RWTH Aachen University
Lehrstuhl D für Mathematik
Prof. Dr. rer. nat. Eva Zerz

Bachelor Thesis

Factorization of polynomials in a class of noncommutative algebras

Kandidat:	Albert Heinle
Betreuer:	Dr. rer. nat. Viktor Levandovskyy
Zweitgutachter:	Prof. Dr. rer. nat. Wilhelm Plesken
Beginn:	12.10.2009
Abgabe:	31.03.2010

Abstract

Factorization problems appear in almost every area of mathematics. Looking at the RSA algorithm, their difficulty guarantees the security of a cryptographical method. The determination of the variety of a set of polynomials may become easier when one makes use of factorization. This can be seen in the factorizing Gröbner bases algorithm.

This thesis deals with a class of polynomials, where factorization could be interesting: the class of graded skew polynomial rings. The precise meaning of this term will be explained later on. Although being graded is not a generic property of skew polynomial rings, there are some of them which people are seriously interested in.

Our main focus will lie on the Weyl algebra. We will demonstrate how to construct a factorization algorithm for polynomials in this ring by using the nice properties the Weyl algebra has as a graded ring. After that, we will see how the concepts in the factorization algorithm for Weyl algebras could be generalized for any graded skew polynomial ring.

The first two chapters serve the purpose of introducing the reader into some definitions and concepts we will make use of. They should also raise the reader's awareness for potentially unfamiliar phenomena that can happen in noncommutative polynomial rings. The end of the second chapter is dedicated to the first Weyl algebra and some interesting facts about it. We will make use of these results when we are going to construct a factorization algorithm, the leitmotif of the third chapter. Subsequently we will develop a way to generalize these concepts to an arbitrary graded skew polynomial ring.

Beside some words on possible future work, the conclusion of this thesis consists of some statements on an experimental implementation of the presented algorithms in the computer algebra system SINGULAR. In general, its performance is compared to current implementations. But in some cases it delivers all possible factorizations quite fast, where other implementations need noticable more time for their calculations.

Contents

CHAPTER 1

Introduction

1. Motivation

"Why dealing with noncommutative rings and their factorization properties?" – This is a question I have been asked a couple of times during the progress of writing this thesis. Answering it and giving examples of noncommutative rings with more than academic relevance was not easy at first. Math students do see noncommutative rings in the first or second term, e.g. square matrices of a fixed size. They are relevant in practice, and with the Jordan normal form, we have already seen some kind of useful factorization of them. But there are more noncommutative rings. One of them is for example the so-called Weyl algebra (which will be defined later). To see why people could be interested in factorizing the elements in this algebra, let's first state an example.

Example 1.1. The Gauss hypergeometric equation
The Gauss hypergeometric equation is the following second order linear ordinary differential equation for an unknown univariate function $f(x)$:

$$\left(x(x-1)\frac{d^2}{dx^2} + (c - x(a+b+1))\frac{d}{dx} - ab\right) \bullet f = 0, \quad a, b, c \in \mathbb{C}$$

As we will show later, the differential operator on the left hand side can be seen as an element of the Weyl algebra. If we can find a way to factorize it, we can split this one big equation into several smaller ones. They may be easier to solve and provide significant information about general solutions.

This example convinces at least people, who are dealing with differential equations daily, of the usefulness of noncommutative rings and the factorization in them. But my own motivation to deal with factorization in noncommutative rings was not only the question how I can use it later in concrete calculations. I wanted to see new problems arising when we dispense with commutativity, and how concepts in the noncommutative case could lead to a generalization of the concepts in the commutative case. Stating all problems (and there are many open ones) would unfortunately go beyond the scope of this thesis. Furthermore, as we will see, not all concepts in the noncommutative can be seen as generalizations of those in the commutative world.

2. Notations and definitions

In this section, we will get familiar to some basic definitions and notations, which will be used throughout this thesis. Occasionally, some of them will be specified otherwise.

- A ring R will denote a nonzero noncommutative ring with 1.
- $\mathbb{N}$ represents the natural numbers without zero.
- $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ denote the sets of the integer, rational, real and complex numbers.
- $\mathbb{K}$ represents an arbitrary field of characteristic zero.
- A ring homomorphism $\varphi : R \to S$ always maps 1_R to 1_S.
- $\mathrm{Hom}(R, S)$, where S, R are groups, defines the set of group homomorphisms from R to S.
- Given a ring R, let $R[x_1, \ldots, x_n]$ denote the associated polynomial ring. The leading coefficient of a polynomial f – with respect to a given order – will be denoted by $\mathrm{lc}(f)$, and the leading monomial (without the coefficient) by $\mathrm{lm}(f)$. For the degree, we will write $\deg(f)$ (if not specified, we will always mean the total degree).
- Let $n \in \mathbb{N}$. Then $\underline{n}$ denotes the set $\{1, \ldots, n\} \subset \mathbb{N}$.

Definition 1.2. Let $r, s \in R$. We say r is a right divisor of s (or s is a left multiple of r), if there exists at least one $q \in R$ with $s = qr$. Left divisibility is defined in a similar way. When we write $r \mid s$, then it will be specified in the text whether we mean division on the left hand side or on the right hand side.

Definition 1.3. An additive subgroup I of R is said to be a left ideal of R, if the following condition holds:

$$\forall r \in R, x \in I : rx \in I.$$

Analogously we define a right ideal. If I is both a left and a right R-ideal, then we call I a two-sided ideal of R.

Definition 1.4. An abelian group $(M, +)$ endowed with a scalar multiplication

$$R \times M \to M : (r, m) \mapsto rm$$

satisfying the following properties for any $r, s \in R$ and $m, n \in M$:

(1) $(r + s)m = rm + sm$
(2) $r(m + n) = rm + sn$
(3) $(rs)m = r(sm)$
(4) $1m = m$

is called a **left** R**-module**. Right R-modules are defined similarly.

Convention: If we are talking about an ideal or a module without specifying whether we mean a left, right, or two-sided one, then it will always be a left one.

Definition 1.5. We call a ring R left Noetherian, if one of the three following (equivalent) conditions is satisfied:

(1) Every ascending chain of ideals $I_i, i \in \mathbb{N}_0$,
$$I_0 \subseteq I_1 \subseteq I_2 \subseteq \dots$$
becomes stationary. This means that there exist a $k \in \mathbb{N}$ such that $I_k = I_i$ for all $i \geq k$.

(2) Every ideal of R is finitely generated.

(3) Every nonempty set of ideals of R has a maximal element.

3. Horrible things happening in the noncommutative case

Now that we have introduced some basics in dealing with noncommutative rings, we are going to see some odd properties they have.

First of all, we will see that we must in general bid farewell to the factor rings induced by an ideal, if this ideal is not two-sided.

Example 1.6. Let R be a ring, and $I \subset R$ a left ideal of R. Then R/I is a **left R-module** *(instead of being even a ring as it is usual in the commutative case).*
The properties R/I needs to fulfill to be a left R-module are easy to check. What does go wrong in the noncommutative case is the multiplication of two elements in R/I:
Let $a + I$, $b + I \in R/I$. Then
$$(a+I)(b+I) = ab + \underbrace{Ib}_{\not\subseteq I \text{ in general}} + \underbrace{aI + I^2}_{\subseteq I \text{ by definition}} \neq ab + I \text{ in general.}$$

Another well studied concept are unique factorization domains. Generalizing this idea to the noncommutative case in a canonical way will not work, because then simple polynomial rings will not be UFDs:

Example 1.7. Let us consider the ring $R = \mathbb{C}[t]$ with the following multiplication rule for all $z \in \mathbb{C}$:
$$t \cdot z = \overline{z} \cdot t.$$
We will see later that this construct fulfills all the properties to be a ring. Then
$$(t + \overline{z})(t - z) = (t - |z|)(t + |z|).$$
If $z \in \mathbb{C}\backslash\mathbb{R}$, then we get two “distinct” factorizations.

Concerning the problem of factorization, a decision problem can arise, as the next example shows:

Example 1.8. Let $\mathbb{K}[x, y]$ be the ring of polynomials with two variables x and y and the commutation rules:

- $\forall k \in \mathbb{K} : kx = xk, ky = yk$
- $yx = xy + 1.$

In the commutative case, when we want to factorize a polynomial $p := yx$, we do not expect one of its factors to have more than one term. In this case, $p = 1 \cdot (xy + 1)$, which is counter-intuitive at first sight. Furthermore, when we want to determine whether x is dividing a given polynomial p or not, we have an easy way to do this in the commutative case: the condition

$$x \mid p \Leftrightarrow \text{every term of } p \text{ is divided by } x$$

holds. But in this example, $x \mid xy + 1$, but $x \nmid 1$.

CHAPTER 2

Graded Rings

1. Skew polynomial rings

After we have specified some basic notions and definitions in the previous chapter, we will now introduce a concrete class of noncommutative rings, the skew polynomial rings. They will lead us to the first noncommutative ring in which factorization could be interesting.

Let $R[x]$ be the ring of polynomials with coefficients in R, where x is not necessarily commuting with the elements in R, but each polynomial can be uniquely expressed in the form $\sum\limits_i a_i x^i$ for finitely many $a_i \in R$. Our goal is to define commutation rules for x and the elements in R, such that we achieve this condition. This will mean that also the polynomials $xa, a \in R$ can be rewritten in this form. For the degrees to behave appropriately when we multiply two polynomials, we require

$$xa \in Rx + R.$$

Thus, we can define

$$xa = \sigma(a)x + \delta(a) \tag{1}$$

for all $a \in R$ with $\sigma, \delta : R \to R$ endomorphisms of the additive group $(R, +)$ (they have to be homomorphisms, because otherwise addition and multiplication in $R[x]$ wouldn't be well defined). Polynomial rings, where such a commutation rule holds, are called **skew polynomial rings**. Moreover, we have

$$x(ab) = \sigma(ab)x + \delta(ab)$$

and

$$(xa)b = (\sigma(a)x + \delta(a))b = \sigma(a)\sigma(b)x + \sigma(a)\delta(b) + \delta(a)b$$

for $a, b \in R$. Since we impose the associativity of the multiplication, it follows that σ is a ring endomorphism of $(R, +, \cdot)$ and we get

$$\forall a, b \in R : \delta(ab) = \sigma(a)\delta(b) + \delta(a)b.$$

Remark 2.1. In particular we have

- $\sigma(1) = 1$
- $\delta(1) = 0$

in this case. The first statement follows from the definition of a ring homomorphism, the second by the rule given above. (Assume that

$\delta(1) = r \in R$. Then $r = \delta(1) = \delta(1 \cdot 1) = \sigma(1)\delta(1) + \delta(1) \cdot 1 = r + r$. This implies that $r = 0$.)

These observations lead to the following definition:

Definition 2.2. Let σ be a ring endomorphism of R. A **σ-derivation** of R is an additive endomorphism $\delta : R \to R$ with the following property:

$$\forall r, s \in R : \delta(rs) = \sigma(r)\delta(s) + \delta(r)s.$$

We will call (σ, δ) a **quasi-derivation** of R.

A question which arises in this construction is the following: Given a ring endomorphism σ and an additive endomorphism δ, where (σ, δ) is a quasi-derivation on R, will the equation (1) be always well defined, if we construct the polynomial ring over R? Fortunately, the answer is yes.

In order to show this, we are going to show the following lemma at first:

Lemma 2.3. For any pair of maps $\sigma, \delta : R \to R$, the following assertions are equivalent:

- (σ, δ) is a quasi-derivation.
- The map

$$\phi : R \to M_2(R), r \mapsto \begin{bmatrix} \sigma(r) & \delta(r) \\ 0 & r \end{bmatrix},$$

where $M_2(R)$ represents the 2×2 matrices with entries in R, is a ring homomorphism.

PROOF. We begin with (σ, δ) being a quasi-derivation and prove that ϕ is a ring homomorphism.

By assumption, σ, δ are $(R, +, \cdot)$ resp. $(R, +)$-endomorphisms. Let $r, s \in R$ arbitrary.

$$\begin{aligned}
\phi(r+s) &= \begin{bmatrix} \sigma(r+s) & \delta(r+s) \\ 0 & r+s \end{bmatrix} \\
&\overset{\sigma,\delta\in\mathrm{Hom}((\mathrm{R},+),(\mathrm{R},+))}{=} \begin{bmatrix} \sigma(r) & \delta(r) \\ 0 & r \end{bmatrix} + \begin{bmatrix} \sigma(s) & \delta(s) \\ 0 & s \end{bmatrix} \\
&= \phi(r) + \phi(s) \\
\phi(r)\phi(s) &= \begin{bmatrix} \sigma(r) & \delta(r) \\ 0 & r \end{bmatrix} \begin{bmatrix} \sigma(s) & \delta(s) \\ 0 & s \end{bmatrix} \\
&= \begin{bmatrix} \sigma(r)\sigma(s) & \sigma(r)\delta(s) + \delta(r)s \\ 0 & rs \end{bmatrix} \\
&\overset{2.2}{=} \begin{bmatrix} \sigma(rs) & \delta(rs) \\ 0 & rs \end{bmatrix} \\
&= \phi(rs) \\
\phi(1) &= \begin{bmatrix} \sigma(1) & \delta(1) \\ 0 & 1 \end{bmatrix} \\
&\overset{\delta(1)=0}{=} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\end{aligned}$$

Thus ϕ is a ring homomorphism.

Assume conversely that ϕ is a ring homomorphism. By the equations given above, we deduce for all $r, s \in R$:

- $\sigma(r+s) = \sigma(r) + \sigma(s)$ and $\delta(r+s) = \delta(r) + \delta(s)$
- $\delta(rs) = \sigma(r)\delta(s) + \delta(r)s$.

