Fakultät für Mathematik, Informatik und
Naturwissenschaften der Rheinisch-Westfälischen
Technischen Hochschule Aachen

Bachelor-Arbeit im Fach Mathematik

Konstruktive Berechnungen in Ore-lokalisierten G-Algebren

Johannes Hoffmann

März 2014

angefertigt am Lehrstuhl D für Mathematik, RWTH Aachen

Erstgutachterin:
Prof. Dr. Eva Zerz
Lehrstuhl D für Mathematik
RWTH Aachen

Zweitgutachter:
Prof. Dr. Sebastian Walcher
Lehrstuhl A für Mathematik
RWTH Aachen

Betreuer:
Dr. Viktor Levandovskyy
Lehrstuhl D für Mathematik
RWTH Aachen

Abstract

Konstruktive Berechnungen in Ore-lokalisierten G-Algebren

Lokalisierung von Ringen und Moduln ist ein wichtiges Werkzeug in der Algebra. Im Vergleich zum kommutativen Fall ist die Situation für nichtkommutative Noethersche Integritätsbereiche vielschichtiger und wird hier im Rahmen der Ore-Lokalisierung von G-Algebren betrachtet.
Diese Arbeit beantwortet die Frage, wie man die der Ore-Lokalisierung zugrunde liegende unkonstruktive Ore-Bedingung in diesem Umfeld konstruktiv berechnen kann.
Dafür wird zunächst die Theorie der Ore-Lokalisierung rekapituliert, wobei zwischen den drei wichtigsten Typen von Ore-Mengen unterschieden wird, was zu monoidalen, geometrischen und rationalen Lokalisierungen führt, deren Formalismus von V. Levandovskyy eingeführt wurde.
Anschließend wird ein algorithmischer Rahmen präsentiert, der arithmetische Operationen wie Addition, Multiplikation oder die Bestimmung von Inversen ermöglicht. Besonderes Augenmerk liegt dabei auf der möglichst allgemeinen Gestaltung der Algorithmen, wobei deutlich gemacht wird, an welchen Stellen der konkrete Typ der vorliegenden Lokalisierung unterschieden werden muss.
Parallel dazu wird eine Implementierung der Algorithmen im Computeralgebra-System Singular vorgestellt.

Constructive computations in Ore-localized G-algebras

Localization of rings and modules is an important tool in algebra. For non-commutative Noetherian domains it is much more subtle, comparing to the commutative case, and is here performed in the framework of Ore localization of G-algebras.
This thesis gives an answer to the question of how to constructively compute the inherently unconstructive Ore condition, which is the main ingredient of Ore localization, in this setting.
We revisit the theory behind Ore localization and distinguish three most common types of Ore sets, leading to monoidal, geometrical and rational localizations, whose formalism has been introduced by V. Levandovskyy.
Afterwards an algorithmic framework is introduced, which allows arithmetic operations like addition, multiplication, and the computation of multiplicative inverses. Notably, the algorithms are designed in a generality, which clearly indicates, when exactly the concretely given type of a localization comes into play.
Alongside the theory an implementation of the algorithms in the computer algebra system Singular is presented.

Inhaltsverzeichnis

Einleitung

In der kommutativen Welt ist die *Lokalisierung* $S^{-1}R$ eines kommutativen Ringes R an einer multiplikativ abgeschlossenen Teilmenge S ein theoretisch wie algorithmisch sehr gut verstandenes Hilfsmittel, welches in vielen Bereichen der Algebra Anwendung findet.
Lässt man allerdings die Kommutativität zurück und versucht diesen Vorgang zu verallgemeinern, so stellen sich sofort Probleme ein, etwa muss nicht für jede Wahl von S die Lokalisierung überhaupt existieren. Abhilfe für das Problem der Existenz schafft die zusätzliche Forderung, dass S zusätzlich die *Ore-Bedingung* erfüllen muss, die in ihre linksseitigen Version besagt, dass für alle $s \in S$ und $r \in R$ Elemente $\tilde{s} \in S$ und $\tilde{r} \in R$ existieren, für die $\tilde{s}r = \tilde{r}s$ gilt; solche Lokalisierungen nennt man *Ore-Lokalisierungen*.
Damit ist das Problem von einem theoretischen Standpunkt aus gelöst und man kann eine ähnliche Theorie wie im kommutativen Fall erarbeiten.
Von einem algorithmischen Standpunkt aus wirft diese Lösung hingegen weitere Fragen auf, da die Ore-Bedingung als unkonstruktive Existenzaussage keine direkte Möglichkeit zur Implementierung schon der einfachsten Funktionen wie Addition oder Multiplikation nahelegt.
Im Rahmen einer Familie von nichtkommutativen Algebren, den *G-Algebren*, werden in dieser Arbeit Theorie und Algorithmen entwickelt, die ein konstruktives Rechnen in drei Typen von Ore-Lokalisierungen ermöglichen. Parallel dazu wird die Implementierung dieser Algorithmen in der Bibliothek `olga.lib` im Computeralgebra-System SINGULAR dokumentiert.
In Kapitel 1 werden zunächst einige grundlegende Notationen festgehalten sowie die Theorie der $\mathbb{N}_0^n$-Monoideale angerissen, bevor in Kapitel 2 die G-Algebren rekapituliert werden. Insbesondere wird dabei auf die Gröbnerbasis-Theorie auf G-Algebren eingegangen, die die Grundlage für die Algorithmen in den folgenden Kapiteln bildet.
Kapitel 3 ist dem Konzept der Ore-Lokalisierung gewidmet, hier werden auch die drei Typen *monoidal*, *geometrisch* sowie *rational* definiert, eine Klassifizierung, die in dieser Form erstmals von V. Levandovskyy beschrieben wurde.
Kapitel 4 beschäftigt sich mit *entgegengesetzten* Ringen und Lokalisierungen, die es ermöglichen, die ausschließlich für Links-Strukturen erarbeitete Theorie auch auf Rechts-Strukturen anwenden zu können.
In Kapitel 5 werden dann die Algorithmen vorgestellt, für die zwischen den drei Lokalisierungstypen unterschieden werden muss. Der erste Abschnitt beschäftigt sich mit der konstruktiven Berechnung der Ore-Bedingung und dem dafür benötigten Schnitt eines Ideals im Ring R mit der Ore-Menge S. Danach wird das Problem des Kürzen von Brüchen besprochen, bevor der dritte Abschnitt sich mit der für Gröbnerbasen relevanten Reduktion befasst.
Kapitel 6 stellt den algorithmischen Rahmen für konstruktive arithmetische Operationen in Ore-Lokalisierungen vor und dient gleichzeitig als eine Art Handbuch für die Bibliothek `olga.lib` mit ausführlichen Beispielen zur Benutzung. Um eine bessere Kompatibilität mit den restlichen SINGULAR-Bibliotheken zu ermöglichen, ist dieses Kapitel in Englisch verfasst.
Das abschließende Kapitel 7 gibt einen Ausblick auf weiterführende Fragestellungen und Probleme.

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich bei der Anfertigung meiner Bachelorarbeit unterstützt haben, insbesondere bei den Folgenden:
Ganz besonderer Dank gebührt meinem Betreuer Viktor Levandovskyy für seine Geduld, sein Engagement, seine schier unerschöpflichen Ideen zur Problemlösung und natürlich die reichliche Versorgung mit Tee während der zahlreichen Besprechungen.
Weiterhin möchte ich mich bei Prof. Eva Zerz bedanken, nicht nur für die Begutachtung dieser Arbeit, sondern auch dafür, dass sie im Verlauf vieler Vorlesungen und einiger Seminare in mir das Interesse an der Algebra geweckt hat.
Prof. Sebastian Walcher bin ich dankbar, dass er sich bereiterklärt hat, die Zweitbegutachtung dieser Arbeit zu übernehmen.
Abschließend gilt mein Dank Andre Ranft für das Korrekturlesen dieser Arbeit sowie für die zahlreichen Diskussionen und die langjährige Zusammenarbeit über den ganzen Zeitraum des Bachelorstudiums.

1. Konvention, Notation und Grundlagen

Konvention. Sei im Folgenden $\underline{n} := \{i \in \mathbb{N} \mid i \leq n\} = \{1, \ldots, n\}$ für $n \in \mathbb{N}$.

Konvention. Für einen Ring R bezeichne R^* die Einheitengruppe von R.

Definition 1.1. Für $i, j \in \mathbb{N}$ definiert man das *Kronecker-Delta* als

$$\delta_{i,j} := \begin{cases} 1 & \text{falls } i = j, \\ 0 & \text{sonst.} \end{cases}$$

Definition 1.2. Auf $\mathbb{N}_0^n$ bezeichne $\leq_c$ die komponentenweise Ordnung: Für $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}_0^n$ und $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}_0^n$ gilt

$$\alpha \leq_c \beta \quad \Leftrightarrow \quad \alpha_i \leq \beta_i \text{ für alle } i \in \underline{n}.$$

Konvention. Zur Vereinfachung der Notation wird im Folgenden häufig die Multiindex-Schreibweise benutzt: Für nicht notwendigerweise kommutative Variablen $x_1, \ldots, x_n$ sowie Exponenten $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}_0^n$ schreibt man $x^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$. Weiterhin sei $|\alpha| := \alpha_1 + \ldots + \alpha_n$.

1.1. $\mathbb{N}_0^n$-Monoideale

Definition 1.3. Eine nichtleere Teilmenge $E \subseteq \mathbb{N}_0^n$ heißt *Monoid-Ideal* oder kurz *Monoideal* von $\mathbb{N}_0^n$, wenn

$$E + \mathbb{N}_0^n := \bigcup_{\alpha \in E} (\alpha + \mathbb{N}_0^n) := \{\alpha + \gamma \mid \alpha \in E, \gamma \in \mathbb{N}_0^n\} = E.$$

Ist $B \subseteq \mathbb{N}_0^n$ nichtleer, so heißt $B + \mathbb{N}_0^n$ das von B *erzeugte* Monoideal. Ist E ein $\mathbb{N}_0^n$-Monoideal und gilt $E = B + \mathbb{N}_0^n$, so heißen die Elemente von B *Erzeuger* von E.

Satz 1.4. *Jedes $\mathbb{N}_0^n$-Monoideal E besitzt eine eindeutige minimale endliche Menge von Erzeugern.*

Beweis: Die Existenz einer endlichen Menge von Erzeugern B folgt aus dem Lemma von Dickson (etwa Lemma 1.10 in [BGTV03]); diese kann minimal gewählt werden in der Hinsicht, dass keine echte Teilmenge von B schon E erzeugt. Nimmt man an, dass es zwei solche Erzeugermengen gibt, so erhält man schnell beidseitige Inklusion und damit Gleichheit. □

Definition 1.5. Für $m \in \mathbb{N}$ operiere das Monoid $\mathbb{N}_0^n$ auf $\mathbb{N}_0^n \times \underline{m}$ mittels

$$\mathbb{N}_0^n \times (\mathbb{N}_0^n \times \underline{m}) \to \mathbb{N}_0^n \times \underline{m}, \quad \alpha + (\beta, i) := (\alpha + \beta, i).$$

Somit lässt sich Definition 1.3 auf Teilmengen von $\mathbb{N}_0^n \times \underline{m}$ erweitern: Eine nichtleere Teilmenge $E \subseteq \mathbb{N}_0^n \times \underline{m}$ heißt $\mathbb{N}_0^n$-*Monoideal*, falls $\mathbb{N}_0^n + E = E$ gilt.

Bemerkung 1.6. Die Aussage von Satz 1.4 gilt auch für Monoideale nach Art von Definition 1.5, hierzu bestimmt man die Erzeugermengen der einzelnen Komponenten und vereinigt.

2. G-Algebren

Definition 2.1. Sei K ein Körper. Sei A eine K-Algebra, gegeben durch

$$A := K\langle x_1, \ldots, x_n \mid \{x_j x_i = c_{i,j} \cdot x_i x_j + d_{i,j}\} \text{ für } 1 \leq i < j \leq n\rangle,$$

wobei die $c_{i,j} \in K^*$ und die $d_{i,j}$ Polynome in A sind, in deren Termen keine Ausdrücke der Form $x_k x_l$ für $l < k$ vorkommen. Dann heißt A eine G-Algebra, falls:

(1) Ordnungsbedingung: Es existiert eine globale Monomordnung auf $K[x_1, \ldots, x_n]$, so dass

$$\mathrm{lm}(d_{i,j}) < x_i x_j$$

für alle $1 \leq i < j \leq n$ gilt, falls $d_{i,j} \neq 0$.

(2) Nichtentartungsbedingungen ($\mathcal{NDC}$): Für alle $1 \leq i < j < k \leq n$ reduziert sich

$$\mathcal{NDC}_{i,j,k} := c_{i,k}c_{j,k} \cdot d_{i,j}x_k - x_k d_{i,j} + c_{j,k} \cdot x_j d_{i,k} - c_{i,j} \cdot d_{i,k}x_j + d_{j,k}x_i - c_{i,j}c_{i,k} \cdot x_i d_{j,k}$$

unter den gegebenen Relationen zu Null.

Definition 2.2. Sei A eine G-Algebra. Die Menge der *Standardmonome* oder kurz *Monome* von A ist definiert als

$$\mathrm{Mon}(A) := \{x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} \mid \alpha_i \in \mathbb{N}_0\} \cong \mathbb{N}_0^n.$$

Satz 2.3. *Sei A eine G-Algebra. Dann gilt:*

(1) A hat eine Poincaré-Birkhoff-Witt-Basis *als K-Vektorraum, d.h.,* $\mathrm{Mon}(A)$ *ist eine K-Basis von A.*

(2) A ist (beidseitig) Noethersch.

(3) A ist ein Integritätsbereich.

Beweis: Theorem 4.7 in [Lev05]. □

Bemerkung 2.4. Eine bekannte G-Algebra ist die *Weyl-Algebra*, die in Abschnitt 2.2 vorgestellt wird. Weitere Beispiele finden sich in Abschnitt 2.3.

2.1. Gröbnerbasen in G-Algebren

Auf G-Algebren lässt sich eine Gröbnerbasis-Theorie aufbauen, die dem kommutativen Fall sehr ähnlich ist. Die Herangehensweise an das Thema in diesem Abschnitt folgt [Lev05], dort sind auch die hier weggelassenen Beweise zu entnehmen.

Konvention. In diesem Abschnitt sei $r \in \mathbb{N}$, K ein Körper sowie

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

eine G-Algebra. Für $i \in \underline{r}$ bezeichne e_i das Element $(0, \ldots, 0, 1, 0, \ldots, 0)$ in A^r (1 in der i-ten Komponente und sonst 0).

Definition 2.5. Die Menge der Monome von A^r ist

$$\mathrm{Mon}(A^r) := \{x^\alpha e_i \mid \alpha \in \mathbb{N}_0^n,\ i \in \underline{r}\} \cong \mathbb{N}_0^n \times \underline{r}.$$

Aufgrund der natürlichen Isomorphie identifiziert man $x^\alpha e_i$ mit dem Exponentenvektor (α, i).

Definition 2.6. Seien $\alpha, \beta \in \mathbb{N}_0^n$ und betrachte $x^\alpha, x^\beta \in \mathrm{Mon}(A)$. Man sagt x^α *teilt* x^β, falls $\alpha_i \leq \beta_i$ für alle $1 \leq i \leq n$ und schreibt $x^\alpha | x^\beta$.

Definition 2.7. Eine Totalordnung $<$ auf $\mathrm{Mon}(A)$ heißt *Monomordnung*, falls:

(1) $<$ ist eine Wohlordnung auf $\mathrm{Mon}(A)$, d.h., jede nichtleere Teilmenge von $\mathrm{Mon}(A)$ hat ein kleinstes Elemente bezüglich $<$.

(2) Für alle $x^\alpha, x^\beta \in \mathrm{Mon}(A)$ folgt aus $x^\alpha < x^\beta$ schon $x^{\alpha+\gamma} < x^{\beta+\gamma}$ für alle $\gamma \in \mathbb{N}_0^n$.

(3) Für alle $p, q, s, r \in \mathrm{Mon}(A)$ folgt aus $s = p \cdot t \cdot q$ und $s \neq t$ schon $t < s$.

Definition 2.8. Sei $<$ eine Monomordnung auf A. Eine Monom-(Modul-)Ordnung auf A^r ist eine Totalordnung $<_m$ auf $\mathrm{Mon}(A^r)$, so dass für alle $\alpha, \beta, \gamma \in \mathbb{N}_0^n$ sowie $i, j \in \underline{r}$

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

(2) $x^\alpha < x^\beta \ \Rightarrow\ x^\alpha e_i <_m x^\beta e_i$ bzw. $\alpha \prec \beta \ \Rightarrow\ (\alpha, i) \prec_m (\beta, i)$.

Hierbei entspricht $\prec_m$ auf der Exponentenebene der Ordnung $<_m$ auf den Monomen.

Definition 2.9. Sei $f \in A \setminus \{0\}$, etwa $f = c_\alpha x^\alpha e_i + g$ mit $c_\alpha \in K^*$ und $x^\beta e_j < x^\alpha e_i$ für alle Terme $d_\beta x^\beta e_j$ von g mit $d_\beta \neq 0$. Man definiert folgende Begriffe:

(1) Das *Leitmonom* von f: $\mathrm{lm}(f) := x^\alpha e_i \in \mathrm{Mon}(A^r)$.

(2) Den *Leitkoeffizienten* von f: $\mathrm{lc}(f) := c_\alpha \in K^*$.

(3) Den *Leitexponenten* von f: $\mathrm{le}(f) := (\alpha, i) \in \mathbb{N}_0^n \times \underline{r}$.

(4) Die *Leitkomponente* von f: $\mathrm{lcomp}(f) := i \in \underline{r}$.

Definition 2.10. Sei $f \in A \setminus \{0\}$, etwa $f = \sum_{\alpha \in \mathbb{N}_0^n} c_\alpha x^\alpha$. Der *Träger* von f ist

$$\mathrm{supp}(f) := \{\alpha \in \mathbb{N}_0^n \mid c_\alpha \neq 0\} \subseteq \mathbb{N}_0^n.$$

Damit lässt sich f schreiben als $f = \sum_{\alpha \in \mathrm{supp}(f)} c_\alpha x^\alpha$. Allgemeiner: Ist $0 \leq s \leq n-1$, so kann man für $\alpha \in \mathrm{supp}(f)$ die Vorfaktoren von $x_{s+1}^{\alpha_{s+1}} \cdots x_n^{\alpha_n}$ in einem Polynom $\tilde{c}_{(\alpha_{s+1}, \ldots, \alpha_n)}(x_1, \ldots, x_s)$ zusammenfassen und erhält die Darstellung

$$f = \sum_{\alpha \in \mathrm{supp}(f)} \tilde{c}_{(\alpha_{s+1}, \ldots, \alpha_n)}(x_1, \ldots, x_s) \cdot x_{s+1}^{\alpha_{s+1}} \cdot x_n^{\alpha_n}.$$

Setzt man $y_i := x_{s+i}$ für $1 \leq i \leq n-s$, so definiert man

$$\mathrm{supp}_y(f) := \left\{\alpha \in \mathbb{N}_0^{n-s} \mid \tilde{c}_\alpha(x_1, \ldots, x_s) \neq 0\right\}.$$

Weiterhin definiert man den *Totalgrad* von f bezüglich y als

$$\mathrm{tdeg}_y(f) := \max\left\{|\alpha| : \alpha \in \mathrm{supp}_y(f)\right\}.$$

Ist $s = 0$, so lässt man den Index y weg und schreibt $\mathrm{supp}(f)$ bzw. $\mathrm{tdeg}(f)$.

Bemerkung 2.11. Sei $<$ eine Monomordnung auf A. Dann kann $<$ unter anderem auf folgende Arten zu einer Monom-Modulordnung erweitert werden: Die Komponenten können sowohl in aufsteigender (C) als auch in absteigender (c) Reihenfolge angeordnet werden.

- Die *aufsteigende POT-Ordnung* (engl. *position over term*) ist $<^C_{\mathrm{POT}} = (C, <)$ mit

$$x^\alpha e_i <^C_{\mathrm{POT}} x^\beta e_j \quad :\Leftrightarrow \quad i < j \text{ oder } (i = j \text{ und } x^\alpha < x^\beta).$$

- Die *aufsteigende TOP-Ordnung* (engl. *term over position*) ist $<^C_{\mathrm{TOP}} = (<, C)$ mit

$$x^\alpha e_i <^C_{\mathrm{TOP}} x^\beta e_j \quad :\Leftrightarrow \quad x^\alpha < x^\beta \text{ oder } (\alpha = \beta \text{ und } i < j).$$

Analog definiert man die absteigenden Ordnungen $<^c_{\mathrm{POT}}$ und $<^c_{\mathrm{TOP}}$.

Definition 2.12. Sei $S \subseteq A^r$ nichtleer.

(1) Als *Monoidideal der Leitexponenten* bezeichnet man

$$\mathcal{L}(S) := \mathbb{N}_0^n + \{(\alpha, i) \mid \mathrm{le}(s) = (\alpha, i) \text{ für ein } s \in S\}.$$

(2) Das *Erzeugnis der Leitmonome* von S ist der K-Vektorraum

$$L(S) := {}_K\langle \{x^\alpha e_i \mid (\alpha, i) \in \mathcal{L}(S)\} \rangle \subseteq A^r.$$

Definition 2.13. Sei $<$ eine Monomordnung auf A^r, $I \subseteq A^r$ ein Links-Untermodul sowie $G \subseteq I$ eine endliche Teilmenge. Die Menge G heißt *(Links-)Gröbnerbasis* von I, falls für alle $f \in I \setminus \{0\}$ ein $g \in G$ existiert mit $\mathrm{lm}(g) | \mathrm{lm}(f)$.

