Diplomarbeit zum Thema

Implementation und Vergleich von Methoden zur Berechnung des Radikals eines Polynomideals

zur Erlangung des akademischen Grades

vorgelegt dem
Fachbereich Mathematik
der Universität Dortmund

von
Carsten Liesen

im März 2007

Betreuer der Diplomarbeit: Prof. Dr. Martin Kreuzer

Inhaltsverzeichnis

Tabellenverzeichnis

1 Einleitung

In der vorliegenden Arbeit werden Methoden zur Berechnung des Radikals eines Polynomideals implementiert und miteinander verglichen. Dabei handelt es sich um die in den Artikeln von Kemper [11], Matsumoto [17] und Laplagne [15] beschriebenen Algorithmen. Diese drei Algorithmen stellen die zur Zeit aktuellsten Methoden zur Berechnung des Radikals eines Polynomideals dar. Der Vergleich dieser drei Methoden soll Unterschiede und Gemeinsamkeiten bei der Berechnung des Radikals eines Polynomideals herausstellen. Auf dieser Grundlage werden die Methoden bzgl. einer optimalen Nutzung für jeweilige Beispiele untersucht.
Das Berechnen des Radikals eines Ideals in einem Polynomring ist eine zentrale Aufgabe in der kommutativen Algebra. So werden Radikale zur Berechnung einer Primärzerlegung eines Ideals in Polynomringen verwendet (siehe Becker und Weispfenning [2] oder Seidenberg [19]) oder auch als Hilfsmittel im Algorithmus von de Jong zur Berechnung der Normalisierung von bestimmten Ringen (siehe de Jong [10]).
Viele Autoren benutzen für die Berechnung des Radikals eines Ideals I die Methode der Reduktion auf den null-dimensionalen Fall (siehe Becker und Weispfenning [2], Krick und Logar [14], Gianni et al. [9]).
In diesem Fall wird eine maximal unabhängige Menge von Variablen modulo des Ideals I berechnet, die eine Teilmenge der Variablenmenge ist. Diese Teilmenge sei beispielsweise die Menge $\{x_1, \ldots, x_d\}$. Das Ideal I in $K[x_1, \ldots, x_n]$ wird zu einem Ideal in $K(x_1, \ldots, x_d)[x_{d+1}, \ldots, x_n]$ erweitert und als $\tilde{I}$ bezeichnet. Das Ideal $\tilde{I}$ ist hier null-dimensional. Es folgt die Berechnung des Radikals des Ideals $\tilde{I}$. Für den null-dimensionalen Fall geben Becker und Weispfenning [2], Gianni [8] Algorithmen an, die das Radikal berechnen, wenn der Körper K vollkommen ist. Kemper [11] hat dagegen einen Algorithmus angegeben, der das Radikal eines Ideals in $K[x_1, \ldots, x_n]$, in dem K ein endlich erzeugter Körper über einen vollkommenen Körper ist, berechnet. Der Algorithmus in Kemper [11] bezieht sich auf den Fall, dass der Körper K nicht vollkommen ist. Dieser Fall ist dann interessant, wenn das Radikal eines höher-dimensionalen Ideals in einem Polynomring mit einem Körper von positiver Charakteristik zu berechnen ist.
Für ein höher-dimensionales Ideal gibt es mehrere Algorithmen, das Radikal zu berechnen. Wenn die Charakteristik des Körpers K hinreichend groß oder null ist, sind dies die Algorithmen in Gianni et al. [9], Krick und Logar [14], Eisenbud et al. [6], Wang [20], Armendáriz und Pablo[1], Caboara et al. [3], Wang [21]. Für kleine Charakteristiken sind die Algorithmen in Eisenbud et. al. [6], Wang [20], Caboara et. al. [3] anwendbar. Der Algorithmus in Eisenbud et. al. [6] ist nur für den Fall umsetzbar, dass der Körper K die Charakteris-

tik null hat oder dass für die Elemente, die das Radikal von I erzeugen, der Index der Nilpotenz kleiner als die Charakteristik des Körpers K ist. Es ist zu erwähnen, dass der Algorithmus in Eisenbud et. al. [6] nicht die Methode anwendet, die höher-dimensionalen Ideale auf den null-dimensionalen Fall zu reduzieren.
Eine weitere Methode das Radikal eines Ideals ohne die Reduktion auf den null-dimensionalen Fall zu verwenden, wird in Matsumoto [17] angegeben. Dieser Algorithmus ist durchführbar, wenn die Charakteristik klein ist. Ist die Charakteristik dagegen nicht hinreichend klein, ist der Algorithmus rechnerisch nicht durchzuführen.
Die von Laplagne vorgebrachte Methode zur Berechnung des Radikals eines beliebigen Ideals verwendet wiederum den Ansatz, das Problem auf den null-dimensionalen Fall zu reduzieren. Dieser Algorithmus beruht auf den Ideen von Gianni et al [9] und von Krick und Logar [14]. Der in Laplagne [15] angegebene Algorithmus ist in einigen Fällen effizienter bei der Berechnung, da im Gegensatz zum Algorithmus in Krick und Logar [14] die Berechnung redundanter Komponenten vermieden wird.
In dem der Einleitung folgenden Kapitel werden die notwendigen mathematischen Grundlagen für die Berechnung der drei Algorithmen von Kemper, Matsumoto und Laplagne aufgeführt und definiert.
Im dritten Kapitel werden die Algorithmen nacheinander vorgestellt sowie mathematisch bewiesen. Das Kapitel schließt mit einem Vergleich der Algorithmen von Kemper, Matsumoto und Laplagne.
Das darauf folgende vierte Kapitel beinhaltet zwei große Abschnitte. In dem ersten Abschnitt werden die Impementierungen der zur jeweiligen Berechnung notwendigen Algorithmen dargestellt und fundiert. In einem weiteren Abschnitt werden die Ergebnisse der Berechnung von Beispielen analysiert. Die einzelnen Beispiele sowie deren Ergebnisse der Berechnung werden tabellarisch im Anhang A dargestellt.
Das Fazit stellt das letzte Kapitel der vorliegenden Arbeit dar. Hier werden die wichtigen Ergebnisse zusammenfassend dargestellt und ein Ausblick auf mögliche neue Fragestellungen gegeben.
Im Anhang ist neben den Tabellen zur Berechnung der Beispiele eine Information zur beiliegenden CD-Rom zu finden.

2 Mathematische Grundlagen

In diesem Kapitel werden die mathematischen Grundlagen für die Berechnung des Radikals eines Polynomideals dargestellt. In einem ersten Abschnitt wird das Radikal eines Ideals definiert. Im weiteren Verlauf werden die verschiedenen zur Berechnung notwendigen mathematischen Voraussetzungen und Hilfsmittel erläutert.

2.1 Das Radikal eines Ideals

In diesem Abschnitt wird das Radikal eines Ideals definiert und anhand von Beispielen beschrieben. In diesem Zusammenhang werden weiterhin die Eigenschaften von Radikalen dargelegt.

Definition 2.1.1.
Sei K ein Körper, $K[x_1, \ldots, x_n]$ ein Polynomring in n Variablen über K und I ein Ideal in $K[x_1, \ldots, x_n]$.
*Das **Radikal** von I ist die Menge $\{f \in K[x_1, \ldots, x_n] \mid f^i \in I \text{ für ein } i \geq 1\}$ und diese wird als $\sqrt{I}$ bezeichnet.*

Beispiel 2.1.2.
Das Radikal des Ideals $I_1 = \langle x_1^2, x_2^4 \rangle$ in $\mathbb{Q}[x_1, x_2]$ ist $\sqrt{I_1} = \langle x_1, x_2 \rangle$ und $\sqrt{I_2} = \langle x_1 - 2, x_2 - 2 \rangle$ ist das Radikal von $I_2 = \langle x_1^2 - x_2^2 - 4x_1 + 4x_2, x_1 - 2 \rangle$ in $Q[x_1, x_2]$.

Bemerkung 2.1.3.
Es ist zu beachten, dass immer $I \subset \sqrt{I}$ gilt. Dies gilt, da $f^1 \in I$ aus $f \in I$ folgt und somit mit der Definition des Radikals $f \in \sqrt{I}$ folgt.
Dabei gilt im Allgemeinen jedoch nicht $I = \sqrt{I}$,
denn für das Ideal $J = \langle x_1^2, x_2^3 \rangle \subset \mathbb{R}[x_1, x_2]$ gilt $x_1 \in \sqrt{J}$ und $x_2 \in \sqrt{J}$ und nicht $x_1 \in J$ und $x_2 \in J$.

Definition 2.1.4.
Sei I ein Ideal in $K[x_1, \ldots, x_n]$.
*Ein Ideal I heißt **Radikalideal**, wenn $I = \sqrt{I}$ gilt.*

Beispiele für Radikalideale sind Primideale, die in der Definition 2.7.1 erläutert werden.
Der folgende Satz zeigt, dass das Radikal ein Ideal ist.

Satz 2.1.5.
Wenn I ein Ideal in $K[x_1, \ldots, x_n]$ ist, dann ist $\sqrt{I}$ ein Ideal in $K[x_1, \ldots, x_n]$, das I enthält. Außerdem ist $\sqrt{I}$ ein Radikalideal.

Der Beweis wird von Cox,Little und O'Shea (vgl. Lemma 4.2.5, [4]) durchgeführt.

Satz 2.1.6.
Seien I und J beliebige Ideale. Dann gilt $\sqrt{I \cap J} = \sqrt{I} \cap \sqrt{J}$

Cox,Little und O'Shea beweisen dies (vgl. Proposition 4.3.16, [4]).

Die Definition und die Eigenschaften eines Radikals sind für alle weiteren Abschnitte der vorliegenden Arbeit grundlegend.

2.2 Der separable Teil eines Polynoms

In diesem Abschnitt werden die separablen Teile von Polynomen betrachtet, die im Algorithmus von Kemper 3.1.6 zur Berechnung eines Radikals verwendet werden. Hier folgen die Eigenschaften des separablen Teils eines Polynoms auf die Definition desselben.

Im Folgenden sei K ein Körper und $\bar{K}$ der algebraische Abschluß von K, d.h. es gibt einen Erweiterungskörper $\bar{K}$ von K, so daß jedes Polynom $f \in K[x]$ eine Faktorisierung in lineare Faktoren in $\bar{K}[x]$ besitzt.
Der größte gemeinsame Teiler von f in $K[x]$ wird hierbei durch $\mathrm{ggT}_{K[x]}(f)$ und das kleinste gemeinsame Vielfache von f in $K[x]$ durch $\mathrm{kgV}_{K[x]}(f)$ dargestellt.

Definition 2.2.1.

1. *Sei $f \in K[x]$ ein irreduzibles Polynom.*
   *Das Polynom f heißt f **separabel**, falls es keine mehrfachen Wurzeln in $\bar{K}$ besitzt.*

2. *Ein beliebiges Polynom $f \in K[x]$ heißt **separabel**, wenn alle seine irreduziblen Faktoren separabel sind.*

Der nächste Satz stellt eine Beziehung zwischen der Teilerfremdheit eines Polynoms $f \in K[x]$ und seiner Ableitung f' zur Separabilität des Polynoms her.

Satz 2.2.2.
Ein Polynom $f \in K[x]$ ist genau dann separabel, wenn $\mathrm{ggT}_{K[x]}(f, f') = 1$ gilt.

Den Beweis legen Becker und Weispfenning (vgl. 7.33, [2]) dar.

Definition 2.2.3.
Sei $f \in K[x]\backslash\{0\}$ gegeben. Seien $\alpha_1, \ldots, \alpha_m \in \bar{K}$ die paarweise verschiedenen Wurzeln von f. Dann gibt es eine Darstellung $f = c\prod_{i=1}^{m}(x-\alpha_i)^{e_i}$ mit einer Einheit $c \in K\backslash\{0\}$ und $e_1, \ldots, e_m \in \mathbb{N}_+$. In dieser Situation heißt $\mathrm{sep}(f) := c \cdot \prod_{i=1}^{m}(x-\alpha_i) \in \bar{K}[x]$ der **separable Teil von** f**.**

Hierbei ist darauf zu achten, dass der separable Teil bis auf eine Einheit eindeutig ist. Die Berechnung des separablen Teils eines Polynoms wird in Algorithmus 3.1.1 der vorliegenden Arbeit gezeigt.

Beispiel 2.2.4.
In einem Körper der Charakteristik 0 ist jedes irreduzible Polynom separabel. Denn für einen Körper K der Charakteristik 0 und ein irreduzibles Polynom $f \in K[x]\backslash K$ ist die Ableitung $f' \neq 0$. Aus Meyberg (siehe Kapitel 6.6 Korollar 1, [18]) folgt, dass f keine mehrfachen Wurzeln hat. Damit ist f separabel.

Neben dem separablen Teil eines Polynoms ist auch der quadratfreie Teil eines Polynoms von Bedeutung. Dieser ist unter bestimmten Voraussetzungen für den Körper mit dem separablen Teil eines Polynoms identisch.

Definition 2.2.5.
Sei K ein Körper. Ein Polynom $f \in K[x]$ heißt ***quadratfrei****, wenn $p^2 \nmid f$ für alle irreduziblen $p \in K[x]$ gilt.*

Definition 2.2.6.
Sei K ein Körper. Sei $f \in K[x] \setminus K$ ein Polynom $f = cf_1^{\alpha_1} \cdots f_m^{\alpha_m}$ mit einer Einheit $c \in K$, mit $m \in \mathbb{N}$ und $\alpha_1, \ldots, \alpha_m \in \mathbb{N}_+$ die Zerlegung von $f \in K[x]$ in normierte, paarweise verschiedene irreduzible Faktoren f_i, $i = \{1, \ldots, m\}$. Das Polynom $\mathrm{sqfree}(f) = cf_1 \cdots f_m$ wird ein ***quadratfreier Teil*** *von f genannt.*

Da sich quadratfreie Teile von f nur um den Faktor einer Einheit voneinander unterscheiden, wird im Folgenden von **dem** quadratfreien Teil von f gesprochen.
Die Berechnung des quadratfreien Teils eines univariaten Polynoms zeigt der folgende Satz.

Satz 2.2.7.
Sei K ein Körper und $f \in K[x]$ ein nicht konstantes Polynom.
Falls der Körper K die Charakteristik Null hat oder die Charakteristik $p > 0$ hat und das Polynom f die Form $f = cf_1^{\alpha_1} \cdots f_t^{\alpha_t}$ hat, in dem $c \in K \setminus \{0\}$ ist, die $\alpha_1, \ldots, \alpha_t > 0$ für $i = 1, \ldots, t$ der Teilbarkeit $p \nmid \alpha_i$ genügen und die

$f_1, \ldots, f_t$ paarweise verschiedene irreduzible monomische Polynome sind, so ist der quadratfreie Teil von f durch $\text{sqfree}(f) = f / \text{ggT}_{K[x]}(f, f')$ *gegeben.*

Der Beweis wird durch Kreuzer und Robbiano (vgl. Proposition 3.7.9 b, [12]) vorgelegt.

Satz 2.2.8.
Sei K ein Körper und $f \in K[x]$ ein nicht konstantes Polynom. Wenn $\text{ggT}_{K[x]}(f, f') = 1$ *gilt, dann ist f quadratfrei.*

Der Beweis ist in Kreuzer und Robbiano (vgl. Proposition 3.7.9 a, [12]) zu finden.

Die Umkehrung ist im Allgemeinen nicht richtig. Hierzu ist die folgende Definition für den Körper K notwendig.

Definition 2.2.9.
Ein Körper K heißt ***vollkommen****, falls K entweder die Charakteristik 0 hat oder die Charakteristik $p > 0$ ist und $K = K^p$ gilt, d.h. jedes Element in K hat eine $p-te$ Wurzel in Kt.*

Beispiel 2.2.10.
Jeder endliche Körper ist vollkommen.

Die oben erwähnte Umkehrung wird im Folgenden gezeigt.

Satz 2.2.11.
Sei K ein vollkommener Körper und $f \in K[x]$ ein nicht konstantes Polynom. Wenn f quadratfrei ist, dann gilt $\text{ggT}_{K[x]}(f, f') = 1$.

Der Beweis wird ebenfalls durch Kreuzer und Robbiano (vgl. Proposition 3.7.9 d, [12]) gezeigt.

Satz 2.2.12.
Ist der Körper K vollkommen, so stimmt der separable Teil eines Polynoms mit dessen quadratfreien Teil überein.

Der oben genannte Satz wird durch die Sätze 2.2.11 und 2.2.2 bewiesen, denn hieraus folgt, dass der separable Teil und der quadratfreie Teil eines Polynoms in vollkommenen Körpern identisch ist.

Bemerkung 2.2.13.
Die Berechnung des quadratfreien Teils eines univariaten Polynoms in einem Polynomring $K[x]$ mit einem vollkommenen Körper der Charakteristik $p > 0$ zeigt Proposition 3.7.12 in [12].

2.3 Gröbner-Basen

In diesem Abschnitt werden Gröbner-Basen definiert, die im weiteren Verlauf unter anderem für die Berechnung von Eliminationsidealen genutzt werden. Dazu werden zunächst einige notwendige Begriffe erläutert, die für die Definition von Gröbner-Basen von Bedeutung sind.

Die Menge $\mathbb{T}^n$ ist die Menge aller Terme in den Variablen $x_1, \ldots, x_n$, d.h. es gilt $\mathbb{T}^n = \{x_1^{\alpha_1} \cdots x_n^{\alpha_n} \mid \alpha_1, \ldots, \alpha_n \in \mathbb{N}\}$. Damit entschieden werden kann, wie die Terme in $\mathbb{T}^n$ geordnet werden können, werden Termordnungen benötigt. Die Definition einer Termordnung wird durch Kreuzer und Robbiano (vgl. Definition 1.4.1, [12]) gegeben.

Sei K ein Körper, $n \geq 1$, $P = K[x_1, \ldots, x_n]$ ein Polynomring und σ eine Termordnung auf $\mathbb{T}^n$.
Jedes Polynom $f \in P \setminus \{0\}$ hat eine eindeutige Darstellung $f = c_1t_1 + \cdots + c_st_s$ mit $c_1, \ldots, c_s \in K \setminus \{0\}$ und $t_1, \ldots, t_s \in \mathbb{T}^n$, so dass $t_1 >_\sigma t_2 >_\sigma \cdots >_\sigma t_s$ gilt.

Definition 2.3.1.
Sei wie oben dargestellt ein Polynom f gegeben.

1. *Der Term* $\mathrm{LT}_\sigma(f) = t_1$ *heißt der* ***Leitterm*** *von f bzgl.* σ.
2. *Der Term* $\mathrm{LC}_\sigma(f) = c_1$ *heißt der* ***Leitkoeffizient*** *von f bzgl.* σ.
3. *Der Term* $\mathrm{LM}_\sigma(f) = c_1t_1$ *heißt das* ***Leitmonom*** *von f bzgl.* σ.

Die entsprechende Definition für Ideale von Leittermen wird nachfolgend beschrieben.

Definition 2.3.2.
Sei I ein Ideal in P. Das Ideal $\mathrm{LT}_\sigma(I) = \langle \mathrm{LT}_\sigma(f) \mid f \in I \setminus \{0\} \rangle$ *heißt das* ***Leittermideal*** *von I bzgl.* σ.

Leittermideale sind monomiale Ideale und werden von endlich vielen Termen erzeugt. Jeder Term t im Leittermideal hat die Form $t = \mathrm{LT}_\sigma(f)$ mit $f \in I$ (vgl. Proposition 1.5.6, [12]).

Diese Begriffe bilden die Grundlage für eine Defintion von Gröbner-Basen, die nachfolgend dargestellt wird.

Definition 2.3.3.
Sei σ *eine Termordnung auf* $\mathbb{T}^n$. *Sei* $G = \{g_1, \ldots, g_s\}$ *eine endliche Menge*

von Polynomen, die das Ideal $I = \langle g_1, \ldots, g_s \rangle \subseteq P$ erzeugen.
Die Menge G heißt ***σ-Gröbner-Basis von I****, wenn*

$$\langle \mathrm{LT}_\sigma(g_1), \ldots, \mathrm{LT}_\sigma(g_s) \rangle = \langle \mathrm{LT}_\sigma(I) \rangle$$

gilt.

Die Existenz von Gröbner-Basen wird von Kreuzer und Robbiano (vgl. Prop. 2.4.3, [12]) bewiesen. Darüber hinaus ist die Berechnung von Gröbner-Basen möglich.

Satz 2.3.4.
Sei σ eine Termordnung auf $\mathbb{T}^n$. Sei $G = \{g_1, \ldots, g_s\}$ eine endliche Teilmenge von Polynomen $g_i \neq 0$, die das Ideal $I = \langle g_1, \ldots, g_s \rangle \subseteq P$ erzeugen. Auf dieser Grundlage berechnet der Buchberger-Algorithmus in endlich vielen Schritten eine σ-Gröbner-Basis von I.

Der Buchberger-Algorithmus ist in Kreuzer und Robbiano (vgl. Theorem 2.5.5, [12]) angegeben und seine Korrektheit wird dort gezeigt.

2.4 Eliminationsideale

Gröbner-Basen können in verschiedenen Gebieten angewendet werden. Ein Anwendungsgebiet stellt die Berechnung von Eliminationsidealen dar, die im Folgenden zunächst beschrieben und schließlich mit Hilfe von Gröbner-Basen berechnet werden.

Sei K ein Körper, $P = K[x_1, \ldots, x_n]$ ein Polynomring und $I \subseteq P$ ein Ideal. Weiterhin sei $L \subseteq \{x_1, \ldots, x_n\}$ eine Teilmenge der Variablenmenge und $\widehat{P} = K[x_i | x_i \notin L]$ der Polynomring in den restlichen Variablen.

Definition 2.4.1.

1. *Eine Termordnung σ auf $\mathbb{T}^n$ heißt eine* ***Eliminationsordnung für L****, wenn jedes Element $m \in P \setminus \{0\}$, dessen Leitterm $LT_\sigma(m)$ in $\widehat{P}$ enthalten ist, auch in $\widehat{P}$ enthalten ist.*

2. *Sei I ein Ideal in P. Dann heißt das Ideal $I \cap \hat{P}$ in $\hat{P}$ das* ***Eliminationsideal von I bzgl. L****.*

Eine Eliminationsordnung für L genügt somit der Eigenschaft, dass jeder Term, der durch eine Variable in L teilbar ist, größer als alle Terme in $\widehat{P}$ ist.

Beispiel 2.4.2.
Sei $L = \{x_1, \ldots, x_j\}$ *für* $j \in \{1, \ldots, n-1\}$. *Seien* $t_1 = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ *und* $t_2 = x_1^{\beta_1} \cdots x_n^{\beta_n}$ *Terme in* $\mathbb{T}^n$. *Es gilt* $t_1 \geq_{\texttt{Elim(L)}} t_2$, *wenn* $\alpha_1 + \cdots + \alpha_j > \beta_1 + \cdots + \beta_j$ *ist oder wenn* $\alpha_1 + \cdots + \alpha_j = \beta_1 + \cdots + \beta_j$ *und* $t_1 \geq_{\texttt{DegRevLex}} t_2$ *ist. Die dadurch definierte Termrdnung auf* $\mathbb{T}^n$ *wird* `Elim(L)` *genannt und ist eine Eliminationsordnung für L.*

Beispiel 2.4.3.
Die lexikografische Termordnung `Lex` *ist eine Eliminationsordnung für* $\{x_1, \ldots, x_i\}$ *mit* $1 \leq i < n$.

Für ein Ideal I kann das entsprechende Eliminationsideal berechnet werden, indem die Gröbner-Basis von I bzgl. einer Eliminationsordnung σ auf $\mathbb{T}^n$ berechnet wird. Daraus läßt sich ein Erzeugendensystem des entsprechenden Eliminationsideals bestimmen.

Satz 2.4.4.
Sei σ *eine Eliminationsordnung für L. Die Restriktion* $\hat{\sigma}$ *von* σ *ist eine Termordnung auf* $\widehat{\mathbb{T}}$, *der Menge der Terme in den Variablen* $\{x_i | x_i \notin L\}$, *in* $\widehat{P}$.
Sei G eine σ*-Gröbner-Bbasis von I und sei* $\widehat{G}$ *die Menge aller Elemente von G, die in* $\widehat{P}$ *enthalten sind.*
Dann ist die Menge $\widehat{G}$ *eine* $\hat{\sigma}$*-Gröbnerbasis von* $I \cap \widehat{P}$.

Der Beweis wird von Kreuzer und Robbiano (vgl. Theorem 3.4.5, [12]) dargestellt.

Dieser Satz hat verschiedene Anwendungsmöglichkeiten. So läßt sich mit Hilfe dieses Satzes zum Beispiel der Durchschnitt von zwei Idealen (vgl. Proposition 3.4.6, [12]) oder auch das Colonideal (vgl. Proposition 3.4.9, [12]) berechnen.

2.5 Unabhängige Variablen und Dimensionen

Im folgenden Abschnitt wird zunächst die maximal unabhängige Menge modulo eines Ideals sowie die Dimension eines Ideals definiert. Weiterhin wird der Algorithmus MAXINDEPSET zur Berechnung einer maximal unabhängigen Menge modulo eines Ideals vorgestellt.

Sei K ein Körper und sei I ein echtes Ideal im Polynomring $P = K[x_1, \ldots, x_n]$. Für jede Teilmenge $Y \subseteq \{x_1, \ldots, x_n\}$ sei der Polynomring $K[x_i | x_i \in Y]$ mit $K[Y]$ bezeichnet und das Ideal $\langle x_i \mid x_i \in Y \rangle$ sei mit $\langle Y \rangle$ bezeichnet.

