Primary Decomposition of Modules

Diploma Thesis
Department of Mathematics
University of Kaiserslautern

Alexander Dreyer

Kaiserslautern
May 2001

Contents

Acknowledgements

This page is dedicated to all the people, who supported me in doing this diploma thesis. First I would like to thank Prof. Dr. Gerhard Pfister for introducing me to this very interesting problem and Dr. Anne Frühbis-Krüger and Thomas Keilen, my tutors in Algebraic Geometry and Computer Algebra, as well as many of my fellow students, in particular Tobias Hirsch, for many fruitful discussions. Apart from the mathematical aspect I am grateful to Hans Schönemann and the SINGULAR-Team for their assistance.

Finally, I want to express my gratitude to my family and my girlfriend Irina Feuerbach for all their love and patience, and for their support during my studies.

Introduction

The primary decomposition of modules is a generalization of the well-known primary decomposition of ideals. To obtain a primary decomposition of a module $\mathbf{N}$ we compute modules containing $\mathbf{N}$ with certain properties—the primary components—such that the intersection of these is equal to $\mathbf{N}$. Geometrically the decomposition of an affine variety into irreducible components corresponds to the primary decomposition of radical ideals. For arbitrary ideals we may obtain embedded components, which are invisible in the decomposition of the corresponding varieties itself, but responsible for many geometric properties, in particular, if we deform the variety slightly. (To make these components "visible", one can use the structure of *affine schemes* instead of affine varieties, cf. [EH, Chapter 2]). Even though there isn't any geometrical interpretation of this kind for modules, it is often advantageous to work with modules instead of ideals.

In Chapter 1 the theory of the primary decomposition of ideals in a ring is generalized to the case of a submodule $\mathbf{N}$ of a finitely generated $\mathbf{R}$-module $\mathbf{M}$. To explain the mathematical background we will follow Eisenbud's approach for associated primes and primary decomposition [Eis, Chapter 3]. Based on this the existence of the decomposition and the Uniqueness Theorems are discussed. As mentioned in [Grä] we will use quotient separation to separate primary components. We will also reduce the problem to the zero-dimensional case. These tools will be used in the formulation of the algorithms of Chapter 2. We will also see that the isolated primes of the module $\mathbf{N}$ are equal to the isolated primes of its annihilator $\operatorname{Ann} \mathbf{M}/\mathbf{N}$. This is the major result of this chapter.

In the second chapter two algorithms for the primary decomposition of submodules of a finitely generated $\mathbb{K}[x_1, \ldots, x_n]$-module are discussed. The first one is a straight forward generalization of the algorithm given by Gianni, Trager and Zacharias [GTZ], the other was suggested by Gräbe [Grä] and is based on the ideas of Shimoyama and Yokoyama [SY]. Furthermore, elementary implementations in the computer algebra system SINGULAR are presented.

Finally, the last chapter describes various improvements of the algorithm based on [GTZ] and gives a comparision of different implementations. The

effects of these enhancements will be discussed on the basis of a series of examples.

Although the additions decribed in Chapter 3 make the GTZ-algorithm really efficient, there are still examples, which can be faster computed using the algorithm based on [SY].

All procedures, mentioned in this thesis, are part of the SINGULAR-library `modec.lib`, which is included in the appendix.

Chapter 1

Mathematical Background

1.1 Associated Primes

The associated primes of an ideal are important in the construction and computation of the primary decomposition of ideals. We are going to introduce a similar notation for modules. Following Eisenbud's approach (cf. [Eis, Chapter 3]) we first show a few important facts about associated primes.

Let $\mathbf{R}$ be a ring and $\mathbf{M}$ be an $\mathbf{R}$-module.

Definition 1.1.1 A prime $\mathfrak{p}$ of $\mathbf{R}$ is *associated* to $\mathbf{M}$ if $\mathfrak{p} = \mathrm{Ann}_{\mathbf{R}}(x)$ for some $x \in \mathbf{M}$. We denote the set of all primes associated to $\mathbf{M}$ by $\mathrm{Ass}_{\mathbf{R}}\,\mathbf{M}$ or simply $\mathrm{Ass}\,\mathbf{M}$ when there can be no confusion about $\mathbf{R}$.

Remark Note that in the case of ideals there is one exception of this notation: Let $\mathbf{I}$ be an ideal of $\mathbf{R}$. Then the associated primes of the module $\mathbf{R}/\mathbf{I}$ are called *associated primes of* $\mathbf{I}$. (Usually the associated primes of $\mathbf{I}$ considered as an $\mathbf{R}$-module are not interesting.) Later in this chapter we will compute the primary decomposition of a submodule $\mathbf{N} \subseteq \mathbf{M}$ using $\mathrm{Ass}\,\mathbf{M}/\mathbf{N}$. By abuse of language we may also call $\mathrm{Ass}\,\mathbf{M}/\mathbf{N}$ in that case the *associated primes of* $\mathbf{N}$.

Proposition 1.1.2 *Let* $\mathbf{M}$ *be a nonzero* $\mathbf{R}$*-module. If* $\mathbf{I} \subseteq \mathbf{R}$ *is an ideal, which is maximal among all ideals that are annihilators of elements of* $\mathbf{M}$*, then* $\mathbf{I}$ *is prime (and thus belongs to* $\mathrm{Ass}\,\mathbf{M}$*).*

Proof: If $rs \in \mathbf{I}$ but $s \notin \mathbf{I}$, then we have to show that $r \in \mathbf{I}$. Let $m \in \mathbf{M}$ such that $\mathrm{Ann}(m) = \mathbf{I}$. Then $rsm = 0$ but $sm \neq 0$. Therefore $(r, \mathbf{I}) \subseteq \mathrm{Ann}(sm)$. Since $\mathbf{I}$ was maximal among the ideals of the form $\mathrm{Ann}(m)$, $(r, \mathbf{I}) = \mathbf{I}$. Thus $r \in \mathbf{I}$. □

In Noetherian rings we always can find ideals as decribed in Proposition 1.1.2, because the ascending chain condition holds. This condition is no re-

striction at all, since all examples of rings we are interested in are Noetherian (e. g. polynomial rings over fields). We formulate this fact as:

Corollary 1.1.3 *If* $\mathbf{R}$ *is a Noetherian ring and* $\mathbf{M} \neq (0)$*, then* $\operatorname{Ass}\mathbf{M}$ *is nonempty.*

From now on we will assume that $\mathbf{R}$ is a Noetherian ring and let $\mathbf{M}$ be a finitely generated nonzero $\mathbf{R}$-module.

Proposition 1.1.4 *The set* $\bigcup_{\{\mathfrak{p}\in\operatorname{Ass}\mathbf{M}\}}\mathfrak{p}$ *consists of 0 and the set of zerodivisors on* $\mathbf{M}$*.*

Proof: The elements of the annihilator of an element in $\mathbf{M}$ are either zero or zerodivisors of $\mathbf{M}$. Conversely, if r annihilates a nonzero element m of $\mathbf{M}$, then $r \in \operatorname{Ann}(m)$. But we can choose $n \in \mathbf{M}$ such that $\operatorname{Ann}(m) \subseteq \operatorname{Ann}(n)$ and $\operatorname{Ann}(n)$ is maximal with this condition. By Proposition 1.1.2 $\operatorname{Ann}(n) \in \operatorname{Ass}\mathbf{M}$. □

Analogous to the case of ideals there are only a finite number of associated primes. To prove this we need the following two lemmas:

Lemma 1.1.5 *If* $0 \to \mathbf{M}' \to \mathbf{M} \to \mathbf{M}'' \to 0$ *is a short exact sequence of* $\mathbf{R}$*-modules, then* $\operatorname{Ass}\mathbf{M}' \subseteq \operatorname{Ass}\mathbf{M} \subseteq \operatorname{Ass}\mathbf{M}' \cup \operatorname{Ass}\mathbf{M}''$*.*

Proof: The first inclusion is clear from the definition. To show the second, suppose that $\mathfrak{p} \in \operatorname{Ass}\mathbf{M}\setminus\operatorname{Ass}\mathbf{M}'$ and let $x \in \mathbf{M}$ such that $\mathfrak{p} = \operatorname{Ann}(x)$, so that $\mathbf{R}x \cong \mathbf{R}/\mathfrak{p}$. Since $\mathbf{R}/\mathfrak{p}$ is an integral domain, nonzero elements can only be annihilated by $\mathfrak{p}$. Hence every nonzero submodule of $\mathbf{R}x$ has annihilator $\mathfrak{p}$. But $\mathfrak{p} \notin \operatorname{Ass}\mathbf{M}'$ and therefore $\mathfrak{p}$ is not annihilator of nonzero elements of $\mathbf{R}x \cap \mathbf{M}'$. So we have $\mathbf{R}x \cap \mathbf{M}' = 0$ and we may assume that $\mathbf{R}x$ is isomorphic to its image in $\mathbf{M}''$. Thus $\mathfrak{p} \in \operatorname{Ass}\mathbf{M}''$ as required. □

Lemma 1.1.6 *For every finitely generated module* $\mathbf{M}$ *exists a chain*

$$0 = \mathbf{M}_0 \subseteq \mathbf{M}_1 \subseteq \cdots \subseteq \mathbf{M}_n = \mathbf{M}$$

of submodules of $\mathbf{M}$*(*a filtration of $\mathbf{M}$*), with* $\mathbf{M}_{i+1}/\mathbf{M}_i \cong \mathbf{R}/\mathfrak{p}_i$ *for some prime ideal* $\mathfrak{p}_i$*. Furthermore* $\operatorname{Ass}\mathbf{M} = \{\mathfrak{p}_i : i = 1, \ldots, n-1\}$*.*

Proof: If $\mathbf{M}$ is the zeromodule, then $\mathbf{M}_0 = \mathbf{M}_n$ and the statement is obvious. If $\mathbf{M} \neq 0$ then $\mathfrak{p}_1 \in \operatorname{Ass}\mathbf{M}$ exists by Proposition 1.1.2, so that there is a submodule $\mathbf{M}_1 \cong \mathbf{R}/\mathfrak{p}_1$. We can repeat this procedure with $\mathbf{M}/\mathbf{M}_1$. So we obtain $\mathbf{M}_2 \cong \mathbf{R}/\mathfrak{p}_2$ and so on. Since $\mathbf{R}$ is Noetherian and $\mathbf{M}$ is finitely generated, all $\mathbf{M}_i$ are Noetherian modules. Hence the chain $\mathbf{M}_1 \subseteq \mathbf{M}_2 \subseteq \cdots$ stabilizes and this means that some $\mathbf{M}_n = \mathbf{M}$, as required.

To prove the last statement we have a look at the $\mathfrak{p}_i$. By construction and Lemma 1.1.5 we have $\mathfrak{p}_i \in \operatorname{Ass}\mathbf{M}$. Assume there exists an associated

prime $\mathfrak{p}$, which does not appear among the $\mathfrak{p}_i$. But this would mean, that $\mathfrak{p} \notin \operatorname{Ass} \mathbf{M}/\mathbf{M}_{n-1}$ (otherwise it would be among the $\mathfrak{p}_i$, i=1, ... , n-1) and $\mathfrak{p} \notin \operatorname{Ass} \mathbf{M}_{n-1}$ (since $\mathbf{M}_i$ is a submodule of $\mathbf{M}_{n-1}$ for $i = 1, \ldots, n-1$). Hence by Lemma 1.1.5 $\mathfrak{p} \notin \operatorname{Ass} \mathbf{M}$ which is a contradiction. □

Proposition 1.1.7 $\operatorname{Ass} \mathbf{M}$ *is a finite, nonempty set of primes, each containing* $\operatorname{Ann} \mathbf{M}$.

