Non–commutative Computer Algebra for polynomial algebras: Gröbner bases, applications and implementation

Viktor Levandovskyy

Vom Fachbereich Mathematik
der Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Naturwissenschaften
(Doctor rerum naturalium, Dr. rer. nat)
genehmigte Dissertation

1. Gutachter: Prof. Dr. G.-M. Greuel
2. Gutachter: Prof. Dr. Yu. Drozd

Vollzug der Promotion: 08.06.2005

D 386

Viktor Levandovskyy. *"Non-commutative Computer Algebra for polynomial algebras: Gröbner bases, applications and implementation".*

Abstract. Non-commutative polynomial algebras appear in a wide range of applications, from quantum groups and theoretical physics to linear differential and difference equations.

In the thesis, we have developed a framework, unifying many important algebras in the classes of G– and GR–algebras and studied their ring–theoretic properties. Let A be a G–algebra in n variables. We establish necessary and sufficient conditions for A to have a Poincaré–Birkhoff–Witt (PBW) basis. Further on, we show that besides the existence of a PBW basis, A shares some other properties with the commutative polynomial ring $\mathbb{K}[x_1, \ldots, x_n]$. In particular, A is a Noetherian integral domain of Gel'fand–Kirillov dimension n. Both Krull and global homological dimension of A are bounded by n; we provide examples of G–algebras where these inequalities are strict. Finally, we prove that A is Auslander–regular and a Cohen–Macaulay algebra.

In order to perform symbolic computations with modules over GR–algebras, we generalize Gröbner bases theory, develop and respectively enhance new and existing algorithms. We unite the most fundamental algorithms in a suite of applications, called "Gröbner basics" in the literature. Furthermore, we discuss algorithms appearing in the non-commutative case only, among others two–sided Gröbner bases for bimodules, annihilators of left modules and operations with opposite algebras.

An important role in Representation Theory is played by various subalgebras, like the center and the Gel'fand–Zetlin subalgebra. We discuss their properties and their relations to Gröbner bases, and briefly comment some aspects of their computation. We proceed with these subalgebras in the chapter devoted to the algorithmic study of morphisms between GR–algebras. We provide new results and algorithms for computing the preimage of a left ideal under a morphism of GR–algebras and show both merits and limitations of several methods that we propose. We use this technique for the computation of the kernel of a morphism, decomposition of a module into central characters and algebraic dependence of pairwise commuting elements. We give an algorithm for computing the set of one–dimensional representations of a G–algebra A, and prove, moreover, that if the set of finite dimensional representations of A over a ground field $\mathbb{K}$ is not empty, then the homological dimension of A equals n.

All the algorithms are implemented in a kernel extension PLURAL of the computer algebra system SINGULAR. We discuss the efficiency of computations and provide a comparison with other computer algebra systems. We propose a collection of benchmarks for testing the performance of algorithms; the comparison of timings shows that our implementation outperforms all of the modern systems with the combination of both broad functionality and fast implementation.

In the thesis, there are many new non-trivial examples, and also the solutions to various problems, arising in different fields of mathematics. All of them were obtained with the developed theory and the implementation in PLURAL, most of them are treated computationally in this thesis for the first time.

Viktor Levandovskyy. *"Nichtkommutative Computeralgebra für Polynomalgebren: Gröbnerbasen, Anwendungen und Implementierung".*

Zusammenfassung. Nichtkommutative Polynomalgebren entstehen in vielen verschiedenen Anwendungen, von Quantengruppen und Theoretischer Physik bis zu linearen Differentiellen und Differenzengleichungen.

In der Arbeit wurde ein Rahmen entwickelt, in dem viele wichtige Algebren der Klassen G– und GR–Algebren zusammengeführt und ihre ringtheoretischen Eigenschaften untersucht wurden. Sei A eine G–Algebra mit n Variablen. Es werden notwendige und hinreichende Bedingungen dafür angegeben, daß A eine Poincaré–Birkhoff–Witt (PBW) Basis besitzt. Es wird gezeigt, dass A neben der Existenz einer PBW Basis, weitere Eigenschaften mit dem kommutativen Polynomring $\mathbb{K}[x_1, \ldots, x_n]$ gemeinsam hat. A ist ein Noetherscher Integritätsbereich der Gel'fand–Kirillov–Dimension n. Die Krull– und die globale homologische Dimension von A sind durch n beschränkt ; es werden Beispiele von G–Algebren gegeben, bei denen diese Ungleichheiten strikt sind. Schließlich wurde bewiesen, dass A eine Auslander–reguläre und eine Cohen–Macaulay Algebra ist.

Für symbolische Berechnungen mit Moduln über GR–Algebren, wurde die Gröbnerbasentheorie verallgemeinert, neue und bestehende Algorithmen werden entwickelt und verbessert. Wir verbinden die grundlegendsten Algorithmen mit einer Reihe von Anwendungen, welche man in der Literatur als "Gröbner basics" bezeichnet. Weiterhin wurden Algorithmen

diskutiert, die nur im nichtkommutativen Fall existieren, darunter zweiseitige Gröbnerbasen für Bimoduln, Annihilatoren von Linksmoduln und Operationen mit entgegengesetzten Algebren. Eine wichtige Rolle in der Darstellungstheorie spielen die verschiedenen Unteralgebren, wie z.B. das Zentrum und die Gel'fand–Zetlin Unteralgebra. Die Eigenschaften und ihre Beziehungen zu Gröbnerbasen wurden untersucht und einige Aspekte ihrer Berechnung diskutiert. Im Kapitel über algorithmische Studien von Morphismen zwischen GR–Algebren wurde die Untersuchung zu diesen Unteralgebren fortgesetzt. Es wurden neue Ergebnisse and Algorithmen zur Berechnung des Urbilds eines Linksideals unter einem Morphismus von GR–Algebren erzielt. Die Ergebnisse und Begrenzungen verschiedener Methoden, die vorgeschlagen wurden, wurden gezeigt. Diese Technik wurde auch für die Berechnung des Kerns eines Morphismus, die Zerlegung eines Moduls in zentrale Charaktere und die algebraische Abhängigkeit von paarweise kommutierenden Elementen verwendet. Es wurde ein Algorithmus für die Berechnung von eindimensionalen Darstellungen einer G–Algebra A erstellt. Es wurde bewiesen, dass die homologische Dimension von A über einem Körper $\mathbb{K}$ gleich der Anzahl der Variablen von A ist, falls es endlich–dimensionale Darstellungen von A existieren.

Alle Algorithmen wurden in eine Kern-Erweiterung Plural des Computeralgebrasystem Singular implementiert. Die Effizienz der Berechnungen wurde diskutiert und ein Vergleich mit anderen Computeralgebrasystemen erstellt. Es wurde eine Reihe von Benchmarks zum Test der Leistung von Algorithmen vorgeschlagen. Der Zeitvergleich zeigt, dass unsere Implementierung alle modernen Systeme hinsichtlich der Kombination von Funktionalität und Geschwindigkeit übertrifft.

In der Arbeit gibt es eine Vielzahl von nichttrivialen Beispielen und Lösungen zu verschiedenen Problemen aus verschiedensten Bereichen der Mathematik. All wurden mit Hilfe der entwickelten Theorie und der Implementation in Plural erzielt, die meisten von ihnen wurden in dieser Arbeit zum ersten Mal berechnet.

Contents

Introduction

This thesis is devoted to symbolic computations in non–commutative algebras with PBW bases (G–algebras). These algebras already have their own history as well as symbolic methods, algorithms and implementations. Yet it is still a long way towards the acceptance and wide usage of these methods by the community of scientists, comparable to the years which were gone before the Buchberger's algorithm for computing a Gröbner basis of an ideal in a commutative ring become accepted by everybody. G–algebras appear in many fields of science, from Non–commutative Algebra, Ring Theory and Representation Theory of Algebras (being of a more theoretical nature) to the concrete applications to manipulating systems of linear operator (differential, shift, difference etc) equations, System and Control Theory et cetera. These algebras arise in Algebraic Geometry, Mathematical and Theoretical Physics, Statistics and many other fields of natural sciences. They appear in different contexts and everywhere some different computations are needed.

We revise G–algebras and enlist a list of their properties in Chapter 1; a short overview of Gröbner bases in free algebras helps us to formulate the exact conditions and, on the other hand, to see similarities and differences between different setups. We point out the role of monomial orderings and show their impact on filtrations, coming to simplified and/or generalized proofs of many known results.

In Chapter 2, we adopt the notion of Gröbner basis to G–algebras and note similarities and differences with the commutative case. Further on, we concentrate on developing the fundament for applications, based on Gröbner bases; continuing a tradition, we call the set of most ubiquitous applications *Gröbner basics*. We provide both theoretical background and efficiency [1] issues together with the implementation. A rich collection of examples, mostly originated from concrete research problems, is a key point of this thesis, since previous publications and even books were rather ascetic with respect to examples.

[1] All Gröbner basis computations are not efficient as the complexity is exponential or even double exponential in the number of variables. When we talk about efficiency, we mean "practical efficiency", that is, implementations which allow to compute interesting and complicated examples in a reasonable time.

The morphisms between GR–algebras (the Chapter 3), appearing as often as such morphisms do in the commutative algebra, deserved however less attention from other authors. We proceed with the partially commutative and purely non–commutative situations separately, describing very interesting applications in much details. A study of subalgebras such as centers and various centralizers is naturally arising in this Chapter.

The implementation of all the algorithms, elaborated in previous sections, in a computer algebra subsystem SINGULAR:PLURAL comprises the material of the Chapter 4. We provide timings, create and evaluate benchmarks and discuss aspects of efficiency and software engineering applied to computer algebra. All the examples, provided with this thesis, are computed with PLURAL. It is freely distributed as an integral part of SINGULAR, starting from the version 3-0-0. One can download binaries of SINGULAR for various platforms, the documentation in several formats and supplementary files from the `http://www.singular.uni-kl.de` (note, that the source code is available on request).

The atlas of important algebras, put in Chapter 5, contains both the explicit presentation of algebras we use in examples and applications, and structural properties like centers of every algebra. Everything in the atlas has been computed with the help of PLURAL.

Every Chapter starts with an explanation on its organization, so we omit such descriptions here.

In the Appendix we put the part of the SINGULAR manual, organized as Chapter 6 and devoted to PLURAL with the detailed description of its data types, functionalities and libraries.

Acknowledgments

I am deeply grateful to my advisors Prof. Dr. Yu. Drozd and Prof. Dr. G.-M. Greuel for supporting me continuously with advises, remarks and suggestions through the whole working period on this thesis. Many interesting topics of the thesis would not have evolved without the influence, bride scope of interests and critics of my advisors. I owe very much to the careful reading of this manuscript by Gert–Martin Greuel and the valuable corrections he made.

Dr. H. Schönemann deserves a very special cordial thanks. Together with him we have created a computer algebra subsystem PLURAL; we have developed it from the very experimental beginning to the integral part of the SINGULAR kernel. On the way I have learned, what does "effective computation" really mean in practice.

I want to thank Dr. O. Khomenko (Freiburg) for fruitful discussions through the years concerning different topics of computational aspects of the representation theory and non–commutative algebra, for hints and critics that contributed greatly to this work.

Moreover, I am especially grateful to Prof. Dr. E. Green (Virginia, USA), Prof. Dr. G. Pfister (Kaiserslautern), Prof. Dr. V. Gerdt (Dubna, Russia) and Dr. S. Ovsienko (Kyiv, Ukraine) for waking the interest on various sections of algebra and inspirations by techniques and examples. The new shores of applications became visible in discussions with Werner Seiler, Eva Zerz, Jose–Maria Ucha and Javier Lobillo; the demand for Non–commutative Computer Algebra in Algebraic Geometry was a dominant topic in communications with Wolfram Decker, Frank–Olaf Schreyer and Christoph Lossen.

I wish to express my appreciation to my wife Tetyana and families of our parents for their love, help and patience. It is hard to imagine how could this work be finished without them.

I would like to thank DFG (Deutsche Forschungsgemeinschaft), Schwerpunkt "Mathematik und Praxis" of the University of Kaiserslautern and CRDF (U.S. Civilian Research and Development Foundation) for providing the partial financial support for my research.

Basic Notations

$\mathbb{K}, \mathbb{F}, \mathbb{F}_p, \mathbb{Q}, \mathbb{R}, \mathbb{C}$	fields
$\mathbb{K}^*$	the multiplicative group of units of a field $\mathbb{K}$
$\bar{x} = (x_1, \ldots, x_n)$	n–tuple or a vector
$\mathbb{K}[S]$, $\mathbb{K}[x_1, \ldots, x_n]$	a commutative polynomial ring, generated by a finite set S or by $\{x_1, \ldots, x_n\}$
$\mathbb{K}\langle S\rangle$, $\mathbb{K}\langle x_1, \ldots, x_n\rangle$	a free associative $\mathbb{K}$–algebra, generated by a finite set S or by $\{x_1, \ldots, x_n\}$
${}_A\langle S\rangle$	a left A–module, generated by S
${}_A\langle S\rangle_B$	a (A, B)–bimodule, generated by S
${}_A\langle S\rangle_A$, $\langle S\rangle$	a two–sided ideal or (A, A)–bimodule, generated by S
$\mathbb{K}\langle S \mid R\rangle = \mathbb{K}\langle S\rangle/\langle R\rangle$	presentation of a $\mathbb{K}$–algebra via S, a set of generators and $R \subset \mathbb{K}\langle S\rangle$, a set of "relations"

CHAPTER 1

From NDC Towards G–algebras

> In his right hand there still had been
> his broad, strange sword, so unusual for
> the eyes of western warriors ...
> "You've got an interesting sword
> there", – she said slowly, – "What are
> these *rings* at the blade for?"
> "It's so funny when they're ringing", –
> answered the warrior.
>
> Nik Perumov, *Henna's Adamant*

The famous Poincaré–Birkhoff–Witt (or, shortly, PBW) theorem, which appeared at first for universal enveloping algebras of finite dimensional Lie algebras ([**23**]), plays an important role in the representation theory as well as in the theory of rings and algebras. Analogous theorem for quantum groups was proved by G. Lusztig and constructively by C. M. Ringel ([**67**]).

Many authors have proved the PBW theorem for special classes of non–commutative algebras they are dealing with ([**40**], [**38**]). Usually one uses Bergman's Diamond Lemma 1.9 (see also [**11**]), although it needs some preparations to be done before applying it. We have defined a class of algebras where the question "Does this algebra have a PBW basis?" reduces to a direct computation involving only basic polynomial arithmetic.

This chapter is organized as follows.

Our approach is constructive and consists of three tasks. Firstly, we want to find the necessary and sufficient conditions for a wide class of algebras to have a PBW basis, secondly, we are going to investigate this class for ring–theoretical and other properties and thirdly, we will apply the results to the study of certain special types of algebras.

The first part resulted in the non–degeneracy conditions (Theorem 2.3), the second one led us to the G– and GR–algebras (3.2, 3.7) and their properties (Theorem 4.7, 4.14), and the third one — to the technique of computing G–quantizations (7.1) and to the structural studying of algebras (7.1, 7.3) together with the description and classification of G–algebras among the quadratic (7.2), diffusion algebras (7.4) and some more advanced applications (7.5).

We have simplified many proofs of known results and unified different notations. We are additionally motivated by the fact, that up to our knowledge, no source before featured a complete treatment of the problems, arising in connection with PBW bases.

1. Gröbner Bases on Free Associative Algebras

Let $\mathbb{K}$ be a field and $T = T_n = \mathbb{K}\langle x_1, \ldots, x_n\rangle$ be a free associative $\mathbb{K}$–algebra, generated by $\{x_1, \ldots, x_n\}$ over $\mathbb{K}$, also called a **tensor algebra** $T(V)$ of the vector space $V = \mathbb{K} \oplus \mathbb{K}x_1 \oplus \cdots \oplus x_n$. We will omit the tensor product sign while writing multiplication and we will mean by an ideal a two–sided ideal, whenever no confusion is possible.

We say that the **monomials** in T are the elements from the set of all words in $\{x_1, \ldots, x_n\}$,

$$\mathrm{Mon}(T) = \{x_{i_1}^{\alpha_1} x_{i_2}^{\alpha_2} \ldots x_{i_m}^{\alpha_m} \mid 1 \leq i_1, i_2, \ldots, i_m \leq n,\ \alpha_k \geq 0\}.$$

Note, that $\mathrm{Mon}(T)$ is a $\mathbb{K}$–basis of T. Moreover, $\mathrm{Mon}(T)$ is a free monoid with the neutral element 1.

The set of **standard monomials** which we will need later is defined as

$$\mathrm{Mon}_S(T) = \{x_{i_1}^{\alpha_1} x_{i_2}^{\alpha_2} \ldots x_{i_m}^{\alpha_m} \mid 1 \leq i_1 < i_2 < \ldots < i_m \leq n,\ \alpha_k \geq 0\} \subset \mathrm{Mon}(T).$$

In what follows, we always assume every associative algebra A to be finitely generated and unital with $\mathbb{K} \subset A$. Moreover, we assume that $\mathbb{K}$ always belongs to the center of A (that is, $\forall k \in \mathbb{K}, a \in A$ we have $ka = ak$) and that any morphism of $\mathbb{K}$–algebras is the identity on $\mathbb{K}$.

Any associative $\mathbb{K}$–algebra is isomorphic to T_n/I for some n and some two–sided ideal $I \subset T_n$. Since T_n is not Noetherian, I need not be finitely generated, but we are interested only in ideals, which are finitely generated. If a fixed isomorphism $A \cong T_n/I$ is given, we say that A is finitely presented by T_n and write $A = T_n/I$ or just $A = T/I$.

If the set of standard monomials is a $\mathbb{K}$–basis of an algebra $A = T/I$, we say that A has a PBW basis (in the variables $x_1, \ldots, x_n$). We say that an abstract algebra A **has a PBW basis**, if there exists an isomorphism $A \cong T/I$, such that T/I has a PBW basis. Of course, any commutative polynomial ring $\mathbb{K}[x_1, \ldots, x_n]$ has a PBW basis. Therefore, algebras which are Noetherian domains with PBW basis are in this sense "close to commutative".

Now we will present the short account of the Gröbner bases theory on tensor algebras. It was first Teo Mora, who considered a unified Gröbner bases framework for commutative and non–commutative algebras ([**61**]),

which has been recently exploited by Li in his book [**56**]. We follow this approach partially and write in the spirit of the book [**35**].

DEFINITION 1.1. Let Γ be a finitely generated monoid.

A total ordering $\prec$ on Γ is called a **well–ordering**, if every non–empty subset of Γ has a least element with respect to $\prec$.

A well–ordering $\prec$ on Γ is called **finitely supported**, if

$$\forall a \in \Gamma \text{ there exist finitely many } b \in \Gamma \text{ such that } b \prec a.$$

DEFINITION 1.2. We call a total ordering $<$ on $\mathrm{Mon}(T)$ a **monomial ordering** if the following conditions hold:

(1) $<$ is a well–ordering on $\mathrm{Mon}(T)$,
(2) $\forall p, q, s, t \in \mathrm{Mon}(T)$, if $s < t$, then $p \cdot s \cdot q < p \cdot t \cdot q$,
(3) $\forall p, q, s, t \in \mathrm{Mon}(T)$, if $s = p \cdot t \cdot q$ and $s \neq t$, then $t < s$.

In this work we are dealing with well–orderings only. As an example of a well–ordering, consider the lexicographical ordering on $\mathbb{K}\langle x_1, \ldots, x_n\rangle$, considering that the variables are ordered in a descending way, that is

$x_n < x_{n-1} < \cdots < x_2 < x_1$. The lexicographical ordering is defined as follows: given two monomials (that is, words in finite alphabet $\{x_1, \ldots, x_n\}$) m_1, m_2 from $\mathrm{Mon}(T)$, we find their biggest common left subword m such that $m_1 = mw_1, m_2 = mw_2$ or set $m = 1$ if no such subword exists. Then, $m_1 < m_2 \Longleftrightarrow w_1 < w_2 \Longleftrightarrow$ the first symbol x_i of w_1 is smaller that the first symbol x_j of $w_2 \Longleftrightarrow x_i < x_j \Longleftrightarrow j < i$.

DEFINITION 1.3. Any $f \in T \setminus \{0\}$ can be written uniquely as

$f = c \cdot m + f'$, where $c \in \mathbb{K}^*$ and $m' < m$ for any non–zero term $c' \cdot m'$ of f'.

We define

$$\begin{aligned} \mathrm{lm}(f) &= m, \quad \text{the } \textbf{leading monomial} \text{ of } f, \\ \mathrm{lc}(f) &= c, \quad \text{the } \textbf{leading coefficient} \text{ of } f. \end{aligned}$$

For a subset $G \subset T$, define a **leading ideal** of G to be the two–sided ideal $L(G) = {}_T\langle\, \{\mathrm{lm}(g) \mid g \in G \setminus \{0\}\}\, \rangle_T \subseteq T$.

DEFINITION 1.4. Let $<$ be a fixed monomial ordering on T. We say that a subset $G \subset I$ is a **(two–sided) Gröbner basis** for the ideal I with respect to $<$ if $L(G) = L(I)$.

Although we can work formally with infinite Gröbner bases ([**4**]), in this work we are interested only in finite bases.

DEFINITION 1.5. Let $m, m' \in \mathrm{Mon}(T)$ be two monomials.
We say that m **divides** m' if there exist $p, q \in \mathrm{Mon}(T)$ such that $m' = p \cdot m \cdot q$. The set $G \subseteq T$ is called **minimal**, if $\forall g_1, g_2 \in G, \quad \mathrm{lm}(g_1)$ does not divide $\mathrm{lm}(g_2)$ and vice versa.

DEFINITION 1.6. Let $\mathcal{G}$ be the set of all finite and ordered subsets of T.
A map $\mathrm{NF} : T \times \mathcal{G} \to T, \ (f, G) \mapsto \mathrm{NF}(f|G)$ is called a
(two–sided) normal form on T if

(i) $\mathrm{NF}(0 \mid G) = 0$,
(ii) $\mathrm{NF}(f|G) \neq 0 \ \Rightarrow \ \mathrm{lm}\big(\mathrm{NF}(f|G)\big) \notin L(G)$, and
(iii) $f - \mathrm{NF}(f|G) \in {}_T\langle G\rangle_T$, for all $f \in T$ and $G \in \mathcal{G}$.

Here we give an algorithm for computing a normal form.

Algorithm 1.1 NF

Input : $f \in T = \mathbb{K}\langle x_1, \ldots, x_n\rangle, \ G \in \mathcal{G}$;
Output: $h \in T$, a normal form of f with respect to G.

$h := f$;
while ($(h \neq 0)$ **and** $(G_h = \{g \in G : \ \mathrm{lm}(g) \text{ divides } \mathrm{lm}(h)\} \neq \emptyset)$) **do**
  choose any $g \in G_h$;
  compute $l = l(g), r = r(g) \in \mathrm{Mon}(T)$ such that $\mathrm{lm}(h) = l \cdot \mathrm{lm}(g) \cdot r$;
  $h := h - \dfrac{lc(h)}{lc(g)} \cdot l \cdot g \cdot r$;
end while
return h;

PROOF. We see that each specific choice of "any" in the algorithm may give us a different normal form function.

Termination:

Let $h_0 := f$, and in the i–th step of the **while** loop we compute h_i. Since $\mathrm{lm}(h_i) < \mathrm{lm}(h_{i-1})$ by construction, we obtain a set $\{\mathrm{lm}(h_i)\}$ of leading monomials of h_i, where $\forall i \ h_{i+1}$ has strictly smaller leading monomial than h_i. Since $<$ is a well–ordering, this set has a minimum, hence the algorithm terminates.

Correctness:

Suppose the minimum is reached at the step m. Let $h = h_m \neq 0$ and l_i, r_i are monomials, corresponding to $g_i \in G$ in the algorithm. Making back

substitutions, we obtain the following expression

$$h = f - \sum_{i=1}^{m-1} l_i g_i r_i,$$

satisfying $\mathrm{lm}(f) = \mathrm{lm}(l_1 g_1 r_1) > \mathrm{lm}(l_i g_i r_i) > \mathrm{lm}(h_m)$.

Moreover, by construction $\mathrm{lm}(h) \notin L(G)$. This proves correctness, independently of the specific choice of "any" in the **while** loop. □

Definition 1.7. Let $f, g \in T$. Suppose that there are $p, q \in \mathrm{Mon}(T)$ such that

(1) $\mathrm{lm}(f)q = p\,\mathrm{lm}(g)$,
(2) $\mathrm{lm}(f)$ does not divide p and $\mathrm{lm}(g)$ does not divide q.

Then the **overlap relation** of f, g by p, q is defined as

$$o(f, g, p, q) = \frac{1}{\mathrm{lc}(f)} fq - \frac{1}{\mathrm{lc}(g)} pg.$$

Overlaps occur if the end of $\mathrm{lm}(f)$ coincides with the beginning of $\mathrm{lm}(g)$, as for example in $f = xyx$ and $g = yx^2$ (both overlapping at yx) or $f = g = x^2$ (with the self–overlap at x). The overlap relation cancels the leading terms of fq and pg and we see that $\mathrm{lm}(o(f, g, p, q)) < \mathrm{lm}(f)q = p\,\mathrm{lm}(g)$. Hence, overlap relation is a generalization of the notion of s–polynomial from the commutative theory (cf. [**35**]). See also the Remark **§2**, 4.13.

Now we cite slightly reformulated *Termination theorem* from [**33**].

Theorem 1.8. Let $<$ be a well–ordering on T and G be a finite set of polynomials from T. If for every overlap relation with $g_1, g_2 \in G$

$$\mathrm{NF}(\ o(g_1, g_2, p, q)\ \mid\ G) = 0,$$

then G is a Gröbner basis for ${}_T\langle G\rangle_T$.

In particular this theorem ensures that for a given finite set F, we are able to check whether F is a Gröbner basis in a finite number of steps. However, even starting with a finite set, we can obtain, in general, an infinite Gröbner basis for it.

The proof of the theorem is relying heavily on the use of the famous Bergman's Diamond Lemma ([**11**], 1978), which can be regarded as the first building stone for the theory of non–commutative Gröbner bases.

Bergman's Diamond Lemma

Following Bergman, let $T = \mathbb{K}\langle x_1, \ldots, x_n\rangle$, $X = \mathrm{Mon}(T)$ and $<$ is a well–ordering on X. Let $S = \{\sigma = (w_\sigma, f_\sigma) \mid\ w_\sigma \in X, f_\sigma \in T,\ \mathrm{lm}(f_\sigma) <$

$w_\sigma\}$. For all $\sigma \in S$ and $x, y \in X$ let $r_{x\sigma y} : T \to T$ denote the $\mathbb{K}$–linear map, defined on the basis $\mathrm{Mon}(T)$ by sending a monomial of the form $pxw_\sigma yq$ to a polynomial $pxf_\sigma yq$ for any $p, q \in \mathrm{Mon}(T)$ and fixing all other monomials, not containing $xw_\sigma y$.

Let R denote the semigroup generated by the $\{r_{x\sigma y} \mid \sigma \in S, x, y \in X\}$. We call $x \in X$ *reduced* if $r(x) = x$ for all $r \in R$. An *ambiguity* of S is a 5–tuple $\{\sigma, \tau; x, y, z\} \in S^2 \times X^3$ for which $w_\sigma = xy$, $w_\tau = yz$. Such an ambiguity is said to be *resolvable* if there exists $r \in R$ such that $r(f_\sigma z) = r(xf_\tau)$. The following Theorem is a short version of Bergman's Diamond lemma.

THEOREM 1.9. ([**11**], Theorem 1.2.) The reduced elements form a $\mathbb{K}$–basis for the quotient algebra $T/\,{}_T\langle w_\sigma - f_\sigma \mid \sigma \in S\rangle_T$ if and only if every ambiguity of S is resolvable.

2. Non–degeneracy Conditions and PBW Theorem

Again, let $T = \mathbb{K}\langle x_1, \ldots, x_n\rangle$ be a free associative algebra and $<$ be a fixed monomial ordering on T. For fixed n, define the set of indices $\mathcal{U}_m := \{(i_1, \ldots, i_m) \mid 1 \le i_1 < \ldots < i_m \le n\}$.

Suppose there are two sets $C = \{c_{ij}\} \subset \mathbb{K}^*$ and $D = \{d_{ij}\} \subset T$, where $(i, j) \in \mathcal{U}_2$. We construct a set $F = \{f_{ji} \mid (i, j) \in \mathcal{U}_2\}$, where

$$(*) \qquad \begin{cases} f_{ji} = x_j x_i - c_{ij} \cdot x_i x_j - d_{ij}, \\ \mathrm{lm}(f_{ji}) = x_j x_i \text{ and } \mathrm{lm}(d_{ij}) < x_i x_j. \end{cases}$$

It means that, in particular, polynomials d_{ij} can involve terms $x_j x_i$ for $j > i$, that is nonstandard monomials. But this situation can be always reduced to the easier case by the following simplification procedure.

Simplification

Assume that there is a fixed well–ordering $<$ with $x_n < \ldots < x_1$, then $x_{n-1}x_n < x_{n-2}x_n < x_{n-2}x_{n-1} < \ldots < x_1x_2$. Hence, $d_{n-1,n}$ consists of standard monomials only. Next, $d_{n-2,n}$ can have only $x_n x_{n-1}$ in addition to standard monomials, but we replace $x_n x_{n-1}$ with $c_{n-1,n}x_{n-1}x_n + d_{n-1,n}$ thus obtaining $d'_{n-2,n}$, consisting of standard monomials only. Since $\forall\ (i, j) \in \mathcal{U}_2$ $\mathrm{lm}(f_{ji}) = x_j x_i$ and $\mathrm{lm}(d_{ij}) < x_i x_j$, after finitely many steps we obtain a set $F := F'$, where each of the d'_{ij} is given in terms of standard monomials. That is, we can assume without loss of generality, that every d_{ij} consists of standard monomials only.

After the simplification of the set F by means of the procedure above, we construct the two–sided ideal $I = {}_T\langle F\rangle_T \subset T$.

For $(i,j,k) \in \mathcal{U}_3$ define the **non–degeneracy condition** for (i,j,k)

$$\mathcal{NDC}_{ijk} = c_{ik}c_{jk}\cdot d_{ij}x_k - x_k d_{ij} + c_{jk}\cdot x_j d_{ik} - c_{ij}\cdot d_{ik}x_j + d_{jk}x_i - c_{ij}c_{ik}\cdot x_i d_{jk}.$$

LEMMA 2.1. *F is a Gröbner basis for I with respect to $<$ if and only if*

$$\forall\ 1 \le i < j < k \le n \quad \mathrm{NF}(\mathcal{NDC}_{ijk} \mid F) = 0.$$

PROOF. We will compute Gröbner basis of I symbolically, but as explicitly as we can. Following the theorem 1.8, we have to consider all the possible overlaps of elements from F. It's straightforward, that the only nonzero overlaps can occur for the set of pairs $\{(f_{ji}\ ,\ f_{kj}) \mid (i,j,k) \in \mathcal{U}_3\}$. Computing the overlap relation of $(f_{ji}\ ,\ f_{kj})$ for fixed $(i,j,k) \in \mathcal{U}_3$, we get

$$o_1 = x_k x_j x_i - c_{ij}x_k x_i x_j - x_k d_{ij} - x_k x_j x_i + c_{jk}x_j x_i + d_{jk}x_i =$$

$$= -c_{ij}x_k x_i x_j + c_{jk}x_j x_k x_i - x_k d_{ij} + d_{jk}x_i.$$

The o_1 can be reduced with f_{kj} to

$$o_2 = c_{jk}x_j x_k x_i - c_{ij}c_{ik}x_i x_k x_j - c_{ij}d_{ik}x_j - x_k d_{ij} + d_{jk}x_i,$$

where o_2 could be further reduced with f_{ki} to

$$o_3 = c_{jk}c_{ik}x_j x_i x_k - c_{ij}c_{ik}x_i x_k x_j - c_{ij}d_{ik}x_j - x_k d_{ij} + d_{jk}x_i + c_{jk}x_j d_{ik}.$$

On its own, we reduce o_3 with f_{ji} to $o_4 =$

$$-c_{ij}c_{ik}x_i x_k x_j + c_{jk}c_{ik}c_{ij}x_i x_j x_k + c_{jk}c_{ik}d_{ij}x_k - c_{ij}d_{ik}x_j - x_k d_{ij} + d_{jk}x_i + c_{jk}x_j d_{ik},$$

and, respectively, f_{kj} finishes the reduction of o_4:

$$o_5 = c_{jk}c_{ik}d_{ij}x_k - x_k d_{ij} + c_{jk}x_j d_{ik} - c_{ij}d_{ik}x_j + d_{jk}x_i - c_{ij}c_{jk}x_i d_{jk}.$$

As we see, $o_5 = \mathcal{NDC}_{ijk}$, and o_5 cannot be further reduced with the elements of F without the more specific information on $\{d_{ij}\}$.
If $\mathrm{NF}(\mathcal{NDC}_{ijk} \mid F) \neq 0$, F is not a Gröbner basis of I. Hence the claim. □

LEMMA 2.2. *Given a $\mathbb{K}$–algebra $A = T/I$ with $I = {}_T\langle F\rangle_T$, F satisfying $(*)$, as above. Then A has a PBW basis with respect to the variables of T if and only if F is a Gröbner basis for I with respect to $<$.*

PROOF. If F is a Gröbner basis for I with respect to $<$, the underlying $\mathbb{K}$–vector space of A is generated by $\{m \in \mathrm{Mon}(T) \mid \mathrm{lm}(f_{ji}) \text{ does not divide } m\}$ by the property of Gröbner bases ([**34**]). We see immediately that this vector space is the set of standard monomials, since no standard monomial is divisible by $\mathrm{lm}(f_{ji})\ \forall j > i$.

Conversely, assume $A = T/I$ has a PBW basis. Then we can interpret it as a $\mathbb{K}$–algebra, generated by $x_1, \ldots, x_n$ with the multiplication

$$(\star) \quad \forall 1 \leq i,j \leq n \qquad x_j \star x_i = \begin{cases} x_j x_i, & \text{if } i \geq j, \\ c_{ij} \cdot x_i x_j + d_{ij}(x), & \text{if } i < j. \end{cases}$$

Since A is an associative algebra, we expect that the multiplication $\star$ is well–defined, in particular, $(x_k \star x_j) \star x_i - x_k \star (x_j \star x_i) = 0 \ \forall (i,j,k)$. It is easy to see that this holds trivially for all the cases except that when $(i,j,k) \in \mathcal{U}_3$, which we analyze. A bit lengthy technical computation similar to the one of Lemma 2.1 in this case delivers

$$\begin{aligned} &(x_k \star x_j) \star x_i - x_k \star (x_j \star x_i) = \\ &= c_{ik}c_{jk} \cdot d_{ij}x_k - x_k d_{ij} + c_{jk} \cdot x_j d_{ik} - c_{ij} \cdot d_{ik}x_j + d_{jk}x_i - c_{ij}c_{ik} \cdot x_i d_{jk} = \\ &= \mathcal{NDC}_{ijk}. \end{aligned}$$

So, $\mathcal{NDC}_{ijk}$ are identically zero in A. Hence $NF(\mathcal{NDC}_{ijk}|I) = 0$ in T and, by the Lemma 2.1, F is a Gröbner basis of I. □

We formalize the lemmata in the following:

THEOREM 2.3. Suppose there is a set
$F = \{f_{ji} \mid 1 \leq i < j \leq n\} \subset T = \mathbb{K}\langle x_1, \ldots, x_n \rangle$, where

$$\forall j > i \quad f_{ji} = x_j x_i - c_{ij} \cdot x_i x_j - d_{ij}, \ c_{ij} \in \mathbb{K}^*, \ d_{ij} \in T.$$

Define the ideal $I = {}_T\langle F \rangle_T \subset T$. Assume that there exists a well–ordering $<$ on T, such that $\mathrm{lm}(f_{ji}) = x_j x_i$ and $\mathrm{lm}(d_{ij}) < x_i x_j$, then the following conditions are equivalent:

1) F is a Gröbner basis for I with respect to $<$,
2) $\forall\ 1 \leq i < j < k \leq n \quad \mathrm{NF}(\mathcal{NDC}_{ijk} \mid F) = 0$,
3) The $\mathbb{K}$–algebra $A = T/I$ has a Poincaré–Birkhoff–Witt basis w.r.t. $x_1, \ldots, x_n$.

REMARK 2.4. Some historical remarks can be found under Remark 3.8.

(1) If we assume that $\forall\ i < j\ c_{ij} = 1$ and d_{ij} are linear polynomials, $\mathcal{NDC}_{ijk}$ becomes a famous Jacobi identity, written in the universal enveloping algebra of a finite dimensional Lie algebra. So, non–degeneracy conditions are generalized Jacobi identities and the Theorem above is clearly a generalization of the PBW Theorem ([**23**]).
(2) At first, the non–degeneracy conditions were written explicitly by Teo Mora ([**60**]) but weren't investigated further there and remained unnoticed by the community for a long time (e.g. R. Berger did not cite Mora's work in his articles [**9**], [**10**]). We reported on

non–degeneracy conditions already in ([**48**]), having found them independently and used them for structural analysis of algebras.

(3) As for recent textbooks, Li ([**56**]) followed both Mora and Berger. Bueso et al. in [**14**] wrote a whole chapter, where they applied Diamond Lemma directly to relations like in the set F above, which resulted in the equivalence of the condition $x_k(x_jx_i) = (x_kx_j)x_i$ with the PBW property of an algebra. These conditions were not developed further and were not commented. The relation of Diamond Lemma to the non–commutative Gröbner bases was not mentioned.

(4) The equivalence 1) $\Leftrightarrow$ 3) with several restrictions appeared already in [**41**], [**61**]. E. Green in [**33**] (Th. 2.14) has proved it under an assumption that d_{ij} are homogeneous quadratic polynomials.

(5) From the proof of the Lemma 2.2 we extract another characterization of PBW property, particularly simple and especially useful for computer algebra systems. Assume that the multiplication $\star$ (from the Lemma) is implemented on $A = T/I$ and $\mathrm{lm}(d_{ij}) < x_ix_j$. Then we can say whether A has a PBW basis w.r.t. $x_1, \ldots, x_n$ by directly checking, that

$$\forall\ 1 \leq i < j < k \leq n\ \ (x_k \star x_j) \star x_i - x_k \star (x_j \star x_i) = 0.$$

Some people before used this "formal associativity" for extracting a kind of non–degeneracy conditions for special algebras (among others, [**40**]).

2.1. Types of Degeneracy. What happens if we are dealing with an algebra, where non–degeneracy conditions do not vanish?

Consider an algebra A, resembling the universal enveloping algebra of a finite dimensional Lie algebra — that is, for all $i < j$ $c_{ij} = 1$ and d_{ij} are linear in generators x_k. Suppose that non–degeneracy conditions do not vanish. Then, there is no Lie algebra $\mathfrak{g}$, such that $A = U(\mathfrak{g})$, since otherwise the Jacobi identity would not hold in $\mathfrak{g}$.

Let $I \subset T$ and $A = T/I$ be as in Theorem 2.3. In general, if the non–degeneracy conditions in the algebra A do not vanish, then I is given not in its Gröbner basis in T. Speaking in the language of generators and relations, we observe the following phenomenon — there are more relations than only those of the type $(\star)$, and these *hidden relations* consist of standard monomials (which total degree do not exceed 3 if $<$ is a degree ordering and $\deg(x_i) = 1$). In some cases it is possible to *remove degeneracy*, thus passing to an isomorphic algebra with less variables.

DEFINITION 2.5. We say that a degenerate algebra $A = T_n/I$ in n generators has *removable degeneracy* if it is isomorphic to another algebra $B = T_k/J$, which is generated by $k < n$ variables $\{x_1, \ldots, x_k\}$ and has PBW basis in these variables (we can also say we are dealing with *surplus* variables or even with *overdeterminacy*).

We say, that A degenerates to B in this case.

Algebras of the type $A = T/I$ with I as in Theorem 2.3, arise from various constructions in mathematics and physics.

EXAMPLE 2.6. Consider the algebra $\mathbb{K}(q_1, q_2)\langle x, y, z\rangle$ with the relations

$$yx = q_2xy + x, \quad zx = q_1xz + z, \quad zy = yz.$$

Here we see that there is non–degeneracy condition, which translates to $((q_2 - 1)y + 1)z = 0$. So, we have found the hidden defining relation in the algebra, since the Gröbner basis of the ideal $\langle yx - q_2xy - x, zx - q_1xz - z, zy - yz\rangle \subset \mathbb{K}(q_1, q_2)\langle x, y, z\rangle$ with respect to, say, degree reverse lexicographical ordering, is

$$G = \{yx - q_2xy - x, zx - q_1xz - z, (q_2 - 1)zy - z, (q_2 - 1)yz - z\}.$$

As we see, $\mathbb{K}(q_1, q_2)\langle x, y, z\rangle/\langle G\rangle$ has a canonical subalgebra, isomorphic to $\mathbb{K}[a, b]/\langle ab\rangle$, hence it has non–removable degeneracy. In particular, it has no PBW basis and there are zero divisors.

If we specialize q_2 at 1, we obtain $G' = \{yx - xy - x, z\}$. Hence, the algebra $\mathbb{K}(q_1)\langle x, y, z\rangle/\langle G'\rangle$ degenerates to the algebra $\mathbb{K}\langle x, y\rangle/\langle yx - xy - x\rangle$, which is integral and has a basis $\{x^a y^b\}$, though a PBW basis for $\mathbb{K}\langle x, y\rangle/\langle yx - xy - x\rangle$ itself, but only a subset of the PBW basis $\{x^a y^b z^c\}$ for the algebra we were starting with, which is expected from the defining relations. The latter shows us a typical example of the algebra which degeneracy is removable.

The first known example of an algebra with relations as before but without PBW basis was given already in [**41**] (Example 1.8). In the paper it is written as $B = \mathbb{K}\langle x, y, z \mid yx = xy + x, \quad zx = xz, \quad zy = yz + z\rangle$. However, B has a PBW basis, we can check that the non–degeneracy conditions indeed vanish. We believe that there should have been a printing error in the original paper: the algebra $\mathbb{K}\langle x, y, z \mid yx = xy + x, \quad zx = xz + z, \quad zy = yz\rangle$ from the previous example looks pretty similar to B and it has removable degeneracy equal to z. Hence, it degenerates to the algebra $\mathbb{K}\langle x, y \mid yx = xy + x\rangle$.

3. Introduction to G–algebras

Now we concentrate on studying the properties of algebras, satisfying the conditions of the Theorem 2.3.

Take an algebra $A = T/I$ as in Theorem 2.3. Since it has a PBW basis, we call the elements of this basis **monomials** of A, which is, in fact, the set of standard monomials of T. This abuse of notation should not create any confusion but is very convenient.

The set of monomials of A is denoted by $\mathrm{Mon}(A)$ and, due to the existence of PBW basis can be identified with the $\mathbb{N}^n$ by

$$x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} \ \mapsto \ (\alpha_1, \alpha_2, \dots, \alpha_n) = \alpha.$$

3.1. Monomial Orderings.

Let us recall orderings on $\mathbb{N}^n$.

Perhaps the most known ordering is the **lexicographical ordering**, which we denote by `lp`: for any $\alpha, \beta \in \mathbb{N}^n$,

$\beta <_{\mathrm{lp}} \alpha \overset{\mathrm{def}}{\iff}$ the first non–zero entry of $\alpha - \beta$ is positive.

By the theorem of Robbiano ([**35**]), for any *monomial* (cf. Def. 3.1) ordering $<$ there exists a (non–unique) matrix $A \in \mathrm{GL}(n, \mathbb{R})$ such that $\beta < \alpha$ if and only if $A \cdot \beta <_{\mathrm{lp}} A \cdot \alpha$.

In the sequel, we will give matrices for all ordering we mention.

$$\mathtt{lp} \sim \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}, \mathtt{Dp} \sim \begin{pmatrix} 1 & \dots & 1 & 1 \\ 1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 1 & 0 \end{pmatrix}, \mathtt{dp} \sim \begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 0 & \dots & -1 \\ \vdots & \vdots & \iddots & \vdots \\ 0 & -1 & \dots & 0 \end{pmatrix}$$

Here, `dp` denotes a **degree reverse lexicographical ordering**, which can also be defined as

$\beta <_{\mathrm{dp}} \alpha \overset{\mathrm{def}}{\iff} \sum \beta_i < \sum \alpha_i$ or $\sum \beta_i = \sum \alpha_i$ and the last non–zero entry of $\alpha - \beta$ is negative.

Definition 3.1. Let $<$ be a total well–ordering on $\mathbb{N}^n$ and A be an integral $\mathbb{K}$–algebra with a PBW basis.

(1) An ordering $<=<_A$ is called **a monomial ordering** on A if the following conditions hold:
   - $\forall \alpha, \beta \in \mathbb{N}^n \ \alpha < \beta \ \Rightarrow \ x^\alpha <_A x^\beta$
   - $\forall \alpha, \beta, \gamma \in \mathbb{N}^n$ such that $x^\alpha <_A x^\beta$ we have $x^{\alpha+\gamma} <_A x^{\beta+\gamma}$.

(2) Any $f \in A \setminus \{0\}$ can be written uniquely as $f = cx^\alpha + f'$, with $c \in \mathbb{K}^*$ and $x^{\alpha'} <_A x^\alpha$ for any non–zero term $c'x^{\alpha'}$ of f'. We define

$$\begin{aligned} \mathrm{lm}(f) &= x^\alpha, \quad \text{the } \textbf{leading monomial} \text{ of } f, \\ \mathrm{lc}(f) &= c, \quad \text{the } \textbf{leading coefficient} \text{ of } f, \\ \mathrm{le}(f) &= \alpha, \quad \text{the } \textbf{leading exponent} \text{ of } f. \end{aligned}$$

3.2. Definition and Examples of G–algebras.

Definition 3.2. Let $\mathbb{K}$ be a field, $T = \mathbb{K}\langle x_1, \ldots, x_n\rangle$ and I be a two–sided ideal of T, generated by elements

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

where $c_{ij} \in \mathbb{K}^*$ and every $d_{ij} \in T$ is a polynomial, involving only standard monomials of T.

A $\mathbb{K}$–algebra $A = T/I$ is called **a G–algebra**, if the following two conditions hold.

• *Ordering condition*:
there exists a monomial well–ordering $<_A$ on $\mathbb{N}^n$ such that

$$\forall i < j \ \mathrm{lm}(d_{ij}) <_A x_i x_j.$$

• *Non–degeneracy condition*:
the sets $C = \{c_{ij}\}$ and $D = \{d_{ij}\}$ satisfy the following identities in A:

$$\mathcal{NDC}_{ijk} = 0 \ \forall\ 1 \leq i < j < k \leq n\ .$$

If A is a G–algebra we usually consider a fixed ordering $<_A$ satisfying the above conditions as part of the data of A.

By Theorem 2.2 and the construction, a G–algebra in variables $x_1, \ldots, x_n$ has a canonical PBW basis $\{x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n} \mid \alpha_k \geq 0\}$. Hence we regard a G–algebra (in n variables) as a generalization of a commutative polynomial ring in n variables.

If all $d_{ij} = 0$, we call an algebra **quasi–commutative**.

If all $c_{ij} = 1$, we are dealing with an **algebra of Lie type**.

If all $c_{ij} = 1$ and $d_{ij} = 0$, an algebra is commutative.

Computational Remark 3.3. In Singular:Plural, a G–algebra can be defined by the following data:

1) a commutative ring $(R, <)$, where $R = \mathbb{K}[x_1, \ldots, x_n]$ and $<$ is a well–ordering on R,

2) a matrix $C \in Mat_{n\times n}(\mathbb{K})$ with entries $C[i, j] = c_{ij} \neq 0$,

3) a matrix $D \in Mat_{n\times n}(R)$ with entries $D[i, j] = d_{ij}$ such that $\forall i < j$ $\mathrm{lm}(d_{ij}) < x_i x_j$.

As soon as one has declared an ordered ring $(R, <)$ with the help of the SINGULAR built–in type `ring` (it is explained in the SINGULAR Manual), one defines C and D of the type `matrix` in R. Finally, one calls `ncalgebra(C,D)`. The function `ncalgebra` checks whether the ordering condition is satisfied and if it is so, turns a ring $(R, <)$ into a non–commutative G–algebra with the relations like in the definition above. Note, that the non–degeneracy condition may be checked with the procedure `ndcond()` from the `nctools.lib`.

Let us illustrate setting the G–algebra and checking the non–degeneracy condition on the Example 2.6.

EXAMPLE 3.4. We consider the algebra
$A = \mathbb{K}(q_1, q_2)\langle x, y, z \mid yx = q_2xy + x, zx = q_1xz + z, zy = yz\rangle$.
Let us write matrices C and D explicitly:

$$C = \begin{pmatrix} 0 & q_2 & q_1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, D = \begin{pmatrix} 0 & x & z \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Then, the SINGULAR:PLURAL code looks as follows:

```
ring A = (0,q1,q2),(x,y,z),dp;
```

Here, we consider a ground field to be the transcendental extension of $\mathbb{Q}$ by q_1, q_2. We choose the degrevlex `dp` as the ordering on the variables $\{x, y, z\}$.

```
matrix C[3][3];
C[1,2] = q2; C[1,3] = q1; C[2,3] = 1;
matrix D[3][3];
D[1,2] = x; D[1,3] = z;
ncalgebra(C,D);  // the non--commutative initialization
```

Since no error message appeared, the given ordering is admissible. We can print the properties of this algebra.

```
A;
==>
  //   characteristic : 0
  //   2 parameter    : q1 q2
  //   minpoly        : 0
  //   number of vars : 3
  //        block   1 : ordering dp
  //                  : names    x y z
  //        block   2 : ordering C
```

```
  //   noncommutative relations:
  //    yx=(q2)*x*y+x
  //    zx=(q1)*x*z+z
```

If two variables commute, their commutative relation is, of course, not printed among "noncommutative relations" above.

Now, let us compute the non–degeneracy conditions.

```
LIB "nctools.lib";   // load the library "nctools.lib"
ideal N = ndcond();  // put the result of ndcond into ideal N
N;
==>
  N[1]=(-q2+1)*y*z-z
```

So, the algebra A violates non–degeneracy condition from the Definition 3.2 and hence is not a G–algebra, cf. Example 2.6. Although it is possible to define an algebra, which do not satisfy the non–degeneracy condition, we do not recommend to perform any computations in such algebra (see the interpretation of non–degeneracy in terms of the multiplication $\star$ in the proof of Lemma 2.2).

Example 3.5. Consider the algebra $B = \mathbb{K}\langle x, y, z \mid yx = xy + x, zx = xz, zy = yz + z\rangle$ from the last part of Example 2.6. As wee see, in this case the matrices look as follows:

$$C = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, D = \begin{pmatrix} 0 & x & 0 \\ 0 & 0 & z \\ 0 & 0 & 0 \end{pmatrix}.$$

Since all the entries of C are equal, we can pass just the corresponding value (that is, 1) as an argument (instead of the matrix) to the function `ncalgebra`. Hence, the code for setting B is the following:

```
ring B = 0,(x,y,z),dp;
matrix D[3][3];
D[1,2] = x; D[2,3] = z;
ncalgebra(1,D);  // the non--commutative initialization
B;
==>
  //   characteristic : 0
  //   number of vars : 3
  //        block   1 : ordering dp
  //                  : names    x y z
  //        block   2 : ordering C
  //   noncommutative relations:
```

```
  //    yx=xy+x
  //    zy=yz+z
LIB "nctools.lib";
ideal N = ndcond();
N;
==>
  N[1]=0
```

REMARK 3.6. (1) **Properties of** lm:

In a commutative polynomial algebra A, lm enjoys the following property: $\forall\ f, g \in A\ \ \mathrm{lm}(f \cdot g) = \mathrm{lm}(f) \cdot \mathrm{lm}(g)$.

In a G–algebra A, a product of two monomials is, in general, a polynomial. But nevertheless, since $\forall\ \alpha, \beta \in \mathbb{N}^n$ the leading term of $x^\alpha x^\beta$ is $c(\alpha, \beta)x^{\alpha+\beta}$ with $c(\alpha, \beta) \in \mathbb{K}^*$, the following weaker property holds:

$$\forall\ f, g \in A\ \ \mathrm{lm}(f \cdot g) = \mathrm{lm}(\mathrm{lm}(f) \cdot \mathrm{lm}(g)) = \mathrm{lm}(g \cdot f).$$

Equivalently, in terms of exponents both properties mean

$$\forall\ f, g \in A\ \ \ \mathrm{le}(f \cdot g) = \mathrm{le}(f) + \mathrm{le}(g).$$

In order to unify both commutative and non–commutative G–algebras, we are going to work rather with exponents than with monomials.

(2) Consider now a $\mathbb{K}$–algebra B, built on the vector space $\{x_1^{\alpha_1} \cdots x_n^{\alpha_n} \mid \alpha_i \geq 0\}$ with a multiplication $(\cdot)$ and a function $\mathrm{le} : B \setminus \{0\} \to \mathbb{N}^n$.

If $\forall\ f, g \in B\ \ \mathrm{le}$ satisfies $\mathrm{le}(f \cdot g) = \mathrm{le}(f) + \mathrm{le}(g)$ and $\mathrm{le}(f + g) = \max_<(\mathrm{le}(f), \mathrm{le}(g))$ (unless $\mathrm{le}(f) = \mathrm{le}(g)$ and $\mathrm{lc}(f) + \mathrm{lc}(g) = 0$) and, in addition, the non–degeneracy conditions vanish, then B is a G–algebra.

(3) **Non–additivity of** lm:

If $\mathrm{lc}(g)\,\mathrm{lm}(g) = -\,\mathrm{lc}(f)\,\mathrm{lm}(f)$ then
$\mathrm{lm}(f + g) < \max_<(\mathrm{lm}(f), \mathrm{lm}(g))$. For all other $f, g \in A$,

$$\mathrm{lm}(f + g) = \max_<(\mathrm{lm}(f), \mathrm{lm}(g)).$$

(4) **Properties of** lc:

For all $f, g \in A\ \ \mathrm{lc}(f \cdot g) = \mathrm{lc}\big(\mathrm{lm}(f) \cdot \mathrm{lm}(g)\big) \cdot \mathrm{lc}(f) \cdot \mathrm{lc}(g)$.
However, in algebras of Lie type ($c_{ij} = 1, \forall j > i$), we have
$\mathrm{lc}(f \cdot g) = \mathrm{lc}(f) \cdot \mathrm{lc}(g)$.

DEFINITION 3.7. An algebra A is called a **Gröbner–ready**, or simply a GR–**algebra**, if there exist an automorphism $\phi : A \to A$ and a well–ordering $<_A$, such that $\phi(A)$ is either a G–algebra or there exist a G–algebra B and a proper nonzero two–sided ideal $I \subset B$ such that $\phi(A) \cong B/I$.

In order to treat factor-algebras constructively, we need, in particular, two–sided Gröbner bases. Therefore we treat GR–algebras in detail in Section **§2**, 3.2.

REMARK 3.8. G–algebras were first introduced by J. Apel ([**3**]), however, without requiring the vanishing of the non–degeneracy conditions; they were omitted also in the work on *PBW algebras* ([**30**]) and in the book on solvable algebras ([**56**]), which are indeed G–algebras since the presence of PBW basis is required in the definition. In the classical work [**41**] on *algebras of solvable type* and in the recent book [**14**] the authors obtained a criterion for non–degeneracy but did not mention the polynomial conditions $\mathcal{NDC}_{ijk}$ explicitly. In [**9**] and [**10**] R. Berger introduced q–algebras (in our notation, these are the G–algebras with the restriction that the polynomials d_{ij} are quadratic), and imposed the vanishing conditions for what he calls "q–Jacobi sums" (which coincide with the non–degeneracy conditions). He treated these conditions as quantized Jacobi identities. We have obtained the non–degeneracy conditions independently ([**49**]) and, moreover, we have shown that the restriction to quadratic polynomials is not really essential. Li in his recent book [**56**] did not overcome the restrictions of Berger.

It is very natural to study G–algebras and their factor–algebras within the same framework. We avoid the name *PBW–algebras* ([**14**]) because, strictly speaking, a factor–algebra of an algebra with a PBW basis does not have a PBW basis itself. Moreover, consider a free algebra $T = \mathbb{K}\langle x, y\rangle$ and a two–sided ideal $I \subset T$, generated by yx. Then, indeed, the algebra T/I has the $\mathbb{K}$–basis $\{x^i y^j\}$ (like a PBW basis of any G–algebra in $\{x, y\}$). But T/I has zero divisors and is quite far from the context of G–algebras.

EXAMPLE 3.9 (Examples of G–algebras). Quasi–commutative polynomial rings (for example, the quantum plane $yx = q{\cdot}xy$), universal enveloping algebras of finite dimensional Lie algebras (see some of them explicitly in **§5**, 1), some iterated Ore extensions (see Def. 3.12 below), many quantum groups ([**14**]), some nonstandard quantum deformations ([**36**], [**38**]), Weyl algebras and most of various flavors of quantizations of Weyl algebras (like additive and multiplicative analogues of Weyl algebras, non–localized Hayashi algebras [**56**] etc), many important operator algebras (Section 6 and [**18**], [**56**]), Witten's deformation of $U(\mathfrak{sl}_2)$, Smith algebras, conformal $\mathfrak{sl}_2$–algebras ([**8**]), some of diffusion algebras ([**40**]) and many more.

REMARK 3.10. Consider the Sklyanin algebra ([**73**])

$$Skl_3(a,b,c) = \mathbb{K}\langle x_0, x_1, x_2\rangle / \langle \{ax_ix_{i+1} + bx_{i+1}x_i + cx_{i+2}^2 \mid i = 1, 2, 3 \bmod 3\}\rangle,$$

where $(a, b, c) \in \mathbb{P}^2 \setminus F$, for a known finite set F. Suppose that $a \neq 0, b \neq 0$. Then we can rewrite the relations in the following way:

$$x_1x_0 = -\frac{a}{b}x_0x_1 - \frac{c}{b}x_2^2, \;\; x_2x_1 = -\frac{a}{b}x_1x_2 - \frac{c}{b}x_0^2, \;\; x_2x_0 = -\frac{b}{a}x_0x_2 - \frac{c}{a}x_1^2.$$

Suppose there is a well–ordering $<$ with $x_2 < x_1 < x_0$, satisfying the inequalities $x_2^2 < x_0x_1$, $x_0^2 < x_1x_2$, $x_1^2 < x_0x_2$. But since $x_0x_2^2 < x_0^2x_1 < x_1^2x_2$, it follows that $x_0x_2 < x_1^2$, a contradiction to the assumption on $<$ to be a monomial ordering. Hence, unless $c = 0$, there is no monomial well–ordering, such that this algebra is a G–algebra (If $c = 0$, $Skl_3(a, b, 0)$ is a quasi–commutative algebra). Note, that the non–degeneracy conditions formally vanish on this non–G–algebra, hence the ordering condition in the definition 3.2 is essential.

EXAMPLE 3.11 (Examples of GR–algebras). Exterior algebras, Clifford algebras, finite dimensional associative algebras ([**24**]), primitive quotients of universal enveloping algebras, some quantum groups and so on.

3.3. Ore Extensions versus G–algebras.

DEFINITION 3.12 ([**59**]). Let R be an associative ring with 1, σ be a ring endomorphism of R and δ be a σ–derivation, that is an additive map $\delta : R \to R$ such that $\forall\ a, b \in R$, $\delta(ab) = \delta(a)\sigma(b) + a\delta(b)$. In particular, $\sigma(1) = 1$ and $\delta(1) = 0$.

Let x be a new indeterminate, satisfying the relation $x \cdot r = \sigma(r) \cdot x + \delta(r)$ for all $r \in R$. Then, we denote the obtained ring by $R[x; \sigma, \delta]$ and call it an **Ore extension**. Applying similar operation with further variables, we obtain an **(iterated) Ore extension** $R[x_1; \sigma_1, \delta_1][x_2; \sigma_2, \delta_2] \dots [x_n; \sigma_n, \delta_n]$.

Let R be a field ($\mathbb{K}$ or $\mathbb{K}(t_1, \dots, t_s)$). Then, an iterated Ore extension $R[x_1; \sigma_1, \delta_1][x_2; \sigma_2, \delta_2] \dots [x_n; \sigma_n, \delta_n]$ is called an **Ore algebra**, if for $1 \leq i, j \leq n$, $\sigma_i\delta_j = \delta_j\sigma_i$ and moreover, for $i < j$, $\sigma_i(x_j) = x_j$ and $\delta_i(x_j) = 0$.

The framework of Ore algebras is used in applications ([**19**], [**18**]) and can be distinguished for the two following situations: purely polynomial Ore algebras ($R = \mathbb{K}$) and rational Ore algebras ($R = \mathbb{K}(t_1, \dots, t_s)$). In both cases it can happen that $\{x_i\}$ do not commute with $\{t_j\}$: as an example we present the *rational Weyl algebra* $\mathbb{K}(x)\langle\partial \mid \partial x = x\partial + 1\rangle$.

In this work we are not going to consider the case of rational non–commutativity (like rational Ore algebras), although it is very interesting and deserves special attention due to several important applications.

For polynomial iterated Ore extensions over a field, there is the following proposition, appearing in [**41**] and, in generalized form, in [**14**]. It establishes a criterion when a G–algebra can be realized as a polynomial iterated Ore extension.

PROPOSITION 3.13. Let A be a G–algebra

$$A = \mathbb{K}\langle x_1, \ldots, x_n \mid \ x_j x_i = c_{ij} x_i x_j + d_{ij},\ 1 \leq i < j \leq n\rangle.$$

If every d_{ij} depends only on variables $x_1, \ldots, x_{i-1}$, then `lp` is an admissible ordering on A. Moreover, (A,`lp`) can be realized as iterated Ore extension $A = \mathbb{K}[x_1][x_2; \sigma_2, \delta_2] \ldots [x_n; \sigma_n, \delta_n]$ with $\sigma_j(x_i) = c_{ij} x_i$ and $\delta_j(x_i) = d_{ij}$, $1 \leq i < j \leq n$.

REMARK 3.14. Handling non–commutative polynomial algebras constructively as iterated Ore extensions makes some sense, but brings also some negative effects. Indeed, we cannot restrict ourself to polynomial Ore extensions only: an important algebra $U(\mathfrak{so}_3)$ (which is a G–algebra, see **§5**, 1.2 for the definition) does not satisfy the conditions of the above proposition.

Meanwhile we know that over $\mathbb{C}$, $U_{\mathbb{C}}(\mathfrak{so}_3) \cong U_{\mathbb{C}}(\mathfrak{sl}_2)$ (note, that this is not true over $\mathbb{Q}$) and the latter algebra (see **§5**, 1.1) can be realized as the following Ore extension: $\mathbb{K}[h][e; \mathbb{I}\mathrm{d}, \delta_e][f; \mathbb{I}\mathrm{d}, \delta_f]$ with $\delta_e(h) = -2e$ and $\delta_f(h) = 2f, \delta_f(e) = -h$ over a field $\mathbb{K}$.

Hence, once we are over $\mathbb{C}$ and have the explicit isomorphism as above, we can present $U_{\mathbb{C}}(\mathfrak{so}_3)$ as an Ore extension. It is, however, not clear whether we can do this when $\mathbb{K} = \mathbb{Q}$.

Another voice "contra" says that Ore extensions seem not to be good objects for unifying algebras and factor-algebras under one roof like we do for GR–algebras.

In what follows, we will use Ore extensions only in an auxiliary role, thus concentrating ourselves on GR–algebras.

4. Filtrations and Properties of G–algebras.

4.1. Preliminaries.

DEFINITION 4.1. We recall some definitions explicitly:

(1) An algebra A is called **filtered**, if for every non–negative integer i there is a subspace A_i such that

$$\mathbf{1})\ A_i \subseteq A_j \text{ if } i \leq j, \quad \mathbf{2})\ A_i \cdot A_j \subseteq A_{i+j} \quad \text{and} \quad \mathbf{3})\ A = \bigcup_{i=0}^{\infty} A_i.$$

The set $\{A_i \mid i \in \mathbb{N}\}$ is called a **filtration** of A.

(2) An **associated graded algebra** $\mathrm{Gr}(A)$ of a filtered algebra A is defined to be

$$\mathrm{Gr}(A) = \bigoplus_{i=1}^{\infty} G_i \quad \text{where } G_i = A_i/A_{i-1} \text{ and } A_{-1} = 0,$$

with the induced multiplication $(a_i + A_{i-1})(a_j + A_{j-1}) = a_i a_j + A_{i+j-1}$.

THEOREM 4.2 (Jacobson, [**59**]). Let A be a filtered algebra and $G = \mathrm{Gr}(A)$ be its associated graded algebra . Then

- If $\mathrm{Gr}(A)$ is left (right) Noetherian, then A is left (right) Noetherian,
- if $\mathrm{Gr}(A)$ has no zero divisors, then A has no zero divisors, too.

THEOREM 4.3 (Goldie-Ore, [**59**]). Let R be an integral associative unital ring. If it is left (resp. right) Noetherian, then it has a left (resp. right) quotient ring.

LEMMA 4.4. Let $Q = \{q_{ij} \mid 1 \leq i < j \leq n\}$. Consider the transcendental extension $\mathbb{K}(Q)$ of $\mathbb{K}$. A *quasi–commutative ring* in n variables, associated to the set Q is defined as follows:

$$\mathbb{K}_Q[x_1, \ldots, x_n] := \mathbb{K}(Q)\langle x_1, \ldots, x_n \mid \ \forall i < j \ \ x_j x_i = q_{ij} x_i x_j \rangle.$$

Then $\mathbb{K}_Q[x_1, \ldots, x_n]$ is a Noetherian domain.

PROOF. We can present $\mathbb{K}_Q[x_1, \ldots, x_n]$ as the iterated Ore extension $\mathbb{K}[x_1; \sigma_1 = 1, 0][x_2; \sigma_2, 0] \ldots [x_n; \sigma_n, 0]$ with $\sigma_j(x_i) = q_{ij} x_i$ and $\sigma_j(x_j) = x_j$ for $i < j$. Thus, every σ_j is an automorphism. By the theorem 1.2.9 from [**59**] and induction by n, $\mathbb{K}_Q[x_1, \ldots, x_n]$ is a Noetherian domain. □

For a G–algebra A, there are two different kinds of filtrations.

4.2. Weighted Degree Filtration.

Let $<_w = (<, \bar{w})$ be a weighted degree ordering on A, i.e. there is an n-tuple of strictly positive weights $\bar{w} = (w_1, w_2, \ldots, w_n)$ and some ordering $<$ (for example, a (reverse) lexicographical ordering) on A such that

$$\alpha <_w \beta \Leftrightarrow \sum_{i=1}^{n} w_i a_i < \sum_{i=1}^{n} w_i b_i \ \text{ or, if } \sum_{i=1}^{n} w_i a_i = \sum_{i=1}^{n} w_i b_i, \ \text{ then } \alpha < \beta.$$

Assume that $w_1 \geq \ldots \geq w_n$ and all the weights are positive integers.

The corresponding matrices for such orderings as weighted deglex `Wp` and weighted degrevlex `wp` look like follows:

$$\texttt{Wp} \sim \begin{pmatrix} \omega_1 & \dots & \omega_{n-1} & \omega_n \\ 1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 1 & 0 \end{pmatrix}, \texttt{wp} \sim \begin{pmatrix} \omega_1 & \omega_2 & \dots & \omega_n \\ 0 & 0 & \dots & -1 \\ \vdots & \vdots & \iddots & \vdots \\ 0 & -1 & \dots & 0 \end{pmatrix}.$$

Let us define $\deg_\omega(x^\alpha) := w_1\alpha_1 + \dots + w_n\alpha_n$ and call it **a weighted degree function on** A. For any polynomial $f \in A$, we define $\deg_\omega(f) := \deg_\omega(\mathrm{lm}(f))$, and we note that $\deg_\omega(x^\alpha) = 0 \Leftrightarrow \alpha = \bar{0}$.

LEMMA 4.5. $\deg_\omega(fg) = \deg_\omega(f) + \deg_\omega(g)$.

PROOF. Since on monomials we have

$$\deg_\omega(x^\alpha x^\beta) = \deg_\omega(x^{\alpha+\beta}) = \sum_{i=1}^{n} w_i(\alpha_i + \beta_i) = \deg_\omega(x^\alpha) + \deg_\omega(x^\beta),$$

hence, using Remark 3.6,

$$\deg_\omega(fg) = \deg_\omega(\mathrm{lm}(\mathrm{lm}(f)\,\mathrm{lm}(g))) = \deg_\omega(\mathrm{lm}(f)\,\mathrm{lm}(g)) = \deg_\omega(f) + \deg_\omega(g).$$

In particular, $\deg_\omega(f \cdot g) = \deg_\omega(g \cdot f)$. □

Let A_n be the $\mathbb{K}$–vector space generated by $\{m \in \mathrm{Mon}(A) \mid \deg_\omega(m) \leq n\}$. So, we see that $A_0 = \mathbb{K}$, $A_{w_n} = \mathbb{K} \oplus \mathbb{K}x_n$ if $w_{n-1} > w_n$, or $A_{w_n} = \mathbb{K} \oplus \bigoplus_{m=1}^{n} \mathbb{K}x_m$ if $w_1 = \dots = w_n$, hence

$$\forall\, 0 \leq i < j \;\; A_i \subseteq A_j \subseteq A \;\text{ and }\; A = \bigcup_{i=1}^{\infty} A_i.$$

From the Lemma 4.5 it follows that $\forall\, 0 \leq i < j \;\; A_i \cdot A_j \subseteq A_{i+j}$. In this case, $G_i = A_i/A_{i-1}$ is the set of weighted homogeneous elements of weighted degree i in A with $G_0 = A_0 = \mathbb{K}$. We have the following:

LEMMA 4.6. Suppose we have an algebra A, where $\forall\, i < j \;\; \deg_\omega(d_{ij}) < \deg_\omega(x_ix_j) = w_i + w_j$. Denote $\overline{x}_i = x_i + A_{i-1}$. Then

$$\mathrm{Gr}_{\deg_\omega}(A) = \bigoplus_{i=1}^{\infty} G_i = \; \mathbb{K}\langle \overline{x}_1, \dots, \overline{x}_n \;\mid\; \overline{x}_j\overline{x}_i = c_{ij}\overline{x}_i\overline{x}_j \;\forall\, j > i \rangle.$$

We see that in this case $\mathrm{Gr}_{\deg_\omega}(A)$ is isomorphic to the quasi–commutative ring in n variables. Hence, by the Jacobson's Theorem (4.2) A is a Noetherian domain.

This Lemma guarantees Noetherian and integral properties for Weyl algebras, universal enveloping algebras and some other algebras. Unfortunately, many important algebras (like positively graded quasi–homogeneous

algebras) do not satisfy the conditions of the Lemma directly. But there is another filtration that behaves better in more general situation.

4.3. Filtration by a Monomial Ordering.

A weighted degree filtration is well–known and used in every source on (non–commutative) computer algebra. However, there is another filtration which mainly remained unnoticed in spite of its natural appearance. Although we should note that the filtration by ordering has nothing to do with the homogeneity in the classical sense, since a *homogeneous part* of every polynomial with respect to the filtration by ordering is always a monomial.

Let $<$ be any monomial well–ordering on A. For $\alpha \in \mathbb{N}^n$, let A_α be the $\mathbb{K}$–vector space, spanned by the set $x^{<\alpha} \cup \{x^\alpha\}$, where $x^{<\alpha} := \{x^\beta \in A \mid x^\beta < x^\alpha\}$. Note, that each A_α is finite dimensional only if $<$ is a finitely supported well–ordering (cf. Def. 1.1). We see immediately that $A_{\bar{0}} = \mathbb{K}$ and

$$\forall\, \beta < \alpha \quad A_\beta \subset A_\alpha \subset A \text{ and } A = \bigcup_{\alpha \in \mathbb{N}^n} A_\alpha.$$

The property $A_\alpha \cdot A_\beta \subseteq A_{\alpha+\beta}$ holds because $\mathrm{lm}(x^\alpha x^\beta) = x^{\alpha+\beta}$ (cf. 3.6). Since the filtration is indexed not by $\mathbb{N}$ as in the classical definition 4.1, but by $\mathbb{N}^n$, we call it a **multi–filtration**, according to [**31**]. So, A is a multi–filtered algebra. Further on, let $\sigma(\alpha) := \max_<\{\gamma \mid \gamma < \alpha\}$, $\sigma(\bar{0}) = \emptyset$. Then $\forall\, \alpha \in \mathbb{N}^n$, $G_\alpha = A_\alpha / A_{\sigma(\alpha)} = \{x^\alpha\}$. It follows that

$$\mathrm{Gr}_<(A) = \bigoplus_{\alpha \in \mathbb{N}^n} G_\alpha \;\cong\; \mathbb{K}\langle \overline{x}_1, \ldots, \overline{x}_n \;\mid\; \overline{x}_j \overline{x}_i = c_{ij} \overline{x}_i \overline{x}_j \;\forall\, j > i \rangle,$$

where $\overline{x}_i = x_i + A_{\sigma(e_i)}$, $e_i = (0, \ldots, \underset{i}{1}, \ldots, 0)$. So, $\mathrm{Gr}_<(A)$ is isomorphic to the quasi–commutative ring in n variables.

Applying the generalizations of the Theorem of Jacobson to the case of multi–filtered algebras ([**31**]) we get a more general statement than using just the weighted–degree filtration.

THEOREM 4.7. Let A be a G–algebra. Then

1) A is left and right Noetherian,
2) A is an integral domain,
3) A has both left and right quotient rings.

REMARK 4.8. One can prove 1) also using the PBW Theorem and the Dixon's Lemma, like it is done in [**15**].

Indeed, the existence of PBW basis in a G–algebra implies 2). By 3.6, for any two nonzero polynomials f, g in a G–algebra A, there exists a finite

index set $I \subset \mathbb{N}^n$, such that $f \cdot g = \sum_{\alpha \in I} c_\alpha x^\alpha$, where $c_\alpha \in \mathbb{K}^*$. Since A has a PBW basis $\{x^\beta \mid \beta \in \mathbb{N}^n\}$, we have $fg = 0 \Leftrightarrow c_\alpha = 0 \; \forall \alpha \in I$, hence either f of g is zero.

However, there are algebras (not G–algebras), which have PBW basis but also zero–divisors: for example, the algebra $\mathbb{K}\langle x, y\rangle/\langle yx\rangle$.

EXAMPLE 4.9. Consider the example 8.3.9 from [**59**]: let R be an Ore extension $R = \mathbb{K}[u, v][x; \mathbb{I}\mathrm{d}, \delta]$ with $\delta = uv^2 \frac{\partial}{\partial u}$. That is,

$$R = \mathbb{K}\langle u, v, x \mid vu = uv,\ xu = ux + uv^2,\ xv = vx\rangle.$$

According to the ordering condition, $\mathrm{lm}(ux) > \mathrm{lm}(uv^2)$, hence $\mathrm{lm}(x) > \mathrm{lm}(v^2)$. Suppose there exists such an ordering, then non–degeneracy condition vanishes. Let us explore two possibilities for choosing the admissible ordering:

1) If we'd like to work with the PBW basis $\{u^a v^b x^c\}$, we have to introduce weights. If we choose the ordering $<_w$ to be, say, weighted degrevlex $\mathtt{wp}(w_u, w_v, w_x)$, then it suffices to take $<_w = \mathtt{wp}(1, w, 2w + w_0)$ for any $w \in \mathbb{Z}_{\geq 1}, w_0 \in \mathbb{Z}_{\geq 0}$. The minimal weight vector is then $(1, 1, 2)$; for any $w \in \mathbb{Z}_{\geq 1}$ the vector $(1, w, 2w)$ is turning R into the algebra with weighted homogeneous relations. By definition, R is a G–algebra with respect to $<_w$, hence it is Noetherian. In [**59**] it has been shown, that $\mathrm{Gr}(R)$, computed with respect to the classical filtration (that is, with weights $(1, 1, 1)$) is not Noetherian. It is not surprising, since any degree–ordering with weights $(1, 1, 1)$ violates the ordering condition from the definition of G–algebra.

2) There exists an ordering, where we do not need weights at all. Let

$$R' = \mathbb{K}\langle x, v, u \mid vx = xv,\ ux = xu - uv^2,\ uv = vu\rangle,$$

and let the ordering $<$ be just the lexicographical ordering $\mathtt{lp}$ with $x > v > u$. Then, $uv^2 < xu$ and R' is a G–algebra with respect to $<$. Note, that in this situation R' can be realized as Ore extension $\mathbb{K}[x, v][u; \mathbb{I}\mathrm{d}, -uv\frac{\partial}{\partial x}]$.

4.4. Filtration on Modules and Gel'fand–Kirillov dimension. Let R be an associative $\mathbb{K}$–algebra with generators $x_1, \ldots, x_m$. Assuming that each x_i has degree 1, we define an increasing degree filtration on R as above in 4.2. Then we have $F_0 = \mathbb{K}$, $F_1 = \mathbb{K} \oplus \bigoplus_{i=1}^m \mathbb{K}x_i$ and so on. Here $x_1, \ldots, x_m$ form a so-called *generating subspace* for R.

For any finitely generated left R–module M, there exists a finite dimensional subspace $M_0 \subset M$ (called a generating subspace for M), such that $RM_0 = M$. An ascending filtration $\{F_n, n \geq 0\}$ on R induces an ascending filtration on M, defined by $\{H_n := F_n M_0, n \geq 0\}$.

DEFINITION 4.10. Let $\{F_n, n \geq 0\}$ and $\{H_n, n \geq 0\}$ be filtrations on R and M as before. The **Gel'fand–Kirillov dimension** of M is defined to be

$$\mathrm{GKdim}(M) \;=\; \lim_{n\to\infty} \sup \log_n(\dim H_n)$$

in particular,

$$\mathrm{GKdim}(R) \;=\; \mathrm{GKdim}({}_RR) \;=\; \lim_{n\to\infty} \sup \log_n(\dim F_n).$$

Indeed, $\mathrm{GKdim}(R)$ and $\mathrm{GKdim}(M)$ are independent of the choice of generating subspaces. For a commutative algebra $C = \mathbb{K}[x_1, \ldots, x_n]$, $\mathrm{GKdim}(C)$ equals the Krull dimension $\mathrm{Kr.dim}(C) = n$.

PROPOSITION 4.11. ([**59**]). Let A be a G–algebra in n variables. Then, for any finitely generated A–module M,

$$\mathrm{GKdim}(M) = \mathrm{GKdim}(\mathrm{Gr}(M)), \text{ hence } \mathrm{GKdim}(A) = \mathrm{GKdim}(\mathrm{Gr}(A)) = n.$$

In both books by Li ([**56**]) and Bueso et al. ([**14**]) there are algorithms for computing the $\mathrm{GKdim}(M)$, so we are not going to discuss this here (but note, that due to the previous proposition, it reduces to the computation of dimension of $\mathrm{Gr}(M)$ over $\mathrm{Gr}(A)$, what is a (quasi–)commutative algebra). The latter algorithm together with accompanying tools has been already implemented in SINGULAR:PLURAL by J. Lobillo and C. Rabelo ([**57**]).

Let us recall several further definitions.

DEFINITION 4.12. ([**59**], 8.5.8) Let M be a module over an algebra A. M is called **holonomic**, if $\mathrm{GKdim}(A/\operatorname{Ann}_A M) = 2 \cdot \mathrm{GKdim}(M)$.

DEFINITION 4.13. Let A be an associative $\mathbb{K}$–algebra and M be a left A–module.

