UNIVERSITÄT KAISERSLAUTERN
Zentrum für Computeralgebra

REPORTS ON COMPUTER ALGEBRA
NO. 36

Plural, a Noncommutative Extension of Singular: Past, Present and Future

by

V.Levandovskyy

June, 2006

The Zentrum für Computeralgebra (Centre for Computer Algebra) at the University of Kaiserslautern was founded in June 1993 by the Ministerium für Wissenschaft und Weiterbildung in Rheinland-Pfalz (Ministry of Science and Education of the state of Rheinland-Pfalz). The centre is a scientific institution of the departments of **Mathematics, Computer Science, and Electrical Engineering** at the University of Kaiserslautern.

The goals of the centre are to advance and to support the use of Computer Algebra in industry, research, and teaching. More concrete goals of the centre include

- the development, integration, and use of software for Computer Algebra
- the development of curricula in Computer Algebra under special consideration of interdisciplinary aspects
- the realisation of seminars about Computer Algebra
- the cooperation with other centres and institutions which have similar goals

The present coordinator of the Reports on Computer Algebra is:
Christoph Lossen (email: `lossen@mathematik.uni-kl.de`)

Zentrum für Computeralgebra

c/o Prof. Dr. G.-M. Greuel, FB Mathematik
Erwin-Schrödinger-Strasse
D-67663 Kaiserslautern; Germany
Phone: 49 - 631/205-2850   Fax: 49 - 631/205-5052
email: `greuel@mathematik.uni-kl.de`

Plural, a Non–commutative Extension of Singular: Past, Present and Future

Viktor Levandovskyy

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University
Altenbergerstrasse 69, 4040 Linz, Austria
levandov@risc.uni-linz.ac.at

Abstract. We describe the non–commutative extension of the computer algebra system Singular, called Plural. In the system, we provide rich functionality for symbolic computation within a wide class of non–commutative algebras. We discuss the computational objects of Plural, the implementation of main algorithms, various aspects of software engineering and numerous applications.

Singular:Plural or, shortly, Plural [1] is a subsystem of a computer algebra system Singular [2]. It provides the framework for symbolic computations with one– and two–sided ideals and modules over non–commutative *GR*–algebras (Def. 2). Most of Gröbner basics (Sect. 2.5) are available in the kernel of the implementation, ranging from the elimination of variables to free resolutions. Additional functions and libraries provide many useful tools and advanced algorithms for non–commutative algebra. The powerful implementation and rich functionality make Plural a very helpful system for supporting the research in many fields on mathematics and its applications.

1 Past

In 1997, Gert–Martin Greuel and Yuriy Drozd proposed to modify the experimental branch of Singular, called SingularD, which contained implementations of Gröbner bases and syzygies for modules over Weyl and exterior algebras. The list of work to be accomplished included the extension of the class of available algebras (having in mind universal enveloping algebras of finite dimensional Lie algebras), thorough implementation of Gröbner bases and of related algorithms for these algebras.

In the year 2000, the author defended his Master Thesis in Kaiserslautern, which was entitled "Gröbner bases of a class of non–commutative algebras" and presented the first version of Plural. The class of implemented algebras was bigger, than it was originally planned. Indeed, it constituted the class, studied by J. Apel under the name of *G–algebras* [3], and by A. Kandri–Rody and V. Weispfennig under the name *algebras of solvable type* [4]. T. Mora investigated these algebras among other in his works [5, 6] without giving them a special name. It is important, that many quantum groups and different flavors of quantizations, applied to various algebras [7–9], are G–algebras (Def. 1) or their factor algebras, GR–algebras (Def. 2).

As a name, PLURAL originates from a wordplay. In the funny informal discussion on the 1^{st} of April 1999 (*the fool's day*), among other jokes around maths, it appeared suddenly as the contrary to the word "Singular" in the meaning of a grammar category. Therefore, the question "how to call the new–born SINGULAR extension" has got a quick answer, which was accepted then by all the authors and principal developers of SINGULAR.

Until the 2005, PLURAL was still separated from SINGULAR de jure, but de facto PLURAL was included in the development structure of SINGULAR, although it was built in a different way, it kept its own separate documentation and so on. During 2001–2005 a standalone PLURAL was released several times, and became available for the free download. Many new algorithms were developed and implemented. A Gröbner basis algorithm was radically enhanced and profited from all the novelties in the kernel of SINGULAR like different kinds of geobuckets, fast internal maps et cetera. The development of the kernel of PLURAL was done by the author together with Hans Schönemann, and we have reported on some aspects of our work in [10].

Finally, in mid 2005 SINGULAR version 3-0-0 was released, with PLURAL as an integral part of it. Almost at the same time the Ph.D. Thesis [8] was defended by the author, where most of the theoretical and algorithmic research, connected to PLURAL, was described in detail.

2 Present

PLURAL operates with ideals and submodules of free modules of finite rank over non–commutative GR–algebras.

2.1 *GR*–algebras and their properties

Let $\mathbb{K}$ be a field, and $T = T_n = \mathbb{K}\langle x_1, \ldots, x_n \rangle$ a free associative $\mathbb{K}$–algebra, generated by $\{x_1, \ldots, x_n\}$ over $\mathbb{K}$. Among the **monomials** $x_{i_1} x_{i_2} \ldots x_{i_s}$, $1 \leq i_1, i_2, \ldots, i_s \leq n$, spanning T as vector space over $\mathbb{K}$, we distinguish the **standard monomials** $x_{i_1}^{\alpha_1} x_{i_2}^{\alpha_2} \ldots x_{i_m}^{\alpha_m}$, where $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ and $\alpha_k \in \mathbb{N}$. Via the correspondence $x^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n} \mapsto (\alpha_1, \alpha_2, \ldots, \alpha_n) =: \alpha$ the set of standard monomials is in bijection with $\mathbb{N}^n$.

Recall, that any finitely generated associative $\mathbb{K}$–algebra is isomorphic to T_n/I, for some n and some proper two–sided ideal $I \subset T_n$. If the set of standard monomials forms a $\mathbb{K}$–basis of an algebra $A = T/I$, we say that A has a Poincaré–Birkhoff–Witt (shortly, PBW) basis in the variables $x_1, \ldots, x_n$. We say that an abstract associative algebra A **has a PBW basis**, if there exists an isomorphism of $\mathbb{K}$–algebras $A \cong T/I$, such that T/I has a PBW basis.