Proof: The finiteness statement follows by Lemma 1.1.6 and the existence of associated primes by Corollary 1.1.3. If $\mathfrak{p}$ is an associated prime, we can find $m \in \mathbf{M}$ such that $\mathfrak{p} = \operatorname{Ann}(m)$ and $\operatorname{Ann} \mathbf{M} \subseteq \operatorname{Ann}(m)$. □

One important tools in Commutative Algebra is the localization. Thus, we are interested in a 'nice' relationship between the associated primes of a module $\mathbf{M}$ and its extension. The next theorem shows that the formation of the set $\operatorname{Ass} \mathbf{M}$ commutes with localization in some sense.(cf. [Eis, Theorem 3.1])

Theorem 1.1.8 *Let* $\mathbf{M}$ *be a nonzero finitely generated* $\mathbf{R}$*-module and* σ *be an arbitrary multiplicatively closed set. Then*

$$\operatorname{Ass}_{\mathbf{R}[\sigma^{-1}]} \mathbf{M}[\sigma^{-1}] = \{\mathfrak{p}\mathbf{R}[\sigma^{-1}] : \mathfrak{p} \in \operatorname{Ass}_{\mathbf{R}} \mathbf{M} \text{ and } \mathfrak{p} \cap \sigma = \emptyset\}.$$

Proof: If $\mathfrak{p} \in \operatorname{Ass}_{\mathbf{R}} \mathbf{M}$, then there is an inclusion $\mathbf{R}/\mathfrak{p} \hookrightarrow \mathbf{M}$. Localizing, we get an injection $\mathbf{R}[\sigma^{-1}]/\mathfrak{p}\mathbf{R}[\sigma^{-1}] \hookrightarrow \mathbf{M}[\sigma^{-1}]$. We know (e. g. from [AM, Proposition 3.11]) that $\mathfrak{p}\mathbf{R}[\sigma^{-1}]$ is a prime ideal of $\mathbf{R}[\sigma^{-1}]$ if $\mathfrak{p}$ doesn't meet σ. In this case $\mathfrak{p}\mathbf{R}[\sigma^{-1}]$ is the annihilator of an element in $\mathbf{M}[\sigma^{-1}]$ and therefore we have $\mathfrak{p}\mathbf{R}[\sigma^{-1}] \in \operatorname{Ass} \mathbf{M}_{\mathbf{R}[\sigma^{-1}]}[\sigma^{-1}]$.

Conversely, suppose $\mathfrak{q}$ is a prime of $\mathbf{R}[\sigma^{-1}]$ that is associated to $\mathbf{M}[\sigma^{-1}]$. Since $\mathfrak{q}$ is an extended (prime) ideal, we can write $\mathfrak{q} = \mathfrak{p}\mathbf{R}[\sigma^{-1}]$ with $\mathfrak{p}$ a prime of $\mathbf{R}$ and $\mathfrak{p} \cap \sigma = \emptyset$. There is an injection $\varphi : \mathbf{R}[\sigma^{-1}]/\mathfrak{p}\mathbf{R}[\sigma^{-1}] \hookrightarrow \mathbf{M}[\sigma^{-1}]$. Since $\mathfrak{p}$ is finitely generated, we know from Commutative Algebra (cf. [Eis, Proposition 2.10])

$$\operatorname{Hom}_{\mathbf{R}[\sigma^{-1}]} \left(\mathbf{R}[\sigma^{-1}]/\mathfrak{p}\mathbf{R}[\sigma^{-1}], \mathbf{M}[\sigma^{-1}]\right) = \operatorname{Hom}_{\mathbf{R}}(\mathbf{R}/\mathfrak{p}, \mathbf{M})[\sigma^{-1}].$$

So we may write $\varphi = s^{-1}f$ for some $f \in \operatorname{Hom}_{\mathbf{R}}(\mathbf{R}/\mathfrak{p}, \mathbf{M})$ and $s \in \sigma$. Since $s \notin \mathfrak{p}$, it is a nonzerodivisor on $\mathbf{R}/\mathfrak{p}$. Hence f is an injection, concluding the proof. □

Proposition 1.1.9 *The set* $\operatorname{Ass} \mathbf{M}$ *includes all primes minimal among primes containing* $\operatorname{Ann} \mathbf{M}$.

Proof: Let $\mathfrak{p}$ be minimal over $\operatorname{Ann} \mathbf{M}$. By Theorem 1.1.8, the localization at $\mathfrak{p}$ commutes with the formation of the associated primes. From Proposition 1.1.2 we know that $\operatorname{Ass}(\mathbf{M}_{\mathfrak{p}}) \neq \emptyset$. Hence there exists an ideal

$\mathfrak{q}\mathbf{R}_\mathfrak{p} \in \operatorname{Ass}(\mathbf{M}_\mathfrak{p})$ and by Proposition 1.1.7 $\mathfrak{q}\mathbf{R}_\mathfrak{p} \supseteq \operatorname{Ann}\mathbf{M}_\mathfrak{p} = (\operatorname{Ann}\mathbf{M})_\mathfrak{p}$. Since $\mathfrak{p}\mathbf{R}_\mathfrak{p}$ is the the maximal ideal of $\mathbf{R}_\mathfrak{p}$ we have $\mathfrak{p}\mathbf{R}_\mathfrak{p} \supseteq \mathfrak{q}\mathbf{R}_\mathfrak{p}$ and therefore $\mathfrak{p} \supseteq \mathfrak{q} \supseteq \operatorname{Ann}\mathbf{M}$. Thus $\mathfrak{p} = \mathfrak{q} \in \operatorname{Ass}\mathbf{M}$ by the minimality of $\mathfrak{p}$. □

1.2 Primary Decomposition

Although the primary decomposition of ideals is well-known, there is only few known about the primary decomposition of arbitrary modules. First we are going to define what a *primary submodule* is (cf. Gräbe [Grä] and Greuel and Pfister [GP, Chapter 2]). The first major results are the existence of the primary decomposition and the uniqueness theorems (cf. Eisenbud [Eis, Chapter 3]).

Let $\mathbf{R}$ be a ring and $\mathbf{M}$ be an $\mathbf{R}$-module. Furthermore, we will assume that $\mathbf{N}$ is a submodule of $\mathbf{M}$.

Definition 1.2.1 $\mathbf{N}$ is a *primary submodule* of $\mathbf{M}$ if $\mathbf{M}/\mathbf{N} \neq 0$ and every zerodivisor of $\mathbf{M}/\mathbf{N}$ is already nilpotent on $\mathbf{M}/\mathbf{N}$. In this case $\operatorname{Ann}\mathbf{M}/\mathbf{N}$ is a primary ideal and we call $\mathbf{N}$ a $\mathfrak{p}$*-primary submodule of* $\mathbf{M}$, where $\mathfrak{p} = \sqrt{\operatorname{Ann}\mathbf{M}/\mathbf{N}}$.

Remark $\mathbf{N}$ is $\mathfrak{p}$-primary if and only if $\operatorname{Ass}_\mathbf{R}\mathbf{M}/\mathbf{N} = \{\mathfrak{p}\}$.

Proposition 1.2.2 *Let* $\mathbf{N}_1, \ldots, \mathbf{N}_m$ *be* $\mathfrak{p}$*-primary submodules of* $\mathbf{M}$*. Then* $\mathbf{N} = \bigcap_{i=1}^m \mathbf{N}_i$ *is a* $\mathfrak{p}$*-primary submodule of* $\mathbf{M}$*.*

Proof: $\mathbf{N}$ is a proper submodule of $\mathbf{M}$, because every single $\mathbf{N}_i$ is. Let x be a zerodivisor of $\mathbf{M}/\mathbf{N}$. Then x is also a zerodivisor of $\mathbf{M}/\mathbf{N}_i$ for all i. Since all $\mathbf{N}_i$ are primary, x is nilpotent in all $\mathbf{M}/\mathbf{N}_i$, hence it is in $\mathbf{M}/\mathbf{N}$. □

Now we define the primary decomposition of modules analogously to the special case of ideals. We will assume in the following that $\mathbf{R}$ is a Noetherian ring and $\mathbf{M}$ is a finitely generated $\mathbf{R}$-module.

Definition 1.2.3 A *primary decomposition* of a submodule $\mathbf{N} \in \mathbf{M}$ is a representation of $\mathbf{N}$ as intersection of finitely many primary submodules of $\mathbf{M}$:

$$\mathbf{N} = \mathbf{N}_1 \cap \cdots \cap \mathbf{N}_m$$

with $\mathfrak{p}_i$-primary submodules $\mathbf{N}_i \subseteq \mathbf{M}$ for $i = 1, \ldots, m$.

We call a primary decomposition *minimal* if:

- $\mathfrak{p}_1, \ldots, \mathfrak{p}_m$ are pairwise distinct and
- $\mathbf{N}_j \not\supseteq \bigcap_{i \neq j} \mathbf{N}_i$ for all $j = 1, \ldots, m$.

To prove the existence of this decomposition, we first show a slightly finer decomposition.

Definition 1.2.4 Let $\mathbf{N}$ be a submodule of $\mathbf{M}$. We call $\mathbf{N}$ *irreducible* if it is not the intersection of two strictly larger submodules of $\mathbf{M}$.

Theorem 1.2.5 *Let $\mathbf{N}$ be a submodule of $\mathbf{M}$.*

Then there exist $\mathbf{N}_i$, $i = 1, \ldots, m$, where the $\mathbf{N}_i$ are irreducible submodules of $\mathbf{M}$ and $\mathbf{N} = \bigcap_{i=1}^{m} \mathbf{N}_i$.

Proof: Suppose there exists a submodule $\mathbf{N} \in \mathbf{M}$, which is not a finite intersection of irreducible submodules of $\mathbf{M}$. We could choose $\mathbf{N}$ maximal among those modules by the ascending chain condition. (Because we are dealing with finitely generated modules.) In particular $\mathbf{N}$ is not irreducible and therefore there exist $\mathbf{N}_1$, $\mathbf{N}_2$ strictly containing $\mathbf{N}$ such that $\mathbf{N} = \mathbf{N}_1 \cap \mathbf{N}_2$. Hence each of $\mathbf{N}_1$ and $\mathbf{N}_2$ is a finite intersection of irreducible submodules:

$$\mathbf{N}_1 = \bigcap \mathbf{N}_{1_i} \text{ and } \mathbf{N}_2 = \bigcap \mathbf{N}_{2_j}$$

Therefore we have $\mathbf{N} = \bigcap \mathbf{N}_{1_i} \cap \bigcap \mathbf{N}_{2_j}$. But this is a contradiction to the choice of $\mathbf{N}$. Thus the irreducible decomposition exists. □

Lemma 1.2.6 *Every irreducible submodule $\mathbf{N} \in \mathbf{M}$ is primary.*

Proof: If $\mathbf{N}$ is irreducible but not primary, there exist at least two different associated primes $\mathfrak{p} \neq \mathfrak{q}$, such that $\mathfrak{p} = \mathrm{Ann}(x)$, $\mathfrak{q} = \mathrm{Ann}(y)$ for some $x, y \in \mathbf{M}/\mathbf{N}$. Hence $\mathbf{R}x \cong \mathbf{R}/\mathfrak{p}$ and $\mathbf{R}y \cong \mathbf{R}/\mathfrak{q}$ and therefore they are integral domains.

We now claim that $\mathbf{R}x \cap \mathbf{R}y = 0$. To show this we assume the opposite. If $\mathbf{R}x \cap \mathbf{R}y \ni rx = sy \neq 0$ for suitable $r, s \in \mathbf{R}$, then $\exists p \in \mathfrak{p}, p \notin \mathfrak{q}$ and we have $(rx)p = r(xp) = 0$ in $\mathbf{R}/\mathfrak{p}$. But $(sy)p \neq 0$ in $\mathbf{R}/\mathfrak{q}$, because $sy, p \neq 0$ in $\mathbf{R}/\mathfrak{q}$. Contradiction.

Since $\mathbf{R}x, \mathbf{R}y \neq 0$, the zero-module is not irreducible in $\mathbf{M}/\mathbf{N}$. If we take $\mathbf{N}_1$, $\mathbf{N}_2$ as the preimages of $\mathbf{R}x$ resp. $\mathbf{R}y$ in $\mathbf{M}$, then $\mathbf{N} = \mathbf{N}_1 \cap \mathbf{N}_2$ and $\mathbf{N}_1, \mathbf{N}_2 \neq \mathbf{N}$. This is indeed a contradiction and therefore $\mathbf{N}$ is primary. □

Immediately, we get:

Proposition 1.2.7 (The Existence of the Primary Decomposition)
Let $\mathbf{N}$ be a submodule of $\mathbf{M}$. Then there exists a primary decomposition $\mathbf{N} = \mathbf{N}_1 \cap \cdots \cap \mathbf{N}_m$ with $\mathfrak{p}_i$-primary submodules $\mathbf{N}_i \subseteq \mathbf{M}$ for $i = 1, \ldots, m$.