1) The **grade** of M is defined to be

$$j(M) = \min\{i \mid \operatorname{Ext}^i_A(M, A) \neq 0\},$$

or $j(M) = \infty$, if no such i exists. By convention $j(\{0\}) = \infty$.

2) It is said, that an algebra A satisfies the **Auslander condition**, if for every finitely generated A–module M, for all $i \geq 0$ and for all submodules $N \subseteq \operatorname{Ext}^i_A(M, A)$ the inequality $j(N) \geq i$ holds.

3) A is called an **Auslander regular** algebra, if it is Noetherian with $\mathrm{gl.dim}(A) < \infty$ and the Auslander condition holds.

4) An algebra A is called a **Cohen–Macaulay** algebra, if $j(M) + \mathrm{GKdim}(M) = \mathrm{GKdim}(A) < \infty$ for every finitely generated nonzero A–module M.

Using the structural results from [**59**] on quasi–commutative rings, which appear as graded rings of G–algebras, we have the following.

PROPOSITION 4.14. Let A be a G–algebra in n variables. Then

1) the global homological dimension $\mathrm{gl.dim}(A) \leq n$,
2) the Krull dimension $\mathrm{Kr.dim}(A) \leq n$,
3) A is Auslander–regular and a Cohen–Macaulay algebra.

REMARK 4.15. 1) and 2) were proved in [**30**] with the multifiltering technique, which was also applied for the proof of 3) in [**31**]. We will give a constructive proof of 1) in Theorem **§2**, 4.16, which was obtained independently using Gröbner bases and our generalization of Schreyer's theorem on syzygies.

Note, that both 1) and 2) may be strict inequalities. It is known, that for a n–th Weyl algebra W_n in $2n$ variables over a field of char 0, $\mathrm{gl.dim}\, W_n = n$. We will discuss the conditions on the equality further in the Proposition **§3**, 4.17.

Concerning the Krull dimension, T. Levasseur showed in [**55**], that for a complex semisimple Lie algebra $\mathfrak{g}$ of dimension d, the Krull (more precisely, Krull–Gabriel–Rentschler) dimension of $U(\mathfrak{g})$ is equal to the dimension of a Borel subalgebra of $\mathfrak{g}$ and hence, is strictly smaller than d.

3) was first proved in [**31**], using the multifiltering technique.

5. Opposite algebras

Let A be an associative algebra over $\mathbb{K}$. Then the *opposite algebra* A^{opp} is defined by taking the same vector-space A and introducing a new "opposite" multiplication on it, that is $f * g := g \cdot f$. Like A, it is an associative $\mathbb{K}$–algebra. Of course, $(A^{\mathrm{opp}})^{\mathrm{opp}} = A$.

Let A be a G–algebra in n variables. What is the opposed algebra of it?

In the two subsections which follow, we present two natural constructions for A^{opp} and for both of them we conclude, that A^{opp} is a G–algebra.

Let M be a left A–module. Then, $\forall\ a \in A,\ m \in M,\ am \in M$. In the opposite algebra, $(am)^{\mathrm{opp}} = m^{\mathrm{opp}} * a^{\mathrm{opp}}$. Hence, setting M^{opp} to be the same vector space as M, it naturally becomes a right A^{opp}–module. Similarly, for any right A–module N, N^{opp} is naturally a left A^{opp}–module.

Hence, we can perform right–sided computations like Gröbner basis, syzygy modules et cetera, having implemented only left–sided functionality and procedures for an effective treatment of opposite algebras and transfer of objects from an algebra into its opposite.

Let $B = A/I$ be a GR–algebra. Then $B^{\mathrm{opp}} = A^{\mathrm{opp}}/I^{\mathrm{opp}}$ is a GR–algebra, since A^{opp} is a G–algebra, as we will see below, and I^{opp} is a proper two–sided ideal in A^{opp}.

For $p \in A$, we denote the opposed polynomial either by p^* or by P.

Note, that using the characterization of non–degeneracy 2.4, $\forall\ 1 \leq i < j < k \leq n$, $0 = NDC_{ijk} = (x_k \cdot x_j) \cdot x_i - x_k \cdot (x_j \cdot x_i)$. In the opposite algebra this looks like $0 = (x_i^* * x_j^*) * x_k^* - x_i^* * (x_j^* * x_k^*)$.

5.1. Opposite Algebra with Reversed PBW Basis. The form of non–degeneracy conditions motivates the following choice PBW basis of A^{opp}.

Since there is a bijection between vector-spaces of A and A^{opp}, we can find a particular one, which induces a bijection on monomials. Choose a bijection, sending x_i to $X_{n+1-i} := x_i^*$. Denote the induced monoid automorphism by $\sigma : \mathbb{N}^n \to \mathbb{N}^n$, $\sigma(\bar{\alpha}) = \sigma((\alpha_1, \ldots, \alpha_n)) := (\alpha_n, \ldots, \alpha_1)$.

Since a $\mathbb{K}$–basis of A is the PBW basis $\mathrm{Mon}(A) = \{x^\alpha \mid \alpha \in \mathbb{N}^n\}$, where $x^\alpha = x_1^{\alpha_1} \cdot \ldots \cdot x_n^{\alpha_n}$, it is quite natural to define a monomial on A^{opp} to be $(x^\alpha)^* := X^{\sigma(\bar{\alpha})} = X_n^{\alpha_n} * \ldots * X_1^{\alpha_1}$. Hence, with this choice of monomial, the non–degeneracy conditions are vanishing.

Then, on A^{opp} with PBW basis $\{X^\beta \mid \beta \in \mathbb{N}^n\}$ there are relations, opposed to A, namely $\forall\ 1 \leq i < j \leq n$, $X_iX_j = C_{ji}X_jX_i + D_{ji}$. Define $C_{ji} := c_{n+1-i,n+1-j}$ and $D_{ji} := d^*_{n+1-i,n+1-j}$, then the pair (X_i, X_j) together with the relation is opposed to the pair (x_{n+1-i}, x_{n+1-j}).

Let $M \in \mathbb{GL}(n, \mathbb{K})$ be the matrix, representing an admissible well–ordering ordering $<_M$ on A. Define a matrix M^* by reverting the order of columns from $M = (M_1 \mid \cdots \mid M_n)$ to $M^* = (M_n \mid \cdots \mid M_1)$. Note that $<_{M^*}$ is a well–ordering if and only if $<_M$ is. Moreover, for any $\alpha \in \mathbb{N}^n$, $M\alpha = M^*\sigma(\alpha)$. Hence, we have the following:

$$x^\alpha <_M x^\beta \Leftrightarrow M\alpha <_{\mathtt{lp}} M\beta \Leftrightarrow M^*\sigma(\alpha) <_{\mathtt{lp}} M^*\sigma(\beta) \Leftrightarrow X^{\sigma(\alpha)} <_{M^*} X^{\sigma(\beta)}.$$

Then, from $\mathrm{lm}(d_{ij}) <_M x_ix_j$ follows, that $\mathrm{lm}(D_{ji}) <_{M^*} X_jX_i$ and the ordering condition is satisfied. Hence, A^{opp} is a G–algebra in n variables by the Definition 3.2.

Example 5.1. In our implementation, the command `opposite` constructs an opposite algebra from a given G–algebra, using the method described above. A command `oppose` computes an opposite object in A^{opp} from a given object in A. We cut some lines from the output in order to present the example in more compact form.

```
LIB "ncalg.lib";
def S = makeQso3();
```

```
setring S; S;
==>

            ...
  //                  : names    x y z
  //   noncommutative relations:
  //    yx=(Q2)*xy+(-Q)*z
  //    zx=1/(Q2)*xz+1/(Q)*y
  //    zy=(Q2)*yz+(-Q)*x
poly p = (xy+z)^2; p;
==>
  (Q2)*x2y2+(-Q3+2)*xyz+(Q3-1)/(Q)*x2+1/(Q)*y2+z2
def Sop = opposite(S);
setring Sop; Sop;
==>

            ...
  //                  : names    Z Y X
  //   noncommutative relations:
  //    YZ=(Q2)*ZY+(-Q)*X
  //    XZ=1/(Q2)*ZX+1/(Q)*Y
  //    XY=(Q2)*YX+(-Q)*Z

poly q = oppose(S,p);  q;
==>
  (Q2)*Y2X2+(-Q3+2)*ZYX+(Q3-1)/(Q)*X2+1/(Q)*Y2+Z2
```

5.2. Opposite Algebra via Opposed Relations. Indeed, there is another canonical way of viewing A^{opp} as a G–algebra. We can keep the PBW basis but change relations instead. Let $Y_i := x_i^*$, a new multiplication be denoted by $\star$ and the basis consists of monomials $\{Y_1^{\alpha_1} \star \ldots \star Y_n^{\alpha_n}\}$. Opposed monomials will be again monomials, but the relations of an algebra will become $\{Y_j \star Y_i = C_{ij} Y_i Y_j + D_{ij}\}$, where $C_{ij} := \frac{1}{c_{ij}}$ and $D_{ij} := -\frac{1}{c_{ij}} d_{ij}^*\}$. Hence, we can use the same ordering as of A on A^{opp}, since it respects the ordering property from the definition of G–algebra.

In order to see that the non–degeneracy conditions vanish, consider one of the three parts of the polynomial $\mathcal{NDC}_{ijk}$:

$(c_{ik}c_{jk} \cdot d_{ij}x_k - x_k d_{ij})^* = c_{ik}c_{jk} \cdot Y_k d_{ij}^* - d_{ij}^* Y_k =$

$= -c_{ij}c_{ik}c_{jk} Y_k D_{ij} + c_{ij} D_{ij} Y_k = \frac{1}{C_{ij}C_{ik}C_{jk}} (C_{ik}C_{jk}D_{ij}Y_k - Y_k D_{ij}).$

We see that the totally symmetric coefficient came out and the same will happen with two other parts of $\mathcal{NDC}_{ijk}$. Hence, $\forall\ 1 \leq i < j < k \leq n$,

$$\mathcal{NDC}_{ijk}^{A^{\text{opp}}} = \frac{1}{C_{ij}C_{ik}C_{jk}}\mathcal{NDC}_{ijk}^{A} = 0.$$

6. The Ubiquity of GR–algebras

Let t be a variable and ∂ denotes the operator of a partial differentiation with respect to t, $\frac{\partial}{\partial t}$. For any function $f \in C^\infty(t)$, we introduce an operator F, $F(t) = f \cdot t$, that is the operator of left multiplication by f. We call f a *representative* of F. In general, the operators F and ∂ do not commute, but there is still a relation between the two actions.

LEMMA 6.1. $\partial \circ F = F \circ \partial + \partial(f)$.

PROOF. $\forall\ h \in C^\infty(t)$, we have the following:

$(\partial \circ F)(h) = \partial(f \cdot h) = f \cdot \partial(h) + \partial(f) \cdot h = (F \circ \partial)(h) + \partial(f) \cdot (h)$,

hence $\partial \circ F = F \circ \partial + \partial(f)$. □

In the sequel, we will denote both the operator and its representative by the same letter.

EXAMPLE 6.2. Taking t and $d = \frac{d}{dt}$ as variables, we obtain the algebra $A_1 = \mathbb{K}\langle t, d \mid dt = td + 1\rangle$, the *first Weyl algebra* — a very famous object in mathematics. Note, that the n–th Weyl algebra is defined to be $A_n = \mathbb{K}\langle x_1, \ldots, x_n, \partial_1, \ldots, \partial_n \mid \partial_i x_j = x_j\partial_i + \delta_{ij}\rangle$, where δ_{ij} is the Kronecker symbol ($\delta_{jj} = 1$ and $\delta_{ij} = 0$, $\forall\ i \neq j$). Here, ∂_i could be viewed as the operator of partial differentiation $\frac{\partial}{\partial x_i}$.

EXAMPLE 6.3. Let e denotes the operator $e^{\lambda t}$, where λ considered as a parameter. Then there is the *exponential algebra*

$$E = \mathbb{K}(\lambda)\langle e, \partial \mid \partial \cdot e = e \cdot \partial + \lambda e\rangle.$$

Both examples 6.2 and 6.3 are G–algebras for any degree well–ordering.

EXAMPLE 6.4. Let sin denotes the operator with the representative $sin(t)$, then $\partial \cdot sin = sin \cdot \partial + cos$ and hence we need the variable cos, corresponding to the operator with the representative $cos(t)$. Then, $\partial \cdot cos = cos \cdot \partial - sin$ and there is the algebra

$$T^0 = \mathbb{K}\langle sin, cos, \partial \mid \partial \cdot cos = cos \cdot \partial - sin, \partial \cdot sin = sin \cdot \partial + cos, cos \cdot sin = sin \cdot cos\rangle.$$

A direct computation shows that the element $c = sin^2 + cos^2$ commutes with ∂ (even more, we can show that c generates the *center* of T^0 if $\operatorname{char}\mathbb{K} =$

0). Hence, we should take the factor–algebra by the the ideal, generated by $cos^2 + sin^2 - 1$. In such a way we obtain

the *trigonometric algebra* $T = T^0/\langle cos^2 + sin^2 - 1\rangle$.

This algebra is a GR–algebra for any degree well–ordering.

Example 6.5. Consider the operator ln, assigned to the natural logarithm $\ln(t)$. Then, $\partial \cdot ln = ln \cdot \partial + t^{-1}$. Adding $s := t^{-1}$ as a variable, we obtain the *logarithmic algebra*:

$$L = \mathbb{K}\langle s, ln, \partial \mid \partial \cdot s = s \cdot \partial - s^2, \partial \cdot ln = ln \cdot \partial + s, ln \cdot s = s \cdot ln\rangle.$$

Note, that in order to be a G–algebra, the monomial ordering $<$ should satisfy the property $s < \partial$. Then, $s^2 < s\partial$ and the ordering condition is satisfied.

We can also consider other operations instead of differentiation.

Consider the **shift operator** with respect to a constant $c \in \mathbb{K}$,

$$\delta_c : f(x) \mapsto f(x - c).$$

Then, by the analogous statement to the Lemma 6.1, there is

$$\text{the } \textit{shift algebra } S(x, \delta_c) = \mathbb{K}\langle x, \delta_c \mid \delta_c \cdot x = x \cdot \delta_c - c\delta_c\rangle.$$

If $c < 0$ (resp. $c > 0$), δ_c is called an *advance operator* (resp. a *time–delay operator*) ([**19**]).

Consider the **difference operator**

$$\Delta : f(x) \mapsto f(x + 1) - f(x).$$

Again, there is an analogue to Lemma 6.1, and the *difference algebra*

$$\mathbb{K}\langle x, \Delta \mid \Delta \cdot x = x \cdot \Delta + \Delta + 1\rangle.$$

Note, that both shift and difference algebras are G–algebras.

Many important operators allow such algebraic presentations as G– and GR–algebras. See [**18**] for details, implementation and applications.

7. Applications of Non–degeneracy Conditions

Definition 7.1. Let A be a GR–algebra over a field $\mathbb{K}$.

We call an algebra $A(q_1, \ldots, q_m)$, depending on parameters $(q_1, \ldots, q_m)$, a G–**quantization of** A, if

- $A(q_1, \ldots, q_m)$ is a GR–algebra over $\mathbb{K}(q_1, \ldots, q_m)$,
- $A(q_1, \ldots, q_m)$ is a GR–algebra for any values of $q_k \in \mathbb{K}$,
- $A(1, \ldots, 1) = A$.

Let A be a G–algebra, generated by $x_1, \ldots, x_n$.
How to determine the set of all G–quantizations of A?

(1) Compute the non–degeneracy conditions and obtain a set S of polynomials in $x_1, \ldots, x_n$ with coefficients depending on $q_1, \ldots, q_m$.
(2) Form the ideal $I_S \in \mathbb{K}[q_1, \ldots, q_m]$ generated by all the coefficients of monomials of every polynomial from S.
(3) Compute the associated primes from the primary decomposition of the radical of I_S.
(4) Throw away every component (that is, an associated prime) which violates $A(1, \ldots, 1) = A$.

REMARK 7.2. We use the computer algebra system SINGULAR:PLURAL [**53**], with its commutative backbone SINGULAR and non–commutative extension PLURAL. We proceed with the described procedure as follows:

We compute the non–degeneracy conditions either with the help of PLURAL or manually. Then, using SINGULAR and its library PRIMDEC [**21**], we compute the Gröbner basis of $\mathcal{I}$ and then the associated primes of the primary decomposition of the radical of $\mathcal{I}$. An implementation of the essential algorithms including the primary decomposition is available in polynomial rings over various ground fields $\mathbb{K}$ (like $\operatorname{char} \mathbb{K} = 0$ or $\operatorname{char} \mathbb{K} \gg 0$ as well as their transcendent and simple algebraic extensions). Here, however, we assume our coefficients $q_1, \ldots, q_m$ to be specialized in the field $\mathbb{C}$.

Of course, one can insert further constraints in order to analyze the set one obtains. Parametric ideals, modules and subalgebras can be studied in a similar way to the investigation of parametric algebras that we present here.

In what follows, we will work with semi–algebraic sets. Recall, that for a field $\mathbb{K}$, a **semi–algebraic set** is a subset of $\mathbb{K}^n$, which is a finite Boolean combination of sets of the form $\{\bar{x} \in \mathbb{K}^n : f(\bar{x}) > 0\}$ and $\{\bar{x} \in \mathbb{K}^n : g(\bar{x}) = 0\}$ for $f, g \in \mathbb{K}[x_1, \ldots, x_n]$. In particular, for any $h \in \mathbb{K}[\bar{x}]$, the set $H_{\neq} = \{\bar{x} \in \mathbb{K}^n : h(\bar{x}) \neq 0\}$ is semi–algebraic.

7.1. G–quantizations of Weyl Algebras.

Let $A_n = \mathbb{K}\langle x_1, \ldots, x_n, y_1, \ldots, y_n \mid y_i x_i = x_i y_i + 1\rangle$ be the classical n-th Weyl algebra, where we can interpret y_i as the differential operator $\partial_{x_i} := \frac{\partial}{\partial x_i}$.

From now on, we use the following compact way for encoding the G–algebra in 4 variables (c_{ij}, d_{ij} are from the Definition 3.2):

$$\begin{pmatrix} \mathbf{x_1} & c_{12} & c_{13} & c_{14} \\ d_{12} & \mathbf{x_2} & c_{23} & c_{24} \\ d_{13} & d_{23} & \mathbf{x_3} & c_{34} \\ d_{14} & d_{24} & d_{34} & \mathbf{x_4} \end{pmatrix}$$

Let's take $A_2 = \mathbb{K}\langle x, \partial_x, y, \partial_y \mid \partial_x x = x\partial_x + 1, \partial_y y = y\partial_y + 1\rangle$. In our new notation it corresponds to the matrix

$$\begin{pmatrix} \mathbf{x} & 1 & 1 & 1 \\ 1 & \partial_{\mathbf{x}} & 1 & 1 \\ 0 & 0 & \mathbf{y} & 1 \\ 0 & 0 & 1 & \partial_{\mathbf{y}} \end{pmatrix}$$

With such an ordering of variables the PBW basis is $\{x^{n_1}\partial_x^{n_2} y^{n_3}\partial_y^{n_4}\}$.

We specify the following constraints to be fulfilled:

- let the general G–quantization be $\Delta(A_2) = \Delta(A_2, \{c_{ij}\}, \{d_{ij}\}, <) = \mathbb{K}\langle x_1, x_2, x_3, x_4 \mid x_j x_i = c_{ij} x_i x_j + d_{ij},\ \forall j > i \rangle$
- $\forall\ i < j \quad c_{ij} \in \mathbb{K}^*,\ d_{ij} \in \mathbb{K}$ (as for d_{ij}, we consider two cases only: $d_{ij} = 0$ and $d_{ij} \neq 0$). It means we investigate only Weyl–like G–quantizations.
- $d_{12} = d_{34} = 1$

Since $\forall\ i < j\ d_{ij} \in \mathbb{K}$, for any well–ordering $<$ on $\Delta(A_2)$ we have $d_{ij} < x_i x_j$ and $\Delta(A_2)$ is a G–algebra in 4 variables, if non–degeneracy conditions vanish. However, if we choose $<$ to be a well–ordering, $\Delta(A_2)$ does not depend on the concrete one. In our encoding it looks the following way:

$$\begin{pmatrix} \mathbf{x} & c_{12} & c_{13} & c_{14} \\ 1 & \partial_{\mathbf{x}} & c_{23} & c_{24} \\ d_{13} & d_{23} & \mathbf{y} & c_{34} \\ d_{14} & d_{24} & 1 & \partial_{\mathbf{y}} \end{pmatrix}$$

Since the set $\mathcal{U}_3 = \{(i,j,k) \mid 1 \leq i < j < k \leq 4\}$ in this case is equal to $\{(1,2,3), (1,2,4), (1,3,4), (2,3,4)\}$, we have four equations derived from the four non–degeneracy conditions which $\forall\ (i,j,k) \in \mathcal{U}_3$ look as follows:

$$d_{ij}(c_{ik}c_{jk} - 1) \cdot x_k + d_{ik}(c_{jk} - c_{ij}) \cdot x_j + d_{jk}(1 - c_{ij}c_{ik}) \cdot x_i.$$

Now we define two sets of commutative variables $C = \{c_{ij} \mid 1 \leq i < j \leq 4\}$ and $D = \{d_{ij} \mid 1 \leq i < j \leq 4\} \setminus \{d_{12}, d_{34}\}$ (since $d_{12} = d_{34} = 1$). Then

we have the following ideal in the commutative polynomial ring $\mathbb{K}[C, D]$ in 10 variables,

$$\mathcal{I} = \langle d_{ij}(c_{ik}c_{jk} - 1), d_{ik}(c_{jk} - c_{ij}), d_{jk}(c_{ij}c_{ik} - 1) \mid (i, j, k) \in \mathcal{U}_3 \rangle.$$

Using a computer algebra system SINGULAR ([**35**]), we compute the Gröbner basis of $\mathcal{I}$ and then the primary decomposition of the radical of $\mathcal{I}$ ([**21**]). Performing the computations, we find out, that the 4–dimensional variety, defined by $\sqrt{\mathcal{I}}$, consists of 8 components (corresponding to associated prime ideals). Let us denote the corresponding types of algebras by $\Delta_1, \ldots, \Delta_8$. Now we list them all, using the following considerations:

- d_{ij}: if there are no restrictions on some d_{ij}, we depict it by $*$ in the matrix, interpreting it as a free ("random") parameter,
- c_{ij}: if no conditions on some c_{ij} are given, we will introduce the parameters q', q'' for "single" (appearing only once) coefficients and q for "block" (appearing more than once) coefficients in the corresponding matrix. These parameters are viewed then as the generators of the transcendental field extension $\mathbb{K}(\underline{q})$.

$$\Delta_1 = \begin{pmatrix} \mathbf{x} & 1 & 1 & 1 \\ 1 & \partial_{\mathbf{x}} & 1 & 1 \\ * & * & \mathbf{y} & 1 \\ * & * & 1 & \partial_{\mathbf{y}} \end{pmatrix}, \qquad \Delta_2 = \begin{pmatrix} \mathbf{x} & -1 & -1 & -1 \\ 1 & \partial_{\mathbf{x}} & -1 & -1 \\ * & * & \mathbf{y} & -1 \\ * & * & 1 & \partial_{\mathbf{y}} \end{pmatrix},$$

$$\Delta_3 = \Delta_3(q') = \begin{pmatrix} \mathbf{x} & q' & 1 & 1 \\ 1 & \partial_{\mathbf{x}} & 1 & 1 \\ 0 & 0 & \mathbf{y} & 1 \\ * & 0 & 1 & \partial_{\mathbf{y}} \end{pmatrix}, \quad \Delta_4 = \Delta_4(q') = \begin{pmatrix} \mathbf{x} & q' & -1 & -1 \\ 1 & \partial_{\mathbf{x}} & -1 & -1 \\ 0 & 0 & \mathbf{y} & -1 \\ * & 0 & 1 & \partial_{\mathbf{y}} \end{pmatrix},$$

$$\Delta_5(q', q'', q) = \begin{pmatrix} \mathbf{x} & q' & q & q^{-1} \\ 1 & \partial_{\mathbf{x}} & q^{-1} & q \\ 0 & 0 & \mathbf{y} & q'' \\ 0 & 0 & 1 & \partial_{\mathbf{y}} \end{pmatrix}, \quad \Delta_6 = \Delta_6(q) = \begin{pmatrix} \mathbf{x} & q & q^{-1} & q \\ 1 & \partial_{\mathbf{x}} & q & q^{-1} \\ 0 & * & \mathbf{y} & q \\ 0 & 0 & 1 & \partial_{\mathbf{y}} \end{pmatrix},$$

$$\Delta_7 = \Delta_7(q) = \begin{pmatrix} \mathbf{x} & q & q^{-1} & q \\ 1 & \partial_{\mathbf{x}} & q & q^{-1} \\ * & 0 & \mathbf{y} & q^{-1} \\ 0 & 0 & 1 & \partial_{\mathbf{y}} \end{pmatrix}, \quad \Delta_8 = \Delta_8(q) = \begin{pmatrix} \mathbf{x} & q & q & q^{-1} \\ 1 & \partial_{\mathbf{x}} & q^{-1} & q \\ 0 & * & \mathbf{y} & q^{-1} \\ 0 & 0 & 1 & \partial_{\mathbf{y}} \end{pmatrix}.$$

Now we check, whether $\Delta_i(1, \ldots, 1) = A$. Using the encoding it turns to be especially simple — all the G–quantizations of A can be represented by

the $\Delta_5(q', q'', q)$, since, substituting everywhere the free parameter $*$ with 0, we have

$$\Delta_1 = \Delta_5(1,1,1), \quad \Delta_3 = \Delta_5(q',1,1), \quad \Delta_6 = \Delta_5(q,q,q^{-1}),$$
$$\Delta_7 = \Delta_5(q,q^{-1},q^{-1}), \quad \Delta_8 = \Delta_5(q,q^{-1},q).$$

If we substitute $*$ with a unit, the only G–quantization of A is $\Delta_5(q', q'', q)$. Note, that in any case Δ_2 and Δ_4 are not G–quantizations. We could have avoided them by restricting c_{ij} to values in $\mathbb{K}_+$.

It's interesting to see how this classification reflects some of classical algebras related to A_2. Recall the encodings of algebras:

$$A_1 \sim \begin{pmatrix} \mathbf{x} & 1 \\ 1 & \partial_{\mathbf{x}} \end{pmatrix}, \quad A_1(q) \sim \begin{pmatrix} \mathbf{x} & q \\ 1 & \partial_{\mathbf{x}} \end{pmatrix}$$

Then it's easy to see that

- $A_2 = A_1 \otimes_{\mathbb{K}} A_1$ is of the type Δ_1,
- $A_1(q') \otimes_{\mathbb{K}(q')} A_1$ is of the type Δ_3,
- $A_1(q') \otimes_{\mathbb{K}(q',q'')} A_1(q'')$ is of the type $\Delta_5(q', q'', 1)$.

What happens to $\Delta_3, \Delta_6, \Delta_7, \Delta_8$, if we substitute the free parameter with some unit? They are not G–quantizations anymore, but still interesting G–algebras, like Δ_7 where $*$ is replaced with 1: we get an algebra with ∂_x, ∂_y acting as classical differentials on x, y (which generate $A_1(q^{-1}) = \mathbb{K}\langle x, y \mid yx = q^{-1}xy + 1\rangle$).

In order to go back to the classical PBW basis $\{x^{n_1}y^{n_2}\partial_x^{n_3}\partial_y^{n_4}\}$ it is enough just to change our encoding, permuting the corresponding entries:

$$\begin{pmatrix} \mathbf{x} & q' & q & q^{-1} \\ 1 & \partial_{\mathbf{x}} & q^{-1} & q \\ 0 & 0 & \mathbf{y} & q'' \\ 0 & 0 & 1 & \partial_{\mathbf{y}} \end{pmatrix} \longrightarrow \begin{pmatrix} \mathbf{x} & q & q' & q^{-1} \\ 0 & \mathbf{y} & q & q'' \\ 1 & 0 & \partial_{\mathbf{x}} & q \\ 0 & 1 & 0 & \partial_{\mathbf{y}} \end{pmatrix}$$

The G–quantization of the type $\Delta := \Delta_5$ can be generalized for higher Weyl algebras.

THEOREM 7.3. Consider $A_n = \mathbb{K}\langle x_1, \ldots, x_n, \partial_{x_1}, \ldots, \partial_{x_n} \mid \partial_{x_i}x_i = x_i\partial_{x_i} + 1\rangle$. Given n "single" parameters $p_1, \ldots, p_n$ and $m = \frac{1}{2}n(n-1)$ "block" parameters $q_1, \ldots, q_m$, then there exists a $m+n$–parameter G–quantization $\Delta_n(\underline{p}, \underline{q})$ which has the following form in the compact encoding:

$$\begin{pmatrix}
\mathbf{x_1} & p_1 & q_1 & q_1^{-1} & \dots\dots & q_i & q_i^{-1} \\
1 & \partial_{\mathbf{x_1}} & q_1^{-1} & q_1 & \dots\dots & q_i^{-1} & q_i \\
0 & 0 & \mathbf{x_2} & p_2 & \dots\dots & \vdots & \vdots \\
0 & 0 & 1 & \partial_{\mathbf{x_2}} & \dots\dots & \vdots & \vdots \\
. & . & . & . & . \ . \ . & . & . \\
0 & 0 & 0 & 0 & \dots\dots & q_m & q_m^{-1} \\
0 & 0 & 0 & 0 & \dots\dots & q_m^{-1} & q_m \\
0 & 0 & 0 & 0 & \dots\dots & \mathbf{x_n} & p_n \\
0 & 0 & 0 & 0 & \dots\dots & 1 & \partial_{\mathbf{x_n}}
\end{pmatrix}$$

We count in such a way that in the matrix above $i = \frac{1}{2}(n-1)(n-2) + 1,\ n \geq 2$.

This deformation has the following properties:

(1) $\forall n \geq 1$ $\Delta_n(\underline{p}, \underline{q})$ is a simple Noetherian domain with the PBW basis

$$\{x_1^{\alpha_1}\partial_{x_1}^{\alpha_{n+1}} \dots x_n^{\alpha_n}\partial_{x_n}^{\alpha_{2n}} \mid \alpha \in \mathbb{N}^{2n}\},$$

which can be easily rewritten as the classical PBW basis.

(2) Let $1 \leq s < m$ and $\nu(k) := \frac{1}{2}k(k-1)$.
Define the index set

$$I = \bigoplus_{t=0}^{m-s-1} I_t, \text{ where } I_t = \{\nu(s+t)+1, \dots, \nu(s+t+1)-t\}\ \forall\ 0 \leq t \leq m-s-1.$$

Set $\underline{q'} := \{q_i \mid i \in I\}$ and $\underline{q''} := \{q_i \mid \nu(s+1)+1 \leq i \leq m\} \setminus \underline{q'}$. Then, if $\underline{q'} = \underline{1}$, we have

$$\Delta_n(\underline{p}, \underline{q}) = \Delta_s(p_1, \dots, p_s; q_1, \dots, q_{\nu(s)}) \otimes_{\mathbb{K}(\underline{p},\underline{q}\setminus\underline{q'})} \Delta_{m-s}(p_{s+1}, \dots, p_n; \underline{q''}).$$

In particular,

$$\underline{q'} := (q_i, \dots, q_m) = \underline{1} \ \Rightarrow\ \Delta_n(\underline{p}, \underline{q}) = \Delta_{n-1}(\underline{p} \setminus p_n, \underline{q} \setminus \underline{q'}) \otimes_{\mathbb{K}(\underline{p})} A_1(p_n),$$
$$\underline{q} = \underline{1} \ \Rightarrow\ \Delta_n(\underline{p}, \underline{q}) = A_1(p_1) \otimes_{\mathbb{K}(\underline{p})} \dots \otimes_{\mathbb{K}(\underline{p})} A_1(p_n),$$
$$\underline{q} = \underline{1}, \underline{p} = \underline{1} \ \Rightarrow\ \Delta_n(\underline{p}, \underline{q}) = A_n = \bigotimes_{i=1}^{n} A_1.$$

Proof. We have to show that Δ is a G–algebra. It becomes clear from the definition we have to show only that the non–degeneracy conditions vanish. We do it by induction on n. $\Delta_1(p_1)$ is a q–Weyl algebra $A_1(p_1)$, and the theorem is obviously true for it. Now assume that Δ_{n-1} is a G–algebra.

We construct Δ_n from Δ_{n-1} with a single parameter p_n and $n-1$ block parameters $q_i, \ldots, q_m$. We have to show that the following equalities hold:

$$\partial_{x_n} * (y_k * y_l) - (\partial_{x_n} * y_k) * y_l = 0, \quad x_n * (y_k * y_l) - (x_n * y_k) * y_l = 0,$$

$$\partial_{x_n} * (x_n * y_k) - (\partial_{x_n} * x_n) * y_k = 0.$$

where (y_k, y_l) are pairs of generators of Δ_{n-1} with $y_k > y_l$ and $*$ is the multiplication on Δ. In fact it suffices to show that $\forall 1 \leq k < n$

$$\begin{aligned}
\partial_{x_n} * (\partial_{x_k} * x_k) - (\partial_{x_n} * \partial_{x_k}) * x_k = 0,\\
x_n * (\partial_{x_k} * x_k) - (x_n * \partial_{x_k}) * x_k = 0,\\
\partial_{x_n} * (x_n * x_k) - (\partial_{x_n} * x_n) * x_k = 0,\\
\partial_{x_n} * (x_n * \partial_{x_k}) - (\partial_{x_n} * x_n) * \partial_{x_k} = 0.
\end{aligned}$$

Let us prove the first equality (the other will follow analogously). Denote by R the polynomial $p_k x_k \partial_{x_k} \partial_{x_n} + \partial_{x_n}$. Then

$$\partial_{x_n} * (\partial_{x_k} * x_k) = \partial_{x_n} * (p_k x_k \partial_{x_k} + 1) = p_k q_{i+k-1}^{-1} x_k \partial_{x_n} \partial_{x_k} + \partial_{x_n} = R,$$

$$(\partial_{x_n} * \partial_{x_k}) * x_k = q_{i+k-1}^{-1} \partial_{x_k} (\partial_{x_n} * x_k) = q_{i+k-1} \partial_{x_k} q_{i+k-1}^{-1} x_k \partial_{x_n} = R,$$

where $i = \frac{1}{2}(n-1)(n-2) + 1$. The claim follows.

The properties of Δ one can read directly from the encoding we use.

□

7.2. Quadratic Algebras in 3 Variables.

Consider a class of G–algebras in n variables which relations are homogeneous of degree 2. We call these algebras *quadratic G–algebras*. Let us have a look at the case when $n = 3$.

Assume that the relations are given in terms of non–deformed commutators (i.e. $c_{ij} = 1 \; \forall j > i$). Let us fix an ordering, say, `Dp` (degree lexicographical ordering) with $x > y > z$. Then the relations of quadratic algebra A, satisfying the ordering condition from the definition of G–algebras, should be of the following form:

$$\begin{aligned}
yx &= xy + a_1 xz + a_2 y^2 + a_3 yz + \xi_1 z^2,\\
zx &= xz + \xi_2 y^2 + a_5 yz + a_6 z^2,\\
zy &= yz + a_4 z^2.
\end{aligned}$$

Computing the non–degeneracy conditions, we construct the ideal

$$\mathcal{I} = \langle 2a_2 a_4 + a_1 a_5 - a_4 a_5,\; 2a_2 a_4^2 - a_4^2 a_5 + a_1 a_6 + a_3 a_4 \rangle.$$

We see that the non–degeneracy conditions do not depend on ξ_1, ξ_2 (this fact motivates us to treat ξ_1, ξ_2 as generic parameters of a different nature than the a_i's) so we are working further within the ring $\mathbb{K}[a_1, \ldots, a_6]$. Moreover, the ideal $\mathcal{I}$ is a radical ideal. Computing the primary decomposition, we get two associated prime ideals $\mathcal{I}_1 = \langle 2a_2a_4 + a_1a_5 - a_4a_5, a_1a_5^2 - a_3a_5 + 2a_2a_6 - a_5a_6, a_1a_4a_5 - a_3a_4 - a_1a_6\rangle$ and $\mathcal{I}_2 = \langle a_1, a_4\rangle$, corresponding to components $\mathcal{V}_1$ and $\mathcal{V}_2$ of the 4-dimensional variety $\mathcal{V}(\mathcal{I}) = \mathcal{V}_1 \cup \mathcal{V}_2$.

Let us start with the component $\mathcal{V}_2$. Since $a_1 = a_4 = 0$ on it , consider the subalgebra $H = \mathbb{K}\langle y, z \mid zy = yz\rangle$. In fact we may call the algebra A "solvable", since then $[H, x] \in H$, $[A, A] = H$ and $[[A, A], A] = 0$. So, the component $\mathcal{V}_2$ gives us the family of "solvable" algebras, depending on six arbitrary parameters $a_2, a_3, a_5, a_6, \xi_1, \xi_2$ having the following relations:

$$yx = xy + a_2y^2 + a_3yz + \xi_1z^2, \quad zx = xz + \xi_2y^2 + a_5yz + a_6z^2, \quad zy = yz.$$

Note that such solvable algebras admit a natural presentation as an Ore extension $\mathbb{K}[y, z][x; \mathbb{I}\mathrm{d}, \delta]$ with

$$\delta = -(a_2y^2 + a_3yz + \xi_1z^2)\tfrac{\partial}{\partial y} - (\xi_2y^2 + a_5yz + a_6z^2)\tfrac{\partial}{\partial z}.$$

For the analysis of the rest of conditions, we use the decomposition $\mathcal{V}(\mathcal{I}) = \mathcal{V}_2 \cup \mathcal{V}_1 = \mathcal{V}_2 \oplus (\mathcal{V}_1 \setminus \mathcal{V}_2)$. On the latter semi–algebraic set the parameters are algebraically dependent, so we can express, for example,

$$a_2 = \frac{1}{2}(1 - \frac{a_1}{a_4})a_5, \quad a_6 = a_4(a_5 - \frac{a_3}{a_1}).$$

We see, that the family of algebras, arising from $\mathcal{V}_1 \setminus \mathcal{V}_2$ depends on two nonzero parameters (here a_1, a_4) and four arbitrary parameters (here a_3, a_5, ξ_1, ξ_2). Moreover, for generic ξ_1, ξ_2 there are no solvable algebras in this class. However, setting $\xi_2 = 0$ and $a_5 = a_3 = 0$ implies that $a_2 = a_6 = 0$ and there is a family of algebras with relations

$$zx = xz, \ zy = yz + a_4z^2, \ yx = xy + a_1xz + \xi_1z^2,$$

which are solvable algebras for the subalgebra $H = \mathbb{K}[x, z]$ (since then $[H, y] \in H$, $[A, A] = H$ and $[[A, A], A] = 0$) and can be realized as an Ore extension $\mathbb{K}[x, z][y; \mathbb{I}\mathrm{d}, \delta]$ with

$$\delta = -a_4z^2\tfrac{\partial}{\partial y} + (a_1xz + \xi_1z^2)\tfrac{\partial}{\partial z}.$$

In order to complete the list of quadratic G–algebras in 3 variables, realizable as Ore extensions, we note, that taking H to be the proper subalgebra, generated by x, y implies that $a_1 = a_3 = 0$ and $\xi_1 = 0$. Then $H = \mathbb{K}\langle x, y \mid yx = xy + a_2y^2\rangle$ and $A \cong H[z; \mathbb{I}\mathrm{d}, \delta]$ if additionally $a_4 = a_5 = a_6 = 0$, then $\delta = \xi_2y^2\frac{\partial}{\partial x}$.

This algebra belongs to algebras, associated with component $\mathcal{V}_1$ and does exist (like the previous one) only for non–generic value of ξ (since ξ_1 has to be zero).

REMARK 7.4. We have divided quadratic G–algebras in 3 variables into two distinct groups. The members of the first group are "solvable" algebras, where each algebra has a natural commutative subalgebra and is easily realizable as an Ore extension.

The variables of algebras from the second group are mutually non–commutative. Moreover, since by construction $a_1 \neq 0$, the relation of y and x will always have a term a_1xz, what makes the realization of such an algebra as an Ore extension impossible. Note, that all these algebras share the natural subalgebra $S = \mathbb{K}(a_4)\langle y, z \mid zy = yz + a_4z^2\rangle$, which is not a "solvable" algebra, too. The connection of a "solvability" with the genericity of parameters ξ_1, ξ_2 may have interesting consequences in further investigations.

7.3. Witten's Deformation of $U(\mathfrak{sl}_2)$.

E. Witten introduced and studied a 7–parameter deformation of the universal enveloping algebra $U(\mathfrak{sl}_2)$. Witten's deformation is a unital associative algebra over a field $\mathbb{K}$ (which is assumed to be of characteristic 0), depending on a 7–tuple of parameters $\underline{\xi} = (\xi_1, \ldots, \xi_7)$. It is the algebra $\mathbb{K}(\xi_1, \ldots, \xi_7)\langle x, y, z\rangle$ subject to relations

$$xz - \xi_1 zx = \xi_2 x,\ zy - \xi_3 yz = \xi_4 y,\ yx - \xi_5 xy = \xi_6 z^2 + \xi_7 z.$$

The resulting algebra is denoted by $W(\underline{\xi})$; in the sequel, we assume that $\xi_1\xi_3\xi_5 \neq 0$. As an admissible ordering one can take `Dp` (degree lexicographical ordering) with $x > y > z$.

The structural analysis of $W(\underline{\xi})$ was done in [**8**]. We are going to use the non–degeneracy conditions to carry such an analysis on our own.

The only non–degeneracy condition equals $(\xi_3\xi_6 - \xi_1\xi_6)z^3 + (\xi_1\xi_4\xi_5 - \xi_2\xi_3\xi_5 + \xi_2\xi_5 - \xi_4\xi_5)xy + (\xi_1\xi_4\xi_6 - \xi_1\xi_7 - \xi_2\xi_3\xi_6 + \xi_3\xi_7)z^2 + (\xi_1\xi_4\xi_7 - \xi_2\xi_3\xi_7)z$, but we better write it in a factorized form,

$$((\xi_3 - \xi_1)z + \xi_1\xi_4 - \xi_2\xi_3)(\xi_6 z^2 + \xi_7 z) + \xi_5(\xi_1\xi_4 - \xi_2\xi_3 + \xi_2 - \xi_4)xy.$$

So, $W(\underline{\xi})$ is a G–algebra if and only if for all values of the parameters $\underline{\xi}$ the polynomial above vanishes.

Let $\mathcal{V}_1 \in \mathbb{A}^7_{\mathbb{K}}$ be the variety, corresponding to the radical ideal $\langle \xi_1 - \xi_3, \xi_2 - \xi_4\rangle$. Denoting $\xi_1 = \xi_3 =: q$, we obtain the family of algebras

$$\mathbf{W_1}(\underline{\xi}):\ zx = q^{-1}xz - q^{-1}\xi_2 x,\ zy = qyz + \xi_2 y,\ yx = \xi_5 xy + \xi_6 z^2 + \xi_7 z,$$

depending on $\{q, \xi_2, \xi_5\} \subset \mathbb{K}^*$ and $\{\xi_6, \xi_7\} \subset \mathbb{K}$.

There is another interesting family of algebras, which are related to the Witten's deformation, namely *conformal* $\mathfrak{sl}_2$*–algebras*. These algebras are generated by x, y, z subject to 3–parameter relations

$$zx = \frac{1}{a}xz - \frac{1}{a}x,\ zy = ayz + y,\ yx = cxy + bz^2 + z.$$

These algebras are G–algebras for any $\{a, c\} \subset \mathbb{K}^*$ and $b \in \mathbb{K}$.

We see, that $W_1(\underline{\xi})$ is isomorphic to the conformal $\mathfrak{sl}_2$–algebra if and only if $\xi_2\xi_7 = 1$. Then $a = q$, $b = \frac{\xi_6}{\xi_7^2}$ and $c = \xi_5$.

The remaining possibilities are described by the semi–algebraic set

$$\xi_4(\xi_1 - 1) + \xi_2(1 - \xi_3) = 0,\ \xi_6 = \xi_7 = 0,\ \xi_1 \neq \xi_3,\ \xi_2 \neq \xi_4.$$

If $\xi_1 = 1$, we have $\xi_2 = 0$ and a family of algebras

$$\mathbf{W_2^{(1)}}(\underline{\xi}):\ zx = xz,\ zy = \xi_3 yz + \xi_4 y,\ yx = \xi_5 xy,$$

depending on $\{\xi_3 \neq 1, \xi_4, \xi_5\} \subset \mathbb{K}^*$.

Similarly, $\xi_3 = 1 \implies \xi_4 = 0$ and there are algebras

$$\mathbf{W_2^{(3)}}(\underline{\xi}):\ zx = \frac{1}{\xi_1}xz - \frac{\xi_2}{\xi_1}x,\ zy = yz,\ yx = \xi_5 xy,$$

depending on $\{\xi_1 \neq 1, \xi_2, \xi_5\} \subset \mathbb{K}^*$.

We see, that $W_2^{(1)}(\underline{\xi}) \cong W_2^{(3)}(\underline{\xi})$ in a natural way.

Algebras of the type $\mathbf{W_2}$ contain both commutative subalgebra in 2 variables and a subalgebra, isomorphic to a quantum plane. They can be realized as polynomial Ore extensions over these subalgebras. It cab be, for instance, an Ore extension of a quantum plane $\mathbb{K}\langle x, y \mid yx = \xi_5 xy\rangle$ by z with two parameters.

In the last two cases ($\xi_1 \neq 1 \neq \xi_3$), we have families

$$\mathbf{W_3^{(2)}}(\underline{\xi}):\ zx = \frac{1}{\xi_1}xz - \frac{\xi_2}{\xi_1}x,\ zy = \xi_3 yz + \xi_2\frac{(1-\xi_3)}{(1-\xi_1)}y,\ yx = \xi_5 xy,$$

depending on $\{\xi_1, \xi_3 \mid \xi_1 \neq \xi_3\} \subset \mathbb{K}^* \setminus \{1\}$, $\xi_5 \in \mathbb{K}^*$ and $\xi_2 \in \mathbb{K}$ and

$$\mathbf{W_3^{(4)}}(\underline{\xi}):\ zx = \frac{1}{\xi_1}xz - \frac{\xi_4}{\xi_1}\frac{(1-\xi_1)}{(1-\xi_3)}x,\ zy = \xi_3 yz + \xi_4 y,\ yx = \xi_5 xy,$$

for $\{\xi_1, \xi_3 \mid \xi_1 \neq \xi_3\} \subset \mathbb{K}^* \setminus \{1\}$, $\xi_5 \in \mathbb{K}^*$ and $\xi_4 \in \mathbb{K}$.

Again, we observe a natural isomorphism $W_3^{(2)} \cong W_3^{(4)}$.

As we can see, we are easily able to give a detailed description of different classes of Witten's deformation, even more detailed than in [**8**]. The conditions on parameters are more natural as well as the division in the subclasses.

REMARK 7.5. An important conclusion should be drawn from the analysis above: Witten's original construction involved three independent quantum parameters $\{\xi_1, \xi_3, \xi_5\}$, which induce three *pairwise different* q–commutators, namely

$$[x,z]_{\xi_1} = \xi_2 x,\ [z,y]_{\xi_3} = \xi_4 y,\ [y,x]_{\xi_5} = \xi_6 z^2 + \xi_7 z.$$

Here, we denote $[x,y]_q := xy - qyx$, which moreover satisfies the property $[x,y]_q = -[y,x]_{q^{-1}}$. We say then, that the q–commutators $[\cdot,\cdot]_q$ and $[\cdot,\cdot]_{q^{-1}}$ are equivalent.

In the analysis above we have shown, that the only possible G–algebra with all three q–commutators being nonzero is of the form

$$\mathbf{W_1}(\underline{\xi}):\ [x,z]_q = \xi_2 x,\ [z,y]_q = \xi_2 y,\ [y,x]_{\xi_5} = \xi_6 z^2 + \xi_7 z,$$

where $\{q, \xi_2, \xi_5\} \subset \mathbb{K}^*$ and $\{\xi_6, \xi_7\} \subset \mathbb{K}$. But we see, that in this case, at most **two** different q–commutators (in our case, $[\cdot,\cdot]_{\xi_5}$ and $[\cdot,\cdot]_q$) are indeed possible. Three different commutators appear only in algebras, where at least two variables constitute a quasi–commutative subalgebra.

7.4. Diffusion Algebras.

A *diffusion algebra* ([**40**]) is generated by $\{D_i, 1 \le i \le n\}$ over a field $\mathbb{K}$ subject to the relations:

$$c_{ij} D_i D_j - c_{ji} D_j D_i = x_j D_i - x_i D_j,\ \forall i < j$$

where $c_{ij} \ge 0$ and x_i are coefficients from the field (for easier comparison, we use the notations of original article).

We will assume that $\forall i,j\ c_{ij} > 0$ and therefore concentrate our attention on revealing the G–algebras among diffusion algebras (the authors of the article [**40**] studied all the possible diffusion algebras including degenerate ones).

For the diffusion algebras we compute the non–degeneracy conditions for a fixed triple (i,j,k) in a similar way as we did for G–algebras. After computing the primary decomposition, we get eight components and we do the classification of algebras, following the approach from [**40**]. Each component of the primary decomposition corresponds to a different form of the algebra. One component corresponds to type **A**, three to type **B**, three to type **C** and one component to type **D**.

A type : every x_m is nonzero. Then there are relations
$c_{jk} = c_{ki} = c_{ik} = c_{ji} = c_{ij} = c_{kj}$, that is, we obtain universal

enveloping algebras of Lie algebras with relations

$$[D_i, D_j] = \frac{x_j}{c_{ij}} D_i - \frac{x_i}{c_{ij}} D_j.$$

B type : one of x_m is equal to zero. In the case $x_i = 0$, we have

$$c_{ki} = c_{ij},\ c_{ik} - c_{ki} = c_{jk} - c_{kj} = c_{ji} - c_{ij} =: c$$

And the relations are the following:

$$\begin{aligned} c_{ij} D_i D_j - (c_{ij} + c) D_j D_i &= x_j D_i, \\ c_{jk} D_j D_k - (c_{jk} - c) D_k D_j &= x_k D_j - x_j D_k, \\ (c_{ij} + c) D_i D_k - c_{ij} D_k D_i &= x_k D_i. \end{aligned}$$

The cases $x_j = 0$ and $x_k = 0$ are handled in an analogous way.

C type : one of x_m is nonzero. Let $x_j = 0, x_k = 0$.
Then $c_{ij} - c_{ji} = c_{ik} - c_{ki} =: c$, and there are relations

$$\begin{aligned} c_{ij} D_i D_j - (c_{ij} - c) D_j D_i &= -x_i D_j, \\ c_{jk} D_j D_k - c_{kj} D_k D_j &= 0, \\ c_{ik} D_i D_k - (c_{ik} - c) D_k D_i &= -x_i D_k. \end{aligned}$$

The cases $x_i = x_k = 0$ and $x_i = x_j = 0$ are done analogously.

D type : every x_m is equal to zero. There are no additional constraints on c_{ij} and it is not surprising, since this type consists of quasi–commutative algebras with relations

$$D_l D_m = \frac{c_{ml}}{c_{lm}} D_m D_l, \quad (l, m) \in \{(i, j), (i, k), (j, k)\}$$

As we can see, we obtained the same classification of G–algebras among the diffusion algebras as in [**40**]. The advantage of our approach lies in the automation of the process, in particular, we can consider more variables and achieve the classification by using the computer algebra methods only. Thus, our approach is limited only by the overall performance of the computing facilities.

7.5. Completion of Relations.

When speaking on algebras, given by generators and relations, nonzero non–degeneracy conditions (if there are some) complete the set of relations for algebras, close to G–algebras to the full set of relations (by means of Gröbner bases in free algebras). We illustrate the use of non–degeneracy conditions in such situation.

The concrete application, we are going to describe, arisen from the investigation of an analogue to the Hall–Ringel algebra for representations of partially ordered sets, [**47**] and was asked to solve by Prof. Yu. Drozd.

We start with three variables $\{e_0, e_1, e_2\}$ and some relations between them. Namely, the pairs (e_0, e_1) and (e_0, e_2), satisfying Serre's relations:

$$[e_i, [e_i, e_j]] = e_i^2 e_j - 2e_i e_j e_i + e_j e_i^2 = 0 \text{ for each pair } (e_i, e_j).$$

Introducing new variables by defining $[e_0, e_1] = e_{01}$ and $[e_0, e_2] = e_{02}$, we obtain two non–degenerate algebras:

$$E_{01} := \mathbb{K}\langle e_0, e_1, e_{01} \mid e_1e_0 = e_0e_1 - e_{01},\ e_{01}e_0 = e_0e_{01},\ e_{01}e_1 = e_1e_{01}\rangle,$$

$$E_{02} := \mathbb{K}\langle e_0, e_2, e_{02} \mid e_2e_0 = e_0e_2 - e_{02},\ e_{02}e_0 = e_0e_{02},\ e_{02}e_2 = e_2e_{02}\rangle.$$

The third pair (e_1, e_2) satisfies a kind of "negative" Serre's relation:

$$\{e_1, \{e_1, e_2\}\} = e_1^2e_2 + 2e_1e_2e_1 + e_2e_1^2 = 0, \text{ where } \{a, b\} = ab + ba.$$

We define a new variable $\{e_1, e_2\} = e_{12}$ and obtain a non–degenerate algebra

$$E_{12} := \mathbb{K}\langle e_1, e_2, e_{12} \mid e_2e_1 = -e_1e_2 + e_{12},\ e_{12}e_1 = -e_1e_{12},\ e_{12}e_2 = -e_2e_{12}\rangle.$$

Problem: Find all the G–algebras, generated by the set of variables $\{e_0, e_1, e_2, e_{01}, e_{02}, e_{12}\}$, satisfying the relations of E_{01}, E_{02}, E_{12}. In other words, given the incomplete set of relations, examine its possible non–degenerate G–completions.

Here is the matrix encoding of the starting situation, where $*$ denotes yet undetermined values:

$$\begin{pmatrix} \mathbf{e_0} & 1 & 1 & 1 & 1 & * \\ -e_{01} & \mathbf{e_1} & -1 & 1 & * & -1 \\ -e_{02} & e_{12} & \mathbf{e_2} & * & 1 & -1 \\ 0 & 0 & * & \mathbf{e_{01}} & * & * \\ 0 & * & 0 & * & \mathbf{e_{02}} & * \\ * & 0 & 0 & * & * & \mathbf{e_{12}} \end{pmatrix}$$

We are computing all the non–degeneracy conditions between the triples of e_i and extract new relations from them.

(1) $(e_0, e_1, e_2, \cdot, \cdot, \cdot)$: $\{e_{01}, e_2\} + \{e_{02}, e_1\} = [e_0, e_{12}]$.
We introduce new variables $\{e_{01}, e_2\} := d$ and $\{e_{02}, e_1\} := q$. Hence there are following new relations: $e_{01}e_2 = -e_2e_{01} + d$, $e_{02}e_1 = -e_1e_{02} + q$ and $e_{12}e_0 = e_0e_{12} - (d + q)$.

(2) $(e_0, e_1, \cdot, e_{01}, \cdot, \cdot)$: 0 from the construction.

(3) $(e_0, e_1, \cdot, \cdot, e_{02}, \cdot)$: (new) $e_{02}e_{01} = -e_{01}e_{02} + [e_0, q]$.

(4) $(e_0, e_1, \cdot, \cdot, \cdot, e_{12})$: (tautological) $e_{02}e_{01} = -e_{01}e_{02} + \{e_1, d + q\}$. Together with 3) it gives us (new) $[e_0, q] = \{e_1, d + q\}$.

(5) $(e_0,\cdot,e_2,e_{01},\cdot,\cdot)$: (tautological) $e_{02}e_{01} = -e_{01}e_{02} + [e_0, d]$. Comparing with 3), we have (new) $[e_0, d-q] = 0$.
(6) $(e_0,\cdot,e_2,\cdot,e_{02},\cdot)$: 0 from the construction.
(7) $(e_0,\cdot,e_2,\cdot,\cdot,e_{12})$: (new) $e_{12}e_{02} = -e_{02}e_{12} - \{e_2, d+q\}$.
(8) $(\cdot,e_1,e_2,e_{01},\cdot,\cdot)$: (new) $e_{12}e_{01} = -e_{01}e_{12} + \{e_1, d\}$.
(9) $(\cdot,e_1,e_2,\cdot,e_{02},\cdot)$: (tautological) $e_{12}e_{02} = -e_{02}e_{12} + \{e_2, q\}$. With the 7) we have (new) $\{e_2, d+2q\} = 0$.
(10) $(\cdot,e_1,e_2,\cdot,\cdot,e_{12})$: 0 from the construction.
(11) $(e_0,\cdot,\cdot,e_{01},e_{02},\cdot)$: (new) $[e_0,[e_0,d]] = 0$ (relation of Serre's type). Note, that it follows also from 5), since $[e_{01}e_{02}, e_0] = 0$.
(12) $(e_0,\cdot,\cdot,e_{01},\cdot,e_{12})$: (new) $\{e_{01}, d+q\} = [e_0, \{e_1, d\}]$.
(13) $(e_0,\cdot,\cdot,\cdot,e_{02},e_{12})$: (new) $\{e_{02}, d+q\} = [e_0, \{e_2, q\}]$.
(14) $(\cdot,e_1,\cdot,e_{01},e_{02},\cdot)$: (new) $\{e_{01}, q\} = \{e_1, [e_0, q]\}$ (together with 12 produces tautology).
(15) $(\cdot,e_1,\cdot,e_{01},\cdot,e_{12})$: (new) $\{e_1, \{e_1, d\}\} = 0$.
(16) $(\cdot,e_1,\cdot,\cdot,e_{02},e_{12})$: (new) $-[e_{12}, q] = [e_1, \{e_2, q\}]$.
(17) $(\cdot,\cdot,e_2,e_{01},e_{02},\cdot)$: (new) $\{e_{02}, d\} = \{e_2, [e_0, d]\}$ (together with 13 produces tautology).
(18) $(\cdot,\cdot,e_2,e_{01},\cdot,e_{12})$: (new) $-[e_{12}, d] = [e_2, \{e_1, d\}]$.
(19) $(\cdot,\cdot,e_2,\cdot,e_{02},e_{12})$: (new) $\{e_2, \{e_2, q\}\} = 0$.
(20) $(\cdot,\cdot,\cdot,e_{01},e_{02},e_{12})$: (new)
$[e_{01}, \{e_2, q\}] + [e_{12}, [e_0, d]] + [e_{02}, \{e_1, d\}] = 0$.

We see, that there are many relations between the generators resembling G–algebra relations and some more. In fact, the following lemma holds.

LEMMA 7.6. The only G–algebras in 6 variables, satisfying the relations of E_{01}, E_{02}, E_{12} are the algebras $\{\mathcal{E}_k, k \in \mathbb{K}\}$ with the encoding

$$\begin{pmatrix} \mathbf{e_0} & 1 & 1 & 1 & 1 & 1 \\ -e_{01} & \mathbf{e_1} & -1 & 1 & -1 & -1 \\ -e_{02} & e_{12} & \mathbf{e_2} & -1 & 1 & -1 \\ 0 & 0 & k\cdot e_{12} & \mathbf{e_{01}} & -1 & -1 \\ 0 & -k\cdot e_{12} & 0 & 0 & \mathbf{e_{02}} & -1 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{e_{12}} \end{pmatrix}$$

PROOF. In fact, after computing all the non–degeneracy conditions, we have the algebra with the relations, encoded in the following matrix

$$\begin{pmatrix} \mathbf{e_0} & 1 & 1 & 1 & 1 & 1 \\ -e_{01} & \mathbf{e_1} & -1 & 1 & -1 & -1 \\ -e_{02} & e_{12} & \mathbf{e_2} & -1 & 1 & -1 \\ 0 & 0 & d & \mathbf{e_{01}} & -1 & -1 \\ 0 & q & 0 & [e_0,d] & \mathbf{e_{02}} & -1 \\ -(d+q) & 0 & 0 & \{e_1,d\} & \{e_2,q\} & \mathbf{e_{12}} \end{pmatrix}$$

And there are even more relations, which d, q should fulfill, namely

- $[e_0,[e_0,d]]=0, \quad [e_0,d]=[e_0,q]$,
- $[e_0,q]=\{e_1,d+q\}, \quad \{e_2,d+2q\}=0$,
- $\{e_{01},d+q\}=[e_0,\{e_1,d\}], \quad \{e_{02},d+q\}=[e_0,\{e_2,q\}]$,
- $\{e_{01},q\}=\{e_1,[e_0,q]\}, \quad \{e_{02},d\}=\{e_2,[e_0,d]\}$,
- $\{e_1,\{e_1,d\}\}=0, \quad \{e_2,\{e_2,q\}\}=0$,
- $[e_{12},q]=-[e_1,\{e_2,q\}], \quad [e_{12},d]=-[e_2,\{e_1,d\}]$,
- $[e_{01},\{e_2,q\}]+[e_{12},[e_0,d]]+[e_{02},\{e_1,d\}]=0$.

First of all, we show that $d=0 \Leftrightarrow q=0$, hence d, q are dependent in a special way. In particular, there is no sense introducing them as new variables.

Let $d=0$. Then $[e_0,q]=\{e_1,q\}=\{e_2,q\}=\{e_{01},q\}=\{e_{02},q\}=0$ and $[e_{12},q]=0$, what could be true if and only if $q=0$. The implication $q=0 \Rightarrow d=0$ holds similarly. Note, that for $d=q=0$ we obtain a G–algebra.

Now let $d\neq 0$. Then $\{e_2,d+2q\}=0 \Rightarrow d+2q=ke_{12}$ for some $k\in\mathbb{K}$. Then $[e_0,d+2q]=3[e_0,d]=3k[e_0,e_{12}]=3k(d+q)$. By Serre's relation, $[e_0,[e_0,d]]=0$ implies $[e_0,[e_0,d+2q]]=0$, since $[e_0,d]=[e_0,q]$. On the other hand, $[e_0,[e_0,d+2q]]=3k[e_0,d+q]=6k[e_0,d]$. Hence, either $k=0$ or $[e_0,d]=0=[e_0,q]$.

If $k=0$, $d=-2q$ and $[e_{12},q]=0$. This could only happen if $q=te_{12}$ what contradicts other consequences like $[e_0,q]=0, \{e_1,q\}=0$.

Now, $d\neq 0, d\neq -2q$ and $[e_0,d]=0=[e_0,q]$. Then we have $\{e_1,d+q\}=0$, what is satisfied only by $d+q=te_{12}$ for some $t\in\mathbb{K}$. Hence, $q=(k-t)e_{12}$ and $0=[e_0,q]=(k-t)[e_0,e_{12}]=(k-t)(d+q)$. So, we have either $k=t$ and $q=0$ (a contradiction to the assumption) or $d+q=0$.

With the latter, we have $[e_0,d]=\{e_1,d\}=\{e_2,d\}=\{e_{01},d\}=$
$=\{e_{02},d\}=0$ and $[e_{12},d]=0$. The only possible value of d, satisfying these is $d=ke_{12}$ (since $\{e_0,e_{12}\}=-(d+q)=0$). □

REMARK 7.7. One of the possible approaches to problems like the described one is computing the Gröbner basis of the ideal, generated by the

initial set of relations, in a free algebra. However, the universality of this recipe turns to be inefficient in some concrete situations. In our example, without introducing "shortcuts" for d and q we would obtain a basis of infinite length. Even our definition of d and q as skew–commutators has been motivated by the condensed form of the non–degeneracy condition of the first explored triple.

These observations show that the full automation of such procedures is, unfortunately, almost impossible. On the other hand, combining both theoretical knowledge and human intuition together with careful and fast implementation produces such interesting and somehow unexpected results like the previous lemma.

8. Conclusion and Future Work

We have shown the nature of non–degeneracy conditions and their interplay with the PBW basis property in a wide class of algebras. By combining both non–commutative and commutative computational methods we are able to classify families of parametric algebras having PBW basis and being Noetherian domains.

We have seen how the non–degeneracy conditions can help us to "normalize" the set of relations of a non–commutative algebra: minimize it like in 2.1 or enlarge by completing in 7.5.

We used the non–degeneracy conditions for revealing realizability of parametric algebras as Ore extensions in 7.2 and 7.3.

It turns out to be very useful to perform computations like before, that is, we need *combined* (i.e. both commutative and non–commutative) applications of Gröbner bases at the same time. As far as we know, our system SINGULAR:PLURAL is the only modern system allowing the user to do such combined computations on a substantially large class of non–commutative algebras.

One of the most interesting perspectives is the investigation of the role of the non–degeneracy conditions for *quantum Lie algebras*, studied by Delius ([**22**]). A generalization of Jacobi identities for such algebras is needed but is not discovered yet. In the classical case of Lie algebras, one utilizes properties of the Lie bracket a lot. The problem in the quantum case, which has been already partially discussed in Example 7.3 is the following: one has to introduce several q–commutators of the form $[a,b]_{c_{ij}} := ab - c_{ij}ba$ for $c_{ij} \in \mathbb{K}$, which are connected to each other in some way (we have shown a variation of this principle in Remark 7.5). However, this treatment is

not really uniform and causes several difficulties. Nevertheless it should be investigated further.

Concerning algebras of linear operators, described with plenty of examples in Section 6, we should note that despite the impression one gets from the impressive list of operators leading to G–algebras, there are important linear operators, which do not behave so well.

For example, the inverse to the partial differentiation ∂_x can be only one–sided. That is, the operator of indefinite partial integration $\int_x : C \to C$, $\int_x(g) = \int g \ dx$ is a right inverse to ∂_x, since $\forall g \in C$, $(\partial_x \circ \int_x)(g) = \frac{\partial}{\partial x}(\int g \ dx) = g$. But it is not a left inverse, since $\int \frac{\partial g}{\partial x} \ dx = g + c$ for $c \in C$ such, that $\frac{\partial c}{\partial x} = 0$, hence $\partial_x \circ \int_x = \int_x \circ \partial_x + c$ for indefinite c. Since $\ker \partial_x$ is a subalgebra of C, we cannot even restrict ourself to an algebra smaller than C and where c is identically zero. However, the authors of [**68**] propose several possibilities to deal with such operator equations in some specified (but still general enough) setting. They end up with an algebra, which is not a G–algebra but still enjoys some nice properties. A further investigation of operators, which lead to operator G–algebras is needed and will be developed further.

CHAPTER 2

Gröbner bases in G–algebras

> "Aren't you acquainted with the red–faced person with three eyes and a necklace from skulls?" – he asked, – "A man, who dances between fires? Eh? A tall one, who likes to brandish with two hooks?"
> "Perhaps I do" – I anwered politely, – "but I cannot get who's the person you're talking about. You know, the description is too general. So many fit in it".
>
> Viktor Pelevin, *Chapayev and Void*

1. Left Gröbner Bases

From now on we mean by a *module* a left submodule of a free module[1] of a finite rank, if not additionally specified. In this section we present Gröbner bases theory for modules in G–algebras.

We have shown that G–algebras are *close to commutative*, in the sense that on one hand they have similar properties as commutative polynomial algebras (which are G–algebras too). On the other hand G–algebras clearly constitute a generalization of commutative polynomials algebras. We exploit this similarity by dividing the properties and algorithms in G–algebras into two groups: the first one, which could be called *native* consists of methods and properties whose generalization behave in the same way as in the classical commutative case. The objects belonging to the second group depend too much on the commutativity. While working with native objects, we will try to keep the notations and arguments as close to the original commutative theory as possible, taking the book [**35**] as our basis. Not only the way we go through the theory follows this book: the implementation of the computer algebra system Singular:Plural, made on the top of the *commutative* system Singular, profited a lot from using the same language and framework for both the non–commutative and commutative

[1]Here module means a unitary module over some $\mathbb{K}$–algebra.

cases of the theory. We reported on aspects of the implementation in the article [**53**]; we make computational remarks in the text which follows. More questions related to our implementation are discussed in the Chapter 4.

Let $A = \mathbb{K}\langle x_1, \dots, x_n \mid \{x_j x_i = c_{ij} x_i x_j + d_{ij}\}_{1 \le i < j \le n}\rangle$ be a G–algebra over a field $\mathbb{K}$.

DEFINITION 1.1. Let $m_1 = x^\alpha$, $m_2 = x^\beta$ be monomials in A. We say that m_1 **divides** m_2 and denote it by $m_1 | m_2$, if $\alpha_i \le \beta_i \ \forall i = 1 \dots n$.

When A is commutative, there is $p \in \mathrm{Mon}(A)$ such that $m_2 = p \cdot m_1$. Otherwise, it means that m_2 is left reducible by m_1, i.e. there exist $c \in \mathbb{K} \setminus \{0\}$, $p \in \mathrm{Mon}(A)$ and $r \in A$ such that $\mathrm{lm}(r) < m_1$ and $m_2 = c \cdot p \cdot m_1 + r$.

EXAMPLE 1.2. Let us take two exponent vectors $\alpha = (1,1)$ and $\beta = (1,2)$ from $\mathbb{N}^2$. Then in any G–algebra in two generators, say, $\{x, \partial\}$, we have $m_1 := x\partial \mid x\partial^2 =: m_2$, but the division of one by another gives quite different answers in different algebras.

In the commutative polynomial ring $R = \mathbb{K}[x, \partial]$, we have $m_2 = \partial m_1$.

In the first quantized Weyl algebra $A_q = \mathbb{K}(q)\langle x, \partial \mid \partial x = q^2 x\partial + 1\rangle$, we obtain $m_2 = q^{-2} \cdot \partial \cdot m_1 - q^{-2}\partial$.

We extend the notion of a monomial ordering to the free left module $A^r = Ae_1 \oplus \dots \oplus Ae_r$, where $e_i = (0, \dots, 1_i, \dots, 0)$. Since any e_i commutes with any element of A, A^r is indeed a free bimodule. Denote by $\mathbb{N}_r := \{1, \dots, r\}$ the set of components.

We call $x^\alpha e_i \in A^r$ a **monomial (involving component i)**. So, $\mathrm{Mon}(A^r) := \{x^\alpha e_i \mid \alpha \in \mathbb{N}^n, 1 \le i \le r\} \cong \mathbb{N}_r \times \mathbb{N}^n$.

It is quite natural to view the component of a free module as the 0's component of the exponent vector, hence we present a monomial $x^\alpha e_i$ by the vector $\bar{\alpha} := (i, \alpha_1, \dots, \alpha_n) \in \mathbb{N}_r \times \mathbb{N}^n$.

DEFINITION 1.3. Let $<$ be a monomial ordering on A. A **monomial (module) ordering** on A^r is a total ordering $<_m$ on the set of monomials $\mathrm{Mon}(A^r)$, satisfying for all $\alpha, \beta, \gamma \in \mathbb{N}^n$, $1 \le i, j \le r$

(1) $x^\alpha e_i <_m x^\beta e_j \ \Rightarrow \ x^{\alpha+\gamma} e_i <_m x^{\beta+\gamma} e_j$,
(2) $x^\alpha < x^\beta \ \Rightarrow \ x^\alpha e_i <_m x^\beta e_i$.

In the language of exponents, both conditions mean nothing but

(1) $(i, \alpha) \prec_m (j, \beta) \ \Rightarrow \ (i, \alpha + \gamma) \prec_m (j, \beta + \gamma)$,
(2) $\alpha \prec \beta \ \Rightarrow \ (i, \alpha) \prec_m (i, \beta)$.

Hence, the action of $\mathbb{N}^n$ on $\mathbb{N}_r \times \mathbb{N}^n$, $\gamma : (i, \alpha) \mapsto (i, \alpha + \gamma)$ makes $\mathbb{N}_r \times \mathbb{N}^n$ an ideal in the monoid $\mathbb{N}^n$ (or, shortly, a $\mathbb{N}^n$–monoideal) with respect to addition.

Since 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 $x^\beta e_j < x^\alpha e_i$ for any nonzero term $dx^\beta e_j$ of g, we define

$\mathrm{lm}(f)$	$=$	$x^\alpha e_i$	$\in \mathrm{Mon}(A^r)$,	the **leading monomial** of f,
$\mathrm{lc}(f)$	$=$	c_α	$\in \mathbb{K}^*$,	the **leading coefficient** of f,
$\mathrm{lcomp}(f)$	$=$	i	$\in \mathbb{N}_r$,	the **leading component** of f,
$\mathrm{le}(f)$	$=$	(i, α)	$\in \mathbb{N}_r \times \mathbb{N}^n$,	the **leading exponent** of f.

REMARK 1.4. Every monomial ordering $<$ on A can be extended to the monomial module ordering in at least two following ways. We can order components either in ascending or in descending way, denoting it by symbols "C" and "c" respectively. Below, we are using the ascending ordering "C", where $e_1 < e_2 < \ldots$. Then,

$<_m = (C, <)$ is a position over term (**POT**) ordering, if

$$x^\alpha e_i <_{\mathrm{POT}} x^\beta e_j \iff^{def} i < j \text{ or, if } i = j,\ x^\alpha < x^\beta.$$

$<_m = (<, C)$ is a term over position (**TOP**) ordering, if

$$x^\alpha e_i <_{\mathrm{TOP}} x^\beta e_j \iff^{def} x^\alpha < x^\beta \text{ or, if } \alpha = \beta,\ i < j.$$

DEFINITION 1.5. Let S be any subset of A^r.

- We define $\mathcal{L}(S) \subseteq \mathbb{N}_r \times \mathbb{N}^n$ to be a $\mathbb{N}^n$–monoideal, generated by the leading exponents of elements of S, that is
$$\mathcal{L}(S) = {}_{\mathbb{N}^n}\langle (i, \alpha) \mid \exists s \in S,\ \mathrm{le}(s) = (i, \alpha) \rangle.$$
We call $\mathcal{L}(S)$ a **monoideal of leading exponents**. By Dixon's Lemma, $\mathcal{L}(S)$ is finitely generated, i.e. there exist $(i_1, \alpha_1), \ldots, (i_m, \alpha_m) \in \mathbb{N}^r \times \mathbb{N}^n$, such that $\mathcal{L}(S) = {}_{\mathbb{N}^n}\langle (i_1, \alpha_1), \ldots, (i_m, \alpha_m) \rangle$.
- The **span of leading monomials of** S is defined to be the $\mathbb{K}$–vector space, spanned by the set $\{x^\alpha e_i \mid (i, \alpha) \in \mathcal{L}(S)\} \subseteq \mathrm{Mon}(A^r)$. We denote it by $L(S) := {}_{\mathbb{K}}\langle \{x^\alpha e_i \mid (i, \alpha) \in \mathcal{L}(S)\} \rangle \subseteq A^r$.

DEFINITION 1.6. Let $<$ be a monomial ordering on 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 $\mathrm{lm}(g) \mid \mathrm{lm}(f)$.

REMARK 1.7. In general, for $S \subset A^r$, $L(S)$ is just a $\mathbb{K}$–vector subspace of A. Using the filtration by the monomial ordering on A^r, we see that indeed, $L(S)$ can be considered as a $\mathbb{K}$–subspace of $\mathrm{Gr}_<(A)$. The set $\Lambda = \{(i, \alpha) \mid \exists f \in S : \mathrm{lm}(f) = x^\alpha e_i\} \subset \mathbb{N}_r \times \mathbb{N}^n$ is equal to $\mathcal{L}(S)$. Hence,

$$\mathrm{Gr}_<(S) = \bigoplus_{(i,\alpha)\in\Lambda} \mathbb{K} x^\alpha e_i = L(S).$$

It follows that $L(S)$ is a $\mathrm{Gr}_<(A)$–module.

Hence, if $A \cong \mathrm{Gr}_<(A)$ as $\mathbb{K}$–algebras, $L(S)$ is an A–module. It means that for commutative and quasi–commutative algebras, we can define $L(S)$ to be $L'(S) = {}_A\langle\{\mathrm{lm}(f) \mid f \in S\}\rangle$ and call it a *leading submodule* of S (clearly, $L(S) = L'(S)$ as $\mathbb{K}$-vector spaces). Then, a finite set S is a Gröbner basis of ${}_A\langle S\rangle$ if and only if $L'(S) = L'({}_A\langle S\rangle)$.

In order to see that the definition of Gröbner basis via leading submodules cannot be transfered directly to the general non–commutative case, consider the following example.

Take the Weyl algebra $A = \langle x, \partial | \partial x = x\partial + 1\rangle$, the set $S = \{x\partial + 1, x\}$ and the ideal $I = {}_A\langle S\rangle$. I is a proper left ideal, equal to ${}_A\langle x\rangle$ with $\{x\}$ a reduced Gröbner basis of I. Hence, the $\mathbb{K}$–vector spaces $L'(I)$ and ${}_A\langle x\rangle$ are equal, but $L'(S) = {}_A\langle\{x\partial, x\}\rangle = A \cdot 1$.

After the remark, we can give an alternative definition for Gröbner basis in terms of $L(S)$: under assumptions of the Definition 1.6, G is called a **left Gröbner basis** of I if and only if $L(G) = L(I)$ (as vector spaces). The latter is equivalent to the equality of $\mathbb{N}^n$–monoideals $\mathcal{L}(G)$ and $\mathcal{L}(I)$. Yet another definition is given in Remark 1.10.

A subset $S \subset A^r$ is called **minimal**, if $0 \notin S$ and $\mathrm{lm}(s) \notin L(S \setminus \{s\})$ for all $s \in S$. We say that $f \in A^r$ is **(completely) reduced with respect to** $S \subset A^r$, if no monomial of f is contained in $L(S)$. A subset $S \subset A^r$ is called **reduced**, if $0 \notin S$, and if for each $s \in S$, s is reduced with respect to $S \setminus \{s\}$, and, moreover, $s - \mathrm{lc}(s)\,\mathrm{lm}(s)$ is reduced with respect to S. It means that for each $s \in S \subset A^r$, $\mathrm{lm}(s)$ does not divide any monomial of every element of S except itself.

Definition 1.8. Denote by $\mathcal{G}$ the set of all finite ordered subsets of A^r.

(1) A map $\mathrm{NF} : A^r \times \mathcal{G} \to A^r, \quad (f, G) \mapsto \mathrm{NF}(f|G)$,
is called a **(left) normal form** on A^r if, for all $f \in A^r$, $G \in \mathcal{G}$,
(i) $\mathrm{NF}(0 \mid G) = 0$,
(ii) $\mathrm{NF}(f|G) \neq 0 \quad \Rightarrow \quad \mathrm{lm}\big(\mathrm{NF}(f|G)\big) \notin L(G)$,
(iii) $f - \mathrm{NF}(f|G) \in {}_A\langle G\rangle$.
NF is called a **reduced normal form** if $\mathrm{NF}(f|G)$ is reduced with respect to G.

(2) Let $G = \{g_1, \ldots, g_s\} \in \mathcal{G}$. A representation of $f \in {}_A\langle G\rangle$,

$$f = \sum_{i=1}^{s} a_i g_i, \quad a_i \in A,$$

satisfying $\mathrm{lm}(f) \geq \mathrm{lm}(a_i g_i)$ for all $i = 1 \ldots s$ such that $a_i g_i \neq 0$ is called a **standard left representation** of f (with respect to G).

LEMMA 1.9. Let $I \subset A^r$ be a submodule, $G \subset I$ be a left Gröbner basis of I and $\mathrm{NF}(\cdot|G)$ be a left normal form on A^r with respect to G.

(1) For any $f \in A^r$ we have $f \in I \Longleftrightarrow \mathrm{NF}(f \mid G) = 0$.
(2) If $J \subset A^r$ is a submodule with $I \subset J$, then $L(I) = L(J)$ implies $I = J$. In particular, G generates I as a left A–module.
(3) If $\mathrm{NF}(\cdot|G)$ is a reduced normal form, then it is unique.