Satz 2.14 (Remark 1.7 in [Lev05])**.** *Sei $<$ eine Monomordnung auf A^r, $I \subseteq A^r$ ein Links-Untermodul sowie $G \subseteq I$ eine endliche Teilmenge. Dann ist G genau dann eine Gröbnerbasis von I, wenn $L(G) = L(I)$ als Vektorräume bzw. $\mathcal{L}(G) = \mathcal{L}(I)$ als $\mathbb{N}_0^n$-Monoideale.*

Definition 2.15. Sei $S \subseteq A^r$.

(1) Die Menge S heißt *minimal*, falls $0 \notin S$ und $\mathrm{lm}(s) \notin L(S \setminus \{s\})$ für alle $s \in S$.

(2) Ein Element $f \in A^r$ heißt *(vollständig) reduziert* bezüglich S, falls kein Monom von f in $L(S)$ enthalten ist.

(3) Die Menge S heißt *reduziert*, falls:
   (i) $0 \notin S$.
   (ii) Für alle $s \in S$ ist s reduziert bezüglich $S \setminus \{s\}$.
   (iii) Für alle $s \in S$ ist $s - \mathrm{lc}(s)\,\mathrm{lm}(s)$ reduziert bezüglich S.

Definition 2.16. Sei $\mathcal{G}$ die Menge aller endlichen geordneten Teilmengen von A^r.

(1) Eine Abbildung $\mathrm{NF} : A^r \times \mathcal{G},\ (f, G) \mapsto \mathrm{NF}(f|G)$, heißt *(Links)-Normalform* auf A^r, falls für alle $f \in A^r$ und alle $G \in \mathcal{G}$ gilt:
   (i) $\mathrm{NF}(0|G) = 0$.

(ii) Ist $\mathrm{NF}(f|G) \neq 0$, so ist $\mathrm{lm}(\mathrm{NF}(f|G)) \notin L(G)$.

(iii) $f - \mathrm{NF}(f|G) \in {}_A\langle G\rangle$.

Eine Normalform NF heißt *reduziert*, falls $\mathrm{NF}(f|G)$ reduziert ist bezüglich G.

(2) Sei $G = \{g_1, \ldots, g_s\} \in \mathcal{G}$. Eine Darstellung von $f \in {}_A\langle G\rangle$ der Gestalt $f = \sum_{i=1}^{s} a_i g_i$ mit $a_i \in A$ heißt *(Links-)Standard-Darstellung* von f bezüglich G, falls $\mathrm{lm}(f) \geq \mathrm{lm}(a_i g_i)$ für alle $i \in \underline{s}$ mit $a_i g_i \neq 0$ gilt.

Lemma 2.17 (Lemma 1.9 in [Lev05])**.** *Sei $I \subseteq A^r$ ein Links-Untermodul, $G \subset I$ eine Links-Gröbnerbasis von I sowie $\mathrm{NF}(\cdot|G)$ eine Links-Normalform auf A^r bezüglich G.*

(1) Für alle $f \in A^r$ gilt $f \in I$ genau dann, wenn $\mathrm{NF}(f|G) = 0$.

(2) Ist $J \subseteq A^r$ ein Links-Untermodul mit $I \subseteq J$, so impliziert $L(I) = L(J)$ schon $I = J$.

(3) Falls $\mathrm{NF}(\cdot|G)$ eine reduzierte Normalform ist, so ist sie eindeutig.

Bemerkung 2.18. Wie im kommutativen Fall definiert man nun Links-S-Polynome und gelangt so zu Algorithmen, die die Links-Normalform berechnen können, sowie zu einem Links-Buchberger-Algorithmus, mit dem Links-Gröbnerbasen bestimmt werden können. Genaueres findet sich in [Lev05].

2.1.1. Schnitt mit freien Untermoduln

Sei $<_A$ eine Monom-Wohlordnung auf A. Sei $<_m = <^c_{\mathrm{POT}}$ die absteigende POT-Ordnung bezüglich $<_A$ auf $A^r = \bigoplus_{i=1}^{r} Ae_i$, also

$$x^\alpha e_i <_m x^\beta e_j \quad \Leftrightarrow \quad j < i \text{ oder } (j = i \text{ and } x^\alpha <_A x^\beta).$$

Lemma 2.19 ([Lev05])**.** *Sei $M \subseteq A^r$ ein Links-Untermodul mit Gröbnerbasis $G = \{g_1, \ldots, g_m\}$ bezüglich $<_m$. Dann ist $G \cap \bigoplus_{i=s}^{r} Ae_i$ eine Gröbnerbasis von $M \cap \bigoplus_{i=s}^{r} Ae_i$ für alle $s \in \underline{r}$.*

Algorithmus 2.20. `elimComponent`

Input: $M \subseteq A^r$ Links-Untermodul, $I = \{i_1, \ldots, i_s\} \subseteq \{1, \ldots, r\}$
Output: Gröbnerbasis G von $M \cap \bigoplus_{i \in I} Ae_i$

```
1 begin
2 |  Wähle eine Eliminationsordnung <_I für die Komponenten e_{i_1}, ..., e_{i_s};
3 |  Berechne eine Gröbnerbasis G' von M bezüglich <_I;
4 |  G := {g ∈ G' | lcomp(g) ∈ {e_i | i ∈ I}};
5 |  return G;
6 end
```

2.1.2. Kern eines Links-Modul-Homomorphismus

Da die Berechnung des Kerns eines Links-Modul-Homomorphismus ein entscheidender Bestandteil der Algorithmen in Kapitel 5 ist, wird hier kurz darauf eingegangen, wie dies algorithmisch umsetzbar ist.

Seien $U \subseteq A^m$ und $V = {}_A\langle v_1, \ldots, v_k\rangle \subseteq A^n$ Links-Untermoduln sowie $M := A^m/U$ und $N := A^n/V$ A-Links-Moduln. Weiter sei

$$\phi : A^m/U \to A^n/V, \quad e_i \mapsto \Phi_i,$$

ein A-Links-Modul-Homomorphismus, gegeben durch eine Matrix $\Phi \in A^{n\times m}$. Betrachtet man

$$\psi : A^m \to A^n/V, \quad e_i \mapsto \Phi_i,$$

so gilt $\mathrm{Ke}(\phi) = \mathrm{NF}(\mathrm{Ke}(\psi) + U|U)$. Somit reicht es aus, nur $\mathrm{Ke}(\psi)$ zu bestimmen.

Lemma 2.21 ([Lev05])**.** *Definiere die Matrix*

$$Y := \begin{pmatrix} \Phi & V \\ I_{m\times m} & 0_{m\times k} \end{pmatrix} \in A^{(n+m)\times(m+k)}.$$

Sei $Z := YA^{m+k} \cap \bigoplus_{i=n+1}^{n+m} Ae_i$*. Dann ist* $\mathrm{Ke}(\phi) = \mathrm{NF}(Z + U|U) \subseteq M$*.*

Algorithmus 2.22. `modulo`

Input: $\Psi \in A^{n\times m}$, $V \in A^{n\times k}$
Output: Matrix K als Darstellung von $\mathrm{Ke}(\psi)$

```
1 begin
2 |  Y := ( Φ      V      );
  |        ( I_{m×m} 0_{m×k} )
3 |  K := elimComponent(Y, {n+1, ..., n+m});
4 |  return K;
5 end
```

2.2. Weyl-Algebren

Die Weyl-Algebren sind eine bekannte Familie von G-Algebren.

Definition 2.23. Sei K ein Körper und $n \in \mathbb{N}$. Die n-te *Weyl-Algebra* über K is definiert als

$$W_n := K\langle x_1, \ldots, x_n, \partial_1, \ldots, \partial_n \mid Q\rangle$$

mit der Relationsmenge

$$Q := \{x_j x_i = x_i x_j,\ \partial_j \partial_i = \partial_i \partial_j,\ \partial_i x_j = x_j \partial_i + \delta_{i,j} \mid 1 \le i, j \le n\}.$$

Bemerkung 2.24. Mit diesen Relationen gilt für alle $f \in K[x_1, \ldots, x_n]$

$$\partial_i f - f\partial_i = \frac{\partial f}{\partial x_i}.$$

Konvention. Um mehrfache Ableitungen nach unterschiedlichen Variablen übersichtlicher zu notieren, bezeichne $\dfrac{\partial^{|\alpha|} f}{\partial x^\alpha}$ für $f \in K[x_1, \ldots, x_n] \subsetneq W_n$ und $\alpha \in \mathbb{N}_0^n$ die $|\alpha|$-fache Ableitung von f, wobei jeweils α_i-oft nach der Variablen x_i abgeleitet wird.

Lemma 2.25. *Seien $n, j \in \mathbb{N}$, $j \leq n$ und $i \in \mathbb{N}_0$ sowie $g \in K[x_1, \ldots, x_n] \setminus K \subsetneq W_n$. Dann gilt*

$$\partial_j g^{i+1} = g^i \left(g\partial_j + (i+1)\frac{\partial g}{\partial x_j} \right).$$

Beweis: Induktion über i:

(IA): $i = 0$: Es gilt $\partial_j g = g\partial_j + \frac{\partial g}{\partial x_j}$.

(IS): $i + 1 \to i + 2$: Es gilt

$$\begin{aligned}\partial_j g^{i+2} = \partial_j g^{i+1} g &= g^i \left(g\partial_j + (i+1)\frac{\partial g}{\partial x_j} \right) g = g^{i+1} \left(\partial_j g + (i+1)\frac{\partial g}{\partial x_j} \right) \\ &= g^{i+1} \left(g\partial_j + \frac{\partial g}{\partial x_j} + (i+1)\frac{\partial g}{\partial x_j} \right) = g^{i+1} \left(g\partial_j + (i+2)\frac{\partial g}{\partial x_j} \right).\end{aligned}$$

□

Lemma 2.26. *Sei $i \in \mathbb{N}_0$ und $\beta \in \mathbb{N}_0^n$ mit $i + 1 \geq |\beta| > 0$ sowie $g \in K[x_1, \ldots, x_n] \setminus K \subsetneq W_n$. Dann gilt*

$$\partial^\beta g^{i+1} = g^{i+1-|\beta|}(g^{|\beta|}\partial^\beta + v_{i+1,\beta}),$$

wobei

(1) $v_{i+1,\beta} \in W_n$,

(2) $\operatorname{tdeg}(v_{i+1,\beta}) < |\beta|$,

(3) $v_{i+1,\beta}$ *enthält nur Ableitungen von g der Form* $\dfrac{\partial^{|\alpha|} g}{\partial x^\alpha}$ *mit $\alpha \leq_c \beta$.*

Beweis: Induktion über $|\beta|$:

(IA): $|\beta| = 1$, also $\partial^\beta = \partial_j$ für ein $j \in \underline{n}$. Aus Lemma 2.25 folgt

$$\partial_j g^{i+1} = g^i \left(g\partial_j + (i+1)\frac{\partial g}{\partial x_j} \right)$$

und $v_{i+1,e_j} = (i+1)\frac{\partial g}{\partial x_j}$ erfüllt die Anforderungen.

(IS): Sei $1 \leq j \leq n$ und betrachte $\beta \to \beta + e_j$: Es gilt

$$\begin{aligned}\partial_j \partial^\beta g^{i+1} &\overset{IV}{=} \partial_j g^{i+1-|\beta|}(g^{|\beta|}\partial^\beta + v_{i+1,\beta}) = \partial_j g^{i+1}\partial^\beta + \partial_j g^{i+1-|\beta|} v_{i+1,\beta} \\ &= g^i \left(g\partial_j + (i+1)\frac{\partial g}{\partial x_j} \right) \partial^\beta + g^{i-|\beta|} \left(g\partial_j + (i+1-|\beta|)\frac{\partial g}{\partial x_j} \right) v_{i+1,\beta} \\ &= g^{i+1-|\beta+e_j|} \left(g^{|\beta+e_j|}\partial^{\beta+e_j} + v_{i+1,\beta+e_j} \right),\end{aligned}$$

wobei $v_{i+1,\beta+e_j} := (i+1)g^{|\beta|}\frac{\partial g}{\partial x_j}\partial^\beta + g\partial_j v_{i+1,\beta} + (i+1-|\beta|)\frac{\partial g}{\partial x_j} v_{i+1,\beta} \in W_n$ nur Ableitungen bis zum Grad $\beta + e_j$ enthält und $\operatorname{tdeg}(v_{i+1,\beta+e_j}) < |\beta + e_j| = |\beta| + 1$ erfüllt.

□

Lemma 2.27. *Sei $f \in W_n$, etwa $f = \sum_{\beta \in B} b_\beta \partial^\beta$ mit $d := \mathrm{tdeg}_\partial(f)$ und $b_\beta \in K[x_1, \ldots, x_n]$, sowie $g \in K[x_1, \ldots, x_n] \setminus K$ gegeben. Weiterhin sei $i \in \mathbb{N}_0$. Dann gilt*

$$f \cdot g^{d+k} = g^k \cdot r \quad \text{sowie} \quad g^{d+k} \cdot f = r' \cdot g^k$$

für gewisse $r, r' \in W_n$.

Beweis: Nach Lemma 2.26 gilt

$$\partial^\beta \cdot g^{d+k} = \partial^\beta \cdot g^{|\beta|+k} \cdot g^{d-|\beta|} = g^k (g^{|\beta|} \partial^\beta + v_{|\beta|+k,\beta}) \cdot g^{d-|\beta|}$$

und somit

$$f \cdot g^{d+k} = \sum_{\beta \in B} b_\beta \partial^\beta \cdot g^{d+k} = \sum_{\beta \in B} g^k (g^{|\beta|} \partial^\beta + v_{|\beta|+k,\beta}) g^{d-|\beta|} = g^k \cdot \sum_{\beta \in B} (g^{|\beta|} \partial^\beta + v_{|\beta|+k,\beta}) g^{d-|\beta|}.$$

Eine analoge Rechnung zeigt die zweite Eigenschaft. □

Lemma 2.28. *Sei $\beta \in \mathbb{N}_0^n$ und $g \in K[x_1, \ldots, x_n] \subsetneq W_n$. Dann gilt*

$$\partial^\beta \cdot g = \sum_{\alpha \leq_c \beta} \frac{\partial^{|\alpha|} g}{\partial x^\alpha} \partial^{\beta - \alpha}.$$

Beweis: Induktion über $|\beta|$:

(IA): $|\beta| = 1$, also $\partial^\beta = \partial_i$ für ein $i \in \underline{n}$. Es gilt $\partial_i \cdot g = g \partial_i + \frac{\partial g}{\partial x_i}$.

(IS): Sei $i \in \underline{n}$ und betrachte $\beta \to \beta + e_i$: Es gilt

$$\begin{aligned}
\partial_i \partial^\beta g &\overset{IV}{=} \partial_i \sum_{\alpha \leq_c \beta} \frac{\partial^{|\alpha|} g}{\partial x^\alpha} \partial^{\beta-\alpha} = \sum_{\alpha \leq_c \beta} \partial_i \frac{\partial^{|\alpha|} g}{\partial x^\alpha} \partial^{\beta-\alpha} = \sum_{\alpha \leq_c \beta} \left(\frac{\partial^{|\alpha|} g}{\partial x^\alpha} \partial_i + \frac{\partial^{|\alpha+e_i|} g}{\partial x^{\alpha+e_i}} \right) \partial^{\beta-\alpha} \\
&= \sum_{\alpha \leq_c \beta} \left(\frac{\partial^{|\alpha|} g}{\partial x^\alpha} \partial_i \partial^{\beta-\alpha} + \frac{\partial^{|\alpha+e_i|} g}{\partial x^{\alpha+e_i}} \partial^{\beta-\alpha} \right) = \sum_{\alpha \leq_c \beta} \frac{\partial^{|\alpha|} g}{\partial x^\alpha} \partial^{\beta+e_i-\alpha} + \sum_{\alpha \leq_c \beta} \frac{\partial^{|\alpha+e_i|} g}{\partial x^{\alpha+e_i}} \partial^{\beta-\alpha} \\
&= \sum_{\alpha \leq_c \beta} \frac{\partial^{|\alpha|} g}{\partial x^\alpha} \partial^{\beta+e_i-\alpha} + \sum_{\alpha+e_i \leq_c \beta+e_i} \frac{\partial^{|\alpha+e_i|} g}{\partial x^{\alpha+e_i}} \partial^{\beta-\alpha} = \sum_{\alpha \leq_c \beta+e_i} \frac{\partial^{|\alpha|} g}{\partial x^\alpha} \partial^{\beta+e_i-\alpha}.
\end{aligned}$$

□

Satz 2.29. *Sei $S \subsetneq K[x] \subsetneq W_n$ mit $0 \notin S$. Sei weiterhin $f \in W_n$ sowie $s \in S$. Ist $f \cdot s \in K[x]$, so gilt $f \in K[x]$.*

Beweis: Sei $f = \sum_{\beta \in \mathrm{supp}_\partial(f)} c_\beta \partial^\beta$ mit $c_\beta \in K[x]$. Es ist mit Lemma 2.28

$$\begin{aligned}
f \cdot s &= \sum_{\beta \in \mathrm{supp}_\partial(f)} c_\beta \partial^\beta \cdot s = \sum_{\beta \in \mathrm{supp}_\partial(f)} c_\beta (\partial^\beta \cdot s) = \sum_{\beta \in \mathrm{supp}_\partial(f)} c_\beta \left(\sum_{\alpha \leq_c \beta} \frac{\partial^{|\alpha|} s}{\partial x^\alpha} \partial^{\beta-\alpha} \right) \\
&= \sum_{\beta \in \mathrm{supp}_\partial(f)} \sum_{\alpha \leq_c \beta} c_\beta \frac{\partial^{|\alpha|} s}{\partial x^\alpha} \partial^{\beta-\alpha} = \sum_{\gamma \in \mathbb{N}_0^n} \underbrace{\sum_{\substack{\beta \in \mathrm{supp}_\partial(f) \\ \beta \geq_c \gamma}} c_\beta \frac{\partial^{|\beta-\gamma|} s}{\partial x^{\beta-\gamma}}}_{=: \tilde{c}_\gamma} \partial^\gamma.
\end{aligned}$$

Da $f \cdot s \in K[x]$, muss $\tilde{c}_\gamma = 0$ für alle $\gamma \in \mathbb{N}_0^n \setminus \{0\}$ gelten. Wähle $\alpha \in \operatorname{supp}_\partial(f)$ maximal bezüglich $\leq_c$, das heißt, es existiert kein $\beta \in \operatorname{supp}_\partial(f)$ mit $\beta >_c \alpha$. Dann ist

$$\tilde{c}_\alpha = \sum_{\substack{\beta \in \operatorname{supp}_\partial(f) \\ \beta \geq_c \alpha}} c_\beta \frac{\partial^{|\beta-\alpha|} s}{\partial x^{\beta-\alpha}} = c_\alpha \frac{\partial^{|\alpha-\alpha|} s}{\partial x^{\alpha-\alpha}} = c_\alpha s \neq 0,$$

da $\alpha \in \operatorname{supp}_\partial(f)$ und $s \in S$. Also muss $\alpha = 0$ gelten und somit, aufgrund der Wahl von α, $\operatorname{supp}_\partial(f) = \{0\}$, also $f \in K[x]$. □

2.3. Weitere G-Algebren

Dieser Abschnitt beschränkt sich auf G-Algebren in zwei Variablen, alle genannten Beispiele lassen sich aber problemlos auf Algebren der Größe $2n$ erweitern.

Satz 2.30 ([LKM11], Theorem 1)**.** *Seien $q \in K^*$ sowie $\alpha, \beta, \gamma \in K$ und betrachte die G-Algebra*

$$A(q, \alpha, \beta, \gamma) := K\langle x, y \mid xy = qxy + \alpha x + \beta y + \gamma \rangle$$

Dann ist $A(q, \alpha, \beta, \gamma)$ bis auf Isomorphie genau eine der sogenannten Modell-Algebren*:*

(1) Die kommutative Algebra $K[x, y] = K\langle x, y \mid y \cdot x = x \cdot y \rangle$.

(2) Die erste Weyl-Algebra $W_n = K\langle x, \partial \mid \partial \cdot x = x \cdot \partial + 1 \rangle$.

(3) Die erste Shift-Algebra *$K\langle x, s \mid s \cdot x = x \cdot s + s = (x + 1) \cdot s \rangle$.*

*(4) Die erste q-*Shift-Algebra *$K\langle x, Q \mid Q \cdot x = q \cdot x \cdot Q \rangle$.*

*(5) Die erste q-*Weyl-Algebra *$K\langle x, \partial \mid \partial \cdot x = q \cdot x \cdot \partial + 1 \rangle$.*

Bemerkung 2.31. Für zwei G-Algebren A und B über einem gemeinsamen Grundkörper K ist das Tensorprodukt von A und B über K wieder eine G-Algebra (Lemma 3.9 in [Lev05]).