Definition 2.5.1.
Sei $Y \subseteq \{x_1, \ldots, x_n\}$ eine Menge von Variablen.

1. *Die Menge Y wird* ***unabhängig modulo I*** *oder eine* ***unabhängige Menge von Variablen modulo I*** *genannt, wenn $I \cap K[Y] = (0)$ gilt.*

2. *Die Menge Y heißt* ***maximal unabhängige Menge modulo I****, wenn Y unabhängig modulo I ist und es keine Menge $Z \subseteq \{x_1, \ldots, x_n\}$ gibt, die $Y \subset Z$ erfüllt und unabhängig modulo I ist.*

3. *Die größte Zahl von Elementen einer maximal unabhängigen Menge von Variablen modulo I wird die* ***Dimension*** *von P/I genannt und mit $\dim(P/I)$ bezeichnet.*

Bemerkung 2.5.2.
Diese Dimension stimmt mit anderen Definitionen der Dimension überein (vgl. Kapitel 5.7, [13]).

Für die Berechnung der Dimension und einer maximal unabhängigen Menge von Variablen ist noch eine spezielle Termordnung zu definieren.

Für ein Polynom f wird im Folgenden mit $\deg(f)$ der Grad eines Polynoms f bezeichnet.

Definition 2.5.3.
Sei σ eine Monoidordnung auf $\mathbb{T}^n$. Die Termordnung σ heißt ***gradkompatibel****, wenn aus $t_1 \geq_\sigma t_2$ für $t_1, t_2 \in \mathbb{T}^n$ folgt, dass $\deg(t_1) \geq \deg(t_2)$ gilt.*

Beispiele für gradkompatible Termordnungen stellen die graduiert-lexikografische Termordnung `DegLex` und die graduiert-umgekehrt-lexikografische Termordnung `DegRevLex` dar, die in Kreuzer und Robbiano (siehe Kapitel 1.4, [12]) beschrieben werden.

Der folgend beschriebene Algorithmus MAXINDEPSET ermöglicht die Berechnung der Dimension und der maximal unabhängigen Mengen modulo eines Ideals I.

In der vorliegenden Arbeit werden alle Algorithmen dem folgenden Aufbau entsprechend dargestellt.
Zunächst wird der Input vorgelegt, der die Startposition des Algorithmus beschreibt. Darauffolgend wird der Output aufgezeigt, der das Ergebnis der Berechnungen des Algorithmus darstellt. Erst danach wird der Algorithmus

mit seinen einzelnen Schritten in der Beschreibung vorgestellt. Im Anschluß wird in einem Satz die Terminiertheit und Korrektheit bewiesen und anhand eines Beispiels wird der Algorithmus getestet.

Algorithmus 2.5.4 (MAXINDEPSET)**.**

Input: *Sei I ein echtes Ideal I in $P = K[x_1, \ldots, x_n]$ und σ eine gradkompatible Termordnung auf $\mathbb{T}^n$.*

Output: *Eine maximal unabhängige Menge Y modulo I.*

Beschreibung: MAXINDEPSET(I)

1. *Berechne das Leittermideal $\mathrm{LT}_\sigma(I)$ mit Hilfe des Buchberger-Algorithmus.*
2. *Berechne das minimale monomiale Erzeugendensystem $t_1, \ldots, t_s$ von $\sqrt{\mathrm{LT}_\sigma(I)}$, indem die quadratfreien Teile der Terme einer monomialen Erzeugendenmenge von $\mathrm{LT}_\sigma(I)$ genommen werden und von diesen Termen , diejenigen Terme zu entfernen sind, die echte Vielfache von anderen Termen sind.*
3. *Wende die Prozedur* INDEPSET(Y, T) *mit $Y = \{x_1, \ldots, x_n\}$ und $T = \{t_1, \ldots, t_s\}$ an. Gebe dieses Ergebnis aus und stoppe.*

Es folgt eine Beschreibung der rekursiven Prozedur INDEPSET.

Input: *Die Mengen Y und T.*

Output: *Eine Menge mit der größten Anzahl von Elementen.*

Beschreibung: INDEPSET(Y, T)

1. *Wenn $T = \{\}$ gilt, dann gebe Y aus und stoppe.*
2. *Für jedes $x_i \in Y$ überprüfe, ob es t_1 teilt. Ist dies der Fall, so sei $T_i = \{t \in T \mid x_i \nmid t$.*
   Wende die Prozedur INDEPSET($Y \setminus \{x_i\}, T_i$) *an und bezeichne ihr Resultat mit L_i.*
3. *Sei L_j eine der Mengen L_i mit der größten Anzahl an Elementen. Gebe dann L_j aus und stoppe.*

Satz 2.5.5.
Der Algorithmus MAXINDEPSET *berechnet für ein Ideal I eine maximal unabhängige Menge Y modulo I mit der größten Elementenanzahl.*
Insbesondere gibt der Algorithmus MAXINDEPSET *durch die Anzahl der Elemente von Y die Dimension* $\dim(P/I)$ *an.*

Kreuzer und Robbiano beweisen dies (vgl. Korollar 5.7.10, [13]).

Die Anwendung des Algorithmus MAXINDEPSET wird im folgenden Beispiel dargestellt.

Beispiel 2.5.6.
Sei $I = \langle x_1^2 + 2x_1x_2x_3 + x_3^4, x_2x_3 - x_3^2\rangle$ *ein Ideal in* $\mathbb{Z}/(7)[x_1, x_2, x_3]$ *und* $\sigma =$ `DegRevLex` *sei als Termordnung auf* $\mathbb{T}(x_1, x_2, x_3)$ *gewählt.*
Es wird $\mathrm{LT}_\sigma(I) = \langle x_2x_3, x_3^4, x_1^2x_2\rangle$ *berechnet und es ist* $\sqrt{I} = \langle x_3, x_1x_2\rangle$. *In Schritt 3 ist* $Y = \{x_1, x_2, x_3\}$ *und* $T = \{x_1x_2, x_3\}$.
Es wird dazu INDEPSET(Y, T) *angewendet.*
In Schritt 2 ist rekursiv INDEPSET$(\{x_2, x_3\}, \{x_3\})$ *über*
INDEPSET$(\{x_2\}, \{\}) = \{x_2\}$ *zu berechnen und es wird*
INDEPSET$(\{x_2, x_3\}, \{x_3\}) = \{x_2\}$ *hierfür ausgegeben.*
Die Menge INDEPSET$(\{x_1, x_3\}, \{x_3\})$ *wird ebenso ermittelt und*
es ergibt sich INDEPSET$(\{x_1, x_3\}, \{x_3\}) = \{x_1\}$.
Der Output von INDEPSET(Y, T) *ist die Menge* $\{x_2\}$ *und der Algorithmus gibt als eine maximal unabhängige Menge modulo* I *die Menge* $\{x_2\}$ *aus.*
Es ist damit $\dim(\mathbb{Z}/(7)[x_1, x_2, x_3]/I) = 1$.

Beispiel 2.5.7.
Sei K *ein Körper und* $P = K[x_1, x_2, x_3]$.
Sei $I = \langle x_1^2+x_2^2+x_3^2\rangle$ *und* $\sigma =$ `DegLex`. *Dann gibt der Algorithmus die Menge* $\{x_2, x_3\}$ *als maximal unabhängige Menge modulo* I *aus. Die Dimension von* P/I *ist somit* 2.

Definition 2.5.8.
Ein Ideal I *in* $K[x_1, \ldots, x_n]$ *heißt* ***null-dimensional****, wenn das Ideal* I *in* $K[x_1, \ldots, x_n]$ *echt ist und* P/I *die Dimension null hat.*

Beispiel 2.5.9.
Die Ideale $I_1 = \langle x_1^3, x_1^2x_2 + x_1, x_2^2\rangle$ *in* $\mathbb{Q}[x_1, x_2]$ *und*
$I_2 = \langle x_1^2 - x_1, x_2^2 - x_2, x_3^2 - x_3, x_1x_2, x_1x_3, x_2x_3\rangle$ *in* $\mathbb{Z}/(3)[x_1, x_2, x_3]$ *sind null-dimensional.*

Bemerkung 2.5.10.
Die null-dimensionalen Ideale werden im nächsten Abschnitt verwendet, wenn höher-dimensionale Ideale zu null-dimensionale Ideale mit geeigneter Anpassung der Ringe kontrahiert werden.

Beispiel 2.5.11.
Für das Ideal $I = \langle x_1^2 + x_2^2 + x_3^2\rangle$ *in* $P = K[x_1, x_2, x_3]$ *aus Beispiel 2.5.7 ist* $\dim(P/I) = 2$ *und für* I *in* $K(x_2, x_3)[x_1]$ *gilt* $\dim(K(x_2, x_3)[x_1]/I) = 0$.

2.6 Kontraktions- und Erweiterungsideale

Kontraktionsideale und Erweiterungsideale eines Ideals werden von Kemper und Laplagne für die Berechnung des Radikals eines Ideals genutzt (vgl. [11] und [15]). In diesem Abschnitt werden die Kontraktionsideale und Erweiterungsideale zunächst definiert. Anschließend werden die Algorithmen CONT und EXTCONT beschrieben, die für die Berechnung der Kontraktionsideale und Erweiterungsideale genutzt werden.

Definition 2.6.1.
Sei K ein Körper und $Y = \{y_1, \ldots, y_m\}$ und $Z = \{z_1, \ldots, z_k\}$ sind algebraisch unabhängige Mengen von Variablen über K.

1. *Sei I ein Ideal in $K(Y)[Z]$. Dann heißt das durch $I \cap K[Y, Z]$ definierte Ideal das **Kontraktionsideal von I nach** $K[Y, Z]$ und wird mit I^c bezeichnet.*

2. *Sei $I \subseteq K[Y, Z]$ ein Ideal. Dann heißt das durch I in $K(Y)[Z]$ erzeugte Ideal das **Erweiterungsideal von I nach** $K(Y)[Z]$ und wird mit I^e bezeichnet.*

Die folgenden Sätze werden zum Beweis der Algorithmen CONT und EXTCONT benötigt.

Satz 2.6.2.
Sei I ein Ideal in einem Ring R und sei M eine multiplikativ abgeschlossene Teilmenge von R mit $1 \in M$ und $0 \notin M$.
Dann gilt $I^e = \{a/s \mid a \in I, s \in M\}$.

Den Beweis liefern Becker und Weispfenning (vgl. Lemma 1.122, [2]).

Auf der Grundlage dieses Satzes werden die Erzeuger von I in R als Erzeuger des Erweiterungsideal betrachtet. Dies wird im Beweis von Algorithmus RADIKALKEMPER verwendet.

Der folgende Satz, der die null-dimensionalen Ideale verwendet, wird ebenfalls für den Beweis des Algorithmus genutzt.

Satz 2.6.3.
Sei I ein Ideal in $K[x_1, \ldots, x_n]$ und I^e das Erweiterungsideal von I nach $K(x_1, \ldots, x_d)[x_{d+1}, \ldots, x_n]$ für ein $d \in \{1, \ldots, n\}$.
Wenn $\{x_1, \ldots, x_d\}$ eine maximal unabhängige Menge modulo I ist, dann ist I^e ein null-dimensionales Ideal in $K(x_1, \ldots, x_d)[x_{d+1}, \ldots, x_n]$.

Becker und Weispfenning beweisen dies (vgl. Lemma 7.47, [2]).

Sei K ein Körper und es sei $K[X] = K[x_1, \ldots, x_n]$.
Im Folgenden wird der Algorithmus CONT beschrieben, der für ein Ideal in $K[X]$ eine Gröbnerbasis des Kontraktionsideals bzgl. einer geeigneten Termordnung berechnet.

Algorithmus 2.6.4 (CONT)**.**

Input: *Sei Y eine Teilmenge von $X = \{x_1, \ldots, x_n\}$ und sei I ein endlich erzeugtes Ideal in $K(Y)[X \setminus Y]$.*

Output: *Eine Gröbnerbasis des Ideals I^c aus $K[x]$*

Beschreibung: CONT(I, Y)

1. *Berechne eine Gröbnerbasis H von I bzgl. einer beliebigen Termordnung auf $\mathbb{T}(X \setminus Y)$.*
2. *Für alle $h \in H$ bestimme den kleinsten gemeinsamen Vielfachen aus allen Nennern der Koeffizienten in $K(Y)$ von h, bezeichne diesen mit q und setze $h := q \cdot h$.*
3. *Berechne den* $\mathrm{kgV}_{K[Y]}(\mathrm{LC}(h)|h \in H)$ *und nenne ihn f.*
4. *Bestimme Erzeugendensystem von $\langle H \rangle : f^\infty := \{g \in K[X] | f^i g \subseteq \langle H \rangle \quad \text{für ein} \quad i \in \mathbb{N}\}$, der Saturierung von $\langle H \rangle$ durch f in $K[X]$. Bezeichne sie mit G.*
5. *Gebe G aus.*

Satz 2.6.5.
Der Algorithmus CONT *berechnet für eine Teilmenge Y von $\{x_1, \ldots, x_n\}$ und einem endlich erzeugten Ideal I in $K(Y)[X \setminus Y]$ eine Gröbnerbasis des Ideals I^c aus $K[x]$*

Den Beweis legen Becker und Weispfenning vor (vgl. Proposition 8.92, [2]).

Für den Algorithmus EXTCONT zur Bestimmung des Erweiterungsideals wird eine spezielle Termordnung benötigt, die nachfolgend eingeführt wird.

Definition 2.6.6.
Sei σ_1 eine Termordnung auf $\mathbb{T}(Y)$ und σ_2 eine Termordnung auf $\mathbb{T}(X \setminus Y)$. Dann sei σ eine Termordnung auf $\mathbb{T}(X)$ von folgender Form:
Für $y_1, y_2 \in \mathbb{T}(Y)$ und $x_1, x_2 \in \mathbb{T}(X \setminus Y)$ gelte $y_1 x_1 \leq_\sigma y_2 x_2$,
wenn $x_1 \leq_{\sigma_2} x_2$ sei oder wenn $x_1 = x_2$ und $y_1 \leq_{\sigma_1} y_2$ sei.
Diese Ordnung heißt ***inverse Blockordnung bzgl. Y.***

Mit den Voraussetzungen der vorhergehenden Definition für x_1, x_2, y_1, y_2 zeigt sich, dass $x_1 \leq_\sigma x_2$ aus $y_1 x_1 \leq_\sigma y_2 x_2$ folgt.
Die Bestimmung der Erzeugenden eines Erweiterungsideals wird durch den folgenden Satz ermöglicht.

Satz 2.6.7.
Sei σ eine inverse Blockordnung auf $\mathbb{T}(X)$ bzgl. Y und σ' die Restriktion von σ auf $\mathbb{T}(X \setminus Y)$. Sei $G \subseteq K[X]$ eine σ-Gröbner-Basis. Dann ist G eine σ'-Gröbner-Basis in $K[X \setminus Y]$.

Der Beweis wird von Becker und Weispfenning angegeben (vgl. Lemma 8.93, [2]).

Im Folgenden wird auf die Beziehung zwischen einen Ideal I in $K[X]$ und dem Ideal I^{ec} eingegangen. Aufgrund der Definition von Kontraktionsidealen und Erweiterungsidealen gilt immer die Inklusion $I \subseteq I^{ec}$. Die Umkehrung ist im allgemeinen falsch.

Für das Ideal I berechnet der nächste Satz aus seiner Gröbner-Basis das Ideal I^{ec}.

Satz 2.6.8.
Sei σ eine inverse Blockordnung auf $\mathbb{T}(X)$ bzgl. Y und σ' die Restriktion von σ auf $\mathbb{T}(X \setminus Y)$. Sei I ein Ideal in $K[X]$ und sei G eine σ-Gröbner-Basis von I. Sei $f = \mathrm{kgV}_{K(Y)[X \setminus Y]}(\{\mathrm{LT}_{\sigma'}(g) \mid g \in G\})$.
Dann gilt $I^{ec} = I : f^\infty$.

Becker und Weispfennig zeigen dies (vgl. Proposition 8.94, [2]).

Auf der Grundlage von I^{ec} kann mit dem folgenden Algorithmus ein Ideal J berechnet werden, dass die Gleichung $I = J \cap I^{ec}$ erfüllt.

Die Ordnung σ sei dazu eine inverse Blockordnung auf $\mathbb{T}(X)$ bzgl. Y und σ' sei die Restriktion von σ auf $\mathbb{T}(X \setminus Y)$.

Algorithmus 2.6.9 (EXTCONT)**.**

Input: *Sei Y eine Teilmenge von X und I ein Ideal in $K[X]$.*

Output: *Ein Polynom $f \in K[Y]$ und ein $s \in \mathbb{N}$, so dass gilt*
$$I = (I + \langle f^s \rangle) \cap I^{ec}.$$

Beschreibung: EXTCONT(I, Y)

1. *Berechne eine σ-Gröbner-Basis G von I.*
2. *Für ein $g \in G$ sei $\mathrm{LC}_{\sigma'}(g) \in K[Y]$ der Leitkoeffizient von g und g sei als Element aus $K(Y)[X \setminus Y]$ zu sehen. Bestimme den $\mathrm{kgV}_{K(Y)[X\setminus Y]}(\mathrm{LC}_{\sigma'}(g)|g \in G) \in K[Y]$ und bezeichne ihn mit f.*
3. *Sei s eine natürliche Zahl mit $I : f^s = I : f^\infty$.*
4. *Gebe (f, s) aus.*

Satz 2.6.10.
Der Algorithmus EXTCONT *berechnet für eine Teilmenge Y von X und einem endlich erzeugten Ideal I in $K[X]$ ein Polynom $f \in K[Y]$ und ein $s \in \mathbb{N}$, so dass $I = (I + \langle f^s \rangle) \cap I^{ec}$ gilt.*

Der Beweis wird von Becker und Weispfenning (vgl. Proposition 8.96, [2]) dargestellt.

Dieser Algorithmus dient als Hilfsmittel das Radikal eines Ideals zu berechnen. Zum Abschluß des Unterkapitels wird auf Beziehungen von Radikalen und Kontraktionsidealen eingegangen.

Satz 2.6.11.
Falls J ein Ideal in $K(Y)[X \setminus Y]$ ist, dann gilt $(\sqrt{J})^c = \sqrt{J^c}$.
In diesem Fall ist das erste Radikal ein Ideal in $K(Y)[X \setminus Y]$ und das zweite ein Ideal in $K[X]$

Den Beweis liefern Becker und Weispfenning (vgl. Lemma 8.97, [2]).

2.7 Primärzerlegung

In diesem Abschnitt werden Primideale und Primärideale sowie die mögliche Zerlegung in Primärideale von Idealen in Noetherschen Ringen vorgestellt. Diese Ideale und deren Zerlegung werden für den Algorithmus 3.3.2 der vorliegenden Arbeit benötigt.

Definition 2.7.1.
Ein echtes Ideal I in einem Ring R heißt ***Primideal****, wenn aus $ab \in I$ folgt, dass $a \in I$ oder $b \in I$ für alle $a, b \in R$ gilt.*
Ein Primideal J in R heißt ***minimales Primideal*** *in R, wenn es kein Primideal J_1 mit $J_1 \subset J$ gibt.*

Noethersche Ringe, d.h. Ringe, in denen jede aufsteigende Kette von Idealen schließlich stationär wird, besitzen endlich viele minimale Primideale. Dies zeigt der folgende Satz.

Satz 2.7.2.
Sei R ein Noetherscher Ring.
Es gibt endlich viele minmale Primideale in R.

Den Beweis führen Kreuzer und Robbiano (vgl. Proposition 5.6.15, [13]).

Im Folgenden sei K ein Körper und $K[X] := K[x_1, \ldots, x_n]$ der Polynomring in n Variablen.

Definition 2.7.3.
Es sei I ein Ideal im Ring R. Ein Ideal I heißt ***Primärideal****, falls für alle $a, b \in R$ mit $ab \in R$ und $a \notin R$ es ein $b \in \sqrt{I}$ gibt.*

Es folgt ein Beispiel für Primärideale.

Beispiel 2.7.4.
Sei $I = \langle x_1^2, x_1 x_2 \rangle$ ein Ideal in $K[x_1, x_2]$. Die Ideale $\langle x \rangle$, $\langle x, y \rangle^2$ und $\langle x^2, y \rangle$ sind Primärideale in $K[x_1, x_2]$.

Definition 2.7.5.
Sei der Ring R Noethersch und sei I ein echtes Ideal in R.
Eine Zerlegung $I = Q_1 \cap \cdots \cap Q_s$ mit Primäridealen $Q_1, \ldots, Q_s$ in R heißt ***Primärzerlegung*** *von I.*
Eine Primärzerlegung von I heißt ***irredundant*** *, falls die folgenden Bedingungen erfüllt sind:*

1. *Für $i, j \in \{1, \ldots, s\}$ mit $i \neq j$ gilt $\sqrt{Q_i} \neq \sqrt{Q_j}$.*
2. *Für $i = 1, \ldots, s$ gilt $\bigcap_{j \neq i} Q_j \not\subseteq Q_i$.*

Bemerkung 2.7.6.
Die Existenz einer Primärzerlegung folgt aus Proposition 5.6.18 in [12].

Die Eindeutigkeit einer Primärzerlegung ist im Allgemeinen nicht gegeben, wie das folgende Beispiel zeigt.

Beispiel 2.7.7.
Sei I das Ideal aus Beispiel 2.7.4. Primärzerlegungen von I sind $\langle x \rangle \cap \langle x, y \rangle^2$ und $\langle x \rangle \cap \langle x^2, y \rangle$. Diese Primärzerlegungen sind zudem irredundant.

Satz 2.7.8.
Sei I ein echtes Ideal in $K[X]$, $\mathcal{P}$ eine Teilmenge von minimalen Primteilern von I und das Ideal $\tilde{P} := \bigcap_{P \in \mathcal{P}} P$ der Durchschnitt dieser minimalen Primteiler.

Es sei angenommen, dass ein $g \in \tilde{P} \setminus \sqrt{I}$ *existiert. Wenn* $I : g^\infty = \cap_{i=1}^{s} Q_i$ *eine irredundante Primärzerlegung ist und* Y *die maximal unabhängige Menge modulo* $I : g^\infty$ *darstellt, dann ist für alle* $i \in \{1, \ldots, s\}$ *mit* $Q_i \cap K[Y] = \{0\}$ *das Ideal* $\sqrt{Q_i}$ *ein minimaler Primteiler von* I*.*
Es gilt weiterhin $\sqrt{Q_i} \notin \mathcal{P}$*.*

Den Beweis liefert Laplagne (vgl. Proposition 4, [15]).

Die in diesem Kapitel vorgestellten mathematischen Grundlagen bilden die Basis für die im folgenden Kapitel dargestellten Methoden zur Berechnung des Radikals eines Ideals.

3 Algorithmen

In diesem Kapitel werden drei verschiedene Methoden zur Berechnung des Radikals eines Ideals vorgestellt. Die verschiedenen Methoden werden nacheinander mit ihren individuellen Vorgaben vorgestellt und in einem letzten Abschnitt zu einem Vergleich gegenübergestellt.

3.1 Der Algorithmus von Kemper

Dieser Algorithmus zur Berechnung des Radikals eines Polynomideals verwendet den separablen Teil eines Polynoms. Zunächst wird daher die Berechnung dieses separablen Teils eines Polynoms mit Hilfe des Algorithmus SEP durchgeführt. Desweiteren wird für den Algorithmus von Kemper das Lemma von Seidenberg als Hilfsmittel benötigt und kurz beschrieben.
Im Anschluß daran werden der Algorithmus nach Kemper sowie der Algorithmus RADIKALKEMPER ausführlich definiert und dargestellt.

Im Folgenden seien $y_1, \ldots, y_m$ Variablen und k sei ein vollkommener Körper der Charakteristik $p > 0$. Der Körper K sei der rationale Funktionenkörper $k(y_1, \ldots, y_m)$. Dabei wird angenommen, dass zumindest im anschließenden Algorithmus für jedes Element aus k die p-te Wurzel berechenbar sei. Der folgende Algorithmus berechnet den separablen Teil $\mathrm{sep}(f)$ von $f \in K[x]$.