PROOF. (1) If $\mathrm{NF}(f|G) = 0$ then $f \in I$. If $\mathrm{NF}(f|G) \neq 0$, then $\mathrm{lm}\big(\mathrm{NF}(f \mid G)\big) \notin L(G) = L(I)$, hence $\mathrm{NF}(f \mid G) \notin I$, which implies $f \notin I$.
(2) Let $f \in J$ and assume that $\mathrm{NF}(f|G) \neq 0$. Then $\mathrm{lm}\big(\mathrm{NF}(f|G)\big) \notin L(G) = L(I) = L(J)$, which is a contradiction since $\mathrm{NF}(f|G) \in J$. Hence, $f \in I$ by (1).
(3) Let $f \in A^r$ and assume that h, h' are two reduced normal forms of f with respect to G. Then $h - h' \in {}_A\langle G\rangle = I$. If $h - h' \neq 0$, then $\mathrm{lm}(h - h') \in L(I) = L(G)$, which contradicts the fact that $\mathrm{lm}(h - h')$ is a monomial of either h or h'.

□

REMARK 1.10. In view of the property 2), we can give another characterization of Gröbner bases. Namely, with the assumptions of the Def. 1.6, a finite subset G is called a **left Gröbner basis** of $I \subset A^r$ if and only if

$$\mathbb{N}^n\langle \mathcal{L}(G)\rangle = \mathcal{L}\big(I\big) = \mathcal{L}\big({}_A\langle G\rangle\big).$$

DEFINITION 1.11. Let x^α and x^β be two monomials from A. For $\forall 1 \leq i \leq n$, set $\mu_i = \max(\alpha_i, \beta_i)$ and $\mu := \mu(\alpha, \beta) = (\mu_1, \ldots, \mu_n)$. Then the **pseudo-lcm** of x^α and x^β is defined to be $\mathrm{lcm}(x^\alpha, x^\beta) := x^{\mu(\alpha,\beta)}$.

It enjoys a nice property: $x^\alpha \mid x^{\mu(\alpha,\beta)}$ and $x^\beta \mid x^{\mu(\alpha,\beta)}$.

Why we use the name pseudo–lcm? Using the latter property, let us define a function $f : \mathbb{K} \to A$, $f(c) = x^{\mu(\alpha,\beta)-\alpha} \cdot x^\alpha - c \cdot x^{\mu(\alpha,\beta)-\beta} \cdot x^\beta$ for $c \in \mathbb{K}$.

Let $c_0 := \frac{\mathrm{lc}(x^{\mu(\alpha,\beta)-\alpha}x^\beta)}{\mathrm{lc}(x^{\mu(\alpha,\beta)-\beta}x^\alpha)}$. Then, we see that c_0 is the unique number, such that $\mathrm{lm}(f(c_0)) < x^\mu$ (otherwise, for $c \neq c_0$, $\mathrm{lm}(f(c)) = x^\mu$).

In the commutative case, $c_0 = 1$, $f(c_0) = 0$ and hence $x^{\mu(\alpha,\beta)}$ is regarded as the generalization of the classical lcm function.

However, in the non–commutative case $f(c_0) \neq 0$ in general, but we will make an essential use of the property $\mathrm{lm}(f(c_0)) < x^\mu$.

If A is a G–algebra of Lie type, $c_0 = 1$ but, in general, $f(c_0) \neq 0$.

If A is quasi–commutative, $f(c_0) = 0$ but, in general, $c_0 \neq 1$.

DEFINITION 1.12. Let $f, g \in A^r \setminus \{0\}$ with $\mathrm{lm}(f) = x^\alpha e_i$ and $\mathrm{lm}(g) = x^\beta e_j$, respectively. Set $\gamma := \mu(\alpha, \beta)$ and define the **(left) s–polynomial** of f and g to be

$$\mathrm{LeftSpoly}(f,g) := \begin{cases} x^{\gamma-\alpha} f - \dfrac{\mathrm{lc}(x^{\gamma-\alpha} f)}{\mathrm{lc}(x^{\gamma-\beta} g)} x^{\gamma-\beta} g, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

Of course, $\mathrm{LeftSpoly}(f,g) \in A^r$ is a polynomial if and only if $r = 1$ and $f, g \in A$, and it is a vector of polynomials otherwise. We will also denote LeftSpoly by spoly below.

REMARK 1.13. It is easy to see that $\mathrm{lm}(\mathrm{spoly}(f,g)) < \mathrm{lm}(f \cdot g)$ holds. If $\mathrm{lm}(g) \mid \mathrm{lm}(f)$, say $\mathrm{lm}(g) = x^\beta e_i$, $\mathrm{lm}(f) = x^\alpha e_i$, then the s–polynomial is especially simple,

$$\mathrm{spoly}(f,g) = f - \frac{lc(f)}{lc(x^{\alpha-\beta} g)} x^{\alpha-\beta} g,$$

and $\mathrm{lm}\big(\mathrm{spoly}(f,g)\big) < \mathrm{lm}(f)$. For the normal form algorithm, the s–polynomial is used in this form, while for the Gröbner basis algorithm we need it in the general form as above.

COMPUTATIONAL REMARK 1.14. The intermediate swell of coefficients is a quite known issue in all Gröbner bases computations. One of the ways to avoid extra growth is to follow the so–called "integer strategy", that is do not divide but multiply both sides with a smallest possible value and then extract the content from the resulted difference.

Suppose that f, g have their leading terms in the same component.

If A is a G–algebra of Lie type, then the symmetric formula is simply

$$\mathrm{SymLeftSpoly}_{Lie}(f,g) = \frac{\mathrm{lc}(g)}{\gcd(\mathrm{lc}(f), \mathrm{lc}(g))} x^{\gamma-\alpha} f - \frac{\mathrm{lc}(f)}{\gcd(\mathrm{lc}(f), \mathrm{lc}(g))} x^{\gamma-\beta} g,$$

and it looks exactly the way it does in the commutative case, hence the same strategy as in the commutative case can be applied.

For a general G–algebra the situation is more difficult.

Since $\mathrm{lc}(x^\delta h) = \mathrm{lc}(h)\,\mathrm{lc}(x^\delta \,\mathrm{lm}(h))$,

$$\mathrm{SymLeftSpoly}(f,g) = \mathrm{lc}(g)\,\mathrm{lc}(x^{\gamma-\beta}\,\mathrm{lm}(g)) x^{\gamma-\alpha} f - \mathrm{lc}(f)\,\mathrm{lc}(x^{\gamma-\alpha}\,\mathrm{lm}(f)) x^{\gamma-\beta} g,$$

and we should moreover compute and divide out gcd's of products of coefficients. This can be done in two different ways. Suppose we have two pairs of coefficients, (a,b) and (c,d). Then

$$a' = \frac{a}{\gcd(a,c)},\ c' = \frac{c}{\gcd(a,c)},\quad a'' = \frac{a'}{\gcd(a',d)},\ d' = \frac{d}{\gcd(a',d)},$$
$$b' = \frac{b}{\gcd(b,c')},\ c'' = \frac{c'}{\gcd(b,c')},\quad b'' = \frac{b'}{\gcd(b',d')},\ d'' = \frac{d'}{\gcd(b',d')}.$$

Thus we obtain two new pairs, (a'', b'') and (c'', d''), which have no common divisors. Of course, the required product $a''b''$ equals also $\frac{ab}{\gcd(ab, cd)}$, but the complexity of computing it directly is evidently much bigger than the method we propose. The overall complexity of operations with coefficients in Gröbner basis–related algorithms will vary with respect to the ground field $\mathbb{K}$ and is a good subject for further investigation.

Having an algorithm for computing s–polynomials, we proceed with the normal form algorithm.

For all algorithms below we assume that A is a G–algebra and $<$ is a fixed monomial (module) ordering on A^r, which is a well–ordering on $\mathrm{Mon}(A)$.

Algorithm 1.1 LeftNF

Input : $f \in A^r,\ G \in \mathcal{G}$;

Output: $h \in A^r$, a left normal form of f with respect to G.

$h := f$;
while ($(h \neq 0)$ **and** $(G_h = \{g \in G :\ \mathrm{lm}(g) \mid \mathrm{lm}(h)\} \neq \emptyset)$) **do**
  choose any $g \in G_h$;
  $h := \mathrm{LeftSpoly}(h, g)$;
end while
return h;

Proof. *(of 1.1)* Note, that the proof is essentially the same, that the one we used for free associative algebras in the Algorithm **§1**, 1.1.

Termination:

Again, every specific choice of "any" in the algorithm may give us a different normal form function. Let $h_0 := f$, and in the i–th step of the **while** loop we compute $h_i = \mathrm{spoly}(h_{i-1}, g)$. Since $\mathrm{lm}(h_i) = \mathrm{lm}(\mathrm{spoly}(h_{i-1}, g)) < \mathrm{lm}(h_{i-1})$, we obtain a set $\{\mathrm{lm}(h_i)\}$ of leading monomials of h_i, where $\forall i$ $\mathrm{lm}(h_{i+1}) < \mathrm{lm}(h_i)$. Since $<$ is a well–ordering, this set has a minimum, hence the algorithm terminates.

Correctness:

Suppose this minimum is reached at the step m. Let $h = h_m$, a_i are *terms* (monomials times coefficients) and $g_i \in G$. Making substitutions backwards, we obtain the following expression

$$h = f - \sum_{i=1}^{m-1} a_i g_i,$$

satisfying $\mathrm{lm}(f) = \mathrm{lm}(a_1 g_1) > \mathrm{lm}(a_i g_i) > \mathrm{lm}(h_m)$. By the construction $\mathrm{lm}(h) \notin L(G)$ if $h \neq 0$, hence the correctness follows. □

We can extend LeftNF easily to the reduced normal form algorithm.

Algorithm 1.2 redLeftNF

Input : $f \in A^r,\ G \in \mathcal{G}$;
Output: $h \in A^r$, a reduced left normal form of f with respect to G.

$h := 0,\ g := f$;
while $(g \neq 0)$ **do**
  $g := \mathrm{LeftNF}(g \mid G)$;
  $h := h + \mathrm{lc}(g)\,\mathrm{lm}(g)$;
  $g := g - \mathrm{lc}(g)\,\mathrm{lm}(g)$;
end while
return h;

Proof. *(of 1.2)* Since the "tail" $g - \mathrm{lc}(g)\,\mathrm{lm}(g)$ of g has strictly smaller leading monomial than g and $<$ is a well–ordering, the algorithm terminates. The correctness of the algorithm follows from the correctness of LeftNF. □

Now we present the Left Buchberger's Algorithm.

Algorithm 1.3 LeftGröbnerBasis

Input : $G \in \mathcal{G}$;
Output : $S \in \mathcal{G}$, a left Gröbner basis of the left submodule
$I = {}_A\langle G\rangle \subset A^r$.

$S := G$;
$P := \{(f, g) \mid f, g \in S\} \subset S \times S$;
while $(\ P \neq \emptyset\ )$ **do**
  choose $(f, g) \in P$;
  $P := P \setminus \{(f, g)\}$;
  $h :=$ LeftNF$\big(\ \mathrm{LeftSpoly}(f, g) \mid S\big)$;
  if $(\ h \neq 0\ )$ **then**
    $P := P \cup \{(h, f) \mid f \in S\}$;
    $S := S \cup h$;
  end if
end while
return S;

PROOF. *(of 1.3)*

Termination:

By the property 1.8,ii) we know that if $h \neq 0$ then $\mathrm{lm}(h) \notin L(S)$. Then, ${}_A\langle S\rangle \subset {}_A\langle\{S, h\}\rangle$ and we obtain a strictly increasing sequence of submodules of A. Since A is Noetherian, this sequence stabilizes. It means that, after finitely many steps, we always have $\mathrm{LeftNF}\big(\mathrm{LeftSpoly}(f, g) \mid S)\big) = 0$ for all $(f, g) \in P$ and, after several more steps, the set P of pairs will become empty. Thus LEFTGRÖBNERBASIS terminates.

Correctness:

the proof of correctness follows from the Left Buchberger's Criterion (Theorem 1.16). □

Note, that the algorithms RIGHTNF and RIGHTGRÖBNERBASIS are analogous.

REMARK 1.15. If LeftNF is a reduced normal form and if G is reduced, then S is a reduced Gröbner basis. If G is not reduced, we may apply LeftNF afterwards to $(f, S \setminus \{f\})$ for all $f \in S$ in order to obtain a reduced Gröbner basis.

In the implementation we use generalized criteria for detecting useless reductions within the set of pairs P, see the Section 4.4.

THEOREM 1.16. Let $I \subset A^r$ be a left submodule and $G = \{g_1, \ldots, g_s\}$, $g_i \in I$. Let $\mathrm{LeftNF}(\cdot|G)$ be a left normal form on A^r with respect to G. Then the following are equivalent:

(1) G is a left Gröbner basis of I,
(2) $\mathrm{LeftNF}(f|G) = 0$ for all $f \in I$,
(3) each $f \in I$ has a left standard representation with respect to G,
(4) $\mathrm{LeftNF}\big(\mathrm{LeftSpoly}(g_i, g_j)|G\big) = 0$ for $1 \leq i, j \leq s$.

PROOF. The implication $(1 \Rightarrow 2)$ follows from Lemma 1.9, $(2 \Rightarrow 3)$ follows from the corresponding definitions. As for implication $(3 \Rightarrow 1)$, we see that if f has a left standard representation with respect to G, then $\mathrm{lm}(f)$ must occur as the leading monomial of $a_i g_i$ for some i. It means that $\mathrm{lm}(g_i) | \mathrm{lm}(f)$, hence G is a left Gröbner basis of I. To prove $(3 \Rightarrow 4)$, we note first that $h = \mathrm{LeftNF}\big(\mathrm{LeftSpoly}(f_i, f_j)|G\big) \in I$ and hence, by 3, if $h \neq 0$, we have $\mathrm{lm}(h) \in L(G)$, what contradicts the property (iii) of NF.

The implication $(4 \Rightarrow 1)$ is an important criterion which allows checking and construction of Gröbner bases in a finite number of steps. This implication follows from the more general Theorem 4.8, where the proof is done using syzygies. □

2. Gröbner basics I

Let us enlist the most basic but important applications of Gröbner bases, called "Gröbner basics" by B. Sturmfels. We removed from this list applications, which are "too commutative" and therefore have less impact on the non–commutative case (such as Zariski closure of the image of a map, solving polynomial equations and radical membership).

- Ideal (resp. module) membership problem
- Intersection with subrings (elimination of variables)
- Intersection of ideals (resp. submodules)
- Quotient of ideals
- Saturation of ideals
- Kernel of a module homomorphism
- Kernel of a ring homomorphism
- Algebraic relations between polynomials
- Hilbert polynomial of graded ideals and modules

We will present algorithms to compute every application at different places of this work, since for some applications we'll need preparatory results. We will not consider Hilbert polynomials in this thesis, since this was treated in details in both books [**14**] and [**56**].

Let $\mathbb{K}$ be a field, $A = \mathbb{K}\langle x_1, \dots, x_n \mid \{x_j x_i = c_{ij} x_i x_j + d_{ij}\}_{1 \le i < j \le n}\rangle$ be a G–algebra and $<$ be a monomial ordering on A^r.

2.1. Equality of Submodules.

Suppose RedMinGB to be the algorithm, computing a reduced minimal Gröbner basis for a given module and, moreover, normalizing the output by dividing by the leading coefficient of every generator. If for two submodules $M, N \in A^r$ RedMinGB(M) = RedMinGB(N), then $M = N$. (See also Lemma 1.9,3).

Equivalently, $M = N$ if and only if NF$(M \mid N) = 0 =$ NF$(N \mid M)$. Here M and N should be given in Gröbner bases, although not necessarily minimal or reduced.

Example 2.1. Let us compute an example with Plural.

```
LIB "ncalg.lib";
def A = makeUsl2();
setring A;
ideal I1 = f^2, h^2-1;
ideal I2 = e^2, f^2, h^2-1;
ideal tst;
```

```
// let us compare I1, I2 as two--sided ideals with the NF
ideal T1 = twostd(I1); ideal T2 = twostd(I2);
tst = NF(T1,T2); print(matrix(tst));
==>
  0,0,0,0,0,0        // T2 lies in T1
tst = NF(T2,T1); print(matrix(tst));
==>
  0,0,0,0,0,0       // T1 lies in T2, hence T1=T2

// now let us compare I1, I2 as left ideals with RedMinGB
option(redSB);   // under these options both std and twostd
option(redTail); // return minimal reduced bases
ideal L1 = std(I1); ideal L2 = std(I2);
print(matrix(L1)); // a compact form
==>
  h2-1,f2
print(matrix(L2));
==>
  h2-1,fh-f,eh+e,f2,2ef-h-1,e2 // hence, L1 < L2
```

2.2. Left Module Membership Problem.

Let $M \subseteq A^r$ be a left submodule, $f \in A^r$ and $G = \{f_1, \ldots, f_m\}$ be a left Gröbner basis of M. Then $f \in M$ if and only if $\mathrm{LeftNF}(f|G) = 0$. (See Lemma 1.9,1).

2.3. Intersection with Free Submodules.

Suppose there is a monomial well–ordering $<_A$ on A. Consider the free left module A^r. Recall from the Remark **§2**, 1.4, that the ordering $<_m = (c, <_A)$ on $A^r = \overset{r}{\underset{i=1}{\oplus}} Ae_i$ is the POT ordering (where the components are ordered in a descending way, $e_1 > e_2 > \ldots$). More concrete, it is defined as follows:

$$x^\alpha e_i <_m x^\beta e_j \;\Leftrightarrow\; j < i \text{ or } j = i \text{ and } x^\alpha <_A x^\beta.$$

Lemma 2.2. Let $M \subseteq A^r$ be a submodule. Let $G = \{g_1, \ldots, g_m\}$ be a Gröbner basis of M with respect to $<_m = (c, <_A)$.

Then $\forall\ 1 \leq s \leq r \quad G \cap \overset{r}{\underset{i=s}{\oplus}} Ae_i$ is a Gröbner basis of $M \cap \overset{r}{\underset{i=s}{\oplus}} Ae_i$.

PROOF. Since G is a Gröbner basis of M, for any $h \in M \cap \overset{r}{\underset{i=s}{\oplus}} Ae_i$ there exists $g \in G$ such that $\mathrm{lm}(g) \mid \mathrm{lm}(h)$. Then $\mathrm{lm}(g) \in \overset{r}{\underset{i=s}{\oplus}} Ae_i$, hence, by definition of the ordering $<_m$, we have $g \in \overset{r}{\underset{i=s}{\oplus}} Ae_i$ and, consequently, $g \in G \cap \overset{r}{\underset{i=s}{\oplus}} Ae_i$. Hence the claim. □

We say, that for fixed s, $1 < s \leq r$ we have eliminated the components $e_1, \ldots, e_{s-1}$ from the module M.

Further we refer to this application as to *elimination of the module components* and for such a module ordering as an *elimination ordering for components* $e_1, \ldots, e_{s-1}$.

COMPUTATIONAL REMARK 2.3. We need to formalize the Lemma to become an algorithm. Suppose there is a submodule $M \subset A^n$ and an index set $I = \{i_1, \ldots, i_s\} \subset \{1, \ldots, n\}$.

Then the procedure ELIMCOMPONENT(MODULE M, LIST I) computes the intersection of M with $\oplus_{i \in I} Ae_i$ as follows:

1. choose an elimination ordering $<_I$ for components $e_{i_1}, \ldots, e_{i_s}$;
2. compute a Gröbner basis of M with respect to $<_I$;
3. throw away all the elements from the result, whose leading components are not in $\{e_i \mid i \in I\}$.

There is no built–in command in PLURAL, performing such an algorithm, hence one has to do the steps as above manually. On the other hand, the algorithm ELIMCOMPONENT is used mostly as a sub-algorithm in various kernel procedures, so it is implemented in the kernel of PLURAL.

2.4. Elimination of Variables.

Let $A = \mathbb{K}\langle x_1, \ldots, x_n \mid \{x_j x_i = c_{ij} x_i x_j + d_{ij}\}_{1 \leq i < j \leq n} \rangle$ be a G–algebra. Consider a subalgebra A_r, generated by some subset, say, $\{x_{r+1}, \ldots, x_n\}$ of $\{x_1, \ldots, x_n\}$ for some $r \geq 1$ subject to the relations from A.

We say that such A_r is an *admissible subalgebra*, if d_{ij} are polynomials in $x_{r+1}, \ldots, x_n$ for $r + 1 \leq i < j \leq n$. Hence, A_r is closed with respect to the multiplication, inherited from A and it is a G–algebra.

DEFINITION 2.4. (Elimination ordering) Let A be a G–algebra in n variables, generated by $\{x_1, \ldots, x_n\}$ such that $\{x_{r+1}, \ldots, x_n\}$ generate an admissible sub–G–algebra $B \subset A$.

An ordering $<$ on A is an **elimination ordering for** $x_1, \ldots, x_r$, if for any $f \in A$, $\mathrm{lm}(f) \in B$ implies $f \in B$. If, moreover, $x_1, \ldots, x_r$ generate an admissible sub–G–algebra C, we say in addition, that $<$ is an elimination ordering for C.

We call such an ordering an *admissible elimination ordering* $<_{A_r}$, if the condition $\forall i < j \ \mathrm{lm}(d_{ij}) <_{A_r} x_i x_j$ is satisfied.

If $f \in B$, then of course, for any monomial ordering $<$, $\mathrm{lm}(f) \in \mathrm{Mon}(B)$. Note, that the converse is true if and only if $<$ is an elimination ordering like in the definition, where we have $\mathrm{lm}(f) \in B \Leftrightarrow f \in B$.

REMARK 2.5. Under "elimination of variables $x_1, \ldots, x_r$ from an ideal I" we mean the intersection $I \cap A_r$ with an admissible subalgebra A_r. In contrast to the commutative case, not every subset of variables determines an admissible subalgebra. It can happen that there could be no admissible elimination ordering $<_{A_r}$ (that is, every elimination ordering is non–admissible, see Example 2.11). If A_r or $<_{A_r}$ are not admissible, we cannot "eliminate" $x_1, \ldots, x_r$.

EXAMPLE 2.6. The classical elimination ordering in the commutative case is `lp` (lexicographical ordering). Since for many non–commutative G–algebras it is not an admissible monomial ordering, a usual elimination ordering in the non–commutative setting is a block ordering of the form `(dp`$(1 \ldots r)$`,dp`$(r+1 \ldots n)$`)`, or the ordering with extra weights `(a(`$w_1, \ldots, w_r$`),<)`.

Let A, B be two ordering matrices. Then the block ordering $(<_A, <_B)$ is given by the matrix $\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$.

A matrix for an ordering with extra weights look like follows:

$$\texttt{(a(}\bar{\omega}\texttt{),Dp)} \sim \begin{pmatrix} \omega_1 & \ldots & \omega_r & 0 & \ldots & 0 & 0 & 0 \\ 1 & \ldots & 1 & 1 & \ldots & 1 & 1 & 1 \\ 1 & \ldots & 0 & 0 & \ldots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 & 0 & 0 \end{pmatrix}.$$

However, for a few algebras (e. g. Weyl algebras) the lexicographical ordering is admissible.

LEMMA 2.7. Let $I \subseteq A$ be an ideal, $B = \mathbb{K}\langle x_{r+1}, \ldots, x_n \mid x_j x_i = c_{ij} x_i x_j + d_{ij} \rangle$ be an admissible subalgebra of A, and $<_B$ an admissible elimination ordering for $x_1, \ldots, x_r$ on A . If $S = \{f_1, \ldots, f_m\}$ is a Gröbner basis of I, then $S \cap B$ is a Gröbner basis of $I \cap B$.

PROOF. Take any $x^\alpha \in L(I)$, then there exists such $f \in I$, that $\mathrm{lm}(f) = x^\alpha$. Since $<$ is an elimination ordering for $x_1, \ldots, x_r$, from $\mathrm{lm}(f) \in \mathrm{Mon}(B)$ follows that $f \in B$. Hence, $L(I) \cap B$ equals to

$$\oplus\{\mathbb{K}x^\alpha \mid \exists f \in I, \ \mathrm{lm}(f) = x^\alpha\} \cap B = \oplus\{\mathbb{K}x^\alpha \mid \exists f \in I \cap B, \ \mathrm{lm}(f) = x^\alpha\},$$

and the latter is just $L(I \cap B)$. Then, $L(S) \cap B = L(I) \cap B = L(I \cap B) = L(S \cap B)$, hence $S \cap B$ is a Gröbner basis of $I \cap B$ by the definition. □

COMPUTATIONAL REMARK 2.8. We formalize the previous lemma into the following algorithm. Suppose the subalgebra S, generated by a subset of the set of variables is given together with an ideal I.

Then the procedure ELIMINATE(IDEAL I, SUBALGEBRA S) computes the intersection of I with S as follows:

1. check whether S is an admissible subalgebra;
2. choose the elimination ordering $<_S$ heuristically;
3. check whether $<_S$ is admissible;
4. compute the Gröbner basis of I with respect to $<_S$;
5. throw away all the elements from the result, whose leading monomials involve variables other than those from S.

The built–in command `eliminate` in PLURAL works exactly in the way we have described above, where S is given in form of a monomial. This command does not require the ordering of a current ring to have the eliminating property.

EXAMPLE 2.9. Let us compute a concrete example with PLURAL. There are two possibilities: one can either use the command `eliminate` or set up the ring with the elimination ordering manually, compute Gröbner basis and pick up the needed elements from it. In the example we show how both of them work.

```
ring r=0,(e,f,h,a),(dp(3),dp(1));
// r is equipped with the elimination ordering for e,f,h
matrix d[4][4];
d[1,2]=-h;  d[1,3]=2*e;  d[2,3]=-2*f;
ncalgebra(1,d); // it is U(sl_2) \otimes_K K[a]
poly  p = 4*e*f+h^2-2*h-a; // this is a central element
ideal I = e^2, f^2, h^2-1, p;
ideal J = std(I); // computes a Groebner basis first
print(matrix(J));
==>
  a-3, h2-1, 6fh+fa-9f, 6eh-ea+9e, f2, 4ef+h2-2h-a, e2
// hence, applying the lemma, the result is just a-3
eliminate(I,e*f*h);  // a direct application
==>
  _[1]=a-3
```

With the next examples we are going to illustrate the crucial difference to commutative elimination. Indeed, there are concrete situations, appearing

in applications, where elimination requires extra computations and it may happen that no elimination is possible.

EXAMPLE 2.10. (Complicated elimination)

Let W_1 be the Weyl algebra $\mathbb{K}\langle x, d \mid [d, x] = 1\rangle$. Consider a deformation algebra X of the algebra $X_0 = \mathbb{K}\langle a, b \mid [b, a] = 3a\rangle \otimes_{\mathbb{K}} W_1$, $X = \mathbb{K}\langle a, b, x, d\rangle$ subject to relations $[b, a] = 3a, [d, a] = 3x^2, [x, b] = -x, [d, b] = d, [d, x] = 1$.

We fix a PBW basis $\{a^p b^q x^r d^s\}$. An admissible well–ordering $<$ on X has to satisfy only one condition $x^2 < ad$, what is easily achieved by any degree ordering like `dp, Dp`. Since both $\{a, b\}$ and $\{x, d\}$ generate admissible subalgebras, any block ordering, giving priority to $\{a, b\}$ and having degree orderings in every block, is an admissible elimination ordering for $\{a, b\}$. Small computation ensures, that with respect to all admissible orderings, the non–degeneracy conditions on X vanish.

Any elimination ordering for $\{x, d\}$ has to satisfy $x^2 < da$ whereas $x \gg a$ and $d \gg a$, what is impossible with standard block orderings (like in Example 2.6). We have to introduce extra weights on $\{x, d\}$, although the algebra X does not have any weight in its definition. The ordering condition on weights is satisfied as soon as $2\deg_\omega(x) \leq \deg_\omega(a) + \deg_\omega(d) = \deg_\omega(d)$. For example, for the algebra X with the PBW basis $\{x^p d^q a^r b^s\}$ possible solutions can be the ordering `(a(1,3),Dp)`, what is an elimination ordering with extra weights 1 resp. 3 for x resp. d.

EXAMPLE 2.11. (No elimination is possible)

Let W_1 be the Weyl algebra $\mathbb{K}\langle X, D \mid [D, X] = 1\rangle$.

Consider the algebra $Y = \mathbb{K}\langle e, f, h, X, D\rangle$ subject to the relations
$[f, e] = -h, [h, e] = 2e, [h, f] = -2f, [D, e] = 1, [X, f] = 2XD,$
$[D, f] = -D^2, [X, h] = -2X, [D, h] = 2D, [D, X] = 1.$
It arises as a deformation of the $Y_0 = U(\mathfrak{sl}_2) \otimes_{\mathbb{K}} W_1$ (the first three relations are defining for $U(\mathfrak{sl}_2)$).

An admissible ordering on Y with the PBW basis $\{e^p f^q h^r X^s D^t\}$ requires that $ef > h$ (hence a degree ordering is needed) and $fX > XD$ and $fD > D^2$, which is true if and only if $f > D$. Of course, in the given PBW basis this holds for any degree ordering, in particular, for any elimination ordering for $\{e, f, h\}$.

But, as we see, the eliminating conditions for $\{X, D\}$, namely $\{X, D\} \gg \{e, f, h\}$ in the PBW basis $\{X^p D^q e^r f^s h^t\}$ are not compatible with the requirement $f > D$. Hence, for any ideal $I_Y \subset Y$ the intersection $I_Y \cap U(\mathfrak{sl}_2)$ cannot be computed.

However, $S = \mathbb{K}\langle e, h | [h, e] = 2e\rangle \subset U(\mathfrak{sl}_2)$ is an admissible subalgebra. Hence, we are able to eliminate $\{f, X, D\}$, taking either $\{f^p X^q D^r e^s h^t\}$ or $\{X^p f^q D^r e^s h^t\}$ as the PBW basis.

EXAMPLE 2.12. (Easy example with no elimination)

Let $A = \mathbb{K}\langle p, q \mid qp = pq + d(p, q)\rangle$ be a G–algebra, that is $\mathrm{lm}(d) < pq$. Then, if for some $m \geq 2$, $\mathrm{lm}(d) = q^m$, the intersection of any ideal $I \subset A$ with the subalgebra $\mathbb{K}[p]$ cannot be computed because of the following objection. The elimination ordering for such computation requires $q \gg p$, what implies $q^2 > pq$ and hence $\mathrm{lm}(d) > pq$, what contradicts the ordering condition for A as a G–algebra.

2.5. Intersection of Ideals.

LEMMA 2.13. Let $I = {}_A\langle f_1, \ldots, f_r\rangle$, $J = {}_A\langle g_1, \ldots, g_s\rangle$ be two left ideals. Consider the ideal $D := t \cdot I + (1 - t) \cdot J \subseteq A_t$, where $A_t = A \otimes_{\mathbb{K}} \mathbb{K}[t]$ is viewed as algebra, generated by $\{x_1, \ldots, x_n, t\}$, t commutes with A. Let $<$ be an elimination ordering for t on A_t. Then $I \cap J = D \cap A$.

PROOF. Assume that $\{f_i\}$ and $\{g_i\}$ are already Gröbner bases of I and J. Let $f \in D \cap A$. Then we can present it as a sum

$$f = \sum_{i=1}^{r} a_i t f_i + \sum_{i=1}^{s} b_i (1 - t) g_i.$$

Specializing t at some value, we obtain the following : $t = 0 \Rightarrow f = \sum_{i=1}^{s} b_i g_i \in J$, $\quad t = 1 \Rightarrow f = \sum_{i=1}^{r} a_i f_i \in I$. Hence, $f \in I \cap J$ and $I \cap J \supseteq D \cap A$.

Conversely, let $f \in I \cap J$. Then f could be represented in two ways:

$$f = \sum_{i=1}^{r} a_i f_i = \sum_{i=1}^{s} b_i g_i.$$

Since t commutes with A, consider

$$f = tf + (1 - t)f = \sum_{i=1}^{r} a_i t f_i + \sum_{i=1}^{s} b_i (1 - t) g_i.$$

Hence, $f \in D \cap A$ and $I \cap J = D \cap A$. □

3. Two–sided Gröbner Bases and GR–algebras

3.1. An Approach for Two–sided Ideals.

For any polynomial f from a left (resp. right) module M in a G–algebra there exists a standard left (resp. right) representation of f. This representation is clearly a generalization of the one in the commutative theory. But for such important objects as two–sided ideals (they appear, for instance, as annihilators of modules) we have no direct analogue.

EXAMPLE 3.1. Consider $U(\mathfrak{sl}_2)$ (**§5**, 1.1) and the two–sided ideal I, generated by the variable f. Consider the polynomial $f + efh \in I$. There are no $a, b \in U(\mathfrak{sl}_2)$, such that $f+efh = afb$. Hence, we have a presentation

$$g = \sum_{i\in\Lambda} l_i g_i r_i,\ g \in {}_A\langle g_1, \ldots, g_n\rangle_A,\ l_i, r_i \in A,$$

for some index set Λ with $\#\Lambda > n$.

This shows that two–sided generators of an ideal or bimodule are rarely revealing essential structural information. The idea is to consider the two–sided structure not as a set of two–sided generators (like we do with the left generators for left modules), but as one–sided (left or right) structure, equal to the given two–sided one (see also [**41**] or [**46**]). Then, in the algorithm which follows we start from the left structure and *complete* (the term itself goes back to J. Apel, [**4**]) it successively to the right structure, keeping the left one.

Of course, the result of such a completion will be both a left and a right Gröbner basis, hence the same could be done for right–sided input by means of completion to the left. If for a minimal set of generators $F = \{f_1, \ldots, f_n\}$ of a two–sided ideal I we have ${}_A\langle F\rangle = {}_A\langle F\rangle_A = I$, the classical result ([**41**]) assures two more equalities $\langle F\rangle_A = {}_A\langle F\rangle_A = I$, and ${}_A\langle F\rangle = \langle F\rangle_A = I$ follows.

We say that F is a **two–sided Gröbner basis** of a two–sided ideal I, if it satisfies one of three conditions above. We compute such bases with the following algorithm.

REMARK 3.2. Note, that if F is a left Gröbner basis of ${}_A\langle F\rangle$ and a right Gröbner basis of $\langle F\rangle_A$, in general both ideals are not two–sided and ${}_A\langle F\rangle \neq \langle F\rangle_A$. Consider the following example: let $A = U(\mathfrak{sl}_2)$ (see **§5**, 1.1) over the field $\mathbb{K}(\alpha)$ and the set of generators $F = \{e, h - \alpha\}$. Then F is both a right and a left Gröbner basis, but not a two–sided one, because the ideals ${}_A\langle F\rangle$ and $\langle F\rangle_A$ are strictly left respectively right ideals.

Let $<$ be a well-ordering on a G–algebra A, which is generated by $x_1, \ldots, x_n$. Assume there is an algorithm UpdateGröbnerBasis(G, S) which computes a (left) Gröbner basis of $G \cup S$ for sets G (being a Gröbner basis) and S (being an arbitrary finite set). Technically it means that during the Buchberger's algorithm, no critical pairs within G are considered.

Let NF be a fixed left normal form on A.

Algorithm 3.1 TwoSidedGröbnerBasis

Input : T, a set of two–sided generators from A;
Output : L, a left Gröbner basis of the ideal $I = {}_A\langle T \rangle_A \subseteq A$.

```
L :=LeftGröbnerBasis(T, NF);
W := ∅;  G := L;
loop
    for i = 1 to size(G) do
        for j = 1 to n do
            g := G[i] · x_j;
            h := LeftNF(g | L);
            if (h ≠ 0) then
                W = W ∪ h;
                if IsConstant(h) then
                    return (1);
                end if
            end if
        end for
    end for
    if W = ∅ then
        break;                          ▷ i.e. quit the loop
    else
        L := UpdateGröbnerBasis(L, W);
    end if
    G := W;
    W := ∅;
end loop
return L;
```

Proof. (of 3.1)

Termination: Since A is Noetherian, any ideal admits a finite generating set, hence we quit the **loop** ... **end loop** cycle after finitely many

steps. However, it is not clear how many iterations we should do before obtaining a result.

Correctness: We have started from the two–sided generating set, obtained its left Gröbner basis as of a left generating set $G = L$ (we can require its minimality furthermore). In the **loop** ... **end loop** cycle we are adding into the set W new generators, multiplying elements from G from the right side by the algebra generators $x_1, \ldots, x_n$, and then reducing them with respect to the L.

After the first execution of the **loop** ... **end loop** cycle, we have computed the set W of all the nonzero normal forms $\{r_{ij}\}$ of products $\{f_i x_j \mid f_i \in G\}$. Now for some element $f_i \in G$ we see that $(f_i \cdot x_j)x_k$ equals to

$$(\sum_p a_p f_p + r_{ij})x_k = \sum_p \left(\sum_q a_p b_q f_q + a_p r_{pk} \right) + r_{ij}x_k.$$

So, the only element we have to reduce further is $r_{ij}x_k$, hence it suffices to continue iterations with the set W only. We update L performing the UpdateGröbnerBasis(L, W). We quit the cycle either if $L = \{1\}$ or if $W = \emptyset$, that is when all the recently computed "candidates" (h in the algorithm) are reduced to zero with respect to L. This means, that L is the complete basis. □

Remark 3.3. Comparing our algorithm with the algorithm Gröbner, proposed in [**41**], we should point out that our algorithm is optimized: we consider less critical pairs by using the UpdateGröbnerBasis algorithm. An algorithm TSGB of Kredel ([**46**]), being a modified Gröbner basis algorithm itself uses a different strategy, which still needs to be compared to what we use. In addition, we use the early detection of the appearance of constants among the elements which indicates that the result coincides with the whole algebra.

The algorithm could also be used for checking, whether a given set of generators is a two–sided Gröbner basis. Provided the given ideal L is already given in a left Gröbner basis, we have to perform $n \cdot m$ reductions (calls of LeftNF) for n, m being the numbers of generators of A and L respectively.

Example 3.4. Let us compute a small but interesting example with Plural. Consider the algebra $U(\mathfrak{sl}_2)$ over $\mathbb{Q}$ and the set
$K = \{e^3,\ f^3,\ h^3 - 4h\} \subset U(\mathfrak{sl}_2)$. Let NF be the fixed left normal form, $I =$ LeftGröbnerBasis(K, NF) and $J =$ TwoSidedGröbnerBasis(K). Moreover, the results are completely reduced Gröbner bases.

```
LIB "ncalg.lib";
def A = makeUsl2();
setring A;
ideal K = e^3, f^3, h^3-4*h;
option(redSB);          // these options ensure that the result
option(redTail);        // is completely reduced
ideal I = std(K);       // left GB
ideal J = twostd(K);    // two--sided GB
```

We obtain
$I = \{e^3,\ f^3,\ h^3 - 4h,\ eh^2 + 2eh,\ fh^2 - 2fh,\ 2efh - h^2 - 2h\}$,
$J = I \cup \{e^2f - eh - 2e,\ ef^2 - fh,\ e^2h + 2e^2,\ f^2h - 2f^2\}$.
One can see the exact difference between the ideals. See the article [**53**] and **§4**, 3 for more examples and timings.

EXAMPLE 3.5. We have reported in [**48**], [**53**] on the following curious example: consider the universal enveloping algebra $U(\mathfrak{g}_2)$ (**§5**, 1.4) in 14 variables over $\mathbb{Q}$ and a two–sided ideal, generated by the third power of the variable x_1, being an image of the shortest positive root from the root system of the Lie algebra $\mathfrak{g}_2$ under the canonical inclusion. Then its minimal two–sided Gröbner basis consists of 106 elements and does not even involve the initial generator. Let us compute the left standard presentation of x_1^3 with respect to the basis B. This can be done with the commando `lift` (see 4.5).

```
LIB "ncalg.lib";
def G2 = makeUg2();
setring G2;
option(redSB);
option(redTail);
ideal I  = x(1)^3;
ideal B  = twostd(I);
size(B);
==>
  106
module T = lift(B,I); // transformation matrix between I and B
T[1];
==>
x(1)*gen(77)-2x(5)*gen(43)-2x(5)*gen(1)+2y(2)*gen(6)+2*gen(47)
```

We see, that the standard representation of the initial generator involves five polynomials. Namely, $x_1^3 = x_1B_{77} - 2x_5B_{43} - 2x_5B_1 + 2y_2B_6 + 2B_{47}$, where $B_1 = y_2h_\beta - y_2$, $B_6 = x_5h_\alpha + x_5h_\beta + x_5$, $B_{43} = x_1y_3 + y_2h_\alpha + 3y_2$,

$B_{47} = x_5y_2$, $B_{77} = x_1^2 + 2x_5y_3$.

This example is a good demonstration of the fact how different left and two–sided ideals can be, built from the same generating set. In particular, $U(\mathfrak{sl}_2)/I$ has Gel'fand–Kirillov dimension 13 and $U(\mathfrak{sl}_2)/B$ dimension 0 (indeed, it is even of dimension 50 over $\mathbb{K}$).

Perspectives

Very recently, there appeared two articles, which give new ideas for computations with two–sided (bimodule) structures.

In [**44**], Kobayashi presents a very general approach to subbimodules over associative algebras, gives algorithm for computing the free bimodule resolution (using ideas of Anick ([**1**]) and generalizing them further for the bimodule situation) and describes as an important application as the computation of Hochschild cohomology groups for associative algebras. However, there is no implementation and all the computations in the article are made by hand.

In the article [**28**], Manuel and Maria García Román propose a new method for computing Gröbner bases for subbimodules of G–algebras which could be better than the one proposed by us, although there is no sufficiently good implementation yet (only a preliminary "check of ideas", done in MAPLE) and no real comparisons with the methods we propose are available. This topic should be carefully investigated for its complexity and efficiency; the authors have already decided to implement it in our system SINGULAR:PLURAL as soon as possible.

3.2. GR–algebras and Gröbner bases in factor algebras.

Having implemented two–sided Gröbner bases, we are able to perform Gröbner basis computations in factor algebras (GR–algebras), what is quite important prerequisite for many applications. Moreover, we are able to do it within the same framework of Gröbner bases we developed for G–algebras.

Notation: We are using standard capital letters for G–algebras and their ideals (say, $I \subset B$) and calligraphic capital letters for GR–algebras and their ideals (say, $\mathcal{J} \subset \mathcal{A}$).

A **Gröbner–ready** algebra was already defined in Def. **§1**, 3.7.

It is clear, that any GR–algebra is Noetherian algebra of PBW type, that is its $\mathbb{K}$–basis is a subset of PBW basis, spanned on the same set of

algebra generators. Working with GR–algebras of the type $\mathcal{A} = B/I$, we will make some assumptions. First of all, we assume that n and I are chosen in such a way that n is a minimal number for which there exists B, a G–algebra in n variables, $I \subset B$, and we have $\mathcal{A} \cong B/I$. Secondly, we assume that the nontrivial ideal I is given by its two–sided Gröbner basis (which we can compute with the help of 3.1).

Since by Lemma 1.9,3 the reduced normal form is unique, we can identify any class $[f] \in \mathcal{A}$ with its canonical representative $\bar{f} := \mathrm{redNF}_B(f \mid I) \in B$.

Definition 3.6. Let B be a G–algebra in n variables and $\mathcal{A} = B/I$ be a GR–algebra.

1) For any $\alpha \in \mathbb{N}^n$, x^α is called a **monomial** in $\mathcal{A}$, if $\overline{x^\alpha} \neq 0$ and $x^\alpha = \overline{x^\alpha}$. The set of monomials in $\mathcal{A}$ is identified with the subset of the PBW basis of B, namely $\mathrm{Mon}(\mathcal{A}) = \{\mathrm{lm}_B(\overline{x^\alpha}) \mid \alpha \in \mathbb{N}^n\} \setminus \{0\}$.

2) An ordering $<$ is called a **monomial ordering** on $\mathcal{A}$, if the following conditions hold:

- $\forall\, \alpha, \beta \in \mathbb{N}^n$ such that x^α and x^β are monomials, $\alpha < \beta \Rightarrow x^\alpha <_{\mathcal{A}} x^\beta$,
- $\forall\, \alpha, \beta, \gamma \in \mathbb{N}^n$, such that x^α and x^β are monomials and the inequalities $x^\alpha <_{\mathcal{A}} x^\beta$, $\overline{x^{\alpha+\gamma}} \neq 0$, $\overline{x^{\beta+\gamma}} \neq 0$ hold, then also holds $\mathrm{lm}(\overline{x^{\alpha+\gamma}}) <_{\mathcal{A}} \mathrm{lm}(\overline{x^{\beta+\gamma}})$.

3) Any $\bar{f} \neq \bar{0}$ can be written uniquely as the sum $\bar{f} = cx^\alpha + f'$, where $c \in \mathbb{K}^*$ and $x^{\alpha'} <_{\mathcal{A}} x^\alpha$ for any nonzero term $c'x^{\alpha'}$ from f'. We define $\mathrm{lm}([f]) := \mathrm{lm}(\bar{f}) = x^\alpha$, the **leading monomial** of $[f]$, respectively $\mathrm{lc}([f]) := c$, the **leading coefficient** of $[f]$.

Lemma 3.7. Let $\mathcal{A} = B/I$ and NF_B be a normal form on B. Suppose we have $[f] \in \mathcal{A}$ and a left ideal $\mathcal{J} \subset \mathcal{A}$. Define $\tilde{J} :=$LeftGröbnerBasis$(I + \mathcal{J}, \mathrm{NF})$. Then $\mathrm{NF}_{\mathcal{A}}([f] \mid J) := \mathrm{NF}_B(f \mid \tilde{J})$ is a **normal form of** $[f]$ with respect to $\mathcal{J}$ in $\mathcal{A}$, according to Definition 1.8.

Notation: For an ideal I and a two–sided ideal T, we also use the notation "$I \mod T$" for the $\mathrm{NF}(I + T_A \mid T_A)$.

Lemma 3.8. Let $\mathcal{A} = B/I$ be a GR–algebra and $\mathcal{J} \subset \mathcal{A}$ be a left ideal. Let $F = \{f_1, \ldots, f_k\}$ be a Gröbner basis of an ideal $I + \mathcal{J} \subseteq B$. Then the set $\{\mathrm{redNF}(f_i \mid I) \mid 1 \leq i \leq k\} \setminus \{0\}$ is a Gröbner basis of $\mathcal{J} \subset \mathcal{A}$.

Examples of important GR–algebras were already given in **§1**, 3.11.

LEMMA 3.9. There is a category $\mathcal{GR}$, with GR–algebras as objects and $\mathbb{K}$-algebra homomorphisms as morphisms. The category $\mathcal{GR}$ is closed under the following operations:

(i) tensor product over the ground field $\mathbb{K}$,

(ii) factor by two–sided ideal,

(iii) taking the opposite algebra.

PROOF. (i). Let A (resp. B) be a G–algebra in n (resp. m) variables with an ordering $<_A$ (resp. $<_B$): $A = \mathbb{K}\langle x_1, \ldots, x_n \mid \ f_{ji} = 0,\ 1 \leq i < j \leq n\rangle$, $B = \mathbb{K}\langle y_1, \ldots, y_m \mid \ g_{ji} = 0,\ 1 \leq i < j \leq m\rangle$. Let, moreover, $T_A \subset A$ and $T_B \subset B$ be proper two–sided ideals, such that $\mathcal{A} = A/T_A$ and $\mathcal{B} = A/T_B$ are GR–algebras. We claim, that $\mathcal{A} \otimes_{\mathbb{K}} \mathcal{B}$ is a GR–algebra.

At first, $C = A \otimes_{\mathbb{K}} B$ is generated by $\{x_i \otimes 1 \mid 1 \leq i \leq n\}$ and $\{1 \otimes y_j \mid 1 \leq j \leq m\}$ which we identify with $\{x_i\}$ and $\{y_j\}$ respectively. A natural block ordering $(<_A, <_B)$ (although it is not the only admissible ordering) makes C into a G–algebra in $m + n$ variables, since $\forall\ i, j\ y_j x_i = x_i y_j$ and consequently all non–degeneracy conditions in C vanish because they do on A and B (by the same argument as in the proof of **§1**, 7.3).

If T_A and T_B were given as two–sided Gröbner bases, their images in C under canonical inclusions keep this property. Hence, the ideal $T_C = T_A + T_B$ is a two–sided ideal, given in its two–sided Gröbner basis.

Then, $\mathcal{A} \otimes_{\mathbb{K}} \mathcal{B} \cong C/T_C = A \otimes_{\mathbb{K}} B/\{T_A + T_B\}$.

(ii). Let $\mathcal{J} \subset A/I$ be a proper two–sided ideal in a GR–algebra. Then, $K := I + \mathcal{J} \subset A$ is a proper two–sided ideal too. Hence, $(A/I)/\mathcal{J} \cong A/K$ is a GR–algebra.

(iii). It has been proved already in **§1**, 5. □

In particular, for every GR–algebra $\mathcal{A}$, its **enveloping** algebra

$\mathcal{A}^e := \mathcal{A} \otimes_{\mathbb{K}} \mathcal{A}^{\mathrm{opp}}$ is a GR–algebra.

Note, that any $(\mathcal{A}, \mathcal{A})$–bimodule M can be presented in a canonical way as a left $\mathcal{A}^e$–module.

4. Syzygies and Free Resolutions

Syzygies and resolutions are very important objects both on their own and as necessary components for many constructions, for example, in homological algebra. In this section we discuss two methods for computing left syzygy modules and free resolutions of modules using Gröbner bases. The first method goes back to a special Gröbner basis computation (4.3) in the free module.

The second one is the generalized Schreyer's method (4.8), which leads us to an elegant proof of Left Buchberger's criterion for a left Gröbner basis as well as to important result on a Gröbner basis of a syzygy module. Moreover, with the help of this method, we established an upper bound for global homological dimension of G–algebras (Theorem 4.16).

4.1. Homomorphisms of Free Modules.

Let $\mathbb{K}$ be a field, A a G–algebra and $<$ a monomial module ordering on A^n for any n.

The free A–module A^n can be viewed both as left and a right A–module. For a vector $v \in A^n$ respectively a matrix M, we denote v^t resp. M^t transposed vector resp. matrix.

Consider two free left A–modules A^m, A^n with canonical bases $\{\varepsilon_i\}$ and $\{e_j\}$ respectively. Any left homomorphism ϕ is given by its values on generators, that is

$$\phi: \quad A^m = \bigoplus_{i=1}^{m} A\varepsilon_i \;\longrightarrow\; A^n = \bigoplus_{j=1}^{n} Ae_j, \qquad \varepsilon_i \longmapsto \bar{f}_i,$$

where $\bar{f}_i = \sum_{j=1}^{n} \Phi_{ji} e_j \in A^m$ are vectors with $\Phi_{ji} \in A$. A left A–linearity implies

$$\phi\big(\sum_{i=1}^{m} a_i\varepsilon_i\big) = \sum_{i=1}^{m} a_i\bar{f}_i = \sum_{j=1}^{n}\big(\sum_{i=1}^{m} a_i\Phi_{ji}\big)e_j \;\; \forall a_i \in A.$$

Hence, we can present ϕ by the matrix $\Phi = \big(\Phi_{kl}\big) = (\bar{f}_1, \ldots, \bar{f}_m) \in A^{n\times m}$. But, in contrast to the commutative intuition, instead of the action $\phi(\bar{x}) = \Phi \cdot \bar{x}$, which is indeed right but not left A–linear, we must take the only correct action

$$\phi(\bar{x}) := (\bar{x}^t \cdot \Phi^t)^t.$$

Then, the image of ϕ is a submodule of A^n, generated by the columns of a matrix Φ. In the sequel, a submodule of a free module and a homomorphism will be both presented by a matrix, which columns constitute the generating set of a module.

Let $\psi : A^m \longrightarrow A^n$ be a right homomorphism of free right modules, given by the same matrix Φ. Then $\psi(\bar{x}) := \Phi \cdot \bar{x}$ is well–defined right homomorphism.

Note that if Φ consists of elements, central in A (in particular, this is true for any Φ over a commutative algebra A), then $(\Phi\bar{x})^t = (\bar{x}^t \cdot \Phi^t)$ and both left and right homomorphisms, associated to the matrix Φ are acting identically.

From the facts above it is easy to conclude the following

LEMMA 4.1. The right A–module of left homomorphisms between two left free A–modules $\mathrm{Hom}_A(A^m, A^n)$ is isomorphic to the right free A–module A^{nm}.

4.2. Syzygies and Their Computation via Gröbner Basis.

DEFINITION 4.2. A **left** (resp. **right**) **syzygy** between k elements $\{\bar{f}_1, \ldots, \bar{f}_k\} \subset A^r$ is a k–tuple $(a_1, \ldots, a_k) \in A^k$ satisfying

$$\sum_{i=1}^{k} a_i \bar{f}_i = 0 \quad (\text{resp. } \sum_{i=1}^{k} \bar{f}_i a_i = 0).$$

The set of all left (resp. right) syzygies between $\{\bar{f}_1, \ldots, \bar{f}_k\}$ is a left (resp. right) submodule $S := \mathrm{Syz}(\{\bar{f}_1, \ldots, \bar{f}_k\})$ of A^k.

Since the definition of the right syzygy is analogous to the left one, we will work only with left syzygies and mention by syzygy (resp. normal form NF) below the left syzygy (resp. LeftNF).

We can view $S = \mathrm{Syz}(\{\bar{f}_1, \ldots, \bar{f}_k\})$ as the kernel of the following homomorphism of left A–modules

$$\varphi_{\mathbf{1}} : \quad F_1 = A^k = \bigoplus_{i=1}^{k} A\varepsilon_i \longrightarrow F_0 = A^r = \bigoplus_{j=1}^{r} Ae_j, \qquad \varepsilon_i \longmapsto \bar{f}_i.$$

Let $I = {}_A\langle \bar{f}_1, \ldots, \bar{f}_k \rangle$ be the left submodule of F_0, then φ_1 surjects onto I and $\mathrm{Syz}(I) := \mathrm{Ker}\, \varphi_1$ is called the **(first) module of syzygies** of I with respect to the set of generators $\{\bar{f}_1, \ldots, \bar{f}_k\}$. Easy computations ensure that the isomorphism class of $\mathrm{Syz}(I)$ as of A–module does only depend on the isomorphism class of I, in particular, it is independent of the set of generators.

In particular, let S be a matrix, corresponding to the left submodule $\mathrm{Syz}(I)$. Then, using the results before, we have $S^t \cdot I^t = 0$.

LEMMA 4.3. Let $I = {}_A\langle \bar{f}_1, \ldots, \bar{f}_k \rangle \subset A^r$. Define a free module $M = \bigoplus_{i=r+1}^{r+k} Ae_i$ and consider canonical embedding i and canonical projection π

$$i : A^r \to A^{r+k} = A^r \oplus M, \quad \pi : A^{r+k} \to A^k.$$

Let $G = \{\bar{g}_1, \ldots, \bar{g}_s\}$ be a left Gröbner basis of $F = {}_A\langle \bar{f}_1 + e_{r+1}, \ldots, \bar{f}_k + e_{r+k} \rangle$ with respect to an ordering, being an elimination ordering for components $e_1, \ldots, e_r$ (see 2.3). Suppose that $G \cap M = \{\bar{g}_1, \ldots, \bar{g}_\ell\}$, then

$$\mathrm{Syz}(I) = {}_A\langle \pi(\bar{g}_1), \ldots, \pi(\bar{g}_\ell) \rangle .$$

PROOF. As an elimination ordering we can take, for example, the ordering $(c,<)$: $x^\alpha e_i < x^\beta e_j$ if $j<i$ or $j=i$ and $x^\alpha < x^\beta$. $G \cap M$ is a left Gröbner basis of $F \cap M$ by Lemma 2.2. On the other hand, $\pi\big(F \cap M\big) = \mathrm{Syz}(I)$. Namely, let $\bar{h} \in F \cap M$, that is, $\bar{h} = \sum_{\nu=r+1}^{r+k} h_\nu e_\nu = \sum_{j=1}^{k} b_j(\bar{f}_j + e_{r+j})$ for suitable $b_j \in A$. This implies that $\sum_{j=1}^{k} b_j \bar{f}_j = 0$ and $b_j = h_{r+j}$.

Conversely, let $\bar{h} = (h_1, \ldots, h_k) \in \mathrm{Syz}(I)$, then if $\sum_{\nu=1}^{k} h_\nu \bar{f}_\nu = 0$, we have $\sum_{\nu=1}^{k} h_\nu(\bar{f}_\nu + e_{r+\nu}) \in F \cap M$. □

REMARK 4.4. In order to illustrate the method, we draw two matrices.

Suppose we are given $I = \{\bar{f}_1, \ldots, \bar{f}_k\} \subset A^r$ and let F be a matrix with $\bar{f}_i$ as columns. We append from below a $r \times r$ unit matrix to F, written column-wise. We put the result of the Gröbner basis computation of this matrix with respect of a fixed ordering in the second matrix, sorting the columns in such a way, that the elements, having first r components zero, are moved to the left. (We denote $\bar{0} = (0, \ldots, 0) \in A^r$).

$$\begin{pmatrix} \bar{f}_1 & \cdots & \bar{f}_k \\ 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix} \xrightarrow{GB} \left(\begin{array}{ccc|ccc} \bar{0} & \ldots & \bar{0} & \bar{h}_1 & \ldots & \bar{h}_t \\ \hline & & & & & \\ & \mathbf{S} & & & \mathbf{T} & \\ & & & & & \end{array}\right).$$

Let H be a matrix with columns $\bar{h}_i$. Then

- $\{\bar{h}_1, \ldots, \bar{h}_t\}$ is a Gröbner basis of I,
- columns of $\mathbf{S}$ generate $\mathrm{Syz}(\{\bar{f}_1, \ldots, \bar{f}_k\})$,
- $\mathbf{T}$ is a left transition matrix between two bases of F, i.e. $H^t = \mathbf{T}^t F^t$.

COMPUTATIONAL REMARK 4.5. Note, that the algorithm for computing the left transition matrix between two bases follows from the previous remark. It is implemented as the command `lift`, whereas the command `liftstd` returns both a left Gröbner basis and a left transition matrix, and the command `syz` returns a generating set (not necessary a Gröbner basis) of the first module of syzygies.

EXAMPLE 4.6. (`liftstd` command)

```
LIB "ncalg.lib";
def A = makeUsl2();    setring A;
ideal i = e2,f;
option(redSB);
option(redTail);
matrix T;   // transformation matrix will be written into T
ideal j = liftstd(i,T);
print(matrix(j));        // ideal in a compact form
```

```
==>  f, 2h2+2h, 2eh+2e, e2
  print(T);
==> 0,f2,          -f,1,
==> 1,-e2f+4eh+8e,e2,0
  ideal tj = ideal(transpose(T)*transpose(matrix(i)));
  std(ideal(j-tj)); // test whether tj coincides with j
==> _[1]=0
```

4.3. Schreyer's Method.

Let us define a monomial ordering on F_1 which is tightly connected with Gröbner bases of modules. This was first introduced and used by F.–O. Schreyer (1980, 1986). Let $>_0$ be some monomial ordering on F_0, then a new ordering $>_1$ on F_1 is defined as follows:

$$\begin{aligned} x^\alpha \varepsilon_i >_1 x^\beta \varepsilon_j \quad \Leftrightarrow \quad & \mathrm{lm}(x^\alpha f_i) >_0 \mathrm{lm}(x^\beta f_j) \ \text{ or} \\ & \mathrm{lm}(x^\alpha f_i) = \mathrm{lm}(x^\beta f_j) \ \text{ and } \ i < j. \end{aligned}$$

Note, that both orderings induce the same ordering on A. We call the ordering $>_1$ the **Schreyer ordering**. It is clear that it depends on $f_1, \ldots, f_k$.

Suppose we have some (left) normal form NF on A^r. For each $i < j$ such that f_i and f_j have the leading term in the same component, say $\mathrm{lm}(f_i) = x^{\alpha_i} e_\nu$, $\mathrm{lm}(f_j) = x^{\alpha_j} e_\nu$, define the monomial

$$m_{ji} := x^{\mu(\alpha_i, \alpha_j) - \alpha_i} \in A,$$

Setting $c_i = \mathrm{lc}(m_{ji} f_i)$ and $c_j = \mathrm{lc}(m_{ij} f_j)$, we have

$$m_{ji} f_i - \frac{c_i}{c_j} m_{ij} f_j = \mathrm{spoly}(f_i, f_j).$$

Assume now that $i < j$ and $\mathrm{NF}\big(\mathrm{spoly}(f_i, f_j) \mid G\big) = 0$. Then we have a standard (left) representation

$$m_{ji} f_i - \frac{c_i}{c_j} m_{ij} f_j = \sum_{\nu=1}^{k} a_\nu^{(ij)} f_\nu, \ \ a_\nu^{(ij)} \in A.$$

Now, for every $i < j$ such that $\mathrm{lm}(f_i)$ and $\mathrm{lm}(f_j)$ involve the same component, define the elements from A^k

$$s_{ij} := m_{ji} \varepsilon_i - \frac{c_i}{c_j} m_{ij} \varepsilon_j - \sum_{\nu=1}^{k} a_\nu^{(ij)} \varepsilon_\nu.$$

Then $s_{ij} \in \mathrm{Syz}(I)$ and the following result holds :

LEMMA 4.7. $\mathrm{lm}(s_{ij}) = m_{ji}\varepsilon_i$.

PROOF. Since $\mathrm{lm}(m_{ij}f_j) = \mathrm{lm}(m_{ji}f_i)$ and $i < j$, we have $\mathrm{lm}(m_{ji}\varepsilon_i) >_1 \mathrm{lm}(m_{ij}\varepsilon_j)$. By the standard representation's property we obtain

$$\mathrm{lm}(a_\nu^{(ij)} f_\nu) \leq \mathrm{lm}(m_{ji}f_i - \frac{c_i}{c_j} m_{ij}f_j) < \mathrm{lm}(m_{ji}f_i).$$

□

THEOREM 4.8. Let $G = \{g_1, \ldots, g_s\}$ be a set of generators of a left submodule $I = {}_A\langle G\rangle \subset A^r$ and there is an index set

$$M = \{(i,j) \mid 1 \leq i < j \leq s,\ \{\mathrm{lm}(g_i), \mathrm{lm}(g_j)\} \text{ involve the same component}\}.$$

Suppose, that for some (left) normal form NF on A^r,

$$\mathrm{NF}\big(\mathrm{spoly}(g_i, g_j) \mid G\big) = 0,\ \forall (i,j) \in M.$$

Then the following holds:

1) G is a (left) Gröbner basis of I.

2) $S := \{s_{ij} \mid (i,j) \in M\}$ is a (left) Gröbner basis of $\mathrm{Syz}(I)$ with respect to the Schreyer ordering.

PROOF. We give a proof of 1) and 2) at the same time.

Since G generates I, consider some $f \in I$ and its preimage $g \in F_1$,

$$g = \sum_{i=1}^{s} a_i\varepsilon_i,\ f = \varphi(g) = \sum_{i=1}^{s} a_i g_i.$$

In case 1), we assume $f \neq 0$, in case 2) $f = 0$. Let $h = \sum h_j\varepsilon_j \in F_1$ be a normal form of g with respect to S for some normal form on F_1. Consider a standard representation of $g - h$,

$$g = \sum_{(i,j)\in M} a_{ij}s_{ij} + h,\ a_{ij} \in A,$$

We have, if $h \neq 0$,

$$\mathrm{lm}(h) = \mathrm{lm}(h_\nu) \cdot \varepsilon_\nu \text{ for some } \nu$$

and $\mathrm{lm}(h) \notin L(S)$ by Lemma 4.7. So, for all j such that $\mathrm{lm}(g_j)$ and $\mathrm{lm}(g_\nu)$ involve the same component, we have $m_{j\nu} \nmid \mathrm{lm}(h_\nu)$. Since $g - h \in \langle S\rangle \subset \mathrm{Syz}(I)$, we obtain

$$f = \varphi(g) = \varphi(h) = \sum h_j g_j.$$

Assume that for some $j \neq \nu$, $\mathrm{lm}(h_j g_j) = \mathrm{lm}(h_\nu g_\nu)$. Then $\mathrm{lm}(h_\nu g_\nu)$ is divisible by $\mathrm{lm}(g_\nu)$ and by $\mathrm{lm}(g_j)$.

Moreover, we are in the following situation: let us denote monomials $\mathrm{lm}(h_\nu) = x^\alpha$, $\mathrm{lm}(g_\nu) = x^\beta$, $\mathrm{lm}(h_j) = x^\gamma$, $\mathrm{lm}(g_j) = x^\delta$. Then $\mathrm{lm}(h_j g_j) = \mathrm{lm}(\mathrm{lm}(h_j)\,\mathrm{lm}(g_j)) = \mathrm{lm}(x^\gamma x^\delta) = x^{\gamma+\delta}$, analogously $\mathrm{lm}(h_\nu g_\nu) = x^{\alpha+\beta}$, so

$\alpha+\beta=\gamma+\delta$. Define τ component-wise by $\tau_i := \max(\beta_i, \delta_i)$, then $x^\tau | x^{\alpha+\beta}$, $x^{\tau-\beta} = m_{j\nu}$ and

$$x^\tau = \mathrm{lm}(\mathrm{lm}(g_\nu)m_{j\nu}) = \mathrm{lm}(g_\nu m_{j\nu}).$$

Thus $\mathrm{lm}(h_\nu g_\nu) = \mathrm{lm}(h_\nu \,\mathrm{lm}(g_\nu))$ is divisible by $\mathrm{lm}(g_\nu m_{j\nu}) = \mathrm{lm}(m_{j\nu}\,\mathrm{lm}(g_\nu))$, hence $m_{j\nu}$ itself divides $\mathrm{lm}(h_\nu)$, what is a contradiction.

In case 1) we obtain $\mathrm{lm}(f) = \mathrm{lm}(h_\nu g_\nu) \in L(G)$, in case 2) it shows that $h \neq 0$ leads to a contradiction. In case 1) G is a Gröbner basis by definition and in case 2) S is a Gröbner basis of $\mathrm{Syz}(I)$ by Theorem 1.16. □

We should note the connection of Schreyer's method with the defining relations on the algebra. Let A be a G–algebra in n variables, such that the maximal left ideal $\mathfrak{m} = {}_A\langle x_1, \ldots, x_n\rangle$ is proper in A. This is equivalent to the fact that $\forall\ i < j$, the structural polynomials d_{ij} do not contain a constant. Then, $d_{ij} = \sum_{k=1}^{n} p_k^{ij} x_k$ and we define a vector $\bar{s} = \bar{s}^{ij}$ component-wise by $s_i = -x_j + p_i^{ij}$, $s_j = c_{ij}x_i + p_j^{ij}$, and $s_k = p_k^{ij}$ for $i \neq k \neq j$. Clearly, $\forall\ i < j\ \bar{s}^{ij} \in \mathrm{Syz}(\mathfrak{m})$. By Schreyer's theorem, $\{\bar{s}^{ij} \mid 1 \leq i < j \leq n\}$ is a Gröbner basis of $\mathrm{Syz}(\mathfrak{m})$. We can prove this fact directly, by using the non–degeneracy conditions.

4.4. Generalized Criteria for Buchberger's Algorithm.

Since their appearance in [**13**], criteria for detecting "useless pairs" are playing an important role in every implementation of Gröbner basis algorithm.

Lemma 4.9. (Chain Criterion) With the notations of Theorem 4.8, assume that $(i,j) \in M$ and $(j,k) \in M$. Let $\mathrm{lm}(f_i) = x^{\alpha_i}e_\nu$, $\mathrm{lm}(f_j) = x^{\alpha_j}e_\nu$ and $\mathrm{lm}(f_k) = x^{\alpha_k}e_\nu$. If x^{α_j} divides $\mathrm{lcm}(x^{\alpha_i}, x^{\alpha_k})$ then $m_{ki}\varepsilon_i \in {}_A\langle m_{ji}\varepsilon_i\rangle$. In particular, if $s_{ij}, s_{ik} \in S$ then $S \setminus \{s_{ik}\}$ is already a Gröbner basis of $\mathrm{Syz}(I)$.

Proof. $x^{\alpha_j} \mid lcm(x^{\alpha_i}, x^{\alpha_k})$ implies that $lcm(x^{\alpha_i}, x^{\alpha_j}) \mid lcm(x^{\alpha_i}, x^{\alpha_k})$. Extracting the exponent vector α_i from both sides, we obtain that m_{ji} divides m_{ki}. □

Remark 4.10. As in the commutative case, we use the chain criterion in the Gröbner Basis Algorithm. Namely, if (f_i, f_j), (f_i, f_k) and (f_j, f_k) are in the set of pairs P and $x^{\alpha_j} \mid \mathrm{lcm}(x^{\alpha_i}, x^{\alpha_k})$, then we can delete (f_i, f_k) from P. Note, that this criterion is quite universal, since it applies without restrictions to any module over any G–algebra. Historically, the ideas, quite similar to the Schreyer's philosophy, were formulated already by D. Anick in mid 80's while studying free and path algebras ([**1**]).

LEMMA 4.11. (Generalized Product Criterion) Let A be a G–algebra of Lie type (that is, all $c_{ij} = 1$). Let $f, g \in A$. Suppose that $\mathrm{lm}(f)$ and $\mathrm{lm}(g)$ have no common factors, then $\mathrm{spoly}(f,g) \to_{\{f,g\}} [g,f]$, where $[g,f] = gf - fg$ denotes the Lie bracket.

PROOF. Assume that f, g are monic polynomials. Let us write $f = L_f + T_f$, where $L_f = \mathrm{lm}(f)$, $T_f = f - L_f$ and, analogously, $g = L_g + T_g$. Then

$$\mathrm{spoly}(f,g) = L_g(L_f + T_f) - L_f(L_g + T_g) = [L_g, L_f] + L_g T_f - L_f T_g.$$

Let us denote $P_1 = [L_g, L_f], P_2 = L_g T_f, P_3 = -L_f T_g$.

If $\mathrm{lm}(P_2) = \mathrm{lm}(P_3)$, we have $\mathrm{lm}(T_f) = \mathrm{lm}(L_f), \mathrm{lm}(T_g) = \mathrm{lm}(L_g)$, a contradiction to the assumption of the lemma. So, in fact $\mathrm{lm}(P_2) \neq \mathrm{lm}(P_3)$. We will examine the following cases and look for values of $S \to_{\{f,g\}}$ for $S = P_1 + P_2 + P_3$.

1) $\mathrm{lm}(P_1) \neq \mathrm{lm}(P_2)$ and $\mathrm{lm}(P_1) \neq \mathrm{lm}(P_3)$;
Then we can perform two reductions and obtain

$$S = [L_g, L_f] + L_g T_f - L_f T_g - T_f g + T_g f = -[f,g]$$

2) $\mathrm{lm}(P_1) = \mathrm{lm}(P_2)$ and $\mathrm{lc}(P_1) = -\mathrm{lc}(P_2)$; we can reduce with g only:

$$S = [L_g, L_f] + L_g T_f - L_f T_g - T_f g = [L_g, L_f] - L_f T_g + [L_g, T_f] - T_f T_g = -[f,g]$$

3) $\mathrm{lm}(P_1) = \mathrm{lm}(P_2)$ and $\mathrm{lc}(P_1) = -\mathrm{lc}(P_2)$; we can reduce only with f:

$$S = [L_g, L_f] + L_g T_f - L_f T_g + T_g f = [L_g, L_f] + L_g T_f - [L_f, T_g] + T_g T_f = -[f,g]$$

4) $\mathrm{lm}(P_1) = \mathrm{lm}(P_2)$ but $\mathrm{lc}(P_1) \neq -\mathrm{lc}(P_2)$:
like case (1), since no real cancellation occur;

5) $\mathrm{lm}(P_1) = \mathrm{lm}(P_3)$ but $\mathrm{lc}(P_1) \neq -\mathrm{lc}(P_3)$:
like case (1), since no real cancellation occur.

Now suppose $p = cf,\ q = dg,\ c, d \in \mathbb{K}^*$. Then $\mathrm{spoly}(p,q) = dL_g cf - cL_f dg = cd(L_g f - L_f g) = cd \cdot \mathrm{spoly}(f,g) \to_{\{f,g\}} cd \cdot gf - cd \cdot fg = [q,p]$. □

REMARK 4.12. In contrast to the Chain Criterion, the Product Criterion is only applicable for ideals and module elements with all module components 0 except one. Of course, if A is commutative or if f and g commute, one gets exactly the statement of the commutative Product Criterion: $\mathrm{spoly}(f,g) \to_{\{f,g\}} 0$. Moreover, the nature of the Product Criterion is *too commutative* with respect to G–algebras, that is, we are profiting from its use significantly only when computing in algebras of Lie type with many commutative variables or dealing with modules, having central elements among their generators.

Unlike the commutative case, when the corresponding pair will be just deleted, we have to perform a reduction, even in the optimized form as above. Here we can gain some speedup, since computing the Lie bracket could be significantly faster than performing monomial-by-monomial reduction steps, especially for large polynomials.

In addition, the routine to compute the Lie bracket can be written more efficient, using the properties of the non–associative bracket:
⋆ Skew symmetry: $\forall a, b \in A \quad [a, b] = -[b, a]$ and $[a, a] = 0$,
⋆ Leibnitz rule: $\forall a, b, c \in A \quad [ab, c] = a[b, c] + [a, c]b$ (that is the map $[\cdot, c] : A \to A$ is a derivation on A).

We are using both criteria and the bracket multiplication in the current implementation ([**53**] and Chapter 4).

Remark 4.13. It is quite intriguing, that on one hand the Product Criterion is too commutative with respect to G–algebras. But on the other hand, it holds in the case of a free associative algebra:

Recall the overlap relation (Def. **§1**, 1.7), and suppose that for two free polynomials $f, g \in T$ their leading monomials have no common sub-word. It means that there exist no such words p, q, that $\mathrm{lm}(f)q = p\,\mathrm{lm}(g)$ and $\mathrm{lm}(f)$ does not divide p, $\mathrm{lm}(g)$ does not divide q.

Then for such pair (f, g) there is no overlap and hence, if there are no self-overlaps (that is overlaps in pairs (f, f) and (g, g)), it does already constitute a Gröbner basis of the two–sided ideal ${}_T\langle f, g\rangle_T$. So, such pairs (f, g) are not even taken into account by the Gröbner basis algorithm in a free associative algebra.

4.5. Hilbert's Syzygy Theorem and SchreyerResolution Algorithm.

We are going to show that the weak Hilbert's syzygy theorem holds for any G–algebra in n variables A, stating that every A–module has a free resolution of length at most n. Although this result is already known ([**30**]), we want to prove it constructively, using Schreyer's method.

Lemma 4.14. Let $G = \{g_1, \ldots, g_s\}$ be a minimal Gröbner basis of $I \subset A^r = \bigoplus_{i=1}^{r} Ae_i$ such that $\mathrm{lm}(g_i) \in \{e_1, \ldots, e_r\}$. Let J denote the set of such indices j, that $e_j \notin \{\mathrm{lm}(g_1), \ldots, \mathrm{lm}(g_s)\}$. Then

$$I = \bigoplus_{i=1}^{s} Ag_i, \quad A^r/I \cong \bigoplus_{j\in J} Ae_j.$$

Proof. The set $G \cup \{e_j \mid j \in J\}$ is A–linearly independent, since the leading terms of its elements are so. This indicates that both sums above

are direct. For $f \in A^r$ consider a standard representation

$$f = \sum_{i=1}^{s} a_i g_i + h,\ \mathrm{lm}(h) \notin L(G).$$

This implies $h \in \underset{j\in J}{\oplus} Ae_j$, and hence the result. □

LEMMA 4.15. Let $G = \{g_1, \ldots, g_s\}$ be a Gröbner basis of $I \subset A^r$, sorted in such a way that the following holds: if $i < j$ and $\mathrm{lm}(g_i) = x^{\alpha_i} e_\nu$, $\mathrm{lm}(g_j) = x^{\alpha_j} e_\nu$ for some ν, then $\alpha_i \geq \alpha_j$ lexicographically. Let s_{ij} denote a generator of the module of syzygies of I as above and we have fixed the Schreyer ordering $>_1$ with respect to G on A^r. If $\mathrm{lm}(g_1), \ldots, \mathrm{lm}(g_s)$ do not depend on the variables $x_1, \ldots, x_k$, then $\mathrm{lm}(s_{ij})$ do not depend on $x_1, \ldots, x_{k+1}$.

PROOF. Given s_{ij}, then $i < j$ and $\mathrm{lm}(g_i)$ and $\mathrm{lm}(g_j)$ involve the same component, say e_ν. By assumption, $\mathrm{lm}(g_i) = x^{\alpha_i} e_\nu$ and $\mathrm{lm}(g_j) = x^{\alpha_j} e_\nu$. The exponent vectors are of the form $\alpha_i = (0, \ldots, 0, \alpha_{i,k+1}, \ldots)$ and $\alpha_j = (0, \ldots, 0, \alpha_{j,k+1}, \ldots)$ with $\alpha_{i,k+1} \geq \alpha_{j,k+1}$. Hence, the exponent vector of m_{ji} has zero at $(k+1)$-th place, since $\max(\alpha_{i,k+1}, \alpha_{j,k+1}) = \alpha_{i,k+1}$. Therefore, $\mathrm{lm}(s_{ij}) = m_{ji} e_i$ does not involve x_{k+1}. □

Applying the lemma to the higher syzygy modules, we obtain the constructive proof of the following theorem.

THEOREM 4.16. Let $<$ be a well–ordering on a G–algebra A in n variables. Then any finitely generated A–module M has a free resolution

$$0 \longrightarrow F_m \longrightarrow F_{m-1} \longrightarrow \ldots \longrightarrow F_0 \longrightarrow M \longrightarrow 0,$$

where F_i are free A–modules, of length $m \leq n$. In particular, $\mathrm{gl.dim}\, A \leq n$.

PROOF. Since A is Noetherian, M has a presentation

$$0 \longrightarrow I \longrightarrow F_0 \longrightarrow M \longrightarrow 0,$$

with $F_0 = \bigoplus_{i=1}^{r_0} Ae_i$. Let $G = \{g_1, \ldots, g_s\}$ be a Gröbner basis of I; assume that the $\mathrm{lm}(g_i)$ do not depend on the variables $x_1, \ldots, x_k$, $k \geq 0$. By the Theorem 4.8, the syzygies $s_{ij}^{(1)} := s_{ij}$ form a Gröbner basis of $\mathrm{Syz}(I)$. By the Lemma 4.15 we have that $\mathrm{lm}(s_{ij})$ do not depend on $x_1, \ldots, x_{k+1}$. Hence, we obtain an exact sequence

$$0 \longrightarrow \mathrm{Ker}\, \varphi_1 = \mathrm{Syz}(I) \longrightarrow F_1 \xrightarrow{\varphi_1} F_0 \longrightarrow M \longrightarrow 0$$

where $F_1 = \bigoplus_{i=1}^{r_1} A\varepsilon_i$, $\varphi_1(\varepsilon_i) = g_i$, $r_1 = s$. By induction, we construct an exact sequence

$$0 \longrightarrow \mathrm{Ker}\, \varphi_{n-k} \longrightarrow F_{n-k} \xrightarrow{\varphi_{n-k}} F_{n-k-1} \longrightarrow \ldots \xrightarrow{\varphi_2} F_1 \xrightarrow{\varphi_1} F_0 \longrightarrow M \longrightarrow 0$$

with F_i free of rank r_i and $\operatorname{Ker}\varphi_{n-k}$ given by a Gröbner basis $\{s_{ij}^{(n-k)}\}$ such that none of the variables appear in $\operatorname{lm}(s_{ij}^{(n-k)})$. By the Lemma 4.14, $F_{n-k}/\operatorname{Ker}\varphi_{n-k}$ is free A–module, hence replacing F_{n-k} by it we obtain, finally, the free resolution we are looking for. So, even if $k = 0$, that is the set $\{\operatorname{lm}(g_i)\}$ depend on all the variables, the length of the resolution is at most n. □

Now we sketch the algorithm for computing a free resolution using Schreyer's method. We assume, that $<$ is a well–ordering on A^r. In the algorithm below, the following auxiliary procedures are used:

- REARRANGE(set of vectors G, set of pairs L): rearranges vectors from G in such a way, that if $i < j$ and $\operatorname{lm}(g_i) = x^{\alpha_i}e_\nu$ and $\operatorname{lm}(g_j) = x^{\alpha_j}e_\nu$, then $\alpha_j <_{\mathtt{lp}} \alpha_i$. The set L of pairs will be filled with all such pairs (i, j) and returned, whereas the set G is changed;
- REFINEWITHCHAINCRITERION(set of vectors G, set of pairs L): refines a set G of vectors with the Chain Criterion 4.9, using the set of pairs L. Returns nothing but changes the set G;
- LIFT (vector p, set of vectors G): computes a left standard representation of p with respect to set $G = \{g_1, \ldots, g_k\}$ (which is assumed to be a Gröbner basis)

$$p = \sum_{s=1}^{k} v_s g_s;$$

and returns a corresponding vector $\bar{v} = (v_1, \ldots, v_k)$. A special case of the more general procedure, see Remark 4.5 and the example thereafter.