Thus (σ, δ) is a quasi-derivation by definition. □

With the help of this lemma, we construct our polynomial ring with the desired properties.

Theorem 2.4. Let (σ, δ) be a quasi-derivation on R. Then there exists a ring S with the following properties:

(1) R is a subring of S.
(2) There exists an element $x \in S$, such that S is freely generated as a left R-module by the non-negative powers $1, x, x^2, \ldots$ of x.
(3) $\forall r \in R : xr = \sigma(r)x + \delta(r)$.

This ring S is a skew polynomial ring and called an **Ore extension of** R, and is further denoted by $R[x; \sigma, \delta]$. (**Convention:** If σ is the identity function, then we will just write $R[x; \delta]$. If $\delta \equiv 0$, then we will denote S by $R[x; \sigma]$.)

Proof. Let E be the ring of module endomorphisms of $R[y]$ viewed as a $\mathbb{Z}$-module, where y is an indeterminate. R can be embedded in

E acting by right multiplication. The quasi-derivation (σ, δ) may be extended to a quasi-derivation on $R[y]$ defined by putting

$$\begin{aligned} \sigma(ry^i) &= \sigma(r)y^i \\ \delta(ry^i) &= \delta(r)y^i. \end{aligned}$$

With the help of the previous lemma, we can check easily that (σ, δ) is still a quasi-derivation on $R[y]$. It suffices to verify that the map

$$\Phi : R[y] \to M_2(R[y]) : ry^i \mapsto \begin{bmatrix} \sigma(ry^i) & \delta(ry^i) \\ 0 & ry^i \end{bmatrix}$$

is a ring homomorphism. Identifying $M_2(R[y])$ with $M_2(R)[y]$, we get

$$\begin{aligned} \Phi(ry^i) &= \begin{bmatrix} \sigma(ry^i) & \delta(ry^i) \\ 0 & ry^i \end{bmatrix} \\ &= \begin{bmatrix} \sigma(r)y^i & \delta(r)y^i \\ 0 & ry^i \end{bmatrix} \\ &= \Phi(r)y^i. \end{aligned}$$

Now define $x \in E$ by

$$x(f) = \sigma(f)y + \delta(f)$$

for $f \in R[y]$ arbitrary. Then

$$(xr)(f) = \sigma(r)x(f) + \delta(r)f$$

for any $r \in R$. Thus

$$xr = \sigma(r)x + \delta(r).$$

In particular, $xR \subseteq Rx + R$, and by iteration of this argument, we get

$$x^iR \subseteq Rx^i + \ldots + Rx + R$$

for $i \in \mathbb{N}$. It follows that $S := \sum_i Rx^i$ is closed under multiplication (which means: S is a subring of E).

It remains to show that the x^i are left linearly independent over R. Let $s = r_0 + \ldots + r_nx^n$ for some $r_i \in R$, and suppose that $s = 0$. Because $x^i(y)$ equals y^i, it follows that

$$r_0 + \ldots + r_ny^n = s(y) = 0.$$

Thus $r_0 = \ldots = r_n = 0$, which proves the assertion. □

Example 2.5. The construction of the polynomial ring $R[x]$ with coefficients in R is nothing but an Ore extension. Choose σ as the identity function on R and let $\delta \equiv 0$.

But what can be said about the degree properties (concerning the degree of polynomials defined by

$$\deg(\sum_{i\in I} r_ix^i) = \max\{i \in I | r_i \neq 0\}, \quad \text{if } \sum_{i\in I} r_ix^i \neq 0 \text{ and}$$

$$\deg(0) = -\infty,$$

where I is a finite set)? In an Ore-extension $R[x;\sigma,\delta]$, can we still make conclusions about the degree of the product of polynomials when we just have information about the degree of the factors? Yes, we can.

Lemma 2.6. Let (σ,δ) be a quasi-derivation on R and consider $f, g \in R[x;\sigma,\delta]$. Then:

(1) $\deg(f+g) \leq \max\{\deg(f), \deg(g)\}$.
(2) $\deg(fg) \leq \deg(f) + \deg(g)$.
(3) If σ is injective and $\mathrm{lc}(f)$ or $\mathrm{lc}(g)$ is not a zero divisor in R, then

$$\deg(fg) = \deg(f) + \deg(g).$$

Now one could ask how properties of a ring R with a quasi-derivation (σ,δ) are inherited by its Ore extension $R[x;\sigma,\delta]$. The following theorem sheds light on this topic:

Theorem 2.7. If σ is injective and R is a domain, then $R[x;\sigma,\delta]$ is a domain. If R is a division ring, then $R[x;\sigma,\delta]$ is a principal ideal domain. If σ is an automorphism and R is Noetherian, then $R[x;\sigma,\delta]$ is Noetherian.

Proof. Pick nonzero elements f, g in $R[x;\sigma,\delta]$. Let $\deg(f) = n$, $\deg(g)=m$, $m, n \in \mathbb{N}_0$ and $\mathrm{lc}(f)=: f_n$, $\mathrm{lc}(g)=: g_m$. Because R is a domain by assumption, $f_n g_m \neq 0$. Thus $\deg(fg) = n + m$, i.e. $fg \neq 0$.

For the second assertion, let I be a nonzero ideal of $R[x;\sigma,\delta]$. Choose an element $0 \neq p \in I$ with minimal degree. Without loss of generality let $\mathrm{lc}(p) = 1$. We claim $I = R[x;\sigma,\delta]\cdot p$. Obviously, we have $R[x;\sigma,\delta]\cdot p \subseteq I$. We prove the reverse inclusion by induction on the degree.

Let $\deg(p) = 1$. The only elements of $R[x;\sigma,\delta]$ with degree less than 1 are the elements in R. If $I = R[x;\sigma,\delta]$, then we would have a contradiction to the minimality of $\deg(p)$. Assume that there is an element $q \in I$ with $q \notin R[x;\sigma,\delta]\cdot p$. If the degree is greater than 1, say m, we have $q - \mathrm{lc}(q)x^{m-1}p \in I$. Thus we can reduce every polynomial of degree greater than 1 by p to a degree that equals 1. Let $\hat{q}$ be the element of degree 1 after the reduction steps and let $\mathrm{lc}(\hat{q})=1$. Assume $p \neq \hat{q}$. Then we have $p - \hat{q} \in I$, but $p - \hat{q} \in R$. Again we arrive to a contradiction to the minimality of the degree of p. Thus $I = R[x;\sigma,\delta]\cdot p$ as desired. The induction step is proved similarly.

Now we show the third assertion. Assume R to be left Noetherian and let I be an ideal of $R[x;\sigma,\delta]$. For any positive integer n, denote by I_n the set of leading coefficients of elements in I of degree at most n. It is easy to check that I_n is an ideal of R. These ideals are called the nth leading left ideals of I. The I_n form an ascending chain

$$I_0 \subseteq I_1 \subseteq I_2 \subseteq \ldots \subseteq I_n \subseteq \ldots$$

which becomes stationary by assumption. Now let us consider an ascending chain

$$K_0 \subseteq K_1 \subseteq \ldots \subseteq K_n \subseteq \ldots$$

of ideals of $R[x;\sigma,\delta]$. Let K_{i_n} denote the corresponding leading left ideals of K_i. Since R is Noetherian, the ascending chain

$$K_{0_0} \subseteq K_{1_1} \subseteq K_{2_2} \subseteq \ldots \subseteq K_{n_n} \subseteq \ldots$$

becomes stationary. Let the associated index be $p_p \in \mathbb{N}$. On the other hand, for every positive integer n with $0 \leq n \leq p-1$, the chain

$$K_{0_n} \subseteq K_{1_n} \subseteq \ldots \subseteq K_{i_n} \subseteq \ldots$$

stabilizes, say at q_n. If we choose m =max$\{p, q_1, \ldots, q_{p-1}\}$, then we obtain $K_{i_n} = K_{m_n}$ for every positive integer n and every $i \geq m$. From this, we deduce that $K_i = K_m$, i.e. the above ascending chain of left $R[x;\sigma,\delta]$-ideals becomes stationary. Thus $R[x;\sigma,\delta]$ is Noetherian as desired. □

Remark 2.8. The construction we introduced for polynomials in just one variable may be iterated to gain "multivariate Ore extensions". Given a quasi-derivation $(\tilde{\sigma}, \tilde{\delta})$ on an Ore extension $R[x;\sigma,\delta]$ of R, then one may associate to it the Ore-extension $R[x_1;\sigma,\delta][x_2;\tilde{\sigma},\tilde{\delta}]$. This procedure can be repeated arbitrarily often.

Remark 2.9. Looking back to Example 1.7, we can now see that the "modified" ring $\mathbb{C}[t]$ was nothing but a Ore extension of $\mathbb{C}$ with the quasi-derivation (σ, δ), $\sigma : \mathbb{C} \to \mathbb{C}, z \mapsto \overline{z}$, $\delta \equiv 0$.

Armed with these results, we are now able to define the ring on which we will put our main focus, the Weyl algebra.

Definition 2.10. Let $\mathbb{K}$ be a field of characteristic zero. Then $A_n(\mathbb{K})$, the nth Weyl algebra, is defined as

$$\begin{aligned} A_n := A_n(\mathbb{K}) := \mathbb{K}\langle x_1, \ldots, x_n, y_1, \ldots, y_n | \quad & y_j x_i - x_i y_j = \delta_{ij}, \\ & x_i x_j - x_j x_i = y_i y_j - y_j y_i = 0\rangle, \end{aligned}$$

where δ_{ij} denotes the Kronecker Delta. $A_0(\mathbb{K})$ will be equal to $\mathbb{K}$ by convention.

Remark 2.11. Mentioning "the" Weyl algebra will always mean the first Weyl algebra A_1.

Remark 2.12. In Example 1.1, we have said that the differential operator given in the Gauss hypergeometric equation can be seen as an element in the Weyl algebra. One can introduce the following linear operators on $\mathbb{K}[x]$:

$$\mathcal{X} : \mathbb{K}[x] \to \mathbb{K}[x], f \mapsto x \cdot f$$

$$\mathcal{D} : \mathbb{K}[x] \to \mathbb{K}[x], f \mapsto \frac{df}{dx},$$

where $\frac{df}{dx}$ denotes the formal differentiation on $\mathbb{K}[x]$. Furthermore consider $\mathcal{D} \circ \mathcal{X}$ defined by

$$(\mathcal{D} \circ \mathcal{X})(f) = \mathcal{D}(\mathcal{X}(f)).$$

The Leibniz rule delivers for all $f \in \mathbb{K}[x]$

$$\mathcal{D}\mathcal{X}f = x\frac{d}{dx}f + f = (\mathcal{X}\mathcal{D} + 1)f.$$

With this formula, we get a commutation rule for $\mathcal{D}$ and $\mathcal{X}$, namely

$$\mathcal{D}\mathcal{X} = \mathcal{X}\mathcal{D} + 1.$$

Remembering the commutation rules of the first Weyl algebra, we can substitute $\mathcal{D}$ by y and $\mathcal{X}$ by x and discover that the two concepts coincide.

Since we are going to deal with the factorization in this ring, the following properties appear to be useful:

Lemma 2.13. *The* n*th Weyl algebra is a skew polynomial ring, a Noetherian domain and fulfills the degree properties stated in Lemma 2.6.*

Proof. We will prove this using induction on $n \in \mathbb{N}$:
For $n = 1$: $\mathbb{K}[x]$ is an Ore extension of $\mathbb{K}$ endowed with the quasi-derivation (σ, δ), where σ is the identity function on $\mathbb{K}$ and $\delta \equiv 0$.

By construction, the first Weyl algebra is a skew polynomial ring. Since $\mathbb{K}$ is a domain and Noetherian, so is $\mathbb{K}[x]$ by Theorem 2.7. Now consider a pair of maps $(\tilde{\sigma}, \tilde{\delta})$, where

$$\tilde{\sigma} : \mathbb{K}[x] \to \mathbb{K}[x] : r \mapsto r$$

and

$$\tilde{\delta} : \mathbb{K}[x] \to \mathbb{K}[x] : \sum_{i=0}^{n} r_i x^i \mapsto \sum_{i=1}^{n} i \cdot r_i x^{i-1}.$$

$\tilde{\delta}$ is thus the formalized differentiation of a polynomial and is clearly a $\tilde{\sigma}$-derivation by construction. Constructing an Ore extension of $\mathbb{K}[x]$ to $\mathbb{K}[x][y; \tilde{\sigma}, \tilde{\delta}]$, we get a ring with the properties:

- $\forall k \in \mathbb{K} : xk = kx, yk = ky$
- $yx = xy + 1$

That is the first Weyl algebra. Since our $\tilde{\sigma}$ is again injective and $\mathbb{K}[x]$ is a Noetherian domain, we inherit again these properties for $\mathbb{K}[x][y; \tilde{\sigma}, \tilde{\delta}]$. The requirements of Lemma 2.6 hold and therefore the degree properties stated in this lemma are fulfilled.

Now consider the argument applying to $n \in \mathbb{N}$. To verify that it holds for $n + 1$, construct A_{n+1} analogously as in the case $n = 1$. The same arguments can be used to show that A_{n+1} is a Noetherian domain and that the degree properties in Lemma 2.6 are fulfilled. $\square$

Now that we have a concrete class of rings and some of their properties, we will start to study some further concepts that will help us to gain a way to factorize an element in this algebra.

2. Filtration and grading

Consider the multivariate polynomial ring $R[x_1, \dots, x_n], n \in \mathbb{N}$. To obtain an order in which every monomial has only finitely many strictly smaller monomials with respect to this order, there are different ways to specify what we call the degree of an element. The total degree induced for example such an order, namely the graded lexicographical order. But there is a more general way to define the degree of monomials: The weighted degree.