Algorithmus 3.1.1 (SEP)**.**

Input: *Ein Polynom $f \neq 0$ in $k(y_1, \ldots, y_m)[x]$*

Output: *Der separable Teil von f als ein Polynom in $k(\sqrt[q]{y_1}, \ldots, \sqrt[q]{y_m})$, wobei q eine Potenz von p ist.*

Beschreibung: SEP(f)

1. *Setze $h := \mathrm{ggT}_{K[x]}(f, f')$.*
2. *Setze $g_1 := f/h$.*
3. *Setze $\tilde{h} := \mathrm{ggT}_{K[x]}(h, h')$.*
4. *Falls $\tilde{h} = h$ gilt, gehe zu 6.*
5. *Setze $h := \tilde{h}$ und gehe zu 3.*
6. *Falls $h = 1$ ist, gebe g_1 aus.*
7. *Schreibe $h = u(x^p)$ mit $u \in k(y_1, \ldots, y_m)[x]$.*

8. *Bilde* $v \in k(\sqrt[p]{y_1}, \ldots, \sqrt[p]{y_m})[x]$ *aus* u*, indem jedes* y_i *in* u *durch* $\sqrt[p]{y_i}$ *und jeder Koeffizient* $a \in k$ *in* u *durch* $\sqrt[p]{a} \in k$ *ersetzt wird.*
9. *Berechne* $g_2 := \mathrm{SEP}(v)$.
10. *Berechne* $g_3 := \mathrm{SEP}(g_1 g_2)$ *und gebe* g_3 *aus.*

Satz 3.1.2.
Der Algorithmus SEP *terminiert nach endlich vielen Schritten und berechnet korrekt den separablen Teil eines Polynoms.*

Beweis: Wähle eine Körpererweiterung L von $K := k(y_1, \ldots, y_m)$, die alle Wurzeln des Polynoms f enthält und schreibe

$$f = c \prod_{i=1}^{r} (x - \alpha_i)^{d_i} \cdot \prod_{i=1}^{s} (x - \beta_i)^{pe_i},$$

wobei r, s, d_i und e_i nicht-negative ganze Zahlen sind mit $0 < d_i < p$ und $e_i > 0$ und die c, α_i und β_i Elemente in L sind mit $\alpha_i \neq \alpha_j$ und $\beta_i \neq \beta_j$ für $i \neq j$. Die Ableitung von f ist somit

$$f' = \sum_{i=1}^{r} d_i \frac{f}{x - \alpha_i}$$

und der größte gemeinsame Teiler von f und f' ist

$$\mathrm{ggT}(f, f') = \prod_{i=1}^{r} (x - \alpha_i)^{d_i - 1} \cdot \prod_{i=1}^{s} (x - \beta_i)^{pe_i}.$$

Nach Schritt 2 ist das Polynom g_1 als

$$g_1 = c \prod_{i=1}^{r} (x - \alpha_i)$$

gegeben. Die Schleife mit den Schritten 3-5 endet nach endlich vielen Schritten, denn der Grad von $\tilde{h}$ ist kleiner als der Grad von h und so hat das Polynom h in Schritt 5 jeweils einen niedrigeren Grad als seine Vorgänger h. Der Fall $\tilde{h} = h$ tritt in Schritt 4 nur ein, wenn $h' = 0$ ist. Ist $h' = 0$, so folgt für das Polynom h, dass nur $h = 1$ oder $h = (\prod_{i=1}^{s} (x - \beta_i)^{e_i})^p = \prod_{i=1}^{s} (x - \beta_i)^{pe_i}$ gelten kann. Tritt der Fall $h = 1$ ein, so wird abgebrochen und g_1 ausgegeben. Andernfalls hat das Polynom h die Form

$$h = \prod_{i=1}^{s} (x - \beta_i)^{pe_i} = \prod_{i=1}^{s} (x^p - \beta_i^p)^{e_i}.$$

Im Polynom h ist die Variable x nur in der p-ten Potenz exzistent. Damit gilt für das Polynom u , dass $u = \prod_{i=1}^{s}(x-\beta_i^p)^{e_i}$ ist, und der Schritt 7 ist somit durchführbar. Falls $s = 0$ ist, ist der Algorithmus korrekt. Wenn $s > 0$ gilt, hat das Polynom v nach Schritt 8 und durch Ersetzen der β_i^p mit β_i die Darstellung

$$v = \prod_{i=1}^{s}(x-\beta_i)^{e_i}.$$

Es gilt für den Grad von v die Ungleichung $\deg(v) < \deg(h) \leq \deg(f)$. Daraus folgt, dass die rekursive Anwendung von SEP in Schritt 9 nach endlich vielen Schritten terminiert und als separablen Teil von v das Polynom

$$g_2 = \prod_{i=1}^{s}(x-\beta_i)$$

ausgibt. Das rekursive Aufrufen von SEP in Schritt 10 endet ebenso nach endlich vielen Schritten, denn es gilt $\deg(g_1g_2) = r+s < r+ps \leq \deg(f)$, d.h. mit jedem rekursiven Schritt verkleinert sich der Grad des Polynoms, für das der separable Teil zu berechnen ist. Es gilt also $g_3 = \mathrm{sep}(g_1g_2) = g_1g_2 = \mathrm{sep}(f)$. □

Bemerkung 3.1.3. *Die Umsetzung von Schritt 8 in ein Computeralgebrasystem erfolgt in der Art, dass das Polynom u nicht verändert wird und die p-te Wurzel nicht direkt berechnet wird. Stattdessen werden die Anzahl der Durchläufe dieses Schrittes gezählt und gemerkt. Nach Durchführung des Algorithmus ist ein Polynom in $k(y_1,\ldots,y_m)[x]$ bestimmt und die Anzahl der Durchläufe sei r. Hieraus kann der separable Teil des Polynoms f durch Ersetzen der y_i durch $\sqrt[p^r]{y_i}$ und der Koeffizienten $a \in k$ durch $\sqrt[p^r]{a}$ bestimmt werden.*

Beispiel 3.1.4.

1. *Es wird der separable Teil von*
   $f_2 = x_1^2x_2^4 + 2x_1^3x_2^2x_1^4 = x_1^2(x_2^2+x_1)^2 \in \mathbb{Z}/(7)(x_1)[x_2]$ *berechnet.*
   In Schritt 1 ist $h = \mathrm{ggT}_{\mathbb{Z}/(7)(x_1)[x_2]}(f_2, f_2') = x_2^2+x_1$. *Das Polynom* g_1 *ist damit* $x_1^2(x_2^2+x_1)$. *Nach wiederholter Anwendung der Schritte 3 bis 6 wird* $g_1 = x_1^2(x_2^2+x_1)$ *als das separable Polynom von* f_2 *ausgegeben. Der separable Teil von*
   $f_3 = x_3^4 + x_1x_3^2 + x_1^2 \in \mathbb{Z}/(7)(x_1)[x_3]$ *ist* $\mathrm{sep}(f_3) = x_3^2 + x_1$.

2. *Sei ein Polynom* $f = x^5 - y \in \mathbb{Z}/(5)(y)[x]$ *gegeben. Der separable Teil von f berechnet sich wie folgt. Nach Schritt 1 ist* $h = f$ *und somit*

folgt $g_1 = f/h = 1$. Das Polynom $\tilde{h}$ ist nach Schritt 4 gleich h. Nach Schritt 7 wird mit $h = x^5 - y$ das Polynom $u = x - y \in \mathbb{Z}/(5)(y)[x]$ geformt. Mit Hilfe von u folgt nach Schritt 8, dass $v = x - \sqrt[p]{y}$ gilt. Nun wird der Algorithmus SEP *zur Berechnung des separablen Teiles von v angewandt. Hier gilt nach Durchführung der Schritte 1-6, dass* $\mathrm{SEP}(v) = x - \sqrt[p]{y}$ *ist. Damit ist $g_2 = x - \sqrt[p]{y}$ in Schritt 9 und es ist* $\mathrm{SEP}(g_1 g_2)$ *für $g_1 g_2 = x - \sqrt[p]{y}$ zu berechnen. Das Polynom $g_3 = x - \sqrt[p]{y}$ ist der separable Teil von f.*

Das Lemma von Seidenberg stellt ein Hilfsmittel zum Beweis des nachfolgenden Algorithmus dar. Diese Proposition ermöglicht es, die Berechnung des Radikals eines null-dimensionalen Ideals auf den univariaten Fall zu reduzieren.

Proposition 3.1.5 (Seidenbergs Lemma)**.**
Sei I ein Ideal in dem Polynomring $K[x_1, \ldots, x_n]$ über einen Körper K. Wenn es für jedes $i \in \{1, \ldots, n\}$ ein separables Polynom $f_i \in I \cap K[x_i]$ gibt, dann ist I ein Radikalideal, d.h. es gilt $I = \sqrt{I}$.

Der Beweis wird von Kreuzer und Robbiano (vgl. Proposition 3.7.15, [12]) vorgelegt.

Der Algorithmus von Kemper berechnet das Radikal eines null-dimensionalen Ideals I, wie im Folgenden beschrieben wird.

Algorithmus 3.1.6 (KEMPER)**.**

Input: *Endlich viele Polynome, die ein null-dimensionales Ideal I in dem Polynomring $K[x_1, \ldots, x_n]$ über den rationalen Funktionenkörper $K = k(y_1, \ldots, y_m)$ erzeugen.*

Output: *Endlich viele Polynome in $K[x_1, \ldots, x_n]$, die das Radikal von I erzeugen.*

Beschreibung: KEMPER(I)

1. *Finde ein $f_i \in I \cap K[x_i]$ für alle $i \in \{1, \ldots, n\}$.*
2. *Berechne für jedes i den separablen Teil* $\mathrm{sep}(f_i) \in k(\sqrt[p^{r_1}]{y_1}, \ldots, \sqrt[p^{r_m}]{y_m})[x_i]$ *mit Hilfe des Algorithmus* SEP.
3. *Für jedes i schreibe* $\mathrm{sep}(f_i) = g_i(\sqrt[q]{y_1}, \ldots, \sqrt[q]{y_m}, x_i)$, *wobei $q := p^r$, $r := \max\{r_1, \ldots, r_m\}$ und $g_i \in K[z_1, \ldots, z_m, x_i]$ mit neuen Unbestimmten $z_1, \ldots, z_m$ sei.*

4. *Forme das Ideal*
   $J := IK[z_1, \ldots, z_m, x_1, \ldots, x_n] + \langle g_1, \ldots, g_n \rangle + \langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle$
   im Körper $K[z_1, \ldots, z_m, x_1, \ldots, x_n]$.
5. *Berechne das Eliminationsideal* $\tilde{J} := J \cap K[x_1, \ldots, x_n]$ *und gebe* $\tilde{J}$ *aus.*

Satz 3.1.7.
Der Algorithmus KEMPER *berechnet für ein null-dimensionales Ideal I in* $K[x_1, \ldots, x_n]$ *korrekt das Radikal.*

Dieser Satz wird wie folgt bewiesen: Die Korrektheit des Algorithmus ist dadurch zu zeigen, dass das in Schritt 5 berechnete Eliminationsideal $\tilde{J}$ dem Radikal von I entspricht. Hierzu wird der Isomorphismus
$K[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle \cong k(\sqrt[q]{y_1}, \ldots, \sqrt[q]{y_m}))$ betrachtet.
Eine kanonische Projektion sei durch die Abbildung
$K[z_1, \ldots, z_m] \to K[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle$ gegeben.
Diese induziert eine Abbildung φ :

$$\begin{array}{ccc} K[z_1, \ldots, z_m, x_1, \ldots, x_n] & \to & K[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle [x_1, \ldots, x_n] \\ g_i & \mapsto & \varphi(g_i) \end{array} .$$

Entsprechend dem Schritt 3 wird unter dem Isomorphismus

$$K[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle \cong k(\sqrt[q]{y_1}, \ldots, \sqrt[q]{y_m}))$$

das $\varphi(g_i)$ auf den separablen Teil $\mathrm{sep}(f_i)$ abgebildet.
Sei $IK[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle [x_1, \ldots, x_n]$ das von I erzeugte Ideal in $K[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle [x_1, \ldots, x_n]$ und betrachte das Ideal

$$\bar{J} := IK[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle [x_1, \ldots, x_n] + \langle \varphi(g_1), \ldots, \varphi(g_n) \rangle$$

in $K[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle [x_1, \ldots, x_n]$.
Unter Berücksichtigung des Satzes 3.1.5 der vorliegenden Arbeit ist $\bar{J}$ ein Radikalideal, d.h. es gilt demnach

$$\bar{J} = \sqrt{IK[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle [x_1, \ldots, x_n]}.$$

Desweiteren wird der kanonische Epimorphismus
$K[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle [x_1, \ldots, x_n] \to$
$\to K[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle [x_1, \ldots, x_n]/\bar{J}$ betrachtet.
Daraus ergibt sich die Komposition
$K[z_1, \ldots, z_m, x_1, \ldots, x_n] \xrightarrow{\varphi}$
$\xrightarrow{\varphi} K[z_1, \ldots, z_m]/\langle z_1^q - y_1, \ldots, z_m^q - y_m \rangle [x_1, \ldots, x_n]$

$\rightarrow K[z_1,\ldots,z_m]/\langle z_1^q - y_1,\ldots,z_m^q - y_m\rangle[x_1,\ldots,x_n]/\bar{J}$.
Der Kern dieser Komposition entspricht dem in Schritt 4 definiertem Ideal J. Denn der Kern enthält das Ideal $\bar{J}$ und das Ideal $\langle z_1^q - y_1,\ldots,z_m^q - y_m\rangle$.
An diese Komposition wird noch die Inklusionsabbildung
$K[x_1,\ldots,x_n] \rightarrow K[z_1,\ldots,z_m,x_1,\ldots,x_n]$ angefügt. Mit dieser Komposition von Abbildungen ergibt sich
$\psi : K[x_1,\ldots,x_n] \rightarrow$
$\rightarrow K[z_1,\ldots,z_m,x_1,\ldots,x_n]$
$\rightarrow K[z_1,\ldots,z_m]/\langle z_1^q - y_1,\ldots,z_m^q - y_m\rangle[x_1,\ldots,x_n]$
$\rightarrow K[z_1,\ldots,z_m]/\langle z_1^q - y_1,\ldots,z_m^q - y_m\rangle[x_1,\ldots,x_n]/\bar{J}$.
Der Kern von ψ ist $J \cap K[x_1,\ldots,x_n]$ und dies ist nach Definition das Ideal $\tilde{J}$.
Eine mögliche Interpretation der Abbildung ψ ist, dass zuerst eine Inklusionsabbildung
von $K[x_1,\ldots,x_n]$ nach $K[z_1,\ldots,z_m]/\langle z_1^q - y_1,\ldots,z_m^q - y_m\rangle[x_1,\ldots,x_n]$
erfolgt und danach eine Reduktion modulo $\bar{J}$ nach
$K[z_1,\ldots,z_m]/\langle z_1^q - y_1,\ldots,z_m^q - y_m\rangle[x_1,\ldots,x_n]/\bar{J}$ anschließt.
Der Kern von ψ kann hier als $\bar{J} \cap K[x_1,\ldots,x_n]$ geschrieben werden. Es ist somit $\tilde{J} = \bar{J} \cap K[x_1,\ldots,x_n]$.
Demnach gilt nun

$$\begin{aligned}
\tilde{J} &= J \cap K[x_1,\ldots,x_n] \\
&= \bar{J} \cap K[x_1,\ldots,x_n] \\
&= \sqrt{IK[z_1,\ldots,z_m]/\langle z_1^q - y_1,\ldots,z_m^q - y_m\rangle[x_1,\ldots,x_n]} \cap K[x_1,\ldots,x_n] \\
&= \sqrt{IK[z_1,\ldots,z_m]/\langle z_1^q - y_1,\ldots,z_m^q - y_m\rangle[x_1,\ldots,x_n] \cap K[x_1,\ldots,x_n]} \\
&= \sqrt{I}.
\end{aligned}$$

Diese letzte Gleichung gilt als erfüllt, denn sie ist nach Kreuzer und Robbiano (vgl. Proposition 2.6.12, [12]) für eine Körpererweiterung
$K \subseteq K[z_1,\ldots,z_m]/\langle z_1^q - y_1,\ldots,z_m^q - y_m\rangle$ richtig. □

Bemerkung 3.1.8.
Dieser Algorithmus steht in Beziehung zum Algorithmus in Kreuzer und Robbiano (vgl. Korollar 3.7.16, [12]). Dort erfolgt die Radikalberechnung eines null-dimensionalen Ideals I in $K[x_1,\ldots,x_n]$ mit einem Körper K, der die Charakteristik 0 hat oder ein vollkommener Körper der Charakteristik $p > 0$ ist, indem die quadratfreien Teile der Polynome anstatt der separablen Teile der Polynome berechnet werden. Die Besonderheit ist, dass im hier vorliegenden Fall der quadratfreie Teil und der separable Teil eines Polynoms nicht übereinstimmen, da die Grundkörper der Algorithmen verschieden sind.

Bemerkung 3.1.9.
In Schritt 1 des Algorithmus KEMPER *wird für alle i ein Polynom f_i über die Berechnung des Eliminationsideals $I \cap K[x_i]$ ermittelt. Dies erfolgt mit Satz 2.4.4 und in Schritt 5 wird der Satz ebenso zur Berechnung des Eliminationsideals $J \cap K[x_1, \ldots, x_n]$ herangezogen.*

Beispiel 3.1.10.
Nach Beispiel 2.5.6 ist das Ideal $I = \langle x_1^2 + 2x_1x_2x_3 + x_3^4, x_2x_3 - x_3^2\rangle$ in $\mathbb{Z}/(7)(x_1)[x_2, x_3]$ null-dimensional. Der Algorithmus KEMPER *berechnet dazu wie folgt das Radikal.*
Es wird zu Beginn eine σ-Gröbner-Basis von I in $\mathbb{Z}/(7)[x_1, x_2, x_3]$ berechnet und die Menge $\{x_2x_3 - x_3^2, x_3^4 + 2x_1x_2x_3 + x_1^2, -x_1^2x_2 + x_1^2x_3\}$ ist eine σ-Gröbner-Basis von I.
In Schritt 1 ist damit $f_2 = -x_1^2x_2^4 - 2x_1^3x_2^2 - x_1^4 \in \mathbb{Z}/(7)(x_1)[x_2]$ und $f_3 = 3x_3^4 - x_1x_3^2 + 3x_1^2 \in \mathbb{Z}/(7)(x_1)[x_3]$.
Nach Beispiel 3.1.4 sind deren separablen Teile gegeben als
$\mathrm{sep}(f_2) = x_1^2(x_2^2+x_1)$ *und* $\mathrm{sep}(f_3) = x_3^2+x_1$. *Die separablen Teile werden umgeschrieben zu $g_2 = y(x_2^2+y)$ und $g_3 = x_3^2+y$ als Polynome in $\mathbb{Z}/(7)[y, x_2, x_3]$. Mit Schritt 4 wird das Ideal $J = I\mathbb{Z}/(7)(x_1)[y, x_2, x_3] + \langle g_2, g_3\rangle + \langle y - x_1\rangle$ geformt und anschließend das Eliminationsideal $J \cap \mathbb{Z}/(7)(x_1)[y, x_2, x_3]$ als das Ideal $\tilde{J} = \langle x_2x_3 - x_3^2, x_3^2 - x_1, -x_1x_2 + x_1x_3\rangle$ berechnet.*

Im Folgenden wird auf der Grundlage des Artikels von Kemper (vgl. [11]) die Berechnung eines Radikals eines polynomialen Ideals in einem Körper mit positiver Charakteristik beschrieben.
Kemper nutzt die Strategie, das Problem auf den null-dimensionalen Fall mit Hilfe der Erweiterungs- und Kontraktionsmethode zu reduzieren.
Der entsprechende Algorithmus basiert auf dem Algorithmus in Becker und Weispfenning (vgl. Theorem 8.9, [2]).

Algorithmus 3.1.11 (RADIKALKEMPER).
Sei K ein berechenbarer und vollkommener Körper der Charakteristik $p > 0$. Es sei angenommen, dass für jedes Element in K die p-te Wurzel berechenbar sei. Sei X die endliche Menge der Variablen $x_1, \ldots, x_n$ und sei im Folgenden der Polynomring $K[x_1, \ldots, x_n]$ als $K[X]$ bezeichnet. Sei $Y = \{y_1, \ldots, y_r\}$ eine beliebige endliche Menge von Variablen. Für eine solche Menge Y sei $K(Y)$ der rationale Funktionenkörper über K.

Input: *Polynome, die ein Ideal I in $K[X]$ endlich erzeugen.*

Output: *Polynome, die $\sqrt{I}$ in $K[X]$ endlich erzeugen.*

Beschreibung: RADIKALKEMPER(I)

1. *Sei* $G = \langle 1 \rangle$.
2. *Falls* $1 \in I$, *gebe* G *aus.*
3. *Berechne eine maximal unabhängige Menge* Y *modulo* I *mit Hilfe von Algorithmus* MAXINDEPSET.
4. *Berechne das Radikal von* I *in* $K(Y)[X \setminus Y]$ *mit Hilfe des Algrithmus* KEMPER *und bezeichne es mit* Z.
5. *Berechne das Kontraktionideal* Z^c *von* Z *nach* $K[X]$.
6. *Bestimme ein Element* f *in* $K[Y]$ *mit* $I = (I + \langle f^s \rangle) \cap I^{ec}$ *für ein* $s \in \mathbb{N}$ *mit Hilfe des Algorithmus* EXTCONT.
7. *Berechne* $G =$ RADIKALKEMPER$(I + \langle f \rangle) \cap Z^c$

Satz 3.1.12.
Der Algorithmus RADIKALKEMPER *berechnet das Radikal eines endlich erzeugten Ideals in* $K[X]$ *korrekt und terminiert nach endlich vielen Schritten.*

Dieser Satz wird wie folgt bewiesen: Die Terminierung des Algorithmus ist trivial für den Fall, dass $1 \in I$ gilt. Ansonsten ist Y nach Schritt 3 eine maximal unabhängige Menge modulo I. Für Y gilt gemäß Definition, dass $I \cap K[Y] = \langle 0 \rangle$ ist. Somit ist die Inklusion $I \subseteq I + \langle f \rangle$ mit dem Element f aus Schritt 6 echt. Durch die rekursive Anwendung von RADIKALKEMPER in Schritt 7 ist diese Inklusion eine streng aufsteigende Folge von Idealen. Da der Polynomring $K[X]$ Noethersch ist, ist die streng aufsteigende Folge von Idealen endlich und somit terminiert der Algorithmus nach endlich vielen Schritten.
Die Korrektheit des Algorithmus ist trivial gegeben, wenn $1 \in I$ gilt.
Gilt dies nicht, ist zu zeigen, dass in Schritt 7 in der Tat das Radikal von I berechnet wird. Die Erzeuger vom Ideal I in $K[X]$ werden dazu in $K(Y)[X \setminus Y]$ betrachtet. Diese Menge erzeugt das Erweiterungsideal I^e von I nach $K(Y)[X \setminus Y]$ nach Satz 2.6.2 der vorliegenden Arbeit.
Das Erweiterungsideal I^e in $K(Y)[X \setminus Y]$ ist nach Satz 2.6.3 null-dimensional und mit Hilfe von Satz 3.1.7 entspricht das Ideal Z dem Radikal von I^e. Es gilt somit $Z^c = (\sqrt{I^e})^c$. Wie in 2.6.11 gilt folglich $(\sqrt{I^e})^c = \sqrt{I^{ec}}$. Da in Schritt 6 ein Element f mit dem Algorithmus EXTCONT berechnet worden ist, gilt in der Tat nach Satz 2.6.10 $I = (I + \langle f^s \rangle) \cap I^{ec}$ für ein $s \in \mathbb{N}$. Somit ist

$$\sqrt{I} = \sqrt{(I + \langle f^s \rangle) \cap I^{ec}} = \sqrt{I + \langle f^s \rangle} \cap \sqrt{I^{ec}} = \sqrt{I + \langle f \rangle} \cap Z^c.$$

Die zweite Gleichung ergibt sich aus Satz 2.1.6. Die Korrektheit des Algorithmus RADIKALKEMPER ist damit gezeigt. □

Bemerkung 3.1.13.
In Schritt 3 ist es möglich, dass die maximal unabhängige Menge Y modulo I die leere Menge ist. In diesem Fall kann die Ausgabe des Ideals Z nach Schritt 4 erfolgen, da die weiteren Schritte keinen Einfluss auf das Ideal Z haben.

Zum Abschluß der Methode von Kemper zur Berechnung des Radikals eines Ideals wird ein Beispiel vorgestellt.

Beispiel 3.1.14.
Sei $I = \langle x_1^2 + 2x_1x_2x_3 + x_3^4, x_2x_3 - x_3^2\rangle$ *das Ideal in* $\mathbb{Z}/(7)[x_1, x_2, x_3]$ *aus Beispiel 2.5.6. Es wird* $G := \langle 1\rangle$ *definiert und es gilt* $1 \notin I$. *Nach Beispiel 2.5.6 ist* $\{x_2\}$ *eine maximal unabhängige Menge modulo* I.
Das Radikal des null-dimensionalen Ideals ist das Ideal $Z = \langle x_2x_3 - x_3^2, x_3^4 + 2x_1x_2x_3 + x_1^2, -x_1x_2^2 - x_1^2, -x_1x_3^3 - x_1^2x_3, x_1^2x_3^2 + x_1^3, -x_1^2x_2 + x_1^2x_3\rangle$. *Das Kontraktionsideal* Z^c *von* Z *nach* $\mathbb{Z}/(7)[x_1, x_2, x_3]$ *ist gegeben durch das Ideal* $\langle x_3^2 + x_1, x_1x_2 - x_1x_3, x_2x_3 + x_1\rangle$. *Der Algorithmus* EXTCONT *liefert das Element* $f := -x_2 \in \mathbb{Z}/(7)(x_2)$ *für* $I + \langle f^s\rangle) \cap I^{ec}$ *mit einem* $s \in \mathbb{N}$. *Es gilt* $1 \in I + \langle f\rangle$ *und das bedeutet* RADIKALKEMPER$(I + \langle f\rangle) = I + \langle f\rangle$. *Das Ideal* $(I + \langle f\rangle) \cap Z^c$ *wird als* G *definiert und ist das Radikal von* I.

3.2 Der Algorithmus von Matsumoto

In diesem Abschnitt wird die Radikalberechnung nach Matsumoto veranschaulicht. Im Artikel von Matsumoto wird die Berechnung eines Radikals von einem Ideal in einem Polynomring mit n Variaben über einen vollkommenen Körper der Charakteristik $p > 0$ beschrieben.
Zunächst werden der Endomorphismus φ sowie Eliminationsideale betrachtet, die für die Anwendung des Algorithmus RADIKALMATSUMOTO notwendig sind. Anschließend wird der Algorithmus RADIKALMATSUMOTO vorgestellt.

Definition 3.2.1.
Sei K *ein Körper der Charakteristik* $p > 0$ *und* $K[X] := K[x_1, \ldots, x_n]$ *ein Polynomring. Sei* q *eine Potenz von* p.
Dann heißt die Abbildung $\varphi_F\colon K[X] \to K[X]$ *definiert durch* $\varphi_F(f) = f^p$ *der* ***Frobenius-Endomorphismus*** *von* $K[X]$.
Die Abbildung $\varphi : K[X] \to K[X]$ *definiert durch* $\varphi(f) = f^q$ *mit* $q = p^e$ *heißt die* **e-*te Potenz des Frobenius-Endomorphismus*** *von* $K[X]$.