As one can immediately see, the commutative polynomial ring $\mathbb{K}[x_1, \ldots, x_n]$ does have a PBW basis, while the free associative algebra $\mathbb{K}\langle x_1, \ldots, x_n \rangle$ does not. The existence of a PBW basis is an important property for a non–commutative algebra. However, we need more assumptions on the particular basis and the relations of an algebra in order to guarantee nice properties. In particular, the algebra $\mathbb{K}\langle x,y\rangle/\langle yx\rangle$ has a PBW basis, but it is not an integral domain.

A total ordering $\prec$ on $\mathbb{N}^n$ is called a monomial ordering on the algebra A with the PBW basis $\{x^\alpha \mid \alpha \in \mathbb{N}^n\}$, if $\forall\ \alpha,\beta,\gamma \in \mathbb{N}^n,\ \alpha \prec \beta \ \Rightarrow\ x^\alpha \prec x^\beta \Rightarrow x^{\alpha+\gamma} \prec x^{\beta+\gamma}$. For $f \in T$, we denote by $\operatorname{lm}(f)$ the leading monomial of f with respect to $\prec$.

Definition 1. *Let $\mathbb{K}$ be a field, $T = \mathbb{K}\langle x_1,\dots,x_n\rangle$ and I be a two–sided ideal of T, generated by the set of elements*

$$x_j x_i - c_{ij}\cdot x_i x_j - d_{ij}, \quad 1 \le i < j \le n,$$

where $c_{ij} \in \mathbb{K}\setminus\{0\}$ and every $d_{ij} \in T$ is a polynomial, involving only standard[1] *monomials of T. A $\mathbb{K}$–algebra $A = T/I$ is called **a G–algebra**, if the following conditions hold:*

- ***Ordering condition**: there exists a monomial well–ordering $\prec$ on $\mathbb{N}^n$, such that $\forall\ 1 \le i < j \le n \ \ \operatorname{lm}(d_{ij}) \prec x_i x_j$.*
- ***Non–degeneracy condition**: $\forall\ 1 \le i < j < k \le n$, to the sets $\{c_{ij}\}$ and $\{d_{ij}\}$ we associate a polynomial $NDC_{ijk} = c_{ik}c_{jk}\cdot d_{ij}x_k - x_k d_{ij} + c_{jk}\cdot x_j d_{ik} - c_{ij}\cdot d_{ik}x_j + d_{jk}x_i - c_{ij}c_{ik}\cdot x_i d_{jk}$. A condition is satisfied, if each NDC_{ijk} reduces to zero with respect to the generators of I.*

The PBW Theorem (from e.g. [11]) generalizes the classical Poincaré–Birkhoff–Witt Theorem from the case of universal enveloping algebras of finite dimensional Lie algebras to the case of general G–algebras. Hence, a G–algebra in variables $x_1,\dots,x_n$ has a canonical PBW basis $\{x_1^{\alpha_1}x_2^{\alpha_2}\dots x_n^{\alpha_n} \mid \alpha_k \in \mathbb{N}\}$.

Definition 2. *Let B be a G–algebra and $I \subset B$ be a proper nonzero two–sided ideal. Then, a factor algebra B/I is called a GR–**algebra**.*

Remark 1 (Setup for G–algebras). There are several ways to input a G–algebra in PLURAL. We extended the SINGULAR type `ring` to the non–commutative situation. A generic way for setting up a G–algebra follows the definition above: one has to define a commutative ring $\mathbb{K}[x_1,\dots,x_n]$ and equip it with the monomial ordering. Then, one defines two $n \times n$ matrices $C = (c_{ij})$ and $D = (d_{ij})$, and types `ncalgebra(C,D)`. The command `ncalgebra` accepts shortcuts for C or D, that is if one passes an argument of types `number` or `poly`, they are interpreted by `ncalgebra` as matrices, where all the entries of the upper triangle are equal to the given argument.

Many commonly used families of algebras are predefined in PLURAL libraries. The number of available predefined algebras increases constantly, one can found them in the libraries NCALG.LIB, NCTOOLS.LIB, and QMATRIX.LIB.

We also provide the possibility to build tensor products of two GR–algebras over the ground field and the construction of the opposite and the enveloping algebra from the given GR–algebra. We discuss the latter algebras below.

[1] we assume this only for simplicity of presentation

Remark 2 (Setup for GR–algebras).

As soon as the G–algebra is given, one can define a factor algebra modulo a two–sided ideal, that is a GR–algebra, which will be of the type `qring`, modified for the non–commutative situation. The only requirement is that a two–sided ideal must be given in its two–sided Gröbner basis. With the help of the command `twostd` one computes such a Gröbner basis and the simplest syntax for defining a GR–algebra reads as `qring Q = twostd(I);`.

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

1) A is left and right Noetherian,
2) A is an integral domain,
3) A is Auslander–regular and Cohen–Macaulay,
4) the Gel'fand–Kirillov dimension $\mathrm{GKdim}(A) = n + \mathrm{GKdim}(\mathbb{K})$,
5) the global homological dimension $\mathrm{gl.\,dim}(A) \leq n$,
6) the Krull dimension $Kr.dim(A) \leq n$.

We refer to [12], [8], [13] for corresponding definitions and proofs. There are examples, where the inequalities 5) and 6) are strict. In particular, 1) and 2) imply that every G–algebra satisfies a left and a right Ore conditions, hence there exist a total Ore localization, producing a left and a right quotient ring. It is known since [3], that one can use Gröbner bases on a G–algebra A for the arithmetic operations with fractions of its left or right quotient ring.

