q*I3; D[1,3]=1/Q*I2; D[2,3]=-Q*I1;
system("PLURAL",C,D);
option(redSB); option(redTail); // any output will be completely reduced
ideal pRep=I1-a,I2-b,I3-c;
ideal Rep=eliminate(pRep,I1*I2*I3); // now Rep is in K[a,b,c]
LIB "primdec.lib";
list Lrep=minAssChar(Rep);   // we need the minimal associated primes
for (i=1;i<=size(Lrep);i++)
{Lrep[i]=simplify(Lrep[i],1);}
Lrep;
```

Note, that all the cases share the same trivial presentation $(0, 0, 0)$ which we ignore below.

A) Let $t = \frac{q^{1/2}}{q-1}$. There are four nontrivial one–dimensional representations :

$$\mathfrak{Rep}_1 = \{\Big((-1)^i t, (-1)^j t, (-1)^{i+j} t \Big) \mid 1 \le i < j \le 4\}.$$

B) Let $t_1 = \frac{2q^{1/2}+1}{3}, t_2 = \frac{2q^{1/2}-1}{3}$. There are eight nontrivial one–dimensional representations:

$$\mathfrak{Rep}_1 = \{\Big((-1)^i t_m, (-1)^j t_m, (-1)^{i+j} t_m \Big) \mid 1 \le i < j \le 4,\ m = 1, 2\}.$$

C) There are no nontrivial one–dimensional representations.

4 Remarks on Implementation

The authors have written an article [4] about the system PLURAL. Here are the crucial points to mention:

- we use generalized "Product" and "Chain" criteria in Buchberger's algorithm
- various flavors of *geobuckets* ([5]) are used for the reduction, multiplication and other operations
- PLURAL will become a dynamical module for SINGULAR

The authors would like to acknowledge the support provided by the Deutsche Forschungsgemeinschaft (DFG).

References

[1] Havlicek, M. and Klimyk, A. and Posta, S. Central elements of the algebras $U_q'(\mathrm{so}_m)$ and $U_q'(\mathrm{iso}_m)$. *arXiv. math. QA/9911130*, 1999.

[2] Greuel, G.-M. and Pfister, G. with contributions by Bachmann, O. ; Lossen, C. and Schönemann, H. *A SINGULAR Introduction to Commutative Algebra*. Springer, 2002.

[3] Levandovskyy, V. PBW Bases, Non–Degeneracy Conditions and Applications. In Buchweitz, R.-O. and Lenzing, H., editor, *Proceedings of the ICRA X conference*, to appear.

[4] Levandovskyy V.; Schönemann, H. Plural — a computer algebra system for noncommutative polynomial algebras. In *Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'03)*. ACM Press, 2003.

[5] T. Yan. The geobucket data structure for polynomials. *J. Symbolic Computation*, 25(3):285–294, March 1998.