Die Abbildung φ ist in der Tat ein Ringhomomorphismus von $K[X]$ nach $K[X]$, denn es gilt für alle $a, b \in K[X]$: $(a+b)^q = a^q + b^q$ und $(ab)^q = a^q b^q$.

Satz 3.2.2.
Sei die Urbildmenge von I unter φ durch das Ideal
$\varphi^{-1}(I) := \{f \in K[X] | \varphi(f) \in I\}$ gegeben.
So gilt $I \subseteq \varphi^{-1}(I) \subseteq \sqrt{I}$.

Der Beweis ergibt sich wie folgt: Sei $f \in I$. Da $\varphi(f) = f^q \in I$ gilt, ist $f \in \varphi^{-1}(I)$ und somit ist die Inklusion $I \subseteq \varphi^{-1}(I)$ gezeigt. Für die Polynome $f \in \varphi^{-1}(I)$ gilt nach Definition $f^q \in I$ und daraus folgt, dass $f \in \sqrt{I}$ gegeben ist. $\square$

Bemerkung 3.2.3.
Wenn $I \neq \sqrt{I}$ gilt, ist I sogar echte Teilmenge von $\varphi^{-1}(I)$ und es gibt die folgende Kette von echten Inklusionen von Idealen:

$$I \subset \varphi^{-1}(I) \subset \varphi^{-2}(I) \subset \ldots.$$

Da $K[X]$ Noethersch ist, existiert ein j, so dass gilt:

$$\varphi^{-j+1}(I) \subset \varphi^{-j}(I) = \sqrt{I}.$$

Desweiteren werden Definitionen betrachtet, die zur Berechnung von $\varphi^{-1}(I)$ hilfreich sind.

Definition 3.2.4.
Sei K ein Körper der Charakteristik $p > 0$ und $K[X] := K[x_1, \ldots, x_n]$ ein Polynomring. Sei q eine Potenz von p.
Der Ringhomomorphismus φ_c von $K[X]$ nach $K[X]$ ist als Abbildung

$$\varphi_c : \begin{array}{ccc} K[X] & \to & K[X] \\ \sum\limits_{m_1,\ldots,m_n \in \mathbb{N}} a_{m_1 \cdots m_n} x_1^{m_1} \cdots x_n^{m_n} & \mapsto & \sum\limits_{m_1,\ldots,m_n \in \mathbb{N}} a^q_{m_1 \cdots m_n} x_1^{m_1} \cdots x_n^{m_n} \end{array}$$

definiert und
der Ringhomomorphismus φ_v von $K[X]$ nach $K[X]$ ist als Abbildung

$$\varphi_v : \begin{array}{ccc} K[X] & \to & K[X] \\ f(x_1, \ldots, x_n) & \mapsto & f(x_1^q, \ldots, x_n^q) \end{array}$$

definiert.

Bemerkung 3.2.5.
Die Abbildungen φ, φ_c und φ_v sind über $\varphi = \varphi_c \circ \varphi_v = \varphi_v \circ \varphi_c$ miteinander verbunden.

Für die Urbildmenge von φ gilt somit folgendes:

$$\varphi^{-1}(I) = \varphi_c^{-1}(\varphi_v^{-1}(I)) = \varphi_v^{-1}(\varphi_c^{-1}(I).$$

Das Ideal $\varphi_v^{-1}(I)$ kann mit dem Buchberger Algorithmus oder einer Umformung der Gröbner-Basis berechnet werden. Diese Möglichkeit der Umformung wird als nächstes betrachtet.

Proposition 3.2.6.
Sei das Ideal $J := IK[x_1, \ldots, x_n, y_1, \ldots, y_n] + (y_1 - x_1^q, \ldots, y_n - x_n^q)$ in $K[x_1, \ldots, x_n, y_1, \ldots, y_n]$ gegeben und es sei $\tilde{J} = J \cap K[y_1, \ldots, y_n]$ das Ideal in $K[y_1, \ldots, y_n]$. Dann entsteht die Urbildmenge $\varphi_v^{-1}(I)$ aus $\tilde{J}$, indem die y_i durch x_i für alle $i \in \{1, \ldots, n\}$ ersetzt werden.

Den Beweis liefern Kreuzer und Robbiano (vgl. Proposition 3.6.2, [12]).

Für das Eliminationsideal kann mit Hilfe des Buchberger Algorithmus eine Gröbner-Basis von J bzgl. einer Eliminationsordnung für $\{x_1, \ldots, x_n\}$ berechnet werden. Eine andere Möglichkeit eine Gröbner-Basis von J zu erhalten, basiert auf dem Algorithmus in Faugère et al. ([7]), der die Gröbner-Basis eines Ideals bzgl. einer gegebenen Ordnung in eine Gröbner-Basis des Ideals bzgl. einer gewünschten Ordnung umformt. Der folgende Satz erlaubt es, eine Gröbner-Basis für J bzgl. einer Eliminationsordnung für $\{y_1, \ldots, y_n\}$ aus einer Gröbner-Basis von I zu ermitteln.

Satz 3.2.7.
Sei I ein Ideal in $K[X]$.
Sei σ_X eine Monomordnung auf $\mathbb{T}(x_1, \ldots, x_n)$ und σ_Y eine auf $\mathbb{T}(y_1, \ldots, y_n)$.
So sei die Monomordnung σ_{XY} mit σ_X und σ_Y festgelegt durch

$$x_1^{a_1} \cdots x_n^{a_n} y_1^{b_1} \cdots y_n^{b_n} \leq_{\sigma_{XY}} x_1^{c_1} \cdots x_n^{c_n} y_1^{d_1} \cdots y_n^{d_n}$$

$$\Leftrightarrow \begin{cases} y_1^{b_1} \cdots y_n^{b_n} \leq_{\sigma_Y} y_1^{d_1} \cdots y_n^{d_n} \\ \quad oder \\ y_1^{b_1} \cdots y_n^{b_n} = y_1^{d_1} \cdots y_n^{d_n} \quad und \quad x_1^{a_1} \cdots x_n^{a_n} \leq_{\sigma_X} x_1^{c_1} \cdots x_n^{c_n} \end{cases}$$

für alle $a_i, b_i, c_i, d_i \in \mathbb{N}$ mit $i \in \{1, \ldots, n\}$.
Die Ordnung σ_{XY} ist eine Eliminationsordnung für $\{y_1, \ldots, y_n\}$.
Sei G nun eine σ_X-Gröbnerbasis von I.
Dann ist $G \cup \{y_1 - x_1^q, \ldots, y_n - x_n^q\}$ eine σ_{XY}-Gröbnerbasis von $J = IK[x_1, \ldots, x_n, y_1, \ldots, y_n] + (y_1 - x_1^q, \ldots, y_n - x_n^q)$.

Beweis: Die Behauptung, dass σ_{XY} eine Eliminationsordnung für $\{y_1, \ldots, y_n\}$ ist, wird zunächst bewiesen. Nach Definition 2.4.1 ist zu zeigen,

dass ein Polynom $f \in K[x_1, \ldots, x_n, y_1, \ldots, y_n] \setminus \{0\}$, dessen Leitterm bzgl. σ_{XY} ein Element von $K[y_1, \ldots, y_n]$ ist, ebenso ein Polynom in $K[y_1, \ldots, y_n]$ ist. Es sei $\mathrm{LT}_{\sigma_{XY}}(f) = x_1^{\alpha_1} \cdots x_n^{\alpha_n} y_1^{\alpha_{n+1}} \cdots y_n^{\alpha_{2n}}$ der Leitterm von f bzgl. σ_{XY} und $t = x_1^{\beta_1} \cdots x_n^{\beta_n} y_1^{\beta_{n+1}} \cdots y_n^{\beta_{2n}}$ sei ein Term im Träger von f. Nach der Definition von σ_{XY} gilt $0 = \alpha_1 + \cdots + \alpha_n \geq \beta_1 + \cdots + \beta_n$. Damit ist $\beta_1 = \cdots = \beta_n = 0$ und das bedeutet, dass der Term t ein Element in $K[y_1, \ldots, y_n]$ ist. Da t einen beliebigen Term im Träger von f darstellt, ist das Polynom f, wie gewünscht, ein Element in $K[y_1, \ldots, y_n]$.
Die zweite Behauptung, das $G \cup \{y_1 - x_1^q, \ldots, y_n - x_n^q\}$ eine σ_{XY}-Gröbnerbasis von J ist, ergibt sich wie folgt. Es gilt, dass der Leitterm $\mathrm{LT}_{\sigma_{XY}}(y_i - x_i^q) = y_i$ für alle $i = \{1, \ldots, n\}$ ist. Es ist zu beachten, dass G eine σ_X-Gröbnerbasis von $I \subset K[X]$ darstellt.
Damit sind die Leitterme von $G \cup \{y_1 - x_1^q, \ldots, y_n - x_n^q\}$ bzgl. der Eliminationsordnung σ_{XY} paarweise teilerfremd. Nach Kreuzer und Robbiano (siehe Korollar 2.5.10, [12]) ist $G \cup \{y_1 - x_1^q, \ldots, y_n - x_n^q\}$ eine σ_{XY}-Gröbnerbasis von J. $\square$

Die Berechnung von $\varphi_v^{-1}(I)$ ist mit den vorherigen Sätzen ermöglicht. Die Berechnung von $\varphi_c^{-1}(I)$ wird im Folgenden gezeigt.

Ist der Körper K vollkommen, so ist die Restriktion von $\varphi_c|_K$ auf K ein Automorphismus auf K und die Abbildung φ_c^{-1} ist ebenso ein Automorphismus auf $K[x_1, \ldots, x_n]$.
Für die Urbildmenge von φ_c gilt somit
$\varphi_c^{-1}(I) = \{f \in K[x_1, \ldots, x_n] \mid \varphi_c(f) \in I\} = \{\varphi_c^{-1}(g) \mid g \in I\}$.
Falls die Menge $\{g_1, \ldots, g_r\}$ das Ideal I erzeugt, so wird $\varphi_c^{-1}(I)$ aufgrund der Injektivität von φ_c von dem Ideal der Menge $\{\varphi_c^{-1}(g_1), \ldots, \varphi_c^{-1}(g_r)\}$ erzeugt. Wenn das Urbild $\varphi_c^{-1}|_K$ der q-ten Potenz eines Elements berechnenbar ist in K, kann $\varphi_c^{-1}(I)$ berechnet werden. In einem endlichen Körper K mit p^m Elementen ist die Restriktion von φ_c auf $\mathbb{F}_{p^m}$ durch die Abbildung

$$\begin{array}{ccc} \mathbb{F}_{p^m} & \to & \mathbb{F}_{p^m} \\ a & \mapsto & a^{p^{(\log_p q) \bmod m}} \end{array}$$

gegeben. Das Urbild von $\varphi_c|_{\mathbb{F}_{p^m}}$ wird durch den Automorphismus

$$\begin{array}{ccc} \mathbb{F}_{p^m} & \to & \mathbb{F}_{p^m} \\ a & \mapsto & a^{p^{m-(\log_p q \bmod m)}} \end{array}$$

dargestellt.
Auf der Grundlage dieser Voraussetzungen berechnet der Algorithmus von Matsumoto das Radikal eines Ideals. Er wird im Folgenden dargestellt.

Algorithmus 3.2.8 (RADIKALMATSUMOTO).
Sei K ein Körper der Charakteristik $p > 0$ und q eine Potenz von p.

Input: *Eine endliche Gröbnerbasis B bezüglich beliebiger Ordnung eines Ideals I im Polynomring $K[x_1, \ldots, x_n]$.*

Output: *Eine endliche Gröbnerbasis für das Radikal $\sqrt{I}$.*

Beschreibung: RADIKALMATSUMOTO(I)

1. *Sei $B = \{b_1, \ldots, b_s\}$. Berechne $\varphi_c^{-1}(b_i)$ für jedes $i \in \{1, \ldots, s\}$.*
2. *Berechne für die Menge $\{\varphi_c^{-1}(b_1), \ldots, \varphi_c^{-1}(b_s), y_1 - x_1^q, \ldots, y_n - x_n^q\}$ eine Gröbnerbasis B' bezüglich einer Eliminationsordnung für $\{x_1, \ldots, x_n\}$.*
   Sei B'' das Eliminationsideal $B' \cap K[y_1, \ldots, y_n]$, bei dem die y_i durch die x_i für jedes $i \in \{1, \ldots, s\}$ ersetzt werden.
3. *Ist das von B'' erzeugte Ideal gleich dem von B erzeugten Ideal, so gebe B'' aus und stoppe. Ansonsten setze $B = B''$ und kehre zu Schritt 1. zurück.*

Satz 3.2.9.
Der Algorithmus RADIKALMATSUMOTO *terminiert nach endlich vielen Schritten und berechnet das Radikal eines Ideal I.*

Der Beweis wird wie folgt dargestellt: Die Terminiertheit des Algorithmus ist dadurch gegeben, dass die Schleife, die von Schritt 3 nach Schritt 1 zurückgeht, nur endlich oft durchgeführt wird. Dies folgt aus Bemerkung 3.2.3. Der Schritt 1 ist richtig, denn wie im Vorwort zum Algorithmus erläutert wurde, wird hier das Ideal $\varphi_c^{-1}(I)$ bestimmt. Für Schritt 2 ist $\varphi_v^{-1}(\varphi_c^{-1}(I))$ zu bestimmen, das aufgrund von Proposition 3.2.6 korrekt berechnet wird. Es wurde nun $\varphi^{-1}(I)$ bestimmt und dies wird nach Schritt 3 mit I verglichen. Falls dies nicht übereinstimmt, werden die vorherigen Schritte endlich oft für $\varphi^{-1}(I)$ wiederholt. Es gilt somit $I \subset \varphi^{-1}(I) \subset \varphi^{-2}(I) \subset \ldots \subset \varphi^{-j+1}(I) \subset \varphi^{-j}(I)$. Nach Bemerkung 3.2.3 terminiert diese Schleife für ein j und liefert $\varphi^{-j}(I) = \sqrt{I}$. □

Bemerkung 3.2.10.
Die Berechnung von $\varphi_c^{-1}(b_i)$ in Schritt 1. wird unter Berücksichtigung der Definition von φ_c und den Erläuterungen vor diesem Algorithmus vorgenommen. In Schritt 2. erfolgt die Berechnung einer Gröbner-Basis B' entweder mit dem Buchberger Algorithmus oder mit dem Satz 3.2.7, bei dem aus einer

vorhandenen Gröbner-Basis von $\{\varphi_c^{-1}(b_1), \ldots, \varphi_c^{-1}(b_s)\}$ *eine Gröbner-Basis von* $\{\varphi_c^{-1}(b_1), \ldots, \varphi_c^{-1}(b_s), y_1 - x_1^q, \ldots, y_n - x_n^q\}$ *bezüglich einer Eliminationsordnung für* $\{x_1, \ldots, x_n\}$ *hergeleitet werden kann.*

Die Vorstellung der Berechnung des Radikals eines Ideals nach der Methode von Matsumoto wird durch ein Beispiel abgeschlossen.

Beispiel 3.2.11.
Sei wiederum das Ideal $I = \langle x_1^2 + 2x_1x_2x_3 + x_3^4, x_1x_3 - x_3^2\rangle$ *in* $\mathbb{F}_5[x_1, x_2, x_3]$ *gewählt und als Termordnung auf* $\mathbb{T}(x_1, x_2, x_3)$ *wird* $\sigma =$ `DegRevLex` *genommen.*
Eine σ*-Gröbner-Basis von I ist* $B := \{x_1x_3 - x_3^2, x_3^4 + 2x_1x_2x_3 + x_1^2, -x_1^3 + x_3^3\}$.
In Schritt 1 wird B nicht verändert.
Eine Gröbner-Basis von $\{x_1x_3 - x_3^2, x_3^4 + 2x_1x_2x_3 + x_1^2, -x_1^3 + x_3^3, y_1^5 - x_1, y_2^5 - x_2, y_3^5 - x_3\}$ *bzgl. einer Eliminiationsordnung für* $\{x_1, x_2, x_3\}$ *ist durch* $\{x_1x_3 - x_3^2,\ -x_1^3 + x_3^3,\ x_3^4 + 2x_1x_2x_3 + x_1^2,\ -x_2^5 + y_2,\ 2x_2x_3^3 + x_3^3 + y_3,\ 2x_2^2x_3^2 + x_1^2x_2 + x_2x_3^2 - 2x_1^2 + 2x_3y_3,\ -x_3^2y_3 - 2x_2y_3 - y_3,\ -2x_2^2x_3y_3 - 2x_2x_3y_3 + 2x_3y_3 - 2y_3^2,\ -2x_2^3y_3 + 2x_2^2y_3 + x_3y_3^2 + x_2y_3 + y_3,\ 2x_1y_3 - 2x_3y_3,\ y_1 - y_3, 2y_3^3 - y_2y_3 + 2y_3, 2x_1^2x_2^4 + x_2x_3^2 - x_3^2y_2 + x_2x_3y_3 - 2x_1^2 + x_2y_3^2 + x_3y_3 + y_3^2,\ -2x_3^3y_2 - x_3^3 + 2x_2x_3y_3^2 + x_3y_3^2, 2x_2x_3^2y_2 + x_2x_3^2 - 2x_2^2y_3^2 + x_1^2y_2 - 2x_1^2 - 2x_2y_3^2 + 2y_3^2\}$ *gegeben. Dies entspricht der Menge* B' *in Schritt 2.*
Die Menge B'' *wird nach Elimination der Variablen* x_1, x_2, x_3 *ermittelt und es ist* $B'' = \{x_1 - x_3,\ 2x_3^3 - x_2x_3 + 2x_3\}$. *Da* $B \neq B''$ *in Schritt 3 gilt, wird der Schritt 1 für* $B = \{x_1 - x_3,\ 2x_3^3 - x_2x_3 + 2x_3\}$ *angewandt. In Schritt 1 wird B wiederum nicht verändert.*
In Schritt 2 ist
$B' = \{x_1 - x_3,\ 2x_3^3 - x_2x_3 + 2x_3,\ -x_2^5 + y_2,\ x_2^2x_3 + x_2x_3 - x_3 + y_3,\ -x_3^2y_3 - 2x_2y_3 - y_3,\ -2x_3^3y_3 + 2x_2^2y_3 + x_3y_3^2 + x_2y_3 + y_3,\ y_1 - y_3,\ 2y_3^3 - y_2y_3 + 2y_3,\ 2x_3y_3^2 - x_3y_2 + 2x_3,\ -2x_3^2y_2 - x_3^2 + 2x_2y_3^2 + y_3^2\}$ *eine Gröbner-Basis von* $\{x_1 - x_3,\ 2x_3^3 - x_2x_3 + 2x_3,\ y_1^5 - x_1,\ y_2^5 - x_2,\ y_3^5 - x_3\}$ *bzgl. einer Eliminiationsordnung für* $\{x_1, x_2, x_3\}$ *und die Menge* B'' *ist wie zuvor* $\{x_1 - x_3,\ 2x_3^3 - x_2x_3 + 2x_3\}$.
Es gilt nun $B = B''$ *und somit wird das Ideal* $\sqrt{I}$ *von der Menge* $\{x_1 - x_3,\ 2x_3^3 - x_2x_3 + 2x_3\}$ *erzeugt.*

3.3 Die Optimierung des Algorithmus von Krick-Logar durch Laplagne

In diesem Abschnitt wird die Berechnung eines Radikals von einem Ideal basierend auf einem Artikel von Laplagne veranschaulicht. Der in diesem Artikel angegebene Algorithmus stellt eine Optimierung des Algorithmus in Krick und Logar (vgl. [14]) dar.

Der Algorithmus von Krick und Logar wird hier nur in der optimierten Version von Laplagne dargestellt, da es sich nicht mehr um einen aktuellen Algorithmus handelt und die Weiterentwicklung aus heutiger Sicht einen höheren Stellenwert hat. Es werden lediglich die von Laplagne genutzten Voraussetzungen, die durch Krick und Logar bestimmt werden, erläutert. Anschließend wird der Algorithmus RADIKALLAPLAGNE dargestellt.

Bemerkung 3.3.1.
In Krick und Logar (vgl. [14]) wird ein Algorithmus zur Radikalberechnung vorgestellt. Als Ansatz für diesen Algorithmus wird die Darstellung des Radikals von einem Ideal I in der Form $\sqrt{I} = \sqrt{I : h} \cap \sqrt{I + \langle h \rangle}$ für ein geeignetes h genutzt. Das Ideal $\sqrt{I : h}$ läßt sich über eine Reduktion auf den null-dimensionalen Fall berechnen und das Ideal $\sqrt{I + \langle h \rangle}$ durch Induktion nach der Dimension. Bei der Berechnung von des Ideals $I + \langle h \rangle$ kommen aber redundante Komponenten vor, dies sind Komponenten, die nicht im ursprünglichen Ideal zu finden sind. Diese beeinflussen die Geschwindigkeit des Algorithmus negativ. Dies verhindert der folgende Algorithmus, der auf die Berechnung von $I + \langle h \rangle$ verzichtet und stattdessen die Saturierung $I : h^\infty$ für ein geeignetes h anwendet.

Im Folgenden sei K ein Körper der Charakteristik $p \geq 0$, $K[X] := K[x_1, \ldots, x_n]$ der Polynomring in n Variablen und $I \subset K[X]$ ein Ideal. Eine Termordnung in $\mathbb{T}(X)$, der Menge der Terme mit den Variablen $x_1, \ldots, x_n$, sei σ.
Basierend auf diesen Voraussetzungen wird nun der optimierte Algorithmus von Laplagne beschrieben.

Algorithmus 3.3.2 (RADIKALLAPLAGNE)**.**

Input: *Polynome , die ein Ideal I in $K[X]$ endlich erzeugen.*

Output: *Polynome, die das Radikal von I endlich erzeugen.*

Beschreibung: RADIKALLAPLAGNE(I)

1. *Definiere das Ideal $\tilde{I} := \langle 1 \rangle$.*
2. *Suche ein $g \in \tilde{I} \setminus \sqrt{I}$, indem durch Abschreiten der Erzeuger von $\tilde{I}$ überprüft wird, ob es ein Element von $\sqrt{I}$ ist.*
3. *Falls es kein g gibt, gehe zu Schritt 9.*
4. *Falls es ein g gibt, setze $J := I : g^\infty$.*
5. *Bestimme Y, eine maximal unabhängige Menge modulo J.*
6. *Berechne das Radikal des null-dimensionalen Ideals $JK(Y)[X \setminus Y]$.*

7. *Berechne das Kontraktionsideal von* $\sqrt{JK(Y)[X \setminus Y]}$ *nach* $K[X]$.
8. *Setze* $\tilde{I} := \tilde{I} \cap (\sqrt{JK(Y)[X \setminus Y]} \cap K[X])$ *und kehre zu Schritt 2 zurück.*
9. *Gebe* $\tilde{I}$ *aus.*

Bemerkung 3.3.3.
Die Überprüfung, ob ein Polynom g ein Element des Radikals $\sqrt{I}$ *ist, erfolgt ohne Berechnung des Radikals. Es wird überprüft, ob die Saturierung* $I : g^\infty$ *das von* 1 *erzeugte Ideal ist oder nicht. Der Abbruch der Schleife in Schritt 3 erfolgt aus dem Grund, da, nachdem es kein Polynom g gibt, die Inklusion* $\tilde{I} \subset \sqrt{I}$ *gilt. Da die Inklusion* $\sqrt{I} \subset \tilde{I}$ *immer gilt, bedeutet das, dass* $\tilde{I} = \sqrt{I}$ *ist. Die Schleife wird unterbrochen und es erfolgt die Ausgabe von* $\tilde{I}$ *als das Radikal von I. Die Ermittlung einer maximal unabhängigen Menge in Schritt 5 erfolgt mit dem Algorithmus 2.5.4 in Abschnitt 2.5 der vorliegenden Arbeit.*
In Schritt 6 kann das Radikal mit Hilfe des Algorithmus in Kreuzer und Robbiano (vgl. Korollar 3.7.16, [12]) berechnet werden, falls der Körper $K(Y)$ *die Charakteristik 0 hat, oder mit Hilfe des Algorithmus* KEMPER*, falls der Körper* $K(Y)$ *die Charakteristik* $p > 0$ *hat. Das Kontraktionsideal in Schritt 7 wird mit dem Algorithmus 2.6.4 aus Abschnitt 2.6 der vorliegenden Arbeit bestimmt.*

Satz 3.3.4.
Der Algorithmus RADIKALLAPLAGNE *berechnet das Radikal eines Ideals korrekt und terminiert.*

Dies wird wie folgt bewiesen: Zunächst wird gezeigt, dass der Algorithmus nach endlich vielen Schritten terminiert. Das Abbruchkriterium des Algorithmus liegt vor, wenn $\tilde{I} \subset \sqrt{I}$ ist. In den Schritten 2-4 wird überprüft, ob es ein Polynom g in $\tilde{I}$ gibt, das nicht in $\sqrt{I}$ liegt. Gibt es ein Polynom g, folgt nach Satz 2.7.8 die Existenz eines minimalen Primteilers, der in Schritt 8 dem Ideal $\tilde{I}$ zugefügt wird. Mit jeden Durchlauf des Algorithmus wird das Ideal $\tilde{I}$ somit mit einem minimalen Primteiler geschnitten. Nach Satz 2.7.2 besitzen Noethersche Ringe aber endlich viele minimale Primideale. Der Algorithmus terminiert daher nach endlich vielen Schritten.
Um die Korrektheit zu beweisen, wird die Durchführbarkeit des Algorithmus gezeigt. Sei $I = Q_1 \cap \cdots \cap Q_s$ eine Primärzerlegung von I und $\sqrt{Q_1}, \ldots, \sqrt{Q_s}$ die zu den Primäridealen gehörigen Primideale. Es wird der l-te Iterationsschritt betrachtet. In diesem Fall gilt $\tilde{I} = \sqrt{Q_1} \cap \cdots \cap \sqrt{Q_l'}$ mit $l' \geq l$. Ist $l' < s$, dann gibt es die Inklusion $\sqrt{I} \subset \tilde{I}$. Demach kann ein Polynom

$g \in \tilde{I} \setminus \sqrt{I}$ wie in Schritt 2 bestimmt werden. Nach Satz 2.7.8 ist ein minimaler Primteiler von I vorhanden, der in Schritt 7 berechnet wird. Mit jedem Iterationsschritt wird somit das Ideal $\tilde{I}$ um ein minimalen Primteiler von I erweitert. Diese Schritte wiederholen sich solange bis $\tilde{I} = \sqrt{I}$ in Schritt 2 ist und das Radikal von I ausgegeben werden kann. □

Das folgende Beispiel wendet den Algorithmus RADIKALLAPLAGNE an.