Beispiel 2.32. Für einen Körper K sei $W_1 = K\langle x, \partial_x \mid \partial_x \cdot x = x \cdot \partial_x + 1 \rangle$ die erste Weyl-Algebra sowie $S_1 = K\langle y, S_y \mid S_y \cdot y = y \cdot S_y + S_y \rangle$ über K. Dann ist $C := W_1 \otimes_K S_1$ eine G-Algebra der Form

$$C = K\langle x, \partial_x, y, S_y \mid Q \rangle,$$

wobei die Relationsmenge Q die Gestalt

$$Q = \{\partial_x x = x\partial_x + 1,\ S_y y = yS_y + S_y,\ yx = xy,\ y\partial_x = \partial_x y,\ S_y x = xS_y,\ S_y \partial_x = \partial_x S_y\}$$

hat, das heißt, dass Variablen aus verschiedenen Komponenten des Tensorprodukts miteinander kommutieren. Mit dieser Algebra kann man Differential-Differenzen-Systeme betrachten, die beispielsweise in der Kontrolltheorie oft vorkommen.

3. Ore-Lokalisierung

3.1. Konstruktion und Eigenschaften

Definition 3.1. Sei R ein Ring. Eine Teilmenge $S \subseteq R$ heißt *multiplikativ abgeschlossen*, falls $1_R \in S$ und für alle $s, t \in S$ auch $s \cdot t \in S$ gilt.

Konvention. Um triviale Situationen zu vermeiden, sei im Folgenden stets $0 \notin S$.

Definition 3.2. Sei S eine multiplikativ abgeschlossene Teilmenge eines Ring R. Dann heißt S eine *Links-Ore-Menge*, falls für alle $r \in R$ und $s \in S$ Elemente $r' \in R$ und $s' \in S$ existieren mit $s'r = r's$ (Ore-Bedingung).
Gilt zusätzlich, dass für alle $r \in R$ und $s \in S$ mit $rs = 0$ ein $s' \in S$ existiert mit $s'r = 0$, so heißt S eine *Linksnennermenge*.
Analog definiert man *Rechts-Ore-Mengen* und *Rechtsnennermengen*.

Bemerkung 3.3. In einem Integritätsbereich sind Links-Ore-Mengen stets Linksnennermengen: Seien $s \in S$ und $r \in R$ mit $rs = 0$. Dann muss $r = 0$ gelten, da $s \in S$ schon $s \neq 0$ impliziert. Dann ist aber $s'r = 0$ für alle $s' \in S$, insbesondere auch $sr = 0$.
Da in dieser Arbeit nur Integritätsbereiche lokalisiert werden, reicht es aus, wenn man Links-Ore-Mengen betrachtet.

Bemerkung 3.4 ([BGTV03], Chapter 8, Proposition 1.6)**.** Ist R Noethersch, so sind Links-Ore-Mengen in R auch Linksnennermengen in R.

Konvention. Im Weiteren sollen primär Links-Objekte studiert werden, deshalb wird auf den Zusatz "Links-" häufig verzichtet.

Definition 3.5. Sei R ein Integritätsbereich und S eine Ore-Menge in R. Dann heißt ein Ring R_S zusammen mit einem Monomorphismus $\varphi : R \to R_S$ eine *(Links-)Ore-Lokalisierung* von R bezüglich S, wenn:

(1) Für alle $s \in S$ ist $\varphi(s)$ eine Einheit in R_S.

(2) Für alle $f \in R_S$ gilt $f = \varphi(s)^{-1}\varphi(r)$ für gewisse $r \in R$, $s \in S$.

Häufig bezeichnet man R_S auch mit $S^{-1}R$, die Elemente hingegen mit (s, r) oder $s^{-1}r$ für $r \in R$, $s \in S$.

Satz 3.6. *Die Ore-Lokalisierung von R an S kann wie folgt konstruiert werden: Setze $S^{-1}R := S \times R/\sim$, wobei $\sim$ die Äquivalenzrelation*

$$(s_1, r_1) \sim (s_2, r_2) \quad \Leftrightarrow \quad \exists\ \tilde{s} \in S, \exists\ \tilde{r} \in R :\ \tilde{s}s_2 = \tilde{r}s_1 \text{ und } \tilde{s}r_2 = \tilde{r}r_1$$

ist. Definiert man die Operationen

$$+ : S^{-1}R \times S^{-1}R \to S^{-1}R, \quad (s_1, r_1) + (s_2, r_2) := (\tilde{s}s_1, \tilde{s}r_1 + \tilde{r}r_2),$$

wobei $\tilde{r} \in R$ und $\tilde{s} \in S$ die Ore-Bedingung $\tilde{s}s_1 = \tilde{r}s_2$ erfüllen, sowie

$$\cdot : S^{-1}R \times S^{-1}R \to S^{-1}R, \quad (s_1, r_1) \cdot (s_2, r_2) := (\tilde{s}s_1, \tilde{r}r_2),$$

wobei $\tilde{r} \in R$ und $\tilde{s} \in S$ die Ore-Bedingung $\tilde{r}s_2 = \tilde{s}r_1$ erfüllen, so wird $(S^{-1}R, +, \cdot)$ zum Ring.

Beweis: Ein ausführlicher Beweis findet sich im Anhang (Abschnitt 8.1). An dieser Stelle sei nur darauf eingegangen, warum man die Operationen derartig definiert:

- Addition: Mit den gefundenen $\tilde{s}$ und $\tilde{r}$ lassen sich die beiden Summanden auf einen gemeinsamen Nenner bringen:

  $$(s_1, r_1) = (\tilde{s}s_1, \tilde{s}r_1) \quad \text{sowie} \quad (s_2, r_2) = (\tilde{r}s_2, \tilde{r}r_2) = (\tilde{s}s_1, \tilde{r}r_2).$$

  Da nun die Nenner äquivalent sind, werden die Zähler addiert wie von gängiger Bruchrechnung gewohnt:

  $$(s_1, r_1) + (s_2, r_2) = (\tilde{s}s_1, \tilde{s}r_1) + (\tilde{s}s_1, \tilde{r}r_2) = (\tilde{s}s_1, \tilde{s}r_1 + \tilde{r}r_2).$$

- Multiplikation: Wechselt man in die Bruchschreibweise, so würde man erwarten, dass

  $$(s_1, r_1) \cdot (s_2, r_2) = s_1^{-1}r_1 \cdot s_2^{-1}r_2$$

  gelten soll. Damit das Ergebnis als Element von $S^{-1}R$ aufgefasst werden kann, müssen r_1 und s_2^{-1} in geeigneter Weise getauscht werden. Die aus der Ore-Bedingung gewonnene Beziehung $\tilde{r}s_2 = \tilde{s}r_1$ impliziert formal $\tilde{s}^{-1}\tilde{r} = r_1 s_2^{-1}$, so dass die Rechnung wie erwartet weitergeht:

  $$(s_1, r_1) \cdot (s_2, r_2) = s_1^{-1}r_1 \cdot s_2^{-1}r_2 = s_1^{-1}\tilde{s}^{-1}\tilde{r}r_2 = (s_1\tilde{s})^{-1}(\tilde{r}r_2) = (\tilde{s}s_1, \tilde{r}r_2).$$

□

Bemerkung 3.7. Das neutrale Element der Addition in $S^{-1}R$ ist $(1_R, 0_R) = (s, 0_R)$ für alle $s \in S$. Das neutrale Element der Multiplikation in $S^{-1}R$ ist $(1_R, 1_R) = (s, s)$ für alle $s \in S$.

Lemma 3.8. *Seien $s \in S$ und $r, t \in R$. Ist $ts \in S$, so gilt $(s, r) = (ts, tr)$.*

Beweis: Damit die Gleichheit gilt, müssen $\tilde{s} \in S$ und $\tilde{r} \in R$ existieren mit $\tilde{s}ts = \tilde{r}s$ sowie $\tilde{s}tr = \tilde{r}r$. Wählt man $\tilde{s} = s$ und $\tilde{r} = st$, so ergibt sich $\tilde{s}ts = sts = \tilde{r}s$ sowie $\tilde{s}tr = str = \tilde{r}r$. □

Bemerkung 3.9. Linksbrüche sind folglich linksseitig erweiterbar, insbesondere mit Elementen aus S.

Lemma 3.10. *Sei R ein Integritätsbereich, $S \subseteq R$ eine Ore-Menge sowie $(s, r) \in S^{-1}R$.*

(1) Es gilt $(s, r) = (1, 0)$ genau dann, wenn $r = 0$.

(2) Es gilt $(s, r) = (1, 1)$ genau dann, wenn $s = r$.

Beweis: *(1)* Ist $r = 0$, so folgt $(s, r) = (1, 0)$ direkt. Sei also $(s, r) = (1, 0)$. Dann existieren $\tilde{s} \in S$ sowie $\tilde{r} \in R$ mit $\tilde{s} \cdot 1 = \tilde{r}s$ und $0 = \tilde{s} \cdot 0 = \tilde{r}r$. Da R ein Integritätsbereich ist, folgt aus der zweiten Gleichung $r = 0$ oder $\tilde{r} = 0$. In letzterem Falle folgt aber $\tilde{s} = \tilde{r}s = 0$, was ein Widerspruch zu $0 \notin S$ ist. Somit gilt $r = 0$.

(2) Ist $s = r$, so folgt $(s, r) = (1, 1)$ direkt. Sei also $(s, r) = (1, 1)$. Dann existieren $\tilde{s} \in S$ sowie $\tilde{r} \in R$ mit $\tilde{s} \cdot 1 = \tilde{r}s$ und $\tilde{s} \cdot 1 = \tilde{r}r$, also $\tilde{r}(s - r) = 0$. Da R ein Integritätsbereich ist, folgt $r = s$ oder $\tilde{r} = 0$. In letzterem Falle folgt aber $\tilde{s} = \tilde{r}s = 0$, was ein Widerspruch zu $0 \notin S$ ist. Somit gilt $r = s$.

□

Lemma 3.11. *Sei R ein Integritätsbereich und $S \subseteq R$ eine Ore-Menge. Dann ist auch $S^{-1}R$ ein Integritätsbereich.*

Beweis: Seien $(s_1, r_1), (s_2, r_2) \in S^{-1}R$ mit $(s_1, r_1) \cdot (s_2, r_2) = (1, 0)$. Dann gilt

$$(1, 0) = (s_1, r_1) \cdot (s_2, r_2) = (\tilde{s}s_1, \tilde{r}r_2),$$

wobei $\tilde{s} \in S$ und $\tilde{r} \in R$ die Bedingung $\tilde{s}r_1 = \tilde{r}s_2$ erfüllen. Mit Lemma 3.10 gilt $\tilde{r}r_2 = 0$, also $\tilde{r} = 0$ oder $r_2 = 0$. In letzterem Fall gilt $(s_2, r_2) = (s_2, 0) = (1, 0)$ und die Aussage folgt. Sei also $\tilde{r} = 0$. Dann gilt $\tilde{s}r_1 = \tilde{r}s_2 = 0$. Da $\tilde{s} \neq 0$ folgt $r_1 = 0$ und damit $(s_1, r_1) = (1, 0)$. □

Lemma 3.12. *Seien $(s_1, r_1), (s_2, r_2) \in S^{-1}R$ mit $(s_1, r_1) \cdot (s_2, r_2) = (1, 1)$. Dann gilt*

$$(s_2, r_2) \cdot (s_1, r_1) = (1, 1).$$

Beweis: Es gilt

$$(s_2, r_2) \cdot (s_1, r_1) \cdot (s_2, r_2) = (s_2, r_2) \cdot (1, 1) = (1, 1) \cdot (s_2, r_2),$$

was zu

$$((s_2, r_2) \cdot (s_1, r_1) - (1, 1)) \cdot (s_2, r_2) = (1, 0)$$

äquivalent ist. Da $(s_2, r_2) \neq (1, 0)$ gelten muss, folgt $(s_2, r_2) \cdot (s_1, r_1) = (1, 1)$. □

Lemma 3.13. *Sei $s \in S$ sowie $r \in R$. Sei R^* die Einheitengruppe von R.*

(1) Das zu (s, r) inverse Element bezüglich Addition ist $-(s, r) = (s, -r)$.

(2) Ist $r \in R^* \cup S$, so ist (s, r) invertierbar in $S^{-1}R$. In diesem Fall ist das inverse Element bezüglich Multiplikation gegeben durch

$$(s, r)^{-1} = \begin{cases} (r, s), & \text{falls } r \in S, \\ (1, xs), & \text{falls } r \in R^* \text{ mit } xr = rx = 1. \end{cases}$$

Beweis: *(1)* Nachrechnen liefert

$$(s, r) + (s, -r) = (\tilde{s}s, \tilde{s}r + \tilde{r}(-r)) = (\tilde{s}s, \tilde{s}r - \tilde{r}r) = (\tilde{s}s, (\tilde{s} - \tilde{r})r),$$

wobei $\tilde{s} \in S$ und $\tilde{r}$ die Bedingung $\tilde{s}s = \tilde{r}s$ erfüllen. Dann ist aber $(\tilde{s} - \tilde{r})s = 0$. Integritätsbereich und $s \neq 0$ liefert $\tilde{s} = \tilde{r}$, es folgt

$$(s, r) + (s, -r) = (\tilde{s}s, (\tilde{s} - \tilde{r})r) = (\tilde{s}s, 0) = (1, 0).$$

(2) Sei $r \in S$. Nachrechnen liefert

$$(s, r) \cdot (r, s) = (\tilde{s}s, \tilde{r}s),$$

wobei $\tilde{s} \in S$ und $\tilde{r} \in R$ die Bedingung $\tilde{s}r = \tilde{r}r$ erfüllen. Da $r \in S$ gilt insbesondere $r \neq 0$, es folgt $\tilde{s} = \tilde{r}$. Dann gilt

$$(s, r) \cdot (r, s) = (\tilde{s}s, \tilde{r}s) = (\tilde{s}s, \tilde{s}s) = (1, 1).$$

Sei nun $r \in R^*$ und $r^{-1} = x \in R$. Es gilt

$$(s, r) \cdot (1, xs) = (\tilde{s}s, \tilde{r}xs),$$

wobei $\tilde{s} \in S$ und $\tilde{r} \in R$ die Bedingung $\tilde{s}r = \tilde{r} \cdot 1 = \tilde{r}$ erfüllen. Dann ist

$$(s, r) \cdot (1, xs) = (\tilde{s}s, \tilde{r}xs) = (\tilde{s}s, \tilde{s}rxs) = (\tilde{s}s, \tilde{s}s) = (1, 1).$$

Sei abschließend $r \in R^* \cap S$ mit $r^{-1} = x$. Dann sind die beiden Darstellungen von $(s, r)^{-1}$ gleich:

$$(1, xs) = (r, rxs) = (r, s).$$

□

Bemerkung 3.14. Die Ore-Bedingung erlaubt es Linksbrüche in Rechtsbrüche umzuschreiben (sofern S beidseitig-Ore ist) und umgekehrt. Weiterhin existieren gemeinsame Links- und Rechtsvielfache endlicher Mengen von Elementen der Lokalisierung.

Bemerkung 3.15. Ein großes Problem beim Versuch, elementare Operationen wie Addition, Multiplikation, Kürzen oder Vergleichen zweier Elemente zu implementieren, ist die Unkonstruktivität der Ore-Bedingung: Es wird nur die Existenz von gewissen Elementen garantiert, allerdings kein Rezept zum Berechnen dieser angegeben. Darauf wird in Kapitel 5 genauer eingegangen.

3.2. Typen von Ore-Lokalisierungen

Diese Arbeit beschäftigt sich mit den folgenden drei Typen von Ore-Lokalisierungen:

3.2.1. Rationale Lokalisierung

Sei A eine G-Algebra über einem Körper K in den Variablen $x_1, \ldots, x_n, y_1, \ldots, y_m$ mit einer Relationsmenge Q derart, dass $K[x_1, \ldots, x_n]$ eine kommutative Unteralgebra von A ist. Weiter sei $S := K[x_1, \ldots, x_n] \setminus \{0\}$ eine Links-Ore-Menge in A. Als *rationale Lokalisierung* bezeichnet man die Ore-Lokalisierung von A an S. Dann ist

$$S^{-1}A \cong K(x_1, \ldots, x_n)\langle y_1, \ldots, y_m \mid Q\rangle.$$

3.2.2. Geometrische Lokalisierung

Sei $p \in K^n$ und betrachte die n-te Weyl-Algebra W_n. Das Ideal $\mathfrak{m}_p := \langle x_1 - p_1, \ldots, x_n - p_n\rangle$ ist maximal. Definiere $S := K[x_1, \ldots, x_n] \setminus \mathfrak{m}_p \subsetneq K[x_1, \ldots, x_n] \subsetneq W_n$.

Bemerkung 3.16. Sei $f \in K[x_1, \ldots, x_n]$. Dann gilt

$$f \in S \quad \Leftrightarrow \quad f \notin \mathfrak{m}_p \quad \Leftrightarrow \quad f(p) \neq 0.$$

Lemma 3.17. *In W_n ist S eine Ore-Menge.*

Beweis: Klarerweise ist $1 \in S$, da $1 \notin \mathfrak{m}_p$. Sind $s, t \in S$ und wäre $s \cdot t \notin S$, so wäre $s \cdot t \in \mathfrak{m}_p$, also $s(p) \cdot t(p) = (s \cdot t)(p) = 0$. Dann impliziert Integritätsbereich aber schon den Widerspruch, nämlich dass $s \notin S$ oder $t \notin S$ gelten muss. Folglich ist S multiplikativ abgeschlossen.
Sei $f \in R$, etwa $f = \sum_{\beta \in B} b_\beta \partial^\beta$ mit $d := \mathrm{tdeg}_\partial(f)$ und $b_\beta \in K[x_1, \ldots, x_n]$, sowie $g \in S$ gegeben. Dann gilt nach Lemma 2.27

$$g^{d+1} \cdot f = r \cdot g$$

für ein $r \in R$. Aus $g \in S$ folgt $g^{d+1} \in S$ und somit die Aussage. □

Folgerung 3.18. *Seien $f \in W_n$ und $g \in S$ sowie $d := \mathrm{tdeg}_\partial(f)$. Dann existiert $y \in W_n$ mit*

$$g^{d+1} \cdot f = y \cdot g.$$

Da S eine Ore-Menge in W_n ist, kann man die sogenannte *geometrische Lokalisierung* $S^{-1}W_n$ bilden (anstelle von W_n kann man auch eine kommutative Algebra wählen, dieser Fall wird hier aber nicht weiter betrachtet).

Beispiel 3.19. Sei $R = K\langle x, s \mid s \cdot x = (x+1) \cdot s\rangle$ die erste Shift-Algebra, $\mathfrak{m} = \langle x \rangle$ und $S = K[x] \setminus \mathfrak{m}$. Angenommen, S wäre eine Ore-Menge in R. Dann gibt es zu $s \in R$ und $x - 1 \in S$ Elemente $f \in R$ sowie $g \in S$ mit $f \cdot (x-1) = g \cdot s$. Sei etwa $g = \sum_{i=0}^{d} c_i x^i$ mit $c_i \in K$; aus Gradvergleich ergibt sich $d = \deg(g) > 0$. Es gilt

$$f \cdot (x-1) = g \cdot s = \sum_{i=0}^{d} c_i x^i s = \sum_{i=0}^{d} c_i s (x-1)^i = s \sum_{i=0}^{d} c_i (x-1)^i,$$

dies kann aber nur dann der Fall sein, wenn $c_i = 0$ ist. Allerdings ist $g \in S$, was $c_i \neq 0$ bedeutet und somit zum Widerspruch führt. Folglich ist S keine Ore-Menge in R.

Bemerkung 3.20. Das voranstehende Beispiel deutet an, dass die Einschränkung auf Weyl-Algebren hier essentiell ist: $K[x] \setminus \mathfrak{m}_p$ ist in der ersten Shift-Algebra keine Ore-Menge. Auf ähnliche Weise kann man auch für die anderen nicht-kommutativen Modell-Algebren q-Shift und q-Weyl Gegenbeispiele konstruieren.

3.2.3. Monoidale Lokalisierung

In diesem Abschnitt sei R eine G-Algebra der Form

$$R = K\langle x_1, \ldots, x_n, y_1, \ldots, y_r \mid Q\rangle,$$

wobei die Relationsmenge Q die Bedingung $x_j x_i = x_i x_j$ für alle $i, j \in \underline{n}$ enthält. Für normierte Polynome $g_1, \ldots, g_k \in K[x_1, \ldots, x_n] \setminus K$ und $g := g_1 \cdots g_k$ definiert man

$$S := \{g_1^{n_1} \cdots g_k^{n_k} \mid n_i \in \mathbb{N}_0,\ 1 \leq i \leq k\} \subseteq R \quad \text{sowie} \quad \tilde{S} := \left\{g^i \mid i \in \mathbb{N}_0\right\}.$$

Lemma 3.21. *In W_n sind S und $\tilde{S}$ Ore-Mengen.*

Beweis: Folgt aus (wiederholter Anwendung von) Lemma 2.27. □

Als *monoidale Lokalisierung* bezeichnet man die Lokalisierung von R an S.

Satz 3.22. *Sind S und $\tilde{S}$ Ore-Mengen in R, so gilt $S^{-1}R \cong \tilde{S}^{-1}R$.*

Beweis: Sei $f \in S^{-1}R$, etwa $f = (s, r)$, wobei $s = g_1^{n_1} \cdots g_k^{n_k} \in S$, $r \in R$ und $n_i \in \mathbb{N}_0$. Sei $m := \max\{n_i\}$, $m_i := m - n_i$ und $t := g_1^{m_1} \cdots g_k^{m_k}$. Dann gilt $t \cdot s = g^m$. Weiterhin ist $(s, r) \sim (ts, tr) = (g^m, tr)$ und kann somit als Element von $\tilde{S}^{-1}R$ aufgefasst werden. □

Bemerkung 3.23. Insbesondere folgt aus Satz 3.22, dass man sich auf den Fall zurückziehen kann, in dem die g_i irreduzibel sind, da die Lokalisierung an der Menge der Potenzen eines Polynoms f isomorph ist zur Lokalisierung an der Menge der Produkte der Potenzen der irreduziblen Faktoren von f. Dann erhält man mit $g := g_1 \cdots g_k$ ein quadratfreies Polynom, was die Berechnung vereinfacht.