Proof: The decomposition of Theorem 1.2.5 is a primary decomposition by Lemma 1.2.6. □

Now we prove a few theorems, which are straight-forward generalizations of the corresponding theorems from the case of ideals. (cf. Atiyah and MacDonald [AM, Chapter 4] for ideals).

Theorem 1.2.8 (First Uniqueness Theorem) *Let* $\mathbf{N} = \mathbf{N}_1 \cap \cdots \cap \mathbf{N}_m$ *be a minimal primary decomposition, where the* $\mathbf{N}_i$ *are* $\mathfrak{p}_i$*-primary submodules.*

Then $\operatorname{Ass} \mathbf{M}/\mathbf{N} = \{\mathfrak{p}_i : i = 1, \ldots, m\}$ *and hence the* $\mathfrak{p}_i$ *are uniquely determined.*

Proof: Let $\mathfrak{p}$ be in $\operatorname{Ass} \mathbf{M}/\mathbf{N}$ and consider the map

$$\varphi : \mathbf{M}/\mathbf{N} \to \bigoplus_{i=1}^{m} \mathbf{M}/\mathbf{N}_i$$

defined by $m \mapsto (m + \mathbf{N}_1, \ldots, m + \mathbf{N}_m)$. φ is injective because $m \in \mathbf{N}_i, \forall i$ if and only if $m \in \bigcap \mathbf{N}_i = \mathbf{N}$. Hence the following sequence is exact:

$$0 \to \mathbf{M}/\mathbf{N} \hookrightarrow \bigoplus_{i=1}^{m} \mathbf{M}/\mathbf{N}_i$$

By Lemma 1.1.5, we get that $\operatorname{Ass} \mathbf{M}/\mathbf{N} \subseteq \operatorname{Ass} \bigoplus_{i=1}^{m} \mathbf{M}/\mathbf{N}_i$. But every associated prime $\mathfrak{q}$ of the right side is of the form $\bigcap_{i=1}^{m} \operatorname{Ann}(x_i)$ with $x_i \in \mathbf{M}/\mathbf{N}_i$. By prime avoidance $\mathfrak{q} = \operatorname{Ann}(x_i)$ for some i, which is the unique associated prime $\mathfrak{p}_i$ of $\mathbf{N}_i$. Hence $\mathfrak{p} \in \{\mathfrak{p}_i : i = 1, \ldots, m\}$

Conversely, let $\mathfrak{p}_j$ be the associated prime of one of the $\mathbf{N}_j$.

Since the decomposition is minimal, we have $\bigcap_{i \neq j} \mathbf{N}_i \neq \mathbf{N}$. On the one hand, we know that $\bigcap_{i \neq j} \mathbf{N}_i/\mathbf{N} = \bigcap_{i \neq j} \mathbf{N}_i/(\mathbf{N}_j \cap \bigcap_{i \neq j} \mathbf{N}_i)$. By the Second Isomorphism Theorem this is isomorphic to $(\bigcap_{i \neq j} \mathbf{N}_i + \mathbf{N}_j)/\mathbf{N}_j \subseteq \mathbf{M}/\mathbf{N}_j$. So $0 \to \bigcap_{i \neq j} \mathbf{N}_i/\mathbf{N} \hookrightarrow \mathbf{M}/\mathbf{N}_j$ is exact and by Lemma 1.1.5 $\operatorname{Ass} \bigcap_{i \neq j} \mathbf{N}_i/\mathbf{N} \subseteq \operatorname{Ass} \mathbf{M}/\mathbf{N}_j = \{\mathfrak{p}_j\}$. On the other hand we have $0 \to \bigcap_{i \neq j} \mathbf{N}_i/\mathbf{N} \hookrightarrow \mathbf{M}/\mathbf{N}$ is exact and therefore $\operatorname{Ass} \mathbf{M}/\mathbf{N} \supseteq \operatorname{Ass} \bigcap_{i \neq j} \mathbf{N}_i/\mathbf{N}$. Hence $\mathfrak{p}_j \in \operatorname{Ass} \mathbf{M}/\mathbf{N}$. □

Definition 1.2.9

1. A set Σ of prime ideals associated to a submodule $\mathbf{N} \subseteq \mathbf{M}$ is called *isolated*, if for all $\mathfrak{p} \in \operatorname{Ass} \mathbf{M}/\mathbf{N}$ with $\mathfrak{p} \subseteq \mathfrak{q}$ for some $\mathfrak{q} \in \Sigma$, then also $\mathfrak{p} \in \Sigma$.

2. The primary components corresponding to the minimal prime ideals are called the *isolated primary components*. The remaining primary ideals are called *embedded*.

To prove the Second Uniqueness Theorem, we use localization to cut off some primary components of the submodule $\mathbf{N}$. Therefore we need the following Lemma [Grä, Lemma 3], which is a consequence of Theorem 1.1.8.

Lemma 1.2.10 *Let* $\mathbf{N}$ *be a* $\mathfrak{p}$*-primary submodule of* $\mathbf{M}$ *Furthermore let* σ *be a multiplicative subset of* $\mathbf{R}$ *and* $\widetilde{\mathbf{R}} = \mathbf{R}[\sigma^{-1}]$, $\widetilde{\mathbf{M}}$, $\widetilde{\mathbf{N}}$, $\widetilde{\mathfrak{p}}$ *the extensions.*

Then one of the following alternatives holds:

1. *If* $\mathfrak{p} \cap \sigma =$ *then* $\widetilde{\mathbf{N}}$ *is a* $\widetilde{\mathfrak{p}}$*-primary submodule of* $\widetilde{\mathbf{M}}$ *and* $\widetilde{\mathbf{N}} \cap \mathbf{M} = \mathbf{N}$.
2. *If* $\mathfrak{p} \cap \sigma \neq$ *then* $\widetilde{\mathbf{N}} = \widetilde{\mathbf{M}}$.

Proof: By Theorem 1.1.8 primarity commutes with localization, so we only have to prove the assertions about the extensions and recontractions.

1. Assume now $\frac{m}{1} = \frac{n}{s}$, with $n \in \mathbf{N}$, $s \in \sigma$ and $m \in \mathbf{M}$. Hence there exists $r \in \sigma$ such that $r(sm - n) = 0$ and therefore $(rs)m = rn \in \mathbf{N}$. But since $s, r \notin \mathfrak{p}$, r and s are nonzerodivisors in $\mathbf{M}/\mathbf{N}$, which leads us to $m \in \mathbf{N}$.
2. Now take $s \in \sigma \cap \mathfrak{p}$. Since s is nilpotent on $\mathbf{M}/\mathbf{N}$ there exists a power $e \gg 0$ such that $s^e\mathbf{M} \subset \mathbf{N}$ and every $m \in \mathbf{M}$ may be represented as $\frac{n}{s^e}$ in $\widetilde{\mathbf{M}}$ for an appropriate $n \in \mathbf{N}$.

□

Using this lemma, we can compute the (minimal) primary decomposition of the extension module in the localized ring:

Corollary 1.2.11 *Let* $\mathbf{R}$ *be a Noetherian ring and* σ *a multiplicatively closed set. Let* $\mathbf{N} = \bigcap \mathbf{N}_i$ *be a minimal primary decomposition, where the* $\mathbf{N}_i$ *are* $\mathfrak{p}_i$*-primary.*

Then

$$\mathbf{N}[\sigma^{-1}] = \bigcap_{\mathfrak{p}_i \cap \sigma = \emptyset} \mathbf{N}_i[\sigma^{-1}],\ \mathbf{N}[\sigma^{-1}] \cap \mathbf{M} = \bigcap_{\mathfrak{p}_i \cap \sigma = \emptyset} \mathbf{N}_i$$

and these are minimal decompositions.

Proof: By Lemma 1.2.10 we only have to show the minimality of the decompositions. But since the $\mathfrak{p}_i$ are distinct, so are the $\mathfrak{p}_i\mathbf{R}[\sigma^{-1}]$ for all $\mathfrak{p}_i$ such that $\mathfrak{p}_i \cap \sigma = \emptyset$. □

Immediately we conclude:

Theorem 1.2.12 (Second Uniqueness Theorem)
The isolated primary components of $\mathbf{N} \in \mathbf{M}$ *are uniquely determined.*

Proof: Let $\mathbf{N} = \bigcap \mathbf{N}_i$ be a minimal primary decomposition, where the $\mathbf{N}_i$ are $\mathfrak{p}_i$-primary. By Corollary 1.2.11 we get:

$$\mathbf{N}[\sigma^{-1}] \cap \mathbf{M} = \bigcap_{\mathfrak{p}_i \cap \sigma = \emptyset} \mathbf{N}_i.$$

If $\mathbf{N}_i$ is an isolated component, then we have $\mathbf{N}[\sigma^{-1}] \cap \mathbf{M} = \mathbf{N}_i$ with $\sigma := \mathbf{R} \backslash \mathfrak{p}_i$. Since the $\mathfrak{p}_i$ depend only on $\mathbf{N}$ by Theorem 1.2.8, the isolated primary components are uniquely determined. □

We see from the proof, that we can separate the isolated components from the submodule $\mathbf{N}$. In the next section we will learn about a different tool for separation.

1.3 Quotient Computations

An important step in computing the primary decomposition is the separation of the primary components corresponding to the minimal associated primes. Our aim is to separate primary components of $\mathbf{N} \supseteq \mathbf{M}$ via quotient computations (cf. [Grä]).

Definition 1.3.1 Let $\mathbf{I} \subseteq \mathbf{R}$ be an ideal.
We call

- $\mathbf{N} :_{\mathbf{M}} \mathbf{I} := \{m \in \mathbf{M} : \mathbf{I} \cdot m \subseteq \mathbf{N}\}$ the *quotient* and
- $\mathbf{N} :_{\mathbf{M}} \mathbf{I}^{\infty} := \{m \in \mathbf{M} : \exists k > 0, \mathbf{I}^k \cdot m \subseteq \mathbf{N}\}$ the *stable quotient*

of $\mathbf{N}$ by $\mathbf{I}$ in $\mathbf{M}$.

Lemma 1.3.2 *Let $\mathbf{N} \subseteq \mathbf{M}$ be a submodule of $\mathbf{M}$.*

1. *If $\mathbf{N}$ is $\mathfrak{p}$-primary, then:*
   (a) $\mathbf{N} :_{\mathbf{M}} (f)^{\infty} = \mathbf{M}$ *if* $f \in \mathfrak{p}$
   (b) $\mathbf{N} :_{\mathbf{M}} (f)^{\infty} = \mathbf{N}$ *if* $f \notin \mathfrak{p}$.
2. *If $\mathbf{N} \subseteq \mathbf{M}$ is an arbitrary submodule and $\mathbf{N} = \mathbf{N}_1 \cap \cdots \cap \mathbf{N}_m$ a primary decomposition, with $\mathfrak{p}_i$-primary submodules $\mathbf{N}_i$, then:*
   (a) $\mathbf{N} :_{\mathbf{M}} (f)^{\infty} = \bigcap\{\mathbf{N}_i : f \notin \mathfrak{p}_i\}$.
   (b) $\mathbf{N} :_{\mathbf{M}} \mathfrak{a}^{\infty} = \bigcap\{\mathbf{N}_i : \mathfrak{a} \not\subseteq \mathfrak{p}_i\}$ *for all ideals* $\mathfrak{a} \subseteq \mathbf{R}$.

Proof:

1. The multiplication by f is either nilpotent or injective on $\mathbf{M}/\mathbf{N}$.
   (a) If $f \in \mathfrak{p} = \sqrt{\operatorname{Ann} \mathbf{M}/\mathbf{N}}$, then $f^k \cdot \mathbf{M} \subseteq \mathbf{N}$.
   So $\mathbf{M} \subseteq \mathbf{N} :_{\mathbf{M}} (f)^{\infty}$.
   (b) Let $m \in \mathbf{N} :_{\mathbf{M}} (f)^{\infty}$. Hence $f^k \cdot \mathbf{M} \subseteq \mathbf{N}$ for some $k > 0$. Suppose that $m \notin \mathbf{N}$. Then f^k is a zerodivisor in $\mathbf{M}/\mathbf{N}$, and therefore f^k is nilpotent. But this would mean that $f \in \mathfrak{p}$, which is a contradiction. So we have $m \in \mathbf{N}$.