Note, that in a G–algebra in n variables, the algorithm will terminate after at most n steps by the Theorem 4.16. But in a GR–algebra, one should give a criterion for termination, what could be either a number of steps or, for certain modules, a presence of a loop in the resolution.

We can build resolutions by other methods, for instance, using successive computation of syzygy modules and optional minimizations. Over G–algebras with homogeneous relations, resolution of homogeneous modules can be computed with the LaScala algorithm, although the generalization of this method is currently under development.

Algorithm 4.1 SchreyerResolution

Input : matrix $G = (g_1, \ldots, g_t)$, a Gröbner basis of $I = {}_A\langle G\rangle$;

Output: a set $\mathcal{F}$ of matrices F_i of size (r_{i-1}, r_i), $i = 1, \ldots, n$ such that

$$\cdots \longrightarrow A^{r_i} \xrightarrow{F_i} A^{r_{i-1}} \longrightarrow \cdots \xrightarrow{F_1} A^{r_0} \longrightarrow A^r/I \longrightarrow 0$$

is a free left resolution.

$J :=$ Rearrange(G) ;
$F_1 := G = \{g_1, \ldots, g_t\}$;
$F_2 := \emptyset$;
for $(i, j) \in J$ **do**
  $h :=$ spoly(g_i, g_j);
  $\bar{a}^{(ij)} :=$ Lift(h, F_1);
  $s_{ij} := m_{ji}\varepsilon_i - \frac{c_i}{c_j} m_{ij}\varepsilon_j - \sum_{v=1}^{t} a_\nu^{(ij)}\varepsilon_\nu$;
  $F_2 := F_2 \cup s_{ij}$;
end for
RefineWithChainCriterion(F_2, J);
$\mathcal{F} := \{F_1\} \cup$ SchreyerResolution(F_2);
return $\mathcal{F}$;

5. Gröbner basics II

5.1. Intersection of Submodules.

Although there exists a generalization of the method 2.13 for intersection of modules, there is another way to compute the intersection of two submodules, which uses syzygies. Moreover, the new method leads us to further advanced applications.

Lemma 5.1. Let $M = {}_A\langle f_1, \ldots, f_k\rangle$ and $N = {}_A\langle g_1, \ldots, g_l\rangle$ be two left submodules of A^r. Let $\{c_1, \ldots, c_{r+k+l}\} \subset A^{2r}$ be the columns of the $2r \times (r + k + l)$–matrix

$$\left(\begin{array}{ccc|ccc|ccc} 1 & & 0 & & & & & & \\ & \ddots & & f_1 & \ldots & f_k & 0 & \ldots & 0 \\ 0 & & 1 & & & & & & \\ \hline 1 & & 0 & & & & & & \\ & \ddots & & 0 & \ldots & 0 & g_1 & \ldots & g_l \\ 0 & & 1 & & & & & & \end{array}\right).$$

Let $S = \mathrm{Syz}(\{c_1, \ldots, c_{r+k+l}\})$. Then $M \cap N = S \cap \bigoplus_{i=1}^{r} Ae_i$.

PROOF.

$$\text{Let } \mathbf{s} = \sum_{i=1}^{r+k+l} s_i e_i \in S \text{ and } \mathbf{s}_1 = \sum_{i=1}^{r} s_i e_i \in S \cap \bigoplus_{i=1}^{r} Ae_i.$$

Since $\mathbf{s}$ is a syzygy,

$$\sum_{i=1}^{r} s_i e_i + \sum_{j=1}^{k} s_{r+j} f_j = 0, \text{ hence } \mathbf{s}_1 = -\sum_{j=1}^{k} s_{r+j} f_j \in M.$$

Using the isomorphism $\tau : \bigoplus_{i=r+1}^{2r} Ae_i \xrightarrow{\cong} \bigoplus_{i=1}^{r} Ae_i$, we obtain

$$0 = \tau(\sum_{i=1}^{r} s_i e_{r+i} + \sum_{j=1}^{l} s_{r+k+j} c_{r+k+j}) = \sum_{i=1}^{r} s_i e_i + \sum_{j=1}^{l} s_{r+k+j} g_j.$$

Hence $\mathbf{s}_1 \in M \cap N$.

Conversely, let $\mathbf{s} = (s_1, \ldots, s_r) \in M \cap N$. Then there are three different presentations of $\mathbf{s}$,

$$\mathbf{s} = \sum_{t=1}^{r} s_t e_t = \sum_{i=1}^{k} a_i f_i = \sum_{j=1}^{l} b_j g_j, \ a_i, b_j \in A,$$

from which we construct the corresponding syzygies. □

Proceeding with the Lemma by induction, we obtain the following result for the intersection of a finite number of modules.

LEMMA 5.2. Let $\{M_i = {}_A\langle f_1^i, \ldots, f_{N_i}^i\rangle \subset A^r,\ 2 \leq i \leq m\}$ be the finite set of modules. Let $t = r + \sum_i N_i$ and $\{c_1, \ldots, c_t\} \subset A^t = \bigoplus_{i=1}^{t} Ae_i$ be the columns of the $(m \cdot r) \times t$–matrix

$$C = \begin{pmatrix} I_{r\times r} & M_1 & 0 & \ldots & 0 \\ I_{r\times r} & 0 & M_2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ I_{r\times r} & 0 & 0 & \ldots & M_m \end{pmatrix}$$

Let $S = \mathrm{Syz}(\{c_1, \ldots, c_t\})$. Then $\bigcap_{i=1}^{m} M_i = S \cap \bigoplus_{i=1}^{r} Ae_i$.

5.2. Kernel of a Module Homomorphism (Modulo).

Let A be a G–algebra, T be a proper two–sided ideal $T \subset A$, already given in its two–sided Gröbner basis $\{t_1, \ldots, t_p\} \subset A$ and consider the GR–algebra $\mathcal{A} = A/T$.

Suppose there are left submodules $U \subset \mathcal{A}^m = \bigoplus_{i=1}^{m} \mathcal{A}e_i$,
$V = {}_A\langle v_1, \ldots, v_k\rangle \subset \mathcal{A}^n$ and left $\mathcal{A}$–modules $M = \mathcal{A}^m/U$ and $N = \mathcal{A}^n/V$.

Consider a left $\mathcal{A}$–module homomorphism

$$\phi: \quad \mathcal{A}^m/U \longrightarrow \mathcal{A}^n/V \qquad e_i \longmapsto \Phi_i,$$

given by the matrix $\Phi \in \mathcal{A}^{n\times m}$.

We are interested in the computation of the kernel of ϕ.

First of all, for a left ideal $I = {}_A\langle g_1, \ldots, g_p\rangle$ we define the *s–th moduliza-tion* to be the left submodule $\mathcal{M}^s(I) \subset A^s$, given by the matrix

$$\mathcal{M}^s(I) = \begin{pmatrix} g_1 & \cdots & g_p & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & g_1 & \cdots & g_p & 0 & \cdots & 0 \\ \vdots & \ddots & & & \ddots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & g_1 & \cdots & g_p \end{pmatrix} \subset A^{s\times ps}.$$

Then $\mathcal{A}^s = (A/T)^s \cong A^s/\mathcal{M}^s(T)$ as A–modules. Defining $U' := U + \mathcal{M}^m(T)$, $V' := V + \mathcal{M}^n(T)$, consider the homomorphism of A–modules

$\psi: \quad A^m \xrightarrow{\Phi} A^n/V'$. Then, $\operatorname{Ker}\phi = \operatorname{NF}(\operatorname{Ker}\psi + U' \mid U')$ and hence it suffices to compute $\operatorname{Ker}\psi$.

As we have already seen in 4.2, the kernel of the homomorphism $A^m \xrightarrow{\Phi} A^n$ is the submodule $\operatorname{Syz}(\Phi) \subset A^m$.

Let $g = \sum_{i=1}^m g_i e_i \in A^m$. It belongs to the $\operatorname{Ker}\psi$ if and only if $\psi(g) \in V' = V + \mathcal{M}^n(T)$, that is if there exist $\{h_i\}, \{r_j\} \subset A$, such that

$$\sum_{i=1}^m g_i \bar{f}_i + \sum_{l=1}^k h_l \bar{v}_l + \sum_{j=1}^{pn} r_j \bar{m}_j = 0.$$

Let $S := \operatorname{Syz}(\{\Phi, V, M^n(T)\}) \subset A^{m+k+pn}$. Then the previous equality means that

$$(g_1, \ldots, g_m, h_1, \ldots, h_k, r_1, \ldots, r_{pn}) \in S.$$

Then, by Lemma 2.2, $\operatorname{Ker}\psi = S \cap \oplus_{i=1}^m Ae_i$.

Computational Remark 5.3. Computing with S as above, we get much overhead. Indeed, we are not interested in syzygies, not relevant to Φ. Therefore, using the Remark 4.4 we can combine two operations (syzygy and Gröbner basis computations) into one. Namely, in order to avoid the computation of irrelevant syzygies, we append the identity matrix from below to Φ and zero matrices to V and $M^n(T)$. In such a manner we obtain a new matrix Y, and call the Gröbner basis routine with the ordering, eliminating module components by the Lemma 2.2, getting the generating set for $\operatorname{Ker}\psi$. This idea appeared in the work of Schönemann ([**70**]). We formalize the described approach in the following Lemma.

LEMMA 5.4. Let $\phi : M \to N$ be a left $\mathcal{A}$–module homomorphism as before. Define the matrix

$$Y = \left(\begin{array}{c|c|c} \Phi & V & \mathcal{M}^n(T) \\ \hline I_{m\times m} & 0 & 0 \end{array}\right) \subset A^{(n+m)\times(m+k+pn)}.$$

Let $Z = Y \cap \overset{n+m}{\underset{i=n+1}{\oplus}} Ae_i$ and $U' = U + \mathcal{M}^m(T)$, then

$$\operatorname{Ker}\phi = \operatorname{NF}(Z + U' \mid U') \subseteq M.$$

Note, that for $K = \operatorname{Ker}\phi$ the inclusion $K^t \cdot \Phi^t \subseteq (V')^t$ holds.

The following Algorithm will be extensively used in this work. Suppose that the procedure for building a matrix Y as in the Lemma is available as MATMODULO(MATRIX Φ, MATRIX V, IDEAL T).

Algorithm 5.1 MODULO

Assume : algebra $\mathcal{A} = A/T$ is given,
Input : matrix Ψ, matrix V, such that
(i) columns of V generate a submodule of $\mathcal{A}^n$
(ii)Ψ defines a homomorphism $\psi : \mathcal{A}^m \xrightarrow{\Psi} \mathcal{A}^n/V$,
Output: matrix K. ▷ $K = \ker\psi$

$Y :=$ MATMODULO(Ψ, V, T);
$Z :=$ ELIMCOMPONENT(Y, $\{n+1, \ldots, n+m\}$); ▷ cf. Remark 2.3
$K := \operatorname{NF}(Z, T)$;
return K;

In our implementation there is a command `modulo` with two arguments. For $\mathcal{A}, \Psi, V$ as above, executing `modulo(`Ψ, V`)` returns the kernel of ψ.

EXAMPLE 5.5 (kernel of a module homomorphism). Consider endomorphisms $\tau : \mathcal{A} \to \mathcal{A}$ with $\mathcal{A} = U(\mathfrak{sl}_2)/I$ with the two–sided ideal I, generated by $\{e^2, f^2, h^2 - 1\}$. After computing the two–sided Gröbner basis of I, we see that $\mathcal{A}$ is indeed finite–dimensional with the basis $\{1, e, f, h\}$.

```
LIB "ncalg.lib";
def A = makeUsl2();  setring A;
option(redSB);  option(redTail);
ideal I = e2,f2,h2-1;
I = twostd(I);
print(matrix(I));     // ideal in a compact form
==> h2-1,fh-f,eh+e,f2,2ef-h-1,e2
qring AI = I;   // we move to a GR--algebra
ideal Ke = modulo(e,0);
```

```
  Ke = std(Ke+std(0)); // normalize Ke wrt factor ideal
  Ke;
==> Ke[1]=h-1
==> Ke[2]=e
  ideal Kh = modulo(h-1,0);
  Kh = std(Kh+std(0));
  Kh;
==> Kh[1]=h+1
==> Kh[2]=f
```

Computing with more homomorphisms like in the example, we get the following table of kernels.

For non–zero $k \in \mathbb{K}$, $\ker(\tau : 1 \mapsto e + k) = \ker(\tau : 1 \mapsto f + k) = 0$.

For $k^2 \neq 1$, $\ker(\tau : 1 \mapsto h + k) = 0$.

$\ker(\tau : 1 \mapsto e) = \ker(\tau : 1 \mapsto h + 1) = {}_{\mathcal{A}}\langle e, h - 1\rangle$.

$\ker(\tau : 1 \mapsto f) = \ker(\tau : 1 \mapsto h - 1) = {}_{\mathcal{A}}\langle f, h + 1\rangle$.

COROLLARY 5.6. (**2nd Isomorphism Theorem**)

Let $M_1, M_2 \in \mathcal{A}^\ell$ be two left submodules. By the 2nd Isomorphism Theorem, we have $M_1/(M_1 \cap M_2) \cong (M_1 + M_2)/M_2$.

Illustrating the situation with the diagram $\mathcal{A}^k \xrightarrow{M_1} \mathcal{A}^\ell \xleftarrow{M_2} \mathcal{A}^m$, we see that indeed, $M_1/(M_1 \cap M_2) \cong \mathcal{A}^\ell / \operatorname{Ker}\phi$, where $\phi : \mathcal{A}^k \xrightarrow{M_1} \mathcal{A}^\ell/M_2$.

The presentation matrix for $M_1/(M_1 \cap M_2)$ equals $\operatorname{Ker}\phi$ and hence could be computed as MODULO(M1,M2).

We can compute the intersection of a finite set of modules with the algorithm MODULO in a more general setting, compared to Lemma 5.2.

PROPOSITION 5.7. Let $\mathcal{A}$ be a GR–algebra and $\{M_i = {}_{\mathcal{A}}\langle f_1^i, \ldots, f_{N_i}^i\rangle \subset \mathcal{A}^r,\ 2 \leq i \leq m\}$ be a finite set of submodules. Assume, that each M_i is actually a submodule of $\mathcal{A}^{n_i}$, where $n_i \leq n_{i+1} \leq r$. Consider the left homomorphism of $\mathcal{A}$–modules

$$\phi : \mathcal{A}^m \longrightarrow (\mathcal{A}^{n_1}/M_1) \oplus \cdots \oplus (\mathcal{A}^{n_m}/M_m), \quad e_i \mapsto I_{n_i \times n_i}.$$

Then $\bigcap_{i=1}^{m} M_i$ can be computed as

$$\text{MODULO}\left(\begin{pmatrix} I_{n_1 \times n_1} \\ \vdots \\ I_{n_m \times n_m} \end{pmatrix}, \begin{pmatrix} M_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & M_m \end{pmatrix}\right).$$

REMARK 5.8. (**Preimage of a Submodule**)
For any proper left submodule $W \subset \mathcal{A}^n/V$, ϕ induces a homomorphism $\phi_W : \mathcal{A}^m/U \to \mathcal{A}^n/(V+W)$ and $\phi^{-1}(W) = \operatorname{Ker}\phi_W$. So, the computation of a kernel of a module homomorphism and a preimage of a submodule are equivalent.

Left, Right Kernels and Formal Adjoint

Some applications (like from Algebraic System Theory, cf. [**19**]) require computations of left and right kernels of a polynomial matrix M. Let us adopt the classical definition to the non–commutative situation.

DEFINITION 5.9. Let M be an $n \times m$ matrix with entries in some $\mathbb{K}$–algebra A. Such matrix is said to have *full column (resp. row) rank*, if its columns (resp. rows) are A–linearly independent.

A *left kernel* of M is a full row rank matrix $L \in Mat(\ell \times n, A)$ with a biggest possible ℓ, such that $L \cdot M = 0$.

A *right kernel* of M is a full row rank matrix $R \in Mat(m \times r, A)$ with a biggest possible r, such that $M \cdot R = 0$.

Since operations of transposition and taking the opposite object mutually commute, A. Quadrat proposed in [**19**] the notation of a *formal adjoint*.

DEFINITION 5.10. Let A be a $\mathbb{K}$–algebra and $M \in Mat(n \times m, A)$.

A **formal adjoint** of a left (resp. right) submodule $M \subset A^n$ is the right (resp. left) submodule $\operatorname{Adj}(M) = (M^{\mathrm{opp}})^t \subset (A^{\mathrm{opp}})^m$.

One can immediately see, that $\operatorname{Adj}(\operatorname{Adj}(M)) = M$, so the formal adjoint (like the transposition in the commutative case) has the property of an involution.

PROPOSITION 5.11. Let $\mathcal{A}$ be a GR–algebra and $M \in Mat(n \times m, \mathcal{A})$.
- A left kernel of M can be computed as $\textsc{Modulo}(M^t, 0)^t$.
- A right kernel of M can be computed as $\operatorname{Adj}(\textsc{Modulo}(\operatorname{Adj}(M), 0))$.

PROOF. Since for $S = \operatorname{Syz}(M)$ holds $S^t \cdot M^t = 0$, a left kernel L of M over a G–algebra A could be computed as $L = \operatorname{Syz}(M^t)^t$. In a GR–algebra $\mathcal{A}$, for $K = \textsc{Modulo}(M^t, 0)$ a left kernel of M over $\mathcal{A}$ is just K^t.

Let R be a right kernel of M. Then, $MR = 0$. Passing to the opposite algebra, we have $R^{\mathrm{opp}} M^{\mathrm{opp}} = 0$, hence R^{opp} is a left kernel for a left submodule M^{opp} and can be computed by taking $R = (\operatorname{Syz}(M^{\mathrm{opp}})^t)^t)^{\mathrm{opp}})$ (cf. **§1**, 5). The more general statement follows analogously. □

5.3. Tensor Product and Intersection of Modules.

Let $U \subset \mathcal{A}^m$, $V, W \subset \mathcal{A}^n$ be submodules.

For $L = \mathcal{A}^n/W$ and $N = \mathcal{A}^n/V$ the intersection $L \cap N$ is isomorphic to $\mathcal{A}^n/_{\mathcal{A}}\langle V, W\rangle$.

For any two left modules $M = \mathcal{A}^m/U$ and $N = \mathcal{A}^n/V$,

$M \otimes_{\mathcal{A}} N$ is a left $\mathcal{A} \otimes_{\mathcal{A}} \mathcal{A}$–module.

Since $\mathcal{A}^m \otimes_{\mathcal{A}} \mathcal{A}^n \cong (\mathcal{A} \otimes_{\mathcal{A}} \mathcal{A})^{nm}$ as $\mathcal{A} \otimes_{\mathcal{A}} \mathcal{A}$–modules, consider a submodule $L := U \otimes I_{n\times n} + I_{m\times m} \otimes V \subseteq (\mathcal{A} \otimes_{\mathcal{A}} \mathcal{A})^{nm}$, where a tensor sign means just a tensor multiplication of matrices. Then $M \otimes_{\mathcal{A}} N$ as an $\mathcal{A} \otimes_{\mathcal{A}} \mathcal{A}$–module is isomorphic to $(\mathcal{A} \otimes_{\mathcal{A}} \mathcal{A})^{nm}/L$ (cf. [**42**]).

5.4. Quotient and Annihilator.

Quotient modules. Let A be a $\mathbb{K}$–algebra. Let S, T be non–empty subsets of A. We define the *left quotient of S by T* to be

$$S :_A T := \{a \in A \mid aT \subseteq S\}.$$

LEMMA 5.12. Let A be an integral $\mathbb{K}$–algebra.

1) If S is a left module, then $S :_A T$ is a left ideal for any T.

2) If T is a left module, then $S :_A T$ is a right ideal for any S.

3) If S and T are left modules, $S :_A T$ is a two–sided ideal.

PROOF.

1) For all $a \in S : T$, we have $at \in S$, $\forall t \in T$. Then $\forall b \in A$, $b(at) = (ba)t \in S$. Hence, $\forall b \in A$, $ba \in S :_A T$.

2) For all $a \in S : T$ and any $b \in A$, $(ab)t = a(bt) = at' \in S$. The last statement follows from 1) and 2). □

LEMMA 5.13. Let A be a GR–algebra.

1) If S is a left module and T is a finite set $\{t_1, \ldots, t_m\} \subset A$, then $S :_A T = \bigcap_{i=1}^m S :_A \{t_i\}$. This left ideal can be efficiently computed as MODULO($(t_1, \ldots, t_m)^t, \mathcal{M}^m(I)$).

2) If S and T are (A, A)–bimodules and the two–sided Gröbner basis of T is $\{t_1, \ldots, t_m\} \subset A^\ell$, then $S :_A T = \bigcap_{i=1}^m S :_A \{t_i\}$. This two–sided ideal can be efficiently computed as
TWOSIDEDGRÖBNERBASIS (MODULO($(t_1, \ldots, t_m)^t, \mathcal{M}^m(I)$), NF).

PROOF.

1) It is clear, that $S : \{t\} = \{a \in A \mid at \in S\}$ is just the result of MODULO(t, S). The rest will be done like in 5.7.

2) See Lemma 3.15 of [**14**]. Note, that we get the system of two–sided generators, which requires the execution of the TWOSIDEDGRÖBNERBASIS algorithm. □

Annihilators. Now, let us turn our attention to annihilators.

DEFINITION 5.14. Let M be an A–module and $v \in M$. Then

- the *annihilator of the element v in M* is a left ideal

$$\mathrm{Ann}_A^M(v) = \{a \in A \mid av = 0\} = \langle 0\rangle : \{v\},$$

- the *annihilator of the module M* is a two–sided ideal

$$\mathrm{Ann}_A M = \{a \in A \mid aM = 0\} = \langle 0\rangle : M,$$

- a *primitive ideal* is an annihilator of a simple left A-module.

The annihilator of an element of a module can be obtained by computing syzygies with the following lemma.

LEMMA 5.15. Let $\mathcal{A}$ be a GR–algebra and $M = \mathcal{A}^N/I_M$ be a left $\mathcal{A}$–module. Suppose that I_M is generated by $\{m_1, \ldots, m_k\} \subset \mathcal{A}^N$. For any $m \in M$, $\mathrm{Ann}_{\mathcal{A}}^M(m)$ is the left ideal generated by the first components of generators of the syzygy module $\mathrm{Syz}(m, m_1, \ldots, m_k) \subseteq \mathcal{A}^{k+1}$ and hence, can be computed by MODULO (m, I_M).

PROOF.

$$\forall \overline{a} = (a_0, a_1, \ldots, a_k) \in \mathrm{Syz}(m, m_1, \ldots, m_k),\ a_0 m + \sum_{i=1}^{k} a_i m_i = 0,$$

hence $a_0 m = 0 \mod I_M$.

□

EXAMPLE 5.16. Let $A = U(\mathfrak{sl}_2)$ over the field $\mathbb{K}(\alpha)$. Consider the ideal $I_M = {}_A\langle e, h - \alpha\rangle \subset A$. Then the Verma module $M = A/I_M$ is equal to $\mathbb{K}[f]$ as a vector-space. Performing computations of annihilators of $\{f^n\}$, we obtain that $\mathrm{Ann}_A^M(f) = {}_A\langle e^2, ef - 2\alpha + 2, h - \alpha + 2\rangle$,
$\mathrm{Ann}_A^M(f^3) = {}_A\langle e^4, ef - 4\alpha + 12, h - \alpha + 6\rangle$ and so on.

We conclude, that

$$\forall\, n \in \mathbb{N} \quad \mathrm{Ann}_A^M(f^n) = {}_A\langle e^{n+1}, ef - (n+1)(\alpha - n), h - \alpha + 2n\rangle.$$

LEMMA 5.17. (Annihilator of a Finite Dimensional Module)

Let $\mathcal{A}$ be a GR–algebra and there is a left A–module M with $\operatorname{GKdim}(M) = 0$ and $\dim_{\mathbb{K}} M = d \geq 1$. Suppose that the $\mathbb{K}$–basis of M is $\{v_1, \ldots, v_d\}$. Then

$$\operatorname{Ann}_{\mathcal{A}} M = \bigcap_{i=1}^{d} \operatorname{Ann}_{\mathcal{A}}^{M} v_i,$$

where $\operatorname{Ann}_A^M v_i$ is computed by the previous Lemma (5.15) and the intersection by Lemma 5.7.

EXAMPLE 5.18. Suppose that $\operatorname{char} \mathbb{K} = 0$. For a positive integer N consider the set

$$F_{N+1} = \{e^{N+1}, f^{N+1}, (h-N)\cdot(h-N+2)\cdot\ldots\cdot(h+N)\} \subset U(\mathfrak{sl}_2)$$

and a left ideal $L_N =_{U(\mathfrak{sl}_2)} \langle F_N \rangle$. Let M_N be a left module $U(\mathfrak{sl}_2)/L_N$. Let us denote by T_N a two–sided ideal, generated by F_N.

L_1 and L_2 are indeed two–sided ideals. For $N \geq 3$, L_N are left ideals (see 3.4 for explicit generators of L_3 and T_3). Computing the annihilators, we obtain

$$\operatorname{Ann}_{U(\mathfrak{sl}_2)} M_N = T_{N+2}, \ \forall N \geq 3.$$

5.5. Annihilator of a Finitely Generated Module.

Let A be a G–algebra in variables $x_1, \ldots, x_n$ and $Z = Z(A)$ be its center.

Let $P \subset A^s$, consider the module $M = A^s/P$, generated by the canonical vectors $e_1, \ldots, e_s$. If $\operatorname{GKdim}(M) = 0$, Lemma 5.17 delivers an algorithm for the annihilator. In what follows we assume $\operatorname{GKdim}(M) \geq 1$.

Define the *pre-annihilator* of M to be the left ideal

$$\operatorname{preAnn}_A M \ := \bigcap_{j=1}^{s} \operatorname{Ann}_A^M(e_j) \ \subseteq A.$$

LEMMA 5.19. The following hold:

1) $\operatorname{Ann}_A M = \operatorname{Ann}_A(A/\operatorname{preAnn}_A M)$,

2) for a left ideal L, $\operatorname{Ann}_A A/L = \{a \in L \mid a \cdot r \in L \ \forall r \in A\}$ is the maximal two–sided ideal contained in L,

3) if the maximal ideal $\mathfrak{m} = {}_A\langle x_1, \ldots, x_n\rangle_A$ is proper in A, for a left ideal L we have $\operatorname{Ann}_A A/L = L :_A \mathfrak{m}$,

4) for the center $Z(A)$ of A, we have

$$Z(A) \cap \operatorname{preAnn}_A M = Z(A) \cap \operatorname{Ann}_A M.$$

PROOF. **1)** Let $L := \operatorname{preAnn}_A M$. It is clear, that $L \supset \operatorname{Ann} M =: S$. Since S is a two–sided ideal, $\forall a \in A$, we have $Sa \subseteq S \subset L$. Conversely, let

$T = \mathrm{Ann}(A/L)$. Consider the normal form representation of $m = \sum_i m_i e_i \in M$. Then $m_i \neq 0$ implies $m_i \notin L$. Then, $\forall t \in T$ and for every i, $tm_i \in L$ and hence, $tm = 0$.

2) Indeed, $\mathrm{Ann}\, A/L$ is the maximal two–sided ideal of a left ideal L.

Let $T = \{a \in L \mid a \cdot r \in L \ \forall r \in A\}$. For some $t \in T$, assume there exists $r_0 \in A$ such that $tr_0 \in L \setminus T$. Then there exists such $r_1 \in A$, that $tr_0r_1 \notin L$, what contradicts $t \in T$. Hence, $\forall r \in A, \forall t \in T\ tr \in T$ and T is the maximal two–sided ideal of L. So, $T = \mathrm{Ann}\, A/L$.

3) Consider first $L :_A A = \{a \in A \mid \forall\ b \in A, ab \subseteq L\}$. For every $x \in L :_A A$, and $b \in \mathbb{K} \subset A$ we have $xb \in L$, so $x \in L$ and $L :_A A \subset L$. Hence $L :_A A$ can be written as $\{a \in L \mid \forall\ b \in A, ab \subseteq L\}$, what is equal to $\mathrm{Ann}\, A/L$ as we have already shown. Moreover, we can pass to $\mathfrak{m}$ instead of A inside the formula and get $\mathrm{Ann}\, A/L = \{a \in L \mid \forall\ b \in \mathfrak{m}, ab \subseteq L\} = L :_A \mathfrak{m}$.

4) Since $\mathrm{preAnn}(M) \supset \mathrm{Ann}_A M$ too, hence $Z(A) \cap \mathrm{preAnn}(M) \supset Z(A) \cap \mathrm{Ann}_A M$. Now, suppose $z \in Z(A) \cap \mathrm{preAnn}(M)$.

$$\forall v \in M,\ \exists \{a_j\} \subset A \text{ such that } v = \sum_{j=1}^{N} a_j e_j. \text{ Then } zv = \sum_{j=1}^{N} a_j z e_j = 0,$$

and hence, $z \in \mathrm{Ann}_A M$.

□

We see, that the computation of the annihilator $\mathrm{Ann}(A/L)$ reduces to the computation of $L :_A \mathfrak{m}$. Since both ideals L and $\mathfrak{m}$ are left ideals, the quotient is a two–sided ideal, hence the method we use in the commutative case (cf. **§3**, 3.6 for a variation of it) is not suitable for this situation.

Nevertheless, we proceed with the investigation. Let $L_0 = L$ and for $i \geq 0$, define the i-th *approximation ideal* $L_{i+1} := \{a \in L_i \mid a \cdot x_k \in L_i\ \forall 1 \leq k \leq n\}$. It is easy to see, that each L_i is indeed a left ideal and $L_i \supset L_{i+1}$.

Suppose there exists $t \in L$, such that this element is common to the completely reduced normalized Gröbner bases (say, obtained with the algorithm RedMinGB, cf. 2.1) of L_i and L_{i+1}. It means, that $\{t, tx_1, \ldots, tx_n\} \subset L_i$, hence t is a two–sided generator inside L_i and L_{i+k}, $\forall\ k \geq 1$. Respectively, all such t in L_i form a two–sided ideal, which we denote by $T_{i+1} \subset L_{i+1}$ (we set $T_0 = {}_A\langle 0 \rangle_A$). It is clear, that $T_i \supseteq T_{i+k}, \forall\ k \geq 1$. There are the following inclusions

$$\begin{array}{ccccccccccc}
L = L_0 & \cdots \supset & L_i & \supset & L_{i+1} & \supset \cdots \supset & L_s & \supset & L_{s+1} & \supset \ldots \\
\cup & & \cup & & \cup & & \cup & & \cup & & \\
T_0 & \cdots \subseteq & T_i & \subseteq & T_{i+1} & \subseteq \cdots \subseteq & T_s & = & T_{s+1} & = \ldots .
\end{array}$$

Since A is Noetherian, the ascending sequence of two–sided ideals T_i will always terminate, although in general we cannot predict at which point the sequence will mutually stabilize. But we are able to compute every T_i as the subsequent use of the following algorithm ApproxStep.

Having the Algorithm ApproxStep, the search for the maximal two–sided ideal in a given left ideal can be described as follows. Suppose we have certain termination criterion, encoded as a boolean function Criterion(X, Y) for two arguments, L and T as above. The computation will stop as soon as this function returns TRUE.

Algorithm MaxTwosidedInLeft(L, T,Criterion):

We initialize $L_0 := L$ and $T_0 := 0$. Further on, we compute $\{L, T\} :=$ ApproxStep$(\{L, T\})$ in the While loop with the break condition (Criterion(L, T) ==TRUE).

Algorithm 5.2 ApproxStep

Input: $\{I, T\}$, where $I = L_i$, a set of left generators;
$T = T_i$, a two–sided Gröbner basis;
Output: $\{J, T'\}$, where $J = L_k$, a set of left generators;
$T' = T_k \supset T_i$, a two–sided Gröbner basis.

repeat
  $I :=$ RedMinGB$(I) = \{f_1, \ldots, f_m\}$;
  for $i = 1$ **to** m **do**
    for $j = 1$ **to** n **do**
      $V[i, j] :=$ NF$\big(f_i \cdot x_j \mid I\big)$;
    end for
  end for
  if $V = \langle 0 \rangle$ **then**
    Print "I is a two–sided ideal"; **return** (I); ▷ quit the loop
  end if
  $S :=$ Modulo (Transpose(V) , I);
  if $S = \langle 0 \rangle$ **then**
    Print "0 is the only maximal ideal"; **return** (0); ▷ quit the loop
  end if
  $J :=$ RedMinGB($S^t \cdot I^t$);
  $T' :=$ GetCommonElements(I, J);
  $T' :=$ TwosidedGröbnerBasis(T');
until $(T' \neq T)$
return $(\{J, T'\})$;

PROOF. (of 5.2) **Correctness:** starting from the set of left generators $I \subset A$, we compute at first its its completely reduced normalized Gröbner basis $I = \{f_1, \ldots, f_m\}$. Then for every nonzero $f \in I$ there exists a standard left presentation

$$f = \sum_{i=1}^{m} a_i f_i, \;\; a_i \in A, \text{ such that } \mathrm{lm}(f) \geq \mathrm{lm}(a_i f_i), a_i f_i \neq 0 \; \forall \; i = 1 \ldots m.$$

Then for any $1 \leq i \leq m$, $1 \leq j \leq n$ we compute the normal forms $v_{ij} = \mathrm{NF}(f_i x_j \mid I)$ and standard presentations

$$f_i x_j = \sum_{k=1}^{m} b_k^{ij} f_k + v_{ij}. \text{ Then } f x_j = \sum_{i=1}^{m} a_i f_i x_j = \sum_{i=1}^{m} a_i \sum_{k=1}^{m} b_k^{ij} f_k + \sum_{i=1}^{m} a_i v_{ij}.$$

If all the v_{ij} are zero, the ideal I is indeed two–sided, hence the output will be I itself. If there are some nonzero v_{ij}, we have

$$f \cdot x_j \in {}_A\langle I \rangle \Leftrightarrow \sum_{i=1}^{m} a_i v_{ij} \in {}_A\langle I \rangle \Leftrightarrow \exists \{b_i\} \subset A, \; \sum_{i=1}^{m} a_i v_{ij} + \sum_{i=1}^{m} b_i f_i = 0.$$

Now we are looking for a_i, satisfying the last equality $\quad \forall 1 \leq j \leq n$. We compute the kernel S of the homomorphism of modules, given by the matrix V (which is a transposed matrix to (v_{ij}))

$$S \longrightarrow A^n \xrightarrow{V} A^n / \mathcal{M}^n(I),$$

where $\mathcal{M}^n(I)$ was defined in the Section 5.2.

If $S = 0$, the only two–sided ideal contained in I is the zero ideal. Otherwise, we compute

$$J = S^t \cdot I^t, \text{ that is } J_i = \sum_{j=1}^{m} S_{ij} f_j,$$

and, after all, a minimal left Gröbner basis of J. We repeat this procedure until we get such J, that it has more common elements (as a set of generators) with I, than T_i and the image of a two–sided ideal, generated by these elements T' in A/T_i is nonzero.

Termination: Unfortunately, the algorithm does not terminate in the general case, but only when there is a two–sided ideal, bigger than T or in two exceptional situations ($V = 0$ or $S = 0$), described above. In particular, if T is already the annihilator, we cannot prove it with the algorithm.

Efficiency: The algorithm makes heavy use of Gröbner bases techniques and therefore should be implemented carefully. During each step subroutines LEFTGRÖBNERBASIS, TWOSIDEDGRÖBNERBASIS and MODULO are called; such computations are quite nontrivial even for small examples. In our experimental implementation, all subroutines like above are using the

LeftGröbnerBasis algorithm. On the other side, we know no alternative algorithm for computing the annihilator of a general module. □

Let us illustrate how AproxStep and MaxTwosidedInLeft work in practice, first without a termination criterion.

Example 5.20. Consider $A = U(\mathfrak{sl}_2)$ over the field $\mathbb{K}(\alpha)$. Then from the parametric ideal $L = {}_A\langle e, h-\alpha\rangle \subset A$ we build the Verma module $M = A/L$.

At first, L is already given in its Gröbner basis and no generator of it has the two–sided property. Hence, $T_0 = 0$ and we describe in detail the first call of the algorithm AproxStep(L, T_0).

Computing v_{ij}, we obtain two nonzero elements $v_{12} = \alpha$ and $v_{22} = -2f$. Then, the result of the Modulo computation is a module, generated by the columns of the matrix

$$\begin{pmatrix} h-\alpha & 2f & 0 & e & 2 \\ 0 & \alpha & h-\alpha+2 & 0 & e \end{pmatrix}.$$

Hence the generators of J are $\{(h-\alpha)e,\ 2fe+\alpha(h-\alpha),\ (h-\alpha+2)(h-\alpha),\ e^2,\ 2e+e(h-\alpha)\}$. After computing the Gröbner basis of J, we get

$$L_{(1)} = J = \{\ h^2-2(\alpha-1)h+\alpha^2-2\alpha,\ \ eh-(\alpha-2)e,\ \ 2ef+(\alpha-2)h-\alpha^2,\ \ e^2\ \}.$$

So, $T_{(1)} = T_0$. Performing the second iteration (that is, calling AproxStep$(L_{(1)}, T_{(1)})$), we obtain

$$L_{(2)} = \{\ 4ef + h^2 - 2h - \alpha^2 - 2\alpha, h^3 - \ldots, eh^2 - \ldots, e^2h - \ldots, e^3\ \}.$$

Since there are no common elements between $L_{(1)}, L_{(2)}$, hence $T_{(2)} = 0$ and we perform the iteration again:

$$L_{(3)} = \{\ 4ef+h^2-2h-\alpha^2-2\alpha, h^4-\ldots, eh^3-\ldots, e^2h^2-\ldots, e^3h-\ldots, e^4\ \}.$$

We arrive at the $T_{(3)} = {}_A\langle 4ef + h^2 - 2h - \alpha(\alpha+2)\rangle_A \subset L$. Proceeding with more iterations experimentally, we get no more elements, hence we may conjecture that $T_{(3)}$ is the answer. (See the proof of it in example 5.21.)

Bounds for Gel'fand–Kirillov dimension.

Let M be an A–module. We know, that in general, due to [**59**],

$$\operatorname{GKdim}(\operatorname{End}_A M) \le \operatorname{GKdim}(A/\operatorname{Ann}_A M) \le \operatorname{GKdim}(A).$$

The lower bound is not effective indeed. Consider a simple infinite–dimensional module M, then $\operatorname{End}_A M = \mathbb{K}$, $\operatorname{GKdim}(\operatorname{End}_A M) = 0$ but $\operatorname{GKdim}(A/\operatorname{Ann}_A M) \ge 1$. However, if there are modules for which we

know that $\mathrm{GKdim}(\mathrm{End}_A M) = \mathrm{GKdim}(A/\operatorname{Ann}_A M)$, the bounds becomes effective, if there would be an algorithm for computing $\mathrm{End}_A M$, what seems to be quite nontrivial.

For $T = \operatorname{Ann}_A M$ we have, in particular, $\mathrm{GKdim}(M) = \mathrm{GKdim}(A/L) \leq \mathrm{GKdim}(A/T) \leq \mathrm{GKdim}(A)$.

If the Gel'fand–Kirillov dimension of $A/\operatorname{Ann}_A M$ can be computed from the given data, the algorithm will stop after finitely many steps. Then the CRITERION function will be just the check for the Gel'fand–Kirillov dimensions of A/L and A/T.

For instance, the algorithm MAXTWOSIDEDINLEFT will terminate for holonomic modules , for which $\mathrm{GKdim}(A/\operatorname{Ann}_A M) = 2 \cdot \mathrm{GKdim}(M)$ holds (recall **§1**, 4.12).

Vogan showed in [**76**], that Harish–Chandra modules (hence, also modules in the category $\mathcal{O}$) over universal enveloping algebras $U(\mathfrak{g})$ of finite dimensional semisimple Lie algebras $\mathfrak{g}$ are holonomic.

Having an algorithm for computing the intersection T of $\mathrm{preAnn}(M)$ with the center of an algebra $Z(A)$ (described as the Algorithm **§3**, 2.1), we can compute the Gel'fand–Kirillov dimension of T and compare it with $\mathrm{GKdim}(M)$. Quite often this will give us a hint to the answer, like the following example illustrates.

EXAMPLE 5.21. Let us continue with the example 5.20. We have obtained the ideal $T_{(3)} = \langle 4ef + h^2 - 2h - \alpha(\alpha+2)\rangle \subset L$ with the MAXTWOSIDEDINLEFT up to the third iteration.

Since A/L is a Verma module (hence, by Vogan it is holonomic), $\mathrm{GKdim}(A/L) = 1$ and $\mathrm{GKdim}(A/T_{(3)}) = 2$, we conclude that $\operatorname{Ann}_A A/L = T_{(3)}$.

In this case we may not perform any iterations at all, since intersecting L with the center of A, we obtain $L \cap Z(A) = T_{(3)}$ and $\mathrm{GKdim}(T_{(3)}) = 2 = 2\,\mathrm{GKdim}(L)$. The generator of $T_{(3)}$ is nothing else but the Casimir element C of A minus the central character of L, $\chi(L) = \alpha(\alpha+2)$ (see Chapter 3 for further details).

In order to complete the treatment, consider more interesting examples.

EXAMPLE 5.22. Consider $A = U(\mathfrak{sl}(3,\mathbb{K}))$ for $\operatorname{char}\mathbb{K} = 0$. We consider two modules M and M', described in detail in the forthcoming Example **§3**, 2.6. The center of A is generated by the elements C_2, C_3, explicitly given in the **§5**, 1.3.

The parametric Verma module $M = A/I$, $I = {}_A\langle x_\alpha, x_\beta, x_\gamma, h_\alpha - a, h_\beta - b\rangle$ is holonomic with $\mathrm{GKdim}(M) = 3$. The intersection of $\mathrm{preAnn}(M) = I$ with the center $Z(A)$ gives us $T = {}_A\langle C_2 - a^2 - ab - b^2 - 3a - 3b, C_3 - 2a^3 -$

$3a^2b+3ab^2+2b^3-6a^2+3ab+12b^2+18b\rangle_A$ (we present it, for simplicity, not in two–sided Gröbner basis). We have $\operatorname{GKdim}(A/T) = 6 = 2\operatorname{GKdim}(M)$, hence $T = \operatorname{Ann}_A M$.

The parametric module $M' = A/I'$, $I' = {}_A\langle x_\beta, x_\gamma, h_\alpha - a, h_\beta - b\rangle$, it is not a Verma module. Its intersection with the center $Z(A)$ is equal to $T' = {}_A\langle 3(a+2b+2)C_2 - C_3 - (a+2b)(a+2b+3)(a+2b+6)\rangle_A$ (T' is given by its two–sided Gröbner basis). Computing both dimensions, we obtain $\operatorname{GKdim}(M') = 4$ and $\operatorname{GKdim}(A/T') = 7$. If M' would be holonomic, its annihilator would have dimension $8 = \operatorname{GKdim}(A)$, what is clearly impossible. As for ideals, we see that $I' \subset I$, hence $\operatorname{Ann}_A M' \subseteq \operatorname{Ann}_A M$ and $8 > \operatorname{GKdim}(A/\operatorname{Ann}_A M') \geq \operatorname{GKdim}(A/T) = 6$.

Using MaxTwosidedInLeft Algorithm, we get no more elements than of T' with several iterations. Since $\operatorname{GKdim}(A/T') = 7$, we conjecture that $\operatorname{Ann}_A M' = T'$.

6. Conclusion and Future Work

In the recent work [**12**], M. Brickenstein applied new interesting techniques and tricks for improving the performance of the Buchberger's algorithm in the commutative case. It seems, that many improvements could be generalized to the setting of G–algebras. In particular, some new Criteria for S–pairs were developed and a good impact on the performance was reported. Replacing the usual standard representation (1.8) with the so-called "t–representation" together with a reformulation and use of the Chain Criterion decreased further the number of useless reductions of S–pairs in the algorithm. Also, ideas of Faugère [**26**] might have a generalization to the non–commutative setting.

In contrast to [**14**] and [**56**], we pay much attention to the efficiency of algorithms, where we make use of our experience from working with concrete and sometimes hard examples in our implementation. We believe, that also the secondary applications (like 5.11), which were described, will make the spectrum of possible applications of Gröbner basics even bigger.

We propose to include the computation of different annihilators (5.4) into non–commutative Gröbner basics, due to their exceptional importance. However, the annihilator of a finitely generated module is available at the moment only for holonomic modules; the algorithm we provide is very expensive. This must be investigated further, we hope there will be more results, connecting dimensions of an annihilator of a module with dimensions of module itself, its endomorphism ring et cetera.

As we have seen, many applications are done by following the guidelines, coming from the commutative case. However, we hope, that the intuitive difference has also become clear to the reader. In particular, the non–commutative anomalies, we have encountered in the elimination (Examples 2.10 and 2.11), the whole idea of two–sided Gröbner basis, different role of Criteria (Subsection 4.4), explicitly more complicated syzygies et cetera show, that despite many similarities with the commutative case, one should develop a distinctly different intuition while working with the GR–algebras.

CHAPTER 3

Morphisms of GR–algebras

> He has drawn a scheme, which, as it became clear later, had no remotest connection to the discussion which followed.
>
> Nikolay Klyuev, *Between two chairs*

In this chapter we are going to investigate morphisms between GR–algebras. In particular, we develop the algorithms for computing the preimage of an ideal under a morphism of two GR–algebras and use them for many interesting applications.

This chapter is organized as follows. In the Section 1 we introduce subalgebras and morphisms. We describe several natural subalgebras, especially the center and centralizers (1.1) and discuss their properties and computational methods, which will be continued in 2.6. In 1.2 we turn our attention to maps and elaborate a criterion for a map to be a morphism.

In the Section 2 we investigate the case, where the source of morphism is a commutative algebra (the target being non–commutative). Then the Algorithm 2.1, computing a preimage of an ideal under such a morphism, is presented. It is a building block for a whole family of algorithms, like the algebraic dependency of pairwise commuting polynomials (2.3) and the membership of a polynomial in a commutative subalgebra (2.4).

The whole Section 3 is devoted to the development of an important central character decomposition algorithm, which is formalized in the Algorithm 3.2.

In the Section 4, we generalize this by allowing also the source to be non–commutative and propose two different approaches for computation of kernels and preimages, showing both the merits and the limitations of them.

We provide many interesting examples, relevant to applications. All of them are computed with the implementation of the methods in Plural.

1. Subalgebras and Morphisms

Two recent books, namely by H. Li ([**56**], 2002) and by J. Bueso et.al. ([**14**], 2003) feature many interesting applications of Gröbner bases, related

in particular to Ring Theory and to Representation Theory of algebras. However, the obviously important question such as the algorithmic treatment of morphisms between GR–algebras and computations with subalgebras was not discussed in general before, to the best of our knowledge. In the article [**64**] the authors presented an algorithm for computing the preimage of a left ideal under a special map $\mathbb{K}[s] \to \mathbb{K}\langle x, d \mid dx = xd + 1\rangle$, $s \mapsto xd$ and its generalization for the multivariate case

$$(\bigotimes_{i=1}^{n})_{\mathbb{K}}\, \mathbb{K}[s_i] \longrightarrow (\bigotimes_{i=1}^{n})_{\mathbb{K}}\, \mathbb{K}\langle x_i, d_i \mid d_i x_i = x_i d_i + 1\rangle,\ s_i \mapsto x_i d_i.$$

We must note, that the approach, proposed in [**64**], seems to be too specific, since it originates from the theory of D–modules and it is indeed based on this theory. Therefore, this approach cannot be transfered to the more general situation. However, the methods we propose allow to compute such preimages using a general framework. The corresponding implementation will be a part of the forthcoming SINGULAR:PLURAL library `dmod.lib`, which contains procedures for algorithms in the theory of D–modules.

Our algorithms are of relevance, since there are many applications in Representation Theory, Theoretical Physics and other fields, requiring the algorithmic treatment of morphisms.

We are going to present the algorithms and applications together with the efficient implementation.

1.1. Subalgebras.

For a G–algebra A, there are several natural commutative subalgebras.

- $Z(A) := \{z \in A \mid za = az\ \forall\ a \in A\} \supseteq \mathbb{K}$ is called the **center** of A ([**23**]); indeed, it is defined for any $\mathbb{K}$–algebra;
- if there exists a Cartan subalgebra $H(A)$ ([**23**]), it is commutative;
- if $H(A)$ exists, we can construct a bigger subalgebra $CZ(A) := H(A) \otimes_{\mathbb{K}} Z(A)$, which is commutative;
- Gel'fand–Zetlin subalgebra $GZ(A)$ ([**25**]), if it exists.

Note, that if both $CZ(A)$ and $GZ(A)$ exist, then $GZ(A) \supseteq CZ(A) \supset Z(A)$ holds. Ovsienko ([**65**]) proved, that if $GZ(A)$ exists, it is the biggest commutative subalgebra of A.

Note, that the computation of $Z(A)$ (up to given degree) is implemented while the construction of Gel'fand–Zetlin subalgebra has not been yet completely algorithmized. A Cartan subalgebra $H(A)$ can be trivially computed for universal enveloping algebras of semi–simple Lie algebras and their quantized counterparts; it is not clear whether this notion makes sense in general.

The center and the Gel'fand–Zetlin subalgebra play a specially important role among the commutative subalgebras.

In Representation Theory there are many constructions involving them and there is a need for, in particular, intersection of modules with such subalgebras. Algebraic dependency of pairwise commuting elements is of big importance, for example, in Mathematical Physics.

Notation: Recall, that $[f, a] := fa - af$ for $a \in A$ and f an element from some (A, A)–bimodule.

LEMMA 1.1. Let A be a G–algebra and $S = \{z_1, \ldots, z_m\}, m \geq 2$ a minimal set of generators of $Z = Z(A)$. Consider the two–sided ideal T, generated by S. Then

1) For any pair (z_i, z_j), $i \neq j$, the syzygy module of the set $\{z_i, z_j\}$ is generated by $(-z_j, z_i)^t \in A^2$.

2) If there exists at least one pair (z_i, z_j), such that $\operatorname{lm}(z_i) \mid \operatorname{lm}(z_j)$, then $\{z_1, \ldots, z_m\}$ is strictly contained in a Gröbner basis of T.

PROOF. **1)** It is clear, that $(-z_j, z_i)^t$ is a syzygy of $\{z_i, z_j\}$. We have to show that all syzygies of $\{z_i, z_j\}$ are generated by the $(-z_j, z_i)^t$.

Let $a_1, a_2 \in A$ be such, that $a_1 z_1 + a_2 z_2 = 0$. If $\operatorname{lm} z_1 \nmid \operatorname{lm} z_2$, then we see immediately that $\operatorname{lm}(z_1) | \operatorname{lm}(a_2)$ and $\operatorname{lm}(z_2) | \operatorname{lm}(a_1)$ and hence, the syzygy $(a_1, a_2)^t$ must be a multiple of $(-z_2, z_1)^t$.

If $\operatorname{lm} z_1 | \operatorname{lm} z_2$, then there exist $b_1, b_2 \in A$, such that $z_2 = b_1 z_1 + b_2$, with $b_2 \neq 0$ and $\operatorname{lm}(z_1) \nmid \operatorname{lm}(b_2)$. Then $(a_1 + a_2 b_1) z_1 + a_2 b_2 = 0$ and $\operatorname{lm}(\operatorname{lm}(z_1) \operatorname{lm}(a_1 + a_2 b_1)) = \operatorname{lm}(a_2 b_2)$. Since $\operatorname{lm}(z_1) \nmid \operatorname{lm}(b_2)$, $\operatorname{lm}(z_1)$ must divide $\operatorname{lm}(a_2)$, and, consequently, $\operatorname{lm}(z_2)$ divides $\operatorname{lm}(a_1)$. Hence $(-z_2, z_1)^t$ is a minimal generator of a syzygy module.

2) Take a pair (z_1, z_2) from S with $\operatorname{lm}(z_1) \mid \operatorname{lm}(z_2)$. Then $z_2 = b_1 z_1 + b_2$ with either $b_2 = 0$ or $b_2 \neq 0$ and $\operatorname{lm}(z_1) \nmid \operatorname{lm}(b_2)$. Suppose $b_2 = 0$, then $b_1 \in Z$ by the following argument. Assume that $b_1 \notin Z$, then $\exists b \in A$, such that $[b_1, b] \neq 0$. Now $0 = [z_2 - b_1 z_1, b] = [b, b_1 z_1] = [b, b_1] z_1$, what is true only if $[b_1, b] = 0$.

Now, let $b_2 \neq 0$, hence $\operatorname{lm}(z_1) \nmid \operatorname{lm}(b_2)$. If $b_2 \in Z$, $z_2 - b_1 z_1 \in Z$, what is true if and only if $b_1 \in Z$, what contradicts the minimality of the set S. Hence, $b_2 \notin Z$ and a Gröbner basis of S will get additional elements. □

LEMMA 1.2. Let A be a G–algebra over a field $\mathbb{K}$ and $S = \{z_1, \ldots, z_m\} \subset A$ be a set of polynomials, such that $\forall i \neq j$, $\operatorname{lm}(z_i) \nmid \operatorname{lm}(z_j)$. Suppose that, in addition, there are $\zeta_{ij} \in \mathbb{K}^*$, such that $\forall i < j$, $z_j z_i = \zeta_{ij} z_i z_j$. Consider a left ideal I, generated by S. Then S is a left Gröbner basis of I.

PROOF. Let $\forall i$, $z_i = L_i + T_i$, $L_i = \mathrm{lm}(z_i) \in \mathrm{Mon}(A)$ and $q_{ij} = \dfrac{\mathrm{lc}(L_j z_i)}{\mathrm{lc}(L_i z_j)}$. Then $\mathrm{spoly}(z_i, z_j) = L_j z_i - q_{ij} L_i z_j = (z_j - T_j) z_i - q_{ij}(z_i - T_i) z_j = (\zeta_{ij} - q_{ij}) z_i z_j - T_j z_i + q_{ij} T_i z_j$. By the same argumentation, as in the proof of the Product Criterion **§2**, 4.11 we see, that $\mathrm{lm}(T_j z_i) \neq \mathrm{lm}(T_i z_j)$ and both of these monomials are strictly smaller than $\mathrm{lm}(z_i z_j)$. Hence, $\mathrm{spoly}(z_i, z_j)$ is reduced to zero with respect to the set $\{z_i, z_j\}$, $\forall 1 \leq i < j \leq m$. Hence, by the Buchberger's Criterion **§2**, 1.16, S is a left Gröbner basis of I. □

The situation, where leading monomials of $\{z_1, \ldots, z_m\}$ do not divide each other at all is quite rare, but important. In the following example, both sets of generators of centers S and T consist only of powers of variables.

EXAMPLE 1.3. Taking a Weyl algebra $W_n(\mathbb{F}_p)$ over a field $\mathbb{F}_p$ of prime characteristic p, we can show that $Z(W_n(\mathbb{F}_p)) = \mathbb{K}[\{x_i^p, d_i^p\}]$. Denote the set of the generators of the center by $S = \{x_i^p, d_i^p \mid 1 \leq i \leq n\}$.

In a similar way, it can be proved that in quasi-commutative algebras $\mathbb{K}_q[x_1, \ldots, x_n]$, where q in a primitive, say, p–th root of unity, holds $Z(\mathbb{K}_q[x_1, \ldots, x_n]) = \mathbb{K}[\{x_i^p\}]$. Denote the set of the generators of the center by $T = \{x_i^p \mid 1 \leq i \leq n\}$.

Applying the Lemma 1.2 to the sets of central elements, we conclude that both S and T are two–sided Gröbner bases of corresponding two–sided ideals. Hence, in particular, the modules $W_n(\mathbb{F}_p)/\langle S \rangle$ and $\mathbb{K}_q[x_1, \ldots, x_n]/\langle T \rangle$ are finite dimensional.

DEFINITION 1.4. For a non–empty subset $F \subseteq A^r$ and a subalgebra $S \subseteq A$, the subalgebra $C_A(F, S) = \{a \in S \mid \ [f, a] = 0 \ \forall f \in F\}$ is called the **centralizer of** F **with respect to** S. If A is fixed, we often write just $C(F, S)$.

It is obvious, that $\forall f \in A$ we have $\mathbb{K} \subseteq Z(A) \subseteq C_A(\{f\}, A) \subseteq A$, but in general, $C_A({}_A\langle f \rangle, A) \subsetneqq C_A(\{f\}, A)$.

LEMMA 1.5. Let A be an integral domain over $\mathbb{K}$. Then, for any finitely generated sub–(A, A)–bimodule M, $C_A(M, A) = Z(A)$.

PROOF. Let $M \subseteq A^k$ be generated by $\{f_1, \ldots, f_n\}$. Since $Z(A) \subseteq C_A(M, A)$, we have to prove $C_A(M, A) \subseteq Z(A)$. $\forall \{b_1, \ldots, b_n\} \subset A$ and for any $a \in C_A(M, A)$,

$$0 = \sum_{i=1}^{n} [b_i f_i, a] = \sum_{i=1}^{n} b_i \underbrace{[f_i, a]}_{0} + \sum_{i=1}^{n} [b_i, a] f_i$$

Suppose there exists such $a \in C_A(M, A)$, that $a \notin Z(A)$. Then the inclusion $C_A(\{a\}, A) \subset A$ is strict and hence, there exists such $b_1 \in A$, that

$[b_1, a] \neq 0$. Choosing all other b_i to be centralizing with respect to a, we obtain $[b_1, a]f_1 = 0$ and, hence, $[b_1, a] = 0$, what is a contradiction. This shows that $C_A(M, A) \subseteq Z(A)$ and indeed $C_A(M, A) = Z(A)$. □

The Lemma shows us that proper (and hence, interesting) subalgebras of integral domains appear as centralizers of finite subsets and not of submodules. In particular, Lemma holds true for one–sided ideals of A.

REMARK 1.6. Very often $C_A(f, A)$ is generated by pairwise commutative elements, although this is not true in general. Indeed, since $C_A(f, A)$ is a subalgebra, for any $a_1, a_2 \in C_A(f, A)$, $[a_i, f] = 0$ and hence, $[[a_1, a_2], f] = 0$. Of course, it does not imply $[a_1, a_2] = 0$.

Consider the second Weyl algebra $W_2 = \mathbb{K}\langle x_1, x_2, \partial_1, \partial_2 \mid [\partial_1, x_1] = 1, [\partial_2, x_2] = 1\rangle$ over $\mathbb{K} = \mathbb{C}$. Then, $C(x_1\partial_1 - x_2\partial_2, W_2)$ is generated by the set of monomials $\{x_1x_2, x_1\partial_1, x_2\partial_2, \partial_1\partial_2\}$ and, for instance, $[\partial_1\partial_2, x_1\partial_1] = \partial_1\partial_2 \neq 0$.

Another interesting subalgebra is the centralizer of the polynomial $x_1\partial_2 - x_2\partial_1$: it is generated by $\{\partial_1^2 + \partial_2^2, x_2\partial_1 - x_1\partial_2, x_1\partial_1 + x_2\partial_2, x_1^2 + x_2^2\}$. Again, these generators do not commute pairwise, e.g. $[\partial_1^2 + \partial_2^2, x_1\partial_1 + x_2\partial_2] = 2(\partial_1^2 + \partial_2^2)$.

LEMMA 1.7. Let A be an associative $\mathbb{K}$–algebra, generated by $\{x_1, \dots x_n\}$. If $\forall 1 \leq i \leq n \quad C(x_i, A)$ is a finitely generated subalgebra of A, then the following holds:

1) $Z(A) = \bigcap_{k=1}^{n} C(x_k, A)$,

2) $Z(A) = C(x_1, C(x_2, \dots, C(x_n, A)) \dots)$.

PROOF. **1)** Suppose $a \in \bigcap_{k=1}^{n} C(x_k, A)$. Then $\forall k \quad [a, x_k] = 0$. Since $\forall i$, $[a, x_ix_k] = x_i[a, x_k] + [a, x_i]x_k = 0$, we conclude that $[a, f] = 0$ for all $f \in A$.

2) Since

$$C(x_{n-1}, C(x_n, A)) = \{a \in A \mid [a, x_n] = 0 = [a, x_{n-1}]\} = C(\{x_{n-1}, x_n\}, A),$$

it follows that $C(x_1, C(x_2, \dots, C(x_n, A)) \dots) = C(\{x_1, \dots, x_n\}, A) = Z(A)$. □

Unless A is finite dimensional, the algorithm for computing the centralizer $C(x_n, A)$ (without any extra information) will not terminate. We therefore consider some finite dimensional filtration $\{A_d \mid d \geq 0\}$ on A (usually one takes a (weighted) degree filtration, cf. **§1**, 4.2) and compute the centers $Z(A_d)$ of the corresponding finite dimensional vector spaces via centralizers, according to the Lemma.

In algebras, where the center is finitely generated, for the maximal degree of generators of the center d, we obtain $Z(A_d) = Z(A)$.

The computational idea with 1.7, **1)** is the following: let $d \geq 1$ be the maximal degree. Then $A_d = \{f \in A \mid \deg(f) \leq d\}$. Suppose M_d to be a $\mathbb{K}$–basis of A_d, and $M_d = \{m_1, \ldots, m_p\}$ such that $m_1 > \cdots > m_p$ with respect to the fixed ordering on A. In order to compute $C(x_t, A_d)$ for some $1 \leq t \leq n$ we do the following.

$$\text{Assume that } z = \sum_{i=1}^{p} a_i m_i, \text{ and } [m_i, x_t] = \sum_{j=1}^{s} b_{ij}^t m_j,\ a_i, b_{ij}^t \in \mathbb{K}. \text{ Then,}$$

$$[z, x_t] = \sum_{j=1}^{s} (\sum_{i=1}^{p} a_i b_{ij}^t) m_j \text{ and } [z, x_t] = 0 \Leftrightarrow \sum_{i=1}^{p} a_i b_{ij}^t = 0\ \forall 1 \leq j \leq s.$$

For fixed t, $1 \leq i \leq p, 1 \leq j \leq s$, form a matrix $B_t = (b_{ij}^t)$ and a vector $\overline{a} = (a_1, \ldots, a_p)^t$. Then

$$z \in C(x_t, A_d) = 0 \Leftrightarrow \overline{a} \in \operatorname{Ker} B_t \text{ and } z \in Z(A_d) \Leftrightarrow \overline{a} \in \bigcap_{t=1}^{n} \operatorname{Ker} B_t.$$

Thus such an algorithm reduces to the linear algebra. However, a good implementation is sophisticated and has to use many tricks and heuristics (the standard linear algebra approaches have to be applied for very big matrices; moreover, we need the exact arithmetics in the field $\mathbb{K}$).

This algorithm for computing the center works in particular for a GR–algebra up to a given degree respectively up to a given number of reduced generators. Recently it has been implemented in SINGULAR:PLURAL as the `center.lib` ([**62**]); we used it for computations of the examples below.

For certain algebras there are explicit theoretical results on generators of the center. In universal enveloping algebras of semi–simple Lie algebras $U(\mathfrak{g})$, there is a notion of Casimir elements and even a formula for a direct computation of them together with the theorem, saying that the center (over a field of characteristic zero) is generated by Casimir elements ([**37**], [**23**]). There is a generalization of this approach to Drinfeld–Jimbo algebras $U_q(\mathfrak{g})$ ([**43**]).

In the framework of G–algebras, it is impossible to provide analogous general results on centers. Hence, we need a general algorithm (though requiring extra input like the degree up to which the algorithm should go) for computing central elements of **any** GR–algebra. Our experience with the implementation in `center.lib` confirms that these principles are right. See Appendix **§5**, 1 and **§5**, 2 for generators of centers of many algebras explicitly.

1.2. Morphisms.

DEFINITION 1.8. Let A and B be associative $\mathbb{K}$–algebras and there is a map $\psi : A \to B$. It is called $\mathbb{K}$–**linear**, if $\psi(1) = 1$ and $\forall x, y \in A$, $\forall \lambda, \mu \in \mathbb{K}$ we have $\psi(\lambda x + \mu y) = \lambda\psi(x) + \mu\psi(y)$.

A $\mathbb{K}$–linear map ψ is called a **homomorphism of $\mathbb{K}$–algebras**,
if $\forall x, y \in A \quad \psi(xy) = \psi(x) \cdot \psi(y)$ holds.

A $\mathbb{K}$–linear map ψ is called an **antihomomorphism of $\mathbb{K}$–algebras**,
if $\forall x, y \in A \quad \psi(xy) = \psi(y) \cdot \psi(x)$ holds.

A $\mathbb{K}$–algebra homomorphism of free associative $\mathbb{K}$–algebras $\psi : A = \mathbb{K}\langle x_1, \ldots, x_n\rangle \longrightarrow B = \mathbb{K}\langle y_1, \ldots, y_m\rangle$ is completely defined by its values on free generators $\{x_i\}$ of A, that is $\psi : x_i \mapsto p_i(y_1, \ldots, y_m)$ for $\{p_1, \ldots, p_n\} \subset B$.

Since every finitely generated associative $\mathbb{K}$–algebra can be presented as a factor–algebra of a free associative $\mathbb{K}$–algebra modulo certain two–sided ideal, a homomorphism $\psi : \mathbb{K}\langle x_1, \ldots, x_n\rangle \longrightarrow \mathbb{K}\langle y_1, \ldots, y_m\rangle$ induces a homomorphism $\Psi : \mathbb{K}\langle x_1, \ldots, x_n\rangle/I \to \mathbb{K}\langle y_1, \ldots, y_m\rangle/J$, if and only if $\psi(I) \subseteq J$.

Moreover, it can be easily seen that every homomorphism of associative $\mathbb{K}$–algebras is induced by some homomorphism of free associative $\mathbb{K}$–algebras. Namely, assume that both I and J as above have finite Gröbner bases for some orderings $<_I$ on $\mathbb{K}\langle x\rangle$ and $<_J$ on $\mathbb{K}\langle y\rangle$. Then, $\Psi(\bar{x}) = \psi(x + I) := \mathrm{NF}(\ \psi(\mathrm{NF}(x, I)),\ J)$ is well–defined, since $\psi(I) \subseteq J$ holds.

Let A and B be G–algebras. Suppose there are proper two–sided ideals $T_A \subset A$, $T_B \subset B$, already given by their two–sided Gröbner bases and there are GR–algebras $\mathcal{A} = A/T_A$ and $\mathcal{B} = B/T_B$.

We call a map $\Phi : \mathcal{A} \to \mathcal{B}$ **a morphism of GR–algebras**, if Φ is a homomorphism of $\mathbb{K}$–algebras and, moreover, $\Phi(T_A) \subseteq T_B$ holds.

Starting with the map $\psi : A \to B$, we define the induced map $\Psi : \mathcal{A} \to \mathcal{B}$ by setting $\Psi(\bar{a}) := \overline{\psi(a)}$, where we can choose $a = \mathrm{NF}(\bar{a} \mid T_A)$ as a representative for $\bar{a} \in \mathcal{A}$.

Notation: The set of all morphisms $\Phi : \mathcal{A} \to \mathcal{B}$ between GR–algebras $\mathcal{A}, \mathcal{B}$ is denoted by $\mathrm{Mor}(\mathcal{A}, \mathcal{B})$. Respectively, we denote by $\mathrm{Mor}(A, B)$ the set of morphisms between G–algebras A, B.

In the sequel, we will deal mostly with morphisms.

DEFINITION 1.9. Let $A, B \in \mathcal{GR}$ and $\Psi : \mathcal{A} \to \mathcal{B}$ be a map.
Define the (i, j) **obstruction polynomial** related to Ψ to be
$o_{ij} := \Psi(\bar{x}_j\bar{x}_i) - \Psi(\bar{x}_j)\Psi(\bar{x}_i)$ and the **ideal of obstructions of** Ψ to be
$O_\Psi := {}_{\mathcal{B}}\langle\{o_{ij} \mid 1 \le i < j \le n\}\rangle$.

REMARK 1.10. In contrast to the commutative case, not every $\mathbb{K}$–linear map of GR–algebras $\Psi : \mathcal{A} \to \mathcal{B}$, satisfying $\Psi(T_A) \subseteq T_B$, is a morphism. From the definition above, we conclude that $\Psi \in \mathrm{Mor}(\mathcal{A}, \mathcal{B}) \Leftrightarrow \mathcal{O}_\Psi = \langle 0 \rangle \subset \mathcal{B} \Leftrightarrow \mathrm{NF}(O_\psi \mid T_B) = 0$. Respectively, $\psi \in \mathrm{Mor}(A, B) \Leftrightarrow O_\psi = \langle 0 \rangle \subset B$.

To every $\phi \in \mathrm{Mor}(A, B)$ and factor–algebras $\mathcal{A}, \mathcal{B}$, which are GR–algebras in the sense of Def. §1, 3.7, we can associate a map $\Phi : x_i \mapsto \overline{\phi(x_i)}$. Then $\mathcal{O}_\Phi = \mathcal{O}_\phi + T_B = T_B$ and hence, $\Phi \in \mathrm{Mor}(\mathcal{A}, \mathcal{B})$ if and only if $\phi(T_A) \subseteq T_B$.

The converse is of course not true: to a morphism $\Psi \in \mathrm{Mor}(\mathcal{A}, \mathcal{B})$ we can not, in general, associate a morphism $\psi \in \mathrm{Mor}(A, B)$ by just setting $\psi(x_i)$ to be the canonical representative of $\Psi(x_i)$, see the following example.

EXAMPLE 1.11. Let $\mathbb{K}$ be a field of characteristic zero, $\mathcal{A} = \mathbb{K}[a, b]/\langle ab \rangle$, $B = \mathbb{K}_q[x, y] = \mathbb{K}(q)\langle x, y \mid yx = q \cdot xy \rangle$ and $\mathcal{B} = \mathbb{K}_q[x, y]/\langle xy \rangle$. Consider the map $\Phi_n : \mathcal{A} \to \mathcal{B}$, given by $a \mapsto x^n$, $b \mapsto y^n$. Since $(xy)^n = q^{\frac{n(n-1)}{2}} x^n y^n$, $\langle \Phi(ab) \rangle = \langle (xy)^n \rangle \subseteq \langle xy \rangle$. Then the obstruction polynomial o_{12} equals $y^n x^n - x^n y^n = (q^{n^2} - 1) x^n y^n$ and we see, that $\mathrm{NF}(o_{12} \mid \langle xy \rangle) = 0$. Hence, Φ_n is a morphism of GR–algebras $\forall n \geq 1$.

A map $\phi_n : \mathbb{K}[a, b] \to \mathbb{K}_q[x, y]$, sending $a \mapsto x^n$, $b \mapsto y^n$, however, is not a morphism unless q is specified at some primitive root of unity, since by the computation above, $o_{12} = (q^{n^2} - 1) x^n y^n$.

REMARK 1.12. In order to illustrate an approach via free algebras, consider the free $\mathbb{K}$–algebras $A = \mathbb{K}\langle a, b \rangle$ and $X = \mathbb{K}(q)\langle x, y \rangle$ together with the ideals $R_A = {}_A\langle ba - ab, ab \rangle_A \subset A$ and $R_X = {}_X\langle yx - qxy, xy \rangle_X \subset X$. Assume that both algebras are equipped with the well–orderings, such that $ba > ab$ in A and $yx > xy$ in X. Then, the reduced Gröbner basis of R_A is $\{ba, ab\}$ and the one of R_X is $\{yx, xy\}$. Let $R'_A = {}_A\langle ba, ab \rangle_A$ and $R'_X = {}_X\langle yx, xy \rangle_X$.

A map of free $\mathbb{K}$–algebras $\varphi_n : A \to X$ which sends $a \mapsto x^n$ and $b \mapsto y^n$ is a morphism. Moreover, $\varphi(R'_A) \subset R'_X$, since $\varphi(ba) = y^n x^n = y^{n-1} \cdot yx \cdot x^{n-1} \in R'_X$ and $\varphi(ab) \in R'_X$ too. Hence, the induced map $\Phi : A/R'_A \to X/R'_X$ is a morphism of $\mathbb{K}$–algebras. Note, that there are the following $\mathbb{K}$–algebra (not GR–algebra!) isomorphisms: $A/R_A \cong A/R'_A \cong \mathcal{A} = \mathbb{K}[a, b]/\langle ab \rangle$ and $X/R_X \cong X/R'_X \cong \mathcal{B}$, for $\mathcal{A}, \mathcal{B}$ from the previous example. Indeed, this shows that we can not in general "lift" morphisms of GR–algebras to morphisms of G–algebras, but to morphisms of free $\mathbb{K}$–algebras instead.

2. Morphisms from Commutative Algebras to GR–algebras

Let $A = \mathbb{K}[y_1, \ldots, y_m]$, $T_A \subset A$ be an ideal and $\mathcal{A} = A/T_A$ be a commutative GR–algebra. Let $B = \mathbb{K}\langle x_1, \ldots, x_n \mid x_j x_i = c_{ij} x_i x_j + d_{ij}, \forall j > i \rangle$ be a G–algebra, $T_B \subset B$ be a two–sided ideal and $\mathcal{B} = B/T_B$ be a GR–algebra.

Let $F = \{f_1, \ldots, f_m\} \subset \mathcal{B}$ be the set of pairwise commuting polynomials. Consider a map of $\mathbb{K}$–algebras $\mathcal{A} \xrightarrow{\phi} \mathcal{B}$, $\phi : y_i \mapsto f_i \in \mathcal{B}$. We assume that $\phi(T_A) \subseteq T_B$.

Then, according to the Remark 1.10, such ϕ is always a morphism.

Suppose there is an ideal $\mathcal{J} \subset \mathcal{B}$. In this section we present an algorithm for computation of the preimage of J under such map.

2.1. Algorithm for Computing the Preimage.

Recall shortly the structure of $\mathcal{E} = \mathcal{A} \otimes_{\mathbb{K}} \mathcal{B}$, described already in **§2**, 3.9.

Let $E = A \otimes_{\mathbb{K}} B$ be the algebra in variables $\{x_i \otimes 1 \mid 1 \leq i \leq n\}$ and $\{1 \otimes y_j \mid 1 \leq j \leq m\}$, which we identify with $\{x_i\}$ and $\{y_j\}$ respectively. Then E is a G–algebra with respect to the block ordering $(<_A, <_B)$

$$E = \mathbb{K}\langle y_1, \ldots, y_m, x_1, \ldots, x_n \mid [y_k, y_\ell] = [y_k, x_i] = 0, x_j x_i = c_{ij} x_i x_j + d_{ij} \rangle,$$

with indices $\forall\ 1 \leq k, \ell \leq m, \forall\ 1 \leq i < j \leq n$.

If T_A and T_B were given as two–sided Gröbner bases, their images in E under canonical inclusions keep this property. Hence, the ideal $T_E = T_A + T_B$ is a two–sided ideal, given by a two–sided Gröbner basis. Then $\mathcal{E} \cong E/T_E$ is a GR–algebra. We denote this construction as $\mathcal{E} = \mathcal{E}(\mathcal{A}, \mathcal{B})$ in the sequel and identify $\mathcal{A}$ and $\mathcal{B}$ with the corresponding admissible subalgebras of $\mathcal{E}$.

THEOREM 2.1. Let $\mathcal{A} = \mathbb{K}[y_1, \ldots, y_m]/T_A$ be commutative and $\mathcal{B}$ be arbitrary GR–algebra, $\Phi \in \mathrm{Mor}(\mathcal{A}, \mathcal{B})$ and $\mathcal{J} \subset \mathcal{B}$ be the left ideal.

Let I_Φ be a left ideal $\langle \{y_i - \phi(y_i) \mid 1 \leq i \leq m\} \rangle \subset \mathcal{E}(\mathcal{A}, \mathcal{B})$. Then

$$\Phi^{-1}(\mathcal{J}) = (I_\phi + \mathcal{J}) \cap \mathcal{A}.$$

PROOF. 1. Consider some polynomial $p = \sum_{\alpha \in \mathbb{N}} c_\alpha y^\alpha \in \mathcal{A}$ with all but a finite number of c_α are zero. For $0 \leq k \leq n$ we define polynomials

$$q_k = \sum_{\alpha \in \mathbb{N}} c_\alpha \left(\prod_{i=1}^{k} y_i^{\alpha_i} \right) \left(\prod_{i=k+1}^{n} \phi(y_i)^{\alpha_i} \right) \in \mathcal{E}.$$

One has $q_0 = \Phi(p)$, $q_n = p$ and $q_k - q_{k+1} \in I_\Phi$ for $0 \leq k \leq n-1$. Then

$$p = q_n + \sum_{k=0}^{n-1} (q_k - q_{k+1}) \in I_\Phi$$

and hence $\forall p \in \mathcal{A}$, $p - \Phi(p) \in I_\Phi$.

2. Since $\Phi(y_i)$ commute pairwise, we have $I_\Phi \cap \mathcal{J} \subseteq I_\Phi \cap \mathcal{B} = 0$. Hence, the sum of ideals is a direct sum and $(I_\Phi + \mathcal{J}) \cap \mathcal{B} = \mathcal{J}$.

3. For any $q \in (I_\Phi + \mathcal{J}) \cap \mathcal{A}$ we can present $\Phi(q)$ as a sum $q + \Phi(q) - q$. Hence, $\Phi(q) \in (I_\Phi + \mathcal{J}) \cap \mathcal{B} = \mathcal{J}$ and the inclusion $\Phi^{-1}(\mathcal{J}) \supset (I_\Phi + \mathcal{J}) \cap \mathcal{A}$ follows.

Let $p \in \Phi^{-1}(\mathcal{J})$. Again one has $p = p - \Phi(p) + \Phi(p) \in (I_\Phi + \mathcal{J}) \cap \mathcal{A}$. This completes the proof. □

The computational part of the theorem is formulated in the following algorithm. We need two sub–algorithms:

TwoSidedGröbnerBasis(ideal I) (**§2**, 3.1)) and

Eliminate(module M, subalgebra S) (**§2**, 2.8).

Note, that the last procedure requires most of the computing time in the algorithm which follows.