Definition 2.14. Let $R[x_1, \dots, x_n], n \in \mathbb{N}$ be the ring of multivariate polynomials with coefficients in R, and let $\omega \in \mathbb{Z}^n$. Then the **weighted degree** with respect to ω of a monomial $\prod_{i=1}^{n} x_i^{\alpha_i}, \alpha_i \in \mathbb{N}_0$ for all $i \in \underline{n}$ is defined by

$$\deg_\omega(\prod_{i=1}^{n} x_i^{\alpha_i}) = \sum_{i=1}^{n} \omega_i \cdot \alpha_i.$$

The degree of a nonzero polynomial is as usual defined as the maximum of the degrees of its monomials. We will further call ω the **weight vector**.

Remark 2.15. We have defined the weight vector to be an element of $\mathbb{Z}^n$. This means that we also allow degrees to be negative. Why this could be useful, will be stated later. Another observation is that the total degree is a weighted degree given by the weight vector $\omega = [1, \dots, 1] \in \mathbb{Z}^n$.

Our goal in this section will be to find a way to write the first Weyl algebra as a direct sum of additive subgroups in $A_1(\mathbb{K})$. To get this, some preliminary work has to be done. We begin with a definition.

Definition 2.16. A **filtered ring** is a ring $R = (R, +, \cdot)$ with a family $\{F_n, n \in G\}$ of subgroups of $(R, +)$, where G is a commutative ordered monoid, such that for all $(i, j) \in G \times G$:

(1) $F_i F_j \subseteq F_{i+j}$
(2) If $i < j$: $F_i \subseteq F_j$
(3) $\bigcup_n F_n = R$

The family $\{F_n\}$ is called a **G-filtration** or simply a filtration on R.

Definition 2.17. A **graded ring** is a ring $R = (R, +, \cdot)$ with a family $\{T_n, n \in G\}$ of subgroups of $(R, +)$, where G is a commutative ordered monoid, such that for all $(i, j) \in G \times G$:

(1) $T_i T_j \subseteq T_{i+j}$

(2) $\bigoplus_n T_n = R$

The family $\{T_n\}$ is called a G**-grading** or simply a grading on R. Elements of T_n are then called **homogeneous of degree** $n \in G$ (with respect to this grading).

Example 2.18. For the first Weyl algebra, there is a canonical $\mathbb{N}_0$-filtration given by

$$F_n := \{ \sum_{i+j \leq n} r_{i,j} x^i y^j | i, j \in \mathbb{N}_0, r_{i,j} \in \mathbb{K} \}.$$

Thus, the first Weyl algebra with this filtration is a filtered ring. But it is not graded, because for $i < j, i, j \in \mathbb{N}_0 : F_i \subseteq F_j$, i.e., we do not have a direct sum.

Note that an $\mathbb{N}_0$-graded ring $T = \bigoplus_n T_n$ has a canonical $\mathbb{N}_0$-filtration $\{F_n\}$ with $F_n = T_0 \oplus \ldots \oplus T_n$. Conversely from an $\mathbb{N}_0$-filtered ring $R = \bigcup_n F_n$ we can always construct an associated graded ring T. We define a grading $\{T_n\}$ by setting $T_0 = F_0$, $T_n = F_n / F_{n-1}$ for $n \in \mathbb{N}$ and $T := \bigoplus_n T_n$. To define multiplication in T, it is enough to consider the multiplication of homogeneous elements. If $a \in F_n \backslash F_{n-1}$, then a is homogeneous of degree n and we write $\overline{a} = a + F_{n-1} \in T_n$. Let further $c \in F_m \backslash F_{m-1}$, i.e. homogeneous of degree m, and $\overline{c} := c + F_{m-1}$. Then $\overline{ac} = ac + F_{m+n-1} \in T_{m+n}$. This is well-defined and thus T is a graded ring.

But does this ring T have the same properties as our original ring R? Unfortunately, it does not. Take Example 2.18 and define a filtration in the mentioned way. We know that in the first Weyl algebra, the commutation rule $yx = xy + 1$ holds. We have $yx, xy \in F_2 \backslash F_1$ and $1 \in F_0$. This means that we have $\overline{yx} = \overline{xy}$ now. This is a ring where $\overline{x}$ and $\overline{y}$ commute.

How can we avoid such problems? We have to deal with gradings such that we do not need factor rings. Concentrating on how we constructed our filtration on A_1, one might recognize that we made use of the total degree of a monomial in A_1. What if we change our ordering? Our task is to find a weight vector ω such that

$$\deg_\omega(yx) = \deg_\omega(xy) = \deg_\omega(1).$$

A first idea could be $\omega = [0, 0]$. Then every element in A_1 has the same degree, namely 0. Not very exciting. A second idea is $\omega = [-1, 1]$. We redefine our family $\{F_n\}$ by:

$$F_n := \{ \sum_{j-i=n} r_{i,j} x^i y^j | i, j \in \mathbb{N}_0 r_{i,j} \in \mathbb{K} \}.$$

Theorem 2.19. The $\{F_n\}$ constructed above is a grading on A_1.

Proof. We prove the conditions given in the definition step by step.

First we prove that for $i, j \in \mathbb{N}_0 : F_iF_j \subseteq F_{i+j}$. Let $p \in F_i, q \in F_j$. Then in p, every monomial has the form x^ky^l, where $k, l \in \mathbb{N}_0, l - k = i$, and in q every monomial has the form $x^{\tilde{k}}y^{\tilde{l}}$, $\tilde{k}, \tilde{l} \in \mathbb{N}, \tilde{l} - \tilde{k} = j$. Multiplying p and q means to multiply each monomial of p with each monomial of q. The monomials in the result have the form $x^ky^lx^{\tilde{k}}y^{\tilde{l}}$. The equation

$$\deg_\omega(x^ky^lx^{\tilde{k}}y^{\tilde{l}}) = \deg_\omega(x^kx^{\tilde{k}}y^ly^{\tilde{l}})$$

holds, because in every permutation step, we get by the commutation rule

$$yx = xy + 1$$

the permuted monomial plus a polynomial of the same degree as the original one. Thus $\deg_\omega(x^ky^lx^{\tilde{k}}y^{\tilde{l}}) = \deg_\omega(x^kx^{\tilde{k}}y^ly^{\tilde{l}}) = l + \tilde{l} - k - \tilde{k} = (l - k) + (\tilde{l} - \tilde{k}) = i + j$ and therefore $pq \in F_{i+j}$.

For $i \in \mathbb{N}_0$ we have $F_i \cap \sum_{j \neq i} F_j = \{0\}$. Thus every polynomial in $A_1(\mathbb{K})$ can be written as a sum of polynomials in different, linearly independent F_n. Therefore $A_1 = \bigoplus_n F_n$ and $\{F_n\}$ is a grading on A_1. □

Remark 2.20. The same holds if we choose $\omega = [u, -u]$, $u \in \mathbb{Z}$ as our weight vector.

Remark 2.21. We are still losing information about the ring we defined a grading on. But this loss does not have an effect on our constructions of the factorization algorithms. For further reading on this topic I recommend [**jac**].

The advantage of writing the first Weyl algebra as a direct sum of this form will be revealed in the next chapter. Until the end of this chapter, we will study some further properties of this algebra and give some new definitions. But before that, we should take a note on something very important: The existence of a grading on a ring R is not a generic property of a ring. That we could find one in this case, is a fortunate coincidence.

3. Investigations in Weyl algebras

From now on, we will denote the homogeneous parts of degree n, $n \in \mathbb{N}_0$, in the first Weyl algebra by $A_1^{(n)}$. We start with revealing some structures, which homogeneous polynomials have.

Lemma 2.22 (Compare with [**sst**]). In A_1, for $\theta := xy$, the following equations hold:

$$\begin{aligned} \theta x^m &= x^m\theta + mx^m \\ \theta y^m &= y^m\theta - my^m, m \in \mathbb{N}. \end{aligned}$$

PROOF. We start an induction on $m \in \mathbb{N}$.
For $m = 1$, we have

$$\theta x = xyx = x(xy+1) = xxy + x = x\theta + x,$$

$$\theta y = xyy = (yx-1)y = y\theta - y.$$

Now assume that the equations hold for one fixed $m \in \mathbb{N}$. We prove the assertion for $m+1$.

$$\begin{aligned}
\theta x^{m+1} &= \theta x^m x \\
&\overset{\text{IH}}{=} (x^m\theta + mx^m)x \\
&= x^m\theta x + mx^{m+1} \\
&= x^m(x\theta + x) + mx^{m+1} \\
&= x^{m+1}\theta + (m+1)x^{m+1}, \\
\theta y^{m+1} &= \theta y^m y \\
&\overset{\text{IH}}{=} (y^m\theta - my^m)y \\
&= y^m\theta y - my^{m+1} \\
&= y^m(y\theta - y) - my^{m+1} \\
&= y^{m+1}\theta - (m+1)y^{m+1}.
\end{aligned}$$

□

Corollary 2.23. Consider $f(\theta) := f \in \mathbb{K}[\theta]$, $\theta := xy$. Then for all $n \in \mathbb{N}$:

$$\begin{aligned}
f(\theta)x^n &= x^n f(\theta+n) \\
f(\theta)y^n &= y^n f(\theta-n)
\end{aligned}$$

PROOF. In Lemma 2.22 we have shown that for all $n \in \mathbb{N}_0$

$$\begin{aligned}
\theta x^n &= x^n\theta + x^n n \\
\theta y^n &= y^n\theta - y^n n.
\end{aligned}$$

Thus for $m, n \in \mathbb{N}_0$ we have

$$\begin{aligned}
\theta^m x^n &= x^n(\theta+n)^m \\
\theta^m y^n &= y^n(\theta-n)^m.
\end{aligned}$$

This can be applied to every monomial in f and we get the desired result. □

Lemma 2.24. In the first Weyl algebra, we have

$$(xy)^m = \sum_{i=1}^{m} c_{i,m} x^i y^i, m \in \mathbb{N}.$$

The coefficients $c_{i,m}$ are recursively defined by

$$\begin{aligned}
c_{i,m} &= 0 \text{ for } i > m \\
c_{0,m} &= 0 \text{ for all } m \neq 0 \\
c_{0,0} &= 1 \\
c_{i,m+1} &= ic_{i,m} + c_{i-1,m}.
\end{aligned}$$

Proof. Again start an induction on $m \in \mathbb{N}$.

For $m = 1$ we have $xy = 1 \cdot x^1y^1$.

Now let the equation hold for $m \in \mathbb{N}$ arbitrary. We prove the correctness for $m + 1$.

$$\begin{aligned}
(xy)^{m+1} &= (xy)^m xy \\
&\overset{\text{IH}}{=} \sum_{i=1}^{m} c_{i,m}x^iy^ixy \\
&\overset{2.22}{=} \sum_{i=1}^{m} c_{i,m}x^i(xyy^i + iy^i) \\
&= \sum_{i=1}^{m} c_{i,m}x^{i+1}y^{i+1} + \sum_{i=1}^{m} c_{i,m}ix^iy^i \\
&\overset{\text{Adding zeros}}{=} \sum_{i=1}^{m+1} c_{i-1,m}x^iy^i + \sum_{i=1}^{m+1} ic_{i,m}x^iy^i \\
&= \sum_{i=0}^{m+1} c_{i,m+1}x^iy^i
\end{aligned}$$

□

Theorem 2.25. $A_1^{(0)}$ is a ring and finitely generated, as a $\mathbb{K}$-algebra, by the element xy. $A_1^{(k)}$ are finitely generated $A_1^{(0)}$-modules by the element x^k, if $k < 0$, or by y^k, if $k > 0$.

Proof. By definition, $A_1^{(0)}$ is an additive subgroup of A_1 and $A_1^{(0)}A_1^{(0)} \subseteq A_1^{(0)}$. Thus $A_1^{(0)}$ is closed under multiplication. We have $1 \in A_1^{(0)}$ and therefore $A_1^{(0)}$ is a ring as desired.

Now we have to prove that $A_1^{(0)}$ is generated by $\theta := xy$ as a $\mathbb{K}$-algebra. It suffices to show that $x^ky^k = p, k \in \mathbb{N}$ for a polynomial $p \in \mathbb{K}[\theta]$. With the argument given in Lemma 2.24, we can construct this polynomial quite easily. Thus $A_1^{(0)}$ is finitely generated as a $\mathbb{K}$-algebra as desired.

For a positive $k \in \mathbb{Z}$, we know that elements in $A_1^{(k)}$ have monomials of the form x^iy^{i+k}, where $i \in \mathbb{N}_0, i + k \geq 0$. Therefore we can generate every element of $A_1^{(k)}$ by multiplying an element of A_1^0 by y^k on the right. Thus $A_1^{(k)}$ is generated as a left $A_1^{(0)}$-module by y^k. A similar construction can be made for a negative k. Then $A_1^{(k)}$ is finitely generated by x^k as a left $A_1^{(0)}$-module. □

This result equips us with a special technique. Considering elements in $A_1^{(0)}$ as elements in $\mathbb{K}[\theta]$, we can execute our computations in a commutative ring and later map our result back to $A_1^{(0)}$. In the next chapter, we will make use of this frequently.

Another result concerning ring homomorphisms can be deduced from the following theorem. It is not of interest for factorization, but it is one of "the" characteristics of Weyl algebras.

Theorem 2.26. *The nth Weyl algebra A_n has no nontrivial two-sided ideals.*

Proof. We show this for the first Weyl algebra. The argument can be applied to A_n.