2. (a) From the first part we get:

$$\begin{aligned} \mathbf{N} :_{\mathbf{M}} (f)^{\infty} &= \bigcap (\mathbf{N}_i :_{\mathbf{M}} (f)^{\infty}) \\ &= \bigcap \{\mathbf{N}_i : f \notin \mathfrak{p}_i\} \end{aligned}$$

(b)
- In the case $\mathfrak{a} \subseteq \mathfrak{p}_i = \sqrt{\operatorname{Ann} \mathbf{M}/\mathbf{N}_i}$ there exists some $k > 0$ such that $\mathfrak{a}^k \cdot \mathbf{M} \subseteq \mathbf{N}_i$.
- If $\mathfrak{a} \not\subseteq \mathfrak{p}_i$ then there exists $f \in \mathfrak{a}$ with $f \notin \mathfrak{p}_i$. By the first part we get $\mathbf{N}_i \subseteq \mathbf{N}_i : \mathfrak{a}^{\infty} \subseteq \mathbf{N}_i : (f)^{\infty} = \mathbf{N}_i$

□

We can reformulate the first part of this lemma as:

Lemma 1.3.3 *Let* $\mathbf{N}$ *be a* $\mathfrak{p}$*-primary submodule of* $\mathbf{M}$.
Then:

1. $\mathbf{N} :_{\mathbf{M}} h = \mathbf{N}$ *if* $h \notin \mathfrak{p}$
2. $\mathbf{N} :_{\mathbf{M}} h = \mathbf{M}$ *if* $h \in \mathfrak{q} = \operatorname{Ann} \mathbf{M}/\mathbf{N}$.

Proof:

1. We only have to show that $\mathbf{N} \supseteq \mathbf{N} :_{\mathbf{M}} h$. Let $f \in \mathbf{N} :_{\mathbf{M}} h$, hence $h \cdot f \in \mathbf{N}$. Assume now $f \notin \mathbf{N}$. Then h is a zerodivisor in $\mathbf{M}/\mathbf{N}$. Since $\mathbf{N}$ is primary, h is nilpotent in $\mathbf{M}/\mathbf{N}$. Then there exists $s > 0$, such that $h^s \in \operatorname{Ann} \mathbf{M}/\mathbf{N}$. But this is a contradiction to $h \notin \sqrt{\operatorname{Ann} \mathbf{M}/\mathbf{N}}$. Therefore we have $f \in \mathbf{N}$.
2. If $h \in \operatorname{Ann} \mathbf{M}/\mathbf{N}$ then $h \cdot \mathbf{M} \subseteq \mathbf{N}$. Hence $h \cdot m \in \mathbf{N}, \forall m \in \mathbf{M}$. Therefore $\mathbf{N} :_{\mathbf{M}} h = \mathbf{M}$.

□

1.4 Isolated Primes of a Module

We have seen the importance of the isolated primes before. By Proposition 1.1.9 we know that $\operatorname{Ass} \mathbf{M}/\mathbf{N}$ contains all primes minimal over $\operatorname{Ann} \mathbf{M}/\mathbf{N}$. Indeed we only need to compute the minimal primes of $\operatorname{Ann} \mathbf{M}/\mathbf{N}$, but it is not obvious that the isolated primes of $\mathbf{M}/\mathbf{N}$ and $\operatorname{Ann} \mathbf{M}/\mathbf{N}$ coincide. This showns the following Lemma:

Lemma 1.4.1

$$\operatorname{minAss}\mathbf{M}/\mathbf{N} = \operatorname{minAss}(\operatorname{Ann}\mathbf{M}/\mathbf{N})(= \operatorname{minAss}(\mathbf{R}/\operatorname{Ann}\mathbf{M}/\mathbf{N}))$$

Proof: Let $\mathfrak{p}$ be minimal over $\mathfrak{a} = \operatorname{Ann}\mathbf{M}/\mathbf{N}$. By Proposition 1.1.9, $\mathfrak{p} \in \operatorname{Ass}\mathbf{M}/\mathbf{N}$. Assume now, that $\mathfrak{p}$ is not minimal in $\operatorname{Ass}\mathbf{M}/\mathbf{N}$. Then there exists a prime ideal $\mathfrak{q} \in \operatorname{Ass}\mathbf{M}/\mathbf{N}$, with $\mathfrak{p} \supseteq \mathfrak{q}$. But $\mathfrak{q} = \operatorname{Ann}(x) \supseteq \mathfrak{a}$ for a suitable $x \in \mathbf{M}/\mathbf{N}$. Since $\mathfrak{p}$ is minimal over $\operatorname{Ann}\mathbf{M}/\mathbf{N}$ it is $\mathfrak{p} = \mathfrak{q} \supseteq \mathfrak{a}$. Hence $\mathfrak{p}$ is a minimal associated prime of $\operatorname{Ass}\mathbf{M}/\mathbf{N}$.

Suppose now $\mathfrak{p} \in \operatorname{minAss}\mathbf{M}/\mathbf{N}$. Let $\mathbf{N} = \bigcap \mathbf{N}_i$ be a minimal primary decomposition. Hence by Theorem 1.2.10 $\mathbf{N}_\mathfrak{p} = \bigcap_{\mathfrak{p}_i \subseteq \mathfrak{p}} (\mathbf{N}_i)_\mathfrak{p}$. Since $\mathfrak{p}$ is a minimal associated prime, $\mathbf{N}_\mathfrak{p}$ is equal to the localization of the $\mathfrak{p}$-primary component of $\mathbf{N}$ at $\mathfrak{p}$. Thus $\mathbf{N}_\mathfrak{p}$ is $\mathfrak{p}\mathbf{R}_\mathfrak{p}$-primary and $\mathfrak{p}\mathbf{R}_\mathfrak{p}$ is the only associated prime of $\mathbf{N}_\mathfrak{p}$. From above we know that

$$\{\mathfrak{p}\mathbf{R}_\mathfrak{p}\} = \operatorname{minAss}(\mathbf{M}/\mathbf{N}_\mathfrak{p}) \supseteq \operatorname{minAss}(\operatorname{Ann}(\mathbf{M}/\mathbf{N}_\mathfrak{p})).$$

But $\operatorname{Ass}\operatorname{Ann}(\mathbf{M}/\mathbf{N}_\mathfrak{p})$ contains at least one prime.
Hence $\mathfrak{p}\mathbf{R}_\mathfrak{p} \in \operatorname{minAss}(\operatorname{Ann}(\mathbf{M}/\mathbf{N}_\mathfrak{p}))$ and therefore $\mathfrak{p} \in \operatorname{minAss}(\operatorname{Ann}(\mathbf{M}/\mathbf{N}))$. □

1.5 Reduction To Dimension Zero

The separation tools from section 1.2 and 1.3 can not separate embedded primes (i. e. associated primes that are not minimal) from the isolated primes. But we can use them to compute a primary decomposition of a module which has no embedded primes. Then by Lemma 1.4.1 $\operatorname{Ass}\mathbf{M}/\mathbf{N}$ consists only of the primes minimal over $\operatorname{Ann}\mathbf{M}/\mathbf{N}$.

Definition 1.5.1 We call $\mathcal{C} = \{\mathfrak{p}_0 \subsetneqq \cdots \subsetneqq \mathfrak{p}_m | \mathfrak{p}_i \text{ prime of } \mathbf{R}\}$ a *chain of primes ideals* in $\mathbf{R}$ and $\operatorname{length}(\mathcal{C}) = m$ the *length* of $\mathcal{C}$.

Then the dimension of $\mathbf{R}$ is defined by

$$\dim\mathbf{R} = \sup\{\operatorname{length}(\mathcal{C}) | \mathcal{C} \text{ is a chain in } \mathbf{R}\}.$$

Let $\mathbf{I}$ be an ideal in $\mathbf{R}$ and $\mathbf{M}$ be an $\mathbf{R}$-module. Then we call $\dim\mathbf{I} := \dim\mathbf{R}/\mathbf{I}$ *the (relative) dimension of the ideal* $\mathbf{I}$ and $\dim\mathbf{M} := \dim\mathbf{R}/(\operatorname{Ann}\mathbf{M})$ *the (relative) dimension of the module* $\mathbf{M}$.

We see immediately that a zero-dimensional modules own only isolated primes.

In this section let $\mathbf{R} = \mathbb{K}[x_1, \ldots, x_n]$, where $\mathbb{K}$ is any field. We will follow Gräbe [Grä] and also extend the results of [GP, Chapter 2] to the more general case of modules.

Definition 1.5.2 (Independent Sets) For any given ideal $\mathbf{I} \subseteq \mathbf{R}$ a set of variables $\{x_\nu, \nu \in V\}$ is called an *independent set* if $\mathbf{I} \cap \mathbb{K}[x_\nu, \nu \in V] = 0$.

We can generalize this notation to submodules of the free module $\mathbf{F}$. Let $\mathbf{N} \subseteq \mathbf{M}$ be submodules of $\mathbf{F}$. Then we call the set of variables $\{x_\nu, \nu \in V\}$ a *relative independent set* for $\mathbf{N} \subseteq \mathbf{M}$ if it is an independent set for $\mathbf{I} = \operatorname{Ann}_{\mathbf{R}} \mathbf{M}/\mathbf{N}$.

Notation 1.5.3 Let $\{x_\nu, \nu \in V\}$ be a maximal relative independent set (with respect to inclusion) for $\mathbf{N} \in \mathbf{M}$.

- We write $\widetilde{\mathbf{R}} := \mathbb{K}(x_\nu, \nu \in V)[x_\nu, \nu \notin V] = [\sigma^{-1}]\mathbf{R}$, where $\sigma = \mathbb{K}[x_\nu, \nu \in V] \backslash \{0\}$.
- We denote by $\widetilde{\mathbf{F}}$, $\widetilde{\mathbf{M}}$, $\widetilde{\mathbf{N}}$ and $\widetilde{\mathbf{I}}$ the module resp. ideal extensions of $\mathbf{F}$, $\mathbf{M}$, $\mathbf{N}$ and $\mathbf{I}$, defined by the map $\mathbf{R} \hookrightarrow \widetilde{\mathbf{R}}$.

Remark

Let $V = \{x_{\nu_1}, \ldots, x_{\nu_m}\}$ be a maximal independent set of the ideal $\mathbf{I}$. Then the prime ideal $\mathfrak{p} = (x_\nu : \nu \in V)$ leads to the following maximal chain of prime ideal: $(0) \subsetneqq (x_{\nu_1}) \subsetneqq \cdots \subsetneqq (x_{\nu_1}, x_{\nu_2}, \ldots, x_{\nu_{m-1}}) \subsetneqq \mathfrak{p}$. Hence the number of the elements of a maximal independent set V of $\mathbf{I} = \dim \mathbf{I}$.

Furthermore the localization commutes with taking annihilators. If $\mathfrak{a} = \operatorname{Ann}_{\mathbf{R}} \mathbf{M}/\mathbf{N}$ then $\widetilde{\mathfrak{a}} = \operatorname{Ann}_{\widetilde{\mathbf{R}}} \widetilde{\mathbf{M}}/\widetilde{\mathbf{N}}$. Since the only independent set of $\widetilde{\mathbf{M}}/\widetilde{\mathbf{N}}$ is the empty set, we have $\dim \widetilde{\mathbf{M}}/\widetilde{\mathbf{N}} = \dim \widetilde{\mathbf{R}}/\widetilde{\mathfrak{a}} = 0$

Our next aim is to compute $\widetilde{\mathbf{N}}$ from $\mathbf{N}$ and $\mathbf{J} \cap \mathbf{F}$ of a submodule $\mathbf{J}$ of $\widetilde{\mathbf{F}}$.

Definition 1.5.4 Let $<_\nu$ and $<_\mu$ be two monomial orders of $\mathbb{K}[x_1, \ldots, x_n]$ in $x_\nu, \nu \in V$ resp. x_μ, μ from the complement of V. We define now the *inverse block order* $<$ with respect to V on $\mathbb{K}[x_1, \ldots, x_n]$ by

$$\begin{aligned} x^{\underline{\nu}_1} x^{\underline{\mu}_1} < x^{\underline{\nu}_2} x^{\underline{\mu}_2} : \Longleftrightarrow \; & x^{\underline{\mu}_1} <_\mu x^{\underline{\mu}_2} \\ \text{or } & x^{\underline{\mu}_1} = x^{\underline{\mu}_2} \text{ and } x^{\underline{\nu}_1} <_\nu x^{\underline{\nu}_2} \end{aligned}$$

The corresponding POT (*position over term*) module term order is defined by

$$\begin{aligned} m e_i \prec n e_j : \Longleftrightarrow \; & i < j \\ \text{or } & i = j \text{ and } m < n \end{aligned}$$

In the next theorem (taken from [Rut, Proposition 4.1]), we show how to obtain a Gröbner basis for the extension of a module.