Die Menge $\tilde{S}$ kann in den kommutativen Hauptidealbereich $K[g]$ eingebettet werden. Für ein Linksideal I in R mit $I \cap \tilde{S} \neq \emptyset$ gilt

$$I \cap \tilde{S} \subsetneq I \cap K[g] = \langle h(g) \rangle$$

für ein $h(g) \in K[g] \setminus \{0\}$, wobei $h(g)$ als normiert angenommen werden kann.

Lemma 3.24. *Es gilt $h(g) = g^m$ für ein $m \in \mathbb{N}_0$.*

Beweis: Es ist $I \cap \tilde{S} \neq \emptyset$, also existiert ein $i \in \mathbb{N}_0$ so, dass $g^i \in I \cap \tilde{S}$. Dann ist auch $g^i \in \langle h(g) \rangle$, also existiert $f(g) \in K[g]$ mit $g^i = f(g) \cdot h(g)$. Damit ist aber schon $h(g) = g^m$ und $f(g) = g^j$ mit $j + m = i$.

□

4. Entgegengesetzte Strukturen

Speziell im Hinblick auf die Implementierung der Algorithmen in SINGULAR ist es notwendig, auf die Theorie der entgegengesetzten Strukturen einzugehen, die es ermöglichen, Rechts-Operationen als Links-Operationen aufzufassen.

4.1. Entgegengesetzte Ringe und Algebren

Definition 4.1. Sei $R = (R, +, \cdot)$ ein Ring. Der *entgegengesetzte Ring* zu R (engl. *opposite ring*) ist der Ring $R^{\text{opp}} := (R, +, *)$ mit der Multiplikation

$$* : R \times R \to R, \quad a * b := b \cdot a.$$

Für $r \in R$ sei r^{opp} das entgegengesetzte Element zu r. Entsprechend sei für $S \subseteq R$

$$S^{\text{opp}} := \{s^{\text{opp}} \mid s \in S\}.$$

Im Folgenden sei das Entgegensetzen von Elementen als Involution vorausgesetzt, sprich $(r^{\text{opp}})^{\text{opp}} = r$ für alle $r \in R$.

Bemerkung 4.2. Sei R ein Ring und $S \subseteq R$.

(1) Es gilt $(R^{\text{opp}})^{\text{opp}} = R$.

(2) Ist R kommutativ, so gilt $R = R^{\text{opp}}$.

(3) Es gilt $f \in S$ genau dann, wenn $f^{\text{opp}} \in S^{\text{opp}}$.

(4) Es gilt $0_{R^{\text{opp}}} = (0_R)^{\text{opp}}$, daher können die Nullelemente identifiziert werden..

(5) Ist $f \in S$, so gilt $f = 0$ genau dann, wenn $f^{\text{opp}} = 0$.

Lemma 4.3. *Sei I ein Linksideal (bzw. Rechtsideal) in R. Dann ist I^{opp} ein Rechtsideal (bzw. Linksideal) in R^{opp}.*

Beweis: Sei $f \in I^{\text{opp}}$ sowie $r \in R^{\text{opp}}$. Es gilt $(f * r)^{\text{opp}} = \underbrace{r^{\text{opp}}}_{\in R} \cdot \underbrace{f^{\text{opp}}}_{\in I} \in I$. Die andere Aussage folgt analog. □

Bemerkung 4.4. Sei K ein Körper und A eine K-Algebra. Dann ist auch A^{opp} eine K-Algebra mit der selben Vektorraumstruktur und wird *entgegengesetzte Algebra* zu A (engl. *opposite algebra*) genannt.

Bemerkung 4.5. Sei M ein R-Linksmodul. Dann ist $rm \in M$ für alle $r \in R$ und $m \in M$. In R^{opp} gilt $(rm)^{\text{opp}} = m^{\text{opp}} * r^{\text{opp}}$, somit wird M^{opp} auf natürliche Weise zu einem R^{opp}-Rechtsmodul.

Bemerkung 4.6 (Konstruktion nach [Lev05])**.** Bezeichnet man für $p \in A$ das entgegengesetzte Polynom mit $p^* \in A^{\text{opp}}$ und die Variablen in A^{opp} mit X_i, so ergibt sich folgende Möglichkeit der Konstruktion von A^{opp}: Da A und A^{opp} als Vektorräume isomorph sind, lässt sich eine spezielle

Bijektion wählen, die eine geeignete Bijektion auf den Monomen induziert. Hierzu bildet man x_i auf $X_{n+1-i} := x_i^*$ ab, der induzierte Monoid-Automorphismus ist

$$\sigma : \mathbb{N}_0^n \to \mathbb{N}_0^n, \quad \alpha = (\alpha_1, \ldots, \alpha_n) \mapsto (\alpha_n, \ldots, \alpha_1).$$

Da $\mathrm{Mon}(A) = \{x^\alpha \mid \alpha \in \mathbb{N}_0^n\}$ eine K-Basis von A ist, bietet es sich an, ein Monom in A^{opp} als

$$(x^\alpha)^* := X^{\sigma(\alpha)} = X_n^{\alpha_n} * \ldots * X_1^{\alpha_1}$$

Diese Wahl der Monome erfüllt die Nichtentartungsbedingungen (vgl. Definition 2.1). Auf A^{opp} existieren dann zu A entgegengesetzte Relationen:

$$X_i X_j = C_{j,i} X_j X_i + D_{j,i} \quad \text{für alle } 1 \leq i < j \leq n.$$

Definiert man $C_{j,i} := c_{n+1-i,n+1-j}$ sowie $D_{j,i} := d^*_{n+1-i,n+1-j}$, so ist ein Paar (X_i, X_j) mit der dazugehörigen Relation dem Paar (x_{n+1-i}, x_{n+1-j}) entgegengesetzt.
Ist $M \in K^{n\times n}$ invertierbar und die Darstellung einer zulässigen Wohlordnung $<_M$ auf A, so definiert man $M^* := \begin{pmatrix} M_n & \ldots & M_1 \end{pmatrix}$, wobei M_i die i-te Spalte von M sei. Nun ist $<_{M^*}$ genau dann eine Wohlordnung, wenn $<_M$ es ist. Die Ordnung $<_{M^*}$ erfüllt die Ordnungsbedingung, was A^{opp} nach Definition 2.1 zu einer G-Algebra macht.

4.2. Entgegengesetzte Lokalisierungen

Lemma 4.7. *Sei R ein Ring und $S \subseteq R$.*

(1) Ist S multiplikativ abgeschlossen in R, so ist S^{opp} multiplikativ abgeschlossen in R^{opp}.

(2) Ist S eine Links-Ore-Menge in R, so ist S^{opp} eine Rechts-Ore-Menge in R^{opp}.

(3) Ist S eine Linksnennermenge in R, so ist S^{opp} eine Rechtsnennermenge in R^{opp}.

Beweis: *(1)* Es gilt $(1_{R^{\mathrm{opp}}})^{\mathrm{opp}} = (1_R^{\mathrm{opp}})^{\mathrm{opp}} = 1_R \in S$, also $1_{R^{\mathrm{opp}}} \in S^{\mathrm{opp}}$. Seien $s^{\mathrm{opp}}, t^{\mathrm{opp}} \in S^{\mathrm{opp}}$. Dann ist

$$(s^{\mathrm{opp}} * t^{\mathrm{opp}})^{\mathrm{opp}} = \underbrace{t}_{\in S} \cdot \underbrace{s}_{\in S} \in S,$$

also $s^{\mathrm{opp}} * t^{\mathrm{opp}} \in S^{\mathrm{opp}}$.

(2) Seien $r^{\mathrm{opp}} \in R^{\mathrm{opp}}$ und $s^{\mathrm{opp}} \in S^{\mathrm{opp}}$. Für $r \in R$ und $s \in S$ existieren $\tilde{s} \in S$ und $\tilde{r} \in R$ mit

$$\tilde{r} \cdot s = \tilde{s} \cdot r \quad \Leftrightarrow \quad \tilde{r} \cdot s - \tilde{s} \cdot r = 0.$$

Setzt man $\hat{r} := \tilde{r}^{\mathrm{opp}} \in R^{\mathrm{opp}}$ sowie $\hat{s} := \tilde{s}^{\mathrm{opp}} \in S^{\mathrm{opp}}$, so folgt

$$(s^{\mathrm{opp}} * \hat{r} - r^{\mathrm{opp}} * \hat{s})^{\mathrm{opp}} = (s^{\mathrm{opp}} * \tilde{r}^{\mathrm{opp}} - r^{\mathrm{opp}} * \tilde{s}^{\mathrm{opp}})^{\mathrm{opp}} = \tilde{r} \cdot s - \tilde{s} \cdot r = 0.$$

(3) Seien $r^{\mathrm{opp}} \in R^{\mathrm{opp}}$ und $s^{\mathrm{opp}} \in S^{\mathrm{opp}}$ mit $s^{\mathrm{opp}} * r^{\mathrm{opp}} = 0$. Es gilt

$$(r \cdot s)^{\mathrm{opp}} = s^{\mathrm{opp}} * r^{\mathrm{opp}} = 0,$$

also $r \cdot s = 0$. Dann existiert $\tilde{s} \in S$ mit $\tilde{s} \cdot r = 0$. Setzt man $\hat{s} := \tilde{s}^{\mathrm{opp}} \in S^{\mathrm{opp}}$, so gilt

$$(r^{\mathrm{opp}} * \hat{s})^{\mathrm{opp}} = (r^{\mathrm{opp}} * \tilde{s}^{\mathrm{opp}})^{\mathrm{opp}} = \tilde{s} \cdot r = 0.$$

□

Satz 4.8. *Sei R eine G-Algebra und S eine Links-Ore-Menge in R. Dann gilt (als Ringe)*

$$(S^{-1}R)^{opp} \cong R^{opp}(S^{opp})^{-1}.$$

Beweis: Betrachte die Abbildung

$$\rho : (S^{-1}R)^{\mathrm{opp}} \to R^{\mathrm{opp}}(S^{\mathrm{opp}})^{-1}, \quad (s,r)^{\mathrm{opp}} \mapsto (r^{\mathrm{opp}}, s^{\mathrm{opp}}).$$

Bijektivität von ρ ist klar, die Umkehrabbildung ist

$$\rho^{-1} : R^{\mathrm{opp}}(S^{\mathrm{opp}})^{-1} \to (S^{-1}R)^{\mathrm{opp}}, \quad (r,s) \mapsto (s^{\mathrm{opp}}, r^{\mathrm{opp}})^{\mathrm{opp}}.$$

Seien $a := (s_1, r_1)^{\mathrm{opp}}, b := (s_2, r_2)^{\mathrm{opp}} \in (S^{-1}R)^{\mathrm{opp}}$.

- Es ist $\rho(0_{(S^{-1}R)^{\mathrm{opp}}}) = \rho((1_R, 0_R)^{\mathrm{opp}}) = (0_R^{\mathrm{opp}}, 1_R^{\mathrm{opp}}) = (0_{R^{\mathrm{opp}}}, 1_{R^{\mathrm{opp}}}) = 0_{R^{\mathrm{opp}}(S^{\mathrm{opp}})^{-1}}$.
- Es gilt
  $$a + b = (s_1, r_1)^{\mathrm{opp}} + (s_2, r_2)^{\mathrm{opp}} = ((s_1, r_1) + (s_2, r_2))^{\mathrm{opp}}.$$
  Einerseits ist
  $$\rho(a) + \rho(b) = \rho((s_1, r_1)^{\mathrm{opp}}) + \rho((s_2, r_2)^{\mathrm{opp}}) = (r_1^{\mathrm{opp}}, s_1^{\mathrm{opp}}) + (r_2^{\mathrm{opp}}, s_2^{\mathrm{opp}}),$$
  andererseits ist
  $$\rho(a + b) = \rho(((s_1, r_1) + (s_2, r_2))^{\mathrm{opp}}) = \rho((\tilde{s} \cdot s_1, \tilde{s} \cdot r_1 + \tilde{r} \cdot r_2)^{\mathrm{opp}}),$$
  wobei $\tilde{r} \in R$ und $\tilde{s} \in S$ die Bedingung $\tilde{s} \cdot s_1 = \tilde{r} \cdot s_2$ erfüllen. Dies impliziert $s_1^{\mathrm{opp}} * \tilde{s}^{\mathrm{opp}} = s_2^{\mathrm{opp}} * \tilde{r}^{\mathrm{opp}}$ und daher
  $$\begin{aligned} \rho(a + b) &= \rho((\tilde{s}s_1, \tilde{s}r_1 + \tilde{r}r_2)^{\mathrm{opp}}) \\ &= (r_1^{\mathrm{opp}} * \tilde{s}^{\mathrm{opp}} + r_2^{\mathrm{opp}} * \tilde{r}^{\mathrm{opp}}, s_1^{\mathrm{opp}} * \tilde{s}^{\mathrm{opp}}) \\ &= (r_1^{\mathrm{opp}}, s_1^{\mathrm{opp}}) + (r_2^{\mathrm{opp}}, s_2^{\mathrm{opp}}), \end{aligned}$$
  somit gilt
  $$\rho(a) + \rho(b) = (r_1^{\mathrm{opp}}, s_1^{\mathrm{opp}}) + (r_2^{\mathrm{opp}}, s_2^{\mathrm{opp}}) = \rho(a + b).$$
- Es gilt
  $$a * b = (s_1, r_1)^{\mathrm{opp}} * (s_2, r_2)^{\mathrm{opp}} = ((s_2, r_2) \cdot (s_1, r_1))^{\mathrm{opp}} = (\tilde{s} \cdot s_2, \tilde{r} \cdot r_1)^{\mathrm{opp}},$$
  wobei $\tilde{s} \in S$ und $\tilde{r} \in R$ die Bedingung $\tilde{s} \cdot r_2 = \tilde{r} \cdot s_1$ erfüllen. Dann ist
  $$\rho(a * b) = \rho((\tilde{s} \cdot s_2, \tilde{r} \cdot r_1)^{\mathrm{opp}}) = (r_1^{\mathrm{opp}} * \tilde{r}^{\mathrm{opp}}, s_2^{\mathrm{opp}} * \tilde{s}^{\mathrm{opp}}) = (r_1^{\mathrm{opp}}, s_1^{\mathrm{opp}}) * (r_2^{\mathrm{opp}}, s_2^{\mathrm{opp}}),$$
  da $r_2^{\mathrm{opp}} * \tilde{s}^{\mathrm{opp}} = s_1^{\mathrm{opp}} * \tilde{r}^{\mathrm{opp}}$. Andererseits ist
  $$\rho(a) * \rho(b) = \rho((s_1, r_1)^{\mathrm{opp}}) * \rho((s_2, r_2)^{\mathrm{opp}}) = (r_1^{\mathrm{opp}}, s_1^{\mathrm{opp}}) * (r_2^{\mathrm{opp}}, s_2^{\mathrm{opp}}).$$
  Dann gilt aber
  $$\rho(a) * \rho(b) = (r_1^{\mathrm{opp}}, s_1^{\mathrm{opp}}) * (r_2^{\mathrm{opp}}, s_2^{\mathrm{opp}}) = \rho(a * b).$$

□

Bemerkung 4.9. Insbesondere folgt aus Satz 4.8, dass $(S^{-1}R)^{\mathrm{opp}}$ eine Rechts-Ore-Lokalisierung ist und somit die Linksbrüche in $S^{-1}R$ direkt Rechtsbrüchen in $(S^{-1}R)^{\mathrm{opp}}$ entsprechen.

5. Grundlegende Algorithmen

5.1. Ore-Bedingung konstruktiv

Gegeben $(s, r) \in S \times R$ bietet es sich an, die Thematik der konstruktiven Ore-Bedingung in mehrere Probleme aufzuteilen:

(1) Finde ein $x \in S$, für das ein $y \in R$ existiert mit $xr = ys$.

(2) Finde die Menge aller $x \in S$, für die $y_x \in R$ existieren mit $xr = y_x s$.

Es bietet sich die Möglichkeit, zumindest eine der gesuchten Größen als Element des Kerns eines R-Linksmodul-Homomorphismus zu bestimmen. Dafür eignen sich grundsätzlich zwei Homomorphismen:

$$\varphi : R \xrightarrow{\cdot r} R/_R\langle -s\rangle, \quad x \mapsto xr - Rs \qquad \text{sowie} \qquad \psi : R \xrightarrow{\cdot s} R/_R\langle -r\rangle, \quad y \mapsto ys - Rr.$$

In beiden Fällen ergeben sich weiterführende Probleme:

φ: Es muss nicht nur $\mathrm{Ke}(\varphi)$, sondern auch $\mathrm{Ke}(\varphi) \cap S$ bestimmt werden (oder zumindest ein Element x daraus). Danach kann die zweite Unbekannte y ohne weitere Einschränkungen bestimmt werden, etwa mittels Division mit Rest.

ψ: Hier kommt grundsätzlich jedes $y \in \mathrm{Ke}(\psi)$ in Frage, allerdings ist nicht garantiert, dass zu jedem $y \in \mathrm{Ke}(\psi)$ auch ein passendes $x \in S$ existiert; die Ore-Bedingung sagt nur aus, dass es ein y mit passendem $x \in R$ gibt. Daher müsste jedes x auf Zugehörigkeit zu S überprüft werden, so dass die Suche nach einem Paar (x, y) im schlechtesten Fall ohne weitere Einschränkungen nicht unbedingt terminieren muss.

Aufgrund der schwerwiegenderen Schwierigkeiten im Falle ψ soll im Weiteren der Fall φ betrachtet werden:
Zentrales Objekt der Untersuchung ist $\mathrm{Ke}(\varphi) \cap S \neq \emptyset$. Bekannterweise ist $\mathrm{Ke}(\varphi)$ ein Linksideal in R, so dass sich folgende allgemeinere Frage stellt:
Gegeben ein Linksideal $\{0\} \neq I$ mit $I \cap S \neq \emptyset$, bestimme $I \cap S$ (oder zumindest ein Element daraus).
Das weitere Vorgehen hängt sehr stark von der Struktur der Menge S ab.

5.1.1. Rationale Lokalisierung

Sei R eine G-Algebra über einem Körper K in den Variablen $x_1, \ldots, x_n, y_1, \ldots, y_m$ zusammen mit $S := K[x_1, \ldots, x_n] \setminus \{0\} \subsetneq R$ derart, dass $S^{-1}R$ eine rationale Lokalisierung ist.
In dieser Situation kann man statt $I \cap S$ auch $I \cap K[x_1, \ldots, x_n]$ bestimmen, letzteres lässt sich etwa mittels Elimination berechnen und ist ein Linksideal in $K[x_1, \ldots, x_n]$. Gilt etwa $I \cap K[x_1, \ldots, x_n] = J$, so ist $I \cap S = J \setminus \{0\}$.

Algorithmus 5.1. Berechnung von $I \cap S$ in rationalen Lokalisierungen:
Input: Echtes Ideal $I \neq \{0\}$ in R.
Output: $I \cap S$.

```
1 begin
2 |  L := K[x_1, ..., x_n] ⊆ R;
3 |  Berechne J := I ∩ L mittels Elimination von y_1, ..., y_m;
4 |  return J \ {0};
5 end
```

Ist man nur an einem $f \in I \cap S$ interessiert, so kann man beispielsweise ein bezüglich Totalgrad minimales Element aus der Erzeugermenge von $I \cap L$ wählen, um Gradwachstum bei weiteren Rechnungen vorzubeugen.

5.1.2. Geometrische Lokalisierung

In der geometrischen Lokalisierung ist der Schnitt $I \cap S$ im Falle $I \cap S \neq \emptyset$ eine multiplikativ abgeschlossene Menge ohne 1.
Im folgenden Algorithmus wird entweder die leere Menge oder eine Menge J von bestimmten Elementen zurückgegeben, die den Schnitt beschreiben. Dabei wird die Struktur von J nur unzureichend wiedergegeben, allerdings ist bisher keine bessere Darstellung bekannt.