Remark 3. As for computation of dimensions, one can count only on the algorithm for the calculation of Gel'fand–Kirillov dimension GKDIM.LIB [7], which is implemented for PLURAL by F. J. Lobillo et. al. The generalized Krull dimension is known for its difficulty and, to the best of our knowledge, there is no algorithmic procedure for computing it for a general GR–algebra. We have proved in [8], that the global homological dimension $\mathrm{gl.\,dim}(A) = n$ provided there exist finite dimensional representations of A over the ground field $\mathbb{K}$. It is still an open question, whether the opposite direction is true. Another open problem is the exact computation of $\mathrm{gl.\,dim}$ of a given algebra in the case, when $\mathrm{gl.\,dim}(A) < n$. The phenomenon, demonstrated by n–th Weyl algebras W_n over a field of characteristic 0, is quite interesting. In this case $\mathrm{gl.\,dim}(W_n) = n$, while W_n is generated by $2n$ variables and is of Gel'fand–Kirillov dimension $2n$. This behavior is extremal in the sense that the global dimension of a G–algebra in $2n$ variables seems to be at least n.

The class of G–algebras unifies many very important and quite different algebras under one roof, among them quasi–commutative polynomial rings (for example, the quantum plane $yx = q \cdot xy$ and multiparameter quantum affine spaces), universal enveloping algebras of finite dimensional Lie algebras, some iterated Ore extensions, many quantum groups, some nonstandard quantum deformations, many algebras associated to the classical operators.

One of the reasons for such unification lies in the common structural properties of these algebras. And the second reason is the Gröbner bases theory.

2.2 Gröbner bases in GR–algebras

We stress the similarities between G–algebras and commutative polynomial rings and use the similarities, when possible. We follow the approach to Gröbner bases, presented in [14]. Let A be a G–algebra in n variables. We say that a **monomial** of a free module A^r (involving component i) is an element of the form $x^\alpha e_i$, where $\alpha \in \mathbb{N}^n$ and e_i is the canonical i–th basis vector.

We say, that $m_1 = x^\alpha e_j$ **divides** $m_2 = x^\beta e_k$ and denote it by $m_1|m_2$, if $j = k$ and $\alpha_i \le \beta_i \ \forall i = 1 \dots n$. Note, that it is rather a pseudo–division on A, since if $m_1|m_2$, then there exist $c \in \mathbb{K} \setminus \{0\}$, a monomial $p \in A$ and $q \in A^r$ such that $\mathrm{lm}(q) \prec m_1$ and $m_2 = c \cdot p \cdot m_1 + q$, where $q \neq 0$ in general.

From the properties of G–algebras it follows, that any $f \in A^r \setminus \{0\}$ can be written uniquely as $f = c_\alpha x^\alpha e_i + g$, with $c_\alpha \in \mathbb{K}^*$, and $x^\beta e_j \prec x^\alpha e_i$ for any nonzero term $c_\beta x^\beta e_j$ of g. Then we define in the usual fashion

$\mathrm{lm}(f) = \ x^\alpha e_i$, the **leading monomial** of f,

$\mathrm{lc}(f) \ = \ c_\alpha$, the **leading coefficient** of f.

Note, that $\forall\ \alpha, \beta \in \mathbb{N}^n$, $\mathrm{lm}(x^\alpha x^\beta) = \mathrm{lm}(x^{\alpha+\beta}) = \mathrm{lm}(x^\beta x^\alpha)$.

Definition 3. *Let $\prec$ be a monomial ordering on the free module A^r, $I \subset A^r$ a left submodule, and $G \subset I$ a finite subset. G is called a **left Gröbner basis** of I if and only if for any $f \in I \setminus \{0\}$ there exists $g \in G$, satisfying* $\mathrm{lm}(g) \mid \mathrm{lm}(f)$.

In order to come up with the more constructive definition, one has to use the notion of a monoideal of leading exponents [7] or a span of leading monomials [8] instead of the leading ideal. The latter works well in the commutative and even in the free associative algebras, but fails in general G–algebras for the reasons, which we discussed in detail in [8].

The normal form, the s–polynomial and the Buchberger's algorithm can be generalized for the left or right ideals in almost the same form they appear in the literature for the commutative case. However, the proofs of main theorems in the Gröbner bases theory are different in spite of similarity. One has to develop a specific intuition, while working with non–commutative algebras, which are in many senses close to commutative algebras.

As the simplest indication of the intrinsic difference we can take the Product Criterion, which is a standard tool in the commutative case. If the leading monomials of two polynomials f and g do not divide each other, we have $\mathrm{spoly}(f, g) \to_{\{f,g\}} 0$. Hence, this is the easiest situation in the set of pairs, built in the Buchberger's algorithm: discard the pair (f, g) if the condition holds.

In the non–commutative case, we can show under some assumptions on the algebra [10], that $\mathrm{spoly}(f, g) \to_{\{f,g\}} g \cdot f - f \cdot g =: [g, f]$. Of course, it allows to discard the pair (f, g) from the pair set if f commutes with g, which happens rather rarely in general. If it is not the case, the number of multiplications and reductions shows that we are perhaps in the worst situation, which might occur in the set of pairs.

On the contrary, the Chain Criterion and its variations generalize to G–algebras in its full generality [7, 15, 8, 9]. The Chain Criterion is actually the most important criterion, used in PLURAL.

2.3 Left, right and two–sided structures

The three kinds of ideals and modules (left, right and two–sided) might make the life of a developer quite complicated. The two–sided ideals and, more generally, bimodules are very special structures, having no analogue in the commutative case. The notion of a two–sided Gröbner basis is different from the one of a one–sided Gröbner basis [3, 4, 10]. The two–sided Gröbner basis is computed with a special algorithm and is in general harder to compute, then the one–sided. A recent algorithm [16] shows superior performance, compared to the variations of the classical approach and will be used in the future. This algorithm utilizes the opposite algebras.

Let A be an associative algebra over $\mathbb{K}$. The **opposite algebra** A^{opp} is defined by taking the same vector space as of A, and by introducing a new "opposite" multiplication $*$ on it, defined by $f \ * \ g \ := \ g \cdot f$. Then, A^{opp} is an associative $\mathbb{K}$–algebra, and $(A^{\text{opp}})^{\text{opp}} = A$ holds. One calls $A \otimes_{\mathbb{K}} A^{\text{opp}}$ an **enveloping algebra** of A.