Theorem 1.5.5 *Let B be a Gröbner basis ob $\mathbf{N}$ with respect to $\prec$.*

Then B is also a Gröbner basis of $\widetilde{\mathbf{N}}$ with respect to the corresponding ordering.

Proof: To simplify the proof, we denote the $x_i \in V$ by x and the variables from the complement by y. Hence $\mathbf{F} = \mathbb{K}[x,y]^s = \bigoplus_{i=1}^s \mathbb{K}[x,y] \cdot e_i$ with $e_1, \ldots, e_s$ the standard unit basis vectors. Furthermore let $\prec_\mu$, $\prec_\nu$ the corresponding orderings on $\mathbb{K}(y)[x]^s$ resp. $\mathbb{K}(x)[y]^s$.

We only have to show that $\big(\mathrm{Lt}_{\prec_\mu}(B)\big) \supseteq \mathrm{Lt}\,\prec_\mu((B))$. Let $n \in \mathbf{N}$ and suppose $B = \{G_1, \ldots, G_r\}$. Since B is a Gröbner basis for $\mathbf{N}$, we can write $n = \sum_{j=1}^t c_j x^{\nu_j} y^{\mu_j} G_{i_j}(x,y)$, where $c_j \in \mathbb{K}$, such that

$$x^{\nu_1} y^{\mu_1} \,\mathrm{lp}_\prec(G_{i_1}) \geq x^{\nu_2} y^{\mu_2} \,\mathrm{lp}_\prec(G_{i_2}) \geq \cdots \geq x^{\nu_r} y^{\mu_r} \,\mathrm{lp}_\prec(G_{i_r})$$

and $\mathrm{lp}_\prec(n) = x^{\nu_1} y^{\mu_1} \,\mathrm{lp}_\prec(G_{i_1})$. Then there exists t_0 such that

$$x^{\nu_1} y^{\mu_1} \,\mathrm{lp}_\prec(G_{i_1}) = \cdots = x^{\nu_{t_0}} y^{\mu_{t_0}} \,\mathrm{lp}_\prec(G_{i_{t_0}}) > x^{\nu_{t_0+1}} y^{\mu_{t_0+1}} \,\mathrm{lp}_\prec(G_{i_{t_0+1}}).$$

Write

$$G_{i_j} = a_{i_j}(y) x^{\beta_{i_j}} e_{\sigma(i_j)} + \text{lower terms in } x \text{ with respect to } \prec_\mu .$$

Since we are using a POT inverse block ordering, $\{x^{\mu_{i_j}+\beta_{i_j}} e_{\sigma(i_j)}\}$ is a decreasing sequence in $\mathbb{K}[x_1, \ldots, x_n]^s$. Then exists $t_1 \geq t_0$ such that

$$x^{\mu_1+\beta_{i_1}} e_{\sigma(i_1)} = \cdots = x^{\mu_{t_1}+\beta_{i_{t_1}}} e_{\sigma(i_{t_1})} > x^{\mu_{t_1+1}+\beta_{i_{t_1+1}}} e_{\sigma(i_{t_1+1})}.$$

Thus

$$\begin{aligned} \mathrm{Lt}_{\prec_\mu}(n) &= \sum_{j=1}^{t_1} c_j y^{\nu_j} x^{\mu_j} a_{i_j}(y) x^{\beta_{i_j}} \\ &= \left(\sum_{j=1}^{t_1} c_j y^{\nu_j} a_{i_j}(y) \right) \cdot x^{\mu_j+\beta_{i_j}} \end{aligned}$$

as long as $\left(\sum_{j=1}^{t_1} c_j y^{\nu_j} a_{i_j}(y)\right) \neq 0$. But this is true, since

$$\mathrm{Lt}_{\prec\nu} \left(\sum_{j=1}^{t_1} c_j y^{\nu_j} a_{i_j}(y)) \right) = \left(\sum_{j=1}^{t_0} c_j y^{\nu_j} \,\mathrm{Lt}_{\prec_\nu}(a_{i_j}(y)) \right) = \frac{\mathrm{Lt}_\prec(n)}{y^{\mu_1+\beta_{i_j}}} \neq 0.$$

Hence $\mathrm{Lt}_{\prec_\mu}(n) = \sum_{j=1}^{t_1} c_j y^{\nu_j} x^{\mu_j} \,\mathrm{Lt}_{\prec_\mu}(G_{i_j})$ and therefore we have $\mathrm{Lt}_{\prec_\mu}(n) \in \big(\mathrm{Lt}_{\prec_\mu}(B)\big)$. □

For retractions the situation is slightly more difficult:

Definition 1.5.6 A *denominator-free basis* B of $\mathbf{J} \subseteq \widetilde{\mathbf{F}}$ is a set of elements in $\mathbf{F}$ such that

$$(B) = \mathbf{J} \subseteq \widetilde{\mathbf{F}}.$$

We can construct a denominator-free basis from an arbitrary basis by clearing denominators. For the next lemma, we will assume that $<$ is a well-ordering.

Lemma 1.5.7 *Let B be a denominator-free Gröbner basis of $\mathbf{J} \subseteq \widetilde{\mathbf{F}}$ and $c \in \mathbf{R}$ the product of the leading coefficients of the elements of B regarded as polynomial vectors in $\widetilde{\mathbf{F}}$.*

Then:

$$\mathbf{J} \cap \mathbf{F} = (B) :_{\mathbf{F}} (c)^{\infty}.$$

Proof: Every element of $(B) :_{\mathbf{F}} (c)^{\infty}$ is in $\mathbf{J}$ and $\mathbf{F}$, since c is invertible in $\widetilde{\mathbf{R}}$. So we only have to show the inclusion: $\mathbf{J} \cap \mathbf{F} \subseteq (B) :_{\mathbf{F}} (c)^{\infty}$. Suppose now that $f \in \mathbf{J} \cap \mathbf{F}$. We denote the elements of the Gröbner basis B by $f_\nu, \nu = 1, \ldots, m$. Using the Normalform Algorithm we get:

$$f = \sum_{i=1}^{m} \xi_\nu \cdot f_\nu + r, \text{ with } \xi_\nu \in \widetilde{\mathbf{R}}.$$

The algorithm requires only dividing by the leading coefficients $\mathrm{lc}(f_\nu)$ of the f_ν. Hence we can find $z \in \mathbf{R}$ such that $z \cdot f \equiv r \bmod (B)$ in $\mathbf{R}$ for a z which is a unit in $\widetilde{\mathbf{R}}$ that can be chosen to be a product of leading coefficients $\mathrm{lc}(f_\nu)$ of the f_ν. Since the f_ν form a Gröbner basis we have $r = 0$ and therefore $z \cdot f \in (B)$. Furthermore $\exists k > 0$ such that $z | c^k$ and hence $c^k \cdot f \in (B)$. So $f \in (B) :_{\mathbf{F}} (c)^{\infty}$. □

Finally we can summarize the results of this section and make some additional conclusions:

Lemma 1.5.8 *With the notation as above:*

1. $\widetilde{\mathbf{M}}/\widetilde{\mathbf{N}}$ *is zero-dimensional.*
2. $$\mathbf{N}' := \widetilde{\mathbf{N}} \cap \mathbf{M} = \mathbf{N} :_{\mathbf{M}} (c)^{\infty}$$
   is equidimensional of relative dimension $\dim \mathrm{Ann}\, \mathbf{M}/\mathbf{N}$.
   If e is an integer such that $c^e \cdot \mathbf{N}' \subseteq \mathbf{N}$, then
   $$\mathbf{N} = \mathbf{N}' \cap (\mathbf{N} + c^e \cdot \mathbf{M})$$
3. *Let $\widetilde{\mathbf{N}} = \mathbf{N}_1 \cap \cdots \cap \mathbf{N}_t$ be a minimal primary decomposition and $\mathbf{F} = \mathbb{K}[x_1, \ldots, x_n]^s$. Then $\widetilde{\mathbf{N}} \cap \mathbf{F} = (\mathbf{N}_1 \cap \mathbf{F}) \cap \cdots \cap (\mathbf{N}_t \cap \mathbf{F})$ is a minimal primary decomposition.*

Proof:

1. Clear by the choice of the x_ν as maximal independent set.

2. By Lemma 1.5.7 we conclude:

$$\widetilde{\mathbf{N}} \cap \mathbf{M} = \widetilde{\mathbf{N}} \cap \mathbf{F} \cap \mathbf{M} = \mathbf{N} :_{\mathbf{F}} (c)^\infty \cap \mathbf{M} = \mathbf{N} :_{\mathbf{M}} (c)^\infty$$

   If the dimension of one of the primary components were strictly smaller than $\dim \operatorname{Ann} \mathbf{M}/\mathbf{N} = \#\{x_\nu : \nu \in V\}$ then $\sqrt{\operatorname{Ann} \mathbf{M}/\mathbf{N}} \cap \mathbb{K}[x_\nu : \nu \in V] \neq \emptyset$. Hence by Lemma 1.2.11 $\mathbf{N}'$ is equal to the intersection of the primary components corresponding to the minimal associated primes.

   Let $x \in (\mathbf{N} + c^e\mathbf{M}) \cap \mathbf{N}'$. Then $x = n + c^e m$, with $n \in \mathbf{N}, m \in \mathbf{M}$. Since x is also in $\mathbf{N}' = \mathbf{N} : c^\infty$, we have that $c^e x \in \mathbf{N}$ and therefore $c^e x = c^e n + c^{2e} m \in \mathbf{N}$. Hence $m \in \mathbf{N} : c^\infty = \mathbf{N} : c^e$, which means that $c^e m \in \mathbf{N}$, and finally $x = n + c^e m \in \mathbf{N}$.

3. Let $\widetilde{\mathbf{N}} = \mathbf{N}_1 \cap \cdots \cap \mathbf{N}_t$ be a minimal primary decomposition and let s be such that $s \cdot m \in \mathbf{N}_i \cap \mathbf{F} \subseteq \mathbf{N}_i$ for some $m \in \mathbf{M}$. Since the $\mathbf{N}_i$ are primary we have $s^k \cdot \mathbf{M} \subseteq \mathbf{N}_i$. Hence $s^k \cdot \mathbf{M} \subseteq \mathbf{N}_i \cap \mathbf{F}$ (because $s \in \mathbb{K}[x_1, \ldots, x_n]$). Obviously we have $\mathfrak{p}_i \cap \mathbb{K}[x_1, \ldots, x_n] \neq \mathfrak{p}_j \cap \mathbb{K}[x_1, \ldots, x_n]$ for $i \neq j$. It is left to prove that we cannot omit one of the $\mathbf{N}_i$. We have to show that $\operatorname{Ass} \mathbf{M}/\mathbf{N} = \{\mathfrak{p}_i : i = 1, \ldots, t\}$. By Lemma 1.4.1 and since this statement holds for ideals we only have to show the following claim:

$$\operatorname{Ann} \widetilde{\mathbf{M}}/\widetilde{\mathbf{N}} = \Big(\operatorname{Ann} \mathbf{M}/(\widetilde{\mathbf{N}} \cap \mathbf{M})\Big) \cdot \widetilde{\mathbf{R}}$$

   Let $x \in \operatorname{Ann} \widetilde{\mathbf{M}}/\widetilde{\mathbf{N}}$ then $x \in \widetilde{\mathbf{R}}$ and $x \cdot \widetilde{\mathbf{M}} \subseteq \widetilde{\mathbf{N}}$. Since $\mathbf{M} \subseteq \widetilde{\mathbf{M}}$, $x \cdot \mathbf{M} \subseteq \widetilde{\mathbf{N}}$ and hence there exists $a \in \mathbb{K}[x_\nu, \nu \in V]$ such that $a \cdot x \cdot \mathbf{M} \subseteq \widetilde{\mathbf{N}} \cap \mathbf{M}$. Thus $x \in \operatorname{Ann} \mathbf{M}/(\widetilde{\mathbf{N}} \cap \mathbf{M}) \cdot \widetilde{\mathbf{R}}$.