Algorithmus 5.2. Berechnung von $I \cap S$ in geometrischen Lokalisierungen:
Input: Echtes Linksideal $I \neq \{0\}$ in R, $p \in K^n$, $S = K[x_1, \ldots, x_n] \setminus \mathfrak{m}_p$.
Output: $J = I \cap S$.

```
1  begin
2  |  Setze S̃ := K[x_1, ..., x_n];
3  |  Berechne Ĩ := I ∩ S̃ = ⟨m_1, ..., m_k⟩ mittels Variablen-Elimination;
4  |  Berechne L := Ĩ ∩ m_p;
5  |  for i from 1 to k do
6  |  |  m̃_i := NF(m_i|L);
7  |  end
8  |  J := ⟨m̃_i | m̃_i ≠ 0⟩ \ {0};   // nur f ∈ J mit NF(f|L) ≠ 0 kommen infrage
9  |  return J;
10 end
```

Satz 5.3. *Algorithmus 5.2 terminiert und ist korrekt.*

Beweis: Ist $I \cap S = \emptyset$, so folgt

$$\tilde{I} = I \cap \tilde{S} = I \cap (S \cup \mathfrak{m}_p) = (I \cap S) \cup (I \cap \mathfrak{m}_p) = I \cap \mathfrak{m}_p \subseteq \mathfrak{m}_p,$$

also $L = \tilde{I} \cap \mathfrak{m}_p = I \cap \mathfrak{m}_p \cap \mathfrak{m}_p = \tilde{I}$. Dann ist aber $\mathrm{NF}(m_i|L) = 0$ für alle $i \in \underline{k}$ und somit $J = \emptyset$. Ist andererseits $I \cap S \neq \emptyset$, so ist auch $\tilde{I} \neq \{0\}$. Wäre $\mathrm{NF}(m_i|L) = 0$ für alle i, so wäre erneut $\tilde{I} \subseteq \mathfrak{m}_p$ und damit

$$I \cap S \subsetneq I \cap \tilde{S} = \tilde{I} \subseteq \mathfrak{m}_p.$$

Aber für alle $f \in S$ gilt $f \notin \mathfrak{m}_p$, was zusammen $I \cap S = \emptyset$ als Widerspruch implizieren würde. Folglich gibt es ein i, so dass $\tilde{m}_i = \mathrm{NF}(m_i|J) \neq 0$, also $\tilde{m}_i \in S$ und damit $J \neq \emptyset$. □

5.1.3. Monoidale Lokalisierung

Im Falle der monoidalen Lokalisierung kann man sich nach Bemerkung 3.23 auf den Fall beschränken, in dem S nur von einem Polynom erzeugt wird.

Algorithmus 5.4. Berechnung von $I \cap \tilde{S}$ in monoidalen Lokalisierungen:

Input: Gröbnerbasis $F = \{f_1, \ldots, f_k\}$ des echten Ideals $I \neq \{0\}$ in R,
$g \in K[x_1, \ldots, x_n] \setminus K$.
Annahme: Es existiert ein $m \in \mathbb{N}$ mit $g^m \in I$.
Output: g^m.

```
1  begin
2      m_0 := ∞;
3      for j from 1 to k do
4          m_min := min{k ∈ ℕ : lm(f_j) | lm(g)^k};
5          m_0 := min{m_0, m_min};
6      end
7      m := m_0;
8      while NF(g^m|F) ≠ 0 do
9          m := m + 1;
10     end
11     return g^m;
12 end
```

Satz 5.5. *Algorithmus 5.4 terminiert und ist korrekt.*

Beweis: Nach Voraussetzung gibt es ein $m \in \mathbb{N}$, so dass $g^m \in I$. Dann existiert ein j mit $\mathrm{lm}(f_j) | \mathrm{lm}(g^m)$ und somit ergibt sich im Algorithmus nach der for-Schleife ein $m_0 < \infty$. Danach wird in der repeat-Schleife das tatsächliche Minimum iterativ ermittelt: Sobald sich g^m unter F auf 0 reduzieren lässt, ist m der gesuchte Wert und minimal per Konstruktion.
Die Berechnung der Minima in der for-Schleife kann durch Betrachten der Exponenten linear in n durchgeführt werden. Da die Voraussetzung die Existenz eines m garantiert, terminiert auch die while-Schleife und somit der ganze Algorithmus. □

Bemerkung 5.6. Im Fall $I = \mathrm{Ke}(\varphi)$, wobei φ für gegebene $(s, r) \in S \times R$ durch

$$\varphi : R \xrightarrow{\cdot r} R/_R\langle -s \rangle, \quad x \mapsto xr - Rs,$$

definiert war (vgl. Anfang von Abschnitt 5.1), ist nach Lemma 2.27 eine obere Schranke für m durch $d + c$ gegeben, wobei $d := \mathrm{tdeg}_y(r)$ und c die Identität $s = g^c$ erfüllt.

Bemerkung 5.7. Die Berechnung der Normalformen kann so gestaltet werden, dass vorherige Ergebnisse wiederverwendet werden können, da

$$\mathrm{NF}(g^{m+1}|F) = \mathrm{NF}(g \cdot \mathrm{NF}(g^m|F)|F)$$

gilt (siehe etwa Corollary 3.15 in [And10]).

5.2. Kürzen

Gegeben sei ein Linksbruch $(t, p) \in S^{-1}R$. Gesucht ist nun ein (in geeignetem Sinne) größtmögliches $u \in S$ mit $t = u \cdot s$ und $p = u \cdot r$.

5.2.1. Rationale Lokalisierung

Sei R eine G-Algebra über einem Körper K in den Variablen $x_1, \ldots, x_n, y_1, \ldots, y_m$ zusammen mit $S := K[x_1, \ldots, x_n] \setminus \{0\} \subsetneq R$ derart, dass $S^{-1}R$ eine rationale Lokalisierung ist.
Sei $p \in R$, dann gibt es eine Darstellung $p = \sum_{\beta \in \mathbb{N}_0^m} c_\beta y^\beta$ mit $c_\beta \in K[x_1, \ldots, x_n]$, wobei $c_\beta = 0$ für fast alle $\beta \in \mathbb{N}_0^m$ gilt.

Erinnerung (vgl. Definition 2.10)**.** Für $p = \sum_{\beta \in \mathbb{N}_0^m} c_\beta y^\beta \in R$ ist $\operatorname{supp}_y(p) := \{\beta \in \mathbb{N}_0^m \mid c_\beta \neq 0\}$.

Bemerkung 5.8. Es gilt $p = u'(x) \cdot r'(x, y)$ genau dann, wenn $u'(x) | c_\beta$ für alle $\beta \in \operatorname{supp}_y(p)$.

Satz 5.9. *Sei $p = \sum_{\beta \in \mathbb{N}_0^m} c_\beta y^\beta \in R \setminus \{0\}$ und $t \in S$. Definiere*

$$u' := \operatorname{ggT}(\{c_\beta \in \operatorname{supp}_y(p)\}) \in S \quad \textit{sowie} \quad u := \operatorname{ggT}(u', t) \in S.$$

Dann gilt $t = u \cdot s$ sowie $p = u \cdot r$ für gewisse $s \in S$ und $r \in R$. Weiterhin ist u maximal mit dieser Eigenschaft.

Beweis: Da $u | t$ per Konstruktion wähle $s := u^{-1}t \in S$, dann gilt $t = u \cdot u^{-1}t = u \cdot s$. Definiere

$$r := \sum_{\beta \in \operatorname{supp}_y(p)} u^{-1} c_\beta y^\beta \in R \setminus \{0\},$$

es gilt

$$u \cdot r = u \cdot \sum_{\beta \in \operatorname{supp}_y(p)} u^{-1} c_\beta y^\beta = \sum_{\beta \in \operatorname{supp}_y(p)} u \cdot u^{-1} c_\beta y^\beta = \sum_{\beta \in \operatorname{supp}_y(p)} c_\beta y^\beta = p.$$

Angenommen, es existiert ein weiteres $w \in S$, so dass $t = w \cdot \tilde{s}$ und $p = w \cdot \tilde{r}$ für gewisse $\tilde{s} \in S$ sowie $\tilde{r} \in R$. Dann gilt $w | c_\beta$ für alle $\beta \in \operatorname{supp}_y(p)$, also insbesondere $w | u'$. Außerdem gilt $w | t$ per Voraussetzung, also $w | u = \operatorname{ggT}(u', t)$ (bis auf Assoziiertheit). □

Algorithmus 5.10. Kürzen in rationalen Lokalisierungen:

Input: $(t, p) \in S^{-1}R$.
Annahme: $p \neq 0$.
Output: $(s, r) \in S^{-1}R$, $u \in S$ mit $p = u \cdot r$ und $t = u \cdot s$, u maximal.

begin
  Schreibe $p = \sum_{\beta \in \operatorname{supp}_y(p)} c_\beta y^\beta$;
  $u' := \operatorname{ggT}(\{c_\beta \in \operatorname{supp}_y(p)\})$;
  $u := \operatorname{ggT}(u', t)$;
  $s := u^{-1}t$;
  $r := \sum_{\beta \in \operatorname{supp}_y(p)} u^{-1} c_\beta y^\beta$;
  return $(s, r), u$;
end

Satz 5.11. *Algorithmus 5.10 terminiert und ist korrekt.*

Beweis: Terminierung ist klar, da $\operatorname{supp}_y(p)$ endlich ist. Korrektheit folgt aus Satz 5.9. □

5.2.2. Geometrische Lokalisierung

Sei $R = W_n$ für ein $n \in \mathbb{N}$, $q \in K^n$ und $S := K[x_1, \ldots, x_n] \setminus \mathfrak{m}_q$. Die Suche nach einem größten $u \in S$ lässt sich ähnlich gestalten wie im Falle der rationalen Lokalisierung. Die einzige Zusatzbedingung ist, dass u nicht im Punkt p verschwinden darf.

Satz 5.12. *Algorithmus 5.10 liefert auch in geometrischen Lokalisierungen das korrekte Ergebnis.*

Beweis: Der Algorithmus liefert das größtmögliche $u \in K[x_1, \ldots, x_n] \setminus \{0\}$, so dass $p = u \cdot r$ und $t = u \cdot s$. Da u per Konstruktion ein Teiler von t ist und $t \notin \mathfrak{m}_q$, folgt dass auch $u \notin \mathfrak{m}_q$ gelten muss. Somit ist $u \in S$. □

5.2.3. Monoidale Lokalisierung

Sei R eine für monoidale Lokalisierung geeignete G-Algebra, $g \in K[x_1, \ldots, x_n] \setminus K$ normiert, $S := \{g^i \mid i \in \mathbb{N}_0\}$ und betrachte die Lokalisierung $S^{-1}R$.

Lemma 5.13. *Seien $(g^{i_1}, r_1), (g^{i_2}, r_2) \in S^{-1}R$ und oBdA $i_1 \leq i_2$. Dann gilt $(g^{i_1}, r_1) = (g^{i_2}, r_2)$ genau dann, wenn $r_2 = g^{i_2 - i_1} r_1$.*

Beweis: Sei zunächst $r_2 = g^{i_2-i_1} r_1$. Definiere $\tilde{s} := 1 \in S$ und $\tilde{r} := g^{i_2-i_1}$. Es gilt

$$\tilde{s} g^{i_2} = g^{i_2} = g^{i_2-i_1} g^{i_1} = \tilde{r} g^{i_1} \quad \text{sowie} \quad \tilde{s} r_2 = r_2 = g^{i_2-i_1} r_1 = \tilde{r} r_1,$$

also $(g^{i_1}, r_1) = (g^{i_2}, r_2)$. Sei andererseits $(g^{i_1}, r_1) = (g^{i_2}, r_2)$. Dann existieren $\tilde{s} \in S$ und $\tilde{r} \in R$ mit $\tilde{s} g^{i_2} = \tilde{r} g^{i_1}$ und $\tilde{s} r_2 = \tilde{r} r_1$. Da $\tilde{s} \in S$ gilt $\tilde{s} = g^k$ für ein $k \in \mathbb{N}_0$. Somit ist

$$0 = \tilde{s} g^{i_2} - \tilde{r} g^{i_1} = g^k g^{i_2} - \tilde{r} g^{i_1} = g^{k+i_2} - \tilde{r} g^{i_1} = (g^{k+i_2-i_1} - \tilde{r}) g^{i_1},$$

also $\tilde{r} = g^{k+i_2-i_1}$. Es folgt

$$0 = \tilde{s} r_2 - \tilde{r} r_1 = g^k r_2 - g^{k+i_2-i_1} r_1 = g^k (r_2 - g^{i_2-i_1} r_1),$$

daher ist $r_2 = g^{i_2-i_1} r_1$. □

Bemerkung 5.14. Sei $f = (g^k, r) \in S^{-1}R$. Für $k = 0$ ist f nicht kürzbar. Für $k \geq 1$ ist f genau dann kürzbar, wenn $r = g^i a$ für gewisse $i \in \mathbb{N}$ und $a \in R$ gilt.

Für $i \in \mathbb{N}$ definiere $b_i := \mathrm{RNF}(r|g^i)$ (RNF steht hier für Rechts-Normalform) und $a_i \in R$ derart, dass $r = g^i a_i + b_i$ gilt. Die Normalform und Division mit Rest stellen dabei sicher, dass entweder $b_i = 0$ oder $\mathrm{lm}(b_i) < \mathrm{lm}(g^i)$ gilt.

Lemma 5.15. *Sei $k \in \mathbb{N}$.*

(1) Ist $b_k = 0$, so gilt $b_i = 0$ für alle $i \leq k$.

(2) Ist $b_k \neq 0$, so gilt $b_i \neq 0$ für alle $i \geq k$.

Beweis: *(1)* Sei $b_k = 0$ und $1 \leq i < k$. Dann gilt $r = g^k a_k = g^i g^{k-i} a_k \in g^i R$, also

$$b_i = \mathrm{RNF}(r|g^i) = \mathrm{RNF}(g^k a_k | g^i) = \mathrm{RNF}(g^i g^{k-i} a_k | g^i) = 0.$$

(2) Folgt direkt aus *(1)*.

□

Folglich gibt es ein eindeutig bestimmtes $m \in \mathbb{N}$ mit $b_i = 0$ für alle $i \leq m$ sowie $b_i \neq 0$ für alle $i > m$.

Lemma 5.16. *Sei $(g^k, r) \in S^{-1}R$ und $m \in \mathbb{N}$ derart, dass $b_i = 0$ für alle $i \leq m$ und $b_i \neq 0$ für alle $i > m$. Sei $j := \min\{m, k\}$. Dann gilt*

$$(g^k, r) = (g^{k-j}, a_j).$$

Beweis: Sei $m \geq k$, also $j = k$. Es ist $b_k = 0$ und somit $r = g^k a_k$, was

$$(g^k, r) = (g^k, g^k a_k) = (1, a_k) = (g^{k-j}, a_j)$$

impliziert. Ist andererseits $j = m < k$, so ist $r = g^m a_m$ und

$$(g^k, r) = (g^k, g^m a_m) = (g^{k-m}, a_m) = (g^{k-j}, a_j).$$

Nach Wahl von m lässt sich (g^{k-m}, a_m) nicht weiter kürzen. □

Im folgenden Algorithmus bezeichne $<_m$ die absteigende POT-Monom-Modulordnung auf R^2 $(e_1 > e_2)$.

Algorithmus 5.17. Kürzen in monoidalen Lokalisierungen:

Input: $(g^k, r) \in S^{-1}R$.
Output: $(g^i, \tilde{r}) \in S^{-1}R$ mit $(g^k, r) = (g^i, \tilde{r})$, wobei $i = 0$ oder $\tilde{r} \notin gR$.

```
1  begin
2      r̃ := r;
3      for i from 1 to k do
4          [b_i; a_i] := RNF_{<_m}( [r̃; 0] | [g; -1] );
5          if b_i ≠ 0 then
6              return (g^{k-i}, r̃);
7          else
8              r̃ := a_i;
9          end
10     end
11     return (1, r̃);
12 end
```

Satz 5.18. *Algorithmus 5.17 terminiert und ist korrekt.*

Beweis: Terminierung ist klar, da die for-Schleife nur endlich oft ausgeführt wird. Die Wahl der Ordnung garantiert, dass in jedem Schritt $b_i = \mathrm{RNF}(\tilde{r}|g)$ gilt, als Nebenprodukt ergibt sich in der zweiten Komponente a_i mit $\tilde{r} = ga_i + b_i$. Ist $b_i \neq 0$, so ist $\tilde{r} \notin gR$ und $(k-i)$-mal konnte der Linksfaktor g herausgekürzt werden. Ist $b_i = 0$, so ersetzt man $\tilde{r}$ durch a_i und führt den nächsten Schritt aus. □

5.3. Reduktion

5.3.1. Rationale Lokalisierung

Sei R eine G-Algebra über einem Körper K in den Variablen $x_1, \ldots, x_n, y_1, \ldots, y_m$ zusammen mit $S := K[x_1, \ldots, x_n] \setminus \{0\}$ derart, dass $S^{-1}R$ eine rationale Lokalisierung ist.
Sei $r \in R \setminus \{0\}$, etwa $r = \sum_{\beta \in \mathrm{supp}_y(r)} c_\beta y^\beta$ mit $c_\beta \in K[x_1, \ldots, x_n]$, sowie $s \in S$. Für $(s, r) \in S^{-1}R$ ist dann

$$(s, r) = s^{-1}r = s^{-1} \sum_{\beta \in \mathrm{supp}_y(r)} c_\beta y^\beta = \sum_{\beta \in \mathrm{supp}_y(r)} (s^{-1}c_\beta) y^\beta \in K(x)\langle y \rangle.$$

Dies ermöglicht es, auf $S^{-1}R \cong K(x)\langle y \rangle$ Begriffe wie Leitmonom zu definieren:

Definition 5.19. Sei $(s, r) \in S^{-1}R \setminus \{(1, 0)\}$ mit

$$r = \sum_{\beta \in \mathrm{supp}_y(r)} c_\beta y^\beta \quad \text{und} \quad (s, r) = \sum_{\beta \in \mathrm{supp}_y(r)} (s^{-1}c_\beta) y^\beta.$$

Sei $\hat{\beta} := \max\left\{\beta \in \mathrm{supp}_y(r)\right\} \in \mathbb{N}_0^m$ (bezüglich einer Monomordnung auf $K[y]$). Dann definiert man:

(1) Die *Monome* von $S^{-1}R$: $\mathrm{Mon}(S^{-1}R) := \left\{y^\beta \mid \beta \in \mathbb{N}_0^m\right\} \subsetneq K(x)\langle y \rangle$.

(2) Das *Leitmonom* von (s, r): $\mathrm{lm}(s, r) := y^{\hat{\beta}} \in \mathrm{Mon}(S^{-1}R)$.

(3) Den *Leitkoeffizienten* von (s, r): $\mathrm{lc}(s, r) := c_{\hat{\beta}} \in K(x)$.

(4) Den *Leitexponenten* von (s, r): $\mathrm{le}(s, r) := \hat{\beta} \in \mathbb{N}_0^m$.

(5) Den *Leitterm* von (s, r): $\mathrm{lt}(s, r) := \mathrm{lc}(s, r) \cdot \mathrm{lm}(s, r) = c_{\hat{\beta}} y^{\hat{\beta}} \in K(x)\langle y \rangle$.

Eine wichtige Rolle in der Gröbner-Theorie nimmt die Reduktion ein: Gegeben Elemente f und g, bestimme $(1, 0) \neq m \in S^{-1}R$ mit $\mathrm{lm}(f) = \mathrm{lm}(\mathrm{lm}(m) \cdot \mathrm{lm}(g))$ (falls es existiert) sowie $0 \neq c \in S^{-1}(S \cup \{0\}) \cong K(x)$ so, dass $\mathrm{lt}(f) = c \cdot \mathrm{lt}(m \cdot g)$, und berechne $f - c \cdot m \cdot g$.

Algorithmus 5.20. Reduktion in rationalen Lokalisierungen: `reduce`

Input: $f, g \in S^{-1}R$.
Annahme: $\mathrm{lm}(g) | \mathrm{lm}(f)$.
Output: $f - c \cdot m \cdot g$ mit $\mathrm{lt}(f) = c \cdot \mathrm{lt}(m \cdot g)$, wobei $c \in S^{-1}(S \cup \{0\})$, und $\mathrm{lm}(f) = \mathrm{lm}(\mathrm{lm}(m) \cdot \mathrm{lm}(g))$ mit $m \in S^{-1}R$.

```
1  begin
2  |  α := le(f) − le(g);
3  |  m := (1, y^α);
4  |  p := mult(m, g);   // p := m · g ∈ S^{-1}R
5  |  (b_f, a_f) := lc(f);
6  |  (b_p, a_p) := lc(p);
7  |  a := a_f · b_p;
8  |  b := a_p · b_f;
9  |  c := (b, a);
10 |  r := add(f, mult(−c, p));   // r := f − c · p = f − c · m · g ∈ S^{-1}R
11 |  return r;
12 end
```

6. Arithmetische Algorithmen und Implementierung

In diesem Kapitel werden die Algorithmen beschrieben, mit denen ein konstruktives Rechnen in Ore-lokalisierten G-Algebren möglich ist, sowie deren Korrektheit bewiesen. Gleichzeitig werden die Funktionen beschrieben, die den jeweiligen Algorithmen in der Implementierung `olga.lib` im Computeralgebra-System SINGULAR entsprechen, sowie deren Funktionsweise anhand von Beispielen erklärt, so dass dieses Kapitel gleichzeitig als eine Art "User Manual" dient. Um eine bessere Kompatibilität mit dem (englischsprachigen) Singular Manual ([SIN]) zu gewährleisten, ist der Rest dieses Kapitels in Englisch verfasst.

6.1. Data structure

In the implementation, localization data is stored in different ways:

- For a monoidal localization $S = \{g_1^{n_1} \cdots g_k^{n_k} \mid n_i \in \mathbb{N}_0\}$, the set of polynomials $g_1, \ldots, g_k$ is stored as data of type `list`.
- For a geometrical localization $S = K[x_1, \ldots, x_n] \setminus \mathfrak{m}_p$, the point p is stored as data of type `vector`.
- For a rational localization $S = K[x_1, \ldots, x_m] \setminus \{0\}$ of an algebra in $n \geq m$ variables, the set of indices $\underline{m}$ is stored as data of type `intvec`.

To provide procedures that work on all three types of localization each type is associated with an `int` number: 0 corresponds to a monoidal localization, 1 to a geometrical localization, and 2 to a rational localization.