Beispiel 3.3.5.
Das Ideal $I = \langle x_1^2 + 2x_1x_2x_3 + x_3^4, x_2x_3 - x_3^2 \rangle$ *in* $\mathbb{Z}/(7)[x_1, x_2, x_3]$ *aus Beispiel 2.5.6 wird wieder genommen.*
In Schritt 2 wird $g := 1$ *gewählt und es ist* $J := I : 1^\infty$.
Eine maximal unabhängige Menge modulo I ist nach Beispiel 2.5.6 die Menge $\{x_2\}$. *Das Radikal des null-dimensionalen Ideals* $J\mathbb{Z}/(7)(x_2)[x_1, x_3]$ *ist das Ideal* $\langle x_2x_3 - x_3^2, x_1^2x_2 - x_1^2x_3, -x_1x_2^2 - x_1^2, -x_2^3x_3 - 2x_1x_2x_3 - x_1^2 \rangle$. *Für das Kontraktionsideal von* $\sqrt{J\mathbb{Z}/(7)(x_2)[x_1, x_3]}$ *nach* $J\mathbb{Z}/(7)[x_1, x_2, x_3]$ *gilt* $\sqrt{J\mathbb{Z}/(7)(x_2)[x_1, x_3]}^c = \langle x_3^2 + x_1, x_1x_2 - x_1x_3, x_2x_3 + x_1 \rangle$.
Es gilt damit $\tilde{I} = \tilde{I} \cap J^c = \langle x_3^2 + x_1, x_1x_2 - x_1x_3, x_2x_3 + x_1 \rangle$.
Der Algorithmus terminiert hier, denn es gibt kein $g \in \tilde{I}$, *das nicht in* $\sqrt{I}$ *ist. Für das Radikal von I gilt daher* $\sqrt{I} = \langle x_3^2 + x_1, x_1x_2 - x_1x_3, x_2x_3 + x_1 \rangle$.

3.4 Vergleich der Algorithmen

In diesem Abschnitt werden die Gemeinsamkeiten und die Unterschiede der Algorithmen RADIKALKEMPER, RADIKALMATSUMOTO und RADIKALLAPLAGNE dargestellt.
Jeder dieser Algorithmen hat als Zielsetzung die Berechnung des Radikals eines Ideals.
Der Grundkörper K des Ideals im Artikel von Laplagne kann die Charakteristik 0 oder $p > 0$ haben, während die beiden anderen Algorithmen für den Fall gelten, dass die Charakteristik des Grundkörpers K positiv ist. Der Input der drei Algorithmen ist in allen Fällen eine Menge von Polynomen, die ein Ideal I in $K[x_1, \ldots, x_n]$ endlich erzeugen.
Der Algorithmus RADIKALMATSUMOTO setzt zusätzlich voraus, dass es sich um eine Gröbner-Basis des Ideals I bzgl. einer beliebigen Ordnung handeln muss.
Die Outputs der drei Algorithmen gleichen sich in der Art, dass jeder Algorithmus eine Menge von Polynomen ausgibt, die das Radikal des Ideals I endlich erzeugen. Der Output des Algorithmus RADIKALMATSUMOTO hat außerdem die Eigenschaft eine Gröbner-Basis zu sein.
Bei der Betrachtung der Arbeitsschritte werden folgende Unterschiede deut-

lich: Die beiden Algorithmen RADIKALKEMPER und
RADIKALLAPLAGNE nutzen zur Radikalberechnung im Gegensatz zum Algorithmus RADIKALMATSUMOTO die Reduktion von höher-dimensionalen Idealen in null-dimensionale Ideale.
Der Algorithmus von Kemper berechnet mit Hilfe einer Abfolge von 7 Schritten das Radikal von einem Ideal, wenn es sich um einen null-dimensionalen Fall handelt. Bei einem höher-dimensionalen Fall müssen mehrere Durchläufe erfolgen. Im Algorithmus nach Kemper wird in Schritt 6 ein Polynom f gesucht, um das Ideal $\sqrt{I}$ aufzuspalten in das Ideal $\sqrt{I + \langle f \rangle} \cap \sqrt{I : f}$. Hier erfolgt zuvor die Berechnung einer maximal unabhängigen Menge modulo I, die eines Radikals eines null-dimensionalen Ideals und die eines Kontraktionsideals, das dem Ideal $\sqrt{I : f}$ entspricht. Es folgt die Berechnung des Ideals $\sqrt{I + \langle f \rangle}$ mit dem Algorithmus nach Kemper. Nach der Bestimmung von f treten bei der weiteren Rechnung mit dem Ideal $I + \langle f \rangle$ redundante Komponenten auf, die nicht im Radikal von I liegen. Dies verlangsamt die Durchführung des Algorithmus.
Der Algorithmus nach Laplagne berechnet mit Hilfe einer Abfolge von 9 Schritten das Radikal von einem Ideal, wenn es sich um einen null-dimensionalen Fall handelt. Bei einem höher-dimensionalen Fall müssen mehrere Durchläufe erfolgen. In Schritt 2 wird ein Polynom g berechnet, mit dem das Ideal $J = I : g^{\infty}$ konstruiert wird. Daran schließt an die Berechnung einer maximal unabhängigen Menge modulo J, die Berechnung des Radikals eines null-dimensionalen Ideals und die eines Kontraktionsideals. Das in Schritt 7 berechnete Ideal ist ein minimaler Primteiler von I. Der Durchschnitt dieses minimalen Primteilers und $\tilde{I}$ werden als das neue Ideal $\tilde{I}$ definiert. Mit diesem neuen Ideal wird ein weiterer Durchlauf des Algorthmus gestartet. Mit jeder Iteration wird somit ein minimaler Primteiler bestimmt. Im Algorithmus nach Laplagne gibt es somit keine Redundanzen.
Der Algorithmus RADIKALMATSUMOTO verfolgt dagegen einen anderen Weg und bestimmt das Radikal von I über die Berechnung der Urbildmenge $\varphi^{-1}(I)$. Dazu bedient er sich einer Abfolge von nur insgesamt 3 Arbeitsschritten. Bei der Durchführung der Schritte 1-3 wird eine Gröbner-Basis in Schritt 2 berechnet. Ergeben sich bei Schritt 3 zwei ungleiche Mengen, beginnt der Algorithmus von vorn. Es erfolgen so viele Durchläufe bis in Schritt 3 zwei gleiche Mengen erzeugt werden können. Nach endlich vielen Schritten gilt für das Radikal $\sqrt{I} = \varphi^{-j}(I)$ für ein $j > 0$.
Das mögliche Auftreten von redundanten Komponenten beim Algorithmus von Kemper kann die Berechnung von Beispielen nach erfolgter Implementation im Hinblick auf den Erfolg der Berechnung und der Berechnungsdauer negativ beeinflussen. Da diese Redundanzen bei dem Algorithmus von Laplagne nicht möglich sind, ist zu vermuten, dass diese Methode in der Regel

ein Ergebnis für die Berechnung eines Radikals von einem Ideal liefern kann. Die Methode von Matsumoto benötigt nur drei Schritte zur Berechnung des Radikals von einem Ideal. Bei der Berechnung von Beispielen könnte die Höhe der Charakteristik Auswirkungen auf den Berechnungserfolg und -dauer haben. Dies soll im nächsten Kapitel, in dem die drei Algorithmen implementiert werden und berechnete Beispiele analysiert werden, überprüft werden.

4 Implementation

In diesem Kapitel wird die Implementierung der drei im vorhergehenden Kapitel beschriebenen Algorithmen von Kemper, Matsumoto und die Optimierung des Algorithmus von Krick und Logar nach Laplagne in das Computeralgebraprogramm CoCoA vorgestellt. Dazu werden die einzelnen Programmkompenenten in ihrer Unterschiedlichkeit dargestellt. Es wird dabei herausgestellt, dass die Implementation mit den Algorithmen übereinstimmt. Anschließend werden Berechnungsbeispiele der Algorithmen in einer Tabelle veranschaulicht und die Timings ausgewertet.

4.1 Beschreibung der Implementierung

Neben den Algorithmen von Kemper, Matsumoto und Laplagne werden die Algorithmen SEP und MAXINDEPSET aufgeführt, da sie als Grundlagen für die Berechnung des Radikals eines Ideals bei den Algorithmen benötigt werden.

4.1.1 Sep

Die folgende Funktion implementiert den Algorithmus SEP. Der Input besteht aus drei Werten. Die erste Angabe gibt das Polynom an, dessen separabler Teil berechnet werden soll. Die zweite Angabe benennt die eine Variable des Polynoms und der letzte Wert stellt die Anzahl der bisherigen Durchläufe fest.
Bei der Umsetzung des Algorithmus stellte sich als Problem heraus, dass die Polynome, deren separabler Teil zu berechnen ist, Elemente von $K[x]$ sind, wobei der Körper K der rationale Funktionenkörper $k(y_1, \ldots, y_m)$ ist.
Da rationale Funktionenkörper nicht direkt eingegeben werden können, wird im Folgenden im Ring $k[y_1, \ldots, y_m, x]$ gerechnet. Für den nach Schritt 1 zu berechnenden größten gemeinsamen Teiler in $k(y_1, \ldots, y_m)[x]$ bedeutet dies, dass einige Fälle gesondert zu betrachten sind. Hierzu wird überprüft, ob der größte gemeinsame Teiler in $k[y_1, \ldots, y_m, x]$ noch Elemente mit $y_1, \ldots, y_m$ aber keine mit x enthält und entsprechend angepasst. Zusätzlich wird noch nach dem größten gemeinsamen Teiler der Koeffizienten in $k(y_1, \ldots, y_m)[x]$ gesucht. Dieser ist aus den Koeffizienten herauszuteilen.
Diese Möglichkeit ergibt sich durch den folgenden Satz.

Satz 4.1.1.
Für $f, g \in k[y_1, \ldots, y_m, x]$ *gilt:*
$\mathrm{ggT}_{k(y_1,\ldots,y_m)[x]}(f,g) = \frac{\mathrm{ggT}_{k[y_1,\ldots,y_m,x]}(f,g)}{\mathrm{LC}_{x_i}(k[y_1,\ldots,y_m,x])}$, *wobei* $\mathrm{LC}_x(k[y_1, \ldots, y_m, x])$ *der Leit-*

koeffizient vom $\mathrm{ggT}_{k(y_1,\ldots,y_m)[x]}(f,g)$ *in* $k(y_1,\ldots,y_m)[x]$ *ist.*

Dies wird nachfolgend bewiesen:
Sei $h := \mathrm{ggT}_{k[y_1,\ldots,y_m,x]}(f,g) \in k[y_1,\ldots,y_m,x]$.
Nach Kreuzer und Robbiano (vgl. Theorem 1.2.13, [12]) hat das Polynom h eine Faktorisierung in irreduzible Faktoren, d.h. es gilt: $h = c \cdot h_1 \cdots h_s$, wobei c eine Einheit ist und die h_j irreduzible Polynome in $k[y_1,\ldots,y_m,x]$ sind.
Offenbar gilt: $h \in k(y_1,\ldots,y_m)[x]$.
Ohne Einschränkung seien die Faktoren $h_1,\ldots,h_t$ diejenigen, die nur von $y_1,\ldots,y_m$ abhängen, und die $h_{t+1},\ldots,h_s$ die restlichen Faktoren.
Da die Faktoren $h_{t+1},\ldots,h_s$ Teiler von f und g in $k(y_1,\ldots,y_m)[x]$ sind, ist das Produkt $h_{t+1}\cdots h_s$ ein gemeinsamer Teiler von f und g in $k(y_1,\ldots,y_m)[x]$.
Das Produkt $h_{t+1}\cdots h_s$ ist sogar der größte gemeinsame Teiler $\mathrm{ggT}_{k(y_1,\ldots,y_m)[x]}(f,g)$ in $k(y_1,\ldots,y_m)[x]$. In $k(y_1,\ldots,y_m)[x]$ gilt, dass h und $h_1 \cdot \ldots \cdot h_t$ assoziiert zueinander sind.
Sei $\mathrm{LC}_x(h)$ der Leitkoeffizient von h in $k(y_1,\ldots,y_m)[x]$.
Der größte gemeinsame Teiler von f und g in $k(y_1,\ldots,y_m)[x]$ ist somit $\mathrm{ggT}_{k(y_1,\ldots,y_m)[x]}(f,g) = h_{t+1}\cdots h_s = \frac{h}{\mathrm{LC}_{x_i}(h)}$. □

Seine Anwendung findet der Satz bei der Implementation des Algorithmus SEP in den Schritten 1 und 3, in denen der größte gemeinsame Teiler von f und f' in $k(y_1,\ldots,y_m)[x]$ zu berechnen ist.

Dies wird durch das folgende Beispiel veranschaulicht.

Beispiel 4.1.2.
Sei $P = K[t,x]$ *und* $Q = K(t)[x]$. *Für* $f = t^2x$ *und* $g = tx^3$ *ist* $\mathrm{ggT}_P(f,g) = tx$, $\mathrm{LC}_x(\mathrm{ggT}_P(f,g)) = t$ *und* $\mathrm{ggT}_Q(f,g) = x$.

In der Funktion werden diese Überlegungen zunächst auf die Berechnung des größten gemeinsamen Teilers der Polynome `F` und `DerF` angewandt. Die erste `If`-Schleife modifiziert die Ableitung des Polynoms für den Fall, dass die Ableitung nur ein Polynom mit `t` ist und keine Variable `X` enthält. Die nächste `If`-Schleife setzt sich damit auseinander, dass einmal die Ableitung gleich 0 ist, und definiert dadurch zunächst den größten gemeinsamen Teiler. Danach folgt eine Anpassung des größten gemeinsamen Teilers, falls der Teiler nur aus einem Term besteht, der keine `X`-Variable enthält. Anschließend wird nach einem gemeinsamen Teiler der Koeffizienten dieses Polynoms gesucht und, wenn es einen gibt, wird er herausgekürzt.

```
Define Sep(F,X,Ri)
```

```
P:=Characteristic();
M:=Len(t);
N:=Len(x);
DerF:=Der(F,X);
If DerF<>0 And Len(Coefficients(DerF,X))=1 Then
  DerF:=X^0;
EndIf;
If DerF =0  Then
  H:=F;
  Else H:=GCD(F,DerF);
EndIf;
CoeffH:=Coefficients(H,X);
If Len(CoeffH)=1 And GCD(H,X)<>X Then
  H:=X^0;
  CoeffH:=Coefficients(H,X);
EndIf;
If CoeffH[1] <> 1 Then
  Bt:=[B-1 |B In 1..Len(CoeffH)];
  Reverse(Bt);
  Xt:=[X^B | B In Bt ] ;
  GCDh:=GCD( CoeffH );
  For A:=1 To M Do
    While GCD(t[A],GCDh ) = t[A] Do
      For B:=1 To Len(CoeffH) Do
        CoeffH[B]:=CoeffH[B]/t[A];
      EndFor;
      GCDh:=GCD( CoeffH );
    EndWhile;
  EndFor;
  H:= ScalarProduct(CoeffH,Xt);
EndIf;
```

Die Berechnung von $\tilde{h}$ aus Schritt 3 erfolgt auf dieselbe Weise wie die von h.

```
G1:=F/H;
DerH:=Der(H,X);
If DerH =0  Then
  Hs:=H;
  Else Hs:=H* DerH / LCM(H,DerH);
EndIf;
CoeffHs:=Coefficients(Hs,X);
If Len(CoeffHs)=1 And GCD(Hs,X)<>X Then
  Hs:=X^0;
  CoeffHs:=Coefficients(Hs,X);
```

```
EndIf;
If CoeffHs[1] <> 1 Then
  GCDhs:=GCD( CoeffHs );
  Bts:=[B-1 |B In 1..Len(CoeffHs)];
  Reverse(Bts);
  Xts:=[X^B | B In Bts ] ;
  For A:=1 To M Do
    While GCD(t[A],GCDhs ) = t[A] Do
      For B:=1 To Len(CoeffHs) Do
        CoeffHs[B]:=CoeffHs[B]/t[A];
      EndFor;
      GCDh:=GCD( CoeffHs );
    EndWhile;
  EndFor;
  Hs:= ScalarProduct(CoeffHs,Xts);
EndIf;
```

Mit der folgenden `While`-Schleife werden die Schritte 4 und 5 des Algorithmus umgesetzt. Die Berechnung von $\tilde{h}$ erfolgt hier wie im Schritt zuvor.

```
While Hs<>H Do
  H:=Hs;
  DerH:=Der(H,X);
  If DerH =0  Then
    Hs:=H;
    Else Hs:=H* DerH / LCM(H,DerH);
  EndIf;
  CoeffHs:=Coefficients(Hs,X);
  If Len(CoeffHs)=1 And GCD(Hs,X)<>X Then
     Hs:=X^0;
     CoeffHs:=Coefficients(Hs,X);
  EndIf;
  If CoeffHs[1] <> 1 Then
    GCDhs:=GCD( CoeffHs );
    Bts:=[B-1 |B In 1..Len(CoeffHs)]; Reverse(Bts);
    Xts:=[X^B | B In Bts ] ;
    For A:=1 To M Do
      While GCD(t[A],GCDhs ) = t[A] Do
        For B:=1 To Len(CoeffHs) Do
          CoeffHs[B]:=CoeffHs[B]/t[A];
        EndFor;
        GCDh:=GCD( CoeffHs );
      EndWhile;
    EndFor;
```

```
    Hs:= ScalarProduct(CoeffHs,Xts);
  EndIf;
EndWhile;
```

Es folgt das erste Abbruchkriterium für den Fall, dass das Polynom `H` gleich 1 ist.

```
If H=1 Then
  Return [G1,Ri];
EndIf;
```

Die Umformung des Polynoms h schließt nun an, indem die Exponenten der `X`-Variablen durch `P` geteilt werden.

```
SuppH:=Support(H);
U1:= NewList(Len(SuppH));
Exponentliste:=NewList(Len(SuppH));
For A:=1 To Len(SuppH) Do
  ExpA:=Log(SuppH[A]);
  Exponentliste[A]:=ExpA;
  For A1 :=1 To Len( ExpA) Do
    If Indet(A1) IsIn x Then
        If (ExpA[A1] <>0 And Mod( ExpA[A1] , P ) =0) Then
          Exponentliste[A][A1]:=ExpA[A1]/P;
        EndIf;
    EndIf;
  EndFor;
  U1[A]:=LogToTerm(Exponentliste[A]);
EndFor;
```

Die Bildung des Polynoms v erfolgt in diesem Falle durch Übernahme der Koeffizienten des Polynoms u. Hier wird ebenso der Zähler `Ri` um 1 erhöht.

```
CoeffV:=Coefficients(H,SuppH);
CoeffV1:=NewList(Len(SuppH));
For K:=1 To Len(SuppH) Do
   CoeffV1[K]:=CoeffV[K];
EndFor;
V:= ScalarProduct(CoeffV1,U1);
Ri:=Ri + 1;
```

Die Funktion `Sep` wird im Folgenden entsprechend dem Algorithmus SEP rekursiv angewandt und anschließend wird eine Liste der Form `[G3,Ri]` ausgegeben, in der `G3` die Bedeutung eines Polynoms im Ring $k(\sqrt[\mathtt{Ri}]{y_1}, \ldots, \sqrt[\mathtt{Ri}]{y_m})$ hat und `Ri` die Anzahl der bisherigen Durchläufe der Funktion `Sep` angibt.

```
  G2i:=Sep(V,X,Ri);
  G2:=G2i[1];
  Ri:=G2i[2];
  If Ri <>0 Then
    Substitiont:=[ [t[A],t[A]^P] | A In 1..M];
    G1:=Subst(G1,Substitiont);
  EndIf;
  G3i:=Sep(G1*G2,X,Ri);
  G3:=G3i[1];
  Ri:=G3i[2];
  Return [G3,Ri];
EndDefine;
```

Der Algorithmus SEP kann also wie gezeigt vollständig implementiert werden.

Im Folgenden wird die Implementation `Sep0` betrachtet, die für Polynome `F` in dem Polynomring mit nur einer Variablen `X` den separablen Teil berechnet. In diesem Fall gibt es keine weiteren Variablen wie zuvor bei `Sep` mit der `t`-Variable.
Der Input von `Sep0` besteht aus zwei Werten. Dies ist einmal das Polynom, dessen separabler Teil zu berechnen ist, und dann die Variable des Polynoms. Die Implementierung erfolgt zu Beginn wie im Algorithmus SEP dargestellt. Zunächst wird `H` bestimmt, der größte gemeinsame Teiler des Ausgangspolynoms `F` und seiner Ableitung. Es folgt die Definition des Polynoms `G1` und von `Hs`, dem größten gemeinsamen Teiler von `H` und seiner Ableitung.

```
Define Sep0(F,X)
  P:=Characteristic();
  N:=NumIndets();
  DerF:=Der(F,X);
  If DerF =0  Then
    H:=F;
    Else H:=F* DerF / LCM(F,DerF);
  EndIf;
  G1:=F/H;
  DerH:=Der(H,X);
  If DerH =0  Then
    Hs:=H;
    Else Hs:=H* DerH / LCM(H,DerH);
  EndIf;
```

Die folgende `While`-Schleife testet nach Schritt 4, ob $\tilde{h} = h$ gilt. Ist dies nicht der Fall, wird $h := \tilde{h}$ gesetzt und $\tilde{h}$ wie in Schritt 3 definiert.

```
  While Hs<>H Do
```

```
    H:=Hs;
    DerH:=Der(H,X);
    If DerH =0  Then
      Hs:=H;
      Else Hs:=H* DerH / LCM(H,DerH);
    EndIf;
  EndWhile;
```

Es folgt das erste Abbruchkriterium. Dieser Fall tritt ein, falls das Polynom `H` gleich 1 ist. Ist dies nicht der Fall, wird aus dem Polynom `H` das Polyom `V` gebildet, indem die Exponenten der `X`-Variablen durch `P` gekürzt werden und die Koeffizienten beibehalten werden.

```
  If H=1 Then
    Return G1
  EndIf;
  SuppH:=Support(H);
  U1:= NewList(Len(SuppH));
  Exponentliste:=NewList(Len(SuppH));
  For A:=1 To Len(SuppH) Do
    ExpA:=Log(SuppH[A]);
    Exponentliste[A]:=NewList(Len(ExpA),0);
    For A1 :=1 To Len( ExpA) Do
      If (ExpA[A1] <>0 And Mod( ExpA[A1] , P ) =0) Then
        Exponentliste[A][A1]:=ExpA[A1]/P;
      EndIf;
    EndFor;
    U1[A]:=LogToTerm(Exponentliste[A]);
  EndFor;
  CoeffV:=Coefficients(H,SuppH);
  CoeffV1:=NewList(Len(SuppH));
  For K:=1 To Len(SuppH) Do
    CoeffV1[K]:=CoeffV[K];
  EndFor;
  V:= ScalarProduct(CoeffV1,U1);
```

Zum Abschluß erfolgt die rekurisve Anwendung von `Sep0` und abschließender Ausgabe des Polynoms `G3`, als dem separablen Teil des Polynoms `F`.

```
  G2:=Sep0(V,X);
  G3:=Sep0(G1*G2,X);
  Return G3;
EndDefine;
```

4.1.2 MaxIndepSet

In diesem Abschnitt wird die Implementation des Algorithmus MAXINDEPSET der vorliegenden Arbeit durch die Funktion `MaxIndepSet(I)` beschrieben.
Die Funktion `MaxIndepSet(I)` ist wie der Algorithmus MAXINDEPSET in drei Schritte unterteilt. Zunächst wird das Leittermideal des Ideals I berechnet und seine Erzeugenden bestimmt.

```
Define MaxIndepSet(I)
  N:=NumIndets();
  P:=Characteristic();
  LTI:=LT(I);
  GensLTI:=Gens(LTI);
```

Die Berechnung des minimalen monomialen Erzeugendensystems von $\sqrt{\mathrm{LT}_\sigma(I)}$ ist in der Funktion in zwei Bereiche unterteilt, die durch die beiden großen `For`-Schleifen gekennzeichnet sind. Die erste `For` - Schleife bestimmt für jeden Erzeuger von $\sqrt{\mathrm{LT}_\sigma(I)}$ seinen quadratfreien Teil und fügt diesen der Liste `Squarefree` hinzu.

```
  Squarefree:=[];
  For K:=1 To Len(GensLTI) Do
     LogK:=Log(GensLTI[K]);
     For L1:=1 To N Do
         If P=0 Then
            If LogK[L1] <> 0 Then
               LogK[L1]:=1;
            EndIf;
            Else If Mod(LogK[L1],P)<>0 Then
                    LogK[L1]:=1;
                 EndIf;
         EndIf;
     EndFor;
     Append(Squarefree,LogToTerm(LogK));
  EndFor;
```

Die zweite `For`-Schleife entfernt aus der Liste `Squarefree` die quadratfreien Teile, die echte Vielfache von anderen quadratfreien Teilen sind. Hierzu wird überprüft, ob ein Listenelement ein Vielfaches von den folgenden Listenelementen ist. Tritt dieser Fall ein, wird das jeweilige Element gleich 0 gesetzt. Zum Abschluß werden diese 0 aus der Liste `Squarefree` entfernt.

```
  For K:=1 To Len(Squarefree)-1 Do
     If Not IsZero( Squarefree[K] ) Then
```

```
        For L1:=K+1 To Len(Squarefree) Do
           If Not IsZero( Squarefree[L1] ) Then
              If NR( Squarefree[K],[ Squarefree[L1] ]) =0 Then
                 Squarefree[K]:=0;
                 Break
                 Elsif NR( Squarefree[L1],[ Squarefree[K] ]) =0 Then
                    Squarefree[L1]:=0;
              EndIf;
           EndIf;
        EndFor;
     EndIf;
  EndFor;
  Squarefree:=NonZero(Squarefree);
```

Zur Umsetzung von Schritt 3 wird nun die Funktion `IndepSet(Y,T)` in Anbindung an die Prozedur INDEPSET in Algorithmus 2.5.4 angewandt und es erfolgt die Ausgabe der Liste `Indep`, einer Menge maximal unabhängiger Variablen.

```
  Indetliste:=Indets();
  Indep:=IndepSet( Indetliste, Squarefree );
  Return Indep;
EndDefine;
```

Die Funktion `IndepSet(Y,T)` ist wie folgt implementiert.
Diese Funktion ist wie die Prozedur INDEPSET in 3 Schritte gegliedert. Zu Beginn wird das Abbruchkriterium der rekursiven Schleife überprüft.

```
Define IndepSet(Y,T);
  If T=[] Then
     Return Y
  EndIf;
```

Die `For`-Schleife setzt Schritt 2 um. Darin entspricht `L[K]:=IndepSet(Yk,Tk)` dem Anwenden der Prozedur INDEPSET in $L_i :=$ INDEPSET$(Y \setminus \{x_i\}, T_i)$.

```
  L:=NewList( Len(Y),0 );
  For K:=1 To Len(Y) Do
     If NR( Max(T) ,[ Y[K] ]) =0 Then
        Tk:=[ K1 In T | NR( K1 , [ Y[K]]) <>0 ];
        Yk:=Diff(Y, [Y[K]] );
        L[K]:=IndepSet( Yk, Tk );
     EndIf;
  EndFor;
```

Den Abschluß bildet die nächste `For`-Schleife, die in der Liste `L` eine Menge L_i mit der größten Elementanzahl sucht. Hierzu wird zuerst maximale Länge einer Liste in `L` bestimmt und anschließend wird die erste Liste mit dieser Länge ausgegeben.

```
  MaxL:=Max( [ Len(Liste) | Liste In L And Liste <>0  ] );
  For K1:=1 To Len(L) Do
     If Len(L [K1]) = MaxL Then
        If L[K1]<>0 Then
           Return L[K1]
        EndIf;
     EndIf;
  EndFor;
EndDefine;
```

Mit der Anwendung von `IndepSet(Y,T)` in der Funktion `MaxIndepSet(I)` erfolgt die Ausgabe, eine maximal unabhängige Menge modulo I.
Der Algorithmus MAXINDEPSET kann demnach ebenfalls vollständig implementiert werden.