   Conversely we have $x \in \operatorname{Ann} \mathbf{M}/(\widetilde{\mathbf{N}} \cap \mathbf{M}) \cdot \widetilde{\mathbf{R}}$. Hence $a \cdot x \cdot \mathbf{M} \subseteq \widetilde{\mathbf{N}} \cap \mathbf{M}$. Therefore $x \cdot \widetilde{\mathbf{M}}$ is contained in the extension of $\widetilde{\mathbf{N}} \cap \mathbf{M}$, which is equal to $\widetilde{\mathbf{N}}$. So $x \in \operatorname{Ann} \widetilde{\mathbf{M}}/\widetilde{\mathbf{N}}$.

□

The Case of $\operatorname{Ann} \mathbf{M}/\mathbf{N} = (0)$

Unlike the case of the zero ideal a module with $\operatorname{Ann} \mathbf{M}/\mathbf{N} = (0)$ need not to be not primary. Indeed it is quasi-primary with isolated prime (0). Consider for example the module

$$\mathbf{N} = \left(\begin{pmatrix} xy \\ xz \end{pmatrix}\right) = \left(\begin{pmatrix} y \\ z \end{pmatrix}\right) \cap \left(\begin{pmatrix} x \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ x \end{pmatrix}\right)$$

We can reduce this kind of modules to a zero-dimensional (0)-primary module using $V = \{x_1, \ldots, x_n\}$ as maximal independent set.

Chapter 2

The Algorithms

2.1 Gianni, Trager and Zacharias

In this section the algorithm of Gianni, Trager and Zacharias [GTZ] is decribed. First we will follow Pfister and Greuel [GP] for the primary decomposition of ideals, then formulate the more general results for the case of modules. Later we will give an implementation of this algorithm in SINGULAR[1] .

Since we can reduce the primary decomposition problem to the zero-dimensional case by Lemma 1.5.8, we first need an algorithm for modules of (relative) dimension zero.

2.1.1 Ideals of Dimension Zero

Let $>_{lex}$ be the lexicographical ordering defined by $x_1 > \cdots > x_n$. Since we have $\dim \mathbf{M}/\mathbf{N} = 0$ and $\dim \mathbf{M}/\mathbf{N} = \dim(\operatorname{Ann} \mathbf{M}/\mathbf{N}) = \#\{x_\nu : \nu \in V\}$ for a maximal independent set V, $\operatorname{Ann} \mathbf{M}/\mathbf{N} \cap \mathbb{K}[x_\nu] \neq 0$ for all $\nu = 1, \ldots, n$. So we may take one of the x_ν—say x_n—for the following definition:

Definition 2.1.1 A maximal ideal $\mathfrak{m} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ is called *in general position* with respect to $>_{lex}$ if $\mathfrak{m} = (x_1 + g_1(x_n), \ldots, x_{n-1} + g_{n-1}(x_n), g_n(x_n))$ for suitable polynomials g_i. A zero-dimensional module $\mathbf{N} \subseteq \mathbf{M}$ (resp. ideal $\mathbf{I} \subseteq \mathbf{R}$) is called *in general position* with respect to $>_{lex}$ if all associated primes $\mathfrak{p}_1, \ldots, \mathfrak{p}_k$ are in general position and if $\mathfrak{p}_i \cap \mathbb{K}[x_n] \neq \mathfrak{p}_j \cap \mathbb{K}[x_n], \forall i \neq j$.

Remark Since we are dealing with zero-dimensional modules we have:
$\mathbf{N} \subseteq \mathbf{M}$ is in general position $\iff$ $\operatorname{Ann} \mathbf{M}/\mathbf{N}$ is in general position (as ideal).

[1] SINGULAR is available via web at `http://singular.mathematik.uni-kl.de/`

The next proposition tells us, that every zero-dimensional ideal is isomorphic to an ideal in general position (given by almost all randomly chosen maps). We will assume for this section, that the field $\mathbb{K}$ is of characteristic 0 or $\operatorname{char}\mathbb{K} \gg 0$.

Proposition 2.1.2 *Let $\mathbf{I} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ be a zero-dimensional ideal.*

Then there is a Zariski open and dense subset $U \subseteq \mathbb{K}^{n-1}$ such that for all $\underline{a} = (a_1, \ldots, a_{n-1}) \in U$ the map $\varphi_{\underline{a}}$, defined by $\varphi_{\underline{a}}(x_i) = x_i$ if $i < n$ and $\varphi_{\underline{a}}(x_n) = x_n + \sum_{i=1}^{n-1} a_i x_i$ has the property that $\varphi_{\underline{a}}((I))$ is in general position with respect to $>_{lex}$.

Proof: If $\mathfrak{p} \in \operatorname{Ass}\mathbf{M}/\mathbf{N}$ then $\mathfrak{p}$ is maximal and hence the field $\kappa = \mathbb{K}[x_1, \ldots, x_n]/\mathfrak{p}$ is a finite extension of $\mathbb{K}$. By the *Theorem of the Primitive Element* there exists a dense open subset $U \subseteq \mathbb{K}^{n-1}$ such that for $\underline{a} = (a_1, \ldots, a_{n-1}) \in U$ the element $z = x_n + \sum_{i=1}^{n-1} a_i x_i$ is a primitive element for the field extension.

Using the corresponding map $\varphi_{\underline{a}}$ we may assume that x_n is a primitive element, that is

$$\mathbb{K}[x_1, \ldots, x_n]/\mathfrak{p} = \mathbb{K}[x_n]/(f_n(x_n))$$

for an irreducible polynomial $f_n(x)$. Now define $f_i(x_n)$ by $x_i \mod \mathfrak{p} \cong f_n(x_n)$ and we obtain

$$(x_1 - f_1(x_n), \ldots, x_{n-1} - f_{n-1}(x_n), f_n(x_n)) = \mathfrak{p}.$$

The set of these generators is obviously a Gröbner basis with the required properties.

Let $\mathbf{I} = \mathfrak{q}_1 \cap \cdots \cap \mathfrak{q}_s$ be a primary decomposition with associated primes $\mathfrak{p}_1, \ldots, \mathfrak{p}_s$, then $\varphi_{\underline{a}}(\mathfrak{p}_j)$ are in general position with respect to $>_{lex}$ for almost all $\underline{a} \in \mathbb{K}^{n-1}$. On the other hand, the zero-set of $\mathbf{I}$ is just a set of s different points which project to different points for almost all $\underline{a}$. □

The next proposition shows how to obtain a primary decomposition from a zero-dimensional ideal in general position.

Proposition 2.1.3 *Let $\mathbf{I} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ be a zero-dimensional ideal. Let $(g) = \mathbf{I} \cap \mathbb{K}[x_n], g = g_1^{\nu_1} \cdot \cdots \cdot g_k^{\nu_k}$, g_i prime and $g_i \neq g_j$ for $i \neq j$.*

Then

1. $\mathbf{I} = \bigcap_{i=1}^{k} (\mathbf{I}, g_i^{\nu_i})$.
2. *If $\mathbf{I}$ is in general position with respect to $>_{lex}$ then the $(\mathbf{I}, g_i^{\nu_i})$ are primary ideals.*

Proof: (2) Let $\mathbf{I} = \mathfrak{q}_1 \cap \cdots \cap \mathfrak{q}_l$ with associated primes $\mathfrak{p}_1, \ldots, \mathfrak{p}_l$ and let $\mathfrak{p}_i \cap \mathbb{K}[x_n] = (p_i)$. Since $\mathbf{I}$ is in general position, $p_1, \ldots, p_l$ are coprime and therefore $\bigcap \mathfrak{p}_i \cap \mathbb{K}[x_n] = \bigcap (p_i) = \left(\prod_{i=1}^{l} p_i\right)$. On the other hand, $\mathbf{I} \cap \mathbb{K}[x_n] =$

(g) implies that g divides a power of $\prod_{i=1}^{l} p_i$. Therefore $l = k$ and that we may assume $g_i = p_i$ for $i = 1, \ldots, s$. Hence $\mathfrak{p}_i \supseteq (\mathbf{I}, g_i^{\rho_i})$ and $\mathfrak{p}_j \not\supseteq (\mathbf{I}, g_i^{\rho_i})$ for $i \neq j$. Because of $\mathbf{I} \subseteq \left(\mathbf{I}, g_k^{\rho_k}\right)$ we know that $\operatorname{Ass}\left(\mathbf{I}, g_k^{\rho_k}\right) \subseteq \operatorname{Ass}(\mathbf{I})$ and therefore (2) is proved.

To prove (1) note that

$$\mathbf{I} \cap \mathbb{K}[x_n] = \bigcap_{k=1}^{s} \left(\mathbf{I}, g_k^{\rho_k}\right) \cap \mathbb{K}[x_n].$$

Let $f \in \bigcap(\mathbf{I}, g_k^{\rho_k}), f = f_\nu + \xi_\nu g_\nu^{\rho_\nu}$ with $f_\nu \in \mathbf{I}$, then $g^{(\nu)} f = g^{(\nu)} f_\nu + \xi_\nu \cdot g$ for $g^{(\nu)} = \frac{g}{g_\nu^{\rho_\nu}}$.

Let $\sum_{\nu=1}^{s} \eta_\nu g^{(\nu)} = 1$, then

$$f = \sum_{\nu=1}^{s} \eta_\nu g^{(\nu)} f_\nu + \left(\sum_{\nu=1}^{s} \eta_\nu \xi_\nu\right) g \in \mathbf{I},$$

which proves (1). □

Remark None of the $\mathfrak{q}_i = (\mathbf{I}, g_i^{\nu_i})$ can be omitted. If a $\mathfrak{q}_i$ could be omitted, then $\frac{g}{g_i^{\nu_i}} \in \mathbf{I} \cap \mathbb{K}[x_n] = (g)$, contradiction. Furthermore, we have that $\sqrt{\mathfrak{q}_i} \neq \sqrt{\mathfrak{q}_j}, \forall i \neq j$ since $\sqrt{\mathfrak{q}_i} \ni g_i, \sqrt{\mathfrak{q}_j} \ni g_j$ and g_i, g_j are coprime.

This means that the decomposition is minimal.

In the algorithm we try to obtain the generial position by a random coordinate chance, but we cannot be sure in practice that the coordinate change was random enough. So we need a test whether the $(\mathbf{I}, g_i^{\nu_i})$ are primary and in general position:

Lemma 2.1.4 *The zero-dimensional ideal $\mathfrak{q} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ is primary and in general position with respect to $>_{lex}$ if*

1. *$\mathfrak{q} \cap \mathbb{K}[x_n] = (g_n^{\nu_n}), g_n$ irreducible.*
2. *For each $i < n$, $\mathfrak{q}$ contains an element $(x_i + g_i(x_n))^{\nu_i}$.*

Proof: $\mathfrak{m} = \sqrt{\mathfrak{q}}$ is generated by irreducible elements and therefore prime. Hence $\mathfrak{q}$ is primary with the unique associated prime $\mathfrak{m} = (x_1 + g_1(x_n), \ldots, x_{n-1} + g_{n-1}(x_n), g_n(x_n))$. By Definition 2.1.1, $\mathfrak{q}$ is in general position. □

For $\mathfrak{q} = (\mathbf{I}, g_i^{\nu_i})$ the first condition is already fulfilled.

To check the second let $f = \{f_1, \ldots, f_s\}$ be a minimal Gröbner basis of $\mathfrak{q}$ such that $\operatorname{Lt}(f_i) < \operatorname{Lt}(f_{i+1}), \forall i$. Then $\mathfrak{q} \cap \mathbb{K}[x_n] = (f_1)$ and f_1 is the power of an irreducible element g. Since $\mathfrak{q}$ is zero-dimensional we know that $\mathfrak{q} \cap \mathbb{K}[x_i] \neq 0$ and therefore there exists polynomials $f_1 = f_{i_n}, f_{i_{n-1}}, \ldots, f_{i_1}$ such that $\operatorname{Lt}(f_{i_a}) = \operatorname{lc}(f_{i_a}) \cdot x_a^{\nu_a}$. So we have to check in the second step whether $\frac{1}{\operatorname{lc}(f_{i_{n-1}})} f_{i_{n-1}} = (x_{n-1} + g_{n-1}(x_n))^{\nu_{n-1}} \mod (g)$.