Consider a two-sided ideal $I \neq 0$ of A_1. Let $0 \neq f = \sum_{i,j\in\mathbb{N}_0} c_{i,j}x^iy^j \in I$. For every monomial $c_{i,j}x^iy^j$ with $j > 0$ we have:

$$\begin{aligned} c_{i,j}x^iy^jx - xc_{i,j}x^iy^j &= c_{i,j}x^iy^{j-1}(xy+1) - c_{i,j}x^{i+1}y^j \\ &\overset{2.22}{=} c_{i,j}x^{i+1}y^j + jc_{i,j}x^iy^{j-1} - c_{i,j}x^{i+1}y^j \\ &= jc_{i,j}x^iy^{j-1}. \end{aligned}$$

For every monomial $c_{i,0}x^i$ we have:

$$c_{i,0}x^ix - xc_{i,0}x^i = 0.$$

Because I is a two-sided ideal, we can deduce that $fx - xf \in I$. As we have seen before, the power of y in every monomial is decremented by this step. While there are monomials in f that contain y with a positive power, we have $fx - xf \neq 0$ (for this it is also necessary that $\mathbb{K}$ has characteristic zero). We repeat this procedure until we have no y in each monomial left. If the result is in $\mathbb{K}\backslash\{0\}$, we are finished. If there are still some monomials with a positive x power, then repeat the procedure with $y\tilde{f} - \tilde{f}y \in I$, where $\tilde{f}$ is the polynomial with no y-powers left. With that we will get a nonzero element of $\mathbb{K}$ in I, which means: $I = A_1$. It therefore follows, that A_1 has only trivial two-sided ideals. □

Because the kernel of a ring homomorphism is always a two-sided ideal, all ring homomorphisms defined on A_n except from $A_n \to \{0\}$ are injective. Furthermore, the question whether they are surjective, is one of today's open problems. It is called the first problem of Dixmier.

CHAPTER 3

Factorization

1. Implementations in current computer algebra systems

Nowadays there are just a few implementations concerning the factorization in noncommutative rings or a class of them, respectively. Remember that even in "nice" noncommutative rings, factorization is not unique in general (Example 1.7). We will start this chapter by getting to know two selected computer algebra systems and their implementations concerning noncommutative factorizations.

1.1. Maple. In Maple one can define a linear differential operator in one variable and factorize it. These elements are not in A_1 as defined before, because according [**mh**] they are considered to be in the differential algebra $\mathbb{C}(x)[y]$ (i.e. x is a rational argument). The package **DEtools** contains a function called **DFactor**, which delivers one possible factorization of a given polynomial.

Example 3.1. Here are a few lines of Maple code to show how to work with the noncommutative subsystem.

```
restart;
with(Ore_algebra):
A := diff_algebra([y,x]):
P1:=x*y^2-1:
P2:=y-x:
P1_P2:=skew_product(P1,P2,A):
DEtools[DFactor](P1_P2,[y,x]);
 ==> [x(y^2-1/x),y - x]
```

Thus Maple is able to deliver one factorization of an arbitrary element in A_1. This is not very powerful, but enough for example to deal with the problem given in Example 1.1. This concrete problem and how to get a solution using Maple is also discussed in [**mh**].

1.2. Reduce. Reduce includes a package to deal with noncommutative polynomials, namely NCPOLY. It is very powerful, because one can define skew polynomial rings in an arbitrary number of unknowns and Reduce is not just able to give one factorization of a polynomial in these rings, it can even compute all possible factorizations. Meanwhile Reduce became an open source software and I was able to have a look at its code. At first I tried to understand the functionality of its algorithm to factorize polynomials and present some of its ideas in this thesis. Unfortunately, it is not written very legibly, because there are just a few comments given in the code. A description of the implemented algorithm was also untraceable[1]. Then I decided to create my own factorization algorithm for the Weyl algebra and made use of REDUCE to compare my results with those delivered by REDUCE. In the meantime, I recognized that the factorization algorithm of REDUCE should be used carefully in general. An example how to use REDUCE by dealing with noncommutative rings and why its output appears to be confusing in some cases will be stated next.

Example 3.2. I have installed REDUCE 3.8 on Mac OS X Snow-Leopard 10.6.2 (just mentioned for the purpose of reconstructing. Another test was made using REDUCE 3.8 on Windows 7).

```
1: load_package "ncpoly";  //Loading the package

2: nc_setup({x,y},{y*x-x*y=1});  //Defining the
                                 //first Weyl algebra

3: p1 := x^7*y^3+11*x^6*y^2+31*x^5*y+21*x^4;

4: nc_factorize(p1);
  2  2
{x *y  - 2*x*y + 3,

 x*y + 3,

 x,

 x,

 x,

 x}

5: (x^2*y^2-2*x*y+3)*(x*y+3)*x^4; //Multiply the factors

 7  3       6  2       5         4
x *y  + 15*x *y  + 55*x *y + 49*x

6: x^4*(x*y+3)*(x^2*y^2-2*x*y+3); //This is not p1.
                                  //The other way
```

[1]The developer of the algorithm could not complete his work for personal reasons. Therefore he was not able to write a description.

```
                                          //around, too:
 7  3      6  2      5       4
x *y  + 3*x *y  - 5*x *y + 9*x
```

But the given factorization is only incorrect in one factor, namely the second one. It should be $xy - 1$ instead of $xy + 3$. All factors are correct, if we permute the output and put $xy + 3$ at the end. Reduce seems to arrange the factors in this example in a specific order which is not necessarily the correct order of the factorization.

2. Ideas for a Weyl factorization algorithm

As mentioned before, the decision to set up an own factorization algorithm for the Weyl algebras was made.

It has its origin in the fact that there is no concrete description of a possible factorization algorithm: Maple delivers a solution for the first Weyl algebra, but how this algorithm works is under lock and key. The same holds for other proprietary software like Mathematica. Ideas for factorization in special skew polynomial rings are also written down in [**dh**], but concepts there are too wide-spread to state them here. Reduce as the only open source computer algebra system offers no documentation about the implemented algorithm and its code is hardly legible. One sometimes even has to try out all permutations of a delivered factorization to find the correct one.

This chapter deals with some basic ideas we will make use of in order to construct an algorithm. First of all, we will concentrate on the first Weyl algebra and present an ansatz to get one factorization of a given polynomial there. Later we will see how this idea can be generalized to $A_n, n \in \mathbb{N}$ arbitrary.

The idea is similar to that of the so called “Hensel descent” algorithm mentioned in [**gs**]. The main difference will lie in the choice of the weight vector for the determination of the degree.

From now on, the degree of an element in the Weyl algebra is the weighted degree with respect to the weight vector $\omega = [-1, 1]$. We choose this degree because it induces a grading on A_1. This grading plays a key role in the factorization algorithm we are going to construct.

2.1. A factorization algorithm for homogeneous polynomials... With the results given in the last chapter, one can get an idea, how to find a way to factorize homogeneous polynomials.

We will start with a simple, but important lemma.

Lemma 3.3. A factorization of a homogeneous polynomial is again homogeneous.

Proof. Let $f = f_{\hat{n}} + \ldots + f_n \in \bigoplus_{i=\hat{n}}^{n} A_1^{(i)}$, $g = g_{\hat{m}} + \ldots + g_m \in \bigoplus_{i=\hat{m}}^{m} A_1^{(i)}$, $n > \hat{n}, m > \hat{m} \in \mathbb{Z}$ be two polynomials. Assume that

$$h = fg = \sum_{l=\hat{n}+\hat{m}}^{n+m} \underbrace{\sum_{i+j=l} f_i g_j}_{=:h_l}$$

is homogeneous. Moreover, suppose that f, g are not homogeneous, and without loss of generality

$$f_n, f_{\hat{n}}, g_m, g_{\hat{m}} \neq 0.$$

Thus we have

$$\begin{aligned} h_{n+m} &= f_n g_m \neq 0 \\ h_{\hat{n}+\hat{m}} &= f_{\hat{n}} g_{\hat{m}} \neq 0. \end{aligned}$$

But since h is homogeneous, the equation

$$n + m = \hat{n} + \hat{m}$$

holds. This is a contradiction to $n > \hat{n}$ and $m > \hat{m}$. Thus h could not have been homogeneous. □

Remember that we proved $A_1^{(0)}$ to be a ring in Theorem 2.25 and $A_1^{(k)}, k \in \mathbb{Z}$ to be finitely generated modules over $A_1^{(0)}$. More precisely, this means that for $k > 0$ and $g \in A_1^{(k)}$, there exists $f \in A_1^{(0)}$ such that

$$g = fy^k.$$

For $k < 0$ and $g \in A_1^{(k)}$ analogously, there exists $f \in A_1^{(0)}$ such that

$$g = fx^k.$$

Thus it suffices to deal with a factorization of this particular f.

But how can we get one? Theorem 2.25 again delivers an answer. We have proven that $A_1^{(0)}$ can be generated by $\theta := xy$. Thus we can rewrite f as a polynomial in θ. In this form, θ commutes with the elements in $\mathbb{K}$ and clearly with itself. We get a polynomial in the commutative polynomial ring $\mathbb{K}[\theta]$. Every common computer algebra system has implemented a way to factorize it. But this is not a complete factorization of the $A_1^{(0)}$-part of f. This becomes clear when we take a look at

$$f = xy + 1 = yx.$$

In $\mathbb{K}[\theta]$, f equals $\theta+1$, an irreducible polynomial. But we have $y \mid f$ (left divisor). Fortunately, we can make a statement about the factorization of the irreducible factors in $\mathbb{K}[\theta]$ when we rewrite them as elements in A_1.

Lemma 3.4. Let $f \in A_1^{(0)}$ be a monic polynomial. Rewrite f as a polynomial in $\mathbb{K}[\theta]$. Let $(f_1, \dots, f_m) \in \mathbb{K}[\theta]^m \setminus \mathbb{K}, m \in \mathbb{N}$, be irreducible factors of f as an element in $\mathbb{K}[\theta]$. If $f_i, i \in \underline{m}$, is reducible in A_1, then $f_i = kxy$ or $f_i = kyx$ for $k \in \mathbb{K}$.

Proof. Consider f_i to be monic and reducible in A_1. Then f_i is by construction still an element in $A_1^{(0)}$. Let φ, ψ be elements in A_1 with $\varphi\psi = f_i$. φ and ψ are homogeneous by Lemma 3.3. We have $\varphi, \psi \notin A_1^{(0)} \setminus \mathbb{K}$ and $\varphi \in A_1^{(k)}, \psi \in A_1^{(-k)}$ for some $k \in \mathbb{N}$. This means that $\varphi = \tilde{\varphi} y^k$ and $\psi = \tilde{\psi} x^k$ with $\tilde{\varphi}, \tilde{\psi} \in A_1^{(0)}$. It follows that $\tilde{\varphi}, \tilde{\psi}$ have to be constants, because otherwise we would have more than one nontrivial factor in $\mathbb{K}[\theta]$ by Corollary 2.23. Furthermore the only possibility is $k = 1$. The same argument can be applied to $\varphi \in A_1^{(-k)}, \psi \in A_1^{(k)}$. Thus $f_i = xy$ or $f_i = yx$ as desired. $\square$

It follows that each factor which is reducible in A_1, equals either kxy or kyx for some $k \in \mathbb{K}$. Checking all our factors with respect to this condition will lead to our final factorization of the homogeneous polynomials in A_1.

Algorithm 1 HomogFac: Factorization of a homogeneous polynomial in the first Weyl algebra

```
Input: $h \in A_1^{(m)}$, where $m \in \mathbb{Z}$
Output: $(f_1, \dots, f_n) \in A_1^n$, such that $f_1 \cdot \dots \cdot f_n = h$, $n \in \mathbb{N}$
 1: if $m \neq 0$ then
 2:    if $m < 0$ then
 3:       Get $\hat{h}$ such that $h = \hat{h}x^m$
 4:       $factor := \underbrace{x, \dots, x}_{m \text{ times}}$
 5:    else
 6:       Get $\hat{h}$ such that $h = \hat{h}y^m$
 7:       $factor := \underbrace{y, \dots, y}_{m \text{ times}}$
 8:    end if
 9: else
10:    $\hat{h} := h$
11:    $factor := 1$
12: end if
13: Transform $\hat{h}$ into the form $\hat{h} = \sum_{i \in \mathbb{N}} c_i(xy)^i$, $c_i \in \mathbb{K}$
14: $(\hat{f}_1, \dots, \hat{f}_n) :=$ factorization of the univariate polynomial $\sum_{i \in \mathbb{N}} c_i\theta^i$
15: $(f_1, \dots, f_n) :=$ Substitute $\theta$ by $xy$ in $(\hat{f}_1, \dots, \hat{f}_n)$
16: for i = 1 to n do
17:    if $f_i = xy$ then
18:       $f_i := x, y$
19:    end if
20:    if $f_i = yx$ then
21:       $f_i := y, x$
22:    end if
23: end for
24: return $(f_1, f_2, \dots, f_n, factor)$
```

Example 3.5. We will see how the algorithm works by an example. We are going to factorize the polynomial

$$h = x^2y^4 + 7xy^3 + 9y^2.$$

h is a homogeneous polynomial of degree 2. Thus we can extract the factor y^2:

$$x^2y^4 + 7xy^3 + 9y^2 = (x^2y^2 + 7xy + 9)y^2.$$

Therefore *factor*= $[y, y]$. Now rewrite $x^2y^2 + 7xy + 9$ as a polynomial in $\theta := xy$. By Lemma 2.24, we have

$$(xy)^2 = xy + x^2y^2$$

and thus

$$x^2y^2 = \theta^2 - \theta.$$

This leads to the fact that $x^2y^2 + 7xy + 9 = \theta^2 + 6\theta + 9$. This can be factorized using the binomial theorem:

$$\theta^2 + 6\theta + 9 = (\theta + 3)^2.$$

Thus our total list of factors is:

$$[1, xy + 3, xy + 3, y, y]$$

Since $xy + 3$ is irreducible in A_1, we have

$$h = (xy + 3)(xy + 3)(y)(y).$$

2.2. ... leads to a factorization algorithm for arbitrary polynomials. Now consider $h = \sum_{i=\hat{n}}^{n} h_i \in \bigoplus_{i=\hat{n}}^{n} A_1^{(i)}, \qquad n \geq \hat{n} \in \mathbb{Z}$, where h_i are homogeneous of degree i. Furthermore, suppose $x \nmid h$ and $y \nmid h$ neither on the left, nor on the right, because otherwise we would have $h = \tilde{h}x$ or $h = \tilde{h}y$ resp. $h = x\tilde{h}$ or $y\tilde{h}$, and we were just interested in the factorization of $\tilde{h}$. This is the polynomial we are going to factorize. If $\hat{n} = n$, then h is homogeneous and we can use our factorization algorithm for homogeneous polynomials. Therefore, let $\hat{n} < n$.