4.1.3 RadikalKemper

In diesem Abschnitt wird mit Hilfe der Funktion `RadikalKemper(I)` die Implementation des Algorithmus RADIKALKEMPER dargestellt.
Zunächst wird überprüft, ob der Input des Ideals `I` auch korrekt im aktuellen Ring definiert ist.

```
Define RadikalKemper(I)
  If RingEnv() <> RingEnv(I) Then
    Print "Error:
           Aktueller Ring und der zum Ideal I gehörende Ring
           stimmen nicht überein.";
    Return;
  EndIf;
```

Danach folgt das erste Abbbruchkriterium, falls das Ideal I das von 1 erzeugte Ideal ist.

```
  G:=[1];
  If 1 IsIn I Then
    Return I;
    Else
```

Ist dies nicht der Fall, wird bei der Berechnung unterschieden, ob das Ideal I einmal null-dimensional ist oder ob I ein höher-dimensionales Ideal ist.

```
If Dim(CurrentRing()/I) =0 Then
```

Wenn das Ideal I null-dimensional ist, erfolgt die Radikalberechnung nach Bemerkung 3.1.8 der vorliegenden Arbeit. Hierzu sind separable Polynome zu bestimmen. Es wird erst das Eliminationsideal $I \cap K[x_i]$ für alle i berechnet und anschließend ist deren jeweiliger Erzeuger zu ermitteln. Aus den Erzeugern wird mit Hilfe der Funktion `Sep0` wiederum der separable Teil des Polynoms berechnet. Die Addition des Ideals I und des von den separablen Teilen erzeugten Ideals ergeben nun das Radikal von I.

```
P:=Characteristic();
N:=NumIndets();
SeparabelF:=NewList(N);
For K:=1 To N Do
  Xk:=WithoutNth(Indets(),K);
  Eliminationsideal:=Elim(Xk,I);
  GensK:=ReducedGBasis(Eliminationsideal);
  FK:=Head(GensK);
  SeparabelF[K]:=Sep0(FK,Indet(K));
EndFor;
J:= I + Ideal(SeparabelF);
Return J
```

Im zweiten Fall ist das Radikal des höher-dimensionalen Ideals I zu berechnen. Diese Berechnung ist in Teilschritte unterteilt. Als erstes wird mit der Funktion `MaxIndepSet(I)` eine maximal unabhängige Menge modulo I bestimmt. Danach wird ein neuer Ring durch
`NewRing::=CoeffRing[t[1..Mi],x[1..(Ni - Mi)]]` definiert, in dem die Variablen `t[1..Mi]` den Variablen der maximal unabhängigen Menge modulo I entsprechen. In Bezug auf den neuen Ring wurden auch die Abbildungsvorschriften festgelegt, um Objekte zwischen den Ringen verschieben zu können.

```
Else
  Liste:=MaxIndepSet(I);
  LenComdim:=Len(Liste);
  Indetsliste:=Indets();
  RMapliste:=NewList(NumIndets());
  RMaplisteA:=NewList(NumIndets());
  Mi:=0;
  Ni:=LenComdim;
  For Xi:=1 To NumIndets() Do
    If Indet(Xi) IsIn Liste Then
       Mi:=Mi+1;
       RMapliste[Mi]:=Xi;
```

```
      RMaplisteA[Mi]:=Indet(Xi);
      Else Ni:=Ni+1;
           RMapliste[Ni]:=Xi;
           RMaplisteA[Ni]:=Indet(Xi);
  EndIf;
EndFor;
NewRing::=CoeffRing[ t[1..Mi], x[1..(Ni - Mi)] ] ;
```

Das Ideal I wird in diesen neuen Ring eingebracht und dort als `Inew` definiert.

```
Using NewRing Do
  RMapliste1:=NewList(NumIndets());
  For Xi:=1 To Mi Do
    RMapliste1[ RMapliste[Xi] ]:=t[Xi];
  EndFor;
  For Xi:=Mi+1 To Ni Do
    RMapliste1[ RMapliste[Xi] ]:=x[Xi - Mi];
  EndFor;
  Fnew:=RMap( RMapliste1 );
  Inew:=Image(I, Fnew );
  P:=Characteristic();
  M:=Len(t);
  N:=Len(x);
  RI:=NewList(N,0);
  K1::=CoeffRing[t[1..M],y[1..M],x[1..N]], Elim(t[1]..y[M]);
```

Es folgt die Definition des Ringes `K1`, in dem das Ideal `Inew` null-dimensional ist. In diesem neuen Ring wird das Radikal des null-dimensionalen Ideals I durch Implementierung des Algorithmus 3.1.7 KEMPER berechnet. Es werden im Folgenden die 5 Schritte des Algorithmus KEMPER implementiert. Wie schon im null-dimensionalen Fall werden für den Schritt 1 in KEMPER die Eliminationsideale $I \cap K[x_i]$ berechnet und danach werden deren Erzeuger bestimmt. Im nächsten Schritt ergeben sich mit der Funktion `Sep` die separablen Teile dieser Erzeuger und diese sind in `SeparabelF` gelistet.

```
  SeparabelF:=NewList(N);
  For K:=1 To N Do
    Xk:=WithoutNth(x,K);
    Eliminationsideal:=Elim(Xk,Inew);
    GensK:=ReducedGBasis(Eliminationsideal);
    FK:=Head(GensK);
    CoeffFK:=Coefficients(FK,Indet(M+K));
    SuppFK:=Support(FK);
    If Len(SuppFK)=1 Then
       LogFK:=Log(FK);
```

```
      FK:=Indet(M+K)^(LogFK[M+K]);
    EndIf;
    SeparabelFK:=Sep(FK,Indet(M+K),RI[K]);
    SeparabelF[K]:=SeparabelFK[1];
    RI[K]:=SeparabelFK[2];
  EndFor;
```

Zur Umsetzung der nächsten Schritte wird in den Ring `K1` gewechselt. Wie in Schritt 3 wird erst das maximale r bestimmt und hier mit `Rmax` bezeichnet. Anschließend wird `SeparabelF` in `K1` eingebracht und in den Polynomen werden die Variablen `t[1..M]` substituiert. Es folgt die Bildung des Ideals J nach Schritt 4. Für den letzten Schritt wird das Eliminationsideal von `J` bzgl. `y[1]..y[M]` berechnet.

```
Using K1 Do
   Rmax:=Max( RI );
   Qr:= P^Rmax;
   Gpolynome:= NewList(N);
   SeparabelF1:=BringIn(SeparabelF);
   For K:=1  To N Do
     Substitution:=[ [t[Kindex], y[Kindex]^(P^(Rmax - RI[K]))] |
                     Kindex In 1..M ];
     Gpolynome[K]:= Subst(SeparabelF1[K],Substitution);
   EndFor;
   Identitaet:=Concat([t[K] | K In 1..M] , [ x[K] | K In 1..N ]);
   Rmap1:=RMap(Identitaet);
   IK1:= Image(Inew, Rmap1);
   YT:= [ y[K]^Qr - t[K] | K In 1..M ];
   J:= IK1 + Ideal(Gpolynome) + Ideal(YT);
   Js := Elim( y[1]..y[M] , J );
 EndUsing;
```

Im Anschluß wird in den Ring `NewRing` zurückgekehrt und das Ideal `Js` dort eingebracht.

```
IdentitaetK1:=Concat([t[K] | K In 1..M] ,
              NewList(M,0), [ x[K] | K In 1..N ]);
Rmap2:=RMap(IdentitaetK1);
Js1:= Image( Js, Rmap2);
```

Die Termordung des aktuellen Rings wird nun in eine Eliminationsordnung für die Variablen `x[1..N]` geändert.
Die Berechnung des Kontraktionsideals des zuvor berechneten Ideals erfolgt durch den Algorithmus CONT.
Hierzu wird mit `H:=GBasis(Js1K)` eine Gröbner-Basis des Ideals ermittelt.

Für jedes Polynom der Gröbner-Basis sollte nun der kleinste gemeinsame Vielfache aus allen Nennern der Koeffizienten in $K(Y)$ berechnet. Da die Polynome aber keine Nenner in dieser Form haben, fällt der Schritt weg. Es folgt die Berechnung des kleinsten gemeinsamen Vielfachen der Leitkoeffizienten der Polynome der Gröbner-Basis, deren Leitkoeffizienten die Elemente aus $K(Y)$ sind. Daran schließt die Berechnung der Saturierung aus dem Algorithmus an und die Bestimmung der Erzeugenden dieses Ideals.

```
NewRingK::= CoeffRing[ t[1..M], x[1..N] ] , Elim(x);
Using NewRingK Do
    RMapK:=RMap(Indets() );
    Js1K:=Image(Js1,RMapK);
    H:=GBasis(Js1K);
    S:=Len(H);
    LcH:=NewList(S);
    NewN:=NewList(N,0);
    For K:=1 To S Do
      LcHK:=LC(H[K]);
      LogHK:=Log(H[K]);
      NewM:=First(LogHK,M);
      NewNM:=Concat(NewM,NewN);
      LcH[K]:=LcHK * LogToTerm(NewNM);
    EndFor;
    LCMH:=LCM(LcH);
    CONTK:=Gens( Saturation(Ideal(H), Ideal(LCMH) ) );
```

Danach wird, wie in Schritt 6 angegeben, mit Hilfe des Algorithmus EXTCONT die Berechnung eines Polynoms f umgesetzt. Hierzu wird eine Gröbner-Basis des Ideals `Inew` im Ring `NewRingK` ermittelt. Die Berechnung des Polynoms f, des kleinsten gemeinsamen Vielfachen der Leitkoeffizienten der Gröbner-Basis, wobei der das Leitkoeffizient als Element in $K(Y)[X \setminus Y]$ zu sehen ist, folgt darauf.

```
GBI:=GBasis(Inew);
S:=Len(GBI);
LcI:=NewList(S);
NewN:=NewList(N,0);
For K:=1 To S Do
  LcIK:=LC(GBI[K]);
  LogIK:=Log(GBI[K]);
  NewM:=First(LogIK,M);
  NewNM:=Concat(NewM,NewN);
  LcI[K]:=LcIK * LogToTerm(NewNM);
EndFor;
```

```
      LCMf:=LCM(LcI);
    EndUsing;
```

Die Erzeuger des zuvor ermittelten Kontraktionsideals und das darauf nach Schritt 6 bestimmte Polynom werden in den Ring, mit dem die Berechnungen begonnen haben, zurückgeführt.

```
    CONT:=BringIn(CONTK);
    ECF:=BringIn(LCMf);
    EndUsing;
    Fback:=RMap(RMaplisteA);
    CONT1:=Image(CONT,Fback);
    ECF1:=Image(ECF,Fback);
```

Es folgt der letzte Schritt des Algorithmus mit der Durchschnittsbildung. Die Ausgabe ist der Durchschnitt jener Ideale.

```
    RadI:=Intersection(RadikalKemper(I+Ideal(ECF1)),Ideal(CONT1));
    EndIf;
  EndIf;
  Return RadI;
EndDefine;
```

Abschließend ist zu erwähnen, dass in den Algorithmen die Ringe beispielsweise die Form $K(Y)[X \setminus Y]$ mit dem rationalen Funktionenkörper $K(X)$ hatten, aber in der Implementierung daraus die Polynomringe $K[Y, X \setminus Y]$ wurden. Die Möglichkeit, die Ringe so zu variieren und ohne rationale Funktionenkörper zu arbeiten, zeigt der folgende Satz.

Satz 4.1.3.
Sei K ein vollkommener Körper,
sei σ eine Termordnung auf $\mathbb{T}(x_1, .., x_n)$, und seien
$P = K(t)[x_1, .., x_n]$ sowie $Q = K[t, x_1, .., x_n]$.
Sei I ein Ideal in P und sei J das von den nennerfreien rationalen Funktionen in I erzeugte Ideal in Q.
Für $i = 1, \ldots, n$ gilt dann:

$$I \cap K(t)[x_i] = (J \cap K[t, x_i])K(t)[x_i]$$

Zunächst beweisen wir die Inklusion $\subseteq$.
Sei $I = \langle \frac{f_1}{g_1}, \ldots, \frac{f_s}{g_s} \rangle$ mit $f_j \in K(t)[x_i], g_j \in K[t]$ für $j = 1, \ldots, s$.
Es sei $f \in I \cap K(t)[x_i]$ mit $f = \frac{h_1}{k_1} \cdot \frac{f_1}{g_1} + \ldots + \frac{h_s}{k_s} \cdot \frac{f_s}{g_s} \in K(t)[x_i]$ mit $h_j \in K[t, x_i]$ und $k_j \in K[t]$ für $j = 1, \ldots, s$.
Wegen $\langle f_1, \ldots, f_s \rangle \subseteq J$ genügt es zu zeigen, dass

$f \in (\langle f_1, \ldots, f_s \rangle \cap K[t, x_i])K(t)[x_i]$ gilt.
Dann sei $\tilde{f} := f \cdot \prod_{j=1}^{s} k_j g_j$.
Es gilt $\tilde{f} = (h_1 \cdot \prod_{j=2}^{s} k_j g_j) f_1 + \cdots + (h_s \cdot \prod_{j=1}^{s-1} k_j g_j) f_s \in \langle f_1, \ldots, f_s \rangle$.
Da $\tilde{f} \in K[t, x_i]$ gilt, folgt somit $\tilde{f} \in (\langle f_1, \ldots, f_s \rangle \cap K[t, x_i])K(t)[x_i]$.
Es folgt also $f \in (\langle f_1, \ldots, f_s \rangle \cap K[t, x_i])K(t)[x_i]$ wegen $f = \tilde{f} / \prod_{j=1}^{s} k_j g_j$.
Als nächstes wird gezeigt $\supseteq$.
Sei τ die Eliminationstermordnung für $\{x_j, j \neq i\}$ und sei $G = \{g_1, \ldots, g_r\}$ die reduzierte $\tau-$Gröbnerbasis von J.
Dann sei $\hat{\tau}$ die Restriktion von τ auf $\hat{\mathbb{T}}$, der Menge von Termen, die nur die Variablen aus $\{x_1, \ldots, x_n\}$ enthalten, die nicht in $\{x_j, j \neq i\}$ sind.
Ohne Einschränkung seien die Polynome g_j so nummeriert, dass $\{g_1, \ldots, g_r\}$ mit $q \leq r$ die reduzierte $\hat{\tau}-$Gröbnerbasis von $J \cap K[t, x_i]$ sei.
Sei $f \in \langle g_1, \ldots, g_q \rangle$.
Schreibe $f = \frac{h_1}{k_1} g_1 + \ldots + \frac{h_q}{k_q} g_q$ mit $h_j \in K[t, x_i]$, $k_j \in K[t]$ für $j = 1, \ldots, q$.
Daraus folgt $f \cdot (k_1 \cdots k_q) \in \langle g_1, \ldots, g_q \rangle \subseteq J \cap K[t, x_i] \subseteq I \cap K(t)[x_i]$.
Es folgt $f \in I \cap K(t)[x_i]$, da $(k_1 \cdots k_q)$ eine Einheit in $K(t)[x_i]$ ist. $\square$

Wie dargestellt läßt sich der Algorithmus RADIKALKEMPER vollständig implementieren.

4.1.4 RadikalMatsumoto

In diesem Abschnitt wird der Algorithmus RADIKALMATSUMOTO durch die Funktion `RadikalMatsumoto` implementiert.
Die Implementierung des Algorithmus RADIKALMATSUMOTO in seine Funktion erfolgt ohne besondere Abweichungen. So entsprechen die Argumente der Funktion dem Input des Algorithmus. Das Argument ist eine Liste von Polynomen, die eine Gröbner-Basis von I bilden. Zur Konstruktion der Schleife wird `Repeat`-Schleife angewandt. Nach dem Schritt 1 des Algorithmus RADIKALMATSUMOTO sollten die Koeffizienten der Polynome angepasst werden, dieser Schritt fällt hier weg, da in den hier behandelten Beispielen sich auf den Fall beschränkt wird, dass die Anzahl der Körperelemente und die Charakteristik des Körpers gleich sind.

```
Define RadikalMatsumoto(B)
  N:=NumIndets();
  R1::=CoeffRing[  x[1..N],y[1..N] ],Elim( x );
  P:=Characteristic();
  Repeat
```

```
Phic1:=B;
```

Für Schritt 2 wird in einem zu konstruierenden Ring `R1` eine Gröbner-Basis für die Menge B bzgl. einer Eliminationsordnung für die $\{x_1, \ldots, x_n\}$ mit der Funktion `GBasis` berechnet. Danach wird für das von der Gröbner-Basis erzeugte Ideal das Eliminationsideal für $\{x_1, \ldots, x_n\}$ bestimmt, indem aus der Gröbner-Basismenge die Polynome entfernt werden, die eine der Variablen aus $\{x_1, \ldots, x_n\}$ enthalten. Im Anschluß werden die y_i-Variablen durch die x_i-Variablen ersetzt.

```
Using R1 Do
  Phi:=BringIn(Phic1);
  YX:=[ y[K] - x[K]^P  | K In 1..N ];
  Phiv:=Concat(Phi,YX);
  IdealPhic:=Ideal(Phiv);
  B1:=GBasis(IdealPhic);
  B2:=B1;
  For K:=Len(B1) To 1 Step -1 Do
     SuppBk:=Support(B1[K]);
     K1:=1;
     While K1 <= Len(SuppBk) Do
        If First(Log(SuppBk[K1]),N)<>NewList(N,0)
           Then Remove(B2, K);
                Break;
           Else K1:=K1 + 1;
        EndIf;
     EndWhile;
  EndFor;
  Substitute:=[ [y[K],x[K]] | K In 1..N ];
  B2:= Subst(B2, Substitute );
EndUsing;
```

Danach erfolgt ein Test, der überprüft, ob das Ideal aus dem Input mit dem zuvor berechneten Ideal gleich ist. Ist dies der Fall, wird die `Repeat`-Schleife unterbrochen und es erfolgt die Ausgabe mit `Return`, andernfalls wird zum Beginn der Schleife zurückgekehrt. Die Ausgabe der Funktion ist eine Liste von Polynomen.

```
  B2s:=BringIn(B2);
  If Ideal(B) = Ideal(B2s) Then
    Break;
    Else  B:=B2s;
  EndIf;
EndRepeat;
```

```
  Return B2s;
EndDefine;
```

Auch hier zeigt sich, dass eine vollständige Implementierung des Algorithmus RADIKALMATSUMOTO möglich ist.