A fraction f is stored as a `vector` with four entries $[s, r, p, t]$, where $(s, r) = s^{-1}r$ is the left fraction representation of f, respectively $(p, t) = pt^{-1}$ is the right fraction representation of f. For f being a valid fraction only one of the representations has to be declared, but if both are declared they have to be equal, i.e., $rt = sp$ has to hold.

Example. Defining fractions and localizations in the second Weyl-algebra:

```
ring R = 0,(x,y,dx,dy),dp;
def S = Weyl();
setring S;
poly g1 = x^2+4*x*y+3*y^2+x-4;
poly g2 = (y-5)^3;
list L = g1,g2;
vector p = [1,-3];
intvec v = 1,2;
```

Now $(0, L)$, $(1, p)$, and $(2, v)$ can be used as localization data.

```
vector a = [ g1^2*g2^3, 3*x*dx+dy-3*x+4, 0, 0 ];
vector b = [ 0, 0, dx, x^2*y+7 ];
vector c = [ x, dx+dy, x*dx+x*dy+2, x^2 ];
```

The `vector` a is a valid left fraction in $(0, L)$, b is a valid right fraction in $(1, p)$, and c is a valid two-sided fraction in $(2, v)$.

6.2. Constructive Ore data

Algorithm 6.1. Computing Ore data: `Ore`

Input: $s \in S$, $r \in R$, $b \in \{0, 1\}$.

Output: $\tilde{s} \in S, \tilde{r} \in R$, $J \subseteq S$:

- If $b = 0$: $\tilde{s}r = \tilde{r}s$ and for all $f \in J$ exists $g \in R$ such that $fr = gs$.
- If $b = 1$: $r\tilde{s} = s\tilde{r}$ and for all $f \in J$ exists $g \in R$ such that $rf = sg$.

```
1  begin
2  |  if b = 1 then
3  |  |  R' := R^opp; s' := s^opp; r' := r^opp; S' := S^opp;
4  |  else if b = 0 then
5  |  |  R' := R; s' := s; r' := r; S' := S;
6  |  end
7  |  Define the left module homomorphism φ : R' → R'/_R⟨−s'⟩, x ↦ xr' − R's';
8  |  Compute the left ideal I := Ke(φ);
9  |  Compute J := I ∩ S via the algorithms described in Chapter 5;
10 |  Choose s̃ ∈ J;
11 |  Compute r̃ via left division with remainder of s̃r divided by s;
12 |  if b = 1 then
13 |  |  s̃' := s̃^opp; r̃' := r̃^opp; J' := J^opp;
14 |  else if b = 0 then
15 |  |  s̃' := s̃; r̃' := r̃; J' := J;
16 |  end
17 |  return (s̃', r̃', J');
18 end
```

Lemma 6.2. *Algorithm 6.1 terminates and is correct.*

Proof: Termination is obvious. By the left Ore condition, $J \neq \emptyset$. The left division yields some $\tilde{r} \in R$ for every $\tilde{s} \in J$ with $\tilde{s}r = \tilde{r}s$, which implies correctness in the case $b = 0$. If $b = 1$, one computes left Ore data in the opposite ring, which, after changing back, yields right Ore data for the input problem. □

Procedure 6.3. ore(g, f, locType, locData, rightOre)

Usage:	poly g, f, int locType, list/vector/intvec locData, int rightOre
Purpose:	compute Ore data for a given one-sided fraction
Assume:	g is in the denominator set, determined via locType and locData
Return:	list
Note:	• returns a list, consisting of vector [s,r], ideal J • if rightOre=0, $sf = gr$ holds, • if rightOre=1, $fs = rg$ holds, • J is a representation of the possible choices for s.

Example. Computing Ore data in various localizations of the second Weyl algebra:

```
ring R = 0,(x,y,dx,dy),dp;
def S = Weyl();
setring S;
poly g1 = x+3; poly g2 = x*y;
list L = g1,g2;
poly g = g1^2*g2; poly f = dx;
list rm = ore(g, f, 0, L, 0); // monoidal localization (left Ore)
print(rm[1]);
-> [x^8*y^4+12*x^7*y^4+54*x^6*y^4+108*x^5*y^4+81*x^4*y^4,
-> x^5*y^3*dx+6*x^4*y^3*dx-3*x^4*y^3+9*x^3*y^3*dx-12*x^3*y^3-9*x^2*y^3]
rm[1][2]*g-rm[1][1]*f;
-> 0
intvec rat = 1;
f = dx+dy; g = x;
list rr = ore(g, f, 2, rat, 0); // rational localization (left Ore)
print(rr[1]);
-> [x^2,x*dx+x*dy-1]
rr[2]; // the ideal J
-> _[1]=x^2
rr[1][2]*g-rr[1][1]*f;
-> 0
vector p = [1,3];
g = x+y; f = dx+dy;
list rg = ore(g, f, 1, p, 1); // geometrical localization (right Ore)
print(rg[1]);
-> [x^2+2*x*y+y^2,x*dx+y*dx+x*dy+y*dy+4]
rg[2];  // the ideal J
-> _[1]=x^2+2*x*y+y^2
f*rg[1][1]-g*rg[1][2];
-> 0
```

6.3. Converting between left and right representations

Algorithm 6.4. Converting a right fraction into a left fraction: `RToL`

Input: $s \in S$, $r \in R$.
Output: $(\tilde{s}, \tilde{r}) \in S^{-1}R$ with $\tilde{s}r = \tilde{r}s$.

1 **begin**
2 $\quad (\tilde{s}, \tilde{r}, J) := \texttt{Ore}(s, r, 0)$;
3 $\quad$ **return** $(\tilde{s}, \tilde{r})$;
4 **end**

Lemma 6.5. *Algorithm 6.4 terminates and is correct.*

Proof: `Ore` yields $\tilde{s} \in S$ and $\tilde{r} \in R$ satisfying $\tilde{s}r = \tilde{r}s$, which is equivalent to $\tilde{s}^{-1}\tilde{r} = rs^{-1}$. □

Procedure 6.6. convertRightToLeftFraction(frac, locType, locData)

Usage: vector frac, int locType, list/vector/intvec locData
Purpose: determine a left fraction representation of a given fraction
Assume:
Return: vector
Note:
- the returned vector contains a representation of frac as a left fraction,
- if the left representation of frac is already present, frac will be returned.

Example. Converting a right to a left fraction in a rational localization of the second shift algebra:

```
ring S = 0,(x,y,Sx,Sy),dp;
matrix D[4][4];
D[1,3] = Sx;
D[2,4] = Sy;
def ncS = nc_algebra(1, D);
setring ncS;
intvec rat = 1;
poly f = Sx+Sy;
poly g = x;
vector fracr = [0,0,f,g];
vector rr = convertRightToLeftFraction(fracr,2,rat);
print(rr);
-> [x^2+x,x*Sx+x*Sy+Sy,Sx+Sy,x]
rr[2]*g-rr[1]*f;
-> 0
```

Algorithm 6.7. Converting a left fraction into a right fraction: `LToR`

Input: $(s, r) \in S^{-1}R$.
Output: $\tilde{s} \in S$, $\tilde{r} \in R$ with $r\tilde{s} = s\tilde{r}$.

begin
  $(\tilde{s}, \tilde{r}, J) :=$ `Ore`$(s, r, 1)$;
  return $(\tilde{s}, \tilde{r})$;
end

Lemma 6.8. *Algorithm 6.7 terminates and is correct.*

Proof: `Ore` yields $\tilde{s} \in S$ and $\tilde{r} \in R$ satisfying $r\tilde{s} = s\tilde{r}$, which is equivalent to $\tilde{r}\tilde{s}^{-1} = s^{-1}r$. □

Procedure 6.9. convertLeftToRightFraction(frac, locType, locData)

Usage: vector frac, int locType, list/vector/intvec locData
Purpose: determine a right fraction representation of a given fraction
Assume:
Return: vector
Note:
- the returned vector contains a representation of frac as a right fraction,
- if the right representation of frac is already specified, frac will be returned.

Example. Converting a left to a right fraction in a rational localization of the second q-shift algebra:

```
ring Q = (0,q),(x,y,Qx,Qy),dp;
matrix C[4][4] = UpOneMatrix(4);
C[1,3] = q;
C[2,4] = q;
def ncQ = nc_algebra(C, 0);
setring ncQ;
poly f = Qx+Qy;
poly g = x^2+1;
vector fracr = [g,f,0,0];
vector rr = convertLeftToRightFraction(fracr,2,rat);
print(rr);
-> [x^2+1,Qx+Qy,(q^4)*x^2*Qx+x^2*Qy+(q^2)*Qx+(q^2)*Qy,x^4+(q^2+1)*x^2+(q^2)]
f*rr[4]-g*rr[3];
-> 0
```

6.4. Addition of left fractions

Algorithm 6.10. Adding two left fractions in $S^{-1}R$: `add`

Input: $(s_1, r_1), (s_2, r_2) \in S^{-1}R$.
Output: $(s, r) \in S^{-1}R$ with $(s, r) = (s_1, r_1) + (s_2, r_2)$.

```
1 begin
2 |  (s̃, r̃, J) := Ore(s_2, s_1, 0);
3 |  (s, r) := (s̃s_1, s̃r_1 + r̃r_2);
4 |  return (s, r);
5 end
```

Lemma 6.11. *Algorithm 6.10 terminates and is correct.*

Proof: `Ore` yields $\tilde{s} \in S$, $\tilde{r} \in R$ satisfying $\tilde{s}s_1 = \tilde{r}s_2$. It holds

$$(s_1, r_1) + (s_2, r_2) = (\tilde{s}s_1, \tilde{s}r_1) + (\tilde{r}s_2, \tilde{r}r_2) = (\tilde{s}s_1, \tilde{s}r_1) + (\tilde{s}s_1, \tilde{r}r_2) = (\tilde{s}s_1, \tilde{s}r_1 + \tilde{r}r_2).$$

□

Procedure 6.12. addLeftFractions(a, b, locType, locData)

Usage:	vector a, b, int locType, list/vector/intvec locData
Purpose:	add two left fractions in the specified localization
Assume:	
Return:	vector
Note:	the returned vector is the sum of a and b as fractions in the localization specified by locType and locData.

Example. Adding left fractions in various localization of the second Weyl algebra:

```
ring R = 0,(x,y,dx,dy),dp;
def S = Weyl();
setring S;
poly g1 = x+3;
poly g2 = x*y+y;
list L = g1,g2;
vector frac1 = [g1,dx,0,0];
vector frac2 = [g2,dy,0,0];
vector rm = addLeftFractions(frac1,frac2,0,L); // monoidal localization
print(rm);
-> [x^3*y+7*x^2*y+15*x*y+9*y,x^2*y*dx+4*x*y*dx+x^2*dy+3*y*dx+6*x*dy+9*dy]
vector p = [1,3];
vector rg = addLeftFractions(frac1,frac2,1,p); // geometrical localization
print(rg);
-> [x^2*y+4*x*y+3*y,x*y*dx+y*dx+x*dy+3*dy]
intvec v = 2;
poly s1 = y^2+y+1;
poly s2 = y-2;
frac1 = [s1,dx,0,0];
frac2 = [s2,dy,0,0];
vector rr = addLeftFractions(frac1,frac2,2,v); // rational localization
print(rr);
-> [y^3-y^2-y-2,y^2*dy+y*dx+y*dy-2*dx+dy]
```

6.5. Multiplication of left fractions

Algorithm 6.13. Multiplying two left fractions in $S^{-1}R$: `mult`

Input: $(s_1, r_1), (s_2, r_2) \in S^{-1}R$.
Output: $(s, r) \in S^{-1}R$ with $(s, r) = (s_1, r_1) \cdot (s_2, r_2)$.

```
1 begin
2 |  (s̃, r̃, J) := Ore(s2, r1, 0);
3 |  (s, r) := (s̃s1, r̃r2);
4 |  return (s, r);
5 end
```

Lemma 6.14. *Algorithm 6.13 terminates and is correct.*

Proof: `Ore` yields $\tilde{s} \in S$, $\tilde{r} \in R$ with $\tilde{s}r_1 = \tilde{r}s_2$. It holds

$$(s_1, r_1) \cdot (s_2, r_2) = s_1^{-1} r_1 s_2^{-1} r_2 = s_1^{-1} \tilde{s}^{-1} \tilde{r} r_2 = (\tilde{s}s_1, \tilde{r}r_2).$$

□

Procedure 6.15. multiplyLeftFractions(a, b, locType, locData)

Usage:	vector a, b, int locType, list/vector/intvec locData
Purpose:	multiply two left fractions in the specified localization
Assume:	
Return:	vector
Note:	the returned vector is the product of a and b as fractions in the localization specified by locType and locData.

Example. Multiplying left fractions in various localization of the second Weyl algebra:

```
ring R = 0,(x,y,dx,dy),dp;
def S = Weyl();
setring S;
poly g1 = x+3;
poly g2 = x*y+y;
list L = g1,g2;
vector frac1 = [g1,dx,0,0];
vector frac2 = [g2,dy,0,0];
vector rm = multiplyLeftFractions(frac1,frac2,0,L); // monoidal localization
print(rm);
-> [x^5*y^2+11*x^4*y^2+46*x^3*y^2+90*x^2*y^2+81*x*y^2+27*y^2,
-> x^3*y*dx*dy+7*x^2*y*dx*dy-x^2*y*dy+15*x*y*dx*dy-6*x*y*dy+9*y*dx*dy-9*y*dy]
vector p = [1,3];
vector rg = multiplyLeftFractions(frac1,frac2,1,p); // geometrical localization
print(rg);
-> [x^3*y+5*x^2*y+7*x*y+3*y,x*dx*dy+dx*dy-dy]
intvec v = 2;
poly s1 = y^2+y+1;
poly s2 = y-2;
frac1 = [s1,dx,0,0];
frac2 = [s2,dy,0,0];
vector rr1 = multiplyLeftFractions(frac1,frac2,2,v); // rational localization
print(rr1);
-> [y^3-y^2-y-2,dx*dy]
vector rr2 = multiplyLeftFractions(frac2,frac1,2,v);
print(rr2);
-> [y^5-y^3-4*y^2-3*y-2,y^2*dx*dy+y*dx*dy-2*y*dx+dx*dy-dx]
```

6.6. Equality of left fractions

Algorithm 6.16. Comparing two left fractions in $S^{-1}R$: `equal`

Input: $(s_1, r_1), (s_2, r_2) \in S^{-1}R$.
Output: TRUE, if $(s_1, r_1) = (s_2, r_2)$, FALSE otherwise.

```
1 begin
2 |  (s, r) := add((s_1, r_1), (s_2, −r_2));
3 |  if r = 0 then
4 |  |  return TRUE;
5 |  else
6 |  |  return FALSE;
7 |  end
8 end
```

Lemma 6.17. *Algorithm 6.16 terminates and is correct.*

Proof: By Lemma 3.13, $(s_1, r_1) = (s_2, r_2)$ holds if and only if

$$(1, 0) = (s_1, r_1) - (s_2, r_2) = (s_1, r_1) + (s_2, -r_2) =: (s, r),$$

which, in turn, holds if and only if $r = 0$. □

Procedure 6.18. areEqualLeftFractions(a, b, locType, locData)

Usage:	vector a, b, int locType, list/vector/intvec locData
Purpose:	check if two given fractions are equal
Assume:	
Return:	int
Note:	• returns 1, if a=b as fractions in the localization specified, • returns 0, otherwise.

Example. Comparing three left fractions in a monoidal localization of the second Weyl algebra:

```
ring R = 0,(x,y,dx,dy),dp;
def S = Weyl();
setring S;
poly g1 = x*y+3;
poly g2 = y^3;
list L = g1,g2;
vector fracm1 = [g1,dx,0,0];
vector fracm2 = [g1*g2,g2*dx,0,0];
vector fracm3 = [g1*g2,g1*dx+3,0,0];
areEqualLeftFractions(fracm1, fracm2, 0, L);
-> 1
areEqualLeftFractions(fracm1, fracm3, 0, L);
-> 0
areEqualLeftFractions(fracm2, fracm3, 0, L);
-> 0
```

6.7. Canceling left fractions

Algorithm 6.19. Canceling a left fraction in $S^{-1}R$: `cancel`

Input: $(s, r) \in S^{-1}R$.
Output: $(\tilde{s}, \tilde{r}) \in S^{-1}R$ with $(s, r) = (\tilde{s}, \tilde{r})$.

```
1 begin
2 |  Compute (s̃, r̃) via the algorithms described in Chapter 5;
3 |  return (s̃, r̃);
4 end
```

Procedure 6.20. cancelLeftFraction(frac, locType, locData)

Usage:	vector frac, int locType, list/vector/intvec locData
Purpose:	cancel a given fraction in the specified localization
Assume:	
Return:	vector
Note:	returns a canceled representation of frac in the specified localization

Example. Canceling fractions in various localizations of the second Weyl algebra:

```
ring R = 0,(x,y,dx,dy),dp;
def S = Weyl();
setring S;
poly cancel, denom, nom;
vector frac;
poly g1 = x+3;
poly g2 = x*y;
list L = g1,g2;
poly g = g1^2*g2;
poly f = dx;
frac = [g^2, g*f, 0, 0];
vector rm = cancelLeftFraction(frac, 0, L); // monoidal localization
print(rm);
-> [x^4*y^2+6*x^3*y^2+9*x^2*y^2,x*y*dx]
areEqualLeftFractions(frac, rm, 0, L);
-> 1
vector p = [0,1];
cancel = -3*(x^2 + 1);
denom = cancel * (y^2 - 3);
nom = cancel * x * (3*y*dx*dy + 7*dx^2 - 19);
frac = [denom, nom, 0, 0];
vector rg = cancelLeftFraction(frac, 1, p); // geometrical localization
print(rg);
-> [-y^2+3,-3*x*y*dx*dy-7*x*dx^2+19*x]
areEqualLeftFractions(frac, rg, 1, p);
-> 1
intvec rLoc = 1,2;
cancel = 3 * x^2 * (x+y);
denom = cancel * (x-y);
nom = cancel * (4*x + 3*y*dx^2 - 42*x^3*y*dx^5*dy^2); // rational localization
frac = [denom,nom,0,0];
vector rr = cancelLeftFraction(frac, 2, rLoc);
print(rr);
-> [x-y,-42*x^3*y*dx^5*dy^2+3*y*dx^2+4*x]
areEqualLeftFractions(frac, rr, 2, rLoc);
-> 1
```

6.8. Inverting left fractions

Algorithm 6.21. Inverting a left fraction in $S^{-1}R$: `inverse`

Input: $(s, r) \in S^{-1}R$.
Assume: $r \in S \cup R^*$.
Output: $(s, r)^{-1} \in S^{-1}R$.

```
1 begin
2 |  if r ∈ R* then
3 |  |  (s̃, r̃) := (1, r^{-1}s);
4 |  else
5 |  |  (s̃, r̃) := (r, s);
6 |  end
7 |  return (s̃, r̃);
8 end
```

Lemma 6.22. *Algorithm 6.21 terminates and is correct.*

Proof: Follows from Lemma 3.13. □

Procedure 6.23. isInvertibleLeftFraction(frac, locType, locData)

Usage:	vector frac, int locType, list/vector/intvec locData
Purpose:	check if a fraction is invertible in the specified localization
Assume:	
Return:	int
Note:	• returns 1, if frac is invertible with the nominator in the denominator set, • returns 2, if frac is invertible with the nominator invertible in the basering, • returns 0, otherwise.

Procedure 6.24. invertLeftFraction(frac, locType, locData)

Usage:	vector frac, int locType, list/vector/intvec locData
Purpose:	invert a fraction in the specified localization
Assume:	frac is invertible in the localization specified by locType and locData
Return:	vector
Note:	• returns the multiplicative inverse of frac in the localization specified, • throws error if frac is not invertible.

Example. Inverting fractions in various localizations of the second Weyl algebra:

```
ring R = 0,(x,y,dx,dy),dp;
def S = Weyl();
setring S;
poly g1 = x+3;
poly g2 = x*y;
list L = g1,g2;
vector frac = [g1*g2, 17, 0, 0];
isInvertibleLeftFraction(frac, 0, L);
-> 2
print(invertLeftFraction(frac, 0, L));
-> [1,1/17*x^2*y+3/17*x*y]
vector p = [1,0];
frac = [g1, x, 0, 0];
isInvertibleLeftFraction(frac, 1, p);
```

```
-> 1
print(invertLeftFraction(frac, 1, p));
-> [x,x+3]
intvec rat = 1,2;
frac = [g1*g2, dx, 0, 0];
isInvertibleLeftFraction(frac, 2, rat);
-> 0
```

6.9. Reduction of a left fraction (rational localization only)

The algorithm is given in Algorithmus 5.20.

Procedure 6.25. reduceLeftFraction(f, g, locType, locData)

Usage:	vector f, g, int locType, list/vector/intvec locData
Purpose:	completes one step of reducing a fraction in the specified localization
Assume:	locType is 2 (rational), $\mathrm{lm}(g) \mid \mathrm{lm}(f)$
Return:	vector
Note:	returns $f - c \cdot m \cdot g$, where • m satisfies $\mathrm{lm}(f) = \mathrm{lm}(\mathrm{lm}(m) \cdot \mathrm{lm}(g))$, • c satisfies $\mathrm{lt}(f) = c \cdot \mathrm{lt}(m \cdot g)$, • the returned fraction is not automatically simplified.