When we look at h_n and $h_{\hat{n}}$, we can make the following assertions.

Lemma 3.6. Using the notations from above, we have:

(1) For $i \neq 0$, $\hat{n} \leq i \leq n$, h_i is always reducible.
(2) If there do not exist $\phi, \psi \in A_1 \backslash \mathbb{K}$, such that $\phi\psi$ has h_n as maximal and $h_{\hat{n}}$ as minimal homogeneous part, then h is irreducible.

Proof. The first statement follows from the fact that for $i \neq 0$, $\hat{n} \leq i \leq n$, $A_1^{(i)}$ is an $A_1^{(0)}$-module generated by x^i resp. y^i. This means that there are at least i nontrivial factors and thus h_i is reducible.

For the second statement, let $f = f_{\hat{l}} + \ldots + f_l$ and $g = g_{\hat{k}} + \ldots + g_k$, where the summands are the homogeneous parts of f and g and $l \geq \hat{l}, k \geq \hat{k} \in \mathbb{Z}$ such that $fg = h$. Then fg can also be rewritten as

$$h = fg = \sum_{m=\hat{l}+\hat{k}}^{l+k} \sum_{i+j=m} f_i g_j = \sum_{m=\hat{l}+\hat{k}}^{l+k} h_m,$$

where h_m is the homogeneous part of h of degree m. It follows that $l + k = n$ and $\hat{l} + \hat{k} = \hat{n}$. Thus f_l and g_k are the exclusive summands of f and g such that $f_l g_k = h_n$ (the same can be said about $f_{\hat{l}}$, $g_{\hat{k}}$ and $h_{\hat{n}}$). We conclude that if we cannot find the desired ϕ and ψ, then h is irreducible. □

Remark 3.7. In the second statement of the previous Lemma, note that ϕ and ψ need not to be a factorization of h. They just have to factorize the maximal and the minimal homogeneous parts. To find these factors can be much easier than to find a factorization of h.

Now assume that h_n and $h_{\hat{n}}$ are both reducible. Let $f_1 \cdots f_l = h_n$ and $g_1 \cdots g_k = h_{\hat{n}}$ $(l, k \in \mathbb{N})$ be factorizations, where we also allow factors to be trivial, i.e. in $\mathbb{K}$. Remember that the f_i and g_j, $i \in \underline{l}, j \in \underline{k}$, are all homogeneous due to Lemma 3.3. An assertion we can make about a factorization of h – if it exists – is that it necessarily has the form

$$h = (\hat{f}_1 + \hat{h}_1 + \hat{g}_1) \cdots (\hat{f}_\mu + \hat{h}_\mu + \hat{g}_\mu),$$

where $\mu \leq \min(l, k)$, $\hat{f}_1 \cdots \hat{f}_\mu = f_1 \cdots f_l$, $\hat{g}_1 \cdots \hat{g}_\mu = g_1 \cdots g_k$ and $\hat{h}_1, \ldots, \hat{h}_\mu$ such that the equation holds. All possibilities to link $g_1, \ldots, g_k$ with $f_1, \ldots, f_l$ in this form have to be checked. In the further text I refer to these possibilities as **combinations** $(\hat{f}_1 + \hat{g}_1, \ldots, \hat{f}_\mu + \hat{g}_\mu)$ between the pair $(g_1, \ldots, g_k)$ and $(f_1, \ldots, f_l)$. This might of course produce a huge amount of cases, but some of them lead to inconsistency. To check the validity of a combination, we can make use of the total resp. the rational degree of our polynomial.

We introduced the weighted degree $\omega = [-1, 1]$ to achieve a grading on the first Weyl algebra. But the weighted degree of a polynomial just gives us information about the highest distance an x-power has to the y-power in the same monomial. Additionally, we know that there is a maximal total degree of h, which our combinations of $f_1, \ldots, f_l$ with $g_1, \ldots, g_k$ should not exceed. But there is also another degree that can help us to decide the validity of a combination.

Definition 3.8. The **rational degree in** y of $\varphi \in A_1$ is defined by the weighted degree

$$\deg_y(\varphi)$$

using the weight vector $\omega = [0, 1]$. $\deg_x(\varphi)$ is defined analogously.

Thus the rational degree delivers information about the highest power x resp. y can have in h. Checking whether a combination of $f_1, \dots, f_l$ with $g_1, \dots, g_k$ exceeds the rational degree for x or y can also eliminate invalid combinations. The last task is to check whether h_n and $h_{\hat{n}}$ are eliminated when subtracting $(\hat{f}_1 + \hat{g}_1) \cdots (\hat{f}_\mu + \hat{g}_\mu)$ from h.

Let M be the set of all possible and valid combinations found. If M is empty, then h is irreducible by Lemma 3.6. Thus consider M not to be empty. Take an element $(g_1, \dots, g_k) \in M$ and let

$$\hat{h} := h - g_1 \cdots g_k.$$

Now, as done with h before, factorize the highest and the lowest homogeneous part of $\hat{h}$. Furthermore detect if $\hat{h}$ has common factors in $\{g_1, \dots, g_k\}$, because then we would have spotted a first factorization.

Let $c_1, \dots, c_\nu$ be a factorization of the highest homogeneous part and $\hat{c}_1, \dots, \hat{c}_{\hat{\nu}}$ be a factorization of the lowest homogeneous part, i.e.

$$\hat{h} = c_1 \cdots c_\nu + \hat{c}_1 \cdots \hat{c}_{\hat{\nu}} + \varphi,$$

where $\varphi \in A_1$ and either $\deg(c_1 \cdots c_\nu) > \deg(\varphi) > \deg(\hat{c}_1 \cdots \hat{c}_{\hat{\nu}})$ or $\varphi = 0$. Put again all combinations of these factors together in one set. Let $(f_1, \dots, f_l)$ be an element in this set.

Now we combine the f_i with the g_j.

Case 1: There exists an $(i, j) \in \underline{l} \times \underline{k}$ with $f_i = g_j$.
We put all the possibilities to link $(f_1, \dots, f_{i-1})$ with $(g_1, \dots, g_{j-1})$ and $(f_{i+1}, \dots, f_l)$ with $(g_1, \dots, g_k)$ – with respect to the total degree and the rational degree in h – in a set $\hat{M}$. Here we have a possibility to separate the wheat from the chaff. If $g_1 = f_r$ for $r > 1$ or $f_r = g_k$ for $r < l$, there will be no way to link the f_i with the g_j, where $i \neq j$, because the factors do not commute in general and thus we need factors at the left side of g_1 resp. on the right side of g_k.

Case 2: There exist no $(i, j) \in \underline{l} \times \underline{k}$ with $f_i = g_j$.
If there are no common factors, there are no criteria how to reduce our possibilities of combinations. Thus we have to add every possible combination between $(f_1, \dots, f_l)$ and $(g_1, \dots, g_k)$ in a set $\hat{M}$ and check the validity of these combinations.

Now we set $M := \hat{M}$. If M is not empty and h is not completely factorized by each element of M, repeat the steps above.

In the end, we will either get $M = \emptyset$, which means that we were not able find a possible factorization in this way, or we get a set with some possible factorizations of h.

This preliminary work can be seen as a schedule. Our next aim is to solve the problems arising in every step by separate algorithms. We begin with an algorithm that computes all possibilities to combine a list of factors $(g_1, \dots, g_k), k \in \mathbb{N}$ to a list of factors $(\hat{g}_1, \dots, \hat{g}_l), l \in \mathbb{N}$ and puts the possibilities into a set. (The condition $l \leq k$ would be suggestive, but one can catch such an input by just returning the empty set). The following algorithm solves this problem:

Algorithm 2 combinekfinlf: combining k factors to l factors

Input: $(g_1, \dots, g_k) \in A_1^k$ (where $k \in \mathbb{N}$) , $l \in \mathbb{N}$, *limits* $\in \mathbb{N}_0^3$
Output: Combinations $(c_1, \dots, c_l) \in A_1^l$ of $(g_1, \dots, g_k)$
Initialization: *result* $:= \emptyset$

```
1: if $k = 0$ then
2:     return $\emptyset$
3: end if
4: if $k = l$ then
5:     return $\{(g_1, \dots, g_k)\}$
6: end if
7: if $k < l$ then
8:     return $\emptyset$
9: end if
10: if $l = 1$ then
11:     return $\{(\prod_{i=1}^{k} g_i)\}$
12: end if
13: for $i = l$ to 2 do
14:     for $j = i$ to 1 do
15:         $result := result \cup \{(g_1, \dots, g_{j-1}, c_1, \dots, c_{k-i}, g_{j+k-i+1}, \dots, g_k) |$
            $(c_1, \dots, c_{k-i}) \in$ combinekfinlf$((g_j, \dots, g_{j+k-i}), l-i+1, limits)\}$
16:     end for
17: end for
18: for $i := 2$ to $\lfloor \frac{k}{l} \rfloor$ do
19:     $result := result \cup$ combinekfinlf$((g_1 \cdots g_i, g_{i+1}, \dots, g_k), l, limits)$
20: end for
21: return $result$
```

PROOF. **Termination**:
The loops processed in this algorithm do only depend on l and k. These are constants and not changed during an instance of the algorithm. Therefore it suffices to show that the recursion used here will not cause an infinite loop. For this, we show that either the second argument for the recursion, the length of the desired list of factors, or the length of the committed list is strict smaller than the one we started the algorithm with. Thus we will terminate at the latest in the $(l + k)$-th recursion.

In line 16, we have $l - i + 1$ as the second argument of the recursion. By line 14, $i > 1$, which means $l - i + 1 < l$. In line 20, we combine at least two factors of $(g_1, \dots, g_k)$ to one. This means that the resulting list is strict smaller than the original one.
Correctness:
We prove this using induction on k (we can disregard l, because it should always be smaller than k. Thus we will always assume $1 < l < k$

arbitrary). For $k = 1$, l arbitrary we either receive the one element in the list in the case $l = 1$, and the empty set for $l > 1$. This is correct. Now assume that the algorithm works correctly for $k \in \mathbb{N}$. We show that it still works for $k + 1$.

$k + 1$: Let the algorithm be started with a list consisting of $k + 1$ elements, namely $(g_1, \ldots, g_{k+1})$. If $l = k + 1$ or $k + 1 < l$ or $l = 1$, the algorithm works correctly. Thus we assume that $k + 1 > l > 1$.

In line 15, the recursion works correctly by the induction hypothesis and we receive all combinations for $(g_j, \ldots, g_{j+k-i})$. In this loop, we get the combinations of the following form: At least one element is not multiplied by the others/another. Because of the induction hypothesis, we receive all possibilities in this form.

In the loop in line 19, we deal with the combinations of at least two and at most $\lfloor k/l \rfloor$ factors and combine them with the others. The recursion works correctly by the induction hypothesis and thus we get all combinations of $(g_1, \ldots, g_{k+1})$. These are all combinations one can find. $\square$

Example 3.9. Let $g := [g_1, g_2, g_3, g_4]$ and $l = 2$ be the input of the algorithm. We have $4 \neq 0, 4 \neq 2, 4 > 2$, and $2 \neq 1$. Thus we will start the loop beginning at $i = 2$.

$$\textit{result} := \textit{result} \cup \{[g_1, g_2 \cdot g_3 \cdot g_4], [g_1 \cdot g_2 \cdot g_3, g_4]\}$$

These are all combinations with at least one factor of the original set not combined to another.

Now we enter the second loop and add

$$\{[g_1 g_2, g_3 g_4]\}$$

to *result*. This are all possibilities to get 2 factors from the given 4 factors.