Lemma 1. *Let $B = A/I$ be a GR–algebra. Then B^{opp} is a GR–algebra, and $B^{\text{opp}} = A^{\text{opp}}/I^{\text{opp}}$.*

The particular importance of opposite algebras lies in the fact, that for right–sided computations with a right module like a Gröbner basis, a syzygy module et cetera, it suffices to implement a left–sided functionality together with procedures for the effective treatment of opposite algebras and transfer of objects between an algebra and its opposite. The implementation in SINGULAR:PLURAL is done along these lines; we provide the commands `opposite` and `envelope` for constructing the algebras and `oppose` for transferring the objects from an algebra to its opposite.

There are several methods for representing an opposite algebra of a given algebra constructively, see [8] for their description.

2.4 Gröbner trinity and Gröbner engine

We can compute Gröbner basis of an ideal, Gröbner basis of its first syzygy module, and the transformation matrix between the original set of generators and the Gröbner basis (sometimes called a *lifting matrix*) basically with the same algorithm. We call these three powerful algorithms a **Gröbner trinity**. The same applies for one–sided Gröbner trinity for ideals over *GR*–algebras and is inherited by PLURAL from SINGULAR. The Gröbner trinity is extremely important for further applications of Gröbner bases: e.g. a free resolution can be computed as the sequence of syzygies, while a lifting matrix allows to control the critical constellations of parameters, or, in other words, to observe the genericity of Gröbner basis computation [17].

The algorithm, which is able to compute all of the Gröbner trinity, is essentially the general version of Buchberger's Gröbner basis algorithm. It must be able to compute with free modules, hence it must accept monomial module orderings as input. Moreover, it is important to have the switch for dividing the set of module components into two disjoint groups. Having such a switch, one can, depending on the situation, compute Gröbner basis only of those vectors, which lie inside of one group and do not compute it for the other group, since the latter will be ignored at the end. Among other cases, this idea is used for computing both the syzygy module and the lifting matrix more easily. The same algorithm must be able to perform computations in a factor algebra, to use extra weights for the ordering or for the generators of a module, to interpret and to use on demand the supplemented information on Hilbert polynomial et cetera.

We call an implementation of the algorithm, which computes a (left) Gröbner basis and which complies with the requirements above, a **Gröbner engine**.

The examples of Gröbner engines in SINGULAR are: Gröbner bases (non–negatively graded orderings), standard bases (local and mixed orderings), and PLURAL (left Gröbner bases for non–negatively graded orderings over G–algebras). All of these are called with the same command, namely `std`. Yet more methods for computing Gröbner bases are on their way to become someday Gröbner engines.

If the internal implementation of a variant of Gröbner basis algorithm is done in the form of Gröbner engine, one gets all the Gröbner basics (see below) available in a much shorter time, compared with the adjustment of every application from Gröbner basics family to the new Gröbner basis routine. Moreover, if internal structure of the implementation of e.g. Gröbner basics is tuned for the use of generic Gröbner engine, one can switch between engines and measure the engine's performance on the suite of applications.

The importance of having not only a fast Gröbner basis algorithm, but also fast Gröbner basics (for working with practice–relevant applications) is clear. However, the development of a fast Gröbner basis algorithm is complicated and requires several years of intensive work. Unfortunately, during this time the initial plans concerning applications often got changed, by shifting the focus to the better Gröbner basis algorithm. In such a situation, the development of Gröbner basics and further applications is often postponed. The concept of Gröbner engine has been used implicitly in SINGULAR. Now it is being formalized and developed further by the author and Hans Schönemann. It provides an interface between Gröbner bases, Gröbner trinity and Gröbner basics in order to overcome the difficulties, described above.

The development of a novel implementation of a Gröbner basis algorithm, which must not necessarily be the Buchberger's Algorithm, usually starts with ideals and aims at a couple of orderings. The evolution of each concrete project is unique, because it depends on the aims, pursued by the developers. But at some point one has to introduce modules, more orderings including elimination and module ones, factor algebras et cetera. Our experience can be illustrated with two algorithms, available in SINGULAR, namely `janet` and `slimgb`.

`janet`. The possibility to compute Gröbner basis via *involutive basis* was proposed independently by Apel and Gerdt et. al. [18], the corresponding algorithm was thoroughly implemented in the group of V. P. Gerdt (`http://invo.jinr.ru`) for ideals of commutative rings and demonstrated quite a good performance. With the help of the principal developer of the project JB ("Janet involutive bases"), Denis Yanovich, in 2003 we have incorporated their routines, written in C, into Singular. We have learned a lot during that process; the amount of re-engineering we needed to do, together with several other factors, led us to the idea of Gröbner engine, because at that time we were interested in having a fast Gröbner basis algorithm, showing especially good performance on the elimination problems.

The Singular command `janet` computes a Gröbner basis of an ideal through the computation of Janet basis and interreduction of the output. The same command, run in the G–algebra, returns a left Gröbner basis of a two–sided ideal. The cooperation with the group of Gerdt continues, and perhaps some day `janet` routines will evolve to the Gröbner engine.

`slimgb`. Slim Gröbner basis is the algorithm of M. Brickenstein [19, 20]. It uses many interesting ideas and techniques, which have been proved to provide an impressive performance, especially over transcendental field extensions and for elimination orderings. One of particular aims was to minimize, if possible, the intermediate coefficient swell.

The methods, used in `slimgb`, were general enough to be applied for the non–commutative case. `slimgb` can compute a left Gröbner basis of a left module. Its performace has been successfully tested on many problems; using `slimgb` we obtained solutions for several long–standing computational challenges. Due to very good timings on examples, where elimination orderings were used, `slimgb` is the primary engine for the dmod.lib (Sect. 3.5). The development of `slimgb` goes further intensively and, as it seems, will lead to the Gröbner engine in the nearest future.

2.5 Gröbner basics

Bruno Buchberger and later Bernd Sturmfels called "Gröbner basics" the most important, yet basic applications of Gröbner bases. We adopt this notion to the non–commutative GR–algebras and remove from this list "too commutative" applications (such as Zariski closure of the image of a map, solving polynomial equations and radical membership). All the algorithms below have been generalized to the context of GR–algebras and implemented in Plural.