Example. Reducing a fraction in a rational localization of the second shift algebra:

```
ring S = 0,(x,y,Sx,Sy),dp;
matrix D[4][4];
D[1,3] = Sx; D[2,4] = Sy;
def ncS = nc_algebra(1, D);
setring ncS;
intvec rat = 1,2;
vector f = [x, 4*x*y+2*x*Sx*Sy-6*y*Sx*Sy, 0, 0];
vector g = [y, Sx, 0, 0];
vector rr = reduceLeftFraction(f, g, 2, rat);
print(rr);
-> [x,4*x*y]
```

As we see, the fraction $(x, 4xy)$ can be cancelled further:

```
rr = cancelLeftFraction(rr, 2, rat);
print(rr);
-> [1,4*y]
```

6.10. Procedures relevant to reduction (geometrical and rational localization only)

Procedure 6.26. rationalForm(input, locType, locData)

Usage: vector/poly input, int locType, list/vector/intvec locData

Purpose: factor out the monomials in non-invertible variables

Assume:
- locType is 1 or 2 (geometrical or rational),
- input is not the zero polynomial and not the zero vector,
- if locType is 2 (rational), locData has to be 1..n for n<=nvars(basering).

Return: list

Note:
- returns a list, consisting of [intvec alpha, poly d, poly c]-lists,
- alpha is the exponent vector of the monomial d,
- each of these lists represents a term via $c \cdot d$,
- the returned list is ordered by the d-values via the ordering on the basering.

Example. Rational form of a fraction in the second Weyl algebra:

```
ring R = 0,(x,y,dx,dy),dp;
def S = Weyl();
setring S;
vector p = [7,-1];
poly g = 4*x*dx^2*dy + 17*y^3*dx^2*dy - 3*x*dx + 4*dx + 4*y + 9;
rationalForm(g, 1, p); // geometrical localization
-> ...
intvec rLoc = 1,2;
poly denom = x^2+y;
poly nom = 4*x*dx^2*dy + 17*y^3*dx^2*dy - 3*x*dx + 4*dx + 4*y + 9;
vector frac = [denom,nom,0,0];
rationalForm(frac, 2, rLoc); // rational localization
-> ...
```

The output in both localizations is the same (reformatted for shortness and readability):

```
[1]: [1]: 2,1 [2]: dx^2*dy [3]: 17*y^3+4*x
[2]: [1]: 1,0 [2]: dx      [3]: -3*x+4
[3]: [1]: 0,0 [2]: 1       [3]: 4*y+9
```

It corresponds to the polynomial

$$(17y^3 + 4x) \cdot \partial_x^2 \partial_y + (-3x + 4) \cdot \partial_x + (4y + 9) \cdot 1.$$

Procedure 6.27. fractionLeadexp(frac, locType, locData)

Usage: vector frac, int locType, list/vector/intvec locData

Purpose: determine the leading exponent of a fraction in a specified rational localization

Assume:
- frac is not zero,
- locType is 2 (rational).

Return: intvec

Note: returns the leading exponent of frac

Example. Leading exponent of $(x, 2xQ_x - 6yQ_x + 4xy)$ in the second q-shift algebra:

```
ring Q = (0,q),(x,y,Qx,Qy),dp;
matrix C[4][4] = UpOneMatrix(4);
C[1,3] = q;
```

```
C[2,4] = q;
def ncQ = nc_algebra(C,0);
setring ncQ;
intvec locData = 1,2;
vector frac = [x, 4*x*y+2*x*Qx-6*y*Qx, 0, 0];
fractionLeadexp(frac, 2, locData);
-> 1,0
```

Procedure 6.28. fractionLeadmonom(frac, locType, locData)

Usage:	vector frac, int locType, list/vector/intvec locData
Purpose:	determine the leading monomial of a fraction in a specified rational localization
Assume:	• frac is not zero, • locType is 2 (rational).
Return:	poly
Note:	returns the leading monomial of frac

Example. Leading monomial of $(x, 2xS_x - 6yS_x + 4xy)$ in the second shift algebra:

```
ring S = 0,(x,y,Sx,Sy),dp;
matrix D[4][4];
D[1,3] = Sx; D[2,4] = Sy;
def ncS = nc_algebra(1, D);
setring ncS;
intvec locData = 1,2;
vector frac = [x, 4*x*y+2*x*Sx-6*y*Sx, 0, 0];
poly rr = fractionLeadmonom(frac, 2, locData);
print(rr);
-> Sx
```

Procedure 6.29. fractionLeadcoef(frac, locType, locData)

Usage:	vector frac, int locType, list/vector/intvec locData
Purpose:	determine the leading coefficient of a fraction in a specified rational localization
Assume:	• frac is not zero, • locType is 2 (rational).
Return:	poly
Note:	returns the leading coefficient of frac

Example. Leading coefficient of $(x, 2x\partial_x - 6y\partial_x + 4xy)$ in the second Weyl algebra:

```
ring W = 0,(x,y,dx,dy),dp;
def ncW = Weyl();
setring ncW;
intvec locData = 1,2;
vector frac = [x, 4*x*y+2*x*dx-6*y*dx, 0, 0];
vector rr = fractionLeadcoef(frac, 2, locData);
print(rr);
-> [x,2*x-6*y]
```

6.11. Validating functions

Procedure 6.30. checkFraction(a, locType, locData)

Usage: vector a, int locType, list/intvec/vector locData
Purpose: check if the given vector is a representation of a fraction in the specified localization
Assume:
Return: nothing
Note:
- throws error if checks were not successful,
- checks if at least one denominator is specified,
- checks if the specified denominators are in the specified denominator set,
- if both denominators are specified, checks if representations are equal.

Procedure 6.31. checkLocData(locType, locData)

Usage: int locType, list/vector/intvec locData
Purpose: check if the given data specifies a denominator set
Assume:
Return: nothing
Note:
- throws error if checks were not successful,
- locType can only be 0, 1, or 2.

Procedure 6.32. isInS(p, locType, locData)

Usage: poly p, int locType, list/vector/intvec locData
Purpose: determine if a polynomial is in a denominator set
Assume:
Return: int
Note:
- returns 1, if p is in the denominator set specified by locType and locData,
- returns 0, otherwise.

Example. Validating a fraction in a rational localization of the second q-shift algebra:

```
ring Q = (0,q),(x,y,Qx,Qy),dp;
matrix C[4][4] = UpOneMatrix(4);
C[1,3] = q;
C[2,4] = q;
def ncQ = nc_algebra(C,0);
setring ncQ;
intvec v = 1,2;
checkLocData(2, v);
```

In this case, no error appears, therefore $(2, v)$ describes a valid rational localization.

```
vector f = [x+14, 4*x*y+2*x*Qx-6*y*Qx, 0, 0];
checkFraction(f, 2, v);
```

Again, no error appears, which means that f is indeed a valid fraction in the localization $(2, v)$.

```
isInS(f[1], 2, v);
-> 1
```

7. Ausblick und Fazit

Aufbauend auf den in dieser Arbeit vorgestellten theoretischen Erkenntnissen und Algorithmen ergeben sich einige offene Fragen und weiterführende Probleme:
Die vielleicht wichtigste Aufgabe ist die Formulierung einer Gröbnerbasis-Theorie für Ore-lokalisierte G-Algebren. Wünschenswert wäre dabei die Beibehaltung des möglichst allgemeinen Rahmen, der in dieser Arbeit erarbeitet wurde. In Abschnitt 5.3 wurde bereits für rationale Lokalisierungen ein Algorithmus für Reduktion von Brüchen angegeben, auf dem man aufbauen könnte. Für den rationalen Fall gibt es in SINGULAR bereits eine Implementierung zur Berechnung von Gröbnerbasen, die Bibliothek `ratgb.lib` von V. Levandovskyy. Dort werden die notwendigen Berechnung mittels eines Tricks komplett bruchfrei vorgenommen, was so leider nur in der rationalen Lokalisierung möglich ist.
Ein weiterer Ansatzpunkt ist die Erweiterung von Algorithmus 5.4, also der Berechnung des Schnitts von Ideal und Ore-Menge in monoidalen Lokalisierungen, auf mehr als nur ein Polynom g_i.
Auf der Implementierungsebene ist es besonders wichtig, überflüssige Berechnungen zu vermeiden, da viele Algorithmen auf zeit- und speicheraufwändigen Gröbner-getriebenen Operationen basieren. Daher bietet sich eine objekt-orientierte Datenstruktur an, was aber zur Zeit in SINGULAR nicht realisierbar ist.
Als Möglichkeit dafür kommt das System SAGE in Frage, wo schon eine bequeme benutzerfreundliche Oberfläche für Manipulationen mit Ore-Strukturen existiert, die von B. Erocal entwickelt wurde. Dieses Paket verwendet SINGULAR als Backend, es scheint daher möglich, die in dieser Arbeit entwickelte Implementierung in das genannte SAGE-Paket zu integrieren.

Zum besten Wissen des Autors ist diese Arbeit die erste, die sich in der hier dargestellten Allgemeinheit mit der Problemstellung auseinandersetzt und nur punktuell auf die speziellen Eigenschaften einer bestimmten Lokalisierung zurückgreift.
Ebenfalls neu ist die in Form der SINGULAR-Bibliothek `olga.lib` präsentierte erste Implementierung der Algorithmen, die dem den Algorithmen unterliegenden Konzept der Allgemeinheit folgt und ein computerbasiertes Rechnen in Ore-lokalisierten G-Algebren ermöglicht.
In Abschnitt 2.2 wird in den Lemmata 2.25 bis 2.27 ein neuer ausführlicher Beweis dafür geliefert, dass die der monoidalen sowie der geometrischen Lokalisierung zugrunde liegenden Mengen tatsächlich Ore-Mengen sind.
Auch die Ergebnisse aus Abschnitt 4.2, die sich mit entgegengesetzten Lokalisierungen befassen, sind in dieser Form in der dem Autor bekannten Literatur nicht vorhanden.
Schließlich ist im Anhang der Beweis zur Konstruktion der Ore-Lokalisierung ausführlich ausgearbeitet, was in der Literatur nicht selten ganz oder in Teilen mit Aussagen wie "einfach zu überprüfen" ([MR01]) oder "Übung für den Leser" ([Š06]) vermieden wird.

8. Anhang

8.1. Konstruktion der Ore-Lokalisierung

Sei R ein Integritätsbereich und $S \subseteq R$ eine Links-Ore-Menge in R.

Satz 8.1. *Die Relation $\sim$ auf $S \times R$, gegeben durch*

$$(s_1, r_1) \sim (s_2, r_2) \quad \Leftrightarrow \quad \exists\ \tilde{s} \in S\ \exists\ \tilde{r} \in R :\ \tilde{s}s_2 = \tilde{r}s_1 \text{ und } \tilde{s}r_2 = \tilde{r}r_1,$$

ist eine Äquivalenzrelation.

Beweis: Seien $(s_1, r_1), (s_2, r_2), (s_3, r_3), (s, r) \in S \times R$.

- Reflexivität: Wählt man $\tilde{s} = 1 \in S$ und $\tilde{r} = 1 \in R$, so gilt $\tilde{s}s = s = \tilde{r}s$ und $\tilde{s}r = r = \tilde{r}r$, somit folgt $(s, r) \sim (s, r)$.
- Symmetrie: Sei $(s_1, r_1) \sim (s_2, r_2)$, dann existieren $\tilde{s} \in S$ und $\tilde{r} \in R$ mit $\tilde{s}s_2 = \tilde{r}s_1$ und $\tilde{s}r_2 = \tilde{r}r_1$. Die Ore-Bedingung liefert $\hat{s}, \bar{s} \in S$ und $\hat{r}, \bar{r} \in R$ mit $\hat{s}s_1 = \hat{r}s_2$ und $\bar{s}\hat{r} = \bar{r}\tilde{s}$. Definiert man $\mathring{s} := \bar{s}\hat{s} \in S$ sowie $\mathring{r} := \bar{r}\tilde{s} \in R$, so gilt

  $$0 = \bar{r}\tilde{s} - \bar{s}\hat{r} = (\bar{r}\tilde{s} - \bar{s}\hat{r})s_2 = \bar{r}\tilde{s}s_2 - \bar{s}\hat{r}s_2 = \bar{r}\tilde{r}s_1 - \bar{s}\hat{s}s_1 = (\bar{r}\tilde{r} - \bar{s}\hat{s})s_1,$$

  was $\bar{r}\tilde{r} = \bar{s}\hat{s}$ impliziert. Dann ist auch

  $$\mathring{s}s_1 = \bar{s}\hat{s}s_1 = \bar{r}\tilde{r}s_1 = \bar{r}\tilde{s}s_2 = \mathring{r}s_2$$

  sowie

  $$\mathring{s}r_1 = \bar{s}\hat{s}r_1 = \bar{r}\tilde{r}r_1 = \bar{r}\tilde{s}r_2 = \mathring{r}r_2,$$

  was $(s_2, r_2) \sim (s_1, r_1)$ beweist.
- Transitivität: Sei $(s_1, r_1) \sim (s_2, r_2)$ sowie $(s_2, r_2) \sim (s_3, r_3)$, dann existieren $\tilde{s}, \hat{s} \in S$ und $\tilde{r}, \hat{r} \in R$ mit $\tilde{s}s_2 = \tilde{r}s_1$, $\tilde{s}r_2 = \tilde{r}r_1$, $\hat{s}s_3 = \hat{r}s_2$ und $\hat{s}r_3 = \hat{r}r_2$. Die Ore-Bedingung liefert $\bar{s} \in S$ und $\bar{r} \in R$ mit $\bar{s}\hat{r} = \bar{r}\tilde{s}$. Definiert man $\mathring{s} := \bar{s}\hat{s} \in S$ sowie $\mathring{r} := \bar{r}\tilde{r}$, so gilt

  $$\mathring{s}s_3 = \bar{s}\hat{s}s_3 = \bar{s}\hat{r}s_2 = \bar{r}\tilde{s}s_2 = \bar{r}\tilde{r}s_1 = \mathring{r}s_1$$

  sowie

  $$\mathring{s}r_3 = \bar{s}\hat{s}r_3 = \bar{s}\hat{r}r_2 = \bar{r}\tilde{s}r_2 = \bar{r}\tilde{r}r_1 = \mathring{r}r_1,$$

  was $(s_1, r_1) \sim (s_3, r_3)$ beweist.

□

Nun kann $S^{-1}R := S \times R/\sim$ definiert werden.
Für das weitere Vorgehen seien $x := (s_1, r_1), y := (s_2, r_2), z := (s_3, r_3), (s, r) \in S^{-1}R$.

Satz 8.2. *Die binäre Operation $+$ auf $S^{-1}R$, gegeben durch*

$$+ : S^{-1}R \times S^{-1}R \to S^{-1}R, \quad (s_1, r_1) + (s_2, r_2) := (\tilde{s}s_1, \tilde{s}r_1 + \tilde{r}r_2),$$

wobei $\tilde{r} \in R$ und $\tilde{s} \in S$ die Ore-Bedingung $\tilde{s}s_1 = \tilde{r}s_2$ erfüllen, ist wohldefiniert.

Beweis: Seien $\tilde{r}, \hat{r} \in R$ sowie $\tilde{s}, \hat{s} \in S$ mit $\tilde{s}s_1 = \tilde{r}s_2$ sowie $\hat{s}s_1 = \hat{r}s_2$. Setze $(t_1, p_1) := (\tilde{s}s_1, \tilde{s}r_1 + \tilde{r}r_2)$ und $(t_2, p_2) := (\hat{s}s_1, \hat{s}r_1 + \hat{r}r_2)$. Nach Definition sind sowohl (t_1, p_1) als auch (t_2, p_2) zulässige Darstellungen für $(s_1, r_1) + (s_2, r_2)$.
Die Ore-Bedingung liefert $\bar{s} \in S$ und $\bar{r} \in R$ mit $\bar{s}t_2 = \bar{r}t_1$. Es folgt

$$0 = \bar{s}t_2 - \bar{r}t_1 = \bar{s}\hat{s}s_1 - \bar{r}\tilde{s}s_1 = (\bar{s}\hat{s} - \bar{r}\tilde{s})s_1,$$

was $\bar{s}\hat{s} = \bar{r}\tilde{s}$ impliziert, sowie

$$0 = \bar{s}t_2 - \bar{r}t_1 = \bar{s}\hat{s}s_1 - \bar{r}\tilde{s}s_1 = \bar{s}\hat{r}s_2 - \bar{r}\tilde{r}s_2 = (\bar{s}\hat{r} - \bar{r}\tilde{r})s_2,$$

was $\bar{s}\hat{r} = \bar{r}\tilde{r}$ zeigt. Dann gilt auch

$$\bar{s}p_2 - \bar{r}p_1 = \bar{s}(\hat{s}r_1 + \hat{r}r_2) - \bar{r}(\tilde{s}r_1 + \tilde{r}r_2) = (\underbrace{\bar{s}\hat{s} - \bar{r}\tilde{s}}_{=0})r_1 + (\underbrace{\bar{s}\hat{r} - \bar{r}\tilde{r}}_{=0})r_2 = 0,$$

was $(t_1, p_1) = (t_2, p_2)$ beweist. □

Satz 8.3. *Die binäre Operation $\cdot$ auf $S^{-1}R$, gegeben durch*

$$\cdot : S^{-1}R \times S^{-1}R \to S^{-1}R, \quad (s_1, r_1) \cdot (s_2, r_2) := (\tilde{s}s_1, \tilde{r}r_2),$$

wobei $\tilde{r} \in R$ und $\tilde{s} \in S$ die Ore-Bedingung $\tilde{r}s_2 = \tilde{s}r_1$ erfüllen, ist wohldefiniert.

Beweis: Seien $\tilde{r}, \hat{r} \in R$ sowie $\tilde{s}, \hat{s} \in S$ mit $\tilde{s}r_1 = \tilde{r}s_2$ sowie $\hat{s}r_1 = \hat{r}s_2$. Setze $(t_1, p_1) := (\tilde{s}s_1, \tilde{r}r_2)$ und $(t_2, p_2) := (\hat{s}s_1, \hat{r}r_2)$. Nach Definition sind sowohl (t_1, p_1) als auch (t_2, p_2) zulässige Darstellungen für $(s_1, r_1) \cdot (s_2, r_2)$.
Die Ore-Bedingung liefert $\bar{s} \in S$ und $\bar{r} \in R$ mit $\bar{s}t_2 = \bar{r}t_1$. Es folgt

$$0 = \bar{s}t_2 - \bar{r}t_1 = \bar{s}\hat{s}s_1 - \bar{r}\tilde{s}s_1 = (\bar{s}\hat{s} - \bar{r}\tilde{s})s_1,$$

was $\bar{s}\hat{s} = \bar{r}\tilde{s}$ impliziert, sowie

$$0 = \bar{s}\hat{s} - \bar{r}\tilde{s} = (\bar{s}\hat{s} - \bar{r}\tilde{s})r_1 = \bar{s}\hat{s}r_1 - \bar{r}\tilde{s}r_1 = \bar{s}\hat{r}s_2 - \bar{r}\tilde{r}s_2 = (\bar{s}\hat{r} - \bar{r}\tilde{r})s_2,$$

was $\bar{s}\hat{r} = \bar{r}\tilde{r}$ zeigt. Dann gilt auch

$$\bar{s}p_2 - \bar{r}p_1 = \bar{s}\hat{r}r_2 - \bar{r}\tilde{r}r_2 = (\underbrace{\bar{s}\hat{r} - \bar{r}\tilde{r}}_{=0})r_2 = 0,$$

was $(t_1, p_1) = (t_2, p_2)$ beweist. □

Erinnerung (Lemma 3.8)**.** Seien $s \in S$ und $r, t \in R$. Ist $ts \in S$, so gilt $(s, r) = (ts, tr)$.

Satz 8.4. *Mit den in Satz 8.2 und Satz 8.3 definierten Operationen wird $(S^{-1}R, +, \cdot)$ zum Ring.*

Beweis: Nachweis der Ringaxiome:

- Kommutativität der Addition: Es ist

$$(s_1, r_1) + (s_2, r_2) = (\tilde{s}s_1, \tilde{s}r_1 + \tilde{r}r_2) =: (t_1, p_1),$$

wobei $\tilde{s} \in S$ und $\tilde{r} \in R$ die Bedingung $\tilde{s}s_1 = \tilde{r}s_2$ erfüllen, sowie

$$(s_2, r_2) + (s_1, r_1) = (\hat{s}s_2, \hat{s}r_2 + \hat{r}r_1) =: (t_2, p_2),$$

wobei $\hat{s} \in S$ und $\hat{r} \in R$ die Bedingung $\hat{s}s_2 = \hat{r}s_1$ erfüllen. Die Ore-Bedingung liefert $\bar{s} \in S$ und $\bar{r} \in R$ mit $\bar{s}t_2 = \bar{r}t_1$. Es folgt

$$0 = \bar{s}t_2 - \bar{r}t_1 = \bar{s}\hat{s}s_2 - \bar{r}\tilde{s}s_1 = \bar{s}\hat{r}s_1 - \bar{r}\tilde{s}s_1 = (\bar{s}\hat{r} - \bar{r}\tilde{s})s_1,$$

was $\bar{s}\hat{r} = \bar{r}\tilde{s}$ impliziert, sowie

$$0 = \bar{s}t_2 - \bar{r}t_1 = \bar{s}\hat{s}s_2 - \bar{r}\tilde{s}s_1 = \bar{s}\hat{s}s_2 - \bar{r}\tilde{r}s_2 = (\bar{s}\hat{s} - \bar{r}\tilde{r})s_2,$$

was $\bar{s}\hat{s} = \bar{r}\tilde{r}$ zeigt. Dann gilt auch

$$\bar{s}p_2 - \bar{r}p_1 = \bar{s}(\hat{s}r_2 + \hat{r}r_1) - \bar{r}(\tilde{s}r_1 + \tilde{r}r_2) = (\underbrace{\bar{s}\hat{s} - \bar{r}\tilde{r}}_{=0})r_2 + (\underbrace{\bar{s}\hat{r} - \bar{r}\tilde{s}}_{=0})r_1 = 0,$$

was $(t_1, p_1) = (t_2, p_2)$ beweist.