We may take $J \subset B$ as input instead of its reduced form $\mathcal{J} = \mathrm{NF}(J + T_B \mid T_B)$, since only the summand $J + T_B$ is used within the algorithm.

Recall, that for an ideal I and a two–sided ideal T_A, we denote $\mathrm{NF}(I + T_A \mid T_A)$ simply by "$I \mod T_A$".

Algorithm 2.1 PreimageInCommutativeAlgebra

Input 1:	$A = \mathbb{K}[y_1, \ldots, y_m]$, $T_A \subset A$ an ideal;	$\triangleright\ \mathcal{A}$
Input 2:	B (G–algebra), $T_B \subset B$ (two–sided ideal);	$\triangleright\ \mathcal{B}$
Input 3:	$J \subset B$ (left ideal);	$\triangleright\ \mathcal{J}$
Input 4:	$\{\Phi(y_i)\} \subset B$ (pairwise commuting polynomials);	$\triangleright\ \Phi$
Output:	$\Phi^{-1}(\mathcal{J})$.	

$T_B = \mathrm{TwoSidedGröbnerBasis}(T_B)$;
$E = A \otimes_{\mathbb{K}} B$; $T_E = T_A + T_B$; $\mathcal{E} = E/T_E$; $\triangleright\ \mathcal{E} = \mathcal{E}(\mathcal{A}, \mathcal{B})$
$I_\Phi = \{y_i - \Phi(y_i) \mid 1 \le i \le m\}$;
$P = T_B + I_\Phi + J$; $\triangleright\ P \subset E$
$P = \mathrm{Eliminate}(P, B)$; $\triangleright\ P = P \cap A$
$P = \mathrm{NF}(T_A + P \mid T_A)$;
return P; $\triangleright\ \Phi^{-1}(\mathcal{J}) = (T_A + (T_B + I_\Phi + J) \cap A) \mod T_A$;

2.2. Kernel of a map.

Since $\ker(\Phi) = \Phi^{-1}(\langle 0 \rangle)$, with this theorem one can compute the kernel of a map between a commutative and a non–commutative G–algebra using the formula

$$\ker(\Phi) = (T_A + (T_B + I_\Phi) \cap A) \mod T_A.$$

For the rest of this section, let A be a G–algebra and a set of pairwise commuting polynomials $f_1, \ldots, f_k \in A$ be given.

2.3. Algebraic Dependency of Elements.

Speaking on the algebraic dependency of non–commuting polynomials, one usually thinks on polynomials in the free algebra. However, if $\{f_i\}$ commute pairwise, the dependency can be expressed by a polynomial from the commutative ring. We say that $\{f_1, \dots, f_k\}$ are *algebraically dependent*, if they are pairwise commutative and there exists a non–zero polynomial $g \in \mathbb{K}[y_1, \dots, y_k]$ such that $g(f_1, \dots, f_k) = 0$.

Define a morphism $\varphi : \mathbb{K}[y_1, \dots, y_k] \to A$, $\varphi(y_i) = f_i$.

Then any $g \in \ker(\varphi) \setminus \{0\}$ defines an algebraic relation between the $f_1, \dots, f_k$. In particular, $f_1, \dots, f_k$ are algebraically independent if and only if $\ker(\varphi) = 0$. Hence, the check for dependency is computable, since $\ker(\varphi)$ can be computed with the formula of 2.2.

Example 2.2. The Fairlie–Odesskii algebra $U'_q(\mathfrak{so}_3)$ ([**36**]) is an associative unital algebra with generating elements I_1, I_2, I_3 and relations, enlisted in **§5**, 2.2. The relations involve a complex number $q \neq 0, \pm 1$, called a *deformation parameter*. In the limit $q \to 1$, the algebra $U'_q(\mathfrak{so}_3)$ reduces to the enveloping algebra $U(\mathfrak{so}_3)$. Both algebras are, of course, G–algebras.

Recall, that the *p–th Chebyshev polynomial of the first kind* is defined to be

$$T_p(x) = \frac{p}{2} \sum_{k=0}^{[p/2]} \frac{(-1)^k (p-k-1)!}{k!(p-2k)!} (2x)^{p-2k},$$

where $[p/2]$ is the integral part of $p/2$. For example, $T_1(x) = x, T_2(x) = 2x^2 - 1, T_3(x) = 4x^3 - 3x, T_4(x) = 8x^4 - 8x^2 + 1$.

Consider the algebra $U'_q(\mathfrak{so}_3)$. For arbitrary q, the algebra $U'_q(\mathfrak{so}_3)$ has the central element $C = -q^{1/2}(q - q^{-1}) I_1 I_2 I_3 + q I_1^2 + q^{-1} I_2^2 + q I_3^2$, which generates the center of $U'_q(\mathfrak{so}_3)$ when q is not a root of unity.

Let q be a p–th primitive root of unity ($p > 2$), that is $q^p = 1$, $q^{p'} \neq 1$, $1 \leq p' < p$. Then elements $C_k = 2\ T_p\Big(I_k(q - q^{-1})/2\Big)$, $k = 1, 2, 3$, where $T_p(x)$ is Chebyshev polynomial, are also central in $U'_q(\mathfrak{so}_3)$.

Using the algorithm from 2.3, we compute a polynomial, describing the algebraic dependency between C, C_1, C_2 and C_3. Let $f_n \in \mathbb{K}[C, C_1, C_2, C_3]$ be such that $f_n(C, C_1, C_2, C_3) = 0$ for q a n–th primitive root of unity. We use $Q = q^{1/2}$ below to simplify the presentation.

Then, using Algorithms 2.1 and 2.2, we compute

$f_3 = (1 - 2Q)C^3 + (Q + 1)C^2 - 243 C_1 C_2 C_3 + 9(1 - 2Q)(C_1^2 + C_2^2 + C_3^2)$,

$f_4 = C^4 - C^2 - 8C^2(C_1 + C_2 + C_3) - 1024 C_1 C_2 C_3 + 16(C_1 - C_2 - C_3)^2$,

$f_5 = C^5 + Q(3Q^2 - 4Q + 3)C^4 + (3Q^3 - 8Q^2 + 8Q - 3)C^3 - (3Q^2 - 5Q + 3)C^2 - 625(3Q^3 + Q^2 + 2Q - 1) C_1 C_2 C_3 - 25(C_1^2 + C_2^2 + C_3^2)$.

We should note that despite the simplicity of formulation of the algorithm, revealing an algebraic dependency with the method above is one of the hardest computational problems we have ever encountered. In the example above it took us a lot of time and memory to obtain needed elements. We use examples like above further as a very good benchmark test for computer algebra systems.

We believe there might exist other methods for finding dependencies which have lower complexity than the Gröbner basis algorithm we use. The approach via Perron polynomials, presented in [**66**] has its preliminary implementation as the PLURAL library `perron.lib` by O. Motsak and shows much better performance than the approach via elimination (the polynomials in the Example 2.4 were computed using both methods).

Universal enveloping algebras in prime characteristic

Consider a classical simple Lie algebra $\mathfrak{g}$ over a field $\mathbb{K} = \mathbb{F}_p$ of positive characteristic p, which is big enough (in most cases $p > 3$ is enough). Here "classical" means that it corresponds to a simple complex Lie algebra, or, equivalently, that its Killing form is non-degenerate. It is known, that the center $Z(U(\mathfrak{g}))$ of the universal enveloping $U(\mathfrak{g})$ is generated by the "Casimir" elements $c_1, \ldots, c_r$, (which coincide with the generators of the center in the case when $\operatorname{char}\mathbb{K} = 0$) and the elements $g^{[p]} - g$, where $g^{[p]}$ denotes the symbolic p-th power (in the case $\mathfrak{g} = \mathfrak{sl}_n$, they coincide with g^p) and g runs through a basis of $\mathfrak{g}$. These elements are not algebraically independent: it is known that $\operatorname{Kr.dim} Z(U(\mathfrak{g})) = \dim_{\mathbb{K}} \mathfrak{g}$. Moreover, the variety $\operatorname{Spec} Z(U(\mathfrak{g}))$ is not smooth. The natural question arises what kind of singularities can occur in $\operatorname{Spec} Z(U(\mathfrak{g}))$. It was investigated in the work [**69**] by Rudakov and Shafarevich. The following formulation of the conjecture is due to Y. Drozd.

CONJECTURE 2.3. The singularities of $\operatorname{Spec} Z(U(\mathfrak{g}))$ are always *simple*. Moreover, their types are just deformations of the type of the Lie algebra $\mathfrak{g}$. (Recall that both of them correspond to Dynkin diagrams $\mathrm{A}, \mathrm{D}, \mathrm{E}$.)

The Conjecture above is still open, after nearly 40 years of its first appearance (in a different form).

To check it, one can start with simple Lie algebras of rank 2, that is $\{A_2\ (= \mathfrak{sl}_3), B_2 = C_2\ (= \mathfrak{so}_5), G_2\ (= \mathfrak{g}_2)$ and $A_1 \times A_1 = D_2\ (= \mathfrak{sl}_2 \times \mathfrak{sl}_2)\}$. Especially interesting is the case of the algebra $\mathfrak{g}_2$, since it does not belong to the ADE serie.

The similar phenomenon with the algebraic dependency happens, as we have seen in 2.2 in quantum algebras too. So, we formulate the common computational problem as follows.

Problem: Let A be a G–algebra over a field k, such that the center of A over the field of $\mathrm{char}\,0$ is strictly bigger than the constants only. For a given prime number p, the field $\mathbb{K}$ will be either an algebraic extension $k(q)$ with a minimal polynomial for the p–th primitive root of unity (if A is a quantum algebra with the quantum parameter q) or $\mathbb{K} = \mathbb{F}_p$ otherwise.
1. Compute the center $Z(A)$ of A over $\mathbb{K}$ (say, with the `center.lib`).
2. Compute algebraic dependencies between the generators of $Z(A)$ (as described above).
3. Study obtained singularities
3a. factorize polynomials over $\mathbb{K}$,
3b. if we are dealing with the isolated hypersurface singularity, determine the type of a singularity (say, with the SINGULAR library `classify.lib`).

Computationally, the hardest part of the whole procedure is the point 2 and the computing time increases with increasing of p. Moreover, we need procedures for the classification of singularities in small characteristics (3,5,7,11), what has to be done by methods which differ from those used inside the library `classify.lib`.

In order to check the conjecture above, we proceed with the simplest case $\mathfrak{g} = \mathfrak{sl}_2$.

EXAMPLE 2.4. The authors of [**69**] showed, that $\mathrm{Spec}\, Z(U(\mathfrak{sl}_2))$ has singularity of type A_1 for all prime p. We are going to reproduce their results via direct computation.

Let p be a prime number and $\mathbb{F}_p$ be a finite field of $\mathrm{char}\, p$. Let c be the Casimir element of $\mathfrak{sl}_2$ and z_1, z_2, z_3 be the p–adic generators (see **§5**, 1.1 for concrete expressions of these elements). In what follows, c, z_1, z_2, z_3 are treated as commutative variables.

For every p, the dependency polynomial is the sum $F_p(c, z_1, z_2, z_3) = F^p(c) + (-1 \cdot (4z_1z_2 + z_3^2) \mod p)$. The polynomials $F^p(c)$ in one variable are given below for primes p, $3 \leq p \leq 29$ in factorized form.
$F_3 = (c+1) \cdot c^2, \quad F_5 = (c+1) \cdot c^2 \cdot (c+2)^2,$
$F_7 = (c+1) \cdot c^2 \cdot (c-3)^2 \cdot (c-1)^2$, $F_{11} = (c+1) \cdot (c-2)^2 \cdot c^2 \cdot (c+3)^2 \cdot (c-4)^2 \cdot (c-3)^2,$
$F_{13} = (c+1)(c+5)^2(c+2)^2(c-2)^2c^2(c+4)^2(c-3)^2,$
$F_{17} = (c+1)(c+5)^2(c-7)^2(c+3)^2(c-8)^2(c+2)^2c^2(c-3)^2(c-1)^2,$
$F_{19} = (c+1)(c-5)^2(c-8)^2(c+4)^2c^2(c-4)^2(c+3)^2(c-6)^2(c+9)^2(c-3)^2,$

$F_{23} = (c+1)(c-2)^2(c-5)^2(c+11)^2(c+6)^2(c+8)^2c^2(c-1)^2(c-8)^2(c-11)^2(c-3)^2(c-7)^2$,
$F_{29} = (c+1)(c-6)^2(c+8)^2(c-4)^2(c+14)^2(c-8)^2c^2(c-3)^2(c-12)^2(c-5)^2(c+6)^2(c+5)^2(c+2)^2(c+10)^2(c+7)^2$.

For polynomials over $\mathbb{F}_p$, $p \geq 13$, a SINGULAR library `classify.lib` performs the classification of a singularity and, in particular, computes a normal form and a type.

However, for small characteristics $(3, 5, 7, 11)$ one has no implementation at the moment. We are grateful to Yousra Lakhal (Kaiserslautern) for computing these cases for us per hand.

Indeed, all the dependencies above have a singularity of type A_1, what coincides with the type A_1 of a simple Lie algebra $\mathfrak{sl}_2$.

2.4. Subalgebra Membership.

Suppose A is a G–algebra, generated by $\{x_1, \ldots, x_n\}$ and we are given some $f \in A$. Let $S \subseteq A$ be a subalgebra , generated by pairwise commuting $f_1, \ldots, f_k$.

How can we check whether f belongs to S?

If f does not commute with all f_i, it can not belong to S. Hence, the first property to check would be to ensure, that f commutes with every f_i.

Then, we have the two following possibilities to compute the polynomial describing the dependency of f on $\{f_1, \ldots, f_k\}$, cf. [**35**].

1. We define a map $\psi : \mathbb{K}[y_0, \ldots, y_k] \to A$, $y_0 \mapsto f$, $y_i \mapsto f_i$ and compute $\ker(\psi)$ with the Algorithm 2.1. Then we take an ordering $<_0$ with y_0 greater than everything containing $y_1, \ldots, y_k$ on $\mathbb{K}[y_0, \ldots, y_k]$ and compute the Gröbner basis G of $\ker(\psi) \subset \mathbb{K}[y_0, \ldots, y_k]$ with respect to $<_0$. G contains an element g with the leading monomial $\mathrm{lm}(g) = y_0$ if and only if $f \in \mathbb{K}[f_1, \ldots, f_k]$. The polynomial f, written in terms of $f_1, \ldots, f_k$, is then $g - \mathrm{lc}(g)\,\mathrm{lm}(g)$.

2. We define a map $\phi : \mathbb{K}[y_1, \ldots, y_k] \to A$, $y_i \mapsto f_i$, an algebra $B := \mathbb{K}[y_1, \ldots, y_k] \otimes_{\mathbb{K}} A$ and a left ideal $I_\phi = {}_B\langle y_1 - f_1, \ldots, y_k - f_k\rangle$ like in the algorithm. We compute a Gröbner basis G of I_ϕ with respect to the elimination ordering for $x_1, \ldots, x_n$. Then we check whether $\mathrm{NF}(f \mid G)$ does not involve any variable from A. This happens if and only if $f \in \mathbb{K}[f_1, \ldots, f_k]$. The formula for f as a polynomial in $f_1, \ldots, f_k$ is just the normal form polynomial.

EXAMPLE 2.5. Let us continue with the example 2.2. There arises a very natural question: since there is an algebraic dependency, could one of the

known generators of the center C, C_1, C_2 and C_3 belong to the subalgebra, generated by the other three?

We have checked it with both methods (the second method is more helpful in the concrete situation) and obtained a negative answer. Note, that in comparison to finding the dependency explicitly, this procedure is much easier and requires less resources.

Our implementation of the algorithms above in SINGULAR:PLURAL was useful for treating the general situation, exploring several conjectures, posed in [**36**]. In the work [**38**] Iorgov used the explicit form of dependency polynomials and finally showed, that there is a general formula for the dependency, which is moreover expressed in terms of Chebyshev polynomials.

Klimyk and Iorgov posed a conjecture that $\{C, C_1, C_2, C_3\}$ is a minimal generating set of the center in the case where q is a primitive root of unity.

2.5. Intersection of Modules with Commutative Subalgebras.

Suppose we have an ideal $I \subset A$. In order to compute the intersection of I with a subalgebra S, we set up the map $\mathbb{K}[y_1, \ldots, y_k] \xrightarrow{\varphi} A$, $\varphi(y_i) = f_i$ and compute its kernel $K = \ker(\varphi)$ with Algorithm 2.1. Then φ induces a monomorphism $\mathbb{K}[y_1, \ldots, y_k]/K \xrightarrow{\varphi} A$. Let $\mathbb{K}[y_1, \ldots, y_k]/K \supset J = \varphi^{-1}(I)$ be the preimage of I. Since the algorithm guarantees that J is given in Gröbner basis $\{g_1, \ldots, g_s\}$, we finish with the computation of the Gröbner basis of $I \cap S = \langle \varphi(g_1), \ldots, \varphi(g_s) \rangle \subset A$.

EXAMPLE 2.6 (Weight vectors with respect to Gel'fand–Zetlin subalgebra). Consider $A = U(\mathfrak{sl}(3, \mathbb{K}))$ for $\operatorname{char} \mathbb{K} = 0$. Both the algebra and its central elements C_2, C_3 are given explicitly in **§5**, 1.3.

By $p_4 := C_2$ and $p_5 := C_3$ we denote the central elements of $U(\mathfrak{sl}_3)$.

Let $p_3 = h_\alpha^2 + 4x_\alpha y_\alpha - 2h_\alpha$ be the central element of the subalgebra of A, generated by $x_\alpha, y_\alpha, h_\alpha$ (it is isomorphic to $U(\mathfrak{sl}_2)$). Let, moreover, $p_1 = h_\alpha$ and $p_2 = h_\beta$ be the generators of the Cartan subalgebra of A. Then $B_1 = Z(A)$ is generated by the $\{p_4, p_5\}$ and B_2, the Gel'fand–Zetlin subalgebra $GZ(A)$, is generated by $\{p_1, p_2, p_3, p_4, p_5\}$.

Consider the natural maps $\phi_i : B_i \to A$. We want to compute $\mathfrak{I}_i := \phi_i^{-1}(I)$ for certain left ideals I, which will give us the central ($i = 1$) and the Gel'fand–Zetlin ($i = 2$) characters of cyclic modules, for which I is the annihilator of a generator. In fact, one of the nice properties of Gel'fand–Zetlin subalgebra implies that it suffices to compute the Gel'fand–Zetlin character of a module, since the central character will be obtained from it.

1. First of all we perform the computations of kernels and obtain $\ker\phi_1 = \ker\phi_2 = 0$. (This is no longer true if $\operatorname{char}\mathbb{K} > 0$ since then there appear additional generators in the center, cf. Subsection 2.3).

2. Consider the parametric ideal $I = {}_A\langle x_\alpha, x_\beta, h_\alpha - a, h_\beta - b\rangle$, corresponding to the Verma module. Its left Gröbner basis is ${}_A\langle x_\alpha, x_\beta, x_\gamma, h_\alpha - a, h_\beta - b\rangle$. Then

$$\begin{aligned}\mathfrak{I}_2 = \big\langle\ & p_1 - a,\ \ p_2 - b,\ \ p_3 - a^2 - 2a,\\ & p_4 - a^2 - ab - b^2 - 3a - 3b,\\ & p_5 - 2a^3 - 3a^2b + 3ab^2 + 2b^3 - 6a^2 + 3ab + 12b^2 + 18b\ \big\rangle.\end{aligned}$$

Moreover, the fourth and the fifth polynomials of $\mathfrak{I}_2$ generate $\mathfrak{I}_1$. Note that both ideals $\mathfrak{I}_1, \mathfrak{I}_2$ are maximal in the corresponding algebras and the parametric parts of p_4, p_5 are indecomposable polynomials in a, b.

3. Now, let us take another ideal $I = {}_A\langle x_\beta, x_\gamma, h_\alpha - a, h_\beta - b\rangle$ (note that it is already a left Gröbner basis). Then

$$\begin{aligned}\mathfrak{I}_2 = \big\langle\ & p_1 - a,\ \ p_2 - b,\\ & 3p_3 - 4p_4 + (a+2b)(a+2b+6),\\ & 3(a+2b+2)p_4 - p_5 - (a+2b)(a+2b+3)(a+2b+6)\ \big\rangle.\end{aligned}$$

The fourth polynomial of $\mathfrak{I}_2$ generates $\mathfrak{I}_1$. Let $c = a+2b$. Then the parametric parts of $p = (p_1, p_2, p_3, p_4, p_5)$ form a one–parameter family, depending on t (we choose $t = p_3$ here):

$$\big(a,\ b,\ \tfrac{4}{3}t - \tfrac{1}{3}c(c+6),\ t,\ 3(c+2)t + c(c+3)(c+6)\big)\,.$$

2.6. Centers of Certain Factor–algebras.

For universal enveloping algebras of semi–simple Lie algebras $U(\mathfrak{g})$ over a field of $\operatorname{char} 0$, there is a result ([**23**]), saying that for any proper two–sided ideal $I \in U(\mathfrak{g})$,

$$(\star) \qquad Z(U(\mathfrak{g})/I) = Z(U(\mathfrak{g}))/I \cap Z(U(\mathfrak{g})).$$

Although looking very natural, the result fails to be true for non–semi–simple algebras: consider the Heisenberg algebra $H = \mathbb{K}\langle x, y, h \mid [x,y] = h, [h,x] = [h,y] = 0\rangle$, which is a universal enveloping algebra of a solvable Lie algebra. For the ideal $I = {}_H\langle h\rangle_H = {}_H\langle h\rangle$, we have $Z(H) = \mathbb{K}[h]$, $I \cap Z(H) = {}_{\mathbb{K}[h]}\langle h\rangle$ and $Z(H)/I \cap Z(H) = \mathbb{K}$. On the other hand, $H/I \cong \mathbb{K}[x,y]$, hence $Z(H/I) = H/I = \mathbb{K}[x,y]$.

Example 2.7. Let us continue with the example **§2**, 3.5.

Let $\operatorname{char}\mathbb{K} = 0$ and $A = U(\mathfrak{g}_2)$. Let $I = {}_A\langle x_1^3\rangle_A$, its two–sided Gröbner basis consists of 106 elements. The center of A is generated by two elements, z_2 and z_6 of degrees 2 and 6 respectively (they are explicitly given in **§5**, 1.4), denote it by $Z(A) = \mathbb{K}[z_2, z_6]$.

Proceeding with the formula ($\star$), we obtain that
$I \cap Z(A) = {}_{\mathbb{K}[z_2,z_6]}\langle z_2^2 - 6z_2, z_6\rangle$. In particular, $z_6 \in I$ and
$Z(A/I) = \mathbb{K}[z_2]/\langle z_2^2 - 6z_2\rangle \cong \mathbb{K} \oplus \mathbb{K} \cdot \mathrm{NF}(z_2, I)$. The center of the factor algebra of $\mathbb{K}$–dimension 50 is generated by the element
$\mathrm{NF}(z_2, I) = 2x_4y_4 + 2h_\alpha^2 + 6h_\alpha h_\beta + 7h_\beta^2 - 2h_\alpha - 3h_\beta$.

It would be interesting to know, which conditions on an algebra would imply that the formula ($\star$) above holds true.

3. Central Character Decomposition of the Module

Decompositions of modules are of special interest for many branches of algebra. In the non–commutative case, especially in Representation Theory, a particularly important role is played by the decomposition into central characters. The algorithmic treatment of this problem goes back to the diploma thesis [**42**], which we follow.

For the whole section we assume $\mathbb{K}$ to be algebraically closed.

Let A be a G–algebra and $C \subset A$ be a finitely generated commutative subalgebra. Denote by $C^* = \mathrm{Hom}(C, \mathbb{K})$ the set of characters of C which can be identified with the set of maximal ideals of C by taking kernels.

Definition 3.1. Let M be a finite generated A-module and $\chi \in C^*$.

• Define the χ–weight subspace of M with respect to C to be

$$M_\chi = \{v \in M \mid \forall c \in C,\ (c - \chi(c))v = 0\}.$$

• Let the generalized χ–weight subspace of M with respect to C be

$$M^\chi = \left\{v \in M \mid \exists n(v) \in \mathbb{N}, \forall c \in C,\ (c - \chi(c))^{n(v)}v = 0\right\}.$$

• We say that M possesses a weight decomposition (resp. generalized weight decomposition) if

$$M = \bigoplus_{\chi \in C^*} M_\chi \quad (\text{resp. } M = \bigoplus_{\chi \in C^*} M^\chi).$$

• $\mathrm{Supp}_C M = \{\chi \in C^* | M^\chi \neq 0\}$ is called a **support of M with respect to C**.

• We say that M possesses a finite (generalized) weight decomposition with respect to C if M possesses a (generalized) weight decomposition and its support is finite.

One can determine, whether a given element $m \in M$ belongs to M_χ (resp. M^χ) for some $\chi \in C$ by analyzing the ideal $\mathrm{Ann}_A^M m \cap C \subset C$. The last ideal can be computed by the Theorem 2.1.

Let us now concentrate on computation of a generalized weight decomposition and the Zariski closure of the support with respect to the center $Z = Z(A)$ of A. In this case the subspaces M_χ and M^χ are submodules for any $\chi \in Z^*$, what is not true, for example, for Gel'fand–Zetlin subalgebras. The generalized weight decomposition with respect to the center will be called **the central character decomposition**.

Notation: By $V(\mathfrak{I}) \subset \mathbb{A}^n_{\mathbb{K}}$ we denote the set of zeros of an ideal $\mathfrak{I} \subset \mathbb{K}[x_1, \ldots, x_n]$.

Recall the result from the Lemma **§2**, 5.19: for a left $\mathcal{A}$–module M, we have $Z(\mathcal{A}) \cap \mathrm{preAnn}(M) = Z(\mathcal{A}) \cap \mathrm{Ann}_{\mathcal{A}} M$.

Using the Nullstellensatz we obtain the following description of the set $\mathrm{Supp}_Z M$ in terms of ideal $\mathrm{preAnn}(M) \cap Z(A)$, where $\mathrm{preAnn}(M)$ can be computed with the Algorithm **§2**, 5.7, and the intersection with the Algorithm 2.1.

Corollary 3.2. Let A be a G–algebra and M be an A–module. Then the Zariski closure of $\mathrm{Supp}_Z M$ equals $V(\mathrm{preAnn}(M) \cap Z(A))$.

To proceed with the discussion of an algorithm for the computation of M^χ, the notions of central quotient ideal and central quotient module are needed. These notions are quite different from the usual non–commutative quotient ideals (see Subsection **§2**, 5.4 and [**14, 35**]). We denote the central quotient by $(I : J)$ instead of $(I :_Z J)$, since classical quotients will not appear in this section.

Definition 3.3. Let $I \subset \mathcal{A}^N$ be a left submodule and $Z = Z(\mathcal{A})$ be the center of $\mathcal{A}$.

- For $z \in Z$ we define the left submodule $(I : z) := \{v \in \mathcal{A}^N \mid zv \in I\}$.
- For an ideal $\mathfrak{I} \subset Z$ the submodule $I : \mathfrak{I}$ is defined to be

$$(I : \mathfrak{I}) := \{v \in \mathcal{A}^N \mid zv \in I \text{ for all } z \in \mathfrak{I}\}.$$

- The submodule $I : z^\infty$ is defined to be $\varinjlim_{n\in\mathbb{N}} I : z^n$.
- The submodule $I : \mathfrak{I}^\infty$ is called a **central saturation** of I by $\mathfrak{I}$ and is defined to be $\varinjlim_{n\in\mathbb{N}} I : \mathfrak{I}^n$.

The usefulness of central quotient modules in our context is indicated by the following proposition.

PROPOSITION 3.4. Let A be a G–algebra and M be an A–module. Suppose M possesses a finite central character decomposition and $\mathrm{Supp}_Z M$ is finite of cardinality s. If $s = 1$, we have $M \cong M^\chi$. Otherwise,

$$M^\chi \cong A^N/(I_M : \mathfrak{I}_\chi^\infty), \text{ where } \mathfrak{I}_\chi = \bigcap_{\psi \in \mathrm{Supp}_Z M \setminus \{\chi\}} \ker \psi.$$

PROOF. By assumption, $M = \bigoplus_{\psi \in Z^*} M^\psi$. Define a left submodule

$$I_\chi = \sum_{\psi \in Z^* \setminus \{\chi\}} M^\psi + I_M \subset A^N.$$

Obviously $M^\chi \cong A^N/I_\chi$. One has to show that $I_M : \mathfrak{I}_\chi^\infty = I_\chi$.

Since $\mathrm{Supp}_Z M$ is finite, there exists such $n \in \mathbb{N}$, that for all $\psi \in \mathrm{Supp}_Z M$ holds $(\ker \psi)^n M^\psi = 0$. For all $x \in I_\chi$ one has $\mathfrak{I}_\chi^n x \in I$. Thus $I_\chi \subset I_M : \mathfrak{I}_\chi^\infty$.

Taking $x \in A^N \setminus I_\chi$, we see that the image v of x in $M^\chi \cong A^N/I_\chi$ is non–zero. Suppose $x \in I_M : \mathfrak{I}_\chi^\infty$, then there exists such $m \in \mathbb{N}$, that $\mathfrak{I}_\chi^m x \in I_M$. Hence we have also $\mathfrak{I}_\chi^m v = 0$, which contradicts the definition of $\mathfrak{I}_\chi$. □

The computation of a central quotient is much easier than the computation of a classical quotient module (see, for example, [**14**] and the Section **§2**, 5.4).

LEMMA 3.5. Let $\mathcal{A}$ be a GR–algebra, $z \in Z(\mathcal{A})$ be a central element in $\mathcal{A}$ and let $F \subset \mathcal{A}^N$ be a left submodule, generated by $\{f_1, \ldots, f_m\}$. Then the central quotient $(F : z) \subseteq \mathcal{A}^N$ is generated by the first N components of the generators of the syzygy module $\mathrm{Syz}(ze_1, \ldots, ze_N, f_1, \ldots, f_m)$ and hence, can be computed by $\mathrm{MODULO}(z \cdot \mathbb{1}_N, F)$.

PROOF. Let $(a_1, \ldots, a_{N+m}) \in \mathrm{Syz}(ze_1, \ldots, ze_N, f_1, \ldots, f_m) \subset \mathcal{A}^{N+m}$.

$$\text{Then,} \quad \sum_{i=1}^{N} z a_i e_i = -\sum_{i=1}^{m} a_{i+N} f_i.$$

Hence, the tuple $(a_1, \ldots, a_N)$ is an element of $(F : z)$ if and only if

$$\bar{a} = (a_1, \ldots, a_{N+m}) \in \mathrm{Syz}(ze_1, \ldots, ze_N, f_1, \ldots, f_m).$$

□

The advantage of this situation is indicated by the lemma, which follows from the fact that $\mathfrak{I}$ is an ideal in the center of A.

LEMMA 3.6. Let $\{c_1, \ldots, c_n\}$ be the Gröbner basis of $\mathfrak{I} \subset Z$.

$$\text{Then,} \quad (I : \mathfrak{I}) = \bigcap_{i=1}^{n} (I : c_i).$$

In the algorithms we shall use the following auxiliary procedures:

- SETRING(ring A): sets the ring A active;
- ANN(module M, vector v): annihilator of v in M (Lemma 5.15);
- INTERSECTMANYMODULES($P_1, \ldots, P_m$) (Prop. **§2**, 5.2);
- MINASSPRIMES(ideal I): minimal associated prime ideals for the zero–dimensional ideal $I \subset \mathbb{K}[z]$; (`primdec.lib`, see [**21**]).

In the following algorithms we formalize the described approach.

Algorithm 3.1 CENTRALSATURATION

Input : M, a left $\mathcal{A}^N$–submodule, T, an ideal in $Z(\mathcal{A})$;
Output: S, a left $\mathcal{A}^N$–submodule; $\triangleright\ S = M : T^\infty$

function CENTRALQUOTIENT(M, T)
  INT s := SIZE(T);
  MATRIX E := IDENTITYMATRIX(s);
  for i=1 **to** s **do**
    $N[i]$:= MODULO($T[i] \cdot E, M$);
  end for
  S := INTERSECTMANYMODULES($N[1], \ldots, N[s]$);
  return S;
end function

MODULE $Q := 0$;
IDEAL T := GRÖBNERBASIS(T);
$S := M$;
repeat
  Q := CENTRALQUOTIENT(S, T);
  S := CENTRALQUOTIENT(Q, T);
until ($S == Q$)
return S;

PROOF. (of Algorithm 3.1).
Termination: The algorithm CENTRALQUOTIENT clearly terminates. For the CENTRALSATURATION, we see that due to the obvious property $(I : \mathfrak{I}) : \mathfrak{I} = I : \mathfrak{I}^2$, one has an increasing sequence $I : \mathfrak{I} \subset I : \mathfrak{I}^2 \subset \ldots$ of submodules in $\mathcal{A}^N$. It stabilizes by the Noetherian property of $\mathcal{A}$, so the

computation of the $I : \mathfrak{I}^\infty$ will be finished after a finite number of steps.
Correctness: Lemmata 3.5, 3.6 imply the correctness of CentralQuotient. □

Both algorithms 3.1 and 3.2 have been implemented by the author in the Singular:Plural library `ncdecomp.lib` ([**50**]); all the examples from the article have been computed with this implementation.

Algorithm 3.2 CentralCharDecomposition

Input 1: A, a G–algebra;
Input 2: $Z = \{Z_1, \ldots, Z_m\} \subset A$, generators of $Z(A)$;
Input 3: I_M, a left A^N–submodule; ▷ $M \cong A^N/I_M$
Output: R, a list of pairs $\{(\chi, I_\chi)\}$.

```
InitRing ℚ[z] := ℚ[z_1, ..., z_m];
InitMap φ : ℚ[z] → A;  φ(z_i) = Z_i;
SetRing A;
for i=1 to N do
    P[i] := Ann(M, e_i);                    ▷ e_i is the i–th basis vector of A^N
end for
J_M := preAnn(M) := IntersectManyModules(P[1], ..., P[N]);
SetRing ℚ[z];
J_z := PreimageInCommutativeAlgebra(ℚ[z], A, J_M, φ);
if (Dim (J_z) > 0) then
    ErrorMessage = "There is no finite decomposition";
    return ERROR;
else
    List L_0 := MinAssPrimes(J_z);
end if
SetRing A;
List L := φ(L_0);  Int s = Size(L);  List S;
for i=1 to s do
    P := IntersectManyModules(L[1], ..., L[î], ..., L[s]);
    S[i] := TwoSidedGröbnerBasis(P);
end for
List R;
for i=1 to s do
    R[i][1] := S[i];
    R[i][2] := CentralSaturation(I_M, S[i]);
end for
return R;
```

EXAMPLE 3.7. Let us continue with the example 2.6.

The central support of the parametric Verma module $M = A/I$, $I = {}_A\langle x_\alpha, x_\beta, h_\alpha - a, h_\beta - b\rangle = {}_A\langle x_\alpha, x_\beta, x_\gamma, h_\alpha - a, h_\beta - b\rangle$ equals $\chi_1 = \langle p_4 - a^2 - ab - b^2 - 3a - 3b, p_5 - 2a^3 - 3a^2b + 3ab^2 + 2b^3 - 6a^2 + 3ab + 12b^2 + 18b\rangle$, a maximal ideal in $\mathbb{K}[p_4, p_5]$ for any value of parameters a, b. Hence, $M \cong M^{\chi_1}$.

For the parametric module $M' = A/I'$, $I' = {}_A\langle x_\beta, x_\gamma, h_\alpha - a, h_\beta - b\rangle$, we have $\mathrm{Supp}_Z M' = \langle 3(a+2b+2)p_4 - p_5 - (a+2b)(a+2b+3)(a+2b+6)\rangle \subset \mathbb{K}[p_4, p_5]$, an ideal of dimension 1 for any value of parameters a, b. Hence, there exists no finite central decomposition.

EXAMPLE 3.8. Let $A = U(\mathfrak{sl}_2)$ (cf. **§5**, 1.1). Consider a set of generators $S = \{e^3, f^3, h^3 - 4h\} \subset A$ and two ideals therein: I_L, a left ideal and I_T, a two–sided ideal, both generated by S. Gröbner basis computations show $I_L \subset I_T$; both bases are listed explicitly in the Example **§2**, 3.4.

We draw our attention at two finite–dimensional modules:

$M_L = U(\mathfrak{sl}_2)/I_L$ (of dimension 15) and

$M_T = U(\mathfrak{sl}_2)/I_T$ (of dimension 10).

Intersection with the center of A, generated by the polynomial $4ef + h^2 - 2h$, gives us the following supports:

$\mathrm{Supp}_Z M_L = \{z, z-8, z-24\}$ and $\mathrm{Supp}_Z M_T = \{z, z-8\}$.

Then, $M_T = M_T^{(z)} \oplus M_T^{(z-8)} = U(\mathfrak{sl}_2)/\mathfrak{m} \ \oplus \ U(\mathfrak{sl}_2)/I_9$ and

$M_L = M_L^{(z)} \oplus M_L^{(z-8)} \oplus M_L^{(z-24)} = U(\mathfrak{sl}_2)/\mathfrak{m} \ \oplus \ U(\mathfrak{sl}_2)/I_9 \ \oplus \ U(\mathfrak{sl}_2)/I_5$.

Here, we used the ideals $\mathfrak{m} = \langle e, f, h\rangle$, $I_5 = \langle e^3, f^3, ef - 6, h\rangle$ and

$I_9 = \langle 4ef + h^2 - 2h - 8, h^3 - 4h, e^3, f^3, fh^2 - 2fh, eh^2 + 2eh, f^2h - 2f^2, e^2h + 2e^2\rangle$. The $\mathbb{K}$–dimensions of corresponding modules are $1, 5, 9$ respectively.

Note, that modules $U(\mathfrak{sl}_2)/\mathfrak{m}$ and $U(\mathfrak{sl}_2)/I_5$ are simple modules, whereas $U(\mathfrak{sl}_2)/I_9$ is a sum of the three following 3–dimensional simple modules $U(\mathfrak{sl}_2)/\langle e^2, f^2, ef - 2, h\rangle \ \oplus \ U(\mathfrak{sl}_2)/\langle e, f^3, h - 2\rangle \ \oplus \ U(\mathfrak{sl}_2)/\langle e^3, f, h + 2\rangle$.

4. Morphisms between GR–algebras

In comparison to the previous sections, computing preimages of ideals under a general morphism between two GR–algebras is more complicated and therefore it is treated separately.

Through the whole section, we will consider two following examples.

Let W_1 be the Weyl algebra $\mathbb{K}\langle x, d \mid [d, x] = 1\rangle$. For $p \in N$, let $I_p = {}_{W_1}\langle x^p, xd + p\rangle$ and $J_p = {}_{W_1}\langle d^p, xd - p + 1\rangle$ be left ideals.

We are interested in preimages of left ideals I_p, J_p for some $p \in N$ under maps from certain algebras to W_1.

EXAMPLE 4.1.

For $t \in \mathbb{N}$ let $S_t = \mathbb{K}\langle a, b \mid [b, a] = t \cdot a\rangle$ be a universal enveloping algebra of a two–dimensional Lie algebra.

For a fixed $t \geq 2$, we consider the map

$$\psi_t : B_t \longrightarrow W_1, \;\; \psi_t(a) = x^t, \psi_t(b) = xd + t.$$

EXAMPLE 4.2. Let $U(\mathfrak{sl}_2)$ be given in its standard presentation, namely as $\mathbb{K}\langle e, f, h \mid [f, e] = -h, [h, e] = 2e, [h, f] = -2f\rangle$. Moreover, let $S_e = \mathbb{K}\langle e, h \mid [h, e] = 2e\rangle$ and $S_f = \mathbb{K}\langle f, h \mid [h, f] = -2f\rangle$ be two subalgebras of $U(\mathfrak{sl}_2)$. We consider the map

$$\tau : U(\mathfrak{sl}_2) \to W_1, \;\; \tau(e) = x, \tau(f) = -xd^2, \tau(h) = 2xd.$$

Our general setup will be as follows. Let $\mathcal{A} = A/T_A$ and $\mathcal{B} = B/T_B$ be two GR–algebras and $\Phi : \mathcal{A} \longrightarrow \mathcal{B}$ be a map (respectively, a map $\phi : A \longrightarrow B$). Define $f_i := \mathrm{NF}(\Phi(x_i), T_B)$ resp. $f_i := \phi(x_i)$.

4.1. Kernels via Opposite Algebras.

Consider the set $X := \{f - \phi(f) \mid f \in A\} \subseteq A \otimes_{\mathbb{K}} B$. It is naturally $\mathbb{K}$–spanned by $\{x^\alpha - \phi(x^\alpha) \mid \alpha \in \mathbb{N}^n\}$. Let $S = \{x_i - \phi(x_i) \mid 1 \leq i \leq n\}$ be another subset of $A \otimes_{\mathbb{K}} B$.

LEMMA 4.3. There are the following inclusions of $\mathbb{K}$–vector-spaces:

$$X \;\subset\; {}_A\langle S\rangle_{\phi(A)} \;\subseteq\; {}_A\langle S\rangle_B.$$

PROOF. For any $k \in \mathbb{N}$, we have $x_i^{k+1} - f_i^{k+1} = x_i \cdot (x_i^k - f_i^k) + (x_i - f_i) \cdot f_i^k$ and at the same time it equals to $x_i(x_i^k - f_i^k) + f_i^k(x_i - f_i)$. As one can easily see, there are several presentations of $x_i^{k+1} - f_i^{k+1}$ in terms of $\{x_i - f_i\}$. We are particularly interested in two of them, namely

$$x_i^{n+1} - f_i^{n+1} = \sum_{k=0}^{n} x_i^k (x_i - f_i) f_i^{n-k} = \sum_{k=0}^{n} f_i^k (x_i - f_i) x_i^{n-k}.$$

The first presentation contains x_i on the left side and f_i on the right; the second presentation is the other way around. We write forthcoming presentations in these two ways.

For $x_1^{\alpha_1} x_2^{\alpha_2} - f_1^{\alpha_1} f_2^{\alpha_2}$ we write two following presentations:

$$(x_1^{\alpha_1} - f_1^{\alpha_1}) f_2^{\alpha_2} + x_1^{\alpha_1}(x_2^{\alpha_2} - f_2^{\alpha_2}) \;=\; (x_1^{\alpha_1} - f_1^{\alpha_1}) x_2^{\alpha_2} + f_1^{\alpha_1}(x_2^{\alpha_2} - f_2^{\alpha_2}).$$

Hence, for every $\alpha \in \mathbb{N}^n$ holds $x^\alpha - \phi(x^\alpha) =$

$$= \sum_{i=1}^{n} \left(\prod_{j=1}^{i-1} x_j^{\alpha_j} \right) (x_i^{\alpha_i} - f_i^{\alpha_i}) \left(\prod_{k=i+1}^{n} f_k^{\alpha_k} \right) =$$

$$= \sum_{i=1}^{n} \left(\prod_{j=1}^{i-1} f_j^{\alpha_j} \right) (x_i^{\alpha_i} - f_i^{\alpha_i}) \left(\prod_{k=i+1}^{n} x_k^{\alpha_k} \right).$$

Plugging in last formulae corresponding presentations of $(x_i^{\alpha_i} - f_i^{\alpha_i})$, we see, that the first presentation belongs to the $(A, \phi(A))$–bimodule ${}_A\langle S\rangle_{\phi(A)}$, the second one — the $(\phi(A), A)$–bimodule ${}_{\phi(A)}\langle S\rangle_A$.

Since $\phi(A) \subseteq B$, we get inclusions of vector-spaces $X \subset {}_A\langle S\rangle_{\phi(A)} \subseteq {}_A\langle S\rangle_B$ and, by swapping sides, also an inclusion $X \subset {}_{\phi(A)}\langle S\rangle_A \subseteq {}_B\langle S\rangle_A$.

Hence, we can move to the left (resp. right) $A \otimes_{\mathbb{K}} \phi(A)^{\mathrm{opp}}$–module, generated by the set $S^{\mathrm{opp}} := \{x_i - \phi(x_i)^{\mathrm{opp}}\}$. Let us denote the left $A \otimes_{\mathbb{K}} \phi(A)^{\mathrm{opp}}$–module by M and the left $A \otimes_{\mathbb{K}} B^{\mathrm{opp}}$–module by M_B. Then $M \subseteq M_B$ and the equality takes place if and only if $\phi(A) = B$.

Clearly, X itself has no A–module structure in this context and is strictly contained in M as a vector-space. □

Let $E^o := A \otimes_{\mathbb{K}} B^{\mathrm{opp}}$ be another G–algebra, $T_E^o := T_A + T_B^{\mathrm{opp}}$ a two–sided ideal (given as a Gröbner basis) and $\mathcal{E}^o := \mathcal{A} \otimes_{\mathbb{K}} \mathcal{B}^{\mathrm{opp}} = E^o/\langle T_E^o\rangle$ a GR–algebra.

Define the set $S^o := \{x_i - \phi(x_i)^{\mathrm{opp}} \mid 1 \leq i \leq n\} \subset E^o$. We view the (A, B)–bimodule ${}_A\langle S\rangle_B$ as the left ideal $I_\phi^o := {}_{A\otimes_{\mathbb{K}} B^{\mathrm{opp}}}\langle S^o\rangle$.

Respectively, $I_\Phi^o = {}_{\mathcal{A}\otimes_{\mathbb{K}}\mathcal{B}^{\mathrm{opp}}}\langle S^o\rangle$, what is nothing else but $\mathrm{NF}(I_\phi^o, T_E^o)$.

Proposition 4.4. For ϕ, Φ and I_ϕ^o as above, the following holds:
(i) $\phi \in \mathrm{Mor}(A, B)$ if and only if $I_\phi^o \cap B^{opp} = \langle 0\rangle$,
(ii) $\Phi \in \mathrm{Mor}(\mathcal{A}, \mathcal{B})$ if and only if $\mathrm{NF}(I_\phi^o \cap B^{opp} \mid I_B^{opp}) = \langle 0\rangle$.

Proof. Consider the ordering $<_A$ on E^o, which is an elimination ordering for $x_1, \ldots, x_n$. Let us explicitly compute $I_\phi^o \cap B^{opp}$, by computing the left Gröbner basis of I_ϕ^o with respect to $<_A$, according to Lemma **§2**, 2.7. Denote $g_i = x_i - f_i$ and compute every s–polynomial (cf. **§2**, 1.12)

$$\mathrm{spoly}(g_i, g_j) = c_{ij}x_i(x_j - f_j) - x_j(x_i - f_i) = -d_{ij} - c_{ij}f_jx_i + f_ix_j.$$

This expression can be reduced with respect to the generators of I_ϕ^o to

$$f_if_j - c_{ij}f_jf_i - \phi(d_{ij}) = \phi(x_jx_i) - \phi(x_i)\phi(x_j),$$

which is nothing else but the obstruction polynomial o_{ij} for ϕ by Definition 1.9. So, we have obtained the ideal of obstructions $O_\phi = \langle\{o_{ij} | 1 \leq i < j \leq n\}\rangle \subseteq B^{opp}$ and hence, the non–reduced Gröbner basis of I_ϕ^o with respect to $<_A$ equals $I_\phi^o + O_\phi$. Then $I_\phi^o \cap B^{opp} = O_\phi$. Following the Remark 1.10, we come to the final conclusions:
(i) ϕ is a morphism $\Leftrightarrow \mathcal{O}_\phi = \langle 0\rangle \Leftrightarrow I_\phi^o \cap B^{opp} = \langle 0\rangle$,
(ii) Φ is a morphism $\Leftrightarrow \mathcal{O}_\phi \subseteq T_B^{opp} \Leftrightarrow \mathrm{NF}(I_\phi^o \cap B^{opp} \mid T_B^{opp}) = \langle 0\rangle$.
□

One can check vanishing of obstructions also without using Gröbner bases, just by checking, that $\{f_i\}$ satisfy the same relations in $\mathcal{B}$ as $\{x_i\}$ do in $\mathcal{A}$, namely $\forall i<j,\ f_jf_i = c_{ij}f_if_j + d_{ij}(\{f_1,\dots,f_n\})$.

On the other hand, the Proposition delivers a nice characterization for morphisms, which will be used further.

PROPOSITION 4.5. Let $\mathcal{A},\mathcal{B} \in \mathcal{GR}$, respectively A, B are G–algebras. Then the following assertions hold:

(i) for any $\phi \in \mathrm{Mor}(A,B)$, $\quad \ker\phi = I^o_\phi \cap A$,

(ii) for any $\Phi \in \mathrm{Mor}(\mathcal{A},\mathcal{B})$,

$$\ker\Phi = I^o_\Phi \cap \mathcal{A} = \mathrm{NF}(T_A + (T^{opp}_B + I^o_\phi) \cap A \mid T_A).$$

PROOF. Let us prove the general statement. Since Φ is a morphism, by 4.4 we have $T^{\mathrm{opp}}_B \supseteq \mathcal{O}_\phi = I^o_\phi \cap B^{\mathrm{opp}}$, hence $I^o_\Phi \cap \mathcal{B}^{\mathrm{opp}} = \langle 0\rangle$.

For any $q \in I_\Phi \cap \mathcal{A}$ its image $\Phi(q) = (\Phi(q)-q)+q \in I^o_\Phi \cap \mathcal{B}^{\mathrm{opp}} = \langle 0\rangle$, hence $\ker\Phi \supseteq I^o_\Phi \cap \mathcal{A}$ holds.

Conversely, let $p \in \ker\Phi \subset \mathcal{A}$. Then $p = p - \Phi(p) + \Phi(p) \in I^o_\Phi \cap \mathcal{A}$. $\square$

Let $\mathcal{J} \subset \mathcal{B}$ be a two–sided ideal. Then we can compute its preimage

$$\Phi^{-1}(\mathcal{J}) = \ker(\mathcal{A} \xrightarrow{\phi_{\mathcal{J}}} \mathcal{B}_{\mathcal{J}} := B/_B\langle T_B + \mathcal{J}\rangle_B),\ \phi_{\mathcal{J}}(x_i) := \mathrm{NF}(\phi(x_i),\mathcal{J}).$$

EXAMPLE 4.6. Let us continue with the example 4.1.

Let us fix some $t \in \mathbb{N}, t \geq 2$ and recall that $S_t = \mathbb{K}\langle a,b \mid [b,a] = t\cdot a\rangle$ and there is a map $\psi_t : B_t \to W_1$, defined by $\psi_t(a) = x^t, \psi_t(b) = xd + t$.

The command `opposite`, applied to W_1, produces the opposite algebra of W_1, build with the "Reversed PBW basis" method, described in the **§1**, 5.1, that is $W_1^{\mathrm{opp}} = \mathbb{K}\langle D, X \mid XD = DX + 1\rangle$. Let $C_t = B_t \otimes_{\mathbb{K}} W_1^{\mathrm{opp}}$ and $f_1 := X^t$ and $f_2 := DX + t$ will be the opposed images of a and b in C_t.

Now, the ideal $I^o_\psi \subset C_t$ is generated by $\{g_1 = a - X^t,\ g_2 = b - (DX+t)\}$. Choose an elimination ordering with $a, b \gg D, X$ and check the correctness by computing the $I^o_{\psi_t} \cap W_1^{\mathrm{opp}}$.

The s–polynomial of (g_1,g_2) is reduced to $[a - X^t, b - (DX + t)]$ by the Product Criterion **§2**, 4.11. Further on, $[a - X^t, b - (DX + t)] = [a,b] + [X^t, DX] = -ta + tX^t = -t(a - X^t) = -tg_1$ and hence, $\mathrm{spoly}(g_1,g_2) = g_2g_1 - g_1g_2 + tg_1 = 0$ So, $\{g_1,g_2\}$ is a Gröbner basis and $I^o_{\psi_t} \cap W_1^{\mathrm{opp}} = 0$, hence ψ_t is a morphism for all $t \in \mathbb{N}$.

In order to compute the kernel of ψ_t, take an elimination ordering with $D, X \gg a, b$ and compute the Gröbner basis of $\{g_1 = X^t - a,\ g_2 = DX - b + t\}$. It is not so easy to compute as the previous one. If $t = 7$, for instance, it will be $\{DX - b + 7, D^4a - X^3b^4 + 46X^3b^3 - 791X^3b^2 + 6026X^3b - 17160X^3, X^4b^3 - 36X^4b^2 + 431X^4b - 1716X^4 - D^3a, X^5b^2 - 25X^5b + 156X^5 -$

$D^2a, X^6b - 13X^6 - Da, X^7 - a\}$. But nevertheless, there are no elements with leading monomials in a, b only. We have checked it for different t and conjecture, that for any $t \in \mathbb{N}$, $\ker \psi_t = \langle 0 \rangle$.

This example is computed by the following code in PLURAL:

```
int t = 7;
ring B = 0,(a,b),dp;
ncalgebra(1,t*a);
ring W1 = 0,(x,d),dp;
ncalgebra(1,1);
poly pa = x^t;
poly pb = x*d+t;
def W1op = opposite(W1);
setring W1op;
poly pa = oppose(W1,pa);
poly pb = oppose(W1,pb);
def C = B+W1op;
setring C;
poly pa = imap(W1op,pa);
poly pb = imap(W1op,pb);
ideal I = a - pa, b - pb;
I;
==>
   I[1]=a-X7
   I[2]=b-DX-7
eliminate(I,a*b); // is map a morphism?
==>
   _[1]=0
eliminate(I,D*X); // kernel
==>
   _[1]=0
```

EXAMPLE 4.7. Here, we continue with the example 4.2. We are investigating the map $\tau : U(\mathfrak{sl}_2) \to W_1$, defined by $\tau(e) = x, \tau(f) = -xd^2, \tau(h) = 2xd$. Let W_1^{opp} be the opposite algebra of W_1 which is defined as in the previous example.

Let $E = U(\mathfrak{sl}_2) \otimes_{\mathbb{K}} W_1^{opp}$ and I_τ^o be generated by $\{g_1 = e - X, g_2 = f + D^2X, g_3 = h - 2DX\}$.

Computing $I_\tau^o \cap W_1^{opp}$ gives zero: applying the Product Criterion, we see that $\text{spoly}(g_1, g_3) = -2g_1$, $\text{spoly}(g_2, g_3) = 2g_2$ and $\text{spoly}(g_1, g_2) = g_3$. Hence,

$\{g_1, g_2, g_3\}$ is a Gröbner basis with respect to an elimination ordering with $e, f, h \gg D, X$ and indeed $\tau \in \mathrm{Mor}(U(\mathfrak{sl}_2), W_1)$.

Let us compute the kernel of τ. We set an elimination ordering with $D, X \gg e, f, h$ and compute the Gröbner basis of I_τ^o with respect to it. We obtain $\{4ef + h^2 - 2h, Dh + 2f, 2De - h, X - e\} \subset E$ and see, that the polynomial $4ef + h^2 - 2h$ generates the kernel. Note, that this element is the generator of the center of $U(\mathfrak{sl}_2)$, hence the two–sided Gröbner basis of $\ker \tau = {}_{U(\mathfrak{sl}_2)}\langle 4ef + h2 - 2h\rangle$.

In particular, τ induces an injective morphism of GR–algebras

$$0 \longrightarrow U(\mathfrak{sl}_2)/_{U(\mathfrak{sl}_2)}\langle 4ef + h^2 + 2h\rangle \xrightarrow{\tau} W_1.$$

Remark 4.8. However, with this technique we are not able to compute preimages of left ideals from $\mathcal{B}$. We transfer the following trivial (A, A)–bimodule structure on A to $A \otimes_{\mathbb{K}} B^{\mathrm{opp}}$: $\forall a, a' \in A, b \in B^{\mathrm{opp}}$,

$a' \circ (a \otimes b) = a' \circ ((a \otimes 1) \cdot (1 \otimes b)) = (a'a \otimes 1) \cdot (1 \otimes b)$ and $(a \otimes b) \circ a' = (1 \otimes b) \cdot (aa' \otimes 1)$. In particular, $a' \circ (a \otimes b) = (a'a \otimes b) = (a' \otimes b) \circ a$.

Then, for a right ideal $L \subset B$, generated by $\{g_1, \ldots, g_s\}$, a left ideal $L^o \subset A \otimes_{\mathbb{K}} B^{opp}$ gets a left A–module structure and becomes a left $A \otimes_{\mathbb{K}} B^{opp}$–module, generated by $\{1 \otimes g_i^{\mathrm{opp}}\}$. But then, $(L^o + I_\phi^o) \cap A$ is a left ideal instead of a right one as a preimage must be. So, this approach works only for two–sided ideals of B. In order to find preimages of one–sided ideals, we will utilize another module structure.

4.2. Preimages via Induced Module Structure.

Let A and B be two G–algebras, generated by $\{x_1, \ldots, x_n\}$ and, respectively, $\{y_1, \ldots, y_m\}$. Moreover, let $\phi : A \to B$ be a map. Consider the set $G = \{g - \phi(g) \mid g \in A\} \subset A \otimes_{\mathbb{K}} B$ (which is clearly closed with respect to addition). There are the following natural actions of A on B, induced by ϕ:

$a \circ_L b := \phi(a)b$ and $b \cdot a := b \circ_R a := b\phi(a)$.

These actions provide a well–defined left and right A–module structures on B, if $\forall a_{1,2} \in A, b \in B$ $a_1 \circ_L a_2 \circ_L b = (a_1 \cdot a_2) \circ_L b$ and, respectively, $b \circ_R a_1 \circ_R a_2 = b \circ_R (a_1 \cdot a_2)$. Computing with the first formula (the second gives analogous result), we get $a_1 \circ_L a_2 \circ_L b - (a_1 \cdot a_2) \circ_L b = (\phi(a_1)\phi(a_2) - \phi(a_1 a_2))b$, that is the action is well–defined if and only if ϕ is a morphism.

Hence, B is an (A, A)–bimodule. Extending both actions naturally to A by $a_1 \circ_L a_2 := a_1 \cdot a_2$, we turn $A \otimes_{\mathbb{K}} B$ into an (A, A)–bimodule.

Since we have the constructive criterion (Proposition 4.4) for checking, whether a map ϕ is a morphism, from now on we deal with morphisms only. We will use the notation $f_i := \phi(x_i) \in B$.

LEMMA 4.9. $G = {}_A\langle\{x_i - \phi(x_i) \mid 1 \leq i \leq n\}\rangle_A \subset A \otimes_{\mathbb{K}} B$.

PROOF. Let $f \in G$ and $\exists g \in A$ such that $f = g - \phi(g)$. For any $a \in A$, $a \circ_L f = ag - \phi(a)\phi(g) = ag - \phi(ag) \in G$ and the same holds for the multiplication from the right: $f \circ_R a = ga - \phi(g)\phi(a) \in G$. Hence, G is a sub-(A, A)–bimodule of $A \otimes_{\mathbb{K}} B$.

Now we show, that as an (A, A)–bimodule, G is generated by the set $\{x_i - \phi(x_i) \mid 1 \leq i \leq n\}$.

Indeed, $\forall i$ and $\forall k \geq 1$ we have $x_i^{k+1} - f_i^{k+1} = x_i^k \circ_L (x_i - f_i) = (x_i - f_i) \circ_R x_i^k$. Now, let $x^{\alpha'}$ be a monomial. Assume its last positive exponent is the n–th one. Then we can present $x^{\alpha'}$ as $x^{\alpha}x_n$ and conclude, that $x^{\alpha'} - \phi(x^{\alpha'}) = x^{\alpha}x_n - f^{\alpha}f_n = x^{\alpha} \circ_L (x_n - f_n) = (x_n - f_n) \circ_R x^{\alpha}$. Since G is additive, we conclude that G is generated by the same set both from the right and from the left. □

In order to represent the action of A on $A \otimes_{\mathbb{K}} B$, we deform $A \otimes_{\mathbb{K}} B$ into another algebra, which we denote by $A \otimes_{\mathbb{K}}^{\phi} B$, by introducing the additional non–commutative relations between elements of A and B.

The action, written in terms of relations, gives $x_i y_j = f_i y_j, y_j x_i = y_j f_i$. Since B is a G–algebra, $\mathrm{lm}(f_i y_j) = \mathrm{lm}(y_j f_i)$.

For $1 \leq i \leq n$, $1 \leq j \leq m$, define $q_{ij} \in \mathbb{K}$ to be $q_{ij} := \dfrac{\mathrm{lc}(y_j f_i)}{\mathrm{lc}(f_i y_j)}$ and $r_{ij} \in B \subset A \otimes_{\mathbb{K}} B$ to be $r_{ij} := y_j f_i - q_{ij} f_i y_j$. Then, for all indices in the same range as above

$$y_j x_i = q_{ij} \cdot x_i y_j + r_{ij} \quad (\text{ or } [y_j, x_i]_{q_{ij}} = [y_j, f_i]_{q_{ij}}).$$

If $q_{ij} = 1$ (e.g. when B is an algebra of Lie type), we have $r_{ij} = y_j f_i - f_i y_j = [y_j, f_i]$ and relation becomes just $[y_j, x_i] = [y_j, f_i]$ for all $1 \leq i \leq n$, $1 \leq j \leq m$.

It remains to incorporate the relations $(x_i - f_i)y_j = 0 = y_j(x_i - f_i)$, $\forall 1 \leq i \leq n$ and $\forall 1 \leq j \leq m$. Since in $A \otimes_{\mathbb{K}}^{\phi} B$, $y_j(x_i - f_i) = (x_i - f_i)y_j$, it suffices to consider a two–sided ideal $R_\phi \subset A \otimes_{\mathbb{K}}^{\phi} B$, generated by $\{(x_i - f_i)y_j \mid \forall 1 \leq i \leq n, \forall 1 \leq j \leq m\}$. One possibility for treating the situation correctly would be to pass to factor–algebra $A \otimes_{\mathbb{K}}^{\phi} B/R_\phi$. On the other hand, $R_\phi \subseteq G$ and hence, in the computations below we do not need it, since we need not $I_\Phi := G$ alone, but rather the sum $I_\Phi + \mathcal{J}$. According to the Lemma **§2**, 3.8, the computation of a Gröbner basis of an ideal J in the factor–algebra modulo R_Φ is done by computing a Gröbner basis of the ideal $J + R_\Phi$. Hence, the Gröbner basis of $I_\Phi + \mathcal{J}$ in the factor–algebra is the Gröbner basis of $I_\Phi + \mathcal{J} + R_\Phi = I_\Phi + \mathcal{J}$, since $R_\phi \subseteq G$. Thus R_Φ can be ignored and the action it represents will still be correct.

If we are given $\mathcal{A}, \mathcal{B} \in \mathcal{GR}$, we construct $\mathcal{A} \otimes^{\Phi}_{\mathbb{K}} \mathcal{B}$ as a factor–algebra of $A \otimes^{\phi}_{\mathbb{K}} B$ by the two–sided ideal $T = T_A + T_B$.

THEOREM 4.10. Let $\mathcal{A}, \mathcal{B} \in \mathcal{GR}$, $\Phi \in \mathrm{Mor}(\mathcal{A}, \mathcal{B})$.

Let I_Φ be the $(\mathcal{A}, \mathcal{A})$–bimodule ${}_{\mathcal{A}}\langle \{x_i - \Phi(x_i) \mid 1 \le i \le n\} \rangle_{\mathcal{A}} \subset \mathcal{A} \otimes_{\mathbb{K}} \mathcal{B}$ and $f_i := \Phi(x_i)$. Suppose there exists an elimination ordering for B on $A \otimes_{\mathbb{K}} B$, such that $1 \le i \le n, 1 \le j \le m$, $\mathrm{lm}(\mathrm{lc}(f_i y_j) y_j f_i - \mathrm{lc}(y_j f_i) f_i y_j) < x_i y_j$. Then

1) $A \otimes^{\phi}_{\mathbb{K}} B$ is a G–algebra (resp. $\mathcal{A} \otimes^{\Phi}_{\mathbb{K}} \mathcal{B}$ is a GR–algebra).

2) Let $\mathcal{J} \subset \mathcal{B}$ be a left ideal. Then

$$\Phi^{-1}(\mathcal{J}) = (I_\Phi + \mathcal{J}) \cap \mathcal{A}.$$

PROOF. **1)** A quite lengthy technical computation (omitted here) ensures, that the non–degeneracy conditions on the $A \otimes^{\phi}_{\mathbb{K}} B$ indeed vanish. We give the proof for the case when both A and B are algebras of Lie type, other cases will follow analogously though more complicated coefficients will be involved. In this case the non–degeneracy condition for $x_i < x_j < y_k$ is $(y_k x_j)x_i - y_k(x_j x_i) = [x_i, [y_k, f_j]] + [[y_k, f_i], x_j] + [y_k, d_{ij}] = [x_i, [y_k, x_j]] + [[y_k, x_i], x_j] + [y_k, d_{ij}] = x_i y_k x_j - x_i x_j y_k - y_k x_j x_i + x_j y_k x_i + y_k x_i x_j - x_i y_k x_j - x_j y_k x_i + x_j x_i y_k + y_k d_{ij} - d_{ij} y_k = (-x_i x_j + x_j x_i - d_{ij}) y_k + y_k(d_{ij} + x_i x_j - x_j x_i) = 0$. The non–degeneracy condition for another triple $x_i < y_j < y_k$ holds analogously.

Note, that any elimination ordering for A is admissible on $A \otimes^{\phi}_{\mathbb{K}} B$, since any such ordering has the property $x_i \gg y_j$, r_{ij} depends only on $\{y_k\}$ and hence, $x_i y_j > \mathrm{lm}(r_{ij})$.

However, this will not always be the case for an elimination ordering for B with its property $y_j \gg x_i$, thus the condition of the theorem is essential.

2) By Lemma 4.9, $\forall g \in A$, $g - \Phi(g) \in I_\Phi$. Since Φ is a morphism and an elimination ordering for A is admissible on $\mathcal{A} \otimes^{\Phi}_{\mathbb{K}} \mathcal{B}$, we have $I_\Phi \cap B^{\mathrm{opp}} = O_\Phi \subseteq T_B^{\mathrm{opp}}$ and hence, $I_\Phi \cap \mathcal{J} \subseteq I_\Phi \cap \mathcal{B}^{\mathrm{opp}} = 0$. Then $(I_\Phi + \mathcal{J}) \cap \mathcal{B} = \mathcal{J}$.

Since there exists an admissible elimination ordering for B, the intersection with A is computable for an ideal in $A \otimes^{\phi}_{\mathbb{K}} B$. For any $q \in (I_\Phi + \mathcal{J}) \cap \mathcal{A}$ we see, that $\Phi(q) = (\Phi(q) - q) + q \in (I_\Phi + \mathcal{J}) \cap \mathcal{B} = \mathcal{J}$ and the inclusion $\Phi^{-1}(\mathcal{J}) \supset (I_\Phi + \mathcal{J}) \cap \mathcal{A}$ holds.

Conversely, let $p \in \Phi^{-1}(\mathcal{J}) \subset \mathcal{A}$. Then $p = p - \Phi(p) + \Phi(p) \in (I_\Phi + \mathcal{J}) \cap \mathcal{A}$. □

REMARK 4.11. For our purposes, we need two different orderings on $A \otimes^{\phi}_{\mathbb{K}} B$, namely an elimination orderings for A and, respectively, for B. In the proof we show, that an elimination ordering for A is always admissible and, furthermore, we imposed a condition on an elimination ordering for B.

Indeed, we have to test whether

$$\mathrm{lm}(r_{ij}) = \mathrm{lm}(y_j f_i - \frac{\mathrm{lc}(y_j f_i)}{\mathrm{lc}(f_i y_j)} f_i y_j) < x_i y_j.$$

Sometimes, it might be a problem to find such an ordering, as the two following examples show. One of possible solutions is to look for extra weights on variables. It could also happen, that for some morphism there exists no elimination ordering for B on $A \otimes^{\phi}_{\mathbb{K}} B$ (see 4.13 and **§2**, 2.11).

EXAMPLE 4.12. We proceed with the examples 4.1 and 4.6.

For $t \in \mathbb{N}, t \geq 2$ we have a morphism $S_t = \mathbb{K}\langle a, b \mid [b, a] = t \cdot a\rangle \stackrel{\psi_t}{\to} W_1$, $\psi_t(a) = x^t, \psi_t(b) = xd + t$.

On $S_t \otimes^{\psi_t}_{\mathbb{K}} W_1$, there will be the following new relations: $[x, b] = [x, xd + t] = -x, [d, a] = [d, x^t] = tx^{t-1}, [d, b] = [d, xd + t] = d$. Hence, the condition of the theorem is satisfied as soon as $x^{t-1} < ad$ and $\{x, d\} \gg a$ at the same time. As in the Example **§2**, 2.10 (where the situation with $t = 3$ is indeed described), assigning the weight t to the variable d, we come to admissible orderings of the type `(a(0,0,1,t),<)` on $S_t \otimes^{\psi_t}_{\mathbb{K}} W_1$ or, equivalently, `(a(1,t),<)` on the $W_1 \otimes^{\psi_t}_{\mathbb{K}} S_t$.

Let us compute some preimages for $t = 7$. Recall, that we are interested in preimages of left ideals $I_p = {}_{W_1}\langle x^p, xd + p\rangle$ and $J_p = {}_{W_1}\langle d^p, xd - p + 1\rangle$ for $p \in \mathbb{N}$. Consider, in addition, a family of ideals $K_{ij} = I_i \cap J_j$. Computing with PLURAL, we obtain the following.

$\psi_7^{-1}(I_3) = \langle a, b - 4\rangle$, $\psi_7^{-1}(I_8) = \langle a^2, b + 1\rangle$, $\psi_7^{-1}(I_{33}) = \langle a^5, b + 26\rangle$.

From this data we conjecture, that $\psi_t^{-1}(I_p) = \langle a^{[\frac{p}{t}]+1}, b + p - t\rangle$. If it holds, we conclude that $\psi_t(\psi_t^{-1}(I_p)) = \langle x^{p+p'}, xd + p\rangle = \langle x^p, xd + p\rangle = I_p$, where $p' > 0$.

$\psi_7^{-1}(J_3) = \langle b - 9\rangle$, $\psi_7^{-1}(J_8) = \langle b - 14\rangle$, $\psi_7^{-1}(J_{33}) = \langle b - 39\rangle$.

We may conjecture, that $\psi_t^{-1}(J_p) = \langle b + 1 - (p + t)\rangle$. If it holds, we conclude that $\psi_t(\psi_t^{-1}(J_p)) = \langle xd + 1 - p\rangle$, which is strictly contained in J_p.

Now, let us compute preimages of intersections:

$K_{3,3} = \langle x^4 d - 2x^3, x^2 d^2 + 2xd - 6, xd^4 + 6d^3\rangle$ and $\psi_7^{-1}(K_{3,3}) = \langle ab - 9a, b^2 - 13b + 36\rangle = \langle a(b-9), (b-4)(b-9)\rangle = \psi_7^{-1}(I_3) \cap \psi_7^{-1}(J_3)$. Further on, $\psi_7^{-1}(K_{8,3}) = \langle a^2(b-9), (b+1)(b-9)\rangle = \psi_7^{-1}(I_8) \cap \psi_7^{-1}(J_3)$, and $\psi_7^{-1}(K_{3,8}) = \langle a(b - 14), (b - 4)(b - 14)\rangle = \psi_7^{-1}(I_3) \cap \psi_7^{-1}(J_8)$.

EXAMPLE 4.13. Here, we continue with the examples 4.2 and 4.7.

For the map $\tau : U(\mathfrak{sl}_2) \to W_1$, $\tau(e) = x, \tau(f) = -xd^2, \tau(h) = 2xd$, we build the algebra $E' = U(\mathfrak{sl}_2) \otimes^{\tau}_{\mathbb{K}} W_1$, introducing new relations $\{[d, e] = 1, [x, f] = 2xd, [d, f] = -d^2, [x, h] = -2x, [d, h] = 2d\}$. As wee see, only two relations impose real restrictions: $fx > xd$ and $fd > d^2$, both being true if and only if $f > d$. But this is incompatible with the elimination ordering

condition for W_1 (compare with **§2**, 2.11). Hence, the conditions of the theorem are not fulfilled and there is no way to compute preimages of left ideals under τ using this approach.

4.3. One–dimensional representations of G–algebras.

Recall, that an algebra of $n \times n$ matrices over $\mathbb{K}$ is denoted by $M_n(\mathbb{K})$. For an associative $\mathbb{K}$–algebra A, a $\mathbb{K}$–algebra homomorphism $A \longrightarrow M_n(\mathbb{K})$ is called a n–**dimensional representation of** A in $\mathbb{K}$.

Assume that A is generated by the variables $\{x_1, \ldots, x_n\}$ and there is a representation $\rho : A \longrightarrow M_n(\mathbb{K}), \quad x_i \mapsto \rho(x_i)$. Then the n–tuple $(\rho(x_1), \ldots, \rho(x_n))$ is also called a representation of A.

Let B be a G–algebra, generated by $x_1, \ldots, x_n$ over a field $\mathbb{K}$. Let $\bar{a} := (a_1, \ldots, a_n) \in \mathbb{K}^n$ and $\mathfrak{m}_{\bar{a}} = {}_B\langle x_1 - a_1, \ldots, x_n - a_n\rangle \subseteq B$ be an ideal. If $\mathfrak{m}_{\bar{a}}$ is proper, it is a two–sided ideal and we have the following exact sequence:

$$0 \longrightarrow \mathfrak{m}_{\bar{a}} \longrightarrow B \longrightarrow B/\mathfrak{m}_{\bar{a}} \cong \mathbb{K} \longrightarrow 0,$$

and there is a residue map

$$B \xrightarrow{\varphi} \mathbb{K} \cong B/\mathfrak{m}_{\bar{a}}, \quad x_i \mapsto a_i.$$

In the following lemma we establish some properties of ideals $\mathfrak{m}_{\bar{a}}$.

Lemma 4.14. Let B be a G–algebra in $x_1, \ldots, x_n$ over $\mathbb{K}$. Consider a map $\varphi : B \mapsto \mathbb{K}, \quad x_i \mapsto a_i$. Then the following are equivalent:

(1) φ is a morphism with $\ker \varphi = \mathfrak{m}_{\bar{a}}$,

(2) $\varphi : B \longrightarrow \mathbb{K} = M_1(\mathbb{K})$ is a one–dimensional representation of B,

(3) $\mathfrak{m}_{\bar{a}}$ is a proper maximal ideal in B.

Proof. (1) $\Rightarrow$ (2) : At first, we compute the obstructions

$o_{ij} = (c_{ij} - 1)a_i a_j + d_{ij}(\overline{a})$. Since φ is a morphism, by Proposition 4.4, O_φ has to be zero in $\mathbb{K}$. But then a_i satisfy the relations between x_i and hence, $\varphi : B \to \mathbb{K}$ is a one–dimensional representation of B.

(2) $\Rightarrow$ (3) : Since $\forall i < j$, $(c_{ij} - 1)a_i a_j + d_{ij}(\overline{a}) = 0$, generators of $\mathfrak{m}_{\bar{a}}$ constitute a Gröbner basis, which does not contain a constant. Hence $\mathfrak{m}_{\bar{a}}$ is proper.

(3) $\Rightarrow$ (1) : Taking into account its simple structure, the fact that $\mathfrak{m}_{\bar{a}}$ is proper ideal implies that it is given in Gröbner basis. But, computing it explicitly, we see it is equal to

$$\{x_1 - a_1, \ldots, x_n - a_n\} \cup \{d_{ij}(\overline{a}) + (c_{ij} - 1)a_i a_j \mid 1 \leq i < j \leq n\}.$$

The latter part, which has to be zero, is exactly the ideal of obstructions O_φ. Hence, φ is a morphism. Moreover, each $x_i - a_i$ is mapped to zero, hence $\ker\varphi = \mathfrak{m}_{\bar{a}}$. □

REMARK 4.15. In fact, as we have seen, there is a 1–to–1 correspondence between the one–dimensional representations and proper maximal ideals of the form $\mathfrak{m}_{\bar{a}}$. On the contrary to the commutative case, these are not all of the maximal ideals of B; inspecting the first Weyl algebra, we see it may happen that an algebra has no maximal ideals of this form at all.

We show that there is an algorithm to compute all one–dimensional representations. We declare $(a_1, \ldots, a_n)$ as new variables commuting with A and build the algebra $\tilde{A} := A \otimes_{\mathbb{K}} \mathbb{K}[a_1, \ldots, a_n]$ with an elimination ordering with respect to $\{x_1, \ldots, x_n\}$. $\tilde{A}$ is obviously a G–algebra, so we compute a Gröbner basis $\tilde{I} \subset \tilde{A}$ of $I = \mathfrak{m}_{\bar{a}}$, and then we set $\hat{I} := \tilde{I} \cap \mathbb{K}[a_1, \ldots, a_n]$. In the commutative ring $\mathbb{K}[a_1, \ldots, a_n]$ we compute minimal associated primes (with the procedure MINASSPRIMES) of $\hat{I}$ (they will be maximal ideals, if $\mathbb{K}$ is algebraically closed) and read the representations from it (with the help of a procedure GETREPRESENTATION). Then we have to test the obtained set for the indecomposability with the procedure FINDINDECOMPOSABLES. We are not going to describe both auxiliary procedures due to their simplicity, but we'll show how they work on a particular example.

Algorithm 4.1 ONEDIMREPRESENTATIONS

Input: B (G–algebra in variables $x_1, \ldots, x_n$);

Output: a list of vectors in $\mathbb{K}^n$, corresponding to indecomposable one–dimensional representations;

$A = \mathbb{K}[a_1, \ldots, a_n]$; $E = B \otimes_{\mathbb{K}} A$;
$I = \{x_i - a_i, 1 \le i \le n\} \subset E$;
$J = \text{ELIMINATE}(I, B)$; ▷ $J = I \cap A$
$\text{MINASSPRIMES}(J) =: \{P_1, \ldots, P_s\}$;
$R_i = \text{GETREPRESENTATION}(P_i)$;
$\text{FINDINDECOMPOSABLES}(R) =: \{R_1, \ldots, R_t\}$;
return R;

EXAMPLE 4.16. Consider the algebra $U_q'(\mathfrak{so}_3)$ (see **§5**, 2.2 and the Example 2.2) over a field of char 0. At first, from the relations of this algebra it follows, that for any q there is a trivial one–dimensional indecomposable representation $\rho_0 = (0, 0, 0)$, which is not interesting for us.

Let $t = \dfrac{q^{1/2}}{q-1}$. Then there are four nontrivial representations of A_q:

$$\{r_{ij} = \big((-1)^i t, (-1)^j t, (-1)^{i+j} t\big) \mid 1 \leq i, j \leq 2\},$$

although $r_{11} + r_{12} + r_{21} + r_{22} = \rho_0$. Since every r_{ij} can be expressed as a direct sum of other four representations, we can exclude one of them as decomposable, in this case we throw r_{21} away. In such a way we obtain three indecomposable one–dimensional representations of $U'_q(\mathfrak{so}_3)$ in the following nice form:

$$\mathfrak{Rep}_1(U'_q(\mathfrak{so}_3)) = \{\big(\ (-1)^i t, (-1)^j t, (-1)^{i+j} t\ \big) \mid 1 \leq i < j \leq 3\}.$$

If we specify q at the n–th primitive root of unity, the algebra $U'_q(\mathfrak{so}_3)\mid_{q^n=1}$ has properties, quite different from the properties of $U'_q(\mathfrak{so}_3)$ — compare for example their centers, enlisted in **§5**, 2.2. In this case, we get again four representations of $U'_q(\mathfrak{so}_3)\mid_{q^n=1}$:

$$\{r_{ij} = \big((-1)^i t', (-1)^j t', (-1)^{i+j} t'\big) \mid 1 \leq i, j \leq 2\},$$

where $t' = \frac{1}{3}(2q^{1/2} + 1)$ for $n = 3$, $t' = \frac{1}{2}q^{1/2}(q+1)$ for $n = 4$,
$t' = \frac{1}{5}(q^{1/2}(3q+4) - (q+2))$ for $n = 5$ and so on.