- Ideal (resp. module) membership problem
- Intersection with subrings (elimination of variables)
- Intersection of ideals (resp. submodules)
- Quotient and saturation of two–sided ideals ($*$)
- Kernel of a module homomorphism
- Kernel of a ring homomorphism
- Algebraic relations between pairwise commuting polynomials

The items, marked with $*$, have constructive interpretation only for two–sided input.

Definition 4. *Let A be a $\mathbb{K}$–algebra and $F \subseteq A$ a set. The subalgebra $C_A(F) = \{a \in A \mid [f, a] = 0 \ \forall f \in F\}$ is called the **centralizer of** F **in** A. Moreover, $Z(A) = C_A(A) = \{z \in A \mid za = az \ \forall a \in A\}$ is called the **center** of A.*

In addition to the classical Gröbner basics, there are typically non–commutative Gröbner basics (all of them are implemented in PLURAL):

- Two–sided Gröbner basis of a bimodule
- Gel'fand–Kirillov dimension of a module
- Annihilator of finite dimensional module
- Central quotient resp. saturation of ideals (if the center is non–trivial)
- Preimage of a left ideal under the morphism of algebras
- Graded Betti numbers (for graded modules over graded algebras)
- Left and right kernel of the presentation of a module
- Central Character Decomposition of the Module

It is interesting, whether it is possible to give an algorithm, which computes N–dimensional irreducible representations of a GR–algebra for a positive N. We have proposed an algorithm, which computes all the one–dimensional representations [21].

For a modern computer algebra system, specializing on the non–commutative algebras, it is quite important to have also non–Gröbner functionality, like the operations with opposite and enveloping algebras (described above), computations with centralizers and even more. Many applications (of e.g. representation theory) require an explicit knowledge of the generators of the center of a GR–algebra as well as the generators of centralizers of finite sets. These algorithms have been implemented in the library CENTER.LIB by O. Motsak. The implementation demonstrated quite a good performance.

While studying algebraic dependence of pairwise commuting polynomials, the method of Perron polynomial was widely used. It has been implemented in the library PERRON.LIB. With this library we have been able to compute several hard examples, which contributed to the progress in studying algebraic dependence in the situation, described in the Sect. 3.2.

3 Work in progress and future development

3.1 Preimage of a left ideal

NCPREIMAGE.LIB is dedicated to the computation of the preimage of a left ideal under a morphism of GR–algebras, as it is described in [22]. The implementation of the main algorithm of the article requires, among other, the procedure for the computation of a tuple of strictly positive weights $(w_1, \ldots, w_m)$, such that the elimination ordering with the extra weight vector $(w_1, \ldots, w_m, 0, \ldots, 0)$ satisfies the ordering condition of the Def. 1. If one works with a positively weighted degree ordering, a similar computation of weights can be achieved with the help of the method, described in e.g. [7]. It is implemented as the procedure `Gweights` in the library NCTOOLS.LIB.

3.2 Algebraic dependence of pairwise commuting polynomials

Consider the universal enveloping algebra A of a finite dimensional simple Lie algebra over a field $\mathbb{K}$. If $\operatorname{char} \mathbb{K} > 0$, it is known from the dimension argument assures, that the generators of the center are algebraically dependent. There are several open questions on the ideal of dependence polynomials which we investigate by using computer algebraic methods. We were able to compute the dependence polynomials explicitly for many prime p over the algebras $U(\mathfrak{sl}_2)$ (see [8]) and $U(\mathfrak{so}_3)$. Up to now, the case of $U(\mathfrak{sl}_3)$ remains unsolved and constitutes an important challenge.

There are more situations, when these methods can be applied. For instance, the algebraic dependence of the generators of the center appears also in quantum algebras, when one considers a quantum parameter q (usually assumed to be transcendental over $\mathbb{K}$) to be some primitive root of unity. We computed several important dependencies in the nonstandard quantum deformation $U'_q(\mathfrak{so}_3)$ and reported on these results in e.g. [8].

The computation of dependencies as described above seems to be one of the hardest tasks in the non–commutative computer algebra. To the best of our knowledge, no computer algebra system except SINGULAR:PLURAL can handle even relatively small examples.

In many cases the obtained dependencies define singularities. There is a conjecture, that for some types of algebras, these singularities will always be simple. The practical experience shows that this conjecture is true for all the computed examples, the investigation is continued.

3.3 Homological algebra in *GR*–algebras

For two left A–modules M, N, $\mathrm{Ext}^i_A(M,N)$ for $i \geq 0$ carries no A–module structure in general. However, it turns out [7], that in the case, when either M or N is a *centralizing bimodule*, $\mathrm{Ext}^i_A(M,N)$ is an A–module and its presentation can be computed algorithmically. In many applications, one of the modules M, N is often appears to be a centralizing bimodule.

Together with G. Pfister we are working on the implementation of the methods above in the library NCHOMOLOG.LIB. It is planned to have procedures for the computation of Ext and Tor modules in the setup as above, accompanied by other useful tools for homological algebra. We will use these also for the algorithmic computation of Hochschild cohomology of bimodules. We need to compute left and right Gröbner bases, and two–sided bases for bimodules; the need for them motivated, among other, the deeper study and the enhanced implementation of opposite and enveloping algebras.

With the help of the library, we are going to check the long–standing conjecture, starting with algebras of rank 2 and 3:

> *for any simple weight module M over a complex finite–dimensional simple Lie algebra $\mathfrak{g}$, $\dim_{\mathbb{C}} H^i(\mathfrak{g}, M) < \infty$ holds for all i.*

All the computations, related to this conjecture can be done in the universal enveloping algebra $U(\mathfrak{g})$, which is a G–algebra. Among other, the library will be applied to the problems, arising in the systems and control theory.

3.4 Systems and control theory

The algorithmic methods of algebraic analysis can be applied to systems of equations involving linear operators like the (partial) differentiation, shift, difference and so on [23, 24]. The algorithms for the case, when a system of equations involves only constant coefficients (hence, the system algebra is commutative), have been implemented in the library CONTROL.LIB (Becker, L., and Yena, 2004).