- Assoziativität der Addition: Es ist

$$x + y = (s_1, r_1) + (s_2, r_2) = (\tilde{s}s_1, \tilde{s}r_1 + \tilde{r}r_2) =: (a_1, b_1),$$

wobei $\tilde{s} \in S$ und $\tilde{r} \in R$ die Bedingung $\tilde{s}s_1 = \tilde{r}s_2$ erfüllen,

$$(x + y) + z = (a_1, b_1) + (s_3, r_3) = (\hat{s}a_1, \hat{s}b_1 + \hat{r}r_3) =: (t_1, p_1),$$

wobei $\hat{s} \in S$ und $\hat{r} \in R$ die Bedingung $\hat{s}a_1 = \hat{r}s_3$ erfüllen,

$$y + z = (s_2, r_2) + (s_3, r_3) = (\bar{s}s_2, \bar{s}r_2 + \bar{r}r_3) =: (a_2, b_2),$$

wobei $\bar{s} \in S$ und $\bar{r} \in R$ die Bedingung $\bar{s}s_2 = \bar{r}s_3$ erfüllen,

$$x + (y + z) = (s_1, r_1) + (a_2, b_2) = (\mathring{s}s_1, \mathring{s}r_1 + \mathring{r}b_2) =: (t_2, p_2),$$

wobei $\mathring{s} \in S$ und $\mathring{r} \in R$ die Bedingung $\mathring{s}s_1 = \mathring{r}a_2$ erfüllen. Die Ore-Bedingung liefert $s' \in S$ und $r' \in R$ mit $s't_2 = r't_1$. Es folgt

$$0 = s't_2 - r't_1 = s'\mathring{s}s_1 - r'\hat{s}a_1 = s'\mathring{s}s_1 - r'\hat{s}\tilde{s}s_1 = (s'\mathring{s} - r'\hat{s}\tilde{s})s_1,$$

was $s'\mathring{s} = r'\hat{s}\tilde{s}$ impliziert,

$$0 = s't_2 - r't_1 = s'\mathring{s}s_1 - r'\hat{s}a_1 = s'\mathring{r}a_2 - r'\hat{s}\tilde{s}s_1 = s'\mathring{r}\bar{s}s_2 - r'\hat{s}\tilde{r}s_2 = (s'\mathring{r}\bar{s} - r'\hat{s}\tilde{r})s_2,$$

was $s'\mathring{r}\bar{s} = r'\hat{s}\tilde{r}$ zeigt, sowie

$$0 = s't_2 - r't_1 = s'\mathring{s}s_1 - r'\hat{s}a_1 = s'\mathring{r}a_2 - r'\hat{r}s_3 = s'\mathring{r}\bar{s}s_2 - r'\hat{r}s_3 = s'\mathring{r}\bar{r}s_3 - r'\hat{r}s_3 = (s'\mathring{r}\bar{r} - r'\hat{r})s_3,$$

was $s'\mathring{r}\bar{r} = r'\hat{r}$ verdeutlicht. Dann gilt auch

$$\begin{aligned} s'p_2 - r'p_1 &= s'(\mathring{s}r_1 + \mathring{r}b_2) - r'(\hat{s}b_1 + \hat{r}r_3) \\ &= s'(\mathring{s}r_1 + \mathring{r}(\bar{s}r_2 + \bar{r}r_3)) - r'(\hat{s}(\tilde{s}r_1 + \tilde{r}r_2) + \hat{r}r_3) \\ &= s'\mathring{s}r_1 + s'\mathring{r}\bar{s}r_2 + s'\mathring{r}\bar{r}r_3 - r'\hat{s}\tilde{s}r_1 - r'\hat{s}\tilde{r}r_2 - r'\hat{r}r_3 \\ &= (\underbrace{s'\mathring{s} - r'\hat{s}\tilde{s}}_{=0})r_1 + (\underbrace{s'\mathring{r}\bar{s} - r'\hat{s}\tilde{r}}_{=0})r_2 + (\underbrace{s'\mathring{r}\bar{r} - r'\hat{r}}_{=0})r_3 \\ &= 0, \end{aligned}$$

was $(x + y) + z = (t_1, p_1) = (t_2, p_2) = x + (y + z)$ beweist.

- Neutrales Element der Addition: Es gilt

$$(s,r)+(1,0)=(\tilde{s}s,\tilde{s}r+\tilde{r}\cdot 0)=(\tilde{s}s,\tilde{s}r)=(s,r),$$

wobei $\tilde{s}\in S$ und $\tilde{r}\in R$ die Bedingung $\tilde{s}s=\tilde{r}\cdot 1=\tilde{r}$ erfüllen. Folglich ist $0_{S^{-1}R}=(1,0)$.

- Inverse Elemente der Addition: Siehe Lemma 3.13.

- Assoziativität der Multiplikation: Es ist

$$x\cdot y=(s_1,r_1)\cdot(s_2,r_2)=(\tilde{s}s_1,\tilde{r}r_2)=:(a_1,b_1),$$

wobei $\tilde{s}\in S$ und $\tilde{r}\in R$ die Bedingung $\tilde{s}r_1=\tilde{r}s_2$ erfüllen,

$$(x\cdot y)\cdot z=(a_1,b_1)\cdot(s_3,r_3)=(\hat{s}a_1,\hat{r}r_3)=:(t_1,p_1),$$

wobei $\hat{s}\in S$ und $\hat{r}\in R$ die Bedingung $\hat{s}b_1=\hat{r}s_3$ erfüllen,

$$y\cdot z=(s_2,r_2)\cdot(s_3,r_3)=(\bar{s}s_2,\bar{r}r_3)=:(a_2,b_2),$$

wobei $\bar{s}\in S$ und $\bar{r}\in R$ die Bedingung $\bar{s}r_2=\bar{r}s_3$ erfüllen,

$$x\cdot(y\cdot z)=(s_1,r_1)\cdot(a_2,b_2)=(\mathring{s}s_1,\mathring{r}b_2)=:(t_2,p_2),$$

wobei $\mathring{s}\in S$ und $\mathring{r}\in R$ die Bedingung $\mathring{s}r_1=\mathring{r}a_2$ erfüllen. Die Ore-Bedingung liefert $s'\in S$ und $r'\in R$ mit $s't_2=r't_1$. Es folgt

$$0=s't_2-r't_1=s'\mathring{s}s_1-r'\hat{s}a_1=s'\mathring{s}s_1-r'\hat{s}\tilde{s}s_1=(s'\mathring{s}-r'\hat{s}\tilde{s})s_1,$$

was $s'\mathring{s}=r'\hat{s}\tilde{s}$ impliziert. Damit ist

$$0=(s'\mathring{s}-r'\hat{s}\tilde{s})r_1=s'\mathring{s}r_1-r'\hat{s}\tilde{s}r_1=s'\mathring{r}a_2-r'\hat{s}\tilde{r}s_2=s'\mathring{r}\bar{s}s_2-r'\hat{s}\tilde{r}s_2=(s'\mathring{r}\bar{s}-r'\hat{s}\tilde{r})s_2,$$

was $s'\mathring{r}\bar{s}=r'\hat{s}\tilde{r}$ zeigt. Nun folgt

$$0=(s'\mathring{r}\bar{s}-r'\hat{s}\tilde{r})r_2=s'\mathring{r}\bar{s}r_2-r'\hat{s}\tilde{r}r_2=s'\mathring{r}\bar{r}s_3-r'\hat{s}b_1=s'\mathring{r}\bar{r}s_3-r'\hat{r}s_3=(s'\mathring{r}\bar{r}-r'\hat{r})s_3,$$

was $s'\mathring{r}\bar{r}=r'\hat{r}$ verdeutlicht. Dann gilt auch

$$s'p_2-r'p_1=s'\mathring{r}b_2-r'\hat{r}r_3=s'\mathring{r}\bar{r}r_3-r'\hat{r}r_3=(\underbrace{s'\mathring{r}\bar{r}-r'\hat{r}}_{=0})r_3=0,$$

was $(x\cdot y)\cdot z=(t_1,p_1)=(t_2,p_2)=x\cdot(y\cdot z)$ beweist.

- Neutrales Element der Multiplikation: Es gilt

$$(s,r)\cdot(1,1)=(\tilde{s}s,\tilde{r}\cdot 1),$$

wobei $\tilde{s}\in S$ und $\tilde{r}\in R$ die Bedingung $\tilde{s}r=\tilde{r}\cdot 1=\tilde{r}$ erfüllen. Damit folgt

$$(s,r)\cdot(1,1)=(\tilde{s}s,\tilde{r})=(\tilde{s}s,\tilde{s}r)=(s,r).$$

Andererseits ist

$$(1,1)\cdot(s,r)=(\tilde{s}\cdot 1,\tilde{r}r),$$

wobei $\tilde{s}\in S$ und $\tilde{r}\in R$ die Bedingung $\tilde{s}=\tilde{s}\cdot 1=\tilde{r}s$ erfüllen. Nun gilt

$$(1,1)\cdot(s,r)=(\tilde{s},\tilde{r}r)=(\tilde{r}s,\tilde{r}r)=(s,r).$$

Folglich ist $1_{S^{-1}R}=(1,1)$.

- Links-Distributivität: Es ist

$$y + z = (s_2, r_2) + (s_3, r_3) = (\tilde{s}s_2, \tilde{s}r_2 + \tilde{r}r_3) =: (a_1, b_1),$$

wobei $\tilde{s} \in S$ und $\tilde{r} \in R$ die Bedingung $\tilde{s}s_2 = \tilde{r}s_3$ erfüllen,

$$x \cdot (y + z) = (s_1, r_1) \cdot (a_1, b_1) = (\hat{s}s_1, \hat{r}b_1) =: (t_1, p_1),$$

wobei $\hat{s} \in S$ und $\hat{r} \in R$ die Bedingung $\hat{s}r_1 = \hat{r}a_1$ erfüllen,

$$x \cdot y = (s_1, r_1) \cdot (s_2, r_2) = (\bar{s}s_1, \bar{r}r_2) =: (a_2, b_2),$$

wobei $\bar{s} \in S$ und $\bar{r} \in R$ die Bedingung $\bar{s}r_1 = \bar{r}s_2$ erfüllen,

$$x \cdot z = (s_1, r_1) \cdot (s_3, r_3) = (\mathring{s}s_1, \mathring{r}r_3) =: (a_3, b_3),$$

wobei $\mathring{s} \in S$ und $\mathring{r} \in R$ die Bedingung $\mathring{s}r_1 = \mathring{r}s_3$ erfüllen,

$$x \cdot y + x \cdot z = (a_2, b_2) + (a_3, b_3) = (s^*a_2, s^*b_2 + r^*b_3) =: (t_2, p_2),$$

wobei $s^* \in S$ und $r^* \in R$ die Bedingung $s^*a_2 = r^*a_3$ erfüllen. Die Ore-Bedingung liefert $s' \in S$ und $r' \in R$ mit $s't_2 = r't_1$. Es folgt

$$0 = s't_2 - r't_1 = s's^*a_2 - r'\hat{s}s_1 = s's^*\bar{s}s_1 - r'\hat{s}s_1 = (s's^*\bar{s} - r'\hat{s})s_1,$$

was $s's^*\bar{s} = r'\hat{s}$ impliziert. Damit ist

$$0 = (s's^*\bar{s} - r'\hat{s})r_1 = s's^*\bar{s}r_1 - r'\hat{s}r_1 = s's^*\bar{r}s_2 - r'\hat{r}a_1 = s's^*\bar{r}s_2 - r'\hat{r}\tilde{s}s_2 = (s's^*\bar{r} - r'\hat{r}\tilde{s})s_2,$$

was $s's^*\bar{r} = r'\hat{r}\tilde{s}$ zeigt. Außerdem ist

$$0 = s't_2 - r't_1 = s's^*a_2 - r'\hat{s}s_1 = s'r^*a_3 - r'\hat{s}s_1 = s'r^*\mathring{s}s_1 - r'\hat{s}s_1 = (s'r^*\mathring{s} - r'\hat{s})s_1,$$

was $s'r^*\mathring{s} = r'\hat{s}$ verdeutlicht. Dann folgt

$$\begin{aligned} 0 &= (s'r^*\mathring{s} - r'\hat{s})r_1 = s'r^*\mathring{s}r_1 - r'\hat{s}r_1 = s'r^*\mathring{r}s_3 - r'\hat{r}a_1 = s'r^*\mathring{r}s_3 - r'\hat{r}\tilde{s}s_2 \\ &= s'r^*\mathring{r}s_3 - r'\hat{r}\tilde{r}s_3 = (s'r^*\mathring{r} - r'\hat{r}\tilde{r})s_3, \end{aligned}$$

was $s'r^*\mathring{r} = r'\hat{r}\tilde{r}$ nahelegt. Dann gilt auch

$$\begin{aligned} s'p_2 - r'p_1 &= s'(s^*b_2 + r^*b_3) - r'\hat{r}b_1 = s's^*b_2 + s'r^*b_3 - r'\hat{r}(\tilde{s}r_2 + \tilde{r}r_3) \\ &= s's^*\bar{r}r_2 + s'r^*\mathring{r}r_3 - r'\hat{r}\tilde{s}r_2 - r'\hat{r}\tilde{r}r_3 \\ &= (\underbrace{s's^*\bar{r} - r'\hat{r}\tilde{s}}_{=0})r_2 + (\underbrace{s'r^*\mathring{r} - r'\hat{r}\tilde{r}}_{=0})r_3 \\ &= 0, \end{aligned}$$

was $x \cdot (y + z) = (t_1, p_1) = (t_2, p_2) = x \cdot y + x \cdot z$ beweist.

- Rechts-Distributivität: Es ist

$$x + y = (s_1, r_1) + (s_2, r_2) = (\tilde{s}s_1, \tilde{s}r_1 + \tilde{r}r_2) =: (a_1, b_1),$$

wobei $\tilde{s} \in S$ und $\tilde{r} \in R$ die Bedingung $\tilde{s}s_1 = \tilde{r}s_2$ erfüllen,

$$(x + y) \cdot z = (a_1, b_1) \cdot (s_3, r_3) = (\hat{s}a_1, \hat{r}r_3) =: (t_1, p_1),$$

wobei $\hat{s} \in S$ und $\hat{r} \in R$ die Bedingung $\hat{s}b_1 = \hat{r}s_3$ erfüllen,

$$x \cdot z = (s_1, r_1) \cdot (s_3, r_3) = (\bar{s}s_1, \bar{r}r_3) =: (a_2, b_2),$$

wobei $\bar{s} \in S$ und $\bar{r} \in R$ die Bedingung $\bar{s}r_1 = \bar{r}s_3$ erfüllen,

$$y \cdot z = (s_2, r_2) \cdot (s_3, r_3) = (\mathring{s}s_2, \mathring{r}r_3) =: (a_3, b_3),$$

wobei $\mathring{s} \in S$ und $\mathring{r} \in R$ die Bedingung $\mathring{s}r_2 = \mathring{r}s_3$ erfüllen,

$$x \cdot z + y \cdot z = (a_2, b_2) + (a_3, b_3) = (s^*a_2, s^*b_2 + r^*b_3) =: (t_2, p_2),$$

wobei $s^* \in S$ und $r^* \in R$ die Bedingung $s^*a_2 = r^*a_3$ erfüllen. Die Ore-Bedingung liefert $s' \in S$ und $r' \in R$ mit $s't_2 = r't_1$. Es folgt

$$0 = s't_2 - r't_1 = s's^*a_2 - r'\hat{s}a_1 = s'r^*a_3 - r'\hat{s}\tilde{s}s_1 = s'r^*\mathring{s}s_2 - r'\hat{s}\tilde{r}s_2 = (s'r^*\mathring{s} - r'\hat{s}\tilde{r})s_2,$$

was $s'r^*\mathring{s} = r'\hat{s}\tilde{r}$ impliziert. Weiterhin gilt

$$0 = s't_2 - r't_1 = s's^*a_2 - r'\hat{s}a_1 = s's^*\bar{s}s_1 - r'\hat{s}\tilde{s}s_1 = (s's^*\bar{s} - r'\hat{s}\tilde{s})s_1,$$

was $s's^*\bar{s} = r'\hat{s}\tilde{s}$ zeigt. Damit ist

$$\begin{aligned} 0 &= (s's^*\bar{s} - r'\hat{s}\tilde{s})r_1 = s's^*\bar{s}r_1 - r'\hat{s}\tilde{s}r_1 = s's^*\bar{r}s_3 - r'\hat{s}(b_1 - \tilde{r}r_2) \\ &= s's^*\bar{r}s_3 - r'\hat{s}b_1 + r'\hat{s}\tilde{r}r_2 = s's^*\bar{r}s_3 - r'\hat{r}s_3 + s'r^*\mathring{s}r_2 \\ &= s's^*\bar{r}s_3 - r'\hat{r}s_3 + s'r^*\mathring{r}s_3 = (s's^*\bar{r} - r'\hat{r} + s'r^*\mathring{r})s_3, \end{aligned}$$

was $s's^*\bar{r} - r'\hat{r} + s'r^*\mathring{r} = 0$ verdeutlicht. Dann gilt auch

$$\begin{aligned} s'p_2 - r'p_1 &= s'(s^*b_2 + r^*b_3) - r'\hat{r}r_3 = s's^*b_2 + s'r^*b_3 - r'\hat{r}r_3 \\ &= s's^*\bar{r}r_3 + s'r^*\mathring{r}r_3 - r'\hat{r}r_3 = (\underbrace{s's^*\bar{r} + s'r^*\mathring{r} - r'\hat{r}}_{=0})r_3 \\ &= 0, \end{aligned}$$

was $(x + y) \cdot z = (t_1, p_1) = (t_2, p_2) = x \cdot z + y \cdot z$ beweist.

□

Literatur

[And10] Daniel Andres. *Algorithms for the computation of Sato's b-functions in algebraic D-module theory*. Diplomarbeit, RWTH Aachen, 2010. URL `http://www.math.rwth-aachen.de/~Daniel.Andres/bfct.pdf`.

[BGTV03] José Bueso, José Gómez-Torrecillas, Alain Verschoren. *Algorithmic Methods in Non-Commutative Algebra: Applications to Quantum Groups*, volume 17 of *Mathematical Modelling Series*. Kluwer Academic Publishers, first edition, 2003. ISBN 978-90-481-6328-1.

[FLW09] Guofeng Fu, Ziming Li, Min Wu. *A Detailed Verification for Ore Localization*, 2009. URL `http://www.mmrc.iss.ac.cn/pub/mm28.pdf/04-liziming.pdf`.

[GP07] Gert-Martin Greuel, Gerhard Pfister. *A Singular Introduction to Commutative Algebra*. Springer, 2nd edition, 2007. ISBN 978-3-540-73541-0.

[Lev05] Viktor Levandovskyy. *Non-commutative Computer Algebra for polynomial algebras: Gröbner bases, applications and implementation*. Dissertation, Universität Kaiserslautern, 2005. URL `https://kluedo.ub.uni-kl.de/frontdoor/index/index/docId/1670`.

[LKM11] Viktor Levandovskyy, Christoph Koutschan, Oleksandr Motsak. *On Two-Generated Non-commutative Algebras Subject to the Affine Relation*. In *Proceedings of Computer Algebra in Scientific Computing (CASC)*, volume 6885 of *Lecture Notes in Computer Science*, pages 309–320. Springer, 2011. ISBN 978-3-642-23567-2.

[LS12] Viktor Levandovskyy, Kristina Schindelar. *Fraction-free algorithm for the computation of diagonal forms matrices over Ore domains using Gröbner bases*. *Journal of Symbolic Computation*, volume 47(10): pages 1214–1232, 2012. ISSN 0747-7171. URL `http://www.sciencedirect.com/science/article/pii/S0747717111002458`.

[MR01] John C. McConnell, J. Chris Robson. *Noncommutative Noetherian Rings*, volume 30 of *Graduate studies in mathematics*. American Mathematical Society, 2001. ISBN 9780821821695.

[SIN] *HTML User Manual for Singular*. URL `http://www.singular.uni-kl.de/Manual/latest/index.htm`.

[Š06] Zoran Škoda. *Noncommutative localization in noncommutative geometry. London Mathematical Society Lecture Note Series 330*, pages 220–310, 2006. URL `http://arxiv.org/abs/math/0403276v2`.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt und durch meine Unterschrift, dass die vorliegende Arbeit von mir selbstständig und ohne fremde Hilfe angefertigt worden ist. Inhalte und Passagen, die aus fremden Quellen stammen und direkt oder indirekt übernommen worden sind, wurden als solche kenntlich gemacht. Ferner versichere ich, dass ich keine andere außer der im Literaturverzeichnis angegebenen Literatur verwendet habe. Diese Versicherung bezieht sich sowohl auf Textinhalte als auch auf alle enthaltenden Abbildungen, Skizzen und Tabellen. Die Arbeit wurde bisher keiner Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Aachen, den 14. Juli 2014