Now we need an algorithm which ignores common factors and links two lists of factors together in every possible way. This is quite easy and done by the following procedure:

Algorithm 3 merge_icf: Finding all combinations of two sets of factors ignoring common factors

Input: $(f_1, \dots, f_l) \subset A_1^l, (g_1, \dots, g_k) \subset A_1^k$, where $l, k \in \mathbb{N}$, WLOG $l \leq k$, $limits \in \mathbb{N}^3$
Output: Combinations $(c_1, \dots, c_l) \subset A_1^l$ of $(f_1, \dots, f_l)$ and $(g_1, \dots, g_k)$ as summands disregarding common factors.
Initialization: $result := \emptyset$

```
1: $temp :=$ combinekfinlf$(\{g_1, \dots, g_k\}, l, limits)$
2: for all $(s_1, \dots, s_l) \in temp$ do
3:    if $(s_1 + f_1) \cdots (s_l + f_l)$ do not exceed the given limits then
4:       $result := result \cup \{(s_1 + f_1, \dots, s_l + f_l)\}$
5:    end if
6: end for
7: return $result$
```

Proof. **Termination:** In line 1, the algorithm combinekfinlf terminates as shown in the previous proof and the set with the results is finite. The return statement just adds a finite list of elements, namely $(f_1, \dots, f_l)$, to a finite list $(s_1, \dots, s_l)$ and thus this terminates, too.
Correctness: We receive all possibilities to combine k factors into l in the first line. It suffices to add the f_i to these factors, which is done in line two. This shows that the algorithm works correctly. □

Example 3.10. Let f, g be the two vectors of same length given by:

- $f := [f_1, f_2, f_3]$
- $g := [g_1, g_2, g_3]$

The output of merge_icf would be

$$[g_1 + f_1, g_2 + f_2, g_3 + f_3].$$

The following figure illustrates how this algorithm works.

Before we set up our main algorithm for the factorization problem, two algorithms are left to discuss. We need a procedure to combine two sets of factors with respect to common factors. The following algorithm computes them.

Algorithm 4 merge_cf: Combine with respect to common factors

```
Input: $(f_1, \dots, f_l) \in A_1^l, (g_1, \dots, g_k) \in A_1^k$, where $l, k \in \mathbb{N}$, WLOG $l \leq k$, limits $\in \mathbb{N}^3$
Output: Combinations $(c_1, \dots, c_l) \in A_1^l$ of $(f_1, \dots, f_l)$ with $(g_1, \dots, g_l)$ with at least one $c_i = f_j = g_m$, $i, j, m \in \mathbb{N}$
Initialization: result := $\emptyset$
1: $M := \{(i,j) \in \underline{l} \times \underline{k} | f_i = g_j\} \setminus \{(i,j) \in \underline{l} \times \underline{k} | (i = 1 \text{ and } j > 1) \text{ or } (j = 1 \text{ and } i > 1) \text{ or } (i = l \text{ and } j < k) \text{ or } (j = k \text{ and } i < l)\}$
2: while $M \neq \emptyset$ do
3:     $temp := \{(i,j)\}$ with $(i,j) = \min(M)$
4:     $M := M \setminus temp$
5:     $result := result \cup \{(c_1, \dots, c_{i-1}, f_i, \hat{c}_1, \dots, \hat{c}_{l-i}) | (c_1, \dots, c_{i-1}) \in$ merge_icf$((f_1, \dots, f_{i-1}), (g_1, \dots, g_{j-1}), limits)$, $(\hat{c}_1, \dots, \hat{c}_{l-i}) \in$ merge_icf$((f_{i+1}, \dots, f_l), (g_{j+1}, \dots, g_k), limits)$ with respect to limits $\}$
6:     $result := result \cup \{(c_1, \dots, c_{i-1}, f_i, \hat{c}_1, \dots, \hat{c}_{l-i}) | \{c_1, \dots, c_{i-1}\} \in$ merge_icf$((f_1, \dots, f_{i-1}), (g_1, \dots, g_{j-1}), limits)$, $\{\hat{c}_1, \dots, \hat{c}_{l-i}\} \in$ merge_cf$((f_{i+1}, \dots, f_l), (g_{j+1}, \dots, g_k), limits)$ with respect to limits $\}$
7: end while
8: return result
```

Proof. **Termination:**
M is a finite set, because there are maximum lk combinations possible. In every run of the loop (line $2-7$) we always delete one element of M. The algorithm merge_icf terminates as shown before. The algorithm merge_cf is called with a strictly shorter list, therefore we can use an inductive argument. Thus we will quit the loop after $|M|$ iterations and the set *result* will be finite, which leads to the termination of the second loop.
Correctness:
After filling the set M, we operate on each element $(i,j) \in M$ in an ascending order. First, we compute all combinations of $(f_1, \dots, f_{i-1})$ with $(g_1, \dots, g_{j-1})$ and $(f_{i+1}, \dots, f_l)$ with $(g_{j+1}, \dots, g_k)$ without dealing with common factors. The correctness of this follows by the correctness of merge_icf. Now we need to deal with the combinations that respect common factors. The factors on the left hand side of our pair, i.e. $(f_1, \dots, f_{i-1})$ and $(g_1, \dots, g_{j-1})$, have either no common factors or we computed the combinations respecting common factors before (remember that we process in an ascending order). Thus we can combine them ignoring common factors. The factors on the right hand side of our pair, i.e. $(f_{i+1}, \dots, f_l)$ and $g_{j+1}, \dots, g_k$, will we combined in both ways. This way we receive all possibilities. □

Example 3.11. Let f and g be as in the example above and let $f_1 = g_1$. Then the output of merge_cf is

$$[f_1, g_2 + f_2, g_3 + f_3].$$

The following figure illustrates what is done by the algorithm and one can see the difference between merge_cf and merge_icf:

Combining these two concepts of finding combinations, we gain the following algorithm.

Algorithm 5 mergence: Finding all combinations of two sets of factors

Input: $(f_1, \dots, f_l) \in A_1^l, (g_1, \dots, g_k) \in A_1^k$, where $l, k \in \mathbb{N}$, WLOG $l \leq k$, *limits* $\in \mathbb{N}^3$
Output: Combinations $(c_1, \dots, c_l) \subset A_1^l$ of $(f_1, \dots, f_l)$ and $(g_1, \dots, g_k)$ as summands..
Initialization: *result* $:= \emptyset$

```
1: for $\hat{l} := l$ to 1 do
2:     $F$ :=combinekfinlf($(f_1, \dots, f_l), \hat{l}$)
3:     for all $(\hat{f}_1, \dots, \hat{f}_{\hat{l}}) \in F$ do
4:         $result := result \cup$ merge_icf($\{\hat{f}_1, \dots, \hat{f}_{\hat{l}}\}, \{g_1, \dots, g_k\}, limits$ )

5:         $result := result \cup$ merge_cf($\{\hat{f}_1, \dots, \hat{f}_{\hat{l}}\}, \{g_1, \dots, g_k\}, limits$)
6:     end for
7: end for
8: return $result$
```

Now we have all the tools we need and can finally draft our algorithm to find a factorization of a polynomial in the first Weyl algebra.

Algorithm 6 FacWA: Factorization of the first Weyl Algebra

```
Input: $h = \sum_{i=\hat{n}}^{n} h_i \in \bigoplus_{i=\hat{n}}^{n} A_1^{(i)}, \qquad n \geq \hat{n} \in \mathbb{Z}$
Output: result $\subseteq \{(f_1, \dots, f_l) \in (\bigoplus_{i \in \mathbb{Z}} A_1^{(i)})^l \mid l \in \mathbb{N}, h = f_1 \cdot \dots \cdot f_l\}$
Initialization: $M := \emptyset$, result $:= \emptyset$
 1: limits $:=$ [tdeg($h$),ratdeg($h$) of $x$, ratdeg($h$) of $y$]
 2: $(f_1, \dots, f_\nu) :=$ HomogFac($h_n$) $\{\nu \in \mathbb{N}\}$
 3: $(\hat{f}_1, \dots, \hat{f}_\mu) :=$ HomogFac($h_{\hat{n}}$) $\{\mu \in \mathbb{N}\}$
 4: $M := M \cup$ mergence($(f_1, \dots, f_\nu), (\hat{f}_1, \dots, \hat{f}_\mu)$,limits)
 5: Delete all invalid combinations in $M$
 6: If there are elements in $M$ that are factorizations of $h$, delete them
    in $M$ and add them to result
 7: while $M \neq \emptyset$ do
 8:   $\hat{M} := \emptyset$
 9:   for all $(g_1, \dots, g_k) \in M$ do
10:     $\hat{h} := h - g_1 \cdots g_k$
11:     if $\hat{h}$ has a factor in $\{g_1, \dots, g_k\}$ on the left resp. on the right
        then
12:       Divide $\hat{h}$ by this factor on the left resp. on the right and add
          the associated factorization with common factors to result
13:     end if
14:     $m :=$ maximal homogeneous degree of $\hat{h}$
15:     $\hat{m} :=$ minimal homogeneous degree of $\hat{h}$
16:     $(c_1, \dots, c_\nu) :=$ HomogFac($\hat{h}_m$) $\{\nu \in \mathbb{N}\}$
17:     $(\hat{c}_1, \dots, \hat{c}_\mu) :=$ HomogFac($\hat{h}_{\hat{m}}$) $\{\mu \in \mathbb{N}\}$
18:     homogtemp $:=$ mergence($(c_1, \dots, c_\nu), (\hat{c}_1, \dots, \hat{c}_\mu)$, limits)
19:     for all $(f_1, \dots, f_l) \in$ homogtemp do
20:       temp $:=$ mergence($(f_1, \dots, f_l), (g_1, \dots, g_k)$,limits)
21:     end for
22:     Filter invalid combinations in temp
23:     $\hat{M} := \hat{M} \cup$ temp
24:   end for
25:   $M := \hat{M}$
26:   If some elements in $M$ are factorizations of $h$, delete them in $M$
      and add them to result
27: end while
28: result $:=$ result $\cup \{(h)\}$
29: return result
```

Example 3.12. In order to show how the algorithm FacWa factorizes a polynomial in the first Weyl algebra, we are going to factorize

$$h = x^2y^2 + y.$$

The limits vector determined at the beginning is

$$\mathit{limits} = [4, 2, 2].$$

We have deg(h)= 1. The maximal homogeneous part is $h_1 := y$ and the minimal is $h_0 = x^2y^2$.

The factorizations of h_1 and h_0 (using HomogFac) are:

```
>homogfac(x2y2);
   [1]:
      1
   [2]:
      xy-1
   [3]:
      x
   [4]:
      y
> homogfac(y);
   [1]:
      1
   [2]:
      y
```

Merging these lists using the procedure mergence, we get 6 combinations.

```
>mergence(homogfac(x2y2),homogfac(y),limits);
[1]:
   [1]:
      1
   [2]:
      x2y2+y
[2]:
   [1]:
      x2y+1
   [2]:
      y
[3]:
   [1]:
      2
   [2]:
      x2y2+y
[4]:
   [1]:
      xy
   [2]:
      xy+y
[5]:
   [1]:
        x2y+1
    [2]:
        2y
[6]:
    [1]:
        x2y2+y
```

The first two combinations

$$[1, x^2y^2 + y].$$

and

$$[x^2y + 1, y]$$

are obviously valid. The other combinations are invalid because if we subtract their product from h, we do not eliminate the highest and the lowest homogeneous part of h.

Thus our set M will be empty, our algorithm terminates, and the factorizations above are the output of our algorithm.

2.3. Are we done? The algorithm FacWa sometimes delivers a trivial factorization, even if there is a nontrivial one, unfortunately. We will see some polynomials which cause problems and describe how one may get a nontrivial factorization of them by applying slight modifications on our algorithm.

Although the factorization in the first Weyl algebra is not unique, we rely in every step on one factorization of a homogeneous part the algorithm HomogFac delivered. This is a first weak spot. Remember that our ideas were based on the fact that a factor of the highest homogeneous part and a factor of the lowest homogeneous part of a given polynomial h must be a summand in every factor of h. If there are different factorizations of the homogeneous parts, one factorization may lead to a factorization, while another does not. Thus every possible factorization of the homogeneous parts has to be checked.

Ergo the next task will be creating an algorithm that returns all possible homogeneous factorizations. Fortunately, as will be shown next, we can use the output of HomogFac to construct all the other possibilities.

2.3.1. *Raiders of the homogeneous factorizations.* The output of HomogFac is a tuple of the form

$$(f_1, \ldots, f_n, g, \ldots, g),$$

where $f_1, \ldots, f_n \in A_1^{(0)}$, $g \in \{x, y\}$ (if we concatenate the factorizations of $xy = \theta$ or $yx = \theta + 1$ again). Considering $f_1, \ldots, f_n$ as elements in $\mathbb{K}[\theta]$, $\theta := xy$, they commute. Thus we can permute them and this will deliver the first set of different factorizations. We can achieve more if we also allow the g to permute with the f_i. The commutation rule was stated in Corollary 2.23. Note at this point, that it is impossible that a polynomial in $\mathbb{K}[\theta]$ becomes reducible in $\mathbb{K}[\theta]$ after application of the commutation rule, since it is well known, that

$$f(\theta) \in \mathbb{K}[\theta] \text{ is irreducible} \Leftrightarrow f(a\theta + b) \text{ is irreducible}, 0 \neq a, b \in \mathbb{K}$$

holds. What we can do afterwards is to go one step deeper and search for θ and $\theta + 1$ to substitute them by x, y respectively y, x.

But there are more possibilities. In every permutation $(g_1, \dots, g_n)$ of $(f_1, \dots, f_n)$, with respect to the permutation rules, we can search again for θ and $\theta + 1$. Having found a factor in this form, we substitute it by x, y resp. y, x. For the further construction assume, without loss of generality, that $g_i = \theta, i \in \underline{n}$ was the factor found.

Now we can split our list of factors in the lists $(g_1, \dots, g_{i-1}, x)$ and $(y, g_{i+1}, \dots, g_n)$. We are interested in the permutations of x with $g_1, \dots, g_{i-1}$ and y with $g_{i+1}, \dots, g_n$. The permutations of $g_1, \dots, g_{i-1}$ and $g_{i+1}, \dots, g_n$ were computed before. Thus we can just swap x with g_{i-1}, then with g_{i-2}, etc. If the element on the left hand side of x is again x, swapping causes no problem. If it is in $A_1^{(0)}$, we can use our commutation rule. If it is y, we can stop, because we have computed/will compute the other permutations in another case (consider xy as θ, we swapped θ with y in another list.). Analogously, we can deal with the second list. Putting all these lists together, we receive our remaining possibilities.