When treating systems with variable polynomial coefficients, the system algebra becomes a GR–algebra. Together with E. Zerz we are working on the library NCONTROL.LIB. This library will provide the procedures for the algebraic analysis of systems over not only G–algebras (like it is done in the package ORE-MODULES, [24]), but also in GR–algebras. The latter requires more efforts and a thorough inspection of the theory and its implementation.

In order to treat systems with rational coefficients, we have to provide Gröbner bases, Gröbner basics, and algorithmic homological algebra for modules over Ore–localized G–algebras (see Sect. 3.7). This is planned to achieve in the near future.

3.5 *D*–modules

The library DMOD.LIB (V. L. and J. Morales, 2006) contains procedures for computations with D–modules. Let $\operatorname{char}\mathbb{K} = 0$. Given a polynomial $F \in \mathbb{K}[x_1, \dots, x_n]$, one is interested in computing the D–module structure of the localization $\mathbb{K}[x_1, \dots, x_n, F^s]$ for negative integer s. That is, one looks for the left ideal I in the Weyl algebra $D := A_n$ in $2n$ variables $\{x_1, \dots, x_n, \partial_1, \dots, \partial_n\}$, such that $\mathbb{K}[x_1, \dots, x_n, F^s] \cong A/I$ as D–modules. The algorithm for the computation of such I is often called $\operatorname{Ann} F^s$.

We have implemented two variants of this algorithm, namely the algorithm of Oaku and Takayama [25, 26] in the procedure `annfsOT`, and the algorithm of Briançon and Maisonobe (e.g. [25]) in the procedure `annfsBM`. One can use both `std` and `slimgb` as underlying Gröbner engine for these complicated algorithms. With the current implementation of DMOD.LIB and `slimgb`, we were recently able to compute several hard examples, e.g. proposed by Castro and Ucha in [25]. In particular, the cases of F being a cusp $x^p - y^q$ (for coprime $p, q \in \mathbb{N}$), a *Reiffen curve* $x^p + y^q + xy^{q-1}$, $q \geq p + 1 \geq 5$, or a hyperplane arrangement are studied; for two latter cases we provide the auxiliary procedures for their easy setup. We plan to extend the functionality of the library in the direction, described in [26] and [27]. We are going to use the families of examples above as benchmarks and compare the performance of computer algebra systems such as KAN/SM1, MACAULAY2 and SINGULAR:PLURAL.

3.6 Applications to algebraic geometry

W. Decker, C. Lossen and G. Pfister created the library SHEAFCOH.LIB, devoted to the computation of the cohomology of coherent sheaves. The procedure `sheafCohBGG` utilizes the Bernstein–Gel'fand–Gel'fand (BGG) correspondence and the Tate resolution [28]. This algorithm, which uses computation of free resolutions over non–commutative exterior algebra (which is a GR–algebra), is sometimes much faster, than the commutative one, implemented in the procedure `sheafCoh`, which is based on local duality, following the ideas of Eisenbud.

D. Eisenbud and F.-O. Schreyer presented an algorithm for the computation on higher direct image complex of a coherent sheaf under a projective morphism. The implementation of this algorithm in SINGULAR will appear soon. Like in the SHEAFCOH.LIB, the BGG correspondence and hence, the computations over exterior algebras are used.

3.7 Directions of future work

Context–based multiplication. In [10] we have described our approaches to the multiplication in general G–algebra. The next enhancement in this field is the implementation of formula–based multiplication for the simplest *contexts*. Namely, for an affine G–algebra with the relation $yx = q \cdot xy + ax + by + r$, $q, a, b, r \in \mathbb{K}$, $q \neq 0$ it is possible to derive a symbolic formula in a closed form for the multiplication $y^s \cdot x^t = \sum c_{ij} x^i y^j$. To the best of our knowledge, even in this simple situation no general closed–form formula is known. Using a formula instead of the updated tables will clearly require less memory, but eventually will consume more time. Subalgebras of affine type as above occur very often in big G–algebras, and the impact of the formula–based multiplication in such subalgebras on the overall performance of Gröbner basis algorithms is very interesting to investigate.

Combined computations. SINGULAR is one of the few systems, being able to perform *combined computations*, that is both commutative and non–commutative computations in one system. It is important to bring this ability further by implementing *context–based* operations, that is computations, which will derive the subalgebra, where the concrete input resides (e.g. a commutative subalgebra of a G–algebra), and provide the set of most optimized and relevant routines for the concrete computation. A similar method is implemented in SINGULAR as so–called `p-Procs` for polynomial operations over different ground fields.

Ore localizations. We plan to extend PLURAL to a bigger class of non–commutative algebras, connected with G–algebras by means of localization. Since from every G–algebra we can built left and right quotient rings, one can extend the machinery we have developed to partial localization of G–algebras. Let $B \subset A$ be two G–algebras, then we can perform the localization on B (e.g. the total Ore localization). If B happens to be commutative, we can apply different localization, e.g. the localization with respect to a maximal ideal. Note, that variables, not belonging to B, remain polynomial. Such algebras are needed in many algebraic constructions and used in various applications. Let R be a ring, containing $\mathbb{K}[x_1, \ldots, x_n]$ as a subring. Then the Weyl algebra with coefficients in R is defined to be $R\langle \partial_1, \ldots, \partial_n \mid [\partial_i, x_i] = 1, [\partial_j, x_k] = 0 \rangle$. Very important examples are *rational Weyl algebras*, where $R = \mathbb{K}(x_1, \ldots, x_n)$ or *local polynomial Weyl algebras*, with $R = \mathbb{K}[x_1, \ldots, x_n]_{\langle x_1, \ldots, x_n \rangle}$. The standard basis algorithm for the latter has been recently discussed in [29].

PBW rings [7, 15] constitute a general framework, describing such algebras and Gröbner bases for modules over them. Under some assumptions, which reflect the common setup for many applications, such an algebra is called an Ore algebra [23], which has nice properties and is much easier to implement, than a general PBW ring. However, Ore algebras do not cover various important cases of algebras. Therefore, we concentrate ourself on investigating the algorithmic aspects of computations in partial Ore localizations of G–algebras.