We see, that as in the general case, $r_{11} + r_{12} + r_{21} + r_{22} = \rho_0$, so indeed there are only three non-trivial indecomposables. Hence, we come to the answer

$$\mathfrak{Rep}_1(A'_q) = \{\big(\ (-1)^i t', (-1)^j t', (-1)^{i+j} t'\ \big) \mid 1 \leq i < j \leq 3\},$$

comprising three indecomposable one–dimensional representations. Realizing that $t' = \dfrac{q^{1/2}}{q-1}\ \mid_{q^3=1} = \ t\ \mid_{q^3=1}$, we see that we obtain all the one–dimensional representations of A'_q for q a root of unity from the representations for generic q. Note, that these representations are non–classical in the sense that there exists no limit when $q \to 1$. Indeed, the algebra $U(\mathfrak{so}_3) = \lim_{q\to 1} U'_q(\mathfrak{so}_3)$ has only the trivial one–dimensional representation $(0,0,0)$. See [**39**] for the discussion on non–classical representations, including finite–dimensional ones.

4.4. Exact values of global dimension of G–algebras.

Proposition 4.17. Let A be a G–algebra in n variables over $\mathbb{K}$, such that A has finite–dimensional representations. Then $\operatorname{gl.dim} A = n$.

Proof. The Gel'fand–Kirillov dimension of a finite–dimensional A–module M is zero. By **§2**, 4.16 (or by **§1**, 4.14), $\operatorname{gl.dim} A \leq n$, hence $\operatorname{Ext}^{n+k}_A(M, A) = 0\ \forall k \geq 1$. Since, by **§1**, 4.14, A is a Cohen–Macaulay algebra, it follows that $j(M) = \operatorname{GKdim}(A) - \operatorname{GKdim}(M) = n$, that is $\forall 1 \leq k \leq n-1$, $\operatorname{Ext}^k_A(M, A) = 0$ and $\operatorname{Ext}^n_A(M, A) \neq 0$. Hence, $\operatorname{gl.dim} A = n$. $\square$

REMARK 4.18. It would be interesting to know, whether the existence of finite–dimensional representations is equivalent to the fact that the global dimension is exactly the number of variables of a G–algebra. We conjecture that this is true, since our experiments with PLURAL did not lead us to a counterexample yet.

EXAMPLE 4.19. Consider a n–th Weyl algebra over a field $\mathbb{K}$ $W_n(\mathbb{K}) = \mathbb{K}\langle x_1, \ldots, x_n, d_1, \ldots, d_n \mid y_i x_i = x_i y_i + 1\rangle$. One can show, that if $\mathbb{K}$ is a field of characteristic 0, there are no representations of finite dimension. The well–known result (e.g. [**59**]) states that the global dimension of $W_n(\mathbb{K})$ (in $2n$ variables!) is just n.

Let $\mathbb{F}$ be a field of characteristic p. Then, one can prove that the center of W_n over $\mathbb{F}$ is $Z(W_n) = \mathbb{F}[S]$ for the set $S = \{x_i^p, d_i^p \mid 1 \leq i \leq n\}$ (we have mentioned this result already at the beginning of this Chapter).

Since S consists of central elements, which leading monomials have no common factors, by applying a Product Criterion **§2**, 4.11 we see that S is a left Gröbner basis of a left ideal $I := {}_{W_n(\mathbb{F})}\langle S\rangle$. From the centrality of generators and the Algorithm **§2**, 3.1 it follows, that the ideal I is indeed two–sided and S is its two–sided Gröbner basis.

Consider a module $M = W_n(\mathbb{F})/I$. The special form of elements of the set S and a Gröbner basis property of S imply $\dim_{\mathbb{K}} M = p^n$. Hence, M is a finite dimensional module, so, we can build a finite dimensional representation from it and then, by the Proposition 4.17, we obtain that $\operatorname{gl.dim} W_n(\mathbb{F}) = 2n$.

Let us give a new short proof of the following known result.

COROLLARY 4.20. Let $\mathbb{K}$ be a field and $\mathfrak{g}$ be a finite dimensional Lie algebra. Then, $\operatorname{gl.dim} U(\mathfrak{g}) = \dim_{\mathbb{K}} \mathfrak{g}$.

PROOF. Let $\{x_1, \ldots, x_n\}$ be a $\mathbb{K}$–basis of $\mathfrak{g}$ and $c_k^{ij} \in \mathbb{K}$ for $1 \leq i, j, k \leq n$ are the structural constants of $\mathfrak{g}$. Thus, the relations on the universal enveloping algebra $U(\mathfrak{g})$ are $\forall j > i\ x_j x_i = x_i x_j + \sum_k c_k^{ij} x_k$. It is easy to see that the trivial representation $(0, \ldots, 0)$ always exists, hence $\operatorname{gl.dim} U(\mathfrak{g}) = n = \dim_{\mathbb{K}} \mathfrak{g}$. □

COROLLARY 4.21. Let $\mathbb{K}$ be a field and A be a G–algebra in n variables. If for every structural polynomial $d_{ij}(\bar{x})$ of A $d_{ij}(\bar{0}) = 0$ holds, then $\operatorname{gl.dim} A = n$.

PROOF. Indeed, $d_{ij}(\bar{0}) = 0$ implies that the polynomial d_{ij} has no constant term and hence, the relation $x_j x_i - c_{ij} x_i x_j - d_{ij} = 0$ holds true when substituting x_i with zero. Hence, A has the trivial representation $(0, \ldots, 0)$ over $\mathbb{K}$ and, by 4.17, $\operatorname{gl.dim} A = n$. □

Among these algebras are G–algebras with quasi–homogeneous relations (like quadratic algebras in **§1**, 7.2), quadric solvable polynomial algebras with $c_{ji} = 0, \forall i < j$ of [**56**], Witten's deformation of $U(\mathfrak{sl}_2)$ in **§1**, 7.3, diffusion algebras in **§1**, 7.4, nonstandard quantum deformations $U'_q(\mathfrak{so}_n)$ of [**36**] and many more.

As we see in the following Example, the Proposition 4.17, using the language of representations, is more general and more suitable for our purposes in comparison with the Corollary 4.21, since it allowed us to prove a more general and more intrinsic statement.

EXAMPLE 4.22. Consider the G–algebra $X_{\mathbb{K}} = \mathbb{K}\langle x, y \mid yx = xy+y^2+1\rangle$. We know already, that $1 \leq \operatorname{gl.dim} X_{\mathbb{K}} \leq 2$. There is a proper ideal $I = {}_{X_{\mathbb{K}}}\langle x, y^2+1\rangle$ with $\operatorname{Syz}(I) = {}_{X_{\mathbb{K}}}\langle(-(y^2+1), x+2y)^t\rangle$. Note, that the left module $M = X_{\mathbb{K}}/I$ is of $\mathbb{K}$–dimension 2.

We say, that a field $\mathbb{K}$ satisfies the property $(*)$, if *equation $i^2 + 1 = 0$ has solutions in* $\mathbb{K}$.

Using the Proposition 4.17, we conclude, that for any field $\mathbb{K}$, satisfying $(*)$ (it might be $\mathbb{K} = \mathbb{F}_2$ with $i = 1$ or $\mathbb{K} \supseteq \mathbb{C}$ with $i = \sqrt{-1}$), we have $\operatorname{gl.dim} X_{\mathbb{K}} = 2$, since then there are one–dimensional representations $\{(0, \pm i)\}$.

For a field $\mathbb{F}$, not satisfying $(*)$ (like $\mathbb{Q}$ or $\mathbb{R}$), there are no one–dimen–sional representations. But there are finite–dimensional representations! In particular, there is a family of representations, parametrized by $a \in \mathbb{F}^*$, given by

$$\rho_a : X_{\mathbb{F}} \to M_2(\mathbb{F}), \qquad x \mapsto \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad y \mapsto \begin{pmatrix} 0 & -a \\ 1/a & 0 \end{pmatrix}.$$

Hence, the G–algebra $X_{\mathbb{K}}$ has global dimension 2 over any field $\mathbb{K}$.

In particular, we can compute a global dimension of a difference algebra $D = \mathbb{K}\langle x, \Delta \mid \Delta \cdot x = x \cdot \Delta + \Delta + 1\rangle$, mentioned already in the Section **§1**, 6. As we see from the relations, for any field $\mathbb{K}$ there is a one–dimensional representation $x \mapsto 0, \Delta \mapsto -1$. Hence, $\operatorname{gl.dim} D = 2$.

5. Conclusion and Future Work

We hope that the nontrivial examples, computed and described in detail, help to understand both usefulness and computational complexity of treated problems. More applications like the investigation of singularities of polynomials, describing algebraic dependency of generators of the center (which we discussed in the Example 2.4) can be supported by the proposed methods.

For checking the Conjecture 2.3, the method of Perron polynomials ([**66**]), originated from the commutative case, should be revised and efficiently implemented. Alternatively, an advanced algorithm for elimination can be a big help for us.

An implementation of classification of isolated hypersurface singularities over fields of small characteristics is needed. Once available, such implementations will help to check whether the conjecture holds for universal enveloping algebras of simple Lie algebras of rank 2.

It is quite interesting to find out, in which G–algebra A the center of a factor–algebra modulo a two–sided ideal Q is equal to the center $Z(A)$ modulo the ideal $Z(A) \cap Q$ (cf. Subsection 2.6).

For a map $\Phi : \mathcal{A} \to \mathcal{B}$ of GR–algebras we obtained the following results.

• The Proposition 4.4 allows to check algorithmically whether a given map is a morphism

• For $\mathcal{A}$ being commutative
  - ◦ the preimage of a left and a two–sided ideal can be computed with the Algorithm 2.1, which generalizes the preimage algorithm for the morphism between two commutative algebras ([**35**]).

• For arbitrary GR–algebra $\mathcal{A}$
  - ◦ for a two–sided ideal $\mathcal{T} \subset \mathcal{B}$, the algorithmic computation of the preimage of $\Phi^{-1}(\mathcal{T})$ follows from the Theorem 4.5,
  - ◦ for a left ideal $\mathcal{L} \subset \mathcal{B}$, the preimage of $\mathcal{L}$ can be computed (along the lines of the Algorithm 2.1), if Φ satisfies the condition of the Theorem 4.10.

So far it is unknown whether an algorithm for the computation of the preimage of an arbitrary left ideal under an arbitrary morphism of GR–algebras exists. We conjecture, that it does not exist in general.

An algorithm for computing the global homological dimension of a given GR–algebra is of special interest. As we have shown in the Proposition 4.17, the existence of finite dimensional representations of a G–algebra A over the field $\mathbb{K}$ implies, that the global dimension of A is equal to the number of variables of A. It would be very interesting to know whether it is possible to determine the global dimension of arbitrary G–algebra in an algorithmic way.

CHAPTER 4

Implementation in the system Singular:Plural

> When he awoke again, he was walking.
> He was walking up the twisted wall-trail of Hellwell.
> As he walked, he passed the imprisoned flames.
> Again, each cried out to him as he went by:
> "Free us, masters!"
> And slowly, about the edges of the ice that was his mind, there was a thawing. Masters.
> *Plural. Not singular.*
> Masters, they had said.
> He knew then that he did not walk alone.
>
> Roger Zelazny, *Lord of Light*

1. Singular and Plural: history and development

SINGULAR is a specialized computer algebra system for polynomial computations. The kernel implements among others a variety of Gröbner basis-type algorithms (generalized Buchberger's algorithm, standard basis in local rings, in rings with mixed order, syzygy computations, algorithms to compute free resolutions of ideals, combinatorial algorithms for computations of invariants from standard bases (vector space dimensions and -bases, Hilbert function etc)) and algorithms for numerical solving of polynomial systems.

The development of SINGULAR started in 1984 by G. Pfister, K.-P. Neuendorf and H. Schönemann, in order to be able to compute standard bases in local rings or, more precisely, in the localization of the polynomial ring at the origin. The main algorithm for doing this was Mora's modification of Buchberger's algorithm to compute Gröbner bases in polynomial rings. To explain the difference between these two algorithms, let us think about a system of polynomial equations in many variables having only finitely many solutions. Using Buchberger's algorithms it is possible to compute the total number of all solutions counted with multiplicities, while Mora's algorithm allows us to compute the multiplicity of a single, specified solution. The need for this local version of Buchberger's algorithm arose from research problems in pure mathematics by G.-M. Greuel and G. Pfister, while trying to find a counterexample for complete intersections

of a theorem of K. Saito about the exactness of the Poincare complex of hypersurfaces.

None of the existing systems offered the possibility for such local computations. This was the starting point of (a forerunner of) SINGULAR. Indeed, using this implementation, a counterexample was found. This early version was mainly able to compute certain special invariants of singularities. In September 1991 the SINGULAR project became a joint venture of the Humboldt University of Berlin and the University of Kaiserslautern. Since 1994 the development of SINGULAR has been continued exclusively at the University of Kaiserslautern.

The main stimulus for enlarging the system by including new algorithms comes from research problems, arising from algebraic geometry, commutative algebra, singularity theory and non–mathematical applications.

In order to be able to compute nontrivial examples we needed an efficient implementation of the Gröbner basis-type algorithms, as well as efficient communication links between independent packages.

Many systems offer a possibility to compute Gröbner bases in polynomial rings. The general purpose Computer Algebra systems (like AXIOM, MAPLE, MATHEMATICA, MAXYMA, REDUCE) are usually not very fast and are limited in the choice of orderings, offering a little more than just the possibility to compute a Gröbner basis and, most seriously, having no method to compute efficiently in local rings. SINGULAR provides, among other things, the following features:

(1) Computations in very general rings, including polynomial rings, localizations hereof at a prime ideal and tensor products of such rings. This includes, in particular, Buchberger's and Mora's algorithm as special cases.
(2) Many ground fields for the above rings, such as the rational numbers; finite fields $\mathbb{Z}/p$, p a prime, $p \leq 2147483629$; finite fields with $q = p^n$ elements, transcendental and algebraic extensions, floating point real numbers with arbitrary precision.
(3) The combination of Gröbner basis techniques with multivariate factorization and characteristic set methods is the basis for an efficient implementation of several algorithms for primary decomposition.
(4) Many ideal– and module–theoretic operations, such as intersection, ideal quotient, elimination and saturation, and more advanced algorithms, based on free resolutions of finitely generated modules, including normalizations of rings and resolution of singularities.

Several combinatorial algorithms for computing dimensions, multiplicities, Hilbert series etc.

(5) Last, but not least, SINGULAR is designed for speed. Although it has a very general standard basis algorithm, it belongs to the fastest for computations of Gröbner bases for, say, homogeneous ideals in polynomial rings over fields of small characteristic, and it is also quite fast in computations over the rationals.

Since 2000, SINGULAR:PLURAL is a kernel extension of SINGULAR, designed to fill the present gap in considerably fast computations within the certain class of non–commutative polynomial algebras (algebras of solvable type ([**41**]) a.k.a. G-algebras ([**53**]) a.k.a. PBW-algebras ([**14**])).

Careful design of the SINGULAR:PLURAL included theoretical inspection, which led to several papers ([**48**], [**49**], [**53**]), as well as many investigations of similarity and difference of commutative and non–commutative approaches to algorithms and basic operations.

The extension SINGULAR:PLURAL allows us to handle many problems, coming from representation theory (including Lie and quantum algebras), algebraic geometry, theoretical physics and differential equations (including more linear operators like shift, difference, their q–analogs and so on, see Section **§1**, 6 and [**18**]). The major tools we use are the generalization of Buchberger's algorithm for computing Gröbner basis and Gröbner–driven algorithm for computing syzygies and free resolutions. There are only few systems, which can handle the non–commutative structures, similar to PLURAL's (see Subsection 1.1); the modern systems which can operate with non–commutative algebras are MGFUN (on MAPLE) and OPENXM, the more general systems like FELIX and MAS are no more supported.

The long lifecycle of SINGULAR, as well as its success in the mathematical community, is a consequence of the continuous application of software engineering methods. Complicated mathematical structures often need very special approach, which lies between mathematics and computer science.

SINGULAR offers a powerful C–like programming language. The growing number of libraries (nearly 60 up to now), written in this language by the SINGULAR team and various contributors, shows that both the interactive interface and the programming language were designed and implemented successfully not only from the developer's point of view but were also accepted by users. Many algorithms could be programmed using the language, supported by the fast implementation of basic functions in the kernel.

SINGULAR offers probably the biggest choice of ground fields among the contemporary specialized computer algebra systems — we have described

them above. Since the coefficients of every computation lie in the ground field, it is therefore of extreme importance to have flexible, general and fast implementation.

The designing of a generally efficient code of course goes back to the implementation of each special case, but all the operations are united under the common ideology and internally homogeneous top–level architecture. These pseudo–calls are unified, of course, not only in the case of fields, but also in general. This makes the development of kernel additions much easier and, on the other side, fully accessible for the analysis of each special sub-case.

Monomials, their presentation and operations with them comprise the heart of the "polynomial" part of SINGULAR, therefore the whole ideology and, of course, the implementation of SINGULAR and, consequently, PLURAL, are heavily influenced by them. We are using quite sophisticated structures and algorithms, partially borrowed from the literature (like several flavours of geobuckets [**77**] for the addition etc.) and partially developed by ourselves. As a result, we have one of the fastest polynomial arithmetics among the specialized computer algebra systems.

The generalized Buchberger's algorithm, one of the key points of SINGULAR, evolved with the time to the "Gröbner engine", providing several algorithms in one framework - Buchberger's, Mora's etc for the commutative case and generalized Buchberger's algorithm for the non–commutative case. It has a complicated structure, and our aim is to make it as flexible and as fast as possible.

With the development of SINGULAR:PLURAL we have enhanced and generalized our internal framework. It turned out, that its design enabled us to change only a relatively small part of the code and to gain most of the functionality (for algorithms, similar in both commutative and non–commutative settings) in such extension as SINGULAR:PLURAL. The development of such functionality from scratch would have taken enormous human resources. Thus,PLURAL inhabits the highly developed SINGULAR's framework, that gives us the opportunity to concentrate on other aspects.

As a system which comply with the GPL license, we provide both binaries and the source code of SINGULAR. Moreover, we add or even replace some parts of SINGULAR with specialized packages, for instance GNU MP for rational and floating-point arithmetics, NTL for univariate factorization and MP for communication links between SINGULAR and itself or a different Computer Algebra System. On the other side, according to the

design, our sub-packages (algorithms and/or code) can be used in other system: for example, MACAULAY2 uses FACTORY and LIBFAC sub-packages of SINGULAR.

The design and implementation of SINGULAR:PLURAL were distinguished by the Richard D. Jenks award for for Excellence in Software Engineering for Computer Algebra at the International Symposium on Symbolic and Algebraic Computation (ISSAC) 2004 in Santander (Spain). ISSAC is the yearly premier international symposium in Symbolic and Algebraic Computation and it is of big importance for the SINGULAR team, that it became the first nominee of the prize.

1.1. Existing computer algebra systems. Among the existing computer algebra systems and packages we can see two different groups: to the first one belong specialized systems, which deal with some very special classes of algebras, while the systems from the second group provide the user with an unified environment for computation in quite general families of algebras.

We give a short account of the most known systems, starting with the first group.

Gröbner basis algorithms for Ore algebras, including elimination are implemented in the MAPLE package MGFUN by F. Chyzak also for the case where the coefficients may be non–commutative with respect to the variables. Indeed, the package is restricted to several cases of operators only, but the implementation of algorithms is reliable.
(`http://algo.inria.fr/chyzak/mgfun.html`, [**18**]).

The system MACAULAY2 by D. Grayson and M. Stillman is a software system devoted to supporting research in algebraic geometry and commutative algebra, but it also includes Gröbner basis algorithms for exterior and Weyl algebras. There is a package for sophisticated computations with D–modules, providing a rich functionality for the D–module specialists.
(`http://www.math.uiuc.edu/Macaulay2`, [**20**]).

The system KAN/SM1 (distributed as a part of the system OPENXM) by N. Takayama et. al. provides Gröbner basis computations in polynomial rings, rings of differential operators, rings of difference and q-difference operators. OPENXM is said to be able to compute with algebras, where the coefficients may be non–commutative with respect to the variables.
(`http://www.math.kobe-u.ac.jp/KAN/index.html`).

The more general systems, where our system PLURAL belongs to, include also the computer algebra systems FELIX and MAS, whose development has unfortunately ceased by now.

FELIX by J. Apel and U. Klaus was released last time in August 1998 (`http://felix.hgb-leipzig.de`, [**2**]). It provides Buchberger's algorithm and its generalizations to (non–) commutative rings, in particular to free $\mathbb{K}$–algebras, polynomial rings and G–algebras. Among the implemented applications there are syzygy computations and basic ideal operations. Felix provides a complete programming language which in standard mode is interpreted but also an on-line compiler and a linker are included.

MAS by H. Kredel and M. Pesch ([**45**]) was last time released as Version 1.01 in March 1998. It contains a large library of implemented Gröbner basis algorithms for computing in (non–) commutative polynomial rings. MAS combines imperative programming facilities with algebraic specification capabilities for design and study of algebraic algorithms. It includes algorithms for real quantifier elimination and parametric real root counting. (`http://krum.rz.uni-mannheim.de/mas.html`).

For free $\mathbb{K}$–algebras, there are the systems OPAL and its obsolete predecessor GRB, which are specialized standalone systems for Gröbner bases in free and path algebras. Consequently, computations in free algebras with OPAL are much faster than with FELIX. Unfortunately, the development of OPAL has been ceased. The last working link was located at
`http://people.cs.vt.edu/~keller/opal`.

There is a MATHEMATICA package NCALGEBRA, created for computing Gröbner bases in free algebras and further manipulations with non–commutative expressions. (`http://www.math.ucsd.edu/~ncalg/`).

A system BERGMAN is a powerful tool to calculate Gröbner bases in commutative and non–commutative algebras, and in modules over them. It may also be used to calculate some invariants of algebras and modules: the Hilbert series, and (in the non-commutative case) the Poincaré–Betti series, the Anick resolution, and the Betti numbers.

BERGMAN offers the user a high level of flexibiliy. Among the alternatives for ring set-ups are: various strategies of Grb̈ner basis computation; a few different monomial orderings; and various coefficient fields, most calculations can be done both for ideals and modules. BERGMAN is written in Standard Lisp, the Lisp dialect underlying Reduce implementations, an experimentative Common Lisp version is also available. (`http://servus.math.su.se/bergman/`)

The only contemporary system for computation with free and path algebras, being developed further, is the GROBNER Package for GAP 4 by A. M. Cohen and D. A. H. Gijsbers.
(See `http://www.win.tue.nl/~amc/pub/grobner/`). With GROBNER, its

authors provide algorithms for computing Gröbner bases of sets of non–commutative polynomials with coefficients from a field implemented in GAP and with respect to the degree lexicographical ordering. Further on, some variations, such as a weighted and truncated version and a tracing facility are provided. At the moment the system features an interface, which is not convenient for the end–user, but we hope that this will be enhanced soon.

1.2. Aims of PLURAL. The development of non–commutative algorithms in SINGULAR started in 1993, when the experimental version SINGULARD 0-9-3 appeared. It contained Gröbner basis algorithms for exterior and Weyl algebras (see the technical details in the article [**63**]) — although based on the SINGULAR sources, it never became an integral part of SINGULAR.

Based on this version, we created SINGULAR:PLURAL 0-9-9 (1999-2000, [**48**]), which was able to compute left and two–sided Gröbner bases for a wide class of non–commutative algebras (G–algebras). With PLURAL 0-9-9 we computed numerous concrete important examples (see [**48**] for details), which were also used by others (like [**38**], [**17**]).

Although the name *Plural* originated from the wordplay (the opposite of Singular), we keep the name PLURAL for the modern specialized computer algebra system for non–commutative polynomial algebras. It is a part of SINGULAR, providing the Gröbner basis family of algorithms, which are to be used within the large class of non–commutative algebras important in applications. It is not only important nowadays to have a non–commutative Gröbner basis engine on its own, but also a connection to the fast and efficient "commutative" system is needed. This is motivated by many applications, like in algebraic geometry ([**20**]), D–modules, differential equations, theoretical physics. Many problems, arising within non–commutative algebras, contain sub-problems which are purely commutative (like primary decomposition, see e.g. **§1**, 7.1). That's why PLURAL is designed to be not a standalone package only, but a part of SINGULAR — in particular, it inherits the programming language, a help system, portability and many other capabilities of SINGULAR.

2. Aspects of Implementation

2.1. Singular Framework. PLURAL is implemented in the framework of SINGULAR, a system for commutative polynomial computations. This system provides memory management, an interpreter, basic types, etc. Experiences showed that Gröbner base computations tend to tremendously

long running times and consumption of huge amounts of memory even in the case of commutative polynomials.

Experience with the implementation of a commutative standard basis ([**35**]) algorithm led to highly tuned data structures, routines for polynomial operations and knowledge, how to optimize such kinds of algorithms.

Some lessons to learn from are the data representation ([**6**]) and fast additions ([**71**], [**77**]). Furthermore, the memory management of SINGULAR is optimized to handle many very small blocks of the same size (monomials) efficiently with a high locality of reference.

2.2. Basic Operations with Polynomials. In the non–commutative case, the importance of efficient basic polynomial operations is even more evident. For an efficient data representation, fast monomial comparisons and addition of polynomial routines from the commutative rings can be copied.

Multiplication of monomials is much more difficult: instead of multiplying several variables at once, we have to do it step by step for every pair of non–commuting variables.

In general, polynomial multiplication is broken down to the multiplication of monomials, where the variables with smaller indices are sorted to the left, the variables with larger indices are sorted to the right. Subexpressions of the form $x_j^a x_i^b$ are transformed to its normal form via the definition of the G–algebra (for $a = b = 1$), via a formula (if one is available, like in the case of Weyl algebras) or via a table lookup (previously computed products $x_j^s x_i^t$ are cached to speed up the computation) or, as a last resort, recursively.

Furthermore, different types of G–algebras allow different multiplication routines (i.e. more efficient than the general case) for multiplying monomials, handling commuting variables or formulas to compute $x_j^a x_i^b$. Let A be a G–algebra with the relations $\forall\ i < j\ x_j x_i = c_{ij} \cdot x_i x_j + d_{ij}(\underline{x})$, where $c_{ij} \in \mathbb{K}^*$ and $d_{ij} \in A$. We decided to implement routines for three following types:

`skew` : quasi–commutative algebras (all the $d_{ij} = 0$, cf. **§1**, 3.2),

`lie` : Lie–type algebras (all the coefficients $c_{ij} = 1$, cf. **§1**, 3.2),

`general` : all the algebras that do not belong to the previous types (for example, quantum algebras).

However, further experiments led us to the following refinement. For the multiplication of two monomials we need subroutines for multiplying a monomial with a univariate monomial from the left and from the right, and a subroutine for multiplying two univariate monomials, say, x_j^a and x_i^b ($j > i$). So, one should concentrate on the efficient implementation of all

these three levels of multiplication. In the last one we analyze the relation between the x_j and x_i and conclude what kind of multiplication will be used. If x_j and x_i (quasi–) commutes we just return $x_j^a x_i^b = c_{ij}^{ab} x_i^b x_j^a$. If there is a formula implemented, we compute the product according to it and return the corresponding result. In the most general case, we should store the values of $x_j^a x_i^b$ in the table. Of course, cashing all the values which were called from the multiplication routines, we gain higher speed of computation, but quite often the multiplication tables will become huge in size. The opposite way (not to cache any products at all) could be very slow but it uses no extra memory. So, there are several strategies of handling the tables.

We implemented at first the "cache-all" multiplication, whose principles become clear from the Figure 1. In some sense it has been the common approach to multiplication before, since "cache-all" multiplication has been implemented in MAS and Felix ([**5**]). However, it seems to us that another type of strategy should be better, namely, the "arrow" multiplication (Figure 2). We store only the values of products $x_j^a x_i, x_j x_i^b$ and $x_j^c x_i^c$, computing them on demand and use them in computations of other requested elements.

Multiplication of polynomials require many intermediate additions of large polynomials. The technique of *geobucket* addition ([**77**]) is used inside the polynomial multiplication routines. It avoids the $O(n^2)$-complexity in additions (repeated merge of sorted lists) by postponing them: the terms

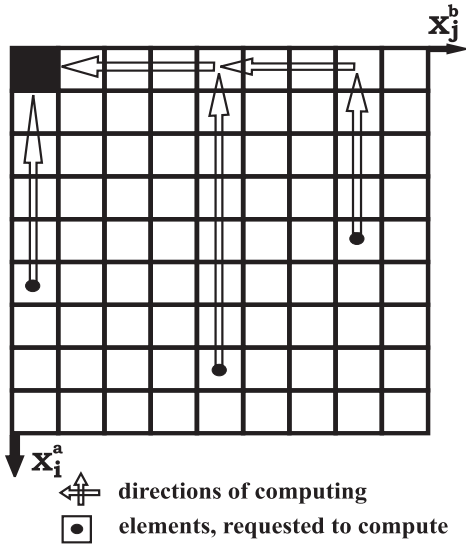

Figure 1. Cache-all multiplication

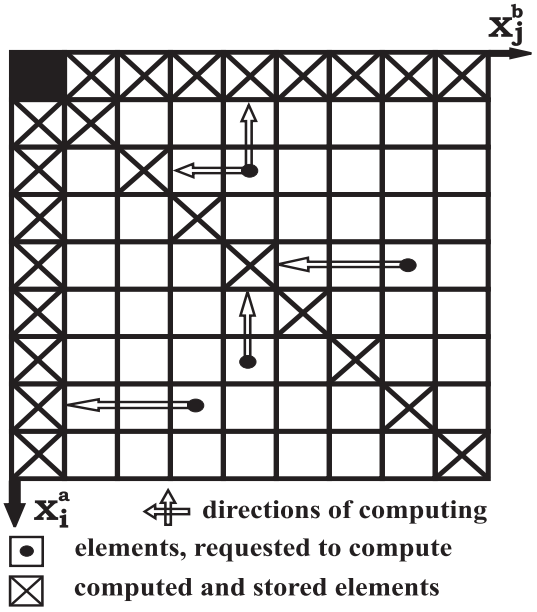

FIGURE 2. Arrow multiplication

will be accommodated in buckets of geometrically increasing length. Unlike the variant of geobuckets used in the reduction step we do not separate the leading term here. At the end of the multiplication all the buckets are added up to the result.

2.3. Connection to the commutative kernel of Singular. In SINGULAR the properties of polynomials are described by defining a base ring they belong to. In order to define such a ring one has to describe the variables, the monomial ordering and the coefficient field. Internally, the ring definition includes all basic monomial operations, corresponding to the size of the monomials, the ordering, the coefficient field, etc. We extended SINGULAR in the following way: one can define a non–commutative base ring, which borrows addition etc. from its corresponding commutative ring, but substitute all the multiplication routines.

Implementing our algorithms this way, we can use many routines from the SINGULAR kernel. Although it might have been possible also to reuse the main loop of the existing standard basis code, we decided not do this: this code can handle many special cases we are not interested in (like local standard bases) and contains too many implicit "commutative" optimizations.

The SINGULAR framework provides many means to process the results of Gröbner bases: computing dimensions, syzygies, free resolutions etc. On the other hand, the most general form of the internal call is implemented for

submodules, and is called from the elimination, intersection, modulo and other routines.

2.4. Plural libraries. Below we enlist libraries from the actual SINGULAR:PLURAL distribution accompanied with short descriptions. I would like to thank all the colleges, contributed to the libraries: Javier Lobillo and Carlos Rabelo (Granada, Spain), Oleksander Motsak, Oleksander Yena and Markus Becker (Kaiserslautern).

- `center.lib`, [**62**]. Computation of centers and centralizers in GR-algebras.
- `involut.lib`, [**7**]. Procedures for computations and operations with involutions.
- `gkdim.lib`, [**57**]. Procedures for calculating the Gel'fand–Kirillov dimension of modules.
- `ncalg.lib`, [**51**]. Definitions of important GR-algebras.
- `ncdecomp.lib`, [**50**]. Computation of the central character decomposition of a module.
- `nctools.lib`, [**54**]. General tools for non–commutative algebras.
- `qmatrix.lib`, [**58**]. Computations with quantum matrices, quantum minors and symmetric groups.

Even more libraries for PLURAL are being developed by ourself and other contributors, like `dmod.lib` for the theory of D–modules ([**17**], [**64**]), `perron.lib` for the computation of Perron polynomial ([**66**]), `delta.lib` for Delta–Gröbner bases of Castro ([**16**]), `nchomolog.lib` for homological computations including Hochschild cohomology of bimodules ([**52**]), `ncontrol.lib` for the non–commutative Control Theory ([**19**]) and more.

3. Timings, Performance and Experience

Until now there is no agreement on benchmarks for non–commutative Gröbner bases, although there were some suggestions and examples. A collection of test problems for D–modules was presented by N. Takayama in [**75**]. Many of them involve computations in rational Weyl algebras and therefore cannot be tested with PLURAL.

We use mainly the examples, originating from some concrete problems in representation theory, D–modules theory and quantum algebra.

For universal enveloping algebras of finite dimensional Lie algebras and their quantized analogues we propose the following tests to be performed. Consider a family of modules, having finite $\mathbb{K}$–dimension. For a member of such a family (say, indexed by a natural number n), presented by a set of

polynomial generators S_n from an algebra A, the following operations can be performed:

1. For a module $L_{S_n} = A/_A\langle S_n\rangle$:
   a) left Gröbner basis of S_n (**§2**, 1.3),
   b) a $\mathbb{K}$–dimension of a module L_{S_n},
   c) annihilator of a finite–dimensional module L_{S_n} (**§2**, 5.17),
   d) intersection of a finite set of left ideals $\{L_{S_i} \mid i \in \Lambda \subset \mathbb{N}\}$.
2. For a module $T_{S_n} = A/_A\langle S_n\rangle_A$:
   a) two–sided Gröbner basis of S_n (**§2**, 3.1),
   b) a $\mathbb{K}$–dimension of a module T_{S_n},
   c) annihilator of a finite–dimensional module T_{S_n},
   d) intersection of a finite set of ideals $\{T_{S_i} \mid i \in \Lambda \subset \mathbb{N}\}$.
3. Intersection of a left ideal L_{S_n} with a two–sided ideal T_{S_m}.
4. The image of L_{S_n} in the algebra $A/_A\langle T_{S_m}\rangle_A$ (**§2**, 1.2).

Various intersections of finite sets of ideals can be computed as described in **§2**, 2.13, **§2**, 5.1 or **§2**, 5.7.

EXAMPLE 3.1. We use a family of examples for computing Gröbner bases for the annihilators of certain modules over $U(\mathfrak{sl}_2)$. Consider the standard presentation of $U(\mathfrak{sl}_2)$ (see **§5**, 1.1).

For a positive integer N consider the set $F(N) = \{e^{N+1}, f^{N+1}, (h-N)\cdot(h-N+2)\cdot\ldots\cdot(h+N)\}$, and compute the left Gröbner basis of the left ideal $F_L(N) = {}_{U(\mathfrak{sl}_2)}\langle F(N)\rangle$ and the two–sided Gröbner basis of the two–sided ideal $F_T(N) = {}_{U(\mathfrak{sl}_2)}\langle F(N)\rangle_{U(\mathfrak{sl}_2)}$. We denote the computation of both bases as `AnnFD-sl2-{`N`}`. More details on this family of examples can be found in **§2**, 3.4 and **§2**, 5.18.

If we take the algebra $U(\mathfrak{g}_2)$, generated by 14 variables, then we are interested in computing the two–sided Gröbner basis of the ideal, generated by the third power of the variable, representing the shortest positive root of $\mathfrak{g}_2$. This problem is denoted by `TwoGB-g2-3`. Its detailed description can be found in **§2**, 3.5.

Another interesting family of examples originating from the D–modules theory were provided by J. M. Ucha (Sevilla).

EXAMPLE 3.2. Consider the algebra $B = \mathbb{K}\langle x, y, t, \partial_x, \partial_y, \partial_t, u, v\rangle$ subject to the relations $\partial_x \cdot x = x\partial_x + 1, \partial_y \cdot y = y\partial_y + 1, \partial_t \cdot t = t\partial_t + 1$; all other pairs of variables commute. Now let $I \subset B$ be a left ideal, generated by the set $\{tu - x^4 - y^5 - xy^4, 4x^3v\partial_t + y^4v\partial_t + \partial_x, 5y^4v\partial_t + 4xy^3v\partial_t + \partial_y\}$. One is interested in eliminating the variables u, v from I, what requires in fact a lot of computing resources. We denote this problem as `ucha2`.

Much easier problem of a similar nature appears in the algebra $C = \mathbb{K}\langle x, y, \partial_x, \partial_y, t, s\rangle$ with the relations $\partial_x \cdot x = x\partial_x + 1, \partial_y \cdot y = y\partial_y + 1, s \cdot t = ts + t$; all other pairs of variables commute. Here one wish to eliminate t, s from the left ideal $J \subset C$, generated by the set $\{s + tx^4 + ty^5 + txy^4, \partial_x + 4tx^3 + ty^4, \partial_y + 5ty^4 + 4txy^3\}$. We denote this problem as `ucha4`.

Both problems `ucha2` and `ucha4` comprise parts of a general algorithm for computing the so-called **Bernstein–Sato ideal** ([**64**], [**17**]), what is a generalization of a well–known notion of Bernstein polynomial ([**64**]). The algorithm for computing a D–module structure of a ring $\mathbb{K}[\bar{x}, f^{-1}]$ for $f \in \mathbb{K}[\bar{x}]$ is usually called AnnFs; it is presented e.g. in [**64**]. Constructively it is a sequence of eliminations, which may be quite complicated even for polynomials f of moderate total degree. It is a future task to compare the performance of Plural (and its forthcoming library `dmod.lib`) with the one of `Kan/sm1` and `Macaulay2` (both systems feature libraries for D–modules); in particular, the cusps $f = x^n - y^m$ for $m, n \in \mathbb{N}$ being coprime and the Reiffen curves $f = x^p + y^q + xy^{q-1}$, $q \geq p + 1 \geq 5$ ([**17**]) are good test examples.

There exists a SymbolicData project [**74**], created for unifying concepts and tools for Computer Algebra Benchmarks and for collecting relevant data from different areas of Computer Algebra. With the time it evolved into an unofficial standard source of examples for, in particular, commutative polynomial computations, see the review [**32**].

Together with Prof. H.-G. Gräbe (Leipzig) and O. Motsak (Kaiserslautern) we have extended SymbolicData mechanisms to handle the non–commutative algebras (G–algebras) and produced several routines for writing examples in languages of systems Singular:Plural, Felix and MAS. We encoded several interesting examples in the internal data standard of SymbolicData and used them in comparisons below. We intend to continue this work by enlarging the set of predefined algebras and ideals in these algebras.

Now, we present comparisons of performance.

Firstly, we compare the two implementations of Plural, namely versions 0-9-9 (an older experimental version) and 3-0-0 (indeed, the last beta-version of the big official release 3-0-0). We test them on HP 160 workstation with 512 MB RAM running HP-UX 10.20, time is measured in seconds.

Test	PLURAL 0-9-9	PLURAL 3-0-0	speedup
AnnFD-sl2-4	3.34	0.81	4.1
AnnFD-sl2-7	22.7	16.3	1.4
TwoGB-g2-3	24577[1]	142[2]	170
ucha4	12.9	7.2	1.8
ucha2	44.3 hours	5 hours	8.9

[1]: with the algorithm **§2**, 3.1, implemented as a PLURAL library
[2]: with the algorithm **§2**, 3.1, implemented in the kernel.

We use the described series to compare the systems FELIX, MAS and PLURAL. The computations were performed on a Linux workstation (Pentium III 866 MHz, 1GB RAM), time is measured in seconds.

Test	FELIX	MAS	PLURAL 3-0-0
AnnFD-sl2-4	0.40	0.63	0.11
AnnFD-sl2-7	1.82	10.05	2.90
AnnFD-sl2-10	6.57	×	21.44
TwoGB-g2-3	6.82[1]	×	22.60
ucha4	0.62	4.43	0.78
ucha2	×	×	2974.06

[1]: instead of 106 elements in the minimal basis, FELIX returns only 105.
×: indicates that the example failed (there were either "out of memory" or other system messages).

As we see PLURAL shows good performance compared to its old implementation and MAS, while sometimes FELIX is faster, having its merits in enveloping algebras. We excluded examples, related to quantum algebras, since such algebras are not fully supported by both FELIX and MAS.

4. Download ans Support

SINGULAR 3-0-0 was released in June of 2005. Starting from this version, PLURAL is an integral part of a SINGULAR distribution. In particular, the documentation and the help system of PLURAL together with test and example files are integrated in corresponding parts of a SINGULAR distribution.

One can freely download SINGULAR 3-0-0 from the official web–site `http://www.singular.uni-kl.de`. There are precompiled binaries for various platforms (including Unix/Linux, MacOS and Windows) and detailed documentation in several formats (including `texinfo, html, ps` and `pdf`).

At the official web–site, there is a Forum, where the requests of users are answered and commented both by members of a SINGULAR Team and various contributors. In addition, support via e–mail is proposed.

There is a semi–official web–site, devoted to PLURAL. It is located at `http://www.singular.uni-kl.de/plural` and contains more information on PLURAL, its libraries as well as applications and details of use.

The readers, having questions, requests and suggestions concerning PLURAL are welcome to contact the author via e–mail address

`levandov@mathematik.uni-kl.de`.

5. Conclusion and Future Work

We have presented prerequisites and the implementation of PLURAL as of part of SINGULAR and compared it with earlier implementations and other systems.

In its current state, PLURAL has proven, through the examples, papers, presentations and contributions, its diversity and usefulness and strong integration in the SINGULAR framework. Moreover, the overall performance of PLURAL is quite reasonable and we are working on further improving it: better treatment of special cases will give a lot of possibilities for this.

Directions of further work include an optimized polynomial multiplication, more distinct separation into special cases and better possibilities to pre– and post-process the computed non–commutative Gröbner bases: all the algorithms should recognize the non–commutative base ring and switch to a corresponding implementation.

It turned out, that recognizing the structure of a complicated algebra (say, as a tensor product of several algebras over the ground field) together with corresponding product of orderings can bring much more than one can think at first. However, correct design of such *contexts* (or building blocks of such tensor products) must be done in order to reveal all the potential strengths and dangers. It is also of big importance for the pure commutative case, as Prof. G.-M. Greuel recently noted.

CHAPTER 5

Small Atlas of Important Algebras

Let $\mathbb{K}$ be a field. The structural coefficients of algebras are written in the field of characteristic 0. If we consider an algebra over a field of positive characteristic, we replace the coefficients by their images in case they are given over the integers. We enlist only the non–commutative relations and omitt commutative ones. The central polynomials are written in their reduced form and with fraction–free coefficients.

1. Universal Enveloping Algebras of Lie Algebras

1.1. $U(\mathfrak{sl}_2)$**.**

Reference(s): [**23**];
Variables: $\{e, f, h\}$;
Relations: $[e, f] = h,\ [h, e] = 2e,\ [h, f] = -2f$;
Casimir element: $C_2 = 4ef + h^2 - 2h$;
$\operatorname{char}\mathbb{K} = 0 : Z(U(\mathfrak{sl}_2)) = \mathbb{K}[C_2]$;
$\operatorname{char}\mathbb{K} = p : Z(U(\mathfrak{sl}_2)) = \mathbb{K}[C_2, e^p, f^p, h^p - h]$;
Realization as Ore algebra: see **§1**, 3.14;
Plural (`ncalg.lib`) function: `makeUsl2(p)` or `makeUsl(2,p)`.

1.2. $U(\mathfrak{so}_3)$**.**

Reference(s): [**23**], [**27**];
Variables: $\{x, y, z\}$;
Relations: $[x, y] = z,\ [z, x] = y,\ [y, z] = x$;
Central element: $C_2 = x^2 + y^2 + z^2$;
$\operatorname{char}\mathbb{K} = 0 : Z(U(\mathfrak{so}_3)) = \mathbb{K}[C_2]$;
$\operatorname{char}\mathbb{K} = p : Z(U(\mathfrak{so}_3)) = \mathbb{K}[C_2, x^p + x, y^p + y, z^p + z]$;
Plural (`ncalg.lib`) function: `makeUso3(p)`.

1.3. $U(\mathfrak{sl}_3)$**.**

Reference(s): [**37**];
Variables: $\{x_\alpha, x_\beta, x_\gamma, y_\alpha, y_\beta, y_\gamma, h_\alpha, h_\beta\}$;
Relations: $[x_\alpha, x_\beta] = x_\gamma, [x_\alpha, y_\alpha] = h_\alpha, [x_\alpha, y_\gamma] = -y_\beta,$
$[x_\alpha, h_\alpha] = -2x_\alpha, [x_\alpha, h_\beta] = x_\alpha, [x_\beta, y_\beta] = h_\beta,$
$[x_\beta, y_\gamma] = y_\alpha, [x_\beta, h_\alpha] = x_\beta, [x_\beta, h_\beta] = -2x_\beta, [x_\gamma, y_\alpha] = -x_\beta,$

$[x_\gamma, y_\beta] = x_\alpha, [x_\gamma, y_\gamma] = h_\alpha + h_\beta, [x_\gamma, h_\alpha] = -x_\gamma, [x_\gamma, h_\beta] = -x_\gamma,$
$[y_\alpha, y_\beta] = -y_\gamma, [y_\alpha, h_\alpha] = 2y_\alpha, [y_\alpha, h_\beta] = -y_\alpha, [y_\beta, h_\alpha] = -y_\beta,$
$[y_\beta, h_\beta] = 2y_\beta, [y_\gamma, h_\alpha] = y_\gamma, [y_\gamma, h_\beta] = y_\gamma;$
Casimir elements:
$C_2 = 3x_\alpha y_\alpha + 3x_\beta y_\beta + 3x_\gamma y_\gamma + h_\alpha^2 + h_\alpha h_\beta + h_\beta^2 - 3h_\alpha - 3h_\beta,$
$C_3 = 27x_\gamma y_\alpha y_\beta + 27x_\alpha x_\beta y_\gamma + 9x_\alpha y_\alpha h_\alpha - 18x_\beta y_\beta h_\alpha + 9x_\gamma y_\gamma h_\alpha + 2h_\alpha^3 +$
$18x_\alpha y_\alpha h_\beta - 9x_\beta y_\beta h_\beta - 9x_\gamma y_\gamma h_\beta + 3h_\alpha^2 h_\beta - 3h_\alpha h_\beta^2 - 2h_\beta^3 - 36x_\alpha y_\alpha +$
$18x_\beta y_\beta - 9x_\gamma y_\gamma - 12h_\alpha^2 - 3h_\alpha h_\beta + 6h_\beta^2 + 18h_\alpha;$
$\operatorname{char}\mathbb{K} = 0 : Z(U(\mathfrak{sl}_3)) = \mathbb{K}[C_2, C_3];$
$\operatorname{char}\mathbb{K} = p : Z(U(\mathfrak{sl}_3)) = \mathbb{K}[C_2, C_3, x_{\alpha,\beta,\gamma}^p, y_{\alpha,\beta,\gamma}^p, h_\alpha^p - h_\alpha, h_\beta^p - h_\beta];$
PLURAL (`ncalg.lib`) function: `makeUsl(3,p)`.

1.4. $U(\mathfrak{g}_2)$**.**

Reference(s): [**27**];
Variables: $\{x_1, \ldots, x_6, y_1, \ldots, y_6, h_\alpha, h_\beta\};$
Relations: $[x_1, x_2] = x_3, [x_1, x_3] = 2x_4, [x_1, x_4] = -3x_5,$
$[x_1, y_1] = h_\alpha, [x_1, y_3] = -3y_2, [x_1, y_4] = -2y_3,$
$[x_1, y_5] = y_4, [x_1, h_\alpha] = -2x_1, [x_1, h_\beta] = x_1,$
$[x_2, x_5] = -x_6, [x_2, y_2] = h_\beta, [x_2, y_3] = y_1, [x_2, y_6] = y_5,$
$[x_2, h_\alpha] = 3x_2, [x_2, h_\beta] = -2x_2, [x_3, x_4] = -3x_6, [x_3, y_1] = -3x_2,$
$[x_3, y_2] = x_1, [x_3, y_3] = h_\alpha + 3h_\beta, [x_3, y_4] = 2y_1, [x_3, y_6] = y_4,$
$[x_3, h_\alpha] = x_3, [x_3, h_\beta] = -x_3, [x_4, y_1] = -2x_3, [x_4, y_3] = 2x_1,$
$[x_4, y_4] = 2h_\alpha + 3h_\beta, [x_4, y_5] = -y_1, [x_4, y_6] = -y_3, [x_4, h_\alpha] = -x_4,$
$[x_5, y_1] = x_4, [x_5, y_4] = -x_1, [x_5, y_5] = h_\alpha + h_\beta, [x_5, y_6] = -y_2,$
$[x_5, h_\alpha] = -3x_5, [x_5, h_\beta] = x_5, [x_6, y_2] = x_5, [x_6, y_3] = x_4,$
$[x_6, y_4] = -x_3, [x_6, y_5] = -x_2, [x_6, y_6] = h_\alpha + 2h_\beta, [x_6, h_\beta] = -x_6,$
$[y_1, y_2] = -y_3, [y_1, y_3] = -2y_4, [y_1, y_4] = 3y_5, [y_1, h_\alpha] = 2y_1,$
$[y_1, h_\beta] = -y_1, [y_2, y_5] = y_6, [y_2, h_\alpha] = -3y_2, [y_2, h_\beta] = 2y_2,$
$[y_3, y_4] = 3y_6, [y_3, h_\alpha] = -y_3, [y_3, h_\beta] = y_3, [y_4, h_\alpha] = y_4,$
$[y_5, h_\alpha] = 3y_5, [y_5, h_\beta] = -y_5, [y_6, h_\beta] = y_6;$
Casimir elements:
$C_2 = x_1y_1 + 3x_2y_2 + x_3y_3 + x_4y_4 + 3x_5y_5 + 3x_6y_6 + h_\alpha^2 + 3h_\alpha h_\beta +$
$3h_\beta^2 - 5h_\alpha - 9h_\beta;$
$C_6 = 4x_1x_2^2y_1y_2^2 + \ldots$ (totally 754 monomials);
$\operatorname{char}\mathbb{K} = 0 : Z(U(\mathfrak{g}_2)) = \mathbb{K}[C_2, C_6];$
$\operatorname{char}\mathbb{K} = p : Z(U(\mathfrak{g}_2)) = \mathbb{K}[C_2, C_6, \{x_i^p\}, \{y_i^p\}, h_\alpha^p - h_\alpha, h_\beta^p - h_\beta];$
PLURAL (`ncalg.lib`) function: `makeUg2(p)`.

2. Quantum Algebras

2.1. $U_q(\mathfrak{sl}_2)$**.**

Reference(s): [**14**], [**43**];
Variables: $\{E, F, K_e, K_f\}$;
Relations: $FE = EF - \frac{q}{q^2-1}K_e + \frac{q}{q^2-1}K_f$,
$K_eE = q^2EK_e, K_fF = q^2FK_f$, $K_fE = \frac{1}{q^2}EK_f, K_eF = \frac{1}{q^2}FK_e$,
$K_fK_e = K_eK_f$;
$U_q(\mathfrak{sl}_2)$ is a factor-algebra modulo the two–sided ideal $\langle K_eK_f - 1\rangle$;
Casimir element: $C_2 = (q^2-1)^2EF + qK_e + q^3K_f$;
q generic: $Z(U_q(\mathfrak{sl}_2)) = \mathbb{K}[C_2]$;
$\exists n, q^n = 1 : Z(U_q(\mathfrak{sl}_2)) = \mathbb{K}[C_2, E^n, F^n, K_e^n, K_f^n]$;
Plural (`ncalg.lib`) function: `makeQsl2(n)`.

2.2. $U_q'(\mathfrak{so}_3)$**.**

Reference(s): [**36**], [**38**];
Variables: $\{I_1, I_2, I_3\}$;
Relations: $I_2I_1 = qI_1I_2 - q^{1/2}I_3, I_3I_1 = q^{-1}I_1I_3 + q^{-1/2}I_2$,
$I_3I_2 = qI_2I_3 - q^{1/2}I_1$;
Casimir elements: $C_3 = -q^{1/2}(q^2-1)I_1I_2I_3 + q^2I_1^2 + I_2^2 + q^2I_3^2$;
q generic: $Z(U_q'(\mathfrak{so}_3)) = \mathbb{K}[C_3]$;
$\exists n, q^n = 1 : Z(U_q'(\mathfrak{so}_3)) = \mathbb{K}[C_3, C_n^{(1)}, C_n^{(2)}, C_n^{(3)}]$, where $C_n^{(k)} = 2T_p(I_k(q-q^{-1})/2)$, $k = 1, 2, 3$, where $T_p(x)$ is a Chebyshev polynomial of the first kind (see **§3**, 2.2 for detailed description);
Plural (`ncalg.lib`) function: `makeQso3(n)`.

6 Singular:Plural Manual

6.1 Getting started with PLURAL

What is and what does PLURAL?

PLURAL is a kernel extension of SINGULAR, providing many algorithms for computations within certain noncommutative algebras (see see Section 6.4.33 [G-algebras], page 195 and Section 6.4 [Mathematical background (plural)], page 195 for detailed information on algebras and algorithms).

PLURAL is compatible with SINGULAR, since it uses the same data structures, sometimes interpreting them in a different way and/or modifying them for its own purposes. In spite of such a difference, one can always transfer objects from commutative rings of SINGULAR to noncommutative rings PLURAL and back.

With PLURAL, one can set up a noncommutative G -algebra with a Poincaré-Birkhoff-Witt (PBW) basis, say, A (see Section 6.4.33 [G-algebras], page 195 for step-by-step building instructions and also Section 6.5 [PLURAL libraries], page 201 for procedures for setting many important algebras easily).

Functionalities of PLURAL (enlisted in Section 6.3 [Functions (plural)], page 170) are accessible as soon as the basering becomes noncommutative (see Section 6.3.21 [ncalgebra], page 182).

One can perform various computations with polynomials and ideals in A and with vectors and submodules of a free module A^n.

One can work also within factor-algebras of G -algebras (see Section 6.2.5 [qring (plural)], page 165 type) by two-sided ideals (see Section 6.3.31 [twostd], page 193).

What PLURAL does not:

PLURAL does not perform computations in free algebra or in its general factor algebras.

One can only work with G -algebras and with their factor-algebras by two-sided ideals.

PLURAL requires a monomial ordering but it does not work with local and mixed orderings.

Right now, one can use only global orderings in PLURAL (see Section B.2.2 [General definitions for orderings], page 255).

This will be enhaced in the future by providing the possibility of computations in a tensor product of a noncommutative algebra (with a global ordering)
with a commutative algebra (with any ordering).

PLURAL does not handle noncommutative parameters.

Defining parameters, one **cannot** impose noncommutative relations on them. Moreover, it is impossible to introduce
parameters which do not commute with variables.

PLURAL **conventions**

***-multiplication (plural)**

in the noncommutative case, the correct multiplication of `y` by `x` must be written as `y*x`.
Both expressions `yx` and `xy` are equal, since they are interpreted as commutative expressions. See example in Section 6.2.4.2 [poly expressions (plural)], page 163.
Note, that PLURAL output consists only of monomials, hence the signs `*` are omitted.

`ideal` **(plural)**

Under an `ideal` PLURAL understands a list of generators of a **left** ideal. For more information see Section 6.2.1 [ideal (plural)], page 152.
For a **two-sided ideal** `T`, use command Section 6.3.31 [twostd], page 193 in order to compute the two-sided Groebner basis of `T`.

`module` **(plural)**

Under a `module` PLURAL understands **either** a fininitely generated **left** submodule of a free module (of finite rank)
or a factor module of a free module (of finite rank) by its left submodule (see Section 6.2.3 [module (plural)], page 159 for details).

`qring` **(plural)**

In PLURAL it is only possible to build factor-algebras modulo **two-sided** ideals (see Section 6.2.5 [qring (plural)], page 165).

6.2 Data types (plural)

This chapter explains all data types of PLURAL in alphabetical order. For every type, there is a description of the declaration syntax
as well as information about how to build expressions of certain types.

The term "expression list" in PLURAL refers to any comma separated list of expressions.

For the general syntax of a declaration see Section 2.5.1 [General command syntax], page 44.

6.2.1 ideal (plural)

For PLURAL ideals are **left** ideals.
Ideals are represented as lists of polynomials which are interpreted as left generators

of the ideal.
For the operations with two-sided ideals see Section 6.3.31 [twostd], page 193.

Like polynomials, ideals can only be defined or accessed with respect to a basering.

Note: `size` counts only the non-zero generators of an ideal whereas `ncols` counts all generators.

6.2.1.1 ideal declarations (plural)

Syntax: `ideal` name `=` list_of_poly_and_ideal_expressions `;`
`ideal` name `=` ideal_expression `;`

Purpose: defines a left ideal.

Default: 0

Example:

```
ring r=0,(x,y,z),dp;
ncalgebra(-1,0); // an anticommutative algebra
poly s1 = x2;
poly s2 = y3;
poly s3 = z;
ideal i =  s1, s2-s1, 0,s2*s3, s3^4;
i;
↦ i[1]=x2
↦ i[2]=y3-x2
↦ i[3]=0
↦ i[4]=y3z
↦ i[5]=z4
size(i);
↦ 4
ncols(i);
↦ 5
```

6.2.1.2 ideal expressions (plural)

An ideal expression is:

1. an identifier of type ideal
2. a function returning an ideal
3. a combination of ideal expressions by the arithmetic operations `+` or `*`
4. a power of an ideal expression (operator `^` or `**`)
   Note that the computation of the product `i*i` involves all products of generators of `i` while `i^2` involves only the different ones, and is therefore faster.
5. a type cast to ideal

Example:

```
ring r=0,(x,y,z),dp;
ncalgebra(-1,0); // an anticommutative algebra
```

```
ideal m = maxideal(1);
m;
↦ m[1]=x
↦ m[2]=y
↦ m[3]=z
poly f = x2;
poly g = y3;
ideal i = x*y*z , f-g, g*(x-y) + f^4 ,0, 2x-z2y;
ideal M = i + maxideal(10);
i = M*M;
ncols(i);
↦ 598
i = M^2;
ncols(i);
↦ 690
i[ncols(i)];
↦ x20
vector v = [x,y-z,x2,y-x,x2yz2-y];
ideal j = ideal(v);
j;
↦ j[1]=x
↦ j[2]=y-z
↦ j[3]=x2
↦ j[4]=-x+y
↦ j[5]=x2yz2-y
```

6.2.1.3 ideal operations (plural)

`+` addition (concatenation of the generators and simplification)

`*` multiplication (with ideal, poly, vector, module; in case of multiplication with ideal, the result will be simplified)

`^` exponentiation (by a non-negative integer)

ideal_expression `[` intvec_expression `]`
: are polynomial generators of the ideal, index 1 gives the first generator.

Note: For simplification of an ideal, see also Section 4.1.118 [simplify], page 232.

Example:

```
ring r=0,(x,y,z),dp;
matrix D[3][3];
D[1,2]=-z;  D[1,3]=y;  D[2,3]=x;
ncalgebra(1,D); // this algebra is U(so_3)
ideal I = 0,x,0,1;
I;
↦ I[1]=0
↦ I[2]=x
↦ I[3]=0
```

```
↦ I[4]=1
I + 0;    // simplification
↦ _[1]=1
I*x;
↦ _[1]=0
↦ _[2]=x2
↦ _[3]=0
↦ _[4]=x
ideal J = I,0,x,x-z;
I * J;   //  multiplication with simplification
↦ _[1]=1
vector V = [x,y,z];
print(I*V);
↦ 0,x2,0,x,
↦ 0,xy,0,y,
↦ 0,xz,0,z
ideal m = maxideal(1);
m^2;
↦ _[1]=x2
↦ _[2]=xy
↦ _[3]=xz
↦ _[4]=y2
↦ _[5]=yz
↦ _[6]=z2
ideal II = I[2..4];
II;
↦ II[1]=x
↦ II[2]=0
↦ II[3]=1
```

6.2.1.4 ideal related functions (plural)

`eliminate`
: elimination of variables (see Section 6.3.10 [eliminate (plural)], page 172)

`intersect`
: ideal intersection (see Section 6.3.14 [intersect (plural)], page 176)

`kbase`
: vector space basis of basering modulo the leading ideal (see Section 6.3.15 [kbase (plural)], page 176)

`lead`
: leading terms of a set of generators (see Section 4.1.63 [lead], page 186)

`lift`
: lift-matrix (see Section 6.3.16 [lift (plural)], page 177)

`liftstd`
: left Groebner basis and transformation matrix computation (see Section 6.3.17 [liftstd (plural)], page 178)

`maxideal`
: generators of a power of the maximal ideal at 0 (see Section 4.1.72 [maxideal], page 193)

`modulo` represents $(h1 + h2)/h1 \cong h2/(h1 \cap h2)$ (see Section 6.3.19 [modulo (plural)], page 180)

`mres` minimal free resolution of an ideal and a minimal set of generators of the given ideal (see Section 6.3.20 [mres (plural)], page 180)

`ncols` number of columns (see Section 4.1.85 [ncols], page 203)

`nres` computes a free resolution of an ideal resp. module M which is minimized from the second free module on (see Section 6.3.22 [nres (plural)], page 184)

`oppose` creates an opposite ideal of a given ideal from the given ring into a basering (see Section 6.3.23 [oppose], page 185)

`preimage` preimage under a ring map (see Section 6.3.25 [preimage (plural)], page 187)

`quotient` ideal quotient (see Section 6.3.26 [quotient (plural)], page 188)

`reduce` left normal form with respect to a left Groebner basis (see Section 6.3.27 [reduce (plural)], page 189)

`simplify` simplify a set of polynomials (see Section 4.1.118 [simplify], page 232)

`size` number of non-zero generators (see Section 4.1.119 [size], page 233)

`std` left Groebner basis computation (see Section 6.3.28 [std (plural)], page 191)

`subst` substitute a ring variable (see Section 6.3.29 [subst (plural)], page 192)

`syz` computation of the first syzygy module (see Section 6.3.30 [syz (plural)], page 192)

`twostd` two-sided Groebner basis computation (see Section 6.3.31 [twostd], page 193)

`vdim` vector space dimension of basering modulo the leading ideal (see Section 6.3.32 [vdim (plural)], page 194)

6.2.2 map (plural)

Maps are ring maps from a preimage ring into the basering.

Note:

- the target of a map is **ALWAYS** the actual basering
- the preimage ring has to be stored "by its name", that means, maps can only be used in such contexts, where the name of the preimage ring can be resolved (this has to be considered in subprocedures). See also Section 5.4 [Identifier resolution], page 277, Section 2.7.2 [Names in procedures], page 56.

Maps between rings with different coefficient fields are possible and listed below.

Canonically realized are

- $Q \to Q(a, \ldots)$ (Q : the rational numbers)

- $Q \to R$ (R : the real numbers)
- $Q \to C$ (C : the complex numbers)
- $Z/p \to (Z/p)(a,\ldots)$ (Z : the integers)
- $Z/p \to GF(p^n)$ (GF : the Galois field)
- $Z/p \to R$
- $R \to C$

Possible are furthermore

- $Z/p \to Q, \quad [i]_p \mapsto i \in [-p/2,\, p/2] \subseteq Z$
- $Z/p \to Z/p', \quad [i]_p \mapsto i \in [-p/2,\, p/2] \subseteq Z,\ i \mapsto [i]_{p'} \in Z/p'$
- $C \to R, \quad$ by taking the real part

Finally, in PLURAL we allow the mapping from rings with coefficient field Q to rings whose ground fields have finite characteristic:

- $Q \to Z/p$
- $Q \to (Z/p)(a,\ldots)$

In these cases the denominator and the numerator of a number are mapped separately by the usual map from Z to Z/p, and the image of the number is built again afterwards by division. It is thus not allowed to map numbers whose denominator is divisible by the characteristic of the target ground field, or objects containing such numbers. We, therefore, strongly recommend using such maps only to map objects with integer coefficients.

6.2.2.1 map declarations (plural)

Syntax: `map` name `=` preimage_ring_name `,` ideal_expression `;`
`map` name `=` preimage_ring_name `,` list_of_poly_and_ideal_expressions `;`
`map` name `=` map_expression `;`

Purpose: defines a ring map from `preimage_ring` to basering.
Maps the variables of the `preimage ring` to the generators of the ideal.
If the ideal contains less elements than the number of variables in the `preimage_ring`, the remaining variables are mapped to 0.
If the ideal contains more elements, extra elements are ignored.
The image ring is always the current basering. For the mapping of coefficients from different fields see Section 6.2.2 [map (plural)], page 156.

Default: none

Note: There are standard mappings for maps which are close to the identity map: `fetch (plural)` and `imap (plural)`.

The name of a map serves as the function which maps objects from the preimage_ring into the basering. These objects must be defined by names (no evaluation in the preimage ring is possible).

Example:

```
// an easy example
ring r1 = 0,(a,b),dp; // a commutative ring
poly P = a^2+ab+b^3;
ring r2 = 0,(x,y),dp;
ncalgebra(1,-1); // a Weyl algebra
map M = r1, x^2, -y^3;
// note: M is a map and not a morphism
M(P);
↦ -y9-x2y3+x4
// now, a more interesting example
LIB "ncalg.lib";
def Usl2 = makeUsl2();
// this algebra is U(sl_2), generated by  e,f,h
setring Usl2;
poly  C  = 4*e*f+h^2-2*h; // the central el-t of Usl2
poly  D  = e^3*f-h^4;     // some polynomial
ring W1  = 0,(D,X),dp;
ncalgebra(1,-1);
// this algebra is the opposite Weyl algebra
option(redSB);
option(redTail);
map F = Usl2, -X, D*D*X, 2*D*X;
F(C); // 0, because C is in the kernel of F
↦ 0
F(D);
↦ -16D4X4+96D3X3-D2X4-112D2X2+6DX3+16DX-6X2
```

See Section 6.3.12 [fetch (plural)], page 174; Section 6.2.1.2 [ideal expressions (plural)], page 153; Section 6.3.13 [imap (plural)], page 174; Section 6.2.2 [map (plural)], page 156; Section 6.2.7 [ring (plural)], page 167.

6.2.2.2 map expressions (plural)

A map expression is:

1. an identifier of type map
2. a function returning map
3. map expressions combined by composition using parentheses (`(`, `)`)

6.2.2.3 map (plural) operations

`( )` composition of maps. If, for example, `f` and `g` are maps, then `f(g)` is a map expression giving the composition $f \circ g$ of `f` and `g`, provided the target ring of `g` is the basering of `f`.

map_expression `[` int_expressions `]`
: is a map entry (the image of the corresponding variable)

Example:

```
LIB "ncalg.lib";
def Usl2 = makeUsl2(); // this algebra is U(sl_2)
setring Usl2;
map F = Usl2, f, e, -h; // endomorphism of U(sl_2)
map G = F(F);
poly p = (f+e*h)^2 + 3*h-e;
p;
↦ e2h2+2e2h+2efh-2ef+f2-h2-e+3h
F(p);
↦ f2h2-2efh-2f2h+e2-2ef+h2-f-h
G(p);
↦ e2h2+2e2h+2efh-2ef+f2-h2-e+3h
(G(p) == p); // G is the identity on p
↦ 1
```

6.2.2.4 map related functions (plural)

`fetch (plural)`
: the identity map between rings and qrings (see Section 6.3.12 [fetch (plural)], page 174)

`imap (plural)`
: a convenient map procedure for inclusions and projections of rings (see Section 6.3.13 [imap (plural)], page 174)

`preimage (plural)`
: preimage under a ring map (see Section 6.3.25 [preimage (plural)], page 187)

`subst` substitute a ring variable (see Section 6.3.29 [subst (plural)], page 192)

6.2.3 module (plural)

Modules are **left** submodules of a free module over the basering with basis `gen(1)`, `gen(2)`, ..., `gen(n)` for some natural number `n`.
They are represented by lists of vectors, which generate the left submodule. Like vectors, they can only be defined or accessed with respect to a basering.

If M is a left submodule of R^n (where R is the basering) generated by vectors $v_1, \ldots, v_k$, then these generators may be considered as
the generators of relations of R^n/M between the canonical generators `gen(1)`,...,`gen(n)`. Hence, any finitely generated R -module can be represented in PLURAL by its module of relations.
The assignments `module M=v1,...,vk; matrix A=M;` create the presentation matrix of size $n \times k$ for R^n/M, i.e. the columns of A are the vectors $v_1, \ldots, v_k$ which generate M.

6.2.3.1 module declarations (plural)

Syntax: `module` name `=` list_of_vector_expressions (which are interpreted as left generators of the module) `;`
`module` name `=` module_expression `;`

Purpose: defines a left module.

Default: [0]

Example:

```
ring r=0,(x,y,z),(c,dp);
matrix D[3][3];
D[1,2]=-z;  D[1,3]=y;  D[2,3]=x;
ncalgebra(1,D); // this algebra is U(so_3)
vector s1 = [x2,y3,z];
vector s2 = [xy,1,0];
vector s3 = [0,x2-y2,z];
poly   f  = -x*y;
module m = s1, s2-s1,f*(s3-s1);
m;
↦ m[1]=[x2,y3,z]
↦ m[2]=[-x2+xy,-y3+1,-z]
↦ m[3]=[x3y-2x2z-xy,xy4-x3y+xy3+2x2z+xy]
// show m in matrix format (columns generate m)
print(m);
↦ x2,-x2+xy,x3y-2x2z-xy,
↦ y3,-y3+1, xy4-x3y+xy3+2x2z+xy,
↦ z, -z,    0
```

6.2.3.2 module expressions (plural)

A module expression is:

1. an identifier of type module
2. a function returning module
3. module expressions combined by the arithmetic operation `+`
4. multiplication of a module expression with an ideal or a poly expression: `*`
5. a type cast to module

6.2.3.3 module operations (plural)

`+` addition (concatenation of the generators and simplification)

`*` right or left multiplication with ideal or poly (but not ‘module‘ * ‘module‘!)

module_expression `[` int_expression `,` int_expression `]`
is a module entry, where the first index indicates the row and the second the column

module_expressions [int_expression]
: is a vector, where the index indicates the column (generator)

Example:

```
ring A=0,(x,y,z),Dp;
matrix D[3][3];
D[1,2]=-z;  D[1,3]=y;  D[2,3]=x;  // this algebra is U(so_3)
ncalgebra(1,D);
module M = [x,y],[0,0,x*z];
module N = (x+y-z)*M - M*(x+y-z);
print(-N);
↦ y+z,0,
↦ x-z,0,
↦ 0,  x2+xy+yz+z2
```

6.2.3.4 module related functions (plural)

`eliminate`
: elimination of variables (see Section 6.3.10 [eliminate (plural)], page 172)

`freemodule`
: the free module of given rank (see Section 4.1.38 [freemodule], page 169)

`intersect`
: module intersection (see Section 6.3.14 [intersect (plural)], page 176)

`kbase` vector space basis of free module over the basering modulo the module of leading terms (see Section 6.3.15 [kbase (plural)], page 176)

`lead` initial module (see Section 4.1.63 [lead], page 186)

`lift` lift-matrix (see Section 6.3.16 [lift (plural)], page 177)

`liftstd` left Groebner basis and transformation matrix computation (see Section 6.3.17 [liftstd (plural)], page 178)

`modulo` represents $(h1 + h2)/h1 \cong h2/(h1 \cap h2)$ (see Section 6.3.19 [modulo (plural)], page 180)

`mres` minimal free resolution of a module and a minimal set of generators of the given ideal module (see Section 6.3.20 [mres (plural)], page 180)

`ncols` number of columns (see Section 4.1.85 [ncols], page 203)

`nres` computes a free resolution of an ideal resp. module M which is minimized from the second free module on (see Section 6.3.22 [nres (plural)], page 184)

`nrows` number of rows (see Section 4.1.88 [nrows], page 205)

`oppose` creates an opposite module of a given module from the given ring into a basering (see Section 6.3.23 [oppose], page 185)

`print` nice print format (see Section 4.1.99 [print], page 213)

`prune` minimize the embedding into a free module (see Section 4.1.101 [prune], page 217)

`quotient` module quotient (see Section 6.3.26 [quotient (plural)], page 188)

`reduce` left normal form with respect to a left Groebner basis (see Section 6.3.27 [reduce (plural)], page 189)

`simplify` simplify a set of vectors (see Section 4.1.118 [simplify], page 232)

`size` number of non-zero generators (see Section 4.1.119 [size], page 233)

`std` left Groebner basis computation (see Section 6.3.28 [std (plural)], page 191)

`subst` substitute a ring variable (see Section 6.3.29 [subst (plural)], page 192)

`syz` computation of the first syzygy module (see Section 6.3.30 [syz (plural)], page 192)

`vdim` vector space dimension of free module over the basering modulo module of leading terms (see Section 6.3.32 [vdim (plural)], page 194)

6.2.4 poly (plural)

Polynomials are the basic data for all main algorithms in Plural. They consist of finitely many terms (coefficient*monomial) which are combined by the usual polynomial operations (see Section 6.2.4.2 [poly expressions (plural)], page 163). Polynomials can only be defined or accessed with respect to a basering which determines the coefficient type, the names of the indeterminants and the monomial ordering.
Example:

```
ring r=32003,(x,y,z),dp;
poly f=x3+y5+z2;
```

Remark: Remember the conventions on polynomial multiplication we follow (*-multiplication in Section 6.1 [Getting started with PLURAL], page 151).

6.2.4.1 poly declarations (plural)

Syntax: `poly` name `=` poly_expression `;`

Purpose: defines a polynomial.

Default: 0

Example:

```
ring r = 32003,(x,y,z),dp;
ncalgebra(-1,1);
// ring of some differential-like operators
r;
↦ //   characteristic : 32003
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    x y z
```

```
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    yx=-xy+1
↦ //    zx=-xz+1
↦ //    zy=-yz+1
yx;      // not correct input
↦ xy
y*x;     // correct input
↦ -xy+1
poly s1  = x3y2+151x5y+186xy6+169y9;
poly s2  = 1*x^2*y^2*z^2+3z8;
poly s3  = 5/4x4y2+4/5*x*y^5+2x2y2z3+y7+11x10;
int a,b,c,t=37,5,4,1;
poly f=3*x^a+x*y^(b+c)+t*x^a*y^b*z^c;
f;
↦ x37y5z4+3x37+xy9
short = 0;
f;
↦ x^37*y^5*z^4+3*x^37+x*y^9
```

6.2.4.2 poly expressions (plural)

A poly expression is (optional parts in square brackets):

1. a monomial (there are NO spaces allowed inside a monomial)

   `[coefficient] ring_var [exponent] [ring_var [exponent] ...]`

   monomials which contain an indexed ring variable must be built from `ring_var` and `coefficient` with the operations `*` and `^`
2. an identifier of type poly
3. a function returning poly
4. poly expressions combined by the arithmetic operations `+`, `-`, `*`, `/`, or `^`.
5. a type cast to poly

Example:

```
ring r=0,(x,y),dp;
ncalgebra(1,1);  // make it a Weyl algebra
r;
↦ //   characteristic : 0
↦ //   number of vars : 2
↦ //        block   1 : ordering dp
↦ //                  : names    x y
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    yx=xy+1
yx;      // not correct input
↦ xy
y*x;     // correct input
```

```
↦ xy+1
poly f = 10x2*y3 + 2y2*x^2 - 2*x*y + y - x + 2;
lead(f);
↦ 10x2y3
leadmonom(f);
↦ x2y3
simplify(f,1);     // normalize leading coefficient
↦ x2y3+1/5x2y2+3/5xy-1/10x+1/10y+3/5
cleardenom(f);
↦ 10x2y3+2x2y2+6xy-x+y+6
```

6.2.4.3 poly operations (plural)

`+` addition

`-` negation or subtraction

`*` multiplication

`/` division by a monomial, non divisible terms yield 0

`^`, `**` power by a positive integer

`<`, `<=`, `>`, `>=`, `==`, `<>`
comparison (of leading monomials w.r.t. monomial ordering)

poly_expression `[` intvec_expression `]`
the sum of monomials at the indicated places w.r.t. the monomial ordering

6.2.4.4 poly related functions (plural)

`bracket` computes the Lie bracket of two polinomials (see Section 6.3.9 [bracket], page 171)

`lead` leading term (see Section 4.1.63 [lead], page 186)

`leadcoef` coefficient of the leading term (see Section 4.1.64 [leadcoef], page 187)

`leadexp` the exponent vector of the leading monomial (see Section 4.1.65 [leadexp], page 188)

`leadmonom`
leading monomial (see Section 4.1.66 [leadmonom], page 188)

`oppose` creates an opposite poly of a given poly from the given ring into a basering (see Section 6.3.23 [oppose], page 185)

`reduce` left normal form with respect to a left Groebner basis (see Section 6.3.27 [reduce (plural)], page 189)

`simplify` normalize a polynomial (see Section 4.1.118 [simplify], page 232)

`size` number of monomials (see Section 4.1.119 [size], page 233)

`subst` substitute a ring variable (see Section 6.3.29 [subst (plural)], page 192)

`var` the indicated variable of the ring (see Section 4.1.137 [var], page 249)

6.2.5 qring (plural)

PLURAL offers the possibility to compute within factor-rings modulo two-sided ideals. The ideal has to be given as a two-sided Groebner basis (see Section 6.3.31 [twostd], page 193 command).

For a detailed description of the concept of rings and quotient rings see Section 2.3 [Rings and orderings], page 33.

Note: we highly recommend to turn on `option(redSB); option(redTail);` while computing in qrings. Otherwise results may have a difficult interpretation.

6.2.5.1 qring declaration (plural)

Syntax: `qring` name `=` ideal_expression `;`

Default: none

Purpose: declares a quotient ring as the basering modulo an `ideal_expression` and sets it as current basering.

Note: reports error if an ideal is not a two-sided Groebner basis.

Example:

```
ring r=0,(z,u,v,w),dp;
ncalgebra(-1,0); // an anticommutative algebra
option(redSB);
option(redTail);
ideal i=z^2,u^2,v^2,w^2;
qring Q = i;  // incorrect call produces error
↦ // ** i is no standard basis
↦ // ** i is no twosided standard basis
kill Q;
setring r;  // go back to the ring r
qring q=twostd(i); // now it is an exterior algebra
q;
↦ //   characteristic : 0
↦ //   number of vars : 4
↦ //        block   1 : ordering dp
↦ //                  : names    z u v w
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    uz=-zu
↦ //    vz=-zv
↦ //    wz=-zw
↦ //    vu=-uv
↦ //    wu=-uw
↦ //    wv=-vw
↦ // quotient ring from ideal
↦ _[1]=w2
```

```
↦ _[2]=v2
↦ _[3]=u2
↦ _[4]=z2
poly k = (v-u)*(zv+u-w);
k; // the output is not yet totally reduced
↦ zuv-zv2-u2-uv+uw-vw
poly ek=reduce(k,std(0));
ek; // the reduced form
↦ zuv-uv+uw-vw
```

6.2.5.2 qring related functions (plural)

`envelope` enveloping ring (see Section 6.3.11 [envelope], page 173)

`nvars` number of ring variables (see Section 4.1.89 [nvars], page 205)

`opposite` opposite ring (see Section 6.3.24 [opposite], page 186)

`setring` set a new basering (see Section 4.1.116 [setring], page 229)

6.2.6 resolution (plural)

The type resolution is intended as an intermediate representation which internally retains additional information obtained during computation of resolutions. It furthermore enables the use of partial results to compute, for example, Betti numbers or minimal resolutions. Like ideals and modules, a resolution can only be defined w.r.t. a basering.