4.1.5 RadikalLaplagne

In diesem Abschnitt wird durch die Funktion `RadikalLaplagne` die Implementation des Algorithmus RADIKALLAPLAGNE dargestellt. Die folgende Funktion gilt für den Fall, das der Grundkörper von positiver Charakteristik ist.
Zu Beginn wird, wie im Algorithmus angegeben ist, das Ideal definiert, das von der 1 erzeugt wird. Nach dem Schritt 2 wird eine `Repeat`-Schleife angelegt, in deren Verlauf ein Erzeuger g von $\tilde{P}$ gesucht wird, der nicht im Radikal von I enthalten ist. Die Überprüfung erfolgt ebenso per `Repeat`-Schleife, indem die Liste der Erzeugerelemente abgeschritten wird. Danach wird die jeweilige Saturierung des Ideals I und des Erzeugerelements berechnet und es wird getestet, ob die 1 ein Element dieser Saturierung ist. Falls kein solches Element gefunden wird, wird die Schleife unterbrochen und es erfolgt die Ausgabe.

```
Define RadikalLaplagne(I)
  P:=Ideal(1);
  GensI:=Gens(I);
  Repeat
    K1:=0;
    G:=0;
    Repeat
      K1:=K1+1;
      GensP:=Gens(P);
      SatK:= Saturation( I, Ideal(GensP[K1]) );
      If Not (1 IsIn SatK) Then
        G:=GensP[K1];
        J1:=SatK;
        Break;
      EndIf;
    Until K1= Len(GensP);
    If G=0 Then
      Break
    EndIf;
```

Danach folgt eine Überprüfung der Dimension des gerade definierten Ideals `J1`. Ist die Dimension 0, so erfolgt die Radikalberechnung von `J1` nach dem

Lemma von Seidenberg über die Bestimmung der separablen Teile von Polynomen. Diese werden mit der Funktion `Sep0` ermittelt.

```
If Dim(CurrentRing()/J1)=0 Then
   N:=NumIndets();
   SeparabelF:=NewList(N);
   For K:=1 To N Do
      Xk:=WithoutNth(Indets(),K);
      Eliminationsideal:=Elim(Xk,J1);
      GensK:=ReducedGBasis(Eliminationsideal);
      FK:=Head(GensK);
      SeparabelF[K]:=Sep0(FK,Indet(K));
    EndFor;
    RadJ:= J1 + Ideal(SeparabelF) ;
```

Ist die Dimension positiv, wird mit der Funktion `MaxIndepSet` eine unabhängige Menge modulo I bestimmt. Danach werden im Zuge der Definition des neuen Ringes die Abbildungsvorschriften festgelegt, damit Objekte zwischen den Ringen verschoben werden können.

```
   Else
Liste:=MaxIndepSet(J1);
LenComdim:=Len(Liste);
Indetsliste:=Indets();
RMapliste:=NewList(NumIndets());
RMaplisteA:=NewList(NumIndets());
Mi:=NumIndets() - LenComdim ;
Ni:=0;
For Xi:=1 To NumIndets() Do
  If Indet(Xi) IsIn Liste Then
    Mi:=Mi+1;
    RMapliste[Mi]:=Xi;
    RMaplisteA[Mi]:=Indet(Xi);
    Else Ni:=Ni+1;
      RMapliste[Ni]:=Xi;
      RMaplisteA[Ni]:=Indet(Xi);
  EndIf;
EndFor;
NewRing::=CoeffRing[ x[1..Ni] ,t[1.. (Mi - Ni)] ] ,Elim(x);
Using NewRing Do
  RMapliste1:=NewList(NumIndets());
  For Xi:=1 To Ni Do
    RMapliste1[ RMapliste[Xi] ]:=x[Xi];
  EndFor;
```

```
For Xi:=Ni +1 To Mi Do
  RMapliste1[ RMapliste[Xi] ]:=t[Xi- Ni];
EndFor;
Fnew:=RMap( RMapliste1 );
Jnew:=Image(J1, Fnew );
J1:=Jnew;
```

Die folgende Berechnung des Radikals des null-dimensionalen Ideals basiert auf dem Algorithmus von KEMPER und die Implementation erfolgt, wie in der Funktion `RadikalKemper` geschehen. Ein Unterschied liegt nur darin, dass der Ring, in dem das Radikal zu berechnen ist, als Ordnung die Eliminationsordnung `Elim(x)` verwendet.

```
P1:=Characteristic();
M:=Len(t);
N:=Len(x);
RI:=NewList(N,0);
NewRingK1::=CoeffRing[x[1..N],t[1..M],y[1..M]], Elim(t[1]..y[M]);
SeparabelF:=NewList(N);
For K:=1 To N Do
  Xk:=WithoutNth(x,K);
  Eliminationsideal:=Elim(Xk,J1);
  GensK:=ReducedGBasis(Eliminationsideal);
  FK:=Head(GensK);
  RIK:=RI[K];
  IndetK:=Indet(K);
  SeparabelFK:=Sep(FK,IndetK,RIK);
  SeparabelF[K]:=SeparabelFK[1];
  RI[K]:=SeparabelFK[2];
EndFor;
Using  NewRingK1 Do
  Rmax:=Max( RI );
  Qr:= P1 ^Rmax;
  Gpolynome:= NewList(N);
  SeparabelF1:=BringIn(SeparabelF);
  For K:=1  To N Do
    Substitution:=[ [t[Kindex], y[Kindex]^(P1^(Rmax - RI[K]))] |
                    Kindex In 1..M ];
    Gpolynome[K]:= Subst(SeparabelF1[K],Substitution);
  EndFor;
  Identitaet:=Concat(  [ x[K] | K In 1..N ],[t[K] | K In 1..M]);
  Rmap1:=RMap(Identitaet);
  IK1:= Image(J1, Rmap1);
```

```
      YT:= [ y[K]^Qr - t[K] | K In 1..M ];
      J:= IK1 + Ideal(Gpolynome) + Ideal(YT);
      Js := Elim( y[1]..y[M] , J );
    EndUsing;
    IdentitaetK1:=Concat( [ x[K] | K In 1..N ],[t[K] | K In 1..M] ,
                          NewList(M,0));
    Rmap2:=RMap(IdentitaetK1);
    RadikalJ:= Image( Js, Rmap2);
```

Nach der Berechnung des Radikals wird ein Kontraktionsideal berechnet. Hierzu wird erst eine Gröbner-Basis des Radikals berechnet, anschließend werden für die Erzeuger der Gröbner-Basis, die als Polynome in $K[Y][X \setminus Y]$ betrachtet werden, die Leitkoeffizienten als Polynome in $K[Y]$ bestimmt. Von diesen Leitkoeffizienten ist der kleinste gemeinsame Vielfache zu ermitteln. Darauf wird eine Saturierung berechnet und das Kontraktionsideal ist bestimmt.

```
    GBRadikalJ:=GBasis(RadikalJ);
    S:=Len(GBRadikalJ);
    LcJ:=NewList(S);
    For K:=1 To S Do
      LcJK:=LC(GBRadikalJ[K]);
      LogJs:=Log(GBRadikalJ[K]);
      NewJs:=Concat( NewList(N,0) , Last(LogJs,M));
      LcJ[K]:=LcJK * LogToTerm(NewJs);
    EndFor;
    H:=LCM(LcJ);
  EndUsing;
  Fback:=RMap(RMaplisteA);
  GBRadJ:=Image(GBRadikalJ,Fback);
  Hx:=Image(H,Fback);
  RadJ:=Saturation(Ideal(GBRadJ), Ideal(Hx));
  EndIf;
```

Der Abschluß ist die Bildung eines Schnittes wie im Algorithmus angegeben und es folgt die Rückkehr zum Beginn der `Repeat`-Schleife. Die Ausgabe bildet das Ideal aus dem zuvor berechneten Durchschnitt.

```
  P:= Intersection(P, RadJ);
EndRepeat;
Return P;
EndDefine;
```

Der Algorithmus RADIKALLAPLAGNE kann demnach vollständig implementiert werden.

4.2 Beispiele und Analyse

Nachdem die Implementierung der drei Algorithmen durch die verschiedenen Funktionen ermöglicht wurde, werden in dem folgenden Abschnitt die Ergebnisse von Berechnungen ausgewählter Beispiele auf der Grundlage der vorgestellten Implementierung analysiert. Die verwendeten Beispiele werden aus Decker, Greuel und Pfister [5] und aus Caboara, Conti und Traverso [3] entnommen. Die Dimensionen der Ideale in den Beispielen sind sowohl positiv als auch 0. Es wurden insgesamt 30 verschiedene Beispiele von höherdimensionalen Idealen verwendet und 7 Beispiele von null-dimensionalen Idealen verwendet. Die Beispiele werden jeweils mit den Charakteristiken 2, 3, 5, 7, 11, 53 und 251 berechnet. Diese Charakteristiken wurden von Kemper [11] und Matsumoto [17] genutzt und diese werden als Grundlage für die vorliegende Arbeit gesehen, so dass die Charakteristiken übernommen wurden. Die Ergebnisse aus der Anwendung der Beispiele in die Implementation der Algorithmen werden tabellarisch im Anhang A aufgeführt. Die Tabelle 1 stellt die Timings der höher-dimensionalen Beispiele aus Caboara, Conti und Traverso [3] dar. Die Tabelle 2 beinhaltet die Timings der höherdimensionalen Beispiele aus Decker, Greuel und Pfister [5]. In der Tabelle 3 werden die null-dimensionalen Beispiele aus Caboara, Conti und Traverso [3] und Decker, Greuel und Pfister [5] gemeinsam dargestellt.
Die Berechnungen wurden auf einem Computer mit einem $1,73$GHz Intel Pentium Prozessor und einem Arbeitsspeicher von 1GB durchgeführt. Die Werte in den Tabellen geben die Berechnungsdauer des Beispielideals in der jeweiligen Funktion bezüglich der entsprechenden Charakteristik des Grundkörpers in Sekunden an.
Die folgende Analyse der Ergebnisse erfolgt in drei Schritten.

1. In einem ersten Schritt werden Berechnungszeiten aus den verschiedenen Artikeln mit den Ergebnissen der eigenen Berechnung auf der Basis der Implementierung verglichen.

2. In einem zweiten Schritt werden die drei Algorithmen im Hinblick auf ihre Leistungsfähigkeit bei der Berechnung der Beispiele analysiert.

3. In einem dritten Schritt wird darauf aufbauend analysiert, welche Beispiele von welchem Algorithmus im Hinblick auf die Berechnungsdauer am schnellsten berechnet werden können.

Die in den Artikeln von Kemper [11], Matsumoto [17] oder Laplagne [15] berechneten Beispiele werden wie oben beschrieben zunächst in Verbindung mit den Ergebnissen der eigenen Berechnung untersucht.

Ein Vergleich der berechneten Beispiele von Kemper [11] und Matsumoto [17] mit den eigenen Berechnungen ist nur bezüglich der Beispiele in Tabelle 1 möglich. Die Beispiele aus Tabelle 2 werden von Kemper und Matsumoto nicht berechnet. Im Vergleich mit den Beispielen aus der Tabelle in Kemper [11] wird deutlich, dass die Beispiele **CCT-L**, **CCT-Acht3** und **CCT-C** mit der Implementation des Algorithmus RADIKALKEMPER sowohl in dieser Arbeit (siehe Tabelle 1) als auch in Kemper [11] nicht innerhalb des vorgegebenen Zeitrahmens von einer halben Stunde vollständig berechnet werden können. Die Zeitbegrenzung auf eine halbe Stunde wird dadurch begründet, dass nicht zu erwarten ist, dass die Beispiele noch effektiv berechnet werden können. Die Beispiele **CCT-M** und **CCT-D** und die null-dimensionalen Beispiele **CCT-E2** und **CCT-E3** können das Radikal bzgl. jeder Charakteristik erfolgreich berechnen. Die eigenen Berechnungen nehmen insgesamt mehr Zeit in Anspruch als bei Kemper [11] beschrieben. Dies kann dadurch erklärt werden, dass Kemper andere technische Voraussetzungen und ein anderes Programm zur Implementation (MAGMA) nutzte.
Bei der Anwendung des Algorithmus RADIKALMATSUMOTO wird nur das Beispiel **CCT-M** für jede angebene Charakteristik von den eigenen Progammen berechnet. Dies ist auch in den in Kemper [11] und Matsumoto [17] angegebenen Tabellen ebenso der Fall. Für das Beispiel **CCT-O** kann der Algorithmus RADIKALMATSUMOTO bis auf Charakteristik 251 bei allen anderen Charakteristiken ein Ergebnis erzielen. Die Beispiele **CCT-L** und **CCT-C** liefern nur für die Charakteristik 2 ein Ergebnis, nicht aber für größere Charakteristiken. Kemper [11] und Matsumoto [17] stellten dies ebenfalls fest. Die Beispiele aus Tabelle 1 lassen sich in dieser Arbeit meist nicht für größere Charakteristiken berechnen. Das Beispiel **CCT-Acht3** kann in der Anwendung in Kemper [11], Matsumoto [17] und in den eigenen Berechnungen für keine Charakteristik das Radikal berechnen. So konnten die null-dimensionalen Beispiele **CCT-E2** und **CCT-E3** nur für die Charakteristik 2 angewandt werden. Kemper [11] und Matsumoto [17] können diese Beispiele auch nicht für größere Charakteristiken als Charakteristik 2 berechnen. Ihre Berechnungen zeigen sich den eigenen Berechnungen darin überlegen, dass die Ergebnisse etwas schneller berechnet werden.
Alle Beispiele, die mit der implementierten Funktion `RadikalLaplagne` berechnet werden, können wie in Laplagne [15] erfolgreich zu Ende geführt werden. Die von Laplagne angegebenen Zeiten können jedoch nicht mit der eigenen Berechnungsdauer verglichen werden, da in Laplagne [15] der Grundkörper K von Charakteristik 0 gewählt ist und in dieser Arbeit der Grundkörper als Körper von positiver Charakteristik betrachtet wird.
Im Folgenden werden die einzelnen Algorithmen hinsichtlich ihrer Leistungsfähigkeit betrachtet. Es fällt zunächst auf, dass von 30 höher-dimensionalen

Beispielen nur 4 Beispiele (**DGP-3**, **DGP-33**, **CCT-M**, **CCT- O**) mit allen drei Methoden berechnet werden konnten. Die Anwendung der Beispiele auf den Algorithmus RADIKALKEMPER ergibt kein einheitliches Muster. Für 8 Beispiele (**DGP-4**, **DGP-21**, **DGP-24**, **DGP-28**, **CCT-L**, **CCT-C**, **CCT-Acht3** und **CCT-C**) kann keine Berechnung bzgl. einer der Charakteristiken erfolgreich durchgeführt werden. Dagegen können 15 Beispiele komplett in allen gewählten Charakteristiken berechnet werden. Insgesamt werden 64 von 210 Berechnungen (30.5%) von der Funktion `RadikalKemper` nicht ausgeführt. Die Berechnungsdauer für die Beispiele **DGP-3**, **DGP-7**, **DGP-27** oder auch **DGP-32** befindet sich über die Charakteristiken verteilt etwa im selben Zeitrahmen. Bei den Beispielen **DGP-2** und **DGP-6** konnte für die Charakteristik 2 kein Ergebnis geliefert werden ebenso wie beim Beispiel **DGP-25** nicht für die Charakteristiken 7 und 251. Diese Ergebnisse deuten daraufhin, dass bei Berechnungen mit dem Algorithmus RADIKALKEMPER die Höhe der Charakteristik keinerlei Einfluss auf die Berechnungsdauer hat.
Der Algorithmus RADIKALMATSUMOTO kann 7 Beispiele (**DGP-3**, **DGP-21**, **DGP-33**, **DGP-24**, **DGP-28**, **CCT-M**, **CCT-O**) in allen Charakteristiken berechnen und nur für das Beispiel **CCT-Acht3** kann kein Ergebnis geliefert werden. Insgesamt können 74 von 210 Berechnungen (35.2%) nicht ausgeführt werden. Für die Beispiele in der Charakteristik 2 kann das Radikal immer bestimmt werden. Wenn die Beispiele mit Körpern größerer Charakteristik berechnet werden, so ist festzustellen, dass die Berechnungen mit dem Ansteigen der Charakteristik länger dauern. Dies setzt sich soweit fort, bis die Berechnung des Radikals nicht mehr durchführbar ist, wie in den Beispielen **DGP-2**, **DGP-4**, **DGP-5**, **DGP-6**, **DGP-7** oder **DGP-30** ersichtlich ist. Bei den Beispielen **DGP-9**, **DGP-21**, **DGP-24** und **DGP-33** zeigt sich dagegen, dass die Berechnungen für jede Charakteristik in einem zeitlich ähnlichen Rahmen berechnet werden kann.
Bei der Betrachtung der Tabellen im Anhang A fällt zunächst auf, dass der Algorithmus RADIKALLAPLAGNE für jedes Beispiel das Radikal berechnen konnte, demnach gelingt es bei 100% Prozent der Beispiele ein Ergebnis zu erzielen. Die beiden anderen Algorithmen können dagegen nicht immer innerhalb der zeitlichen Frist zu einem Ergebnis kommen.
Die Zeiten der Berechnungen innerhalb eines Beispiels variieren in der Regel nicht stark mit der Charakteristik. Die Fälle mit den größten Zeitunterschieden in einem Beispiel sind **CCT-M**, **DGP-1**, **DGP-5** und **DGP-30**.
Im Hinblick auf die Beispiele der null-dimensionalen Ideale in Tabelle 3 ist festzustellen, dass der Algorithmus von Kemper nur für das Beispiel **DGP-17** mit Charakteristik 3 kein Ergebnis berechnen kann, während in den anderen Fällen immer ein Ergebnis geliefert werden kann. Dies entspricht einem Pro-

zentsatz von 2.0%. Auffällig ist hierbei, dass der Algorithmus von Kemper demnach bei der Berechnung von null-dimensionalen Beispielen eine höhere Zuverlässigkeit im Hinblick auf die Berechnung des Radikals eines Polynomideals hat. Die Berechnungszeiten liegen meist im selben zeitlichen Rahmen, bis auf die im Beispiel **DGP-18** mit Charakteristik 3 berechneten Zeiten.
Der Algortihmus RADIKALMATSUMOTO kann wiederum kein Beispiel in allen Charakteristiken berechnen. Insgesamt werden 29 von 49 Berechnungen (59.2%) nicht durchgeführt. Es können aber alle Beispiele in Charakteristik 2 berechnet werden. Wenn die Beispiele in größeren Charakteristiken berechnet werden, dauern die Berechnungen länger an. In Charakteristik 53 konnte nur ein Beispiel berechnet werden und in Charakteristik 251 ließ sich kein Beispiel mehr berechnen. Der Algorithmus RADIKALLAPLAGNE berechnet wie im Falle von Beispielen höher-dimensionaler Ideale auch im Falle von Beispielen null-dimensionaler Ideale immer das Radikal von einem Polynomideal.
Im Folgenden wird untersucht, welche Beispiele von welchem Algorithmus im Hinblick auf die Berechnungsdauer am schnellsten berechnet werden. Es werden zunächst die Beispiele der höher-dimensionalen Ideale betrachtet, um dann anschließend die Beispiele der null-dimensionalen Ideale in den Blick zu nehmen. Für die Charakteristik 2 bestätigt sich die Beobachtung von Kemper (siehe Kapitel 3, [11]), dass der Algorithmus von Matsumoto bei fast allen Beispielen die Berechnungen schneller als der Algorithmus von Kemper ausführt. Das Beispiel **DGP-12** ist das einzige Beispiel, auf das diese Aussage nicht zutrifft. Hier kann der Algorithmus von Kemper schneller zu einem Ergebnis gelangen.
Bei allen **DGP**-Beispielen stellt sich im Vergleich der drei Algorithmen heraus, dass der Algorithmus RADIKALMATSUMOTO für die Berechnungen weniger Zeit benötigt als der Algorithmus RADIKALKEMPER und RADIKALLAPLAGNE, wenn die Berechnung erfolgreich endet. Der Algorithmus RADIKALLAPLAGNE zeigt sich lediglich bei den drei Beispielen **DGP-21**, **DGP-27**, **DGP-28** dem Algorithmus von Matsumoto zeitlich überlegen. Der Vergleich der Ergebnisse der Berechnungen von Beispielen mit den Algorithmus RADIKALKEMPER und RADIKALLAPLAGNE in der Charakteristik 2 zeigt, daß die mit dem Algorithmus RADIKALLAPLAGNE berechneten Beispiele entweder ähnlich schnell oder schneller berechnet werden als mit dem Algorithmus RADIKALKEMPER. Dies zeigen die Beispiele **DGP-3**, **DGP-5** und **DGP-16** deutlich. Eine Ausnahme stellt hierbei lediglich das Beispiel **DGP-30** dar, das die Berechnung mit dem Algrithmus von Kemper viel schneller als mit dem Algorithmus von Laplagne durchführt.
Die Ergebnisse der Berechnungen bei Beispielen mit höheren Charakteristiken als 2 weisen deutlich daraufhin, dass der Algorithmus RADIKALMATSUMOTO den Algorithmen von Kemper und Laplagne un-

terlegen ist. Der Algorithmus RADIKALMATSUMOTO kann nur wenige Beispiele mit höherer Charakteristik bearbeiten. Dazu fällt auf, dass wenn die Berechnung eines Beispiels mit einer Charakteristik wie z.B. bei Charakteristik 5 von Beispiel **DGP-20** nicht gelingt, auch alle höher liegenden Charakteristiken des Beispiels nicht berechnet werden können. An den mit dem Algorithmus von Matsumoto berechneten Beispielen hängt die Dauer der Berechnung entscheidend von der einen Gröbner-Basisberechnung ab, wie dies konkret an den Beispielen zu sehen. In den Fällen, in denen die Berechnung nicht zeitig endet, dauert die Gröbner-Basisberechnung an.
Dies stellt sich bei dem Algorithmus von Kemper anders dar. Die Ergebnisse der Berechnungen bewegen sich, wie schon im vorigen Abschnitt erwähnt, innerhalb eines Beispiels meist innerhalb eines ähnlichen Zeitrahmens. Im Vergleich der Algorithmen RADIKALKEMPER und RADIKALMATSUMOTO ist festzuhalten, dass der Algorithmus RADIKALKEMPER in Bezug auf die angewendeten Beispiele insgesamt mehr Berechnungen erfolgreich beenden konnte. Vor allem bei der Berechnung von Radikalen in größeren Charakteristiken als 2 zeigt sich der Algorithmus RADIKALKEMPER zuverlässiger als der Algorithmus RADIKALMATSUMOTO. Der Algorithmus RADIKALKEMPER kann jedoch bei der Berechnung von einigen Beispiele überhaupt nicht zu einem Ergebnis gelangen.
Der Algorithmus RADIKALLAPLAGNE wiederum zeigt sich bei der Berechnung der Beispiele bei höheren Charakteristiken als 2 dem Algorithmus RADIKALKEMPER dahingehend überlegen, dass es ihm möglich ist, alle Beispiele innerhalb des vorgegebenen Zeitrahmens erfolgreich bis zum Ende zu berechnen. Im Hinblick auf die Berechnungsdauer der einzelnen Beispiele unterscheiden sich die Algorithmen RADIKALKEMPER und
RADIKALLAPLAGNE nur geringfügig. Bei den Beispielen, die mit dem Algorithmus von Kemper berechnet wurden, ist im Vergleich zu den Beispielen, die mit dem Algorithmus RADIKALLAPLAGNE berechnet wurden, zu erkennen, dass hier die Anzahl der Durchläufe des Algorithmus ungleich höher ist, aber deren jeweilige Dauer sich im allgemeinen meist im selben Zeitrahmen bewegt. Hierbei ist festzuhalten, dass der Algorithmus vom Matsumoto den Algorithmen von Kemper und Laplagne nur bei Beispielen mit Charakteristik 2 in der Berechnungsdauer überlegen ist. Bei Charakteristiken, die größer als 2 sind, benötigt RADIKALMATSUMOTO im Vergleich zu den Algorithmen nach Kemper und Laplagne meist mehr Zeit für die Berechnung oder kann gar kein Ergebnis liefern. RADIKALKEMPER und RADIKALLAPLAGNE berechnen Beispiele mit höheren Charakteristiken in einem ähnlichen Zeitrahmen und sind dem Algorithmus
RADIKALMATSUMOTO daher überlegen.
Der Algorithmus RADIKALLAPLAGNE zeichnet sich dadurch aus, dass er

als einziger der drei Algorithmen alle Beispiele höherer Charakteristiken in einem zeitlichen Rahmen berechnen kann. Er stellt sich daher als sehr zuverlässig in der Berechnung des Radikals eines Ideals dar und kann in einem guten zeitlichen Rahmen zu einem Ergebnis kommen.
Bei der Berechnung der null-dimensionalen Ideale stellt sich ein anderes Bild dar. Die Beobachtung von Kemper (siehe Kapitel 3, [11]), dass sein Algorithmus diese Ideale schneller als der Algorithmus von Matsumoto berechnet, kann unter Berücksichtigung der eigenen Berechnungen bestätigt werden. Im Vergleich dazu berechnet der Algorithmus RADIKALLAPLAGNE diese Beispiele meistens langsamer als der Algorithmus RADIKALKEMPER. Die Ausnahme sind die Beispiele **DGP-17** und die Beispiele **DGP-26** mit den Charakteristiken 2 und 3. Bei den null-dimensionalen Beispielen stellt sich demnach der Algorithmus von Kemper als der Algorithmus mit den schnellsten Berechnungszeiträumen heraus und ist dem Algorithmus von Laplagne und Matsumoto vorzuziehen.

5 Fazit

In der vorliegenden Arbeit wurden auf der Basis der mathematischen Grundlagen drei Methoden zur Berechnung des Radikals von einem Polynomideal implementiert und miteinander verglichen.
Die mathematische Fundierung der Algorithmen von Kemper, Matsumoto und Laplagne und der anschließende Vergleich der drei Algorithmen zeigte die unterschiedlichen Wege bei der Berechnung des Radikals von einem Polynomideal auf, die ebenfalls unterschiedliche Ergebnisse bei den Berechnungen von Beispielen erwarten ließen. Die Implementation der drei Algorithmen gelang jeweils vollständig, so dass die Berechnung von höher-dimensionalen und null-dimensionalen Beispielen mit unterschiedlicher Charakteristik mit den implementierten Algorithmen innerhalb des Computerprogramms CoCoA erfolgen konnte.
Die Analyse der Berechnungsergebnisse stellte deutliche Unterschiede bezüglich der Zuverlässigkeit der Berechnungen des Radikals von einem Polynomideal und der benötigten Dauer für die Berechnung heraus.
Der implementierte Algorithmus von Laplagne ist im Unterschied zu den implementierten Algorithmen von Kemper und Matsumoto in der Lage alle in der vorliegenden Arbeit berechneten Beispiele innerhalb des zeitlichen Limits von dreißig Minuten zu berechnen. Dies gilt sowohl für die Beispiele höherdimensionaler Ideale als auch null-dimensionaler Ideale.
Mit einer Ausfallquote von 30.5 Prozent bei der Berechnung von höherdimensionalen Beispielen ist die Berechnungsmethode von Kemper für das Radikal von einem Polynomideal nicht so zuverlässig wie die Berechnungsmethode von Laplagne. Des Weiteren sind diese Ausfälle nicht vorhersehbar und treten nicht nach einem erklärbaren Schema auf. So können sie sich als Komplett-Ausfall der Berechnung eines Beispieles für jede Charakteristik oder auch nur einer Charakteristik (z.B. 2, 53 oder 251) für ein Beispiel darstellen. Die Beispiele, die von der Berechnungsmethode von Kemper erfolgreich berechnet werden können, liefern das Ergebnis zeitlich schneller oder ähnlich schnell wie die Berechnungsmethode von Laplagne.
Die Vorzüge der Berechnungsmethode von Kemper stellen sich vor allem bei der Berechnung von Beispielen null-dimensionaler Ideale heraus. Hier ist die Ausfallquote der Ergebnisberechnung mit nur 2.0 Prozent viel geringer als bei den höher-dimensionalen Beispielen. Dazu kommt, dass die Berechnungen wie bei den höher-dimensionalen Beispielen sehr schnell erfolgen und den Berechnungsmethoden von Matsumoto und Laplagne bis auf wenige Ausnahmen überlegen ist.
Die Berechnungsmethode von Matsumoto für das Radikal von einem Polynomideal ist mit einer Ausfallquote von 35.2 Prozent für die Berechnung von

höher-dimensionalen Beispielen weniger zuverlässig als die Berechnungsmethode von Kemper. Bei der Betrachtung der nicht durchführbaren Berechnungen wurde deutlich, dass im Unterschied zu Kemper ein Schema erkennbar ist, das darauf hinweist, dass Beispiele mit höherer Charakteristik als 2 häufig mehr Zeit benötigen werden und nicht erfolgreich bis zum Ende berechnet werden können.
Die Methode von Matsumoto stellt ihre Vorzüge jedoch bei der Berechnung der Charakteristik 2 heraus, denn sie kann die Berechnungen jeweils schneller zum Ende führen als die Methoden von Kemper oder Laplagne.
Wenn bei der Berechnung eines Beispiels in jedem Fall ein Ergebnis erzielt werden soll, ist die Berechnungsmethode für das Radikal von einem Polynomideal von Laplagne den Berechnungsmethoden von Kemper und Matsumoto vorzuziehen, da diese Methode in der vorliegenden Arbeit immer ein Ergebnis erzielen konnte. Allerdings benötigt diese Berechnungsmethode im Vergleich zu den beiden anderen Methoden zum Teil etwas mehr Zeit um zu einem Ergebnis zu kommen.
Die dargestellten Vorzüge der einzelnen Berechnungsmethoden zur Berechnung des Radikals von einem Polynomideal stellen heraus, dass keine der drei Methoden sowohl für höher-dimensionale und null-dimensionale Beispiele in allen vorgegebenen Charakteristiken (2, 3, 5, 7, 11, 53, 251) den anderen beiden Methoden im Hinblick auf die Zuverlässigkeit der Berechnung und der geringen Berechnungsdauer deutlich überlegen ist. Im Vergleich zu den anderen Berechnungsmethoden wurde aber deutlich aufgezeigt, dass jede Methode in einem bestimmten Bereich von Beispielen im Bereich der Zuverlässigkeit oder der Berechnungsdauer den anderen überlegen ist.
Weitere Untersuchungen könnten gerade an diesem Punkt ansetzen, denn das Ziel bei der Berechnung eines Radikals von einem Polynomideal ist eine höchstmögliche Zuverlässigkeit in Verbindung mit einer größtmöglichen Schnelligkeit.
Eine Weiterentwicklung der Berechnungsmethode von Kemper sollte vor allem den Bereich der Zuverlässigkeit bei der Berechnung von Beispielen in den Vordergrund stellen, während eine Weiterentwicklung der Berechnungsmethode von Matsumoto sich vorrangig auf die erfolgreiche Berechnung von Beispielen mit einer höheren Charakteristik spezialisieren sollte. Die von Laplagne entwickelte Methode zur Berechnung des Radikals von einem Polynomideal wurde aufgrund ihrer hohen Zuverlässigkeit bereits in anderen mathematischen Zusammenhängen zur Anwendung gebracht. Laplagne verknüpfte dies in [16] zur Berechnung von minimalen assoziierten Primteilern. Dies eröffnet einen Blick darauf, wie vielfältig die Nutzungsmöglichkeiten der Methoden zur Berechnung des Radikals eines Polynomideals sind.