The computations in PBW rings are more complicated, than in G–algebras. Even basic arithmetics with one–sided fractions requires the computation of syzygies and hence Gröbner bases [3]. Hence, the implementation of Gröbner bases in such algebras must be done quite carefully. On the other hand, the powerful implementation opens new perspectives for applications of symbolic computation in this segment of non–commutative algebra.

Non–commutative Computer Algebra Systems

We have reviewed in detail the modern Computer Algebra Systems with the non–commutative abilities in [8]. The following systems are designed for the computations in free associative algebras and path algebras:

- BERGMAN by J. Backelin et.al. [30] is a powerful and flexible tool to calculate Gröbner bases, Hilbert and Poincaré–Betti series, Anick resolution, and Betti numbers in non–commutative algebras and in modules over them,
- NCGB by J. W. Helton et.al. [31] is a MATHEMATICA package, being a part of the NCALGEBRA suite,
- OPAL by B. Keller et.al. [32] is the specialized standalone system for Gröbner bases in free and path algebras,
- GBNP (GROBNER) by A. Cohen and D. Gijsbers [33] is a package for GAP 4 with the implementation of non–commutative Gröbner bases for free and path algebras, following the algorithmic approach of Mora [5, 6].

The systems below are mostly restricted to some classes of non–commutative associative algebras, but the computations with them are usually more efficient.

- FELIX by J. Apel and U. Klaus [34] provides generalizations of Buchberger's algorithm to free $\mathbb{K}$–algebras, polynomial rings and G–algebras. Also, the syzygy computations and basic ideal operations are implemented.
- MAS by H. Kredel and M. Pesch [35] contains a large library of Gröbner basis algorithms for computing in non–commutative polynomial rings,
- GROEBNER by F. Chyzak [36] is a MAPLE package, providing Gröbner basis algorithms (including elimination) for Ore algebras,
- a MAPLE package by R. Pearce [37] contains an implementation of Faugère's F4 algorithm for Ore algebras,
- KAN/SM1 by N. Takayama [38], distributed as a part of the system OPENXM, provides Gröbner basis computations in polynomial rings, rings of differential operators, rings of difference and q-difference operators.
- MACAULAY2 by D. Grayson and M. Stillman [39] includes Gröbner basis algorithms for exterior and Weyl algebras and a package for D–module theory.

Acknowledgments

I am grateful to Gert–Martin Greuel, Hans Schönemann and Gerhard Pfister for long and fruitful cooperation, and for their role in the development of Plural. I wish to thank Michael Brickenstein, Denis Yanovich, Javier Lobillo, and all other colleagues for their cooperation and contributions to Singular:Plural.

References

1. Greuel, G.-M., Levandovskyy, V., and Schönemann H.: Plural. A Singular 3.0 Subsystem for Computations with Non–commutative Polynomial Algebras. Centre for Computer Algebra, University of Kaiserslautern (2006) Available from `http://www.singular.uni-kl.de`.
2. Greuel, G.-M., Pfister G., and Schönemann H.: Singular 3.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern (2005) Available from `http://www.singular.uni-kl.de`.
3. Apel, J.: Gröbnerbasen in nichtkommutativen Algebren und ihre Anwendung. Dissertation, Universität Leipzig (1988)
4. Kandri-Rody, A. and Weispfenning, V.: Non–commutative Gröbner bases in algebras of solvable type. J. Symbolic Computation **9**(1) (1990) 1–26
5. Mora, T.: Groebner bases in non-commutative algebras. In: Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'88), LNCS 358 (1989) 150–161
6. Mora, T.: An introduction to commutative and non-commutative Groebner bases. Theor. Comp. Sci. **134** (1994) 131–173
7. Bueso, J., Gómez–Torrecillas, J. and Verschoren, A.: Algorithmic methods in non-commutative algebra. Applications to quantum groups. Kluwer Academic Publishers (2003)
8. Levandovskyy, V.: Non–commutative computer algebra for polynomial algebras: Gröbner bases, applications and implementation. Doctoral Thesis, Universität Kaiserslautern (2005) Available from http://kluedo.ub.uni-kl.de/volltexte/2005/1883/.
9. Li, H.: Noncommutative Gröbner bases and filtered-graded transfer. Springer (2002)
10. Levandovskyy, V. and 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)
11. Levandovskyy, V.: PBW Bases, Non–Degeneracy Conditions and Applications. In Buchweitz, R.-O. and Lenzing, H., ed.: Representation of algebras and related topics. Proceedings of the ICRA X conference. Volume 45., AMS. Fields Institute Communications (2005) 229–246
12. Gómez–Torrecillas, J. and Lobillo, F.J.: Auslander-regular and Cohen-Macaulay quantum groups. J. Algebr. Represent. Theory **7**(1) (2004) 35–42
13. McConnell, J.C. and Robson, J.C.: Noncommutative Noetherian rings. With the cooperation of L. W. Small. Graduate Studies in Mathematics. 30. Providence, RI: American Mathematical Society (AMS) (2001)
14. Greuel, G.-M. and Pfister, G. with contributions by Bachmann, O., Lossen, C. and Schönemann, H.: A SINGULAR Introduction to Commutative Algebra. Springer (2002)
15. Kredel, H.: Solvable polynomial rings. Shaker (1993)