Assume we have found $g = g_n, g_{n-1}, \ldots, g_{n-k+1}$.
In the k-th step we have to check $\frac{1}{\mathrm{lc}(f_{i_{n-k}})} \cdot f_{i_{n-k}} = (x_{n-k} + g_{n-k}(x_n))^{\nu_{n-k}}$ mod $(g_n, x_{n-1} + g_{n-1}, \ldots, x_{n-k+1} + g_{n-k+1})$.

2.1.2 Modules of Dimension Zero

We ganz generalize the primary decomposition algorithm for zero-dimensional ideals to the case of zero-dimensional modules. But before we can prove a more general version of Proposition 2.1.3, we will need some lemmas. They will show that there is a close relationship between the associated primes of $\mathbf{M}/\mathbf{N}$ and $\mathrm{Ass}\,(\mathrm{Ann}\,\mathbf{M}/\mathbf{N})$.

Lemma 2.1.5 *Let* $\mathbf{M}$, $\mathbf{N}$ *as before and* $h \in \mathbf{R}$. *Then*

$$\mathrm{Ann}\,\mathbf{M}/(\mathbf{N} : h) = (\mathrm{Ann}\,\mathbf{M}/\mathbf{N}) : h.$$

Proof: Take $r \in \mathbf{R}$. Then $r \in \mathrm{Ann}\,\mathbf{M}/(\mathbf{N} : h)$

$$\begin{aligned} &\iff r \cdot \mathbf{M} &&\subseteq \mathbf{N} : h \\ &\iff h \cdot r \cdot \mathbf{M} &&\subseteq \mathbf{N} \\ &\iff h \cdot r &&\in \mathrm{Ann}\,\mathbf{M}/\mathbf{N} \\ &\iff r &&\in (\mathrm{Ann}\,\mathbf{M}/\mathbf{N}) : h \end{aligned}$$

□

Lemma 2.1.6 *Let* $\mathbf{N}$ *be a zero-dimensional submodule of* $\mathbf{M}$ *and consider the minimal primary decompositions* $\mathrm{Ann}\,\mathbf{M}/\mathbf{N} = \bigcap \mathfrak{q}_i$, $\mathbf{N} = \bigcap \mathbf{N}_i$, *such that the* $\mathfrak{q}_i$ *(resp. the* $\mathbf{N}_i$*) are the* $\mathfrak{p}_i$*-primary components.*

Then $\mathfrak{q}_i = \mathrm{Ann}\,\mathbf{M}/\mathbf{N}_i$.

Proof: Since $\mathbf{N}$ is zero-dimensional, we can find separators $f_i \in \mathfrak{p}_i, f \notin \mathfrak{p}_i, \forall i \neq j$ for every $\mathfrak{p}_i$. We know from Lemma 1.3.2 that

$$\mathbf{N} : f^\infty = \mathbf{N}_i, \mathrm{Ann}(\mathbf{M}/\mathbf{N}) : f^\infty = \mathfrak{q}_i$$

and there exists $e > 0$ such that $\mathbf{N} : f^e = \mathbf{N} : f^\infty, \mathrm{Ann}(\mathbf{M}/\mathbf{N}) : f^e = \mathrm{Ann}(\mathbf{M}/\mathbf{N}) : f^\infty$ because the modules are noetherian. Hence by Lemma 2.1.5: $\mathrm{Ann}\,\mathbf{M}/\mathbf{N}_i = \mathrm{Ann}\,\mathbf{M}/(\mathbf{N} : f^e) = \mathrm{Ann}(\mathbf{M}/\mathbf{N}) : f^e = \mathfrak{q}_i$. □

Proposition 2.1.7 *Let* $\mathbf{N} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ *be a zero-dimensional module in general position with respect to* $>_{lex}$. *Let* $(g) = \mathrm{Ann}\,\mathbf{M}/\mathbf{N} \cap \mathbb{K}[x_n], g = g_1^{\nu_1} \cdot \ \ldots \ \cdot g_k^{\nu_k}$, g_i *prime and* $g_i \neq g_j$ *for* $i \neq j$ *and* $h_i := \prod_{i \neq j} g_j^{\nu_j}$.

Then $\mathbf{N} = \bigcap_{i=1}^{k} (\mathbf{N} : h_i)$ *is the uniquely determined minimal primary decomposition.*

Proof: Let $\mathbf{N} = \bigcap \mathbf{N}_i$ be the minimal primary decomposition, which is uniquely determined, since the only primes belonging to $\mathbf{N}$ are the minimal associated primes. Therefore no associated primes can be omitted. But $\mathbf{N} : h_i = (\bigcap \mathbf{N}_j) : h_i = \bigcap(\mathbf{N}_j : h_i) = \mathbf{N}_i$, because $\mathbf{N}_i : h_i = \mathbf{N}_i$ and $\mathbf{N}_j : h_i = \mathbf{M}$ if $i \neq j$. Hence $\mathbf{N} : h_i$ are the primary components of $\mathbf{N}$.

We still have to show that $\mathfrak{p}_i \neq \mathfrak{p}_j$ for $i \neq j$. But this follows directly by Lemma 2.1.6 from the case of ideals (Lemma 2.1.3). □

As in the case of ideals we need to apply a random coordinate change. So we cannot be sure that the $\mathbf{N}_i$ are primary. But the following Corollary (of Lemma 1.4.1) shows that we can test the annihilators instead of the submodules.

Corollary 2.1.8 *Let* $\mathbf{N}$ *be a zero-dimensional submodule of* $\mathbf{M}$. *Then* $\mathbf{N}$ *is a primary submodule if and only if* $\operatorname{Ann} \mathbf{M}/\mathbf{N}$ *is primary.*

Proof: By Lemma 1.4.1 we get that $\operatorname{minAss} \mathbf{M}/\mathbf{N} = \operatorname{minAss}(\operatorname{Ann} \mathbf{M}/\mathbf{N})$. But in the zero-dimensional case all associated primes are minimal. Hence $\operatorname{Ass} \mathbf{M}/\mathbf{N} = \{\mathfrak{p}\} \iff \operatorname{Ass}(\operatorname{Ann} \mathbf{M}/\mathbf{N}) = \{\mathfrak{p}\}$. □

Remark We can use the primarity test for zero-dimensional ideals also as a primarity test for zero-dimensional submodules!

Now we can write down the procedure for the zero-dimensional decomposition:

Algorithm 2.1.1

ZeroDecMod($\mathbf{N}$,$\mathbf{M}$)

Input: A zero-dimensional submodule $\mathbf{N} \subseteq \mathbf{M}$
Output: The primary decomposition of $\mathbf{N}$ in $\mathbf{M}$

- Result $:= \emptyset, \mathbf{I} := \operatorname{Ann} \mathbf{M}/\mathbf{N}$
- Choose a random $a \in \mathbb{K}^{n-1}$
- apply the random coordinate change:
  $\mathbf{I} := \varphi_a(\mathbf{I}), \mathbf{N} := \varphi_a(\mathbf{N}), \mathbf{M} := \varphi_a(\mathbf{M})$
- $(G) := \mathbf{I} \cap \mathbb{K}[x_n]$
- factorize $G = g_1^{\nu_1} \cdot \ \ldots \ \cdot g_t^{\nu_t}$
- for $i = 1$ to t do

  $h_i := \prod_{j \neq i} g_j^{\nu_j}$
  $\mathbf{N}_i := \mathbf{N} : h_i, \mathfrak{q}_i := (\mathbf{I}, g_i^{\nu_i})$
  If the primarity test fails then Result = Result∪**ZeroDecMod**($\mathbf{N}_i$, $\mathbf{M}$)

else Result = Result $\cup \{(\mathbf{N}_i, \mathfrak{p}_i\}$

- return Result

In the programming language of Singular it looks as follows (for the case $\mathbf{M} = \mathbb{K}[x_1, \ldots, x_n]^s$). The optional argument is not needed for the zero-dimensional decomposition. In the higher dimensional decomposition it is used to avoid redundant components.

2.1.3 Modules of Higher Dimension

By Lemma 1.5.8 we can decompose the module $\mathbf{N} \in \mathbf{M}$ into an equidimensional module $\mathbf{N}'$ and another module $\mathbf{N}''$. The primary components of $\mathbf{N}'$ can be computed using reduction to dimension zero (via localization) and **ZeroDecMod**. We can apply this to $\mathbf{N}''$ once more, but we have to check that this procedure terminates. Therefore we use induction on the number of the maximal independent sets and $\dim \mathbf{M}/\mathbf{N}$ (i. e. the size of a maximal independent set). If apply the first part of the algorithm to $\mathbf{N}$ using the maximal independent set V, then V is not an independent set for $\mathbf{N}''$. So we reduce either the number of maximal independent sets or (if we have used all maximal independent sets of $\mathbf{N}$) the dimension.

Now we can formulate the complete algorithm using Lemma 1.5.8:

Algorithm 2.1.2

MoGTZ(N,M)

Input: A submodule $\mathbf{N} \subseteq \mathbf{M}$
Output: The primary decomposition von $\mathbf{N}$ in $\mathbf{M}$

- $\mathbf{I} := \operatorname{Ann} \mathbf{M}/\mathbf{N}$, $v :=$ MaxIndepSet($\mathbf{I}$)
- Change ring to $\mathbb{K}(v)[x\backslash v]$.
- Compute a Gröbner Basis $B = \{G_1, \ldots, G_s \in (\mathbb{K}[x_1, \ldots, x_n])^s\}$ of $\mathbf{N}$ (in $\mathbb{K}(v)[x\backslash v]$).
- $h := \prod \operatorname{lc}(G_i)$
- Compute e such that $(B) : h^e = (B) : h^{e+1}$
- $\{(\mathbf{N}_i, \mathfrak{p}_i)\} :=$ **ZeroDecMod**($\mathbf{N}$)
- Primary := $\{(\mathbf{N}_i \cap \mathbf{M}, \mathfrak{p}_i \cap \mathbb{K}[x_1, \ldots, x_n])\}$
- Primary := Primary $\cup$ **MoGTZ**($\mathbf{N} + h^e \cdot \mathbf{M}$)
- return Primary

In the language of Singular, we have:

2.1.4 Minimal Decomposition

Although the decomposition obtained by our algorithm is minimal for the equidimensional part of the modules, we may get redundant components from the decomposition of the module $\mathbf{N}+h^e\cdot\mathbf{M}$. We avoid computing them if we introduce a module *check*, which is the intersection of the components computed before.

We give a revised version of the algorithm in the following way:

Algorithm 2.1.3

ZeroDecMin($\mathbf{N}$,$\mathbf{M}$[, *check*])

Input: A zero-dimensional submodule $\mathbf{N} \subseteq \mathbf{M}$
optionally a module *check*
Output: A primary decomposition von $\mathbf{N}$

- If *check* does not exist then *check* $= \mathbf{M}$
- else If $\mathbf{N} \subseteq$ *check* then return $\emptyset$
- Result $:= \emptyset, \mathbf{I} := \operatorname{Ann} \mathbf{M}/\mathbf{N}$
- Choose a random $a \in \mathbb{K}^{n-1}$
- apply the random coordinate change:
  $\mathbf{I} := \varphi_a(\mathbf{I}), \mathbf{N} := \varphi_a(\mathbf{N}), \mathbf{M} := \varphi_a(\mathbf{M})$
- $(G) := \mathbf{I} \cap \mathbb{K}[x_n]$
- factorize $G = g_1^{\nu_1} \cdot \ \ldots \ \cdot g_t^{\nu_t}$
- for $i = 1$ to t do

  $h_i := \prod_{j\neq i} g_j^{\nu_j}$
  $\mathbf{N}_i := \mathbf{N} : h_i, \mathfrak{q}_i := (\mathbf{I}, g_i^{\nu_i})$
  If *check* $\not\supseteq \mathbf{N}_i$
  If the primary test fails then Result = Result$\cup$**ZeroDecMod**($\mathbf{N}_i$, $\mathbf{M}$)
  else Result = Result $\cup \{(\mathbf{N}_i, \mathfrak{p}_i\}$
- return Result

And the reduction to the zero-dimensional case:

Algorithm 2.1.4

MinGTZ($\mathbf{N}$,$\mathbf{M}$[, *check*])

Input: A submodule $\mathbf{N} \subseteq \mathbf{M}$
optional a module *check*
Output: The primary decomposition von $\mathbf{N}$

- If *check* $\subseteq \mathbf{I}$ then return $\emptyset$
- $\mathbf{I} := \operatorname{Ann} \mathbf{M}/\mathbf{N}$, $v :=$ MaxIndepSet($\mathbf{I}$)
- Change ring to $\mathbb{K}(v)[x \backslash v]$.
- Compute a Gröbner $B = \{G_1, \ldots, G_s \in (\mathbb{K}[x_1, \ldots, x_n])^s\}$ of $\mathbf{N}$(in $\mathbb{K}(v)[x \backslash v]$).
- $h := \prod \operatorname{lc}(G_i)$
- Compute e such that $(B) : h^e = (B) : h^{e+1}$
- $\{(\mathbf{N}_i, \mathfrak{p}_i)\} :=$ **ZeroDecMin**($\mathbf{N}$)
- Primary $:= \{(\mathbf{N}_i \cap \mathbf{M}, \mathfrak{p}_i) : \mathbf{N}_i \not\subseteq$ *check*$\}$
- Primary := Primary $\cup$ **MinGTZ**$(\mathbf{N} + h^e \cdot \mathbf{M}), \mathbf{M},$ *check*$)$
- return Primary

2.2 Shimoyama-Yokoyama

The Algorithm of Shimoyama and Yokoyama uses quotient computations to separate the components corresponding to the isolated primes. In contrast to the algorithm of Gianni, Trager and Zacharias, we do not cut down to dimension zero in one step. The following generalization of the algorithm is due to Gräbe [Grä].