A Timings

In diesem Abschnitt werden die Resultate der Anwendung der Algorithmen auf die Beispiele in mehreren Tabellen dargestellt. Die Tabelle 2 zeigt die Anwendung der Algorithmen auf die höher-dimensionalen Ideale, die in Decker, Greuel und Pfister [5] angegeben sind. Weitere Beispiele mit höher-dimensionalen Idealen, die in Caboara, Conti und Traverso [3] zu finden sind, stehen in Tabelle 1. Die Beispiele mit null-dimensionalen Idealen aus Decker, Greuel und Pfister [5] und Caboara, Conti und Traverso [3] sind in Tabelle 3 aufgelistet.
Die Tabellen sind wie folgt strukturiert:
In der ersten Spalte ist der Name für das Beispiel angegeben, der aus den angegebenen Artikeln übernommen wurde, und darunter steht der Name für die Implementation des Algorithmus, mit dem das Beispiel berechnet wird. So steht *Kemper* für die Implementation des Algorithmus RADIKALKEMPER, *Matsumoto* für RADIKALMATSUMOTO und
Laplagne für RADIKALLAPLAGNE.
Die folgenden Spalten stehen für die Charakteristik des Körpers, mit dem das Beispiel berechnet wird. Die Berechnungen erfolgen in den Charakteristiken 2, 3, 5, 7, 11, 53, 251. Diese Charakteristiken entsprechen den Vorgaben aus den Artikeln von Kemper [11] und Matsumoto [17]. Die Werte in den Tabellen stellen die Zeiten, also die für die Berechnung des Beispiels notwendige Dauer dar, die in der Einheit Sekunden angegeben wird. Der Eintrag $*$ bedeutet, dass die Berechnung nach einer halben Stunde nicht beendet war.

Tabelle 1: Timings der höher-dimensionalen Beispiele aus [3]

Beispiel	Charakteristik						
	2	3	5	7	11	53	251
CCT-L							
Kemper	*	*	*	*	*	*	*
Matsumoto	1017.90	*	*	*	*	*	*
Laplagne	27.45	38.89	70.53	88.57	97.64	107.43	107.53
CCT-M							
Kemper	1.48	1.43	1.31	1.29	1.31	1.31	1.31
Matsumoto	0.73	0.62	0.51	0.46	0.46	0.37	1.12
Laplagne	3.81	110.93	471.09	652.12	626.12	1050.45	1057.65
CCT-Acht3							
Kemper	*	*	*	*	*	*	*
Matsumoto	*	*	*	*	*	*	*
Laplagne	3.95	3.56	3.65	3.71	3.87	3.92	3.85
CCT-C							
Kemper	*	*	*	*	*	*	*
Matsumoto	22.39	*	*	*	*	*	*
Laplagne	17.26	22.85	22.73	22.76	22.67	22.76	22.95
CCT-O							
Kemper	3.20	2.87	2.53	2.57	2.62	2.60	2.65
Matsumoto	0.42	0.29	0.14	0.18	7.87	832.92	*
Laplagne	1.26	1.81	0.11	0.10	0.09	0.09	0.09
Kemper-D							
Kemper	0.65	0.50	0.46	0.50	0.50	0.46	0.48
Matsumoto	0.12	0.03	2.62	*	*	*	*
Laplagne	0.29	0.09	0.11	0.10	0.09	0.09	0.09

Tabelle 2: Timings der höher-dimensionalen Beispiele aus [5]

Beispiel	Charakteristik						
	2	3	5	7	11	53	251
DGP-1							
Kemper	21.17	*	*	*	*	*	*
Matsumoto	1.67	*	*	*	*	*	*
Laplagne	2.82	1272.00	10.40	10.78	11.45	12.06	12.26
DGP-2							
Kemper	*	150.43	155.75	152.70	160.70	161.31	161.15
Matsumoto	0.70	2.37	3.32	5.26	14.54	*	*
Laplagne	0.89	19.40	19.32	19.53	19.32	19.37	19.37
DGP-3							
Kemper	1.07	0.96	0.96	0.98	0.96	0.96	0.96
Matsumoto	0.14	0.12	0.12	0.14	0.11	0.12	1.48
Laplagne	0.84	0.79	0.82	0.79	0.79	0.79	0.79
DGP-4							
Kemper	*	*	*	*	*	*	*
Matsumoto	0.78	4.93	146.87	*	*	*	*
Laplagne	1.09	1.12	1.09	1.09	1.12	1.10	1.09
DGP-5							
Kemper	11.57	11.59	*	*	*	*	*
Matsumoto	0.15	0.37	2.37	6.14	69.34	*	*
Laplagne	0.25	2.50	949.25	717.93	1192.32	1029.25	1002.40
DGP-6							
Kemper	*	362.42	561.53	740.17	798.37	856.98	886.25
Matsumoto	6.18	10.89	35.81	235.81	*	*	*
Laplagne	860.93	26.20	26.51	26.65	26.70	26.70	26.70
DGP-7							
Kemper	50.09	34.89	43.48	46.32	46.00	46.54	46.65
Matsumoto	0.96	0.96	2.62	4.26	7.45	*	*
Laplagne	2.92	7.86	7.00	7.26	7.29	7.34	7.43
DGP-9							
Kemper	2.06	1.62	3.12	2.54	1.92	3.09	3.12
Matsumoto	0.32	0.34	0.25	0.78	0.53	*	*
Laplagne	2.75	3.90	2.67	4.54	3.18	5.14	5.20

Fortsetzung auf der nächsten Seite

Fortsetzung der Timings

DGP-12							
Kemper	3.37	2.95	3.21	2.86	2.81	3.14	3.21
Matsumoto	3.46	13.34	2.18	173.09	*	*	*
Laplagne	8.85	10.20	15.70	9.39	9.42	9.76	9.86
DGP-14							
Kemper	0.42	0.26	0.26	0.26	0.28	0.26	0.26
Matsumoto	0.23	0.09	0.10	0.11	0.09	3.26	*
Laplagne	0.48	0.26	0.26	0.28	0.26	0.26	0.26
DGP-16							
Kemper	67.70	*	*	*	*	*	*
Matsumoto	0.53	*	*	*	*	*	*
Laplagne	1.82	31.50	49.09	52.84	55.51	57.00	57.92
DGP-20							
Kemper	875.51	859.29	998.46	1015.90	1247.29	*	*
Matsumoto	1.92	81.73	*	*	*	*	*
Laplagne	2.36	1.92	1.90	1.92	1.95	1.90	1.92
DGP-21							
Kemper	*	*	*	*	*	*	*
Matsumoto	0.85	0.32	0.29	0.35	0.15	0.17	0.17
Laplagne	0.73	0.75	0.75	0.75	0.75	0.81	0.76
DGP-22							
Kemper	14.82	11.85	7.28	23.85	7.48	8.03	7.87
Matsumoto	0.43	2.17	0.96	0.23	*	*	*
Laplagne	1.01	4.39	4.43	0.76	4.56	4.48	4.50
DGP-23							
Kemper	13.84	<0.01	4.71	4.78	4.82	4.81	5.15
Matsumoto	0.35	0.01	*	*	*	*	*
Laplagne	1.03	0.01	8.61	9.12	9.25	9.46	9.57
DGP-24							
Kemper	*	*	*	*	*	*	*
Matsumoto	0.84	0.59	0.60	0.60	0.62	0.62	0.67
Laplagne	2.59	2.50	2.50	2.54	2.51	2.51	2.48
DGP-25							
Kemper	0.32	0.90	19.31	*	70.29	16.65	*
Matsumoto	0.14	0.14	1.65	166.89	0.31	246.98	*

Fortsetzung auf der nächsten Seite

Fortsetzung der Timings

Laplagne	0.31	0.26	2.51	6.53	2.46	1.82	20.96
DGP-27							
Kemper	10.59	9.43	9.71	9.75	9.84	9.89	9.87
Matsumoto	1.04	42.56	*	*	*	*	*
Laplagne	0.40	0.43	0.42	0.42	0.43	0.42	0.42
DGP-28							
Kemper	*	*	*	*	*	*	*
Matsumoto	8.95	358.34	*	*	*	*	*
Laplagne	0.85	1.12	1.36	1.43	1.40	1.42	1.42
DGP-29							
Kemper	3.54	2.50	4.36	5.67	5.43	7.89	7.92
Matsumoto	1.10	2.29	8.75	214.67	108.90	3.37	*
Laplagne	3.25	2.36	4.62	4.93	5.12	5.82	5.81
DGP-30							
Kemper	80.60	2.68	11.39	11.76	15.95	15.93	16.54
Matsumoto	16.26	9.42	393.93	130.93	1380.43	*	*
Laplagne	1151.89	12.15	29.15	30.04	33.10	34.43	34.21
DGP-31							
Kemper	0.87	0.37	0.37	0.37	0.35	0.39	0.39
Matsumoto	0.18	0.59	*	*	*	*	*
Laplagne	0.59	0.23	0.23	0.23	0.25	0.23	0.25
DGP-32							
Kemper	8.64	8.09	6.81	6.68	6.68	6.73	6.70
Matsumoto	0.84	2.01	1.26	0.87	1.42	87.35	*
Laplagne	7.42	8.15	9.40	9.50	9.51	9.53	9.50
DGP-33							
Kemper	1.73	1.53	1.21	1.26	1.28	1.36	1.32
Matsumoto	0.28	0.17	0.29	0.54	0.39	0.34	0.56
Laplagne	1.35	1.04	1.01	0.98	1.00	0.98	1.00

Tabelle 3: Timinigs der null-dimensionalen Beispiele

Beispiel	Charakteristik						
	2	3	5	7	11	53	251
CCT-E2							
Kemper	0.15	0.20	1.56	0.87	0.76	0.79	0.65
Matsumoto	0.73	*	*	*	*	*	*
Laplagne	0.50	0.46	1.73	1.20	1.46	1.62	1.57
CCT-E3							
Kemper	0.42	0.89	1.00	1.45	1.43	0.89	0.87
Matsumoto	4.48	*	*	*	*	*	*
Laplagne	1.37	5.03	5.21	3.32	3.23	3.62	3.59
DGP-8							
Kemper	0.07	0.17	0.14	0.18	0.15	0.15	0.17
Matsumoto	0.14	29.50	0.40	1.73	10.43	*	*
Laplagne	0.43	3.28	17.81	10.89	7.12	16.01	15.82
DGP-13							
Kemper	0.15	0.18	0.28	0.23	0.20	0.21	0.21
Matsumoto	0.48	1.76	0.29	14.23	0.43	*	*
Laplagne	0.87	1.06	2.01	1.18	1.20	1.40	1.37
DGP-17							
Kemper	2.21	*	8.93	6.37	4.26	3.79	3.89
Matsumoto	13.12	*	*	*	*	*	*
Laplagne	0.76	30.53	1.92	2.15	2.15	2.46	2.46
DGP-18							
Kemper	1.31	134.40	5.46	4.39	11.10	11.81	12.42
Matsumoto	0.12	*	*	*	*	*	*
Laplagne	0.45	5.98	9.98	11.62	13.89	14.37	14.76
DGP-26							
Kemper	0.11	0.09	0.18	0.20	0.20	0.21	0.21
Matsumoto	0.07	0.11	0.81	0.79	0.81	381.51	*
Laplagne	0.21	0.23	1.43	1.61	2.10	2.12	2.18

B Info zur beiliegenden CD-Rom

Dieser Abschnitt dient als Hilfestellung für den Umgang und der Handhabung mit des Inhaltes der CD.
Die CD-Rom beinhaltet:

1. die vorliegende Arbeit als `pdf`-Format.
2. eine `Readme.txt`-Datei, die die Gliederung der CD-Rom enthält.
3. das Package `Radikal.cpkg`, das die implementierten Algorithmen enthält und Informationen zu den Funktionen.
4. das Programm CoCoA, in dem das Package eingesetzt und Berechnungen durchgeführt werden können.
5. eine `Beispiele.coc`-Datei, die alle Beispiele enthält, die in der Arbeit verwendet werden.
6. einen `Ergebnis`-Ordner, der jedem Beispiel eine Datei zuweist,in der das jeweilige Ergebnis der Radikalberechnung und die Timings der Tabellen aus Anhang A zu finden sind.

Die Datei `Radikal.cpkg` ist ein Package, das in das Programm `CoCoA` eingebunden werden kann. Das Package beinhaltet als Definitionen die Implementationen der Algorithmen RADIKALKEMPER,
RADIKALMATSUMOTO und RADIKALLAPLAGNE. Mit diesem Package ist es möglich das Radikal eines Ideals entsprechend dieser drei Algorithmen zu berechnen. Die Funktionen des Package können mit den Beispielen der Datei `Beispiele.coc` getestet werden. Dazu müssen die Beispiele bestimmte Vorgaben erfüllen, die in der Syntax dargestellt sind. Die verwendeten Beispiele entsprechen diesen Vorgaben. Wenn ein Beispiel selbst konstruiert werden soll, dann müssen diese Vorgaben eingehalten werden, um eine erfolgreiche Berechnung des Radikals von einem Polynomideal erhalten zu können. Die Beispiele und ihre Timings sind in den Tabellen des folgenden Abschnittes aufgelistet. Der Ordner `Ergebnis` enthält unter dem Namen des Beispiels als `.coc`-Datei für das jeweilige Beispiel das Ergebnis der Radikalberechnung in den Charakteristiken 2, 3, 5, 7, 11, 53 und 251 mit Hilfe der drei Algorithmen und die Timings, also die Angabe der Dauer der Berechnung.
Es folgt die Beschreibung der Funktionen entsprechend der Formatierung aus dem CoCoA-Manual.

```
SYNTAX       --Radikalkemper--
     RadikalKemper(I: IDEAL) : IDEAL

BESCHREIBUNG
Diese Funktion berechnet das Radikal eines Ideals in einem Polynomring
mit positiver Charakteristik nach dem Algorithmus im Artikel

G.Kemper:
  The Calculation of Radical Ideals in Positive Characteristic,
  Journal of Symbolic Computation, (34), Seite 229-238, 2002.

BEISPIELE
Use R::=Z/(2)[x[1..4]];
I:=Ideal(x[2]+x[3],x[1]x[3]^2x[4],x[1]^2x[3]^2);
RadikalKemper(I);
Ideal(x[2] + x[3], x[1]x[3])
------------------------------
Use R::=Z/(7)[x[1..4]];
B:=[x[4]^7-x[2]x[3]x[1]^5,x[3]^4-x[2]^3x[1]];
I:=Ideal(B);
RadikalKemper(I);
Ideal(x[1]x[2] - x[3]x[4],
      x[3]^3 - x[2]^2x[4],
      x[1]x[3]^2 - x[2]x[4]^2,
      x[1]^2x[3] - x[4]^3)
------------------------------

-------------------------------------------------------------------------

SYNTAX       --RadikalMatsumoto--
RadikalMatsumoto(B : LIST) : LIST

BESCHREIBUNG
Die Funktion berechnet das Radikal des von der Gröbnerbasis B
erzeugten Ideals in einem Polynomring der Charakteristik P>0 nach
dem Algorithmus, der im Artikel

R.Matsumoto:
  Computing the Radical of an Ideal in Positive Charakteristic,
  Journal of Symbolic Computation, (32), Seite 263-271, 2001.

beschrieben wird.
```

```
BEISPIELE
Use R::=Z/(7)[x[1..4]];
B:=[x[4]^7-x[2]x[3]x[1]^5,x[3]^4-x[2]^3x[1]];
I:=Ideal(B);
RadikalMatsumoto(B);
[x[1]^2x[3] - x[4]^3, x[1]x[3]^2 - x[2]x[4]^2,
-x[1]x[2] + x[3]x[4], -x[3]^3 + x[2]^2x[4]]
-------------------------------
Use R::=Z/(2)[x[1..4]];
I:=Ideal(x[2]+x[3],x[1]x[3]^2x[4],x[1]^2x[3]^2);
GB:=GBasis(I);
RadikalMatsumoto(GB);
[x[2] + x[3], x[1]x[3]]
-----------------------------------------------------------------------

SYNTAX       --RadikalLaplagne--
RadikalLaplagne(I : Ideal) : IDEAL

BESCHREIBUNG
Die Funktion berechnet das Radikal von einem Ideal I in einem
Polynomring nach einem Algorithmus im Artikel

S.Laplagne:
  An Algorithm for the Computation of the Radical of an Ideal,
  In: ISAAC'06: Proceedings of the 2006 international symposium on
    Symbolic and algebraic computation, New York, NY, USA,
    ACM Press (2006), Seite 191-195, 2006.

BEISPIELE
Use R::=Z/(2)[x[1..4]];
I:=Ideal(x[2]+x[3],x[1]x[3]^2x[4],x[1]^2x[3]^2);
RadikalLaplagne(I);
Ideal(x[2] + x[3], x[1]x[3])
-------------------------------
Use R::=Z/(7)[x[1..4]];
B:=[x[4]^7-x[2]x[3]x[1]^5,x[3]^4-x[2]^3x[1]];
I:=Ideal(B);
Rad.RadikalLaplagne(I);
Ideal(x[1]x[2] - x[3]x[4], x[1]^2x[3] - x[4]^3,
      x[1]x[3]^2 - x[2]x[4]^2, x[3]^3 - x[2]^2x[4])
-------------------------------

-----------------------------------------------------------------------
```

```
--  Hilfsfunktionen

SYNTAX    -- Sep, Sep0 --
Sep(F: POLY, X: POLY, Ri: INT): LIST
Sep0(F: POLY,X: POLY): POLY

BESCHREIBUNG
Diese Funktionen berechnen den separablen Teil eines Polynoms
nach einem Algorithmus im Artikel von

G.Kemper:
  The Calculation of Radical Ideals in Positive Characteristic,
  Journal of Symbolic Computation, (34), Seite 229-238, 2002.

Im Input von Sep  ist F das Polynom, dessen separabler Teil berechnet
wird, X ist eine Varible und Ri gibt die Anzahl der Durchläufe an.
Der Output ist eine Liste, deren erstes Element ein Polynom ist und
deren zweites Element eine ganze Zahl ist.
Beachte: Die Funktion Sep  wird in der Funktion RadikalKemper und
  RadikalLaplagne im Fall höher-dim. Ideale verwendet.
  Die Funktion Sep0 wird in der Funktion Radikalkemper und
  RadikalLaplagne im Fall null-dim. Ideale verwendet.

BEISPIELE
Use R::=Z/(5)[ t[1..1], x[1] ];
F:=x[1]^25  - t[1];
Sep(F,x[1],0);
[-t[1] + x[1], 2]
-------------------------------
-------------------------------
Use R::=Z/(5)[  x[1] ];
F:=x[1]^5- 1;
Sep0(F,x[1]);
x[1] - 1
-------------------------------

SYNTAX     -- MaxIndepSet, IndepSet --
MaxIndepSet(I: IDEAL): LIST

BESCHREIBUNG
Die Funktion MaxIndepSet berechnet eine maximal unabhängige Menge
modulo eines Ideals I nach dem Algorithmus
```

```
in Kreuzer und Robbiano: Computational Commutative Algebra 2,
     Korollar 5.7.10.

BEISPIELE
Use R::=Q[x,y,z], DegRevLex;
I:=Ideal(xy^2 -x, y^2z -z );
MaxIndepSet(I);
[x, z]
-------------------------------
I:=Ideal(x^2+y^2+z^2);
MaxIndepSet(I);
[y, z]
-------------------------------
Use R::=Q[x,y,z], DegRevLex ;
Y:=[x,y,z];
T:=[xy, yz];
IndepSet(Y,T);
T:=[x];
IndepSet(Y,T);
[x, z]
-------------------------------
[y, z]
-------------------------------
```

Literatur

[1] Armendáriz, Inés, Solernó, Pablo: *On the Computation of the Radical of Polynomial Complete Intersection Ideals.* In Cohen, G. et al. eds, Proc. AAECC-11, Lecture Notes in Computer Science 948, 106-119, 1995.

[2] Becker, Thomas und Weispfenning, Volker: *Gröbner Bases.* Springer-Verlag, Berlin, Heidelberg, New York, 1993.

[3] Caboara, Massimo, Conti, Pasqualina und Traverso, Carlo: *Yet Another Ideal Decomposition Algorithm.* In Matteson, H., Mora, T. eds., Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC-12), Lecture Notes in Computer Science 1255, 39-54, Springer Verlag, Berlin, Heidelberg, New York, 1997.

[4] Cox, David, Little, John und O'Shea, Donal: *Ideals, Varieties, and Algorithms.* Springer-Verlag, New York, 1996.

[5] Decker, W., Greuel, G.-M. und Pfister, G.: *Primary decomposition: Algorithms and comparisions.* Algorithmic algebra and number theory, Springer Verlag, Heidelberg, 187-220, 1998.

[6] Eisenbud, David, Huneke, Craig und Vasconcelos, Wolmer V.: *Direct Methods for primary decomposition.* Invent. Math. 110, 207-235, 1992.

[7] Faugère,J. C., Gianni, P., Lazard, D. und Mora, T.: *Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering.* Journal of Symbolic Computation (16), 329-344, 1993.

[8] Gianni, Patrizia, Mora, Teo: *Algebraic Solution of Asystems of Polynomial Equations Using Groebner Bases.* Lect. Notes Comput. Sci. 356, Springer-Verlag, Berlin, Heidelberg, New York, 247-257, 1989.

[9] Gianni, Patrizia, Trager, Barry und Zacharias, Gail: *Gröbner Bases and Primary Decomposition of Polynomial Ideals.* Journal of Symobolic Computation 6, 149-167, 1988.

[10] Jong, Theo de: *An algorithm for computing the integral closure.* Journal of Symbolic Computation 26, 273-277, 1998.

[11] Kemper, Gregor: *The Calculation of Radical Ideals in Positive Characteristic.* Journal of Symbolic Computation, (34), 229-238, 2002.

[12] Kreuzer, Martin und Robbiano, Lorenzo: *Computational Commutative Algebra 1.* Springer-Verlag, Berlin, 2000.

[13] Kreuzer, Martin und Robbiano, Lorenzo: *Computational Commutative Algebra 2.* Springer-Verlag, Berlin, 2005.

[14] Krick, Teresa und Logar, Alessandro: *An Algorithm for the Computation of the Radical of an Ideal in the Ring of Polynomials.* AAECC9, Springer LNCS (539), 195-205, 1991.

[15] Laplagne, Santiago: *An Algorithm for the Computation of the Radical of an Ideal.* In: ISAAC'06: Proceedings of the 2006 international symposium on Symbolic and algebraic computation, New Yor, NY, USA, ACM Press (2006), 191-195, 2006.

[16] Laplagne, Santiago: *Computation of the Minimal Associated Primes.* In: Decker, W. et al., Dagstuhl Seminar Proceedings (06271), Challenges in Symbolic Computation Software, 2006.

[17] Matsumoto, Ryutaroh: *Computing the Radical of an Ideal in Positive Characteristic.* Journal of Symbolic Computation (32), 263-271, 2001.

[18] Meyberg, Kurt: *Algebra Teil 2.* Carl Hanser Verlag, München, Wien, 1976.

[19] Seidenberg, A.: *Constructions in algebra.* Trans. Amer. Math. Soc. 197, 273-313, 1974.

[20] Wang, Dongming: *An Elimination Method for Polynomial Systems.* Journal of Symbolic Computation 16, 83-114, 1993.

[21] Wang, Dongming: *Unmixed and Prime Decomposition of Radicals of Polynomial Ideals.* SIGSAM Bull., 32(4), 2-9, 1998.