Note: to access the elements of a resolution, it has to be assigned to a list. This assignment also completes computations and may therefore take time, (resp. an access directly with the brackets `[ , ]` causes implicitly a cast to a list).

6.2.6.1 resolution declarations (plural)

Syntax: `resolution` name `=` resolution_expression `;`

Purpose: defines a resolution.

Default: none

Example:

```
ring r=0,(x,y,z),dp;
matrix D[3][3];
D[1,2]=z;
ncalgebra(1,D); // it is a Heisenberg algebra
ideal i=z2+z,x+y;
resolution re=nres(i,0);
re;
↦  1      2      1
↦ r <--  r <--  r
↦
```

```
↦ 0      1      2
↦ resolution not minimized yet
↦
list l = re;
l;
↦ [1]:
↦    _[1]=z2+z
↦    _[2]=x+y
↦ [2]:
↦    _[1]=z2*gen(2)-x*gen(1)-y*gen(1)+z*gen(2)
↦ [3]:
↦    _[1]=0
print(matrix(l[2]));
↦ -x-y,
↦ z2+z
```

6.2.6.2 resolution expressions (plural)

A resolution expression is:

1. an identifier of type resolution
2. a function returning a resolution
3. a type cast to resolution from a list of ideals, resp. modules.

6.2.6.3 resolution related functions (plural)

`betti` Betti numbers of a resolution (see Section 6.3.8 [betti (plural)], page 170)

`minres` minimizes a free resolution (see Section 6.3.18 [minres (plural)], page 179)

`mres` computes a minimal free resolution of an ideal resp. module and a minimal set of generators of the given ideal resp. module (see Section 6.3.20 [mres (plural)], page 180)

`nres` computes a free resolution of an ideal resp. module M which is minimized from the second module on (see Section 6.3.22 [nres (plural)], page 184)

6.2.7 ring (plural)

Rings are used to describe properties of polynomials, ideals etc. Almost all computations in PLURAL require a basering. For a detailed description of the concept of rings see Section 2.3 [Rings and orderings], page 33.

Note: PLURAL works with global orderings only (see Section 6.1 [Getting started with PLURAL], page 151).

6.2.7.1 ring declarations (plural)

Syntax: `ring` name `=` `(` coefficient_field `),` `(` names_of_ring_variables `),` `(` ordering `);`

Default: `2147483629,(x,y,z),(dp,C);`

Purpose: declares a ring and sets it as the actual basering.

The coefficient_field is given by one of the following:

1. a non-negative int_expression less or equal 2147483629.
2. an expression_list of an int_expression and one or more names.
3. the name `real`.
4. an expression_list of the name `real` and an int_expression.
5. an expression_list of the name `complex`, an optional int_expression and a name.

'names_of_ring_variables' must be a list of names or indexed names.

'ordering' is a list of block orderings where each block ordering is either

1. `lp`, `dp`, `Dp`, optionally followed by a size parameter in parentheses.
2. `wp`, `Wp`, or `a` followed by a weight vector given as an intvec_expression in parentheses.
3. `M` followed by an intmat_expression in parentheses.
4. `c` or `C`.

If one of coefficient_field, names_of_ring_variables, and ordering consists of only one entry, the parentheses around this entry may be omitted.

In order to create the non-commutative extension, use Section 6.3.21 [ncalgebra], page 182.

6.2.7.2 ring operations (plural)

`+` construct a tensor product $C = A \otimes_{\mathbf{K}} B$ of two G-algebras A and B over the groung field.

Let

$A = k_1\langle x_1, \ldots, x_n \mid \{x_j x_i = c_{ij} \cdot x_i x_j + d_{ij}\}, 1 \leq i < j \leq n\rangle$, and $B = k_2\langle y_1, \ldots, y_m \mid \{y_j y_i = q_{ij} \cdot y_i y_j + r_{ij}\}, 1 \leq i < j \leq m\rangle$

be two G -algebras, then C is defined to be the algebra

$C = K\langle x_1, \ldots, x_n, y_1, \ldots, y_m \mid \{x_j x_i = c_{ij} \cdot x_i x_j + d_{ij}, 1 \leq i < j \leq n\}, \{y_j y_i = q_{ij} \cdot y_i y_j + r_{ij}, 1 \leq i < j \leq m\}, \{y_j x_i = x_i y_j, 1 \leq j \leq m, 1 \leq i \leq n\}\rangle$.

Concerning the ground fields k_1 resp. k_2 of A resp. B , take the following guide lines for $A \otimes_{\mathbf{K}} B$ into consideration:

- Neither k_1 nor k_2 may be R or C .
- If the characteristic of k_1 and k_2 differs, then one of them must be Q .

- At most one of k_1 and k_2 may have parameters.
- If one of k_1 and k_2 is an algebraic extension of Z/p it may not be defined by a `charstr` of type `(p^n,a)`.

Example:

```
LIB "ncalg.lib";
def a = makeUsl2();       // U(sl_2) in e,f,h presentation
ring W = 0,(x,d),dp;
Weyl();              // 1st Weyl algebra in x,d
def S = a+W;
setring S;
S;
↦ //   characteristic : 0
↦ //   number of vars : 5
↦ //        block   1 : ordering dp
↦ //                  : names    e f h
↦ //        block   2 : ordering dp
↦ //                  : names    x d
↦ //        block   3 : ordering C
↦ //   noncommutative relations:
↦ //    fe=ef-h
↦ //    he=eh+2e
↦ //    hf=fh-2f
↦ //    dx=xd+1
```

6.2.7.3 ring related functions (plural)

`charstr` description of the coefficient field of a ring (see Section 4.1.6 [charstr], page 147)

`envelope` enveloping ring (see Section 6.3.11 [envelope], page 173)

`npars` number of ring parameters (see Section 4.1.86 [npars], page 203)

`nvars` number of ring variables (see Section 4.1.89 [nvars], page 205)

`opposite` opposite ring (see Section 6.3.24 [opposite], page 186)

`ordstr` monomial ordering of a ring (see Section 4.1.93 [ordstr], page 211)

`parstr` names of all ring parameters or the name of the n-th ring parameter (see Section 4.1.96 [parstr], page 212)

`qring` quotient ring (see Section 6.2.5 [qring (plural)], page 165)

`setring` set a new basering (see Section 4.1.116 [setring], page 229)

`varstr` names of all ring variables or the name of the n-th ring variable (see Section 4.1.138 [varstr], page 250)

6.3 Functions (plural)

This chapter gives a complete reference of all functions and commands of the PLURAL kernel, i.e. all built-in commands (for the PLURAL libraries see Section 6.5 [PLURAL libraries], page 201).

The general syntax of a function is

[target =] function_name (<arguments>);

Note, that both **Control structures** and **System variables** of PLURAL are the same as of SINGULAR (see Section 4.2 [Control structures], page 253, Section 4.3 [System variables], page 264).

6.3.8 betti (plural)

Syntax: `betti ( list_expression )`
`betti ( resolution_expression )`
`betti ( list_expression , int_expression )`
`betti ( resolution_expression , int_expression )`

Type: intmat

Note: in the noncommutative case, computing Betti numbers makes sense only if the basering R has homogeneous relations

Purpose: with 1 argument: computes the graded Betti numbers of a minimal resolution of R^n/M, if R denotes the basering and M a homogeneous submodule of R^n and the argument represents a resolution of R^n/M.
The entry d of the intmat at place (i, j) is the minimal number of generators in degree i+j of the j-th syzygy module (= module of relations) of R^n/M (the 0th (resp. 1st) syzygy module of R^n/M is R^n (resp. M)).
The argument is considered to be the result of a `mres` or `nres` command. This implies that a zero is only allowed (and counted) as a generator in the first module.
For the computation `betti` uses only the initial monomials. This could lead to confusing results for a non-homogeneous input.

If the optional second argument is non-zero, the Betti numbers will be minimized.

Example:

```
int i;int N=2;
ring r=0,(x(1..N),d(1..N),q(1..N)),Dp;
matrix D[3*N][3*N];
for (i=1;i<=N;i++)
{ D[i,N+i]=q(i)^2; }
ncalgebra(1,D);
// this algebra is a kind of homogenized Weyl algebra
r;
↦ // characteristic : 0
```

```
↦ // number of vars : 6
↦ //   block   1 : ordering Dp
↦ //             : names x(1) x(2) d(1) d(2) q(1) q(2)
↦ //   block   2 : ordering C
↦ // noncommutative relations:
↦ //   d(1)x(1)=x(1)*d(1)+q(1)^2
↦ //   d(2)x(2)=x(2)*d(2)+q(2)^2
ideal I = x(1),x(2),d(1),d(2),q(1),q(2);
option(redSB);
option(redTail);
resolution R = mres(I,0);
// thus R will be the full length minimal resolution
print(betti(R),"betti");
↦            0     1     2     3     4     5     6
↦ --------------------------------------------------
↦     0:     1     6    15    20    15     6     1
↦ --------------------------------------------------
↦ total:     1     6    15    20    15     6     1
```

6.3.9 bracket

Syntax: `bracket` (poly_expression, poly_expression)

Type: poly

Purpose: Computes the Lie bracket `[p,q]=pq-qp` of the first polynomial with the second. Uses special routines, based on the Leibniz rule.

Example:

```
ring r=(0,Q),(x,y,z),Dp;
minpoly=Q^2-Q+1;
matrix C[3][3];  matrix D[3][3];
C[1,2]=Q2;    C[1,3]=1/Q2;  C[2,3]=Q2;
D[1,2]=-Q*z;  D[1,3]=1/Q*y; D[2,3]=-Q*x;
ncalgebra(C,D);
// this is a quantum deformation of U(so_3),
// where Q is a 6th root of unity
poly p=Q^4*x2+y2+Q^4*z2+Q*(1-Q^4)*x*y*z;
// p is the central element of the algebra
p=p^3; // any power of a central element is central
poly q=(x+Q*y+Q^2*z)^4;
// take q to be some big noncentral element
size(q); // check how many monomials are in big poly q
↦ 28
bracket(p,q); // check p*q=q*p
↦ 0
// a more common behaviour of the bracket follows:
bracket(x+Q*y+Q^2*z,z);
↦ (Q+1)*xz+(Q+1)*yz+(Q-1)*x+(Q-1)*y
```

6.3.10 eliminate (plural)

Syntax: `eliminate` (ideal_expression, product_of_ring_variables)
`eliminate` (module_expression, product_of_ring_variables)

Type: the same as the type of the first argument

Purpose: eliminates variables occurring as factors of the second argument from an ideal (resp. a submodule of a free module), by intersecting it (resp. each component of the submodule) with the subring not containing these variables.

Note: `eliminate` does not need neither a special ordering on the basering nor a Groebner basis as input.

Remark: in a noncommutative algebra, not every subset of a set of variables generates a proper subalgebra. But if it is so, there may be cases, when no elimination is possible. In these situations error messages will be reported.

Example:

```
ring r=0,(e,f,h,a),Dp;
matrix d[4][4];
d[1,2]=-h; d[1,3]=2*e; d[2,3]=-2*f;
ncalgebra(1,d);
// this algebra is U(sl_2), tensored with K[a] over K
option(redSB);
option(redTail);
poly  p = 4*e*f+h^2-2*h - a;
// p is a central element with parameter
ideal I = e^3, f^3, h^3-4*h, p; // take this ideal
// and intersect I with the ring K[a]
ideal J = eliminate(I, e*f*h);
// if we want substitute 'a' with a value,
// it has to be a root of this polynomial:
J;
↦ J[1]=a3-32a2+192a
// now we try to eliminate h,
// that is we intersect I with the subalgebra S,
// generated by e and f.
// But S is not closed in itself, since f*e-e*f=-h !
// the next command will definitely produce an error
eliminate(I,h);
↦ ? no elimination is possible:
                subalgebra is not admissible
↦ ? error occurred in line 13: `eliminate(I,h);
// since a commutes with e,f,h, we can eliminate it:
eliminate(I,a);
↦ _[1]=h3-4h
↦ _[2]=fh2-2fh
```

```
↦ _[3]=f3
↦ _[4]=eh2+2eh
↦ _[5]=2efh-h2-2h
↦ _[6]=e3
```

See Section 6.2.1 [ideal (plural)], page 152; Section 6.2.3 [module (plural)], page 159; Section 6.3.28 [std (plural)], page 191.

6.3.11 envelope

Syntax: `envelope ( ring_name )`

Type: ring

Purpose: creates an enveloping algebra of a given algebra, that is $A^{env} = A \otimes_K A^{opp}$, where A^{opp} is the opposite algebra of A.

Remark: You have to activate the ring with the `setring` command. For the presentation, see explanation of `opposite` in Section 6.3.24 [opposite], page 186.

```
LIB "ncalg.lib";
def A = makeUsl2();
setring A; A;
↦ //   characteristic : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    e f h
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    fe=ef-h
↦ //    he=eh+2e
↦ //    hf=fh-2f
def Aenv = envelope(A);
setring Aenv;
Aenv;
↦ //   characteristic : 0
↦ //   number of vars : 6
↦ //        block   1 : ordering dp
↦ //                  : names    e f h
↦ //        block   2 : ordering a
↦ //                  : names    H F E
↦ //                  : weights  1 1 1
↦ //        block   3 : ordering ls
↦ //                  : names    H F E
↦ //        block   4 : ordering C
↦ //   noncommutative relations:
↦ //    fe=ef-h
↦ //    he=eh+2e
↦ //    hf=fh-2f
↦ //    FH=HF-2F
```

```
↦ //    EH=HE+2E
↦ //    EF=FE-H
```

See Section 6.3.23 [oppose], page 185; Section 6.3.24 [opposite], page 186.

6.3.12 fetch (plural)

Syntax: `fetch ( ring_name, name )`

Type: number, poly, vector, ideal, module, matrix or list (the same type as the second argument)

Purpose: maps objects between rings. `fetch` is the identity map between rings and qrings, the i-th variable of the source ring is mapped to the i-th variable of the basering. The coefficient fields must be compatible. (See Section 6.2.2 [map (plural)], page 156 for a description of possible mappings between different ground fields).
`fetch` offers a convenient way to change variable names or orderings, or to map objects from a ring to a quotient ring of that ring or vice versa.

Note: Compared with `imap`, `fetch` uses the position of the ring variables, not their names.

Example:

```
LIB "ncalg.lib";
def Usl2 = makeUsl2(); // this algebra is U(sl_2)
setring Usl2;
option(redSB);
option(redTail);
poly  C  = 4*e*f+h^2-2*h; // the central element of Usl2
ideal I  = e^3,f^3,h^3-4*h;
ideal J  = twostd(I);
print(matrix(J));  // print a compact presentation of J
↦ h3-4h,fh2-2fh,eh2+2eh,f2h-2f2,2efh-h2-2h, \
             e2h+2e2,f3,ef2-fh,e2f-eh-2e,e3
ideal QC = twostd(C-8);
qring Q  = QC;
ideal QJ = fetch(Usl2,J);
QJ = std(QJ);
// thus QJ is the image of I in the factor-algebra QC
print(matrix(QJ)); // print QJ compactly
↦ h3-4h,fh2-2fh,eh2+2eh,f2h-2f2,e2h+2e2,f3,e3
```

See Section 6.3.13 [imap (plural)], page 174; Section 6.2.2 [map (plural)], page 156; Section 6.2.5 [qring (plural)], page 165; Section 6.2.7 [ring (plural)], page 167.

6.3.13 imap (plural)

Syntax: `imap ( ring_name, name )`

Type: number, poly, vector, ideal, module, matrix or list (the same type as the second argument)

Purpose: identity map on common subrings. `imap` is the map between rings and qrings with compatible ground fields which is the identity on variables and parameters of the same name and 0 otherwise. (See Section 6.2.2 [map (plural)], page 156 for a description of possible mappings between different ground fields). Useful for mappings from a homogenized ring to the original ring or for mappings from/to rings with/without parameters. Compared with `fetch`, `imap` uses the names of variables and parameters. Unlike `map` and `fetch`, `imap` can map parameters to variables.

Example:

```
LIB "ncalg.lib";
ring ABP=0,(p4,p5,a,b),dp; //  a commutative ring
def Usl3 = makeUsl(3);
def BIG  = Usl3+ABP;
setring BIG;
poly P4 = 3*x(1)*y(1)+3*x(2)*y(2)+3*x(3)*y(3);
P4 = P4 +h(1)^2+h(1)*h(2)+h(2)^2-3*h(1)-3*h(2);
// P4 is a central element of Usl3 of degree 2
poly P5 = 4*x(1)*y(1) + h(1)^2 - 2*h(1);
// P5 is a central element of the subalgebra of Usl3,
// generated by x(1),y(1),h(1)
ideal J = x(1),x(2),h(1)-a,h(2)-b;
// we are interested in the module U(sl_3)/J,
// which depends on parameters a,b
ideal I = p4-P4, p5-P5;
ideal K = I, J;
poly el = x(1)*x(2)*x(3)*y(1)*y(2)*y(3)*h(1)*h(2);
ideal E = eliminate(K,el);
E; // this is the ideal of central characters in ABP
↦ E[1]=a*b+b^2-p4+p5+a+3*b
↦ E[2]=a^2-p5+2*a
↦ E[3]=b^3+p4*a-p5*a-a^2-p4*b+3*b^2
// what are the characters on nonzero a,b?
ring abP = (0,a,b),(p4,p5),dp;
ideal abE = imap(BIG, E);
option(redSB);
option(redTail);
abE = std(abE);
// here come characters (indeed, we have only one)
// that is a maximal ideal in K[p4,p5]
abE;
↦ abE[1]=p5+(-a^2-2*a)
↦ abE[2]=p4+(-a^2-a*b-3*a-b^2-3*b)
```

See Section 6.3.12 [fetch (plural)], page 174; Section 6.2.2 [map (plural)], page 156; Section 6.2.5 [qring (plural)], page 165; Section 6.2.7 [ring (plural)], page 167.

6.3.14 intersect (plural)

Syntax: `intersect` (expression_list of ideal_expression)
`intersect` (expression_list of module_expression)

Type: ideal, resp. module

Purpose: computes the intersection of ideals, resp. modules.

Example:

```
ring r=0,(x,y),dp;
ncalgebra(-1,0);
module M=[x,x],[y,0];
module N=[0,y^2],[y,x];
option(redSB);
module Res;
Res=intersect(M,N);
print(Res);
↦ y2, 0,
↦ -xy,xy2
kill r;
//--------------------------------
LIB "ncalg.lib";
ring r=0,(x,d),dp;
Weyl(); // make r into Weyl algebra
ideal I = x+d^2;
ideal J = d-1;
ideal H = intersect(I,J);
H;
↦ H[1]=d4+xd2-2d3-2xd+d2+x+2d-2
↦ H[2]=xd3+x2d-xd2+d3-x2+xd-2d2-x+1
```

6.3.15 kbase (plural)

Syntax: `kbase` (ideal_expression)
`kbase` (module_expression)
`kbase` (ideal_expression, int_expression)
`kbase` (module_expression, int_expression)

Type: the same as the input type of the first argument

Purpose:

computes the vector space basis of the factor-module that equals ring (resp. free module) modulo the ideal (resp. submodule), generated by the initial terms of the given generators.
If the factor-module is not of finite dimension, -1 is returned.

If the generators form a Groebner basis, this is the same as the vector space basis of the factor-module.

Note: in the noncommutative case, a ring modulo an ideal has a ring stucture if and only if an ideal is two-sided.

Example:

```
ring r=0,(x,y,z),dp;
matrix d[3][3];
d[1,2]=-z;  d[1,3]=2x;  d[2,3]=-2y;
ncalgebra(1,d); // this algebra is U(sl_2)
ideal i=x2,y2,z2-1;
i=std(i);
print(matrix(i));  // print a compact presentation of i
↦ z2-1,yz-y,xz+x,y2,2xy-z-1,x2
kbase(i);
↦ _[1]=z
↦ _[2]=y
↦ _[3]=x
↦ _[4]=1
vdim(i);
↦ 4
ideal j=x,z-1;
j=std(j);
kbase(j,3);
↦ _[1]=y3
```

See Section 6.2.1 [ideal (plural)], page 152; Section 6.2.3 [module (plural)], page 159; Section 6.3.32 [vdim (plural)], page 194.

6.3.16 lift (plural)

Syntax: `lift` (ideal_expression, subideal_expression)
`lift` (module_expression, submodule_expression)

Type: matrix

Purpose: computes the (left) transformation matrix which expresses the (left) generators of a submodule in terms of the (left) generators of a module. Uses different algorithms for modules which are (resp. are not) represented by a Groebner basis.
More precisely, if `m` is the module, `sm` the submodule, and `T` the transformation matrix returned by lift, then `transpose(matrix(sm)) = transpose(T)*transpose(m)` and `module(transpose(sm)) = module(transpose(T)*transpose(m))`. If `m` and `sm` are ideals, `ideal(sm) = ideal(transpose(T)*transpose(m))`.

Note: Gives a warning if `sm` is not a submodule.

Example:

```
ring r = (0,a),(e,f,h),(c,dp);
matrix D[3][3];
D[1,2]=-h;  D[1,3]=2*e;  D[2,3]=-2*f;
```

```
ncalgebra(1,D); // this algebra is a parametric U(sl_2)
ideal i = e,h-a; // consider this parametric ideal
i = std(i);
print(matrix(i)); // print a compact presentation of i
↦ h+(-a),e
poly Z = 4*e*f+h^2-2*h; // a central element
Z = Z - NF(Z,i); // a central character
ideal j = std(Z);
j;
↦ j[1]=4*ef+h2-2*h+(-a2-2a)
matrix T = lift(i,j);
print(T);
↦ h+(a+2),
↦ 4*f
ideal tj = ideal(transpose(T)*transpose(matrix(i)));
std(ideal(j-tj)); // test
↦ _[1]=0
```

See Section 6.2.1 [ideal (plural)], page 152; Section 6.3.17 [liftstd (plural)], page 178; Section 6.2.3 [module (plural)], page 159.

6.3.17 liftstd (plural)

Syntax: `liftstd` (ideal_expression, matrix_name)
`liftstd` (module_expression, matrix_name)

Type: ideal or module

Purpose: returns a Groebner basis of an ideal or module and the transformation matrix from the given ideal, resp. module, to the Groebner basis.
That is, if `m` is the ideal or module, `sm` is the Groebner basis of `m`, returned by `liftstd`, and `T` is the transformation matrix, then `transpose(matrix(sm))=transpose(T)*transpose(matrix(m))` and `sm=module(transpose(transpose(T)*transpose(matrix(m))))`.
If `m` is an ideal, `sm=ideal(transpose(T)*transpose(matrix(m)))`.

Example:

```
LIB "ncalg.lib";
def A = makeUsl2();
setring A;  // this algebra is U(sl_2)
ideal i = e2,f;
option(redSB);
option(redTail);
matrix T;
ideal j = liftstd(i,T);
// the Groebner basis in a compact form:
print(matrix(j));
↦ f,2h2+2h,2eh+2e,e2
print(T);  // the transformation matrix
↦ 0,f2,         -f,1,
```

```
↦ 1,-e2f+4eh+8e,e2,0
ideal tj = ideal(transpose(T)*transpose(matrix(i)));
std(ideal(j-tj)); // test
↦ _[1]=0
```

See Section 6.2.1 [ideal (plural)], page 152; Section 6.2.7 [ring (plural)], page 167; Section 6.3.28 [std (plural)], page 191.

6.3.18 minres (plural)

Syntax: `minres` (list_expression)

Type: list

Syntax: `minres` (resolution_expression)

Type: resolution

Purpose: minimizes a free resolution of an ideal or module given by the list_expression, resp. resolution_expression.

Example:

```
LIB "ncalg.lib";
def A = makeUsl2();
setring A; // this algebra is U(sl_2)
ideal i=e,f,h;
i=std(i);
resolution F=nres(i,0);   F;
↦  1      3      3      1
↦ A <--  A <--  A <--  A
↦
↦ 0      1      2      3
↦ resolution not minimized yet
↦
print(matrix(F[1])); // print F's compactly
↦ h,f,e
print(matrix(F[2]));
↦ f,   e,   -1,
↦ -h-2,0,   e,
↦ 0,   -h+2,-f
print(matrix(F[3]));
↦ e,
↦ -f,
↦ h
resolution MF=minres(F);  MF;
↦  1      3      3      1
↦ A <--  A <--  A <--  A
↦
↦ 0      1      2      3
↦
print(matrix(MF[1]));
```

```
↦ f,e
print(matrix(MF[2]));
↦ -ef+2h+2,-e2,
↦ f2,      ef+h-2
print(matrix(MF[3]));
↦ e,
↦ -f
```

See Section 6.3.20 [mres (plural)], page 180; Section 6.3.22 [nres (plural)], page 184.

6.3.19 modulo (plural)

Syntax: `modulo ( ideal_expression, ideal_expression )`
`modulo ( module_expression, module_expression )`

Type: module

Purpose: `modulo(h1,h2)` represents $h_1/(h_1 \cap h_2) \cong (h_1 + h_2)/h_2$, where h_1 and h_2 are considered as submodules of the same free module R^s (s=1 for ideals).
Let H_1 (resp. H_2) be the matrix of size $l \times k$ (resp. $l \times m$), having the generators of h_1 (resp. h_2) as columns.
Then $h_1/(h_1 \cap h_2) \cong R^k/ker(\overline{H_1})$, where $\overline{H_1} : R^k \to R^s/Im(H_2) = R^s/h_2$ is the induced map.
`modulo(h1,h2)` returns generators of the kernel of this induced map.

Example:

```
LIB "ncalg.lib";
def A = makeUsl2();
setring A; // this algebra is U(sl_2)
option(redSB);
option(redTail);
ideal I = e2,f2,h2-1;
I = twostd(I);
print(matrix(I)); // print I in a compact form
↦ h2-1,fh-f,eh+e,f2,2ef-h-1,e2
ideal E = std(e);
ideal T = modulo(E,I);
T = NF(std(I+T),I);
T = std(T);
T;
↦ T[1]=h-1
↦ T[2]=e
```

See also Section 6.3.30 [syz (plural)], page 192.

6.3.20 mres (plural)

Syntax: `mres ( ideal_expression, int_expression )`
`mres ( module_expression, int_expression )`

Type: resolution

Purpose: computes a minimal free resolution of an ideal or module M with the Groebner basis method. More precisely, let A=`matrix`(M), then `mres` computes a free resolution of $coker(A) = F_0/M$

$$... \longrightarrow F_2 \xrightarrow{A_2} F_1 \xrightarrow{A_1} F_0 \longrightarrow F_0/M \longrightarrow 0,$$

where the columns of the matrix A_1 are a (possibly) minimal set of generators of M . If the int expression k is not zero, then the computation stops after k steps and returns a list of modules $M_i =$ `module`(A_i), $i = 1 \dots k$.
`mres(M,0)` returns a resolution consisting of at most n+2 modules, where n is the number of variables of the basering. Let `list L=mres(M,0);` then `L[1]` consists of a minimal set of generators of the input, `L[2]` consists of a minimal set of generators for the first syzygy module of `L[1]`, etc., until `L[p+1]`, such that `L[i]` $\neq 0$ for $i \leq p$, but `L[p+1]` (the first syzygy module of `L[p]`) is 0 (if the basering is not a qring).

Note: Accessing single elements of a resolution may require that some partial computations have to be finished and may therefore take some time.

Example:

```
LIB "ncalg.lib";
def A = makeUsl2();
setring A; // this algebra is U(sl_2)
option(redSB);
option(redTail);
ideal i = e,f,h;
i = std(i);
resolution M=mres(i,0);
M;
↦  1      2      2      1
↦ A <--  A <--  A <--  A
↦
↦ 0      1      2      3
↦
print(matrix(M[1])); // print M's in a compact way
↦ f,e
print(matrix(M[2]));
↦ ef-2h-2,e2,
↦ -f2,    -ef-h+2
// see the exactness at this point
std(ideal(transpose(M[2])*transpose(M[1])));
↦ _[1]=0
print(matrix(M[3]));
↦ e,
↦ -f
// see the exactness at this point
std(ideal(transpose(M[3])*transpose(M[2])));
```

```
↦ _[1]=0
```

See Section 6.2.1 [ideal (plural)], page 152; Section 6.3.18 [minres (plural)], page 179; Section 6.2.3 [module (plural)], page 159; Section 6.3.22 [nres (plural)], page 184.

6.3.21 ncalgebra

Syntax:

`ncalgebra(` matrix_expression C, matrix_expression D `)`
`ncalgebra(` number_expression n, matrix_expression D `)`
`ncalgebra(` matrix_expression C, poly_expression p `)`
`ncalgebra(` number_expression n, poly_expression p `)`

Type: ring

Purpose: Executed in the basering `r`, say, in k variables $x_1, \ldots, x_k$, `ncalgebra` creates the noncommutative extension of `r` subject to relations $\{x_j x_i = c_{ij} \cdot x_i x_j + d_{ij}, 1 \leq i < j \leq k\}$, where c_{ij} and d_{ij} must be put into two strictly upper triangular matrices `C` with entries c_{ij} from the ground field of `r` and `D` with polynomial entries d_{ij} from `r`. See all the details in Section 6.4.33 [G-algebras], page 195.
If $\forall i < j$, $c_{ij} = n$, one can input a number `n` instead of matrix `C`.
If $\forall i < j$, $d_{ij} = p$, one can input a poly `p` instead of matrix `D`.

Remark: At present, Plural does not check the non-degeneracy conditions (see Section 6.4.33 [G-algebras], page 195) while setting an algebra.

Example:

```
LIB "nctools.lib";
// ----- first example: C, D are matrices ------
ring r1 = (0,Q),(x,y,z),Dp;
minpoly = rootofUnity(6);
matrix C[3][3];
matrix D[3][3];
C[1,2]=Q2;   C[1,3]=1/Q2;  C[2,3]=Q2;
D[1,2]=-Q*z; D[1,3]=1/Q*y; D[2,3]=-Q*x;
ncalgebra(C,D);
// this algebra is a quantum deformation U'_q(so_3),
// where Q is a 6th root of unity
r1;
↦ //   characteristic : 0
↦ //   1 parameter    : Q
↦ //   minpoly        : (Q2-Q+1)
↦ //   number of vars : 3
↦ //        block   1 : ordering Dp
↦ //                  : names    x y z
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    yx=(Q-1)*xy+(-Q)*z
↦ //    zx=(-Q)*xz+(-Q+1)*y
```

```
↦ //    zy=(Q-1)*yz+(-Q)*x
kill r1;
// ------ second example : number n=1, D is a matrix
ring r2=0,(Xa,Xb,Xc,Ya,Yb,Yc,Ha,Hb),dp;
matrix d[8][8];
d[1,2]=-Xc; d[1,4]=-Ha; d[1,6]=Yb;  d[1,7]=2*Xa;
d[1,8]=-Xa; d[2,5]=-Hb; d[2,6]=-Ya; d[2,7]=-Xb;
d[2,8]=2*Xb; d[3,4]=Xb; d[3,5]=-Xa; d[3,6]=-Ha-Hb;
d[3,7]=Xc;   d[3,8]=Xc; d[4,5]=Yc; d[4,7]=-2*Ya;
d[4,8]=Ya;  d[5,7]=Yb; d[5,8]=-2*Yb;
d[6,7]=-Yc; d[6,8]=-Yc;
ncalgebra(1,d);  // this algebra is U(sl_3)
r2;
↦ //   characteristic : 0
↦ //   number of vars : 8
↦ //      block   1 : ordering dp
↦ //                : names    Xa Xb Xc Ya Yb Yc Ha Hb
↦ //      block   2 : ordering C
↦ //   noncommutative relations:
↦ //    XbXa=Xa*Xb-Xc
↦ //    YaXa=Xa*Ya-Ha
↦ //    YcXa=Xa*Yc+Yb
↦ //    HaXa=Xa*Ha+2*Xa
↦ //    HbXa=Xa*Hb-Xa
↦ //    YbXb=Xb*Yb-Hb
↦ //    YcXb=Xb*Yc-Ya
↦ //    HaXb=Xb*Ha-Xb
↦ //    HbXb=Xb*Hb+2*Xb
↦ //    YaXc=Xc*Ya+Xb
↦ //    YbXc=Xc*Yb-Xa
↦ //    YcXc=Xc*Yc-Ha-Hb
↦ //    HaXc=Xc*Ha+Xc
↦ //    HbXc=Xc*Hb+Xc
↦ //    YbYa=Ya*Yb+Yc
↦ //    HaYa=Ya*Ha-2*Ya
↦ //    HbYa=Ya*Hb+Ya
↦ //    HaYb=Yb*Ha+Yb
↦ //    HbYb=Yb*Hb-2*Yb
↦ //    HaYc=Yc*Ha-Yc
↦ //    HbYc=Yc*Hb-Yc
kill r2;
// --- third example : C is a matrix, p=0 is a poly
ring r3=0,(a,b,c,d),lp;
matrix c[4][4];
c[1,2]=1; c[1,3]=3; c[1,4]=-2;
c[2,3]=-1; c[2,4]=-3; c[3,4]=1;
ncalgebra(c,0); // it is a quasi--commutative algebra
r3;
```

```
↦ //   characteristic : 0
↦ //   number of vars : 4
↦ //        block   1 : ordering lp
↦ //                  : names    a b c d
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    ca=3ac
↦ //    da=-2ad
↦ //    cb=-bc
↦ //    db=-3bd
kill r3;
// -- fourth example: number n = -1, poly p = 3w
ring r4=0,(u,v,w),dp;
ncalgebra(-1,3w);
r4;
↦ //   characteristic : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    u v w
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    vu=-uv+3w
↦ //    wu=-uw+3w
↦ //    wv=-vw+3w
kill r4;
```

See also Section 6.5.41 [ncalg_lib], page 213; Section 6.5.43 [nctools_lib], page 226; Section 6.5.44 [qmatrix_lib], page 236.

6.3.22 nres (plural)

Syntax: `nres` (ideal_expression, int_expression)
`nres` (module_expression, int_expression)

Type: resolution

Purpose: computes a free resolution of an ideal or module which is minimized from the second module on (by the Groebner basis method).

Example:

```
LIB "ncalg.lib";
def A = makeUsl2();
setring A; // this algebra is U(sl_2)
option(redSB);
option(redTail);
ideal i = e,f,h;
i = std(i);
resolution F=nres(i,0);
F;
↦  1      3      3      1
```

```
↦ A <--  A <--  A <--  A
↦
↦ 0      1      2      3
↦ resolution not minimized yet
↦
// print the resolution componentwise:
print(matrix(F[1]));
↦ h,f,e
print(matrix(F[2]));
↦ f,   e,   -1,
↦ -h-2,0,   e,
↦ 0,   -h+2,-f
// see the exactness at this point:
std(ideal(transpose(F[2])*transpose(F[1])));
↦ _[1]=0
print(matrix(F[3]));
↦ e,
↦ -f,
↦ h
// see the exactness at this point:
std(ideal(transpose(F[3])*transpose(F[2])));
↦ _[1]=0
```

See Section 6.2.1 [ideal (plural)], page 152; Section 6.3.18 [minres (plural)], page 179; Section 6.2.3 [module (plural)], page 159; Section 6.3.20 [mres (plural)], page 180.

6.3.23 oppose

Syntax: `oppose ( ring_name, name )`

Type: poly, vector, ideal, module or matrix (the same type as the second argument)

Purpose: for a given object in the given ring, creates its opposite object in the opposite (Section 6.3.24 [opposite], page 186) ring (the last one is assumed to be the current ring).

Remark: for any object O, $(O^{opp})^{opp} = O$.

```
LIB "ncalg.lib";
def r = makeUsl2();
setring r;
matrix m[3][4];
poly   p  = (h^2-1)*f*e;
vector v  = [1,e*h,0,p];
ideal  i  = h*e, f^2*e,h*f*e;
m         = e,f,h,1,0,h^2, p,0,0,1,e^2,e*f*h+1;
module mm = module(m);
def b     = opposite(r);
// we will oppose these objects: p,v,i,m,mm
setring b; b;
```

```
↦ //   characteristic : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering a
↦ //                  : names    H F E
↦ //                  : weights  1 1 1
↦ //        block   2 : ordering ls
↦ //                  : names    H F E
↦ //        block   3 : ordering C
↦ //   noncommutative relations:
↦ //    FH=HF-2F
↦ //    EH=HE+2E
↦ //    EF=FE-H
poly P    = oppose(r,p);
vector V  = oppose(r,v);
ideal  I  = oppose(r,i);
matrix M  = oppose(r,m);
module MM = oppose(r,mm);
setring r;   // now let's check the correctness:
// print compact presentations of objects
print(matrix(oppose(b,P)-p));
↦ 0
print(matrix(oppose(b,V)-v));
↦ 0
print(matrix(oppose(b,I)-i));
↦ 0,0,0
print(matrix(oppose(b,M)-m));
↦ 0,0,0,0,
↦ 0,0,0,0,
↦ 0,0,0,0
print(matrix(oppose(b,MM)-mm));
↦ 0,0,0,0,
↦ 0,0,0,0,
↦ 0,0,0,0
```

See Section 6.3.11 [envelope], page 173; Section 6.3.24 [opposite], page 186.

6.3.24 opposite

Syntax: `opposite ( ring_name )`

Type: ring

Purpose: creates an opposite algebra of a given algebra.

Note: activate the ring with the `setring` command.
An opposite algebra of a given algebra (A ,.) is an algebra (A ,*) with the same vectorspace but with the opposite multiplication, i.e.
$\forall\ f, g \in A^{opp}$, a new multiplication $*$ on A^{opp} is defined to be $f*g := g \cdot f$.

Remark: Starting from the variables x_1,...,x_N and the ordering `<` of the given algebra, an opposite algebra will have variables X_N,...,X_1 (where the

case and the position are reverted). Moreover, it is equipped with an opposed ordering `<_opp` (it is given by the matrix, obtained from the matrix ordering of `<` with the reverse order of columns).

```
LIB "ncalg.lib";
def B = makeQso3(3);
// this algebra is a quantum deformation of U(so_3),
// where the quantum parameter is a 6th root of unity
setring B; B;
↦ //   characteristic : 0
↦ //   1 parameter    : Q
↦ //   minpoly        : (Q2-Q+1)
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    x y z
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    yx=(Q-1)*xy+(-Q)*z
↦ //    zx=(-Q)*xz+(-Q+1)*y
↦ //    zy=(Q-1)*yz+(-Q)*x
def Bopp = opposite(B);
setring Bopp;
Bopp;
↦ //   characteristic : 0
↦ //   1 parameter    : Q
↦ //   minpoly        : (Q2-Q+1)
↦ //   number of vars : 3
↦ //        block   1 : ordering a
↦ //                  : names    Z Y X
↦ //                  : weights  1 1 1
↦ //        block   2 : ordering ls
↦ //                  : names    Z Y X
↦ //        block   3 : ordering C
↦ //   noncommutative relations:
↦ //    YZ=(Q-1)*ZY+(-Q)*X
↦ //    XZ=(-Q)*ZX+(-Q+1)*Y
↦ //    XY=(Q-1)*YX+(-Q)*Z
```

See Section B.2.6 [Matrix orderings], page 257; Section 6.3.11 [envelope], page 173; Section 6.3.23 [oppose], page 185.

6.3.25 preimage (plural)

Syntax: `preimage` (ring_name, map_name, ideal_name)
`preimage` (ring_name, ideal_expression, ideal_name)

Type: ideal

Purpose: returns the preimage of an ideal under a given map. The second argument has to be a map from the basering to the given ring (or an ideal defining such a map), and the ideal has to be an ideal in the given ring.

Note: To compute the kernel of a map, the preimage of zero has to be determined. Hence there is no special command for computing the kernel of a map in PLURAL.

Remark: In the noncommutative case, it is implemented only for maps A -> B , where A is a commutative ring.

Example:

```
LIB "ncalg.lib";
ring   R = 0,a,dp;
def Usl2 = makeUsl2();
setring Usl2;
poly  C = 4*e*f+h^2-2*h;
// C is a central element of U(sl2)
ideal I = e^3, f^3, h^3-4*h;
ideal J = twostd(I); // two-sided GB
ideal K = std(I);    // left GB
map Phi = R,C;
setring R;
ideal PreJ = preimage(Usl2,Phi,J);
// PreJ gives the central character of J
PreJ;
↦ PreJ[1]=a2-8a
factorize(PreJ[1],1);
// hence, there are two simple characters for J
↦ _[1]=a
↦ _[2]=a-8
ideal PreK = preimage(Usl2,Phi,K);
// the central character of K
PreK;
↦ PreK[1]=a3-32a2+192a
factorize(PreK[1],1);
// hence, there are three simple characters for K
↦ _[1]=a
↦ _[2]=a-8
↦ _[3]=a-24
```

See Section 6.2.1 [ideal (plural)], page 152; Section 6.2.2 [map (plural)], page 156; Section 6.2.7 [ring (plural)], page 167.

6.3.26 quotient (plural)

Syntax: `quotient` (ideal_expression, ideal_expression)
`quotient` (module_expression, module_expression)

Type: ideal

Syntax: `quotient` (module_expression, ideal_expression)

Type: module

Purpose: computes the ideal quotient, resp. module quotient. Let `R` be the base-ring, `I,J` ideals and `M` a module in $\mathtt{R}^n$. Then

`quotient(I,J)`$= \{a \in R \mid aJ \subset I\}$,

`quotient(M,J)`$= \{b \in R^n \mid bJ \subset M\}$.

Note: It must used for two-sided ideals (bimodules) only, otherwise the result may have no meaning.

Example:

```
//------ a very easy example ------------
ring r=(0,q),(x,y),Dp;
ncalgebra(q,0); // this algebra is a quantum plane
option(returnSB);
poly f1  = x^3+2*x*y^2+2*x^2*y;
poly f2  = y;
poly f1' = x^2;
poly f2' = x+y;
ideal i = f1,f2;
ideal I = twostd(i);
ideal j = f1',f2';
ideal J = twostd(j);
quotient(I,J);
↦ _[1]=y
↦ _[2]=x2
kill r;
//------- a bit more complicated example
LIB "ncalg.lib";
def Usl2 = makeUsl2();
// this algebra is U(sl_2)
setring Usl2;
ideal i = e3,f3,h3-4*h;
ideal I = std(i);
poly  C = 4*e*f+h^2-2*h;
ideal H = twostd(C-8);
option(returnSB);
ideal Q = quotient(I,H);
// print a compact presentation of Q:
print(matrix(Q));
↦ h,f3,ef2-4f,e2f-6e,e3
```

See Section 6.2.1 [ideal (plural)], page 152; Section 6.2.3 [module (plural)], page 159.

6.3.27 reduce (plural)

Syntax:

`reduce` (poly_expression, ideal_expression)
`reduce` (poly_expression, ideal_expression, int_expression)
`reduce` (vector_expression, ideal_expression)
`reduce` (vector_expression, ideal_expression, int_expression)

`reduce` (vector_expression, module_expression)
`reduce` (vector_expression, module_expression, int_expression)
`reduce` (ideal_expression, ideal_expression)
`reduce` (ideal_expression, ideal_expression, int_expression)
`reduce` (module_expression, ideal_expression)
`reduce` (module_expression, ideal_expression, int_expression)
`reduce` (module_expression, module_expression)
`reduce` (module_expression, module_expression, int_expression)

Type: the type of the first argument

Purpose: reduces a polynomial, vector, ideal or module to its **left** normal form with respect to an ideal or module represented by a left Groebner basis. Returns 0 if and only if the polynomial (resp. vector, ideal, module) is an element (resp. subideal, submodule) of the ideal (resp. module). The result may have no meaning if the second argument is not a left Groebner basis.
The third (optional) argument 1 of type int forces a reduction which considers only the leading term and does no tail reduction.

Note: The commands `reduce` and `NF` are synonymous.

Example:

```
ring r=(0,a),(e,f,h),Dp;
matrix d[3][3];
d[1,2]=-h;  d[1,3]=2e;  d[2,3]=-2f;
ncalgebra(1,d);
// this algebra is a parametric U(sl_2)
ideal I=e2,f2,h2-1;
I=std(I);
// print a compact presentation of I
print(matrix(I));
↦ h2-1,fh-f,f2,eh+e,2*ef-h2-h,e2
ideal J=e,h-a;
J=std(J);
// print a compact presentation of J
print(matrix(J));
↦ h+(-a),e
poly z=4*e*f+h^2-2*h;
// z is the central element of U(sl_2)
NF(z,I);  // the central character of I:
↦ 3
NF(z,J); // the central character of J:
↦ (a2+2a)
poly nz = z - NF(z,J); // nz will belong to J
NF(nz,J);
↦ 0
```

See also Section 6.2.1 [ideal (plural)], page 152; Section 6.2.3 [module (plural)], page 159; Section 6.3.28 [std (plural)], page 191.

6.3.28 std (plural)

Syntax: `std ( ideal_expression)`
`std ( module_expression)`
`std ( ideal_expression, poly_expression )`
`std ( module_expression, vector_expression )`

Type: ideal or module

Purpose: returns a left Groebner basis (see Section 6.4.34 [Groebner bases in G-algebras], page 196 for a definition) of an ideal or module with respect to the monomial ordering of the basering.

Use an optional second argument of type poly, resp. vector, to construct the Groebner basis from an already computed one (given as the first argument) and one additional generator (the second argument).

Note: To view the progress of long running computations, use `option(prot)`. (see Section 4.1.91 [option], page 206(prot)).

Example:

```
LIB "ncalg.lib";
def R = makeUsl2();
// this algebra is U(sl_2)
setring R;
ideal I = e2, f2, h2-1;
I=std(I);
I;
↦ I[1]=h2-1
↦ I[2]=fh-f
↦ I[3]=eh+e
↦ I[4]=f2
↦ I[5]=2ef-h-1
↦ I[6]=e2
kill R;
//------------------------------------------
def RQ = makeQso3(3);
// this algebra is U'_q(so_3),
// where Q is a 6th root of unity
setring RQ;
RQ;
↦ //   characteristic : 0
↦ //   1 parameter    : Q
↦ //   minpoly        : (Q2-Q+1)
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    x y z
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    yx=(Q-1)*xy+(-Q)*z
```

```
↦ //    zx=(-Q)*xz+(-Q+1)*y
↦ //    zy=(Q-1)*yz+(-Q)*x
ideal J=x2, y2, z2;
J=std(J);
J;
↦ J[1]=z
↦ J[2]=y
↦ J[3]=x
```

See also Section 6.2.1 [ideal (plural)], page 152; Section 6.2.7 [ring (plural)], page 167.

6.3.29 subst (plural)

Syntax: `subst ( poly_expression,ring_variable, poly_expression )`
`subst ( vector _expression,ring_variable, poly_expression )`
`subst ( ideal_expression,ring_variable, poly_expression )`
`subst ( module _expression,ring_variable, poly_expression )`

Type: poly, vector, ideal or module (corresponding to the first argument)

Purpose: substitutes a ring variable by a polynomial.

Example:

```
LIB "ncalg.lib";
def R = makeUsl2();
// this algebra is U(sl_2)
setring R;
poly C  = e*f*h;
poly C1 = subst(C,e,h^3);
C1;
↦ fh4-6fh3+12fh2-8fh
poly C2 = subst(C,f,e+f);
C2;
↦ e2h+efh
```

6.3.30 syz (plural)

Syntax: `syz ( ideal_expression )`
`syz ( module_expression )`

Type: module

Purpose: computes the first syzygy (i.e., the module of relations of the given generators) of the ideal, resp. module.

Note: if `S` is a matrix of a left syzygy module of left submodule given by matrix `M`, then `transpose(S)*transpose(M) = 0`.

Example:

```
LIB "ncalg.lib";
def R = makeQso3(3);
setring R;
```

```
option(redSB);
 // we wish to have completely reduced bases:
option(redTail);
ideal tst;
ideal J = x3+x,x*y*z;
print(syz(J));
↦ -yz,
↦ x2+1
ideal K = x+y+z,y+z,z;
module S = syz(K);
print(S);
↦ (Q-1),       (-Q+1)*z,   (Q-1)*y,
↦ (Q)*z+(-Q+1),(Q-1)*z+(Q),(Q)*x+(-Q+1)*y,
↦ y+(-Q)*z,    x+(-Q),     (-Q)*x-1
tst = ideal(transpose(S)*transpose(K));
// check the property of a syzygy module (tst=0):
size(tst);
↦ 0
 // now compute the Groebner basis of K ...
K = std(K);
// ... print a matrix presentation of K ...
print(matrix(K));
↦ z,y,x
S = syz(K); // ... and its syzygy module
print(S);
↦ y,    (-Q)*yz+(Q)*x,y2+1,
↦ (Q)*z,-z2-1,        (Q)*yz+(-Q)*x,
↦ (Q-1),0,            0
tst = ideal(transpose(S)*transpose(K));
// check the property of a syzygy module (tst=0):
size(tst);
↦ 0
// but the "commutative" syzygy property does not hold
size(ideal(matrix(K)*matrix(S)));
↦ 1
```

See also Section 6.2.1 [ideal (plural)], page 152; Section 6.3.18 [minres (plural)], page 179; Section 6.2.3 [module (plural)], page 159; Section 6.3.20 [mres (plural)], page 180; Section 6.3.22 [nres (plural)], page 184.

6.3.31 twostd

Syntax: `twostd(` ideal_expression`);`

Type: ideal or module

Purpose: returns a left Groebner basis of the two-sided ideal, generated by the input, treated as a set of two-sided generators. see Section 4.1.125 [std], page 240

Remark: There are algebras with no two-sided ideals except 0 and the whole algebra (like Weyl algebras).

Example:

```
LIB "ncalg.lib";
def U = makeUsl2(); // this algebra is U(sl_2)
setring U;
ideal i= e^3, f^3, h^3 - 4*h;
option(redSB);
option(redTail);
ideal I = std(i);
// print a compact presentation of I:
print(matrix(I));
↦ h3-4h,fh2-2fh,eh2+2eh,2efh-h2-2h,f3,e3
ideal J = twostd(i);
// print a compact presentation of J:
print(matrix(J));
↦ h3-4h,fh2-2fh,eh2+2eh,f2h-2f2,2efh-h2-2h, \
             e2h+2e2, f3,ef2-fh,e2f-eh-2e,e3
// compute the set of elements present in J but not in I
ideal K = NF(J,I);
K = K+0; // simplify K
print(matrix(K));
↦ f2h-2f2,e2h+2e2,ef2-fh,e2f-eh-2e
```

6.3.32 vdim (plural)

Syntax: `vdim` (ideal_expression)
`vdim` (module_expression)

Type: int

Purpose: computes the vector space dimension of the factor-module that equals ring (resp. free module) modulo the ideal (resp. submodule), generated by the leading terms of the given generators.
If the factor-module is not of finite dimension, -1 is returned.

If the generators form a Groebner basis, this is the same as the vector space dimension of the factor-module.

Note: In the noncommutative case, a ring modulo an ideal has a ring stucture if and only if the ideal is two-sided.

Example:

```
ring R=0,(x,y,z),dp;
matrix d[3][3];
d[1,2]=-z;  d[1,3]=2x;  d[2,3]=-2y;
ncalgebra(1,d); //U(sl_2)
option(redSB); option(redTail);
ideal I=x3,y3,z3-z;
I=std(I);
```

```
I;
↦ I[1]=z3-z
↦ I[2]=y3
↦ I[3]=x3
↦ I[4]=y2z2-y2z
↦ I[5]=x2z2+x2z
↦ I[6]=x2y2z-2xyz2-2xyz+2z2+2z
vdim(I);
↦ 21
```

See also Section 6.2.1 [ideal (plural)], page 152; Section 6.3.15 [kbase (plural)], page 176; Section 6.3.28 [std (plural)], page 191.

6.4 Mathematical background (plural)

This section introduces some of the mathematical notions and definitions used throughout the PLURAL manual. For details, please, refer to appropriate articles or text books (see Section 6.4.36 [References (plural)], page 199). A detailed discussion of the subjects in this section can be found in the doctoral thesis [LV] of V. Levandovskyy (see Section 6.4.36 [References (plural)], page 199).

All algebras are assumed to be associative K -algebras for some field K .

6.4.33 G-algebras

Definition (PBW basis)

Let K be a field, and let a K-algebra A be generated by variables $x_1, \ldots, x_n$ subject to some relations. We call A an algebra with **PBW basis** (Poincaré-Birkhoff-Witt basis), if a K–basis of A is $\mathrm{Mon}(x_1, \ldots, x_n) = \{x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n} \mid a_i \in N \cup \{0\}\}$, where a power-product $x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n}$ (in this particular order) is called **a monomial**. For example, $x_1 x_2$ is a monomial, while $x_2 x_1$ is, in general, not a monomial.

Definition (G-algebra)

Let K be a field, and let a K-algebra A be given in terms of generators subject to the following relations:

$A = K\langle x_1, \ldots, x_n \mid \{x_j x_i = c_{ij} \cdot x_i x_j + d_{ij}\}, 1 \leq i < j \leq n\rangle$, where $c_{ij} \in K^*, d_{ij} \in K[x_1, \ldots, x_n]$.

A is called **a G–algebra, if the following conditions hold:**

- there is a monomial well-ordering ¡ such that $\forall i < j$ $\mathrm{LM}(d_{ij}) < x_i x_j$,
- **non-degeneracy conditions**: $\forall\ 1 \leq i < j < k \leq n\ :\ \mathcal{NDC}_{ijk} = 0$, where

$$\mathcal{NDC}_{ijk} = c_{ik}c_{jk} \cdot d_{ij}x_k - x_k d_{ij} + c_{jk} \cdot x_j d_{ik} - c_{ij} \cdot d_{ik}x_j + d_{jk}x_i - c_{ij}c_{ik} \cdot x_i d_{jk}.$$

Theorem (properties of G-algebras)

Let A be a G-algebra. Then

A has a PBW (Poincaré-Birkhoff-Witt) basis,

A is left and right noetherian,

A is an integral domain.

Setting up a G-algebra

In order to set up a G–algebra in Plural, one has to do the following steps:

define a commutative ring $R = K[x_1, \ldots, x_n]$, equipped with a global monomial ordering $<$ (see Section 6.2.7.1 [ring declarations (plural)], page 168).
This provides us with the information on a field K (together with its parameters), variables $\{x_i\}$ and an ordering `<`.
From the sequence of variables we will build a G-algebra with the PBW basis $\{x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n}\}$.

Define strictly $n \times n$ upper triangular matrices (of type `matrix`)

1. $C = \{c_{ij}, i < j\}$, with nonzero entries c_{ij} of type number (c_{ij} for $i \geq j$ will be ignored).
2. $D = \{d_{ij}, i < j\}$, with polynomial entries d_{ij} from R (d_{ij} for $i \geq j$ will be ignored).

Call the initialization function `ncalgebra(C,D)` (see Section 6.3.21 [ncalgebra], page 182) with the data C and D.

At present, Plural does not check automatically whether the non-degeneracy conditions hold but it provides a procedure Section 6.5.43.3 [ndcond], page 229 from the library Section 6.5.43 [nctools_lib], page 226 to check this.

6.4.34 Groebner bases in G-algebras

We follow the notations, used in the Singular Manual (e.g. in Section C.1 [Standard bases], page 261).

For a G–algebra A, we denote by ${}_A\langle g_1, \ldots, g_s\rangle$ the left submodule of a free module A^r, generated by elements $\{g_1, \ldots, g_s\} \subset A^r$.

Let $<$ be a fixed monomial well-ordering on the G–algebra A with the PBW basis $\{x^\alpha = x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n}\}$. For a given free module A^r with the basis $\{e_1, \ldots, e_r\}$, $<$ denotes also a fixed module ordering on the set of monomials $\{x^\alpha e_i \mid \alpha \in \mathbf{N}^n, 1 \leq i \leq r\}$.

Definition

For a set $S \subset A^r$, define $L(S)$ to be the K–vector space, spanned on the leading monomials of elements of S, $L(S) = \oplus\{Kx^\alpha e_i \mid \exists s \in S, \mathrm{LM(s)} = x^\alpha e_i\}$. We call $L(S)$ the **span of leading monomials of S.**

Let $I \subset A^r$ be a left A–submodule. A finite set $G \subset I$ is called **a left Groebner basis** of I if and only if $L(G) = L(I)$, that is for any $f \in I \setminus \{0\}$ there exists a $g \in G$ satisfying $\mathrm{LM}(g) \mid \mathrm{LM}(f)$, i.e., if $\mathrm{LM}(f) = x^\alpha e_i$, then $\mathrm{LM}(f) = x^\beta e_i$ with $\beta_j \leq \alpha_j$, $1 \leq j \leq n$.

Remark: In the non-commutative case we are working with well ordering only (see Section 6.1 [Getting started with PLURAL], page 151, Section B.2 [Monomial orderings], page 254 and Section 2.3.3 [Term orderings], page 37).

A Groebner basis $G \subset A^r$ is called **minimal** (or **reduced**) if $0 \notin G$ and if $\mathrm{LM}(g) \notin L(G \setminus \{g\})$ for all $g \in G$. Note, that any Groebner basis can be made minimal by deleting successively those g with $\mathrm{LM}(h) \mid \mathrm{LM}(g)$ for some $h \in G \setminus \{g\}$.

For $f \in A^r$ and $G \subset A^r$ we say that f is **completely reduced with respect to** G if no monomial of f is contained in $L(G)$.

Left Normal Form

A map $\mathrm{NF} : A^r \times \{G \mid G \text{ a (left) Groebner basis}\} \to A^r, (f|G) \mapsto \mathrm{NF}(f|G)$, is called a **(left) normal form** on A^r if for any $f \in A^r$ and any left Groebner basis G the following holds:

(i) $\mathrm{NF}(0|G) = 0$,

(ii) if $\mathrm{NF}(f|G) \neq 0$ then $\mathrm{LM}(g)$ does not divide $\mathrm{LM}(\mathrm{NF}(f|G))$ for all $g \in G$,

(iii) $f - \mathrm{NF}(f|G) \in {}_A\langle G\rangle$.

$\mathrm{NF}(f|G)$ is called a **left normal form of** f **with respect to** G (note that such a map is not unique).

Remark: As we have already mentioned in the definitions `ideal` and `module` (see Section 6.1 [Getting started with PLURAL], page 151), Plural works with left normal form only.

Left ideal membership

For a left Groebner basis G of I the following holds: $f \in I$ if and only if the left normal form $\mathrm{NF}(f|G) = 0$.

6.4.35 Syzygies and resolutions (plural)

Syzygies

Let K be a field and $<$ a well ordering on $A^r = \oplus_{i=1}^r Ae_i$. A **left** (resp. **right**) **syzygy** between k elements $\{f_1, \ldots, f_k\} \subset A^r$ is a k-tuple $(g_1, \ldots, g_k) \in A^k$ satisfying

$$\sum_{i=1}^{k} g_i f_i = 0 \quad \text{resp.} \quad \sum_{i=1}^{k} f_i g_i = 0.$$

The set of all left (resp. right) syzygies between $\{f_1, ..., f_k\}$ is a left (resp. right) submodule S of A^k.

Remark: With respect to the definitions of `ideal` and `module` (see Section 6.1 [Getting started with PLURAL], page 151), PLURAL works with left syzygies only (by `syz` we understand a left syzygy). If `S` is a matrix of a left syzygy module of left submodule given by matrix `M`, then `transpose(S)*transpose(M) = 0` (but, in general, $M \cdot S \neq 0$).

Note, that the syzygy modules of I depend on a choice of generators $\{g_1, \ldots, g_s\}$, but one can show that they depend on I uniquely up to direct summands.

Free resolutions

Let $I = {}_A\langle g_1, \ldots, g_s\rangle \subseteq A^r$ and $M = A^r/I$. A **free resolution of** M is a long exact sequence

$$\ldots \longrightarrow F_2 \xrightarrow{B_2} F_1 \xrightarrow{B_1} F_0 \longrightarrow M \longrightarrow 0,$$

with $\texttt{transpose}(B_{i+1}) \cdot \texttt{transpose}(B_i) = 0$

and where the columns of the matrix B_1 generate I . Note, that resolutions over factor-algebras need not to be of finite length.

Generalized Hilbert Syzygy Theorem

For a G–algebra A, generated by n variables, there exists a free resolution of length smaller or equal than n .

Example:

```
ring R=0,(x,y,z),dp;
matrix d[3][3];
d[1,2]=-z;  d[1,3]=2x;  d[2,3]=-2y;
ncalgebra(1,d); // this algebra is U(sl_2)
option(redSB); option(redTail);
ideal I=x3,y3,z3-z;
I=std(I);
I;
↦ I[1]=z3-z
↦ I[2]=y3
↦ I[3]=x3
↦ I[4]=y2z2-y2z
↦ I[5]=x2z2+x2z
↦ I[6]=x2y2z-2xyz2-2xyz+2z2+2z
resolution resI = mres(I,0);
resI;
↦  1      5      7      3
↦ R <--  R <--  R <--  R
↦
```

```
↦ 0      1      2      3
↦
// The matrix A_1 is given by
print(matrix(resI[1]));
↦ z3-z,y3,x3,y2z2-y2z,x2z2+x2z
// We see that the columns of A_1 generate I.
// The matrix A_2 is given by
print(matrix(resI[2]));
↦ 0, 0, y2, x2, 6yz, -36xy+18z+24,-6xz,
↦ z2+11z+30,0, 0, 0, 2x2z+12x2, 2x3, 0,
↦ 0, z2-11z+30,0, 0, 0,-2y3, 2y2z-12y2,
↦ -y, 0, -z-5, 0, x2y-6xz-30x,  9x2, x3,
↦ 0, -x, 0, -z+5,-y3, -9y2, -xy2-4yz+28y
// now, let us show that the resolution is exact
ideal tst;
matrix TST;
// the 2nd term ...
TST = transpose(resI[3])*transpose(resI[2]);
tst = std(ideal(TST));
tst;
↦ tst[1]=0
// the 1st term ...
TST = transpose(resI[2])*transpose(resI[1]);
tst = std(ideal(TST));
tst;
↦ tst[1]=0
```

6.4.36 References (plural)

The Centre for Computer Algebra Kaiserslautern publishes a series of preprints which are electronically available at `http://www.mathematik.uni-kl.de/~zca/Reports_on_ca`. Other sources to check are the following books and articles:

Text books

- Y. Drozd and V. Kirichenko. Finite dimensional algebras. With an appendix by Vlastimil Dlab. Springer, 1994
- [GPS] Greuel, G.-M. and Pfister, G. with contributions by Bachmann, O. ; Lossen, C. and Schönemann, H. A SINGULAR Introduction to Commutative Algebra. Springer, 2002
- [BGV] Bueso, J.; Gomez Torrecillas, J.; Verschoren, A. Algorithmic methods in non-commutative algebra. Applications to quantum groups. Kluwer Academic Publishers, 2003
- Kredel, H. Solvable polynomial rings. Shaker, 1993
- [Li] Huishi Li. Noncommutative Gröbner bases and filtered-graded transfer. Springer, 2002

- [MR] 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

Descriptions of algorithms and problems

- 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)
- J. Apel. Gröbnerbasen in nichtkommutativen algebren und ihre anwendung. Dissertation, Universität Leipzig, 1988.
- Apel, J. Computational ideal theory in finitely generated extension rings. Theor. Comput. Sci.(2000), 244(1-2):1-33
- O. Bachmann and H. Schönemann. Monomial operations for computations of Gröbner bases. In Reports On Computer Algebra 18. Centre for Computer Algebra, University of Kaiserslautern (1998)
- D. Decker and D. Eisenbud. Sheaf algorithms using the exterior algebra. In Eisenbud, D.; Grayson, D.; Stillman, M.; Sturmfels, B., editor, Computations in algebraic geometry with Macaulay 2, (2001)
- Jose L. Bueso, J. Gomez Torrecillas and F. J. Lobillo. Computing the Gelfand-Kirillov dimension II. In A. Granja, J. A. Hermida and A. Verschoren eds. Ring Theory and Algebraic Geometry, Lect. Not. in Pure and Appl. Maths., Marcel Dekker, 2001.
- Jose L. Bueso, J. Gomez Torrecillas and F. J. Lobillo. Re-filtering and exactness of the Gelfand-Kirillov dimension. Bulletin des Sciences Mathematiques 125(8), 689-715 (2001).
- J. Gomez Torrecillas and F.J. Lobillo. Global homological dimension of multifiltered rings and quantized enveloping algebras. J. Algebra, 225(2):522-533, 2000.
- N. Iorgov. On the Center of q-Deformed Algebra $U'_q(\mathrm{so}_3)$ Related to Quantum Gravity at q a Root of 1. In Proceedings of IV Int. Conf. "Symmetry in Nonlinear Mathematical Physics",(2001) Kyiv, Ukraine
- A. Kandri-Rody and V. Weispfenning. Non-commutative Gröbner bases in algebras of solvable type. J. Symbolic Computation, 9(1):1-26, 1990.
- Levandovskyy, V. On Gröbner bases for non-commutative G-algebras. In Kredel, H. and Seiler, W.K., editor, Proceedings of the 8th Rhine Workshop on Computer Algebra, 2002.
- [L1] Levandovskyy, V. PBW Bases, Non-degeneracy Conditions and Applications. In Buchweitz, R.-O. and Lenzing, H., editor, Proceedings of the ICRA X conference, Toronto, 2003.
- [LS] 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.

- [LV] Levandovskyy, V. Non-commutative Computer Algebra for polynomial algebras: Gröbner bases, applications and implementation. Doctoral Thesis, Universität Kaiserslautern, 2005.
- [L2] Levandovskyy, V. On preimages of ideals in certain non-commutative algebras. In Pfister G., Cojocaru S. and Ufnarovski, V. (editors), Computational Commutative and Non-Commutative Algebraic Geometry, IOS Press, 2005.
- Mora, T. Gröbner bases for non-commutative polynomial rings. Proc. AAECC 3 Lect. N. Comp. Sci, 229: 353-362, 1986.
- Mora, T. An introduction to commutative and non-commutative Groebner bases. Theor. Comp. Sci., 134: 131-173, 1994.
- T. Nüßler and H. Schönemann. Gröbner bases in algebras with zero-divisors. Preprint 244, Universität Kaiserslautern, 1993.
- Ringel, C. M. PBW-bases of quantum groups. J. Reine Angew. Math., 470:51-88, 1996.
- Schönemann, H. Singular in a Framework for Polynomial Computations. In Joswig, M. and Takayama, N., editor, Algebra, Geometry and Software Systems, pages 163-176. Springer, 2003.
- T. Yan. The geobucket data structure for polynomials. J. Symbolic Computation, 25(3):285-294, March 1998.

6.5 PLURAL libraries

PLURAL comes with a set of standard libraries. Their content is described in the following subsections.

Use the `LIB` command for loading of single libraries.

Note: For any computation in PLURAL, the monomial ordering must be a global ordering.

6.5.37 center_lib

Library: center.lib

Purpose: computation of central elements of G-algebras and their factor-algebras.

Author: Oleksandr Motsak, motsak@mathematik.uni-kl.de.

Overview: This is a library for computing the central elements and centralizers of elements in various noncommutative algebras. Implementation is based on algorithms, written in the frame of the diploma thesis by O. Motsak (advisor: Prof. S.A. Ovsienko, support: V. Levandovskyy), at Kyiv Taras Shevchenko University (Ukraine) with the title 'An algorithm for the computation of the center of noncommutative polynomial algebra'.

Support: Forschungsschwerpunkt 'Mathematik und Praxis', University of Kaiserslautern

Procedures:

6.5.37.1 center

Procedure from library `center.lib` (see Section 6.5.37 [center_lib], page 201).

Return: ideal, generated by elements of degree at most MaxDeg

Purpose: computes a minimal set of central elements up to degree MaxDeg.

Note: In general, one cannot predict the number or the heighest degree of central elements. Hence, one has to specify a termination condition via arguments MaxDeg and/or N.
If MaxDeg is positive, the computation stops after all central elements of degree at most MaxDeg has been found.
If MaxDeg is negative, the termination is determined by N only.
If N is given, the computation stops if at least N central elements has been found.
Warning: if N is given and bigger than the real number of generators, the procedure may not terminate.

Example:

```
LIB "center.lib";
ring A = 0,(x,y,z,t),dp;
matrix D[4][4]=0;
D[1,2]=-z;  D[1,3]=2*x;  D[2,3]=-2*y;
ncalgebra(1,D); // this algebra is U(sl_2) tensored with K[t]
ideal Z = center(3); // find all central elements of degree <= 3
Z;
↦ Z[1]=t
↦ Z[2]=4xy+z2-2z
inCenter(Z);
↦ 1
// find the generator of the center of the lowest degree
ideal ZZ = center(-1, 1);
ZZ;
↦ ZZ[1]=t
inCenter(ZZ);
↦ 1
```

Section 6.5.37.2 [centralizer], page 202, Section 6.5.37.3 [inCenter], page 204

6.5.37.2 centralizer

Procedure from library `center.lib` (see Section 6.5.37 [center_lib], page 201).

Return: ideal, generated by elements of degree <= MaxDeg

Purpose: computes a minimal set of elements centralizer(S) up to degree MaxDeg.

Note: In general, one cannot predict the number or the heighest degree of centralizing elements. Hence, one has to specify a termination condition via arguments MaxDeg and/or N.

If MaxDeg is positive, the computation stops after all centralizing elements of degree at most MaxDeg has been found.
If MaxDeg is negative, the termination is determined by N only.
If N is given, the computation stops if at least N centralizing elements has been found.
Warning: if N is given and bigger than the real number of generators, the procedure may not terminate.

Example:

```
LIB "center.lib";
ring A = 0,(x,y,z),dp;
matrix D[3][3]=0;
D[1,2]=-z; D[1,3]=2*x; D[2,3]=-2*y;
ncalgebra(1,D); // this algebra is U(sl_2)
poly f = 4*x*y+z^2-2*z; // a central polynomial
f;
↦ 4xy+z2-2z
// find generators of the centralizer of f:
ideal c = centralizer(f, 2);
// of degree <= 2
c;  // since f is central, these are the generators of A
↦ c[1]=z
↦ c[2]=y
↦ c[3]=x
inCentralizer(c, f);
↦ 1
// find at least two generators of the centralizer of f:
ideal cc = centralizer(f,-1,2);
cc;
↦ cc[1]=z
↦ cc[2]=y
↦ cc[3]=x
inCentralizer(cc, f);
↦ 1
poly g = z^2-2*z; // some non-central polynomial
// find all elements of the centralizer of g:
// of degree <= 2
c = centralizer(g, 2);
c;
↦ c[1]=z
↦ c[2]=xy
inCentralizer(c, g);
↦ 1
 // find the element of the lowest degree in the centralizer:
centralizer(g,-1,1);
↦ _[1]=z
// find at least two elements of the centralizer of g:
cc;
```

```
cc = centralizer(g,-1,2);
↦ cc[1]=z
↦ cc[2]=xy
inCentralizer(cc, g);
↦ 1
```

Section 6.5.37.1 [center], page 202, Section 6.5.37.4 [inCentralizer], page 204

6.5.37.3 inCenter

Procedure from library `center.lib` (see Section 6.5.37 [center_lib], page 201).

Return: integer, 1 if a in the center, 0 otherwise

Purpose: check whether a given element is central

Example:

```
LIB "center.lib";
ring r=0,(x,y,z),dp;
matrix D[3][3]=0;
D[1,2]=-z;
D[1,3]=2*x;
D[2,3]=-2*y;
ncalgebra(1,D); // this is U(sl_2)
poly p=4*x*y+z^2-2*z;
inCenter(p);
↦ 1
poly f=4*x*y;
inCenter(f);
↦ 0
list l= list( 1, p, p^2, p^3);
inCenter(l);
↦ 1
ideal I= p, f;
inCenter(I);
↦ 0
```

6.5.37.4 inCentralizer

Procedure from library `center.lib` (see Section 6.5.37 [center_lib], page 201).

Return: integer, 1 if a in the centralizer(S), 0 otherwise

Purpose: check whether a given element is centralizing with respect to elements of S

Example:

```
LIB "center.lib";
ring r=0,(x,y,z),dp;
matrix D[3][3]=0;
D[1,2]=-z;
ncalgebra(1,D); // the Heisenberg algebra
```

```
poly f = x^2;
poly a = z; // we know this element if central
poly b = y^2;
inCentralizer(a, f);
↦ 1
inCentralizer(b, f);
↦ 0
list  l = list(1, a);
inCentralizer(l, f);
↦ 1
ideal I = a, b;
inCentralizer(I, f);
↦ 0
```

6.5.37.5 isCartan

Procedure from library `center.lib` (see Section 6.5.37 [center_lib], page 201).

Purpose: check whether f is Cartan's element

Return: 1 if f is Cartan's element and 0 otherwise.

Note: f is Cartan's element <=> $\forall g \in A, \exists \alpha \in K$ such that $[f, g] = \alpha * g$. <=> $\forall v_i$ (variables of A), $\exists \alpha \in K$ such that $[f, v_i] = \alpha * v_i$.

Example:

```
LIB "center.lib";
ring r=0,(x,y,z),dp;
matrix D[3][3]=0;
D[1,2]=-z;
D[1,3]=2*x;
D[2,3]=-2*y;
ncalgebra(1,D); // this is U(sl_2) with cartan - z
isCartan(z); // yes!
↦ 1
poly p=4*x*y+z^2-2*z;
isCartan(p); // central elements are cartans!
↦ 1
poly f=4*x*y;
isCartan(f); // no way!
↦ 0
isCartan( 10 + p + z ); // scalar + central + cartan
↦ 1
```

6.5.37.6 sa_reduce

Procedure from library `center.lib` (see Section 6.5.37 [center_lib], page 201).

Purpose: subalgebra reduction of a set of pairwise commuting polynomials.

6.5.37.7 sa_poly_reduce

Procedure from library `center.lib` (see Section 6.5.37 [center_lib], page 201).

Purpose: subalgebra reduction of a given polynomial f wrt a set of pairwise commuting polynomials.

6.5.38 involut_lib

Library: involut.lib

Purpose: Procedures for Computations and Operations with Involutions

Authors: Oleksandr Iena, yena@mathematik.uni-kl.de,
Markus Becker, mbecker@mathematik.uni-kl.de,
Viktor Levandovskyy, levandov@mathematik.uni-kl.de

Theory: Involution is an antiisomorphism of a noncommutative algebra with the property that applied an involution twice, one gets an identity. Involution is linear with respect to the ground field. In this library we compute linear involutions, distinguishing the case of a diagonal matrix (such involutions are called homothetic) and a general one.

Support: Forschungsschwerpunkt 'Mathematik und Praxis' (Project of Dr. E. Zerz and V. Levandovskyy), Uni Kaiserslautern

Note: This library provides algebraic tools for computations and operations with algebraic involutions and linear automorphisms of noncommutative algebras

Procedures:

6.5.38.1 findInvo

Procedure from library `involut.lib` (see Section 6.5.38 [involut_lib], page 206).

Usage: findInvo();

Return: a ring containing a list L of pairs, where
L[i][1] = Groebner Basis of an i-th associated prime,
L[i][2] = matrix, defining a linear map, with entries, reduced with respect to L[i][1]

Purpose: computed the ideal of linear involutions of the basering

Note: for convenience, the full ideal of relations `idJ` and the initial matrix with indeterminates `matD` are exported in the output ring

Example:

```
LIB "involut.lib";
def a = makeWeyl(1);
setring a; // this algebra is a first Weyl algebra
def X = findInvo();
```

```
setring X;
// ring with new variables,
// which correspond to unknown coefficients
L;
↦ [1]:
↦    [1]:
↦       _[1]=a11+a22
↦       _[2]=a12*a21+a22^2-1
↦    [2]:
↦       _[1,1]=-a22
↦       _[1,2]=a12
↦       _[2,1]=a21
↦       _[2,2]=a22
// look at the matrix in the new variables,
// defining the linear involution
print(L[1][2]);
↦ -a22,a12,
↦ a21, a22
L[1][1];  // where new variables obey these relations
↦ _[1]=a11+a22
↦ _[2]=a12*a21+a22^2-1
```

Section 6.5.38.2 [findInvoDiag], page 207, Section 6.5.38.5 [involution], page 210

6.5.38.2 findInvoDiag

Procedure from library `involut.lib` (see Section 6.5.38 [involut_lib], page 206).

Usage: findInvoDiag();

Return: a ring together with a list of pairs L, where
L[i][1] = Groebner Basis of an i-th associated prime,
L[i][2] = matrix, defining a linear map, with entries, reduced with respect to L[i][1]

Purpose: compute the ideal of homothetic (diagonal) involutions of the basering

Note: for convenience, the full ideal of relations `idJ` and the initial matrix with indeterminates `matD` are exported in the output ring

Example:

```
LIB "involut.lib";
def a = makeWeyl(1);
setring a; // this algebra is a first Weyl algebra
def X = findInvoDiag();
setring X; // ring with new variables,
// which correspond to unknown coefficients
// print matrices, defining linear involutions
print(L[1][2]);  // a first matrix: we see it is constant
↦ -1,0,
↦ 0, 1
```

```
print(L[2][2]);  // a second possible matrix; it is constant too
↦ 1,0,
↦ 0,-1
L; // let us take a look on the whole list
↦ [1]:
↦    [1]:
↦       _[1]=a22-1
↦       _[2]=a11+1
↦    [2]:
↦       _[1,1]=-1
↦       _[1,2]=0
↦       _[2,1]=0
↦       _[2,2]=1
↦ [2]:
↦    [1]:
↦       _[1]=a22+1
↦       _[2]=a11-1
↦    [2]:
↦       _[1,1]=1
↦       _[1,2]=0
↦       _[2,1]=0
↦       _[2,2]=-1
```

Section 6.5.38.1 [findInvo], page 206, Section 6.5.38.5 [involution], page 210

6.5.38.3 findAuto

Procedure from library `involut.lib` (see Section 6.5.38 [involut_lib], page 206).

Usage: findAuto(n); n an integer

Return: a ring together with a list of pairs L, where
L[i][1] = Groebner Basis of an i-th associated prime,
L[i][2] = matrix, defining a linear map, with entries, reduced with respect to L[i][1]

Purpose: computes the ideal of linear automorphisms of the basering, given by a matrix, n-th power of which gives identity (i.e. unipotent matrix)

Note: if n=0, a matrix, defining an automorphism is not assumed to be unipotent. For convenience, the full ideal of relations `idJ` and the initial matrix with indeterminates `matD` are exported in the output ring

Example:

```
LIB "involut.lib";
def a = makeWeyl(1);
setring a; // this algebra is a first Weyl algebra
def X = findAuto(2);
setring X; // ring with new variables - unknown coefficients
// look at matrices, defining linear automorphisms:
print(L[1][2]);  // a first one: we see it is constant
```

```
↦ 1,0,
↦ 0,1
print(L[2][2]);  // a second possible matrix; it is constant too
↦ -1,0,
↦ 0, -1
L; // let us take a look on the whole list
↦ [1]:
↦    [1]:
↦       _[1]=a22-1
↦       _[2]=a21
↦       _[3]=a12
↦       _[4]=a11-1
↦    [2]:
↦       _[1,1]=1
↦       _[1,2]=0
↦       _[2,1]=0
↦       _[2,2]=1
↦ [2]:
↦    [1]:
↦       _[1]=a22+1
↦       _[2]=a21
↦       _[3]=a12
↦       _[4]=a11+1
↦    [2]:
↦       _[1,1]=-1
↦       _[1,2]=0
↦       _[2,1]=0
↦       _[2,2]=-1
```

Section 6.5.38.1 [findInvo], page 206

6.5.38.4 ncdetection

Procedure from library `involut.lib` (see Section 6.5.38 [involut_lib], page 206).