16. García Román, M. and García Román, S.: Gröbner bases and syzygies on bimodules over PBW algebras. J. Symbolic Computation **40**(3) (2005) 1039–1052
17. Levandovskyy, V. and Zerz, E.: Algebraic systems theory and computer algebraic methods for some classes of linear control systems. In: Proc. of the International Symposium on Mathematical Theory of Networks and Systems (MTNS'06). (2006)
18. Gerdt, V.P.: Involutive Algorithms for Computing Groebner Bases. In Pfister G., Cojocaru S. and Ufnarovski, V., ed.: Computational Commutative and Non-Commutative Algebraic Geometry, IOS Press (2005)
19. Brickenstein, M.: Neue Varianten zur Berechnung von Gröbnerbasen. Diplomarbeit, Universität Kaiserslautern (2004)
20. Brickenstein, M.: Slimgb: Gröbner Bases with Slim Polynomials. In: Reports On Computer Algebra No. 35. Centre for Computer Algebra, University of Kaiserslautern (2005) Available from http://www.mathematik.uni-kl.de/∼zca.
21. Levandovskyy, V.: On preimages of ideals in certain non–commutative algebras. In Pfister G., Cojocaru S. and Ufnarovski, V., ed.: Computational Commutative and Non-Commutative Algebraic Geometry, IOS Press (2005)
22. Levandovskyy, V.: Intersection of ideals with non–commutative subalgebras. In: Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'06), ACM Press (2006, to appear)
23. Chyzak, F. and Salvy, B.: Non–commutative Elimination in Ore Algebras Proves Multivariate Identities. J. Symbolic Computation **26**(2) (1998) 187–227
24. Chyzak, F., Quadrat, A. and Robertz, D.: Linear control systems over Ore algebras. Effective algorithms for the computation of parametrizations. In: Proc. of Workshop on Time-Delay Systems (TDS03), INRIA (2003)
25. Castro-Jiménez, F.J. and Ucha, J.M.: On the computation of Bernstein–Sato ideals. J. Symbolic Computation **37** (2004) 629–639
26. Oaku, T. and Takayama, N.: An algorithm for de Rham cohomology groups of the complement of an affine variety via D-module computation. Journal of Pure and Applied Algebra **139**(1–3) (1999) 201–233
27. Saito, S., Sturmfels, B. and Takayama, N.: Gröbner Deformations of Hypergeometric Differential Equations. Springer (2000)
28. Eisenbud, D., Fløystad, G. and Schreyer, F.-O.: Sheaf algorithms using the exterior algebra. Trans. Am. Math. Soc. **355**(11) (2003) 4397–4426
29. Granger, M. and Oaku, T. and Takayama, N.: Tangent cone algorithm for homogenized differential operators. J. Symbolic Computation **39**(3–4) (2005) 417–431
30. J. Backelin et. al.: The Gröbner basis calculator Bergman (2006) Available from `http://servus.math.su.se/bergman/`.
31. Helton, J.W. and Stankus, M.: NCGB 3.1, a Noncommutative Gröbner Basis Package for Mathematica (2001) Available from `http://www.math.ucsd.edu/~ncalg/`.
32. Green, E., Heath, L., and Keller, B.: Opal: A System for Computing Noncommutative Gröbner Bases. In: RTA '97: Proceedings of the 8th International Conference on Rewriting Techniques and Applications, Springer (1997) 331–334
33. Cohen, A.M. and Gijsbers D.A.H.: GBNP, a Non–commutative Gröbner Bases Package for GAP 4 (2003) Available from `http://www.win.tue.nl/~amc/pub/grobner/`.
34. Apel, J. and Klaus, U.: FELIX, a Special Computer Algebra System for the Computation in Commutative and Non-commutative Rings and Modules (1998) Available from `http://felix.hgb-leipzig.de/`.
35. Kredel, H. and Pesch, M.: MAS, Modula-2 Algebra System (1998) Available from `http://krum.rz.uni-mannheim.de/mas.html`.

36. Chyzak, F.: The GROEBNER Package for MAPLE (2003) Available from http://algo.inria.fr/libraries/.
37. Pearce, R.: The F4 Algorithm for Gröbner Bases, Package for MAPLE (2005) Available from http://www.cecm.sfu.ca/~rpearcea/.
38. Takayama, N.: KAN/SM1, a Gröbner engine for the ring of differential and difference operators (2003) Available from http://www.math.kobe-u.ac.jp/KAN/index.html.
39. Grayson, D. and Stillman, M.: MACAULAY 2, a Software System for Research in Algebraic Geometry (2005) Available from http://www.math.uiuc.edu/Macaulay2/.

Papers published in the Reports on Computer Algebra series

[RCA:36] V. Levandovskyy. Plural, a Noncommutative Extension of Singular: Past, Present and Future. June 2006.

[RCA:35] M. Brickenstein. Slimgb: Grbner Bases with Slim Polynomials. September 2005.

[RCA:34] E.A. Tobis. Libraries for Counting Real Roots. July 2005.

[RCA:33] A. Frühbis-Krüger, G. Pfister. Practical Aspects of Algorithmic Resolution of Singularities. October 2004.

[RCA:32] A. Frühbis-Krüger, K. Krüger, and H. Schönemann. Dynamic Modules in SINGULAR. December 2003.

[RCA:31] G.-M. Greuel, C. Lossen, and M. Schulze. Three Algorithms in Algebraic Geometry, Coding Theory, and Singularity Theory. March 2001.

[RCA:30] G.-M. Greuel. Applications of Computer Algebra to Algebraic Geometry, Singularity Theory and Symbolic-Numerical Solving. June 2000.

[RCA:29] G.-M. Greuel. Computer Algebra and Algebraic Geometry – Achievements and Perspectives. Feb 2000.

[RCA:28] T. Siebert. Recursive Computation of Free Resolutions and a Generalized Koszul Complex. Jan 2000.

[RCA:27] O. Bachmann and H.-G. Gräbe. The SymbolicData Project: Towards an Electronic Repository of Tools and Data for Benchmarks of Computer Algebra Software. Jan 2000.

[RCA:26] B. Reinert. Solving One-Sided Equations in Integer Monoid Rings. Jan 2000.

[RCA:25] B. Reinert and D. Zeckzer. Coset Enumeration using Prefix Gröbner Bases in Mrc – An Experimental Approach. Nov 1999.

[RCA:23] Birgit Reinert. Observations on coset enumeration. Nov 1998.

[RCA:22] K. Madlener and F. Otto. Some Applications Of Prefix-Rewriting In Monoids, Groups, And Rings. November 1998.

[RCA:21] G.-M. Greuel, G. Pfister, and H. Schönemann. Singular version 1.2 user manual. June 1998.

[RCA:20] B. Reinert and D. Zeckzer. MRC – A System for Computing Gröbner Bases in Monoid and Group Rings. July 1998.

[RCA:19] B. Reinert, K. Madlener, and T. Mora. A note on nielsen reduction and coset enumeration. February 1998.