The set of all factorizations computed in this way is the set of all factorizations of a given homogeneous polynomial. This can be shown as follows:

Assume $h \in A_1^{(k)}, k \in \mathbb{Z}$ and a factorization $(f_1, \dots, f_n)$ of h, f_i irreducible for all $i \in \underline{n}$. Because h can be divided $|k|$ times by $g := x$ if $k < 0$ resp. $|k|$ times by $g := y$ if $k \geq 0$, there are k factors f_i with $f_i = g$. There will be just elements in $A_1^{(0)}$ left, and these have to be the same as the ones our algorithm delivered. Thus $(f_1, \dots, f_n)$ is one of the factorizations achieved by the mentioned permutations.

Now that our preliminary work is done, we can draft an algorithm returning all possible factorizations of a homogeneous polynomial:

Algorithm 7 HomogFacAll: All factorizations of a homogeneous polynomial in the first Weyl algebra

```
Input: $h \in A_1^{(m)}$, where $m \in \mathbb{Z}$
Output: $\{(f_1, \dots, f_n) \in A_1^n | f_1 \cdot \dots \cdot f_n = h, n \in \mathbb{N}\}$
 1: $(f_1, \dots, f_\nu, g, \dots, g) :=$ HomogFac($h$)
    $\{\nu \in \mathbb{N}_0, g \in \{x, y\}, k \in \mathbb{K}, f_i \in A_1^{(0)}$ concatenated, if needed$\}$
 2: Rewrite each $f_i$ as element in $\mathbb{K}[\theta]$, $\theta := xy$
 3: *result* $:= \{$Permutations of $(k, f_1, \dots, f_n, g, \dots, g)$ with respect to
    the commutation rules$\}$
 4: **for** $(g_1, \dots, g_n) \in$ *result* **do**
 5:   **for** $i$ from 1 to $n$ **do**
 6:     **if** $g_i = \theta$ (resp. $g_i = \theta + 1$) **then**
 7:       $g_i := x, y$
 8:       *leftpart* $:= (g_1, \dots, g_{i-1}, x)$
 9:       *rightpart* $:= (y, g_{i+1}, \dots, g_n)$
10:       swap $x$ down in the first list and $y$ up in the second
11:       put all possibilities of the left and the right part together
          and add them to *result*
12:     **end if**
13:   **end for**
14: **end for**
15: **return** *result*
```

Proof. **Termination:**

HomogFac terminates as shown before. The rewriting of the f_i as elements of $\mathbb{K}[\theta]$ is also a terminating algorithm. We can not find more than $(n + m + 1)!$ permutations. Thus *result* is a finite set and our algorithm terminates.

Correctness:

Correctness follows by our preliminary work. □

Remark 3.13. One can see the actions performed in line 3 in the algorithm above from a different angle. Let S_n be the group of all permutations on a set with $n \in \mathbb{N}$ elements. S_n can be generated by transpositions that swap two neighboring entries, i.e. elements of the form $(i, i+1), i \in \underline{n-1}$. Here, $(\cdot, \cdot)$ means the cycle notation. Consider a homogeneous polynomial $h \in A_1^{(k)}, k \in \mathbb{Z}$. Without loss of generality, let $k > 0$. Define $\mathcal{F} := \{(f_1, \ldots, f_n) | f_i = y \text{ or } f_i \in K[\theta]\}$ as the set of the factorizations of h we get in line 3. All lists in $\mathcal{F}$ have the same length, because when we swap two elements, no factors will vanish. Now we can define an operation of S_n on $\mathcal{F}$:

$$(i, i+1).(f_1, \ldots, f_n) :=$$

$$\begin{cases} (f_1, \ldots, f_{i-1}, f_{i+1}, f_i, f_{i+2}, \ldots, f_n), & f_i, f_{i+1} \in \mathbb{K}[\theta] \\ (f_1, \ldots, f_{i-1}, f_{i+1}, f_i(\theta - 1), f_{i+2}, \ldots, f_n), & f_i \in \mathbb{K}[\theta], f_{i+1} = y \\ (f_1, \ldots, f_{i-1}, f_{i+1}(\theta + 1), f_i, f_{i+2}, \ldots, f_n), & f_i = y, f_{i+1} \in \mathbb{K}[\theta] \\ (f_1, \ldots, f_{i-1}, f_{i+1}, f_i, f_{i+2}, \ldots, f_n), & f_i = f_{i+1} = y. \end{cases}$$

We have to prove that the action above defines an operation on the elements of $\mathcal{F}$. Our preliminary work shows that $(i, i+1).(f_1, \ldots, f_n)$ is still a factorization of h. The neutral element clearly performs no transformation on an element of $\mathcal{F}$. Now let $(i, i+1), (j, j+1)$ be two generating elements of S_n and without loss of generality $j > i$. We have to deal with different cases:

Case 1: If $j \neq i+1$, then $(i, i+1).((j, j+1).(f_1, \ldots, f_n)) = ((i, i+1)(j, j+1)).(f_1, \ldots, f_n)$, because the elements $(i, i+1)$ and $(j, j+1)$ do swap in the given list are different.

Case 2: If $j = i+1$, then we have the following possibilities:

(1) $f_i = f_{i+1} = f_{i+2} = y$ or $f_i, f_{i+1}, f_{i+2} \in \mathbb{K}[\theta]$
(2) Exactly one element of $\{f_i, f_{i+1}, f_{i+2}\}$ equals y
(3) Exactly two elements of $\{f_i, f_{i+1}, f_{i+2}\}$ equal y.

In the first possibility, we have clearly $(i, i+1).((j, j+1).(f_1, \ldots, f_n)) = ((i, i+1)(j, j+1)).(f_1, \ldots, f_n)$, because f_i, f_{i+1}, f_{i+2} commute in that case. The second and the third case lead also to this result after some calculations.

Example 3.14. We will compute all factorizations of the polynomial given in Example 3.5. Our output after applying HomogFac to the polynomial was

$$[1, xy + 3, xy + 3, y, y].$$

We set $xy := \theta$ again and obtain

$$[1, \theta + 3, \theta + 3, y, y].$$

The possible permutations of this list are (without permuting the trivial factor)

$$[1, \theta+3, y, \theta+2, y]$$
$$[1, \theta+3, y, y, \theta+1]$$
$$[1, y, \theta+2, \theta+2, y]$$
$$[1, y, \theta+2, y, \theta+1]$$
$$[1, y, y, \theta+1, \theta+1].$$

After substitution of $\theta+1$ by yx and θ by xy, no new factorizations can be found. Thus we have 6 different factorizations of h, namely

$$\begin{aligned} h &= (xy+3)(xy+3)(y)(y) \\ &= (xy+3)(y)(xy+2)(y) \\ &= (xy+3)(y)(y)(y)(x) \\ &= (y)(xy+2)(xy+2)(y) \\ &= (y)(xy+2)(y)(y)(x) \\ &= (y)(y)(y)(x)(y)(x). \end{aligned}$$

Example 3.15. For the purpose to illustrate what happens in the last step, assume that we have a factorization of the form

$$[1, \theta^2+1, \theta, \theta^2+1, y]$$

over $\mathbb{R}$. Clearly, θ is reducible and equals xy. The following figure illustrates what happens in the last step.

Thus we have four different further factorizations:

$$[1, \theta^2+1, x, y, \theta^2+1, y]$$
$$[1, x, (\theta+1)^2+1, y, \theta^2+1, y]$$
$$[1, x, (\theta+1)^2+1, (\theta+1)^2+1, y, y]$$
$$[1, \theta^2+1, x, (\theta+1)^2+1, y, y].$$

With this algorithm, a slight modification of FacWa has to be done such that it accounts for every homogeneous factorization in the (before) critical steps. Thus we have found a way to let FacWA find some more factorizations.

2.3.2. *Trivial factors are not negligible.* Consider the polynomial

$$(1+x^2y)^4$$

Here, we have 1 as our maximal homogeneous and $(x^2y)^4$ as our minimal homogeneous part. Our algorithm FacWA, as constructed above, would deliver just the trivial factorization of this polynomial. Why? 1 only has one factor, namely itself. Thus in the first step, our algorithm would try to find combinations of this one factor with those of $(x^2y)^4$. The result is only the trivial factorization. A way to fix this would be to put 1 as a factor (in every combination) as often as we have factors in $(x^2y)^4$. Then we gain our desired result. But we paid this with more calculation steps that need to be performed.

Another task would be to find all possible factorizations. In the inhomogeneous case, we cannot permute the elements as we did in the homogeneous case. Other concepts have to be found, but in this thesis we will not go deeper into this topic.

3. Factorization in the nth Weyl algebra

The factorization algorithm developed above is able to compute a factorization (at least the trivial one) of one polynomial in the first Weyl algebra. But what about $A_2, A_3, \ldots$? In this section, we will get an idea how we can construct an algorithm for the other Weyl algebras with the help of our ideas in the previous section.

Assume that we have the ability to compute all factorizations of polynomials in the first Weyl algebra. We will see a way how to find factorizations of polynomials in the second Weyl algebra, and these steps can be iterated for the nth Weyl algebra.

Let $h \in A_2$. Again, we can choose a homogeneous grading on A_2, when we define the weighted degree of the elements in A_2 given by the weight vector $\omega = [-u_1, u_1, -u_2, u_2], u_1, u_2 \in \mathbb{Z}$. For our purposes, we choose $\omega = [0, 0, -1, 1]$. The homogeneous monomials of degree n in A_2 are those of the form

$$x_1^\alpha y_1^\beta x_2^s y_2^{s+n}, \alpha, \beta, s \in \mathbb{N}_0.$$

3.1. At first the homogeneous factorization again. For h, we start again our technique from above. Compute the smallest and the greatest homogenous part of h (further denoted by $h_{\hat{n}}$ and h_n). Without loss of generality let $n \geq 0$. We can rewrite h_n such that we have a polynomial in $\theta_2 = x_2y_2$ and y_2^n with coefficients in $\mathbb{K}\langle x_1, y_1\rangle$. Since y_2^n can be found in every monomial of h_n, it can be excluded. After that, our polynomial is in $A_1[\theta_2]$. Thus θ_2 commutes with all the other elements. For this polynomial, choose a new weight vector, in which θ_2 has weight 0, x_1 has weight -1 and y_1 has weight 1. With this, we have constructed the first Weyl algebra with coefficients not in $\mathbb{K}$ but in $\mathbb{K}[\theta_2]$. Now we can compute our factorizations (notice here that we do not have a homogeneous polynomial any more) in exactly the same way, as it was done in the case where we had our coefficients in $\mathbb{K}$. After this step, we can again consider our polynomial in $A_2^{(0)}$ with $\omega = [0, 0, -1, 1]$, and compute all the possibilities to permute it with the y^n. That is: we get all factorizations of the homogeneous part.

3.2. And now the inhomogeneous one. For the general factorization, nothing needs to be done. Our factorization algorithm was based on merging all possibilities of homogeneous factorizations. Above, we found a way to compute them. Thus we are done.

3.3. We conclude. Factorizing polynomials in A_n using the way above is therefore nothing but another (more complicated and complex) way of factorizing in the first Weyl algebra. Because there are no new ideas arising in the construction of the adapted algorithms, we will not discuss detailed constructions of them here and conclude this work with some words on generalizing our concept to arbitrary graded skew polynomial rings and on the implementation of the presented factorization algorithms.

4. Generalizing to arbitrary graded skew polynomial rings

What about other graded skew polynomial rings? As mentioned before, the existence of a nontrivial grading (i.e. not induced by the zero weight vector) is not a generic property of skew polynomial rings. But there are more graded skew polynomial rings than the Weyl algebras that are of great interest. First of all, here is an example for another graded ring.

Example 3.16. Consider the shift operator S_n defined as operator on $\mathbb{K}[n]$ with the following property for $f(n) \in \mathbb{K}[n]$:

$$S_n f(n) = f(n+1).$$

Analogously to the differential operator and the Weyl algebra, one can define for the shift operator the shift algebra $\mathbb{K}\langle n, S_n | S_n n = nS_n + S_n\rangle$. The commutation rule follows from the following simple calculation. Let $f(n) \in \mathbb{K}[n]$. Then

$$\begin{array}{rcl}(S_n \cdot n) \bullet f(n) & = & S_n \bullet (n \bullet f(n)) \\ & = & (n+1) \cdot f(n+1) \\ & = & n \cdot f(n+1) + f(n+1) \\ & = & n \cdot S_n \bullet f(n) + S_n \bullet f(n) \\ & = & (n \cdot S_n + S_n) \bullet f(n).\end{array}$$

A possible weight vector ω is of the form $\omega = [0, u]$ for $u \in \mathbb{Z}$. Thus, unlike the Weyl algebra, for the shift algebra there exists a nontrivial $\mathbb{N}_0$-grading.

And now an example, where there is no nontrivial $\mathbb{Z}$-grading.

Example 3.17. Consider the difference operator Δ on $\mathbb{K}[n]$ defined by the following property for $f(n) \in \mathbb{K}[n]$:

$$\Delta f(n) = f(n) - f(n-1).$$

This example is more similar to the Weyl algebras. The associated difference algebra is given by $\mathbb{K}\langle n, \Delta | \Delta n = n\Delta + \Delta + 1\rangle$. The commutation rule follows from the following calculation.