Usage: ncdetection();

Return: ideal, representing an involution map

Purpose: compute classical involutions (i.e. acting rather on operators than on variables) for some particular noncommutative algebras

Assume: the procedure is aimed at noncommutative algebras with differential, shift or advance operators arising in Control Theory. It has to be executed in the ring.

Example:

```
LIB "involut.lib";
ring r=0,(x,y,z,D(1..3)),dp;
matrix D[6][6];
D[1,4]=1; D[2,5]=1;  D[3,6]=1;
```

```
ncalgebra(1,D);
ncdetection();
↦ _[1]=x
↦ _[2]=y
↦ _[3]=z
↦ _[4]=-D(1)
↦ _[5]=-D(2)
↦ _[6]=-D(3)
kill r;
//----------------------------------------
ring r=0,(x,S),dp;
ncalgebra(1,-S);
ncdetection();
↦ _[1]=-x
↦ _[2]=S
kill r;
//----------------------------------------
ring r=0,(x,D(1),S),dp;
matrix D[3][3];
D[1,2]=1;  D[1,3]=-S;
ncalgebra(1,D);
ncdetection();
↦ _[1]=-x
↦ _[2]=D(1)
↦ _[3]=S
```

6.5.38.5 involution

Procedure from library `involut.lib` (see Section 6.5.38 [involut_lib], page 206).

Usage: involution(m, theta); m is a poly/vector/ideal/matrix/module, theta is a map

Return: object of the same type as m

Purpose: applies the involution, presented by theta to the object m

Theory: for an involution theta and two polynomials a,b from the algebra, theta(ab) = theta(b) theta(a); theta is linear with respect to the ground field

Example:

```
LIB "involut.lib";
ring r = 0,(x,d),dp;
ncalgebra(1,1); // Weyl-Algebra
map F = r,x,-d;
poly f =  x*d^2+d;
poly If = involution(f,F);
f-If;
↦ 0
poly g = x^2*d+2*x*d+3*x+7*d;
```

```
poly tg = -d*x^2-2*d*x+3*x-7*d;
poly Ig = involution(g,F);
tg-Ig;
↦ 0
ideal I = f,g;
ideal II = involution(I,F);
II;
↦ II[1]=xd2+d
↦ II[2]=-x2d-2xd+x-7d-2
I - involution(II,F);
↦ _[1,1]=0
↦ _[1,2]=0
module M  = [f,g,0],[g,0,x^2*d];
module IM = involution(M,F);
print(IM);
↦ xd2+d,          -x2d-2xd+x-7d-2,
↦ -x2d-2xd+x-7d-2,0,
↦ 0,              -x2d-2x
print(M - involution(IM,F));
↦ 0,0,
↦ 0,0,
↦ 0,0
```

6.5.39 gkdim_lib

Library: GKdim.lib

Purpose: Procedures for calculating the Gelfand-Kirillov dimension

Authors: Lobillo, F.J., jlobillo@ugr.es
Rabelo, C., crabelo@ugr.es

Support: 'Metodos algebraicos y efectivos en grupos cuanticos', BFM2001-3141, MCYT, Jose Gomez-Torrecillas (Main researcher).

Procedures:

6.5.39.1 GKdim

Procedure from library `gkdim.lib` (see Section 6.5.39 [gkdim_lib], page 211).

Usage: GKdim(L); L is a left ideal/module/matrix

Return: int

Purpose: compute the Gelfand-Kirillov dimension of the factor-module, whose presentation is given by L

Note: if the factor-module is zero, -1 is returned

Example:

```
LIB "gkdim.lib";
ring r = 0,(x,y,z),Dp;
matrix C[3][3]=0,1,1,0,0,-1,0,0,0;
matrix D[3][3]=0,0,0,0,0,x;
ncalgebra(C,D);
r;
↦ //   characteristic : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering Dp
↦ //                  : names    x y z
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    zy=-yz+x
ideal I=x;
GKdim(I);
↦ 2
ideal J=x2,y;
GKdim(J);
↦ 1
module M=[x2,y,1],[x,y2,0];
GKdim(M);
↦ 3
ideal A = x,y,z;
GKdim(A);
↦ 0
ideal B = 1;
GKdim(B);
↦ -1
```

6.5.40 ncall_lib

The library `ncall.lib` provides a convenient way to load all libraries, featuring noncommutative algorithms of the SINGULAR distribution.

Example:

```
option(loadLib);
LIB "ncall.lib";
↦ // ** loaded ncall.lib (1.4,2005/06/07)
↦ // ** loaded involut.lib (1.8,2005/06/10)
↦ // ** loaded primdec.lib (1.105,2005/05/06)
↦ // ** loaded triang.lib (1.9,2005/05/10)
↦ // ** loaded matrix.lib (1.30,2005/05/06)
↦ // ** loaded ring.lib (1.26,2005/05/18)
↦ // ** loaded inout.lib (1.25,2005/05/10)
↦ // ** loaded random.lib (1.16,2001/01/16)
↦ // ** loaded elim.lib (1.18,2005/05/18)
↦ // ** loaded general.lib (1.47,2005/05/10)
↦ // ** loaded poly.lib (1.37,2005/05/18)
↦ // ** loaded qmatrix.lib (1.12,2005/05/18)
```

```
↦ // ** loaded gkdim.lib (1.9,2005/05/09)
↦ // ** loaded nctools.lib (1.17,2005/08/12)
↦ // ** loaded ncdecomp.lib (1.11,2005/05/18)
↦ // ** loaded ncalg.lib (1.15,2005/08/12)
↦ // ** loaded toric.lib (1.11,2001/02/06)
↦ // ** loaded center.lib (1.16,2005/05/18)
```

6.5.41 ncalg_lib

Library: ncalg.lib

Purpose: Definitions of important GR-algebras

Authors: Viktor Levandovskyy, levandov@mathematik.uni-kl.de,
Oleksandr Motsak, motsak@mathematik.uni-kl.de.

Conventions:

This library provides pre-defined important noncommutative algebras.
For universal enveloping algebras of finite dimensional Lie algebras sl_n, gl_n and g_2 there are functions `makeUsl`, `makeUgl` and `makeUg2`.
There are quantized enveloping algebras U_q(sl_2) and U_q(sl_3) (via functions `makeQsl2`, `makeQsl3`)
and non-standard quantum deformation of so_3, accessible via `makeQso3` function.

Procedures:

6.5.41.1 makeUsl

Procedure from library `ncalg.lib` (see Section 6.5.41 [ncalg_lib], page 213).

Usage: makeUsl(n,[p]); n an integer, n>1; p an optional integer (field characteristic)

Return: ring

Purpose: set up the U(sl_n) in the variables (x(i),y(i),h(i) | i=1..n+1) over the field of char p

Note: activate this ring with the `setring` command
This presentation of U(sl_n) is the standard one, i.e. positive resp. negative roots are denoted by x(i) resp. y(i) and the Cartan elements are denoted by h(i).
The variables are ordered as x(1),...x(n),y(1),...,y(n),h(1),...h(n).

Example:

```
LIB "ncalg.lib";
def a = makeUsl(3);
setring a;
a;
↦ //   characteristic : 0
↦ //   number of vars : 8
```

```
↦ //     block 1: ordering dp
↦ //            : names x(1) x(2) x(3) y(1) y(2) y(3) h(1) h(2)
↦ //     block 2: ordering C
↦ //   noncommutative relations:
↦ //    x(2)x(1)=x(1)*x(2)-x(3)
↦ //    y(1)x(1)=x(1)*y(1)-h(1)
↦ //    y(3)x(1)=x(1)*y(3)+y(2)
↦ //    h(1)x(1)=x(1)*h(1)+2*x(1)
↦ //    h(2)x(1)=x(1)*h(2)-x(1)
↦ //    y(2)x(2)=x(2)*y(2)-h(2)
↦ //    y(3)x(2)=x(2)*y(3)-y(1)
↦ //    h(1)x(2)=x(2)*h(1)-x(2)
↦ //    h(2)x(2)=x(2)*h(2)+2*x(2)
↦ //    y(1)x(3)=x(3)*y(1)+x(2)
↦ //    y(2)x(3)=x(3)*y(2)-x(1)
↦ //    y(3)x(3)=x(3)*y(3)-h(1)-h(2)
↦ //    h(1)x(3)=x(3)*h(1)+x(3)
↦ //    h(2)x(3)=x(3)*h(2)+x(3)
↦ //    y(2)y(1)=y(1)*y(2)+y(3)
↦ //    h(1)y(1)=y(1)*h(1)-2*y(1)
↦ //    h(2)y(1)=y(1)*h(2)+y(1)
↦ //    h(1)y(2)=y(2)*h(1)+y(2)
↦ //    h(2)y(2)=y(2)*h(2)-2*y(2)
↦ //    h(1)y(3)=y(3)*h(1)-y(3)
↦ //    h(2)y(3)=y(3)*h(2)-y(3)
```

Section 6.5.41.2 [makeUsl2], page 214, Section 6.5.41.3 [makeUg2], page 215, Section 6.5.41.4 [makeUgl], page 216, Section 6.5.41.8 [makeQsl3], page 220, Section 6.5.41.5 [makeQso3], page 217

6.5.41.2 makeUsl2

Procedure from library `ncalg.lib` (see Section 6.5.41 [ncalg_lib], page 213).

Usage: makeUsl2([p]), p an optional integer (field characteristic)

Return: ring

Purpose: set up the U(sl_2) in the variables e,f,h over the field of char p

Note: activate this ring with the `setring` command

Example:

```
LIB "ncalg.lib";
def a=makeUsl2();
setring a;
a;
↦ //   characteristic : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    e f h
```

```
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    fe=ef-h
↦ //    he=eh+2e
↦ //    hf=fh-2f
```

Section 6.5.41.1 [makeUsl], page 213, Section 6.5.41.3 [makeUg2], page 215, Section 6.5.41.4 [makeUgl], page 216

6.5.41.3 makeUg2

Procedure from library `ncalg.lib` (see Section 6.5.41 [ncalg_lib], page 213).

Usage: makeUg2([p]), p an optional int (field characteristic)

Return: ring

Purpose: set up the U(g_2) in variables (x(i),y(i),Ha,Hb) for i=1..6 over the field of char p

Note: activate this ring with the `setring` command
the variables are ordered as x(1),...x(6),y(1),...,y(6),Ha,Hb.

Example:

```
LIB "ncalg.lib";
def a = makeUg2();
setring a;  a;
↦ //   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)
↦ //    x(3)x(1)=x(1)*x(3)-2*x(4)
↦ //    x(4)x(1)=x(1)*x(4)+3*x(5)
↦ //    y(1)x(1)=x(1)*y(1)-Ha
↦ //    y(3)x(1)=x(1)*y(3)+3*y(2)
↦ //    y(4)x(1)=x(1)*y(4)+2*y(3)
↦ //    y(5)x(1)=x(1)*y(5)-y(4)
↦ //    Hax(1)=x(1)*Ha+2*x(1)
↦ //    Hbx(1)=x(1)*Hb-x(1)
↦ //    x(5)x(2)=x(2)*x(5)+x(6)
↦ //    y(2)x(2)=x(2)*y(2)-Hb
↦ //    y(3)x(2)=x(2)*y(3)-y(1)
↦ //    y(6)x(2)=x(2)*y(6)-y(5)
↦ //    Hax(2)=x(2)*Ha-3*x(2)
↦ //    Hbx(2)=x(2)*Hb+2*x(2)
↦ //    x(4)x(3)=x(3)*x(4)+3*x(6)
↦ //    y(1)x(3)=x(3)*y(1)+3*x(2)
```

```
↦ //    y(2)x(3)=x(3)*y(2)-x(1)
↦ //    y(3)x(3)=x(3)*y(3)-Ha-3*Hb
↦ //    y(4)x(3)=x(3)*y(4)-2*y(1)
↦ //    y(6)x(3)=x(3)*y(6)-y(4)
↦ //    Hax(3)=x(3)*Ha-x(3)
↦ //    Hbx(3)=x(3)*Hb+x(3)
↦ //    y(1)x(4)=x(4)*y(1)+2*x(3)
↦ //    y(3)x(4)=x(4)*y(3)-2*x(1)
↦ //    y(4)x(4)=x(4)*y(4)-2*Ha-3*Hb
↦ //    y(5)x(4)=x(4)*y(5)+y(1)
↦ //    y(6)x(4)=x(4)*y(6)+y(3)
↦ //    Hax(4)=x(4)*Ha+x(4)
↦ //    y(1)x(5)=x(5)*y(1)-x(4)
↦ //    y(4)x(5)=x(5)*y(4)+x(1)
↦ //    y(5)x(5)=x(5)*y(5)-Ha-Hb
↦ //    y(6)x(5)=x(5)*y(6)+y(2)
↦ //    Hax(5)=x(5)*Ha+3*x(5)
↦ //    Hbx(5)=x(5)*Hb-x(5)
↦ //    y(2)x(6)=x(6)*y(2)-x(5)
↦ //    y(3)x(6)=x(6)*y(3)-x(4)
↦ //    y(4)x(6)=x(6)*y(4)+x(3)
↦ //    y(5)x(6)=x(6)*y(5)+x(2)
↦ //    y(6)x(6)=x(6)*y(6)-Ha-2*Hb
↦ //    Hbx(6)=x(6)*Hb+x(6)
↦ //    y(2)y(1)=y(1)*y(2)+y(3)
↦ //    y(3)y(1)=y(1)*y(3)+2*y(4)
↦ //    y(4)y(1)=y(1)*y(4)-3*y(5)
↦ //    Hay(1)=y(1)*Ha-2*y(1)
↦ //    Hby(1)=y(1)*Hb+y(1)
↦ //    y(5)y(2)=y(2)*y(5)-y(6)
↦ //    Hay(2)=y(2)*Ha+3*y(2)
↦ //    Hby(2)=y(2)*Hb-2*y(2)
↦ //    y(4)y(3)=y(3)*y(4)-3*y(6)
↦ //    Hay(3)=y(3)*Ha+y(3)
↦ //    Hby(3)=y(3)*Hb-y(3)
↦ //    Hay(4)=y(4)*Ha-y(4)
↦ //    Hay(5)=y(5)*Ha-3*y(5)
↦ //    Hby(5)=y(5)*Hb+y(5)
↦ //    Hby(6)=y(6)*Hb-y(6)
```

Section 6.5.41.1 [makeUsl], page 213, Section 6.5.41.4 [makeUgl], page 216

6.5.41.4 makeUgl

Procedure from library `ncalg.lib` (see Section 6.5.41 [ncalg_lib], page 213).

Usage: makeUgl(n,[p]); n an int, n>1; p an optional int (field characteristic)

Return: ring

Purpose: set up the U(gl_n) in the (e_ij (1<i,j<n)) presentation (where e_ij corresponds to a matrix with 1 at i,j only) over the field of char p

Note: activate this ring with the `setring` command
the variables are ordered as e_12,e_13,...,e_1n,e_21,...,e_nn.

Example:

```
LIB "ncalg.lib";
def a=makeUgl(3);
setring a; a;
↦ //   characteristic : 0
↦ //   number of vars : 9
↦ //        block   1 : ordering dp
↦ //                  : names    e_1_1 e_1_2 e_1_3 e_2_1 e_2_2 \
↦                       e_2_3 e_3_1 e_3_2 e_3_3
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    e_1_2e_1_1=e_1_1*e_1_2-e_1_2
↦ //    e_1_3e_1_1=e_1_1*e_1_3-e_1_3
↦ //    e_2_1e_1_1=e_1_1*e_2_1+e_2_1
↦ //    e_3_1e_1_1=e_1_1*e_3_1+e_3_1
↦ //    e_2_1e_1_2=e_1_2*e_2_1-e_1_1+e_2_2
↦ //    e_2_2e_1_2=e_1_2*e_2_2-e_1_2
↦ //    e_2_3e_1_2=e_1_2*e_2_3-e_1_3
↦ //    e_3_1e_1_2=e_1_2*e_3_1+e_3_2
↦ //    e_2_1e_1_3=e_1_3*e_2_1+e_2_3
↦ //    e_3_1e_1_3=e_1_3*e_3_1-e_1_1+e_3_3
↦ //    e_3_2e_1_3=e_1_3*e_3_2-e_1_2
↦ //    e_3_3e_1_3=e_1_3*e_3_3-e_1_3
↦ //    e_2_2e_2_1=e_2_1*e_2_2+e_2_1
↦ //    e_3_2e_2_1=e_2_1*e_3_2+e_3_1
↦ //    e_2_3e_2_2=e_2_2*e_2_3-e_2_3
↦ //    e_3_2e_2_2=e_2_2*e_3_2+e_3_2
↦ //    e_3_1e_2_3=e_2_3*e_3_1-e_2_1
↦ //    e_3_2e_2_3=e_2_3*e_3_2-e_2_2+e_3_3
↦ //    e_3_3e_2_3=e_2_3*e_3_3-e_2_3
↦ //    e_3_3e_3_1=e_3_1*e_3_3+e_3_1
↦ //    e_3_3e_3_2=e_3_2*e_3_3+e_3_2
```

Section 6.5.41.1 [makeUsl], page 213, Section 6.5.41.3 [makeUg2], page 215

6.5.41.5 makeQso3

Procedure from library `ncalg.lib` (see Section 6.5.41 [ncalg_lib], page 213).

Usage: makeQso3([n]), n an optional int

Purpose: set up the U_q(so_3) in the presentation of Klimyk; if n is specified, the quantum parameter Q will be specialized at the (2n)-th root of unity

Return: ring

Note: activate this ring with the `setring` command

Example:

```
LIB "ncalg.lib";
def K = makeQso3(3);
setring K;
K;
↦ //   characteristic : 0
↦ //   1 parameter    : Q
↦ //   minpoly        : (Q2-Q+1)
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    x y z
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    yx=(Q-1)*xy+(-Q)*z
↦ //    zx=(-Q)*xz+(-Q+1)*y
↦ //    zy=(Q-1)*yz+(-Q)*x
```

Section 6.5.41.1 [makeUsl], page 213, Section 6.5.41.3 [makeUg2], page 215, Section 6.5.41.4 [makeUgl], page 216, Section 6.5.41.7 [makeQsl2], page 219, Section 6.5.41.8 [makeQsl3], page 220, Section 6.5.41.6 [Qso3Casimir], page 218

6.5.41.6 Qso3Casimir

Procedure from library `ncalg.lib` (see Section 6.5.41 [ncalg_lib], page 213).

Usage: Qso3Casimir(n [,m]), n an integer, m an optional integer

Return: list (of polynomials)

Purpose: compute the Casimir (central) elements of U_q(so_3) for the quantum parameter specialized at the n-th root of unity; if m!=0 is given, polynomials will be normalized

Assume: the basering must be U_q(so_3)

Example:

```
LIB "ncalg.lib";
def R = makeQso3(5);
setring R;
list C = Qso3Casimir(5);
C;
↦ [1]:
↦    1/5*x5+(1/5Q3-1/5Q2+2/5)*x3+(1/5Q3-1/5Q2+1/5)*x
↦ [2]:
↦    1/5*y5+(1/5Q3-1/5Q2+2/5)*y3+(1/5Q3-1/5Q2+1/5)*y
↦ [3]:
↦    1/5*z5+(1/5Q3-1/5Q2+2/5)*z3+(1/5Q3-1/5Q2+1/5)*z
list Cnorm = Qso3Casimir(5,1);
Cnorm;
↦ [1]:
```

```
↦     x5+(Q3-Q2+2)*x3+(Q3-Q2+1)*x
↦ [2]:
↦     y5+(Q3-Q2+2)*y3+(Q3-Q2+1)*y
↦ [3]:
↦     z5+(Q3-Q2+2)*z3+(Q3-Q2+1)*z
```

Section 6.5.41.5 [makeQso3], page 217

6.5.41.7 makeQsl2

Procedure from library `ncalg.lib` (see Section 6.5.41 [ncalg_lib], page 213).

Usage: makeQsl2([n]), n an optional int

Return: ring

Purpose: define the U_q(sl_2) as a factor-ring of a ring V_q(sl_2) modulo the ideal `Qideal`

Note: the output consists of a ring, presenting V_q(sl_2) together with the ideal called `Qideal` in this ring
activate this ring with the `setring` command
in order to create the U_q(sl_2) from the output, execute the command like `qring Usl2q = Qideal;`
If n is specified, the quantum parameter q will be specialized at the n-th root of unity

Example:

```
LIB "ncalg.lib";
def A = makeQsl2(3);
setring A;
Qideal;
↦ Qideal[1]=Ke*Kf-1
qring Usl2q = Qideal;
Usl2q;
↦ //   characteristic : 0
↦ //   1 parameter    : q
↦ //   minpoly        : (q^2+q+1)
↦ //   number of vars : 4
↦ //        block   1 : ordering dp
↦ //                  : names    E F Ke Kf
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    FE=E*F+(2/3*q+1/3)*Ke+(-2/3*q-1/3)*Kf
↦ //    KeE=(-q-1)*E*Ke
↦ //    KfE=(q)*E*Kf
↦ //    KeF=(q)*F*Ke
↦ //    KfF=(-q-1)*F*Kf
↦ // quotient ring from ideal
↦ _[1]=Ke*Kf-1
```

Section 6.5.41.1 [makeUsl], page 213, Section 6.5.41.8 [makeQsl3], page 220, Section 6.5.41.5 [makeQso3], page 217

6.5.41.8 makeQsl3

Procedure from library `ncalg.lib` (see Section 6.5.41 [ncalg_lib], page 213).

Usage: makeQsl3([n]), n an optional int

Return: ring

Purpose: define the U_q(sl_3) as a factor-ring of a ring V_q(sl_3) modulo the ideal `Qideal`

Note: the output consists of a ring, presenting V_q(sl_3) together with the ideal called `Qideal` in this ring
activate this ring with the `setring` command
in order to create the U_q(sl_3) from the output, execute the command like `qring Usl3q = Qideal;`
If n is specified, the quantum parameter q will be specialized at the n-th root of unity

Example:

```
LIB "ncalg.lib";
def B = makeQsl3(5);
setring B;
qring Usl3q = Qideal;
Usl3q;
↦ //   characteristic : 0
↦ //   1 parameter    : q
↦ //   minpoly        : (q^4+q^3+q^2+q+1)
↦ //   number of vars : 10
↦ //      block   1 : ordering wp
↦ //                : names  f12 f13 f23 k1 k2 l1 l2 e12 e13 e23
↦ //                : weights 2  3  2  1  1  1  1  2  3  2
↦ //      block   2 : ordering C
↦ //   noncommutative relations:
↦ //    f13f12=(q^3)*f12*f13
↦ //    f23f12=(q^2)*f12*f23+(-q)*f13
↦ //    k1f12=(q^3)*f12*k1
↦ //    k2f12=(q)*f12*k2
↦ //    l1f12=(q^2)*f12*l1
↦ //    l2f12=(-q^3-q^2-q-1)*f12*l2
↦ //    e12f12=f12*e12+(1/5*q^3-3/5*q^2-2/5*q-1/5)*k1^2+ \
↦       (-1/5*q^3+3/5*q^2+2/5*q+1/5)*l1^2
↦ //    e13f12=f12*e13+(q^3+q^2+q+1)*l1^2*e23
↦ //    f23f13=(q^3)*f13*f23
↦ //    k1f13=(-q^3-q^2-q-1)*f13*k1
↦ //    k2f13=(-q^3-q^2-q-1)*f13*k2
↦ //    l1f13=(q)*f13*l1
```

```
↦ //    l2f13=(q)*f13*l2
↦ //    e12f13=f13*e12+(q)*f23*k1^2
↦ //    e13f13=f13*e13+(-1/5*q^3+3/5*q^2+2/5*q+1/5)*k1^2*k2^2+ \
↦       (1/5*q^3-3/5*q^2-2/5*q-1/5)*l1^2*l2^2
↦ //    e23f13=f13*e23+(q^3+q^2+q+1)*f12*l2^2
↦ //    k1f23=(q)*f23*k1
↦ //    k2f23=(q^3)*f23*k2
↦ //    l1f23=(-q^3-q^2-q-1)*f23*l1
↦ //    l2f23=(q^2)*f23*l2
↦ //    e13f23=f23*e13+(q)*k2^2*e12
↦ //    e23f23=f23*e23+(1/5*q^3-3/5*q^2-2/5*q-1/5)*k2^2+ \
↦       (-1/5*q^3+3/5*q^2+2/5*q+1/5)*l2^2
↦ //    e12k1=(q^3)*k1*e12
↦ //    e13k1=(-q^3-q^2-q-1)*k1*e13
↦ //    e23k1=(q)*k1*e23
↦ //    e12k2=(q)*k2*e12
↦ //    e13k2=(-q^3-q^2-q-1)*k2*e13
↦ //    e23k2=(q^3)*k2*e23
↦ //    e12l1=(q^2)*l1*e12
↦ //    e13l1=(q)*l1*e13
↦ //    e23l1=(-q^3-q^2-q-1)*l1*e23
↦ //    e12l2=(-q^3-q^2-q-1)*l2*e12
↦ //    e13l2=(q)*l2*e13
↦ //    e23l2=(q^2)*l2*e23
↦ //    e13e12=(q^3)*e12*e13
↦ //    e23e12=(q^2)*e12*e23+(-q)*e13
↦ //    e23e13=(q^3)*e13*e23
↦ // quotient ring from ideal
↦ _[1]=k2*l2-1
↦ _[2]=k1*l1-1
```

Section 6.5.41.1 [makeUsl], page 213, Section 6.5.41.7 [makeQsl2], page 219, Section 6.5.41.5 [makeQso3], page 217

6.5.41.9 GKZsystem

Procedure from library `ncalg.lib` (see Section 6.5.41 [ncalg_lib], page 213).

Usage: GKZsystem(A, sord, alg, [,v]); A intmat, sord, alg string, v intvec

Return: ring

Purpose: define a ring (Weyl algebra) and create a Gelfand-Kapranov-Zelevinsky (GKZ) system of equations in a ring from the following data:
`A` is an intmat, defining the system,
`sord` is a string with desired term ordering,
`alg` is a string, saying which algorithm to use (exactly like in toric_lib),
`v` is an optional intvec.
In addition, the ideal called `GKZid` containing actual equations is calculated and exported to the ring.

Note: activate the ring with the `setring` command. This procedure is elaborated by Oleksandr Yena

Assume: This procedure uses toric_lib and therefore inherits its input requirements:
possible values for input variable `alg` are: "ect","pt","blr", "hs", "du".
As for the term ordering, it should be a string `sord` in Singular format like "lp","dp", etc.
Please consult the toric_lib for allowed orderings and more details.

Example:

```
LIB "ncalg.lib";
// example 3.1.4 from the [SST] without vector w
intmat A[2][4]=3,2,1,0,0,1,2,3;
print(A);
↦      3     2     1     0
↦      0     1     2     3
def D1 = GKZsystem(A,"lp","ect");
setring D1;
D1;
↦ //   characteristic : 0
↦ //   2 parameter    : b(1) b(2)
↦ //   minpoly        : 0
↦ //   number of vars : 8
↦ //      block 1 : ordering lp
↦ //              : names x(1) x(2) x(3) x(4) d(1) d(2) d(3) d(4)
↦ //      block 2 : ordering C
↦ //   noncommutative relations:
↦ //    d(1)x(1)=x(1)*d(1)+1
↦ //    d(2)x(2)=x(2)*d(2)+1
↦ //    d(3)x(3)=x(3)*d(3)+1
↦ //    d(4)x(4)=x(4)*d(4)+1
print(GKZid);
↦ 3*x(1)*d(1)+2*x(2)*d(2)+x(3)*d(3)+(-b(1)),
↦ x(2)*d(2)+2*x(3)*d(3)+3*x(4)*d(4)+(-b(2)),
↦ d(2)*d(4)-d(3)^2,
↦ d(1)*d(4)-d(2)*d(3),
↦ d(1)*d(3)-d(2)^2
// now, consider A with the vector w=1,1,1,1
intvec v=1,1,1,1;
def D2 = GKZsystem(A,"lp","blr",v);
setring D2;
print(GKZid);
↦ 3*x(1)*d(1)+2*x(2)*d(2)+x(3)*d(3)+(-b(1)),
↦ x(2)*d(2)+2*x(3)*d(3)+3*x(4)*d(4)+(-b(2)),
↦ d(2)*d(4)-d(3)^2,
↦ d(1)*d(4)-d(2)*d(3),
↦ d(1)*d(3)-d(2)^2
```

Section D.4.16 [toric_lib], page 489

6.5.42 ncdecomp_lib

Library: ncdecomp.lib

Purpose: Decomposition of a module into its central characters

Authors: Viktor Levandovskyy, levandov@mathematik.uni-kl.de.

Overview: This library presents algorithms for the central character decomposition of a module, i.e. a decomposition into generalized weight modules with respect to the center. Based on ideas of O. Khomenko and V. Levandovskyy (see the article [L2] in the References for details).

Procedures:

6.5.42.1 CentralQuot

Procedure from library `ncdecomp.lib` (see Section 6.5.42 [ncdecomp_lib], page 223).

Usage: CentralQuot(M, G), M a module, G an ideal

Assume: G is an ideal in the center

Return: module

Purpose: compute the central quotient M:G

Theory: for an ideal G of the center of an algebra and a submodule M of A^n, the central quotient of M by G is defined to be
M:G := { v in A^n | z*v in M, for all z in G }.

Note: the output module is not necessarily given in a Groebner basis

Example:

```
LIB "ncdecomp.lib";
option(returnSB);
def a = makeUsl2();
setring a;
ideal I = e3,f3,h3-4*h;
I = std(I);
poly C=4*e*f+h^2-2*h;
ideal G = (C-8)*(C-24);
ideal R = CentralQuot(I,G);
R;
↦ R[1]=h
↦ R[2]=f
↦ R[3]=e
```

Section 6.5.42.2 [CentralSaturation], page 224, Section 6.5.42.3 [CenCharDec], page 224

6.5.42.2 CentralSaturation

Procedure from library `ncdecomp.lib` (see Section 6.5.42 [ncdecomp_lib], page 223).

Usage: CentralSaturation(M, T), for a module M and an ideal T

Assume: T is an ideal in the center

Return: module

Purpose: compute the central saturation of M by T, that is M:T^{\infty}, by repititive application of `CentralQuot`

Note: the output module is not necessarily a Groebner basis

Example:

```
LIB "ncdecomp.lib";
option(returnSB);
def a = makeUsl2();
setring a;
ideal I = e3,f3,h3-4*h;
I = std(I);
poly C=4*e*f+h^2-2*h;
ideal G = C*(C-8);
ideal R = CentralSaturation(I,G);
R=std(R);
vdim(R);
↦ 5
R;
↦ R[1]=h
↦ R[2]=ef-6
↦ R[3]=f3
↦ R[4]=e3
```

Section 6.5.42.1 [CentralQuot], page 223, Section 6.5.42.3 [CenCharDec], page 224

6.5.42.3 CenCharDec

Procedure from library `ncdecomp.lib` (see Section 6.5.42 [ncdecomp_lib], page 223).

Usage: CenCharDec(I, C); I a module, C an ideal

Assume: C consists of generators of the center

Return: a list L, where each entry consists of three records (if a finite decomposition exists)
L[*][1] ('ideal' type), the central character as the maximal ideal in the center,
L[*][2] ('module' type), the Groebner basis of the weight module, corresponding to the character in L[*][1],
L[*][3] ('int' type) is the vector space dimension of the weight module (-1 in case of infinite dimension);

Purpose: compute a finite decomposition of C into central characters or determine that there is no finite decomposition

Note: actual decomposition is a sum of L[i][2] above;
some modules have no finite decomposition (in such case one gets warning message)

Example:

```
LIB "ncdecomp.lib";
option(returnSB);
def a = makeUsl2(); // U(sl_2) in characteristic 0
setring a;
ideal I = e3,f3,h3-4*h;
I = twostd(I);           // two-sided ideal generated by I
vdim(I);                 // it is finite-dimensional
↦ 10
ideal Cn = 4*e*f+h^2-2*h; // the only central element
list T = CenCharDec(I,Cn);
T;
↦ [1]:
↦    [1]:
↦       _[1]=4ef+h2-2h-8
↦    [2]:
↦       _[1]=h
↦       _[2]=f
↦       _[3]=e
↦    [3]:
↦       1
↦ [2]:
↦    [1]:
↦       _[1]=4ef+h2-2h
↦    [2]:
↦       _[1]=4ef+h2-2h-8
↦       _[2]=h3-4h
↦       _[3]=fh2-2fh
↦       _[4]=eh2+2eh
↦       _[5]=f2h-2f2
↦       _[6]=e2h+2e2
↦       _[7]=f3
↦       _[8]=e3
↦    [3]:
↦       9
// consider another example
ideal J = e*f*h;
CenCharDec(J,Cn);
↦ There is no finite decomposition
↦ 0
```

Section 6.5.42.1 [CentralQuot], page 223, Section 6.5.42.2 [CentralSaturation], page 224

6.5.42.4 IntersectWithSub

Procedure from library `ncdecomp.lib` (see Section 6.5.42 [ncdecomp_lib], page 223).

Usage: IntersectWithSub(M,Z), M an ideal, Z an ideal

Assume: Z consists of pairwise commutative elements

Return: ideal, of two-sided generators, not a Groebner basis

Purpose: computes an intersection of M with the subalgebra, generated by Z

Note: usually Z consists of generators of the center

Example:

```
LIB "ncdecomp.lib";
ring r=(0,a),(e,f,h),Dp;
matrix @d[3][3];
@d[1,2]=-h;
@d[1,3]=2e;
@d[2,3]=-2f;
ncalgebra(1,@d); // parametric U(sl_2)
ideal I = e,h-a;
ideal C;
C[1] = h^2-2*h+4*e*f; // the center of U(sl_2)
ideal X = IntersectWithSub(I,C);
X;
↦ X[1]=4*ef+h2-2*h+(-a2-2a)
ideal G = e*f, h; // the biggest comm. subalgebra of U(sl_2)
ideal Y = IntersectWithSub(I,G);
Y;
↦ Y[1]=h+(-a)
↦ Y[2]=ef+(-a)
```

6.5.43 nctools_lib

Library: nctools.lib

Purpose: General tools for noncommutative algebras

Authors: Levandovskyy V., levandov@mathematik.uni-kl.de,
Lobillo, F.J., jlobillo@ugr.es,
Rabelo, C., crabelo@ugr.es.

Support: DFG (Deutsche Forschungsgesellschaft) and Metodos algebraicos y efectivos en grupos cuanticos, BFM2001-3141, MCYT, Jose Gomez-Torrecillas (Main researcher).

Main procedures: Auxiliary procedures:

6.5.43.1 Gweights

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: Gweights(r); r a ring or a square matrix

Return: intvec

Purpose: compute the weight vector for the following G-algebra:
for r itself, if it is of the type ring,
or for a G-algebra, defined by the square polynomial matrix r

Theory: `Gweights` returns a vector, which must be used to redefine the G-algebra. If the input is a matrix and the output is the zero vector then there is not a G-algebra structure associated to these relations with respect to the given variables. Another possibility is to use `weightedRing` to obtain directly the G-algebra with the new weighted ordering.

Example:

```
LIB "nctools.lib";
ring r = (0,q),(a,b,c,d),lp;
matrix C[4][4];
C[1,2]=q; C[1,3]=q; C[1,4]=1; C[2,3]=1; C[2,4]=q; C[3,4]=q;
matrix D[4][4];
D[1,4]=(q-1/q)*b*c;
ncalgebra(C,D);
r;
↦ //   characteristic : 0
↦ //   1 parameter    : q
↦ //   minpoly        : 0
↦ //   number of vars : 4
↦ //        block   1 : ordering lp
↦ //                  : names    a b c d
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    ba=(q)*ab
↦ //    ca=(q)*ac
↦ //    da=ad+(q2-1)/(q)*bc
↦ //    db=(q)*bd
↦ //    dc=(q)*cd
Gweights(r);
↦ 2,1,1,1
Gweights(D);
↦ 2,1,1,1
```

Section 6.5.43.2 [weightedRing], page 227

6.5.43.2 weightedRing

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: weightedRing(r); r a ring

Return: ring

Purpose: equip the variables of a ring with such weights,that the relations of new ring (with weighted variables) satisfies the ordering condition for G-algebras

Note: activate this ring with the "setring" command

Example:

```
LIB "nctools.lib";
ring r = (0,q),(a,b,c,d),lp;
matrix C[4][4];
C[1,2]=q; C[1,3]=q; C[1,4]=1; C[2,3]=1; C[2,4]=q; C[3,4]=q;
matrix D[4][4];
D[1,4]=(q-1/q)*b*c;
ncalgebra(C,D);
r;
↦ //   characteristic : 0
↦ //   1 parameter    : q
↦ //   minpoly        : 0
↦ //   number of vars : 4
↦ //        block   1 : ordering lp
↦ //                  : names    a b c d
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    ba=(q)*ab
↦ //    ca=(q)*ac
↦ //    da=ad+(q2-1)/(q)*bc
↦ //    db=(q)*bd
↦ //    dc=(q)*cd
def t=weightedRing(r);
setring t; t;
↦ //   characteristic : 0
↦ //   1 parameter    : q
↦ //   minpoly        : 0
↦ //   number of vars : 4
↦ //        block   1 : ordering M
↦ //                  : names    a b c d
↦ //                  : weights  2 1 1 1
↦ //                  : weights  0 0 0 1
↦ //                  : weights  0 0 1 0
↦ //                  : weights  0 1 0 0
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    ba=(q)*ab
↦ //    ca=(q)*ac
↦ //    da=ad+(q2-1)/(q)*bc
↦ //    db=(q)*bd
```

```
↦ //    dc=(q)*cd
```

Section 6.5.43.1 [Gweights], page 227

6.5.43.3 ndcond

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: ndcond();

Return: ideal

Purpose: compute the non-degeneracy conditions of the basering

Note: if `printlevel` > 0, the procedure displays intermediate information (by default, `printlevel`=0)

Example:

```
LIB "nctools.lib";
ring r = (0,q1,q2),(x,y,z),dp;
matrix C[3][3];
C[1,2]=q2; C[1,3]=q1; C[2,3]=1;
matrix D[3][3];
D[1,2]=x; D[1,3]=z;
ncalgebra(C,D);
r;
↦ //   characteristic : 0
↦ //   2 parameter    : q1 q2
↦ //   minpoly        : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    x y z
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    yx=(q2)*x*y+x
↦ //    zx=(q1)*x*z+z
ideal j=ndcond(); // the silent version
j;
↦ j[1]=(-q2+1)*y*z-z
printlevel=1;
ideal i=ndcond(); // the verbose version
↦ Processing degree : 1
↦ 1 . 2 . 3 .
↦ failed: (-q2+1)*y*z-z
↦ done
i;
↦ i[1]=(-q2+1)*y*z-z
```

6.5.43.4 Weyl

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: Weyl([p]); p an optional integer

Return: nothing

Purpose: create a Weyl algebra structure on a basering

Note: suppose the number of variables of a basering is 2k.
(if this number is odd, an error message will be returned)
by default, the procedure treats first k variables as coordinates x_i and the last k as differentials d_i
if nonzero p is given, the procedure treats 2k variables of a basering as k pairs (x_i,d_i), i.e. variables with odd numbers are treated as coordinates and with even numbers as differentials

Example:

```
LIB "nctools.lib";
ring A1=0,(x(1..2),d(1..2)),dp;
Weyl();
A1;
↦ //   characteristic : 0
↦ //   number of vars : 4
↦ //        block   1 : ordering dp
↦ //                  : names    x(1) x(2) d(1) d(2)
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    d(1)x(1)=x(1)*d(1)+1
↦ //    d(2)x(2)=x(2)*d(2)+1
kill A1;
ring B1=0,(x1,d1,x2,d2),dp;
Weyl(1);
B1;
↦ //   characteristic : 0
↦ //   number of vars : 4
↦ //        block   1 : ordering dp
↦ //                  : names    x1 d1 x2 d2
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    d1x1=x1*d1+1
↦ //    d2x2=x2*d2+1
```

Section 6.5.43.5 [makeWeyl], page 230

6.5.43.5 makeWeyl

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: makeWeyl(n,[p]); n an integer, n>0; p an optional integer (field characteristic)

Return: ring

Purpose: create an n-th Weyl algebra

Note: activate this ring with the "setring" command.
The presentation of an n-th Weyl algebra is classical: D(i)x(i)=x(i)D(i)+1,
where x(i) correspond to coordinates and D(i) to partial differentiations, i=1,...,n.

Example:

```
LIB "nctools.lib";
def a = makeWeyl(3);
setring a;
a;
↦ //   characteristic : 0
↦ //   number of vars : 6
↦ //        block   1 : ordering dp
↦ //                  : names    x(1) x(2) x(3) D(1) D(2) D(3)
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    D(1)x(1)=x(1)*D(1)+1
↦ //    D(2)x(2)=x(2)*D(2)+1
↦ //    D(3)x(3)=x(3)*D(3)+1
```

Section 6.5.43.4 [Weyl], page 229

6.5.43.6 makeHeisenberg

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: makeHeisenberg(n, [p,d]); int n (setting 2n+1 variables), optional int p (field characteristic), optional int d (power of h in the commutator)

Return: nothing

Purpose: create an n-th Heisenberg algebra in the variables x(1),y(1),...,x(n),y(n),h

Note: activate this ring with the "setring" command

Example:

```
LIB "nctools.lib";
def a = makeHeisenberg(2);
setring a;   a;
↦ //   characteristic : 0
↦ //   number of vars : 5
↦ //        block   1 : ordering lp
↦ //                  : names    x(1) x(2) y(1) y(2) h
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    y(1)x(1)=x(1)*y(1)+h
↦ //    y(2)x(2)=x(2)*y(2)+h
def H3 = makeHeisenberg(3, 7, 2);
setring H3;  H3;
```

```
↦ //   characteristic : 7
↦ //   number of vars : 7
↦ //        block   1 : ordering lp
↦ //                  : names    x(1) x(2) x(3) y(1) y(2) y(3) h
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    y(1)x(1)=x(1)*y(1)+h^2
↦ //    y(2)x(2)=x(2)*y(2)+h^2
↦ //    y(3)x(3)=x(3)*y(3)+h^2
```

Section 6.5.43.5 [makeWeyl], page 230

6.5.43.7 Exterior

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: Exterior();

Return: qring

Purpose: create the exterior algebra of a basering

Note: activate this qring with the "setring" command

Theory: given a basering, this procedure introduces the anticommutative relations x(j)x(i)=-x(i)x(j) for all j>i,
moreover, creates a factor algebra modulo the two-sided ideal, generated by x(i)^2 for all i

Example:

```
LIB "nctools.lib";
ring R = 0,(x(1..3)),dp;
def ER = Exterior();
setring ER;
ER;
↦ //   characteristic : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    x(1) x(2) x(3)
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    x(2)x(1)=-x(1)*x(2)
↦ //    x(3)x(1)=-x(1)*x(3)
↦ //    x(3)x(2)=-x(2)*x(3)
↦ // quotient ring from ideal
↦ _[1]=x(3)^2
↦ _[2]=x(2)^2
↦ _[3]=x(1)^2
```

6.5.43.8 findimAlgebra

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: findimAlgebra(M,[r]); M a matrix, r an optional ring

Return: nothing

Purpose: define a finite dimensional algebra structure on a ring

Note: the matrix M is used to define the relations x(j)*x(i) = M[i,j] in the basering (by default) or in the optional ring r.
The procedure equips the ring with the noncommutative structure.
The procedure exports the ideal (not a two-sided Groebner basis!), called `fdQuot`, for further qring definition.

Theory: finite dimensional algebra can be represented as a factor algebra of a G-algebra modulo certain two-sided ideal. The relations of a f.d. algebra are thus naturally divided into two groups: firstly, the relations on the variables of the ring, making it into G-algebra and the rest of them, which constitute the ideal which will be factored out.

Example:

```
LIB "nctools.lib";
ring r=(0,a,b),(x(1..3)),dp;
matrix S[3][3];
S[2,3]=a*x(1); S[3,2]=-b*x(1);
findimAlgebra(S);
fdQuot = twostd(fdQuot);
qring Qr = fdQuot;
Qr;
↦ //   characteristic : 0
↦ //   2 parameter    : a b
↦ //   minpoly        : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    x(1) x(2) x(3)
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    x(3)x(2)=(-b)/(a)*x(2)*x(3)
↦ // quotient ring from ideal
↦ _[1]=x(3)^2
↦ _[2]=x(2)*x(3)+(-a)*x(1)
↦ _[3]=x(1)*x(3)
↦ _[4]=x(2)^2
↦ _[5]=x(1)*x(2)
↦ _[6]=x(1)^2
```

6.5.43.9 ncRelations

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: ncRelations(r); r a ring

Return: list L with two elements, both elements are of type matrix:
L[1] = matrix of coefficients C,
L[2] = matrix of polynomials D

Purpose: recover the noncommutative relations via matrices C and D from a noncommutative ring

Example:

```
LIB "nctools.lib";
ring r = 0,(x,y,z),dp;
matrix C[3][3]=0,1,2,0,0,-1,0,0,0;
print(C);
↦ 0,1,2,
↦ 0,0,-1,
↦ 0,0,0
matrix D[3][3]=0,1,2y,0,0,-2x+y+1;
print(D);
↦ 0,1,2y,
↦ 0,0,-2x+y+1,
↦ 0,0,0
ncalgebra(C,D);
r;
↦ //   characteristic : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    x y z
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    yx=xy+1
↦ //    zx=2xz+2y
↦ //    zy=-yz-2x+y+1
def l=ncRelations(r);
print (l[1]);
↦ 0,1,2,
↦ 0,0,-1,
↦ 0,0,0
print (l[2]);
↦ 0,1,2y,
↦ 0,0,-2x+y+1,
↦ 0,0,0
```

Section 4.1.114 [ringlist], page 227, Section 6.4.33 [G-algebras], page 195

6.5.43.10 isCentral

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: isCentral(p); p poly

Return: int, 1 if p commutes with all variables and 0 otherwise

Purpose: check whether p is central in a basering (that is, commutes with every generator of a ring)

Note: if `printlevel` > 0, the procedure displays intermediate information (by default, `printlevel`=0)

Example:

```
LIB "nctools.lib";
ring r=0,(x,y,z),dp;
matrix D[3][3]=0;
D[1,2]=-z;
D[1,3]=2*x;
D[2,3]=-2*y;
ncalgebra(1,D); // this is U(sl_2)
poly c = 4*x*y+z^2-2*z;
printlevel = 0;
isCentral(c);
↦ 1
poly h = x*c;
printlevel = 1;
isCentral(h);
↦ Noncentral at: y
↦ Noncentral at: z
↦ 0
```

6.5.43.11 isNC

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: isNC();

Purpose: check whether a basering is commutative or not

Return: int, 1 if basering is noncommutative and 0 otherwise

Example:

```
LIB "nctools.lib";
def a = makeWeyl(2);
setring a;
isNC();
↦ 1
kill a;
ring r = 17,(x(1..7)),dp;
isNC();
↦ 0
kill r;
```

6.5.43.12 UpOneMatrix

Procedure from library `nctools.lib` (see Section 6.5.43 [nctools_lib], page 226).

Usage: UpOneMatrix(n); n an integer

Return: intmat

Purpose: compute an n x n matrix with 1's in the whole upper triangle

Note: helpful for setting noncommutative algebras with complicated coefficient matrices

Example:

```
LIB "nctools.lib";
ring   r = (0,q),(x,y,z),dp;
matrix C = UpOneMatrix(3);
C[1,3]   = q;
print(C);
↦ 0,1,(q),
↦ 0,0,1,
↦ 0,0,0
ncalgebra(C,0);
r;
↦ //   characteristic : 0
↦ //   1 parameter    : q
↦ //   minpoly        : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    x y z
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    zx=(q)*xz
```

6.5.44 qmatrix_lib

Library: qmatrix.lib

Purpose: Quantum matrices, quantum minors and symmetric groups

Authors: Lobillo, F.J., jlobillo@ugr.es
Rabelo, C., crabelo@ugr.es

Support: 'Metodos algebraicos y efectivos en grupos cuanticos', BFM2001-3141, MCYT, Jose Gomez-Torrecillas (Main researcher).

Main procedures: Auxiliary procedures:

6.5.44.1 quantMat

Procedure from library `qmatrix.lib` (see Section 6.5.44 [qmatrix_lib], page 236).

Usage: quantMat(n [, p]); n integer (n>1), p an optional integer

Return: ring (of quantum matrices). If p is specified, the quantum parameter q will be specialized at the p-th root of unity

Purpose: compute the quantum matrix ring of order n

Note: activate this ring with the "setring" command.
The usual representation of the variables in this quantum algebra is not used because double indexes are not allowed in the variables. Instead the variables are listed by reading the rows of the usual matrix representation.

Example:

```
LIB "qmatrix.lib";
def r = quantMat(2); // generate O_q(M_2) at q generic
setring r;   r;
↦ //   characteristic : 0
↦ //   1 parameter    : q
↦ //   minpoly        : 0
↦ //   number of vars : 4
↦ //        block   1 : ordering Dp
↦ //                  : names    y(1) y(2) y(3) y(4)
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    y(2)y(1)=1/(q)*y(1)*y(2)
↦ //    y(3)y(1)=1/(q)*y(1)*y(3)
↦ //    y(4)y(1)=y(1)*y(4)+(-q^2+1)/(q)*y(2)*y(3)
↦ //    y(4)y(2)=1/(q)*y(2)*y(4)
↦ //    y(4)y(3)=1/(q)*y(3)*y(4)
kill r;
def r = quantMat(2,5); // generate O_q(M_2) at q^5=1
setring r;   r;
↦ //   characteristic : 0
↦ //   1 parameter    : q
↦ //   minpoly        : (q^4+q^3+q^2+q+1)
↦ //   number of vars : 4
↦ //        block   1 : ordering Dp
↦ //                  : names    y(1) y(2) y(3) y(4)
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    y(2)y(1)=(-q^3-q^2-q-1)*y(1)*y(2)
↦ //    y(3)y(1)=(-q^3-q^2-q-1)*y(1)*y(3)
↦ //    y(4)y(1)=y(1)*y(4)+(-q^3-q^2-2*q-1)*y(2)*y(3)
↦ //    y(4)y(2)=(-q^3-q^2-q-1)*y(2)*y(4)
↦ //    y(4)y(3)=(-q^3-q^2-q-1)*y(3)*y(4)
```

Section 6.5.44.2 [qminor], page 237

6.5.44.2 qminor

Procedure from library `qmatrix.lib` (see Section 6.5.44 [qmatrix_lib], page 236).

Usage: qminor(I,J,n); I,J intvec, n int

Return: poly, the quantum minor

Assume: I is the ordered list of the rows to consider in the minor,
J is the ordered list of the columns to consider in the minor,
I and J must have the same number of elements,
n is the order of the quantum matrix algebra you are working with (quantMat(n)).

Example:

```
LIB "qmatrix.lib";
def r = quantMat(3); // let r be a quantum matrix of order 3
setring r;
intvec u = 1,2;
intvec v = 2,3;
intvec w = 1,2,3;
qminor(w,w,3);
↦ y(1)*y(5)*y(9)+(-q)*y(1)*y(6)*y(8)+(-q)*y(2)*y(4)*y(9)+ \
↦ (q^2)*y(2)*y(6)*y(7)+(q^2)*y(3)*y(4)*y(8)+(-q^3)*y(3)*y(5)*y(7)
qminor(u,v,3);
↦ y(2)*y(6)+(-q)*y(3)*y(5)
qminor(v,u,3);
↦ y(4)*y(8)+(-q)*y(5)*y(7)
qminor(u,u,3);
↦ y(1)*y(5)+(-q)*y(2)*y(4)
```

Section 6.5.44.1 [quantMat], page 236

6.5.44.3 SymGroup

Procedure from library `qmatrix.lib` (see Section 6.5.44 [qmatrix_lib], page 236).

Usage: SymGroup(n); n an integer (positive)

Return: intmat

Purpose: represent the symmetric group S(n) via integer vectors (permutations)

Note: each row of the output integer matrix is an element of S(n)

Example:

```
LIB "qmatrix.lib";
//  "S(3)={(1,2,3),(1,3,2),(3,1,2),(2,1,3),(2,3,1),(3,2,1)}";
SymGroup(3);
↦ 1,2,3,
↦ 1,3,2,
↦ 3,1,2,
↦ 2,1,3,
↦ 2,3,1,
↦ 3,2,1
```

Section 6.5.44.5 [LengthSym], page 239, Section 6.5.44.4 [LengthSymElement], page 239

6.5.44.4 LengthSymElement

Procedure from library `qmatrix.lib` (see Section 6.5.44 [qmatrix_lib], page 236).

Usage: LengthSymElement(v); v intvec

Return: int

Purpose: determine the length of v

Assume: v represents an element of S(n); otherwise the output may have no sense

Example:

```
LIB "qmatrix.lib";
intvec v=1,3,4,2,8,9,6,5,7,10;
LengthSymElement(v);
↦ 9
```

Section 6.5.44.3 [SymGroup], page 238, Section 6.5.44.5 [LengthSym], page 239

6.5.44.5 LengthSym

Procedure from library `qmatrix.lib` (see Section 6.5.44 [qmatrix_lib], page 236).

Usage: LengthSym(M); M an intmat

Return: intvec

Purpose: determine a vector, where the i-th element is the length of the i-th row of M

Assume: M represents a subset of S(n) (each row must be an element of S(n)); otherwise, the output may have no sense

Example:

```
LIB "qmatrix.lib";
def M = SymGroup(3); M;
↦ 1,2,3,
↦ 1,3,2,
↦ 3,1,2,
↦ 2,1,3,
↦ 2,3,1,
↦ 3,2,1
LengthSym(M);
↦ 0,1,2,1,2,3
```

Section 6.5.44.3 [SymGroup], page 238, Section 6.5.44.4 [LengthSymElement], page 239

A.6 Noncommutative Algebra: Examples

A.6.1 Left and two-sided Groebner bases

For a set of polynomials (resp. vectors) `S` in a noncommutative G-algebra, SINGULAR:PLURAL provides two algorithms for computing Groebner bases.

The command `std` computes a left Groebner basis of a left module, generated by the set `S` (see Section 6.3.28 [std (plural)], page 191). The command `twostd` computes a two-sided Groebner basis (which is in particular also a left Groebner basis) of a two-sided ideal, generated by the set `S` (see Section 6.3.31 [twostd], page 193).

In the example below, we consider a particular set `S` in the algebra $A := U(sl_2)$ with the degree reverse lexicographic ordering. We compute a left Groebner basis `L` of the left ideal generated by `S` and a two-sided Groebner basis `T` of the two-sided ideal generated by `S`.
Then, we read off the information on the vector space dimension of the factor modules `A/L` and `A/T` using the command `vdim` (see Section 6.3.32 [vdim (plural)], page 194).
Further on, we use the command `reduce` (see Section 6.3.27 [reduce (plural)], page 189) to compare the left ideals generated by `L` and `T`.

We set `option(redSB)` and `option(redTail)` to make SINGULAR compute completely reduced minimal bases of ideals (see Section 4.1.91 [option], page 206 and Section 6.4.34 [Groebner bases in G-algebras], page 196 for definitions and further details).

For long running computations, it is always recommended to set `option(prot)` to make SINGULAR display some information on the performed computations (see Section 4.1.91 [option], page 206 for an interpretation of the displayed symbols).

```
// ----- 1.  setting up the algebra
  ring A = 0,(e,f,h),dp;
  matrix D[3][3];
  D[1,2]=-h; D[1,3]=2*e; D[2,3]=-2*f;
  ncalgebra(1,D);
// ----- equivalently, you may use the following:
//  LIB "ncalg.lib";
//  def A = makeUsl2();
//  setring A;
// ----- 2.  defining the set S
  ideal S = e^3, f^3, h^3 - 4*h;
  option(redSB);
  option(redTail);
  option(prot);       // let us see the protocol
  ideal L = std(S);
↦ 3(2)s
↦ s
↦ s
↦ 5s
```

```
↦ s
↦ (4)s
↦ 4(5)s
↦ (7)s
↦ 3(6)s
↦ (8)(7)(6)4(5)(4)(3)(2)s
↦ (3)(2)s
↦ 3(3)(2)45
↦ (S:5)-----
↦ product criterion:6 chain criterion:16
  L;
↦ L[1]=h3-4h
↦ L[2]=fh2-2fh
↦ L[3]=eh2+2eh
↦ L[4]=2efh-h2-2h
↦ L[5]=f3
↦ L[6]=e3
  vdim(L);   // the vector space dimension of the module A/L
↦ 15
  option(noprot); // turn off the protocol
  ideal T = twostd(S);
  T;
↦ T[1]=h3-4h
↦ T[2]=fh2-2fh
↦ T[3]=eh2+2eh
↦ T[4]=f2h-2f2
↦ T[5]=2efh-h2-2h
↦ T[6]=e2h+2e2
↦ T[7]=f3
↦ T[8]=ef2-fh
↦ T[9]=e2f-eh-2e
↦ T[10]=e3
  vdim(T); // the vector space dimension of the module A/T
↦ 10
 // reduce L with respect to T:
  print(matrix(reduce(L,T)));
↦ 0,0,0,0,0,0
  // as we see, L is included in the left ideal generated by T
 // now, reduce T with respect to L:
  print(matrix(reduce(T,L)));
↦ 0,0,0,f2h-2f2,0,e2h+2e2,0,ef2-fh,e2f-eh-2e,0
  // the non-zero elements belong to T only
  ideal LT = twostd(L);
  // LT is the two-sided Groebner basis of L
  // LT and T coincide as left ideals:
  size(reduce(LT,T));
↦ 0
  size(reduce(T,LT));
↦ 0
```

A.6.2 Right Groebner bases and syzygies

Most of the SINGULAR:PLURAL commands correspond to the *left-sided* computations, that is left Groebner bases, left syzygies, left resolutions and so on. However, the *right-sided* computations can be done, using the *left-sided* functionality and *opposite* algebras.

In the example below, we consider the algebra $A := U(sl_2)$ and a set of generators $I = \{e^2, f\}$.

We will compute a left Groebner basis `LI` and a left syzygy module `LS` of a left ideal, generated by the set I .

Then, we define the opposite algebra `Aop` of `A`, set it as a basering, and create opposite objects of already computed ones.

Further on, we compute a right Groebner basis `RI` and a right syzygy module `RS` of a right ideal, generated by the set I in A .

```
// ----- setting up the algebra:
LIB "ncalg.lib";
def A = makeUsl2();
setring A; A;
↦ //   characteristic : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    e f h
↦ //        block   2 : ordering C
↦ //   noncommutative relations:
↦ //    fe=ef-h
↦ //    he=eh+2e
↦ //    hf=fh-2f
// ----- equivalently, you may use
// ring A = 0,(e,f,h),dp;
// matrix D[3][3];
// D[1,2]=-h; D[1,3]=2*e; D[2,3]=-2*f;
// ncalgebra(1,D);
option(redSB);
option(redTail);
matrix T;
// --- define a generating set
ideal   I = e2,f;
ideal  LI = std(I);  // the left Groebner basis of I
LI;                  // we see that I was not a Groebner basis
↦ LI[1]=f
↦ LI[2]=h2+h
↦ LI[3]=eh+e
↦ LI[4]=e2
module LS = syz(I);  // the left syzygy module of I
print(LS);
↦ -ef-2h+6,-f3,  -ef2-fh+4f,  -e2f2-4efh+16ef-6h2+42h-72,
```

```
↦ e3,  e2f2-6efh-6ef+6h2+18h+12,e3f-3e2h-6e2,e4f
// check: LS is a left syzygy, if T=0
T  = transpose(LS)*transpose(I);
print(T);
↦ 0,
↦ 0,
↦ 0,
↦ 0
// --- let us define the opposite algebra of A
def Aop = opposite(A);
setring Aop; Aop;                // see how Aop looks like
↦ //   characteristic : 0
↦ //   number of vars : 3
↦ //        block   1 : ordering a
↦ //                  : names    H F E
↦ //                  : weights  1 1 1
↦ //        block   2 : ordering ls
↦ //                  : names    H F E
↦ //        block   3 : ordering C
↦ //   noncommutative relations:
↦ //    FH=HF-2F
↦ //    EH=HE+2E
↦ //    EF=FE-H
// --- we "oppose" (transfer) objects from A to Aop
ideal   Iop = oppose(A,I);
ideal  RIop = std(Iop); // the left Groebner basis of Iop in Aop
module RSop = syz(Iop); // the left syzygy module of Iop in Aop
module LSop = oppose(A,LS);
module RLS  = syz(transpose(LSop));
// RLS is  the left syzygy of transposed LSop in Aop
// --- let us return to A and transfer (i.e. oppose)
// all the computed objects back
setring A;
ideal  RI = oppose(Aop,RIop); // the right Groebner basis of I
RI;              // it differs from the left Groebner basis LI
↦ RI[1]=f
↦ RI[2]=h2-h
↦ RI[3]=eh+e
↦ RI[4]=e2
module RS = oppose(Aop,RSop); // the right syzygy module of I
print(RS);
↦ -ef+3h+6,-f3, -ef2+3fh,-e2f2+4efh+4ef,
↦ e3, e2f2+2efh-6ef+2h2-10h+12,e3f, e4f
// check: RS is a right syzygy, if T=0
T = matrix(I)*RS;
T;
↦ T[1,1]=0
↦ T[1,2]=0
```

```
↦ T[1,3]=0
↦ T[1,4]=0
module RLS;
RLS = transpose(oppose(Aop,RLS));
// RLS is the right syzygy of a left syzygy of I
// it is I itself?
print(RLS);
↦ e2,f
```

Conclusion and Future Work

The implicit aim of this thesis was to create a possibly complete picture of symbolic methods for GR–algebras, together with numerous motivating and illustrating examples.

We have presented the Gröbner basics, enlisted in Section 2 and elaborated the theory in Chapters 2 and 3. Beyond our scope are Hilbert series and polynomials, more detailed discussion on dimensions, definition and computation of a multiplicity $e(M)$ of a module (what is done in the book [**56**]). The Betti numbers for graded modules over graded algebras (for example, algebras, where the relations can be made quasi–homogeneous) are implemented in PLURAL, but are also not discussed in details.

The last topics fit into a wide area of computations, related to homological algebra together with computations of Hom_A, Ext_A, Tor^A modules with the applications to computing of torsion submodules, Hochschild cohomology of algebras and modules, cohomology of modules over universal enveloping algebras and so on. In the lecture notes [**52**] we have already proposed an approach, which differs from the one described in the book [**14**]. Meanwhile, yet another way of computations was used in [**19**], applying algebraic involutions instead of continuous transitions between an algebra and it opposite. The work on this topic is in progress and shows a big potential in the future. I particular, the algorithms, related to homological algebra will be implemented in the library `nchomolog.lib`, which is already under development.

There are some generalizations of G–algebras, like rings of solvable type of Kredel [**46**], enveloping Lie fields of Apel ([**3**] and references therein), PBW rings of Bueso et.al. [**14**], Ore algebras of Chyzak and Salvy [**18**].

One of the most important setting is the non–commutative localization of a G–algebra with respect to a subset of a set of variables. Since a left resp. right quotient fields exists, as soon as the Ore condition is satisfied and a quotient field is, in particular, a localization, a needed algebra can be described as follows. Let $\mathbb{K}$ be a numeric field $(\mathbb{Z}_p, \mathbb{Q}, \mathbb{R}, \mathbb{C})$ and the set of variables is $S = X \cup D = \{x_1, \ldots, x_n\} \cup \{\partial_1, \ldots, \partial_n\}$. Let there be a set of relations R, such that $A := \mathbb{K}\langle S \mid R\rangle$ is a G–algebra and, say, $B := \mathbb{K}\langle X \mid R\mid_X\rangle$ is a proper G–subalgebra. Then, there exists a localization

of A with respect to B; in particular, we are interested in considering the quotient field of B, thus declaring x_i to rational variables, while ∂_i remain polynomial. Such an algebra is denoted by $A_B = \mathbb{K}(X)\langle S \backslash X \mid R\rangle$. Algebras of this type play an increasingly important role in modern computer algebra and its applications. In the theory of linear differential equations it is quite natural to consider partial differential ∂_i as polynomial generators, while the coordinates x_i are understood rather as rational generators. Nearly the same applies to linear operator algebras. In such cases, X is usually commutative and relations R on D do not involve x.

It seems that the framework of rings of solvable type and PBW rings is too general to be efficiently implemented in whole generality, on the other hand there are only a few applications, requiring this. A better perspective has the approach of Ore algebras, which has been successfully applied to concrete problems of System and Control Theory ([**19**]). The Gröbner basis theory in such algebras is a variation of the polynomial case we have presented, however, only two systems, OpenXM (Kan) and OreAlgebra (Mgfun) are able to perform Gröbner basis and some related computations. The efficiency of implementations was merely discussed; we are not aware of existence of new criteria and of revision of older. There are only some basic tests to measure the performance and so on. The natural generalization of our methods and a new task for Plural would be to review the methods, to adopt them into some reasonable framework (like we did for G–algebras) and implement several possible algorithms for computing Gröbner bases.

The "Delta Gröbner bases" method, proposed by F. Castro in [**16**] looks very promising, since it utilizes the commutative subalgebras at a full scale. The "involutive bases" ([**29, 72**]) are getting more sophisticated implementations and sometimes show a better performance, than the Gröbner bases. Since the proof of the equivalence of an involutive basis of a module to its non–reduced Gröbner basis, involutive basis algorithm is often used as an alternative to Buchberger's. The involutive approach is quite universal and exists both in G–algebras ([**72**]) and in rational PBW algebras described above.

Summarizing, we note that the movement towards rational noncommutativity is strategically important both for Singular:Plural research group and for the Computer Algebra in general, since numerous applications in natural sciences require a symbolical intervention, which, however, should have been done on a fast and reliable computer algebra system.

Bibliography

[1] D. Anick. On the homology of associative algebras. *Trans. Am. Math. Soc.*, 296:641–659, 1986.

[2] J. Apel and U. Klaus. FELIX – an assistant for algebraists . In *Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'91)*, pages 382–389. ACM Press, 1991.

[3] Apel, J. Gröbnerbasen in nichtkommutativen Algebren und ihre Anwendung. *Dissertation, Universität Leipzig*, 1988.

[4] Apel, J. Computational ideal theory in finitely generated extension rings. *Theor. Comput. Sci.*, 244(1-2):1–33, 2000.

[5] Apel, J. and Klaus, U. Implementation Aspects for Non-Commutative Domains. In *Proc. Computer Algebra in Physical Research*, pages 127–132. World Scientific, 1991.

[6] O. Bachmann and H. Schönemann. Monomial Representations for Gröbner Bases Computations. In *Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'98)*, pages 309–316. ACM Press, 1998.

[7] Becker, M., Levandovskyy, V. and Yena, O. A Singular 3.0 library for computations and operations with involutions `involut.lib`. 2003.

[8] Benkart, G. and Roby, T. Down–up algebras. *J. Algebra*, 209(1):305–344, 1998.

[9] Berger, R. Quantification de l'identité de Jacobi. (Quantization of the Jacobi identity). *C. R. Acad. Sci., Paris, Sér. I*, 312(10):721–724, 1991.

[10] Berger, R. The quantum Poincaré-Birkhoff-Witt theorem. *Commun. Math. Phys.*, 143(2):215–234, 1992.

[11] Bergman, G. The diamond lemma for ring theory. *Adv. Math.*, 29:178–218, 1978.

[12] Brickenstein, M. Neue Varianten zur Berechnung von Gröebnerbasen. *Diplomarbeit, Universität Kaiserslautern*, 2004.

[13] Buchberger, B. A Criterion for Detecting Unnecessary Reductions in the Construction of a Gröbner Basis. In Bose, N. K., editor, *Recent trends in multidimensional system theory*. 1985.

[14] Bueso, J., Gómez–Torrecillas, J. and Verschoren, A. *Algorithmic methods in non-commutative algebra. Applications to quantum groups*. Kluwer Academic Publishers, 2003.

[15] Bueso, J. L.; Gómez Torrecillas, J.; Lobillo, F.J.; Castro, F.J. An introduction to effective calculus in quantum groups. In Caenepeel, S. and Verschoren, A., editor, *Rings, Hopf algebras, and Brauer groups*, pages 55–83. Marcel Dekker, 1998.

[16] Castro-Jiménez, F.J. and Moreno-Frías, M.A. Gröbner δ-bases and Gröbner bases for differential operators. *arXiv. math. AG/0111266*, 2001.

[17] Castro-Jiménez, F.J. and Ucha, J.M. On the computation of Bernstein–Sato ideals. *J. Symbolic Computation*, 37:629–639, 2004.

[18] Chyzak, F. and Salvy, B. Non–commutative Elimination in Ore Algebras Proves Multivariate Identities. *J. Symbolic Computation*, 26(2):187–227, 1998.

[19] 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.

[20] Decker, D. and Eisenbud, D. Sheaf algorithms using the exterior algebra. In Eisenbud, D., Grayson, D., Stillman, M., Sturmfels, B., editor, *Computations in algebraic geometry with Macaulay 2*, 2001.

[21] Decker, W., Pfister, G. and Schönemann, H. A Singular 2.0 library for computing the primary decomposition and radical of ideals `primdec.lib`. 2001.

[22] Delius, G.W. and Gould, M.D. Quantum Lie algebras, their existence, uniqueness and q-antisymmetry. *Commun. Math. Phys.*, 185(3):709–722, 1997.

[23] Dixmier, J. *Enveloping Algebras*. AMS, 1996.

[24] Drozd, Y. and Kirichenko, V. *Finite dimensional algebras. With an appendix by Vlastimil Dlab*. Springer, 1994.

[25] Drozd, Y., Ovsienko, S. and Futornyj, V. On Gelfand-Zetlin modules. *Geometry and physics, Proc. Winter Sch., Srni/Czech. 1990, Suppl. Rend. Circ. Mat. Palermo*, II. Ser(26):143–147, 1991.

[26] Faugère, J.-C. A new Efficient Algorithm for Computing Gröbner Bases Without Reduction to Zero (F_5). In *Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'02)*, pages 75–83. ACM Press, 2002.

[27] Fulton, W. and Harris, J. *Representation theory. A first course*. Springer, 1991.

[28] García Román, M. and García Román, S. Gröbner bases and syzygies on bimodules. *arXiv. math. RA/0405550*, 2004.

[29] Gerdt, V.P. Involutive Algorithms for Computing Groebner Bases. In Pfister G., Cojocaru S. and Ufnarovski, V., editor, *Computational Commutative and Non-Commutative Algebraic Geometry*. IOS Press, 2005.

[30] Gómez–Torrecillas, J. and Lobillo, F.J. Global homological dimension of multifiltered rings and quantized enveloping algebras. *J. Algebra*, 225(2):522–533, 2000.

[31] Gómez–Torrecillas, J. and Lobillo, F.J. Auslander-regular and Cohen-Macaulay quantum groups. *arXiv. math. RA/0104283*, 2001.

[32] Gräbe, H.-G. The SymbolicData Benchmark Problems Collection of Polynomial Systems. In Calmet, J., Hausdorf, M. and Seiler, W., editor, *Proceedings "Workshop on Under- and Overdetermined Systems of Algebraic or Differential Equations" in Karlsruhe*, 2002.

[33] Green, E. Noncommutative Groebner bases, and projective resolutions. In Draexler, P., editor, *Computational methods for representations of groups and algebras. Proc. of the Euroconference in Essen, Germany, April*, pages 29–60. Birkhaeuser, 1999.

[34] Green, E. Multiplicative Bases, Gröbner Bases, and Right Gröbner Bases. *J. Symbolic Computation*, 29(4/5), 2000.

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

[36] Havlicek, M. and Klimyk, A. and Posta, S. Central elements of the algebras $U'_q(\mathrm{so}_m)$ and $U'_q(\mathrm{iso}_m)$. *Czech. J. Phys. (also arXiv. math. QA/9911130)*, 50(1):79–84, 2000.

[37] Humphreys, J. *Introduction to Lie algebras and representation theory*. Springer, 1980.

[38] Iorgov, N. On the Center of q-Deformed Algebra $U'_q(\mathrm{so}_3)$ Related to Quantum Gravity at q a Root of 1. In *Proceedings of IV Int. Conf. "Symmetry in Nonlinear Mathematical Physics", Kyiv, Ukraine*, 2001.

[39] Iorgov, N. and Klimyk, A. Nonclassical type representations of the q-deformed algebra $U'_q(\mathrm{so}_n)$. *Czech. J. Phys.*, 50(1):85–90, 2000.

[40] Isaev, A., Pyatov, P. and Rittenberg, V. Diffusion algebras. *arXiv. math. QA/0103603*, 2001.

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

[42] Khomenko, A. On a structure of induced modules over semi-simple Lie algebras. *Master Thesis, Universität Kaiserslautern*, 2000.

[43] Klimyk, A. and Schmüdgen, K. *Quantum groups and their representations*. Springer, 1997.

[44] Kobayashi, Yu. Gröbner bases of associative algebras and the Hochschild cohomology. *Trans. Am. Math. Soc.*, 2004.

[45] Kredel, H. MAS Modula-2 Algebra System. In *Proc. DISCO 90 Capri*, pages 270–271. LNCS 429, 1990.

[46] Kredel, H. *Solvable polynomial rings*. Shaker, 1993.

[47] Kubichka, O. Composition algebras for representations of finite type posets. *Bulletin of the University of Kyiv. Series: Physics and Mathematics*, 4, 2002.

[48] Levandovskyy, V. Gröbner bases of a class of non–commutative algebras. *Master Thesis, Universität Kaiserslautern*, 2000.

[49] Levandovskyy, V. On Gröbner bases for non–commutative G-algebras. In Kredel, H. and Seiler, W.K., editor, *Proceedings of the 8th Rhine Workshop on Computer Algebra*, 2002.

[50] Levandovskyy, V. A SINGULAR 3.0 library for computing the central character decomposition of a module `ncdecomp.lib`. 2004.

[51] Levandovskyy, V. and Motsak, O. A SINGULAR 3.0 library for definitions of important GR-algebras `ncalg.lib`. 2003.

[52] Levandovskyy, V. and Pfister G. An Introduction to the Commutative and Non–commutative Computer Algebra: Gröbner bases as a Tool for Homological Algebra. In Laudal O.A., editor, *Advanced School and Conference on Non–commutative Geometry*. ICTP, Trieste, 2004.

[53] 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.

[54] Levandovskyy, V., Lobillo, F.J. and Rabelo, C. A SINGULAR 3.0 library, providing general tools for noncommutative algebras `nctools.lib`. 2004.

[55] Levasseur, T. Krull dimension of the enveloping algebra of a semisimple lie algebra. *Proc. Am. Math. Soc.*, 130(12):3519–3523, 2002.

[56] Li, H. *Noncommutative Gröbner bases and filtered-graded transfer*. Springer, 2002.

[57] Lobillo, F.J. and Rabelo, C. A SINGULAR 3.0 library for calculating the Gelfand-Kirillov dimension of modules `gkdim.lib`. 2004.

[58] Lobillo, F.J. and Rabelo, C. A SINGULAR 3.0 library for computations with quantum matrices, quantum minors and symmetric groups `qmatrix.lib`. 2004.

[59] 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.

[60] Mora, T. Groebner bases in non-commutative algebras. In *Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'02)*, pages 150–161. LNCS 358, 1989.

[61] Mora, T. An introduction to commutative and non-commutative Groebner bases. *Theor. Comp. Sci.*, 134:131–173 , 1994.

[62] Motsak, O. A SINGULAR 3.0 library for computing centers of GR-algebras `center.lib`. 2004.

[63] Nüßler, T. and Schönemann, H. Gröbner bases in algebras with zero–divisors. *Preprint 244, Universität Kaiserslautern*, 1993.

[64] 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):201–233, 1999.

[65] Ovsienko, S. Strongly nilpotent matrices and Gelfand–Zetlin modules. *Linear Algebra Appl.*, 365:349–367, 2003.

[66] Płoski, A. Algebraic dependence of polynomials after O. Perron and some applications. In Pfister G., Cojocaru S. and Ufnarovski, V., editor, *Computational Commutative and Non-Commutative Algebraic Geometry*. IOS Press, 2005.

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

[68] Rosenkranz, M., Buchberger, B. and Engl, H.W. Solving linear boundary value problems via non-commutative gröbner bases. *Appl. Anal.*, 82(7):665–675, 2003.

[69] Rudakov, A.N. and Shafarevich, I.R. Irreducible representations of a simple three-dimensional lie algebra over a field of finite characteristic. *Math. Notes of Acad. Sci. USSR*, 2:760–767, 1967.

[70] Schönemann, H. Algorithms in Singular. In *Reports On Computer Algebra No. 2*. Centre for Computer Algebra, University of Kaiserslautern, 1996. Available at http://www.mathematik.uni-kl.de/∼zca.

[71] Schönemann, H. Singular in a Framework for Polynomial Computations. In Joswig, M. and Takayama, N., editor, *Algebra, Geometry and Software Systems*, pages 163–176. Springer, 2003.

[72] Seiler, W. Involution, the formal theory of differential equations and its applications in computer algebra and numerical analysis. *Habilitation thesis, Universität Mannheim*, 2002.

[73] Stafford, J.T. and Van den Bergh, M. Noncommutative curves and noncommutative surfaces. *arXiv. math. RA/9910082*, 1999.

[74] The SymbolicData Project, 2000–2002. see `http://www.SymbolicData.org`.

[75] Takayama, N. A benchmark test for Gröbner basis systems of differential operators I, 1995. Available from `http://www.math.sci.kobe-u.ac.jp/∼taka/`.

[76] Vogan, D. A. Gelfand-Kirillov dimension for Harish-Chandra modules. *Invent. Math.*, 48:75–98, 1978.

[77] Yan, T. The Geobucket Data Structure for Polynomials. *J. Symbolic Computation*, 25(3):285–294, March 1998.

Wissenschaftlicher Werdegang

Name: Viktor Levandovskyy

Geburtstag, -ort: 22. August 1976 in Kiew, Ukraine

1983–1993	Besuch und Abschluss der Schule in Kiew, Ukraine
1993–1998	Studium der Mathematik an der Kiewer Staatlichen Taras Schewtchenko Universität
1998	Diplom in Mathematik (Universität Kiew)
1997–2000	Studium der Mathematik mit Nebenfach Informatik an der Universität Kaiserslautern, in Rahmen des Studiengangs "Mathematics International".
2000	Diplom in Mathematik (Universität Kaiserslautern)
2000–2005	wissenschaftlicher Mitarbeiter in der AG AG (Algebraische Geometrie) des Fachbereiches Mathematik der Universität Kaiserslautern.
ab 2005	wissenschaftlicher Mitarbeiter im RISC (Research Institute for Symbolic Computation) der Universität Linz, Österreich.

Scientific Career

Name: Viktor Levandovskyy

Date/Place of Birth: 22 of August 1976 in Kyiv, Ukraine

1983–1993	School in Kyiv, Ukraine
1993–1998	Study of Mathematics at the Kyiv State Taras Shevchenko University
1998	Diploma in Mathematics (Kyiv State University)
1997–2000	Study of Mathematics with Computer Science as secondary subject at the University of Kaiserslautern, participation in the "Mathematics International" program.
2000	Diploma in Mathematics (University of Kaiserslautern)
2000–2005	research scientist at the Algebraic Geometry Group (Faculty of Mathematics, University of Kaiserslautern).
from 2005	research scientist at the RISC (Research Institute for Symbolic Computation), University of Linz, Austria.