Let $f(n) \in \mathbb{K}[n]$. Then

$$\begin{array}{rcl}(\Delta \cdot n) \bullet f(n) & = & (n+1)\cdot f(n+1) - n\cdot f(n) \text{ and}\\ n\cdot\Delta \bullet f(n) & = & n\cdot f(n+1) - n\cdot f(n)\end{array}$$

holds. Thus

$$(\Delta \cdot n) \bullet f(n) - n\cdot \Delta \bullet f(n) = f(n+1) = \Delta \bullet f(n) + f(n).$$

Here we cannot find a nontrivial weight vector ω such that we can define a grading on the difference algebra, because we have to fulfill

$$\deg_\omega(n\Delta) = \deg_\omega(\Delta) = \deg_\omega(1).$$

Thus, for our weight vector $\omega := [\omega_1, \omega_2]$, the equations

$$\omega_1 + \omega_2 = \omega_2 = 0$$

must hold. It follows that $\deg_\omega(\Delta) = 0$ and $\deg_\omega(n) = 0$.

The choice of these two examples was no coincidence. They show directly that the concept of grading is not stable under algebra isomorphisms. We will discuss this result just for the purpose of sensibilizing the reader to the fact that grading can be "forgotten" as one calls this phenomenon in category theory.

Lemma 3.18. The shift algebra is isomorphic to the difference algebra.

Proof. Define the algebra homomorphism

$$\varphi : \mathbb{K}\langle n, S_n | S_n n = nS_n + S_n\rangle \to \mathbb{K}\langle n, \Delta | \Delta n = (n+1)\Delta + 1\rangle$$

by

$$\varphi\Big(\sum_{(i,j)\in\mathbb{N}_0^2} c_{i,j} n^i S_n^j\Big) = \sum_{(i,j)\in\mathbb{N}_0^2} c_{i,j} n^i (\Delta+1)^j, \qquad c_{i,j} \in \mathbb{K}.$$

That this defines a module homomorphism is clear.

The nontrivial property is that it defines an algebra homomorphism. But this follows by

$$\begin{array}{rcl}\varphi(S_n n) & = & \varphi(nS_n + S_n)\\ & = & n(\Delta+1) + \Delta + 1\\ & = & n\Delta + n + \Delta + 1\\ & = & \Delta n + n\\ & = & (\Delta+1)n\\ & = & \varphi(S_n)\varphi(n).\end{array}$$

It remains to show that we have an isomorphism. The injectivity follows by the trivial kernel of φ, because only the 0 element can be mapped to zero. For the surjectivity, let

$$\sum_{(i,j)\in\mathbb{N}_0^2} c_{i,j} n^i \Delta^j \in \mathbb{K}\langle n, \Delta | \Delta n = (n+1)\Delta + 1\rangle.$$

Choose

$$\sum_{(i,j)\in\mathbb{N}_0^2} c_{i,j} n^i (S_n - 1)^j \in \mathbb{K}\langle n, S_n | S_n n = nS_n + S_n\rangle.$$

This is an element which is mapped to the chosen element in $\mathbb{K}\langle n, \Delta | \Delta n = (n+1)\Delta + 1\rangle$. $\square$

But let us return to the factorization topic. When we generalized our technique to find a factorization of a polynomial in the first Weyl algebra, we noticed that we just had to find a way to factorize the homogeneous parts of the polynomials. With that information, we were able to construct a factorization of arbitrary ones. Thus for our ansatz it suffices to deal with homogeneous factorizations and we get inhomogeneous case by the way.

When we deal with arbitrary graded skew polynomial rings we therefore only have to deal with their homogeneous factorization. Factorization concepts for homogeneous polynomials can be elegant, as one could see when we factorized the homogeneous elements in the first Weyl algebra.

Take as an example for another nice homogeneous factorization concept the shift algebra with the weight vector $\omega = [0, 1]$. The homogeneous polynomials of degree 0 are exactly those in $\mathbb{K}[n]$. Again they are generated by one element as a commutative $\mathbb{K}$-algebra. The homogeneous elements of degree $k, k \in \mathbb{N}_0$, are those of the form ϕS_n^k, $\phi \in \mathbb{K}[n]$. Thus they are also finitely generated as a $\mathbb{K}[n]$-module by one element, namely S_n^k. We have to factorize the homogeneous part of degree 0 and then – using the commutation rule – permute the factors with the S_n.

In a nutshell: Our concept scaled the factorization problem down to the factorization of homogeneous parts of polynomials in graded skew polynomial rings. Thus we have constructed one way to deal with the factorization problem in an arbitrary graded skew polynomial ring.

CHAPTER 4

Implementation

After having done some theoretical work, I concentrated on the implementation of these algorithms in a computer algebra system. The choice fell on SINGULAR, because it contains all the functions one needs for the implementation of the algorithms described in the last chapter. A second advantage is that the syntax – as in many current programming languages – is close to C and thus understandable for many people. So the circle of people one can discuss one's code with is wider than with other computer algebra systems. Furthermore, a noncommutative subsystem called PLURAL is implemented in SINGULAR with many useful functions for writing the algorithms. Last but not least, my supervisor Dr. rer. nat. Viktor Levandovskyy as one of the developers of PLURAL was a near source for questions when problems aroused.

In general, the algorithms given in the previous chapter are implemented one-to-one in SINGULAR. In some cases, one has to adapt the ideas to the possibilities of SINGULAR, which means that one line in the pseudocode are sometimes many lines in SINGULAR. But in the algorithms dealing with the homogeneous factorizations, a result stated in [**sst**] is used, which is not proved there:

Theorem 4.1. Let $x^m y^m \in A_1$, $m \in \mathbb{N}$, and $\theta := xy$. Then the following identities hold:

$$x^m y^m = \prod_{i=0}^{m-1}(\theta - i), \quad y^m x^m = \prod_{i=1}^{m}(\theta + i)$$

PROOF. We prove this theorem using induction on m.

For $m = 1$, we have $xy = \theta$ and $yx = xy + 1$. Now let the equations hold for $m \in \mathbb{N}$ arbitrary. Then for $m + 1$, we have

$$\begin{aligned}
x^{m+1}y^{m+1} &= x(x^m y^m)y \\
&\overset{\text{IH}}{=} x(\prod_{i=0}^{m-1}(\theta - i))y \\
&\overset{2.23}{=} \theta \prod_{i=0}^{m-1}(\theta - (i+1)) \\
&= \prod_{i=0}^{m}(\theta - i)
\end{aligned}$$

and

$$\begin{aligned}
y^{m+1}x^{m+1} &= y(y^m x^m)x \\
&\overset{\text{IH}}{=} y(\prod_{i=1}^{m}(\theta + i))x \\
&\overset{2.23}{=} (\theta + 1)\prod_{i=1}^{m}(\theta + i + 1) \\
&= \prod_{i=1}^{m+1}(\theta + i).
\end{aligned}$$

Therefore, the equations hold for an arbitrary $m \in \mathbb{N}$. □

With this result, it is much easier to rewrite a polynomial in $A_1^{(0)}$ as a polynomial in $\mathbb{K}[\theta]$ than in the way we did it in 2.24. The reason lies in the fact that we do not need to compute the different $c_{i,m}$ in order to rewrite the polynomial. For theoretical results, Lemma 2.24 was more interesting, but in practice, we make use of the formula above.

Example 4.2. We will now see an example how to use the implemented algorithms in Singular. First of all, when we start Singular, we have to load the new library ncfactor.lib and define the first Weyl algebra, further adressed by r:

```
>LIB "<Your path>/ncfactor.lib"
>ring R = 0,(x,y),Ws(-1,1);
>def r = nc_algebra(1,1);
>setring(r);
```

Now we are going to factorize a homogeneous polynomial:

```
>poly h = (x^2*y^2+1)*(x^4);
>homogfac(h);
```

If we want to get all possible factorizations, we type:

```
>poly h = (x^2*y^2+1)*(x^4);
>homogfac_all(h);
```

For an inhomogeneous polynomial, we have to write:

```
>poly h = (x^2*y^2+x)*x;
>facwa(h);
```

One can also view these examples and their outputs by writing

```
>example homogfac;
>example homogfac_all;
>example facwa;
```

The homogeneous factorization we constructed for the first Weyl algebra terminates very fast. The following polynomials were chosen and we let Reduce, Maple and our algorithm factorize them to count the seconds they need to factorize the homogeneous polynomials. In Reduce and in our implementation, we always chose the algorithm to compute all factorizations. All the computations were made on a computer with 2,33 GHz Dual Core and 2 GB RAM. Reduce version 3.8, Singular version 3.1.0 and Maple 11 were used.

We begin with the polynomial $h = (x^7y^7+x^5y^5+8x^3y^3+4)(x^{10}y^{10}+4x^5y^5+6)$:
HomogFacAll: 0.63s, and there were two results.
Reduce: Calculation stopped after more than nine hours.
Maple: 2.612s, but Maple did not find a factorization of h (this also holds for Maple 13. It just takes longer to get no result).

Another polynomial is $h = (x^4y^4 + xy)x^4$:
HomogFacAll: 1.23s, 25 different factorizations.
Reduce: Again the computation was cancelled after nine hours.
Maple: 0.26s, but the result is hardly legible: $[x^8 * y + 1/2 * x^7 * (3 + \mathit{RootOf}(Z^3 - 21 * Z^2 + 146 * Z - 335) + 2 * \mathit{RootOf}(4 * Z^2 + 143 - 42 * \mathit{RootOf}(Z^3 - 21 * Z^2 + 146 * Z - 335) + 3 * \mathit{RootOf}(Z^3 - 21 * Z^2 + 146 * Z - 335)^2)), y - 1/2 * (2 * \mathit{RootOf}(4 * Z^2 + 143 - 42 * \mathit{RootOf}(Z^3 - 21 * Z^2 + 146 * Z - 335) + 3 * \mathit{RootOf}(Z^3 - 21 * Z^2 + 146 * Z - 335)^2) - 1 - \mathit{RootOf}(Z^3 - 21 * Z^2 + 146 * Z - 335))/x, y - (-10 + \mathit{RootOf}(Z^3 - 21 * Z^2 + 146 * Z - 335))/x, y + 4/x]$.

These were examples with a very small total degree. Now we choose $h = (x^{350}y^{350} + 333x^{25}y^{25} + 44xy + 4)(x^{20}y^{20} + 15x^3y^3 + 40000)$:
HomogFacAll: 191.44s, delivering two results.
Reduce: Stopped after nine hours.
Maple: After one hour, Maple loses its connection to the kernel and stops computing.

Of course, for inhomogeneous polynomials FacWA does not work very fast because of the huge amount of combinatorial computations that have to be done. Examples pointing in this direction and the time our algorithm needs are not mentioned here, because these results are not as surprizing as the results in the homogeneous case.

CHAPTER 5

Conclusion/Future Work

What about other skew polynomial rings? For instance, skew polynomial rings where no grading can be found? Since this topic appears to be quite complex, it is not touched here. For further reading on this, I would route the reader to [**dh**]. In our special class of rings, namely the graded skew polynomial rings, we have seen the techniques with which the concept "grading" equips us for dealing with the factorization problem.

One task for future work could be to find new heuristics how to scale down the (still) huge amount of combinatorial computations done in our main algorithm. There may be some elegant ways how to decide much faster which combinations are possible and which are not. For the homogeneous case, we have a very fast solution. Ergo the future work here should just touch some details in the implementation.

The Weyl algebras have a great relevance in practice, so a factorization algorithm particulary designed for them is not too special. There are current papers dealing with factoring of linear partial differential operators in two variables, i.e. elements in A_2 ([**bk**]). Of course, a general factorization algorithm for skew polynomial rings is something more powerful, but it can appear to be slow for special problems. Another task pointing in this direction could be to improve the factorization algorithm in Reduce such that the output is always in a correct order. As said before, there is no documentation but the code available. So someone else should now work on a documentation and an improvement. Furthermore, Maple and Reduce seem to have no methods implemented to deal especially with homogeneous polynomials in our $[-1, 1]$-graded meaning. As seen in the previous chapter, one can get results faster using the techniques described in this thesis.

I will conclude this work with a discussion of a differential equation. So our circle from the beginning will be closed. Consider the differential equation

$$\left(x^3\left(\frac{d}{dx}\right)^3 + x\left(\frac{d}{dx}\right) - 1\right) \bullet f = 0.$$

After starting the factorization algorithm on the differential operator, we gain

$$\left(x\frac{d}{dx}-1\right)^3 \bullet f = 0.$$

Thus it suffices to solve differential equation above three times:

$$\begin{aligned} x\frac{d}{dx}\bullet f - f &= 0 \\ \Leftrightarrow x\frac{d}{dx}\bullet f &= f. \end{aligned}$$

The solution of this differential equation can be calculated using Bernoulli's technique of separation of variables and is $f = k_1 x$ for $k_1 \in \mathbb{R}$. Now we have to solve

$$\left(x\frac{d}{dx}-1\right) \bullet f = k_1 x.$$

Our solution for this equation is $f = k_1 \ln(x)x + k_2 x$ for $k_2 \in \mathbb{R}$. Using this approach a last time, we get $f = \frac{k_1}{2}\ln(x)^2 x + k_2(\ln(x))x + k_3 x$, $k_3 \in \mathbb{R}$, as the general solution of our ordinary differential equation. To get to this solution, we did not need to solve a differential equation of order greater than 1. One can check the correctness of this solution using MAPLE.

Acknowledgements

First of all I acknowledge Viktor Levandovskyy and Eva Zerz for the great support I received from them while writing this thesis. They always answered my questions patiently and were accessible at any time.

I acknowledge my family and my friends for their backing in the last six terms. It was a time full of hard work and great fun, and I do not want to miss it. Thank you!

Bibliography