Before we start with the algorithm, we need the following definition:

Definition 2.2.1 Let $L = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_k\}$ be the isolated primes of a submodule $\mathbf{N} \subseteq \mathbf{M}$. Then we call $f_i \in \mathbf{R}$ a *separator* and say f_i *separates* $\mathfrak{p}_i$ from $L \backslash \{\mathfrak{p}_i\}$ if $f_i \in \mathfrak{p}_j$ for $i \neq j$, but $f_i \notin \mathfrak{p}_i$.

Remark The construction of the f_i is easy. Lacking embedded primes, we can find for each $i \neq j$ a polynomial $p_{ij} \in \mathfrak{p}_j$ but not contained in $\mathfrak{p}_i$. Then $f_i := \prod_{i \neq j} p_{ij}$ has the desired property.

We start from a set of isolated primes and use ideal separation to compute the corresponding primary decomposition. First we illustrate this for modules without embedded primes.

Proposition 2.2.2 *Let* $\mathbf{N} \in \mathbf{M}$ *be as above and assume that* $\operatorname{Ass} \mathbf{M}/\mathbf{N} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_m\}$ *contains only isolated primes. For* $i, j = 1, \ldots, m$ *we take separators* $f_i \in \mathbf{R}$ *such that* $f_i \in \mathfrak{p}_j$ *if* $i \neq j$*, but* $f_i \notin \mathfrak{p}_i$.

Then

$$\mathbf{N} = \bigcap (\mathbf{N} :_{\mathbf{M}} (f_i)^{\infty})$$

is a (minimal) primary decomposition of $\mathbf{N}$ *in* $\mathbf{M}$.

Proof: This is an immediate consequence of Lemma 1.3.2. □

If $\mathbf{N}$ has embedded primes, we can not separate the primes from each other completely. But we can decompose $\mathbf{N}$ in a set of submodules of $\mathbf{M}$ such that each of them has only one minimal associated prime.

Definition 2.2.3 A submodule $\mathbf{N}$ is called a *quasi-$\mathfrak{p}$-primary* submodule in $\mathbf{M}$, if $\mathbf{N}$ has a unique isolated prime $\mathfrak{p}$ (and possibly embedded primes).

Following Gräbe we can generalize the algorithm of to the case of modules.

Proposition 2.2.4 *Let $\mathbf{N} \subseteq \mathbf{M}$ be defined as above and assume that $L := \{\mathfrak{p}_1, \ldots, \mathfrak{p}_k\}$ are the isolated primes of $\mathbf{M}/\mathbf{N}$. Take f_i as separators of the $\mathfrak{p}_i$, $\mathbf{N}_i := \mathbf{N} :_{\mathbf{M}} (f_i)^\infty$ and $e_i \in \mathbf{M}/\mathbf{N}$ such that $f_i^{e_i}\mathbf{N}_i \subseteq \mathbf{N}$.*
Then:

1. *$\mathbf{N}_i$ is a quasi-$\mathfrak{p}_i$-primary module in $\mathbf{M}$.*
2. *The sets $A_i := \operatorname{Ass}\mathbf{M}/\mathbf{N}_i = \{\mathfrak{p} \in \operatorname{Ass}\mathbf{M}/\mathbf{N} : f_i \notin \mathfrak{p}\}$ are pairwise disjoint.*
3. *For $(J) := \left(f_1^{e_1}, \ldots, f_k^{e_k}\right)$ we have*

$$\mathbf{N} = \left(\bigcap \mathbf{N}_i\right) \cap (\mathbf{N} + \mathbf{J} \cdot \mathbf{M}).$$

This is a decomposition of $\mathbf{N}$ into quasi-primary components $\mathbf{N}_i$ and a component $\mathbf{N}' := \mathbf{N} + \mathbf{J} \cdot \mathbf{M} \subseteq \mathbf{M}$ of lower (relative) dimension.

Proof:

1. Let $\mathbf{N} = \bigcap \mathbf{M}_i$ be a minimal primary decomposition. By definition $f_i \in \mathfrak{p}_j$ for $i \neq j$ and $f_i \notin \mathfrak{p}_i, \forall i$. Therefore f_i vanishes on all associated primes of $\mathbf{M}/\mathbf{N}$, not embedded in or equal to $\mathfrak{p}_i$ (and it may also vanish on some of the primes strictly containing $\mathfrak{p}_i$). By Lemma 1.3.2 a stable quotient with respect to f_i cuts off all such components:

$$\mathbf{N}_i = \mathbf{N} :_{\mathbf{M}} (f_i)^\infty = \bigcap \{\mathbf{M}_j : f_i \notin \mathfrak{p}_j\}$$

   Hence $\mathfrak{p}_i$ is the unique isolated prime of $\mathbf{N}_i$.

2. At most one of the separators is not contained in $\mathfrak{p} \in \operatorname{Ass}\mathbf{M}/\mathbf{N}$ by construction of the f_i.

3. Since $\mathbf{N} \subseteq \mathbf{N}' \subseteq \mathbf{M}$ we conclude that $\dim \mathbf{M}/\mathbf{N}' \leq \dim \mathbf{M}/\mathbf{N}$. Equality does not occur, because $\mathfrak{p} \not\supseteq \mathbf{J}$ and therefore $(\mathbf{M}/\mathbf{N}')_\mathfrak{p} = 0, \forall \mathfrak{p} \in L$.

   The remaining assertion follows as in the case of ideals: First we claim, that $\mathbf{N}_i :_{\mathbf{M}} \left(f_j^{e_j}\right) = \mathbf{M}, \forall j \neq i$:

We have $\mathbf{N}_i :_{\mathbf{M}} \left(f_j^{e_j}\right) \supseteq \mathbf{N} :_{\mathbf{M}} \left(f_j^{e_j}\right) = \mathbf{N}_j$ (since $\mathbf{N}_i \supseteq \mathbf{N}$) and $\mathbf{N}_i :_{\mathbf{M}} \left(f_j^{e_j}\right) \supseteq \mathbf{N}_i$ (by the definition of the quotient). We conclude by Lemma 1.1.5:

$$\begin{aligned} \operatorname{Ass}\left(\mathbf{M}/\mathbf{N}_i :_{\mathbf{M}} \left(f_j^{e_j}\right)\right) &\supseteq \operatorname{Ass} \mathbf{M}/\mathbf{N}_i = A_i \\ \operatorname{Ass}\left(\mathbf{M}/\mathbf{N}_i :_{\mathbf{M}} \left(f_j^{e_j}\right)\right) &\supseteq \operatorname{Ass} \mathbf{M}/\mathbf{N}_j = A_j \\ \Rightarrow \operatorname{Ass}\left(\mathbf{M}/\mathbf{N}_i :_{\mathbf{M}} \left(f_j^{e_j}\right)\right) &\supseteq A_i \cap A_j = \emptyset \end{aligned}$$

Hence $\mathbf{M}/\mathbf{N}_i :_{\mathbf{M}} \left(f_j^{e_j}\right)$ has no associated primes and therefore the module $\mathbf{N}_i :_{\mathbf{M}} \left(f_j^{e_j}\right)$ can only be equal to $\mathbf{M}$.

If $x \in (\bigcap \mathbf{N}_i) \cap (\mathbf{N} + \mathbf{J} \cdot \mathbf{M})$, then $x = n + \sum_j f_i^{e_i} m_j \in \bigcap \mathbf{N}_i$, with $n \in \mathbf{N}, m_j \in \mathbf{M}$. Furthermore $f_j^{e_j} m_j \in \mathbf{N}_i$ because $\mathbf{N}_i :_{\mathbf{M}} \left(f_j^{e_j}\right) = \mathbf{M}, \forall j \neq i$. But then $f_i^{e_i} m_i = x - n - \sum_{i \neq j} f_i^{e_i} m_j \in \mathbf{N}_i$. Hence $f_i^{e_i} m_i \in \mathbf{N} : (f_i^{e_i})$ and so $f_i^{2e_i} m_i \in \mathbf{N}, \forall i$, which means that $m_i \in (\mathbf{N} :_{\mathbf{M}} f_i^{\infty}) = (\mathbf{N} :_{\mathbf{M}} f_i^{e_i})$. Thus $f_i^{e_i} m_i \in \mathbf{N} \forall i$ and finally $x = n + \sum_i f_i^{e_i} m_i \in \mathbf{N}$.

□

It remains to decompose the quasi-primary components $\mathbf{N}_i$. Here we apply reduction to dimension zero once more. So let us assume that $\mathbf{M}/\mathbf{N}$ has a unique isolated prime $\mathfrak{p}$. Choose a maximal independent set $\{x_\nu, \nu \in V\}$ for $\mathbf{N} \in \mathbf{M}$ and let $\widetilde{\mathbf{N}}$, $\widetilde{\mathbf{M}}$ etc. be as before the extension modules.

Lemma 2.2.5 *Assume that B is a Gröbner basis of $\mathbf{N}$ with respect to an inverse module term block order with respect to V on $\mathbf{F}$, $B' \in B$ a denominator-free Gröbner basis for $\widetilde{\mathbf{N}}$ and $c \in \mathbf{R}$ the product of the leading coefficients of the elements of B' regarded as polynomial vectors in $\widetilde{\mathbf{F}}$.*

Then

1. $$\mathbf{N}' := \widetilde{\mathbf{N}} \cap \mathbf{M} = \mathbf{N} :_{\mathbf{M}} (c)^{\infty}$$

   is the (uniquely determined) $\mathfrak{p}$-primary component of $\mathbf{N}$ in $\mathbf{M}$.

2. *If e is an integer such that $c^e \cdot \mathbf{N}' \subseteq \mathbf{N}$, then*

   $$\mathbf{N} = \mathbf{N}' \cap (\mathbf{N} + c^e M)$$

   is a decomposition of $\mathbf{N}$ into a $\mathfrak{p}$-primary component and another module of lower (relative) dimension.

Proof:

1. By Lemma 1.5.8 $\mathbf{N}'$ is equal to the $\mathfrak{p}$-primary component of $\mathbf{N}$.

2. The decomposition follows again by Lemma 1.5.8. It is left to prove that $\dim \mathbf{M}/\mathbf{N}'' < \dim \mathbf{M}/\mathbf{N}$. But this follows from $\overline{(\mathbf{M}/\mathbf{N}'')_{\mathfrak{p}}} = 0$ since $c^e \notin \mathfrak{p}$. The latter is true, because $\mathfrak{p} = \sqrt{\operatorname{Ann} \mathbf{M}/\mathbf{N}}$ and $\sqrt{\operatorname{Ann} \mathbf{M}/\mathbf{N}} \cap (\mathbb{K}[x_\nu : \nu \in V]/\{0\}) = \emptyset$.

□

Lemma 2.2.5 shows how to reduce the primary decomposition problem to the decomposition of quasi-primary ideals. Hence we can state the following procedure (cf. Gräbe [Grä]):

Algorithm 2.2.5

PrimDecA($\mathbf{N}$,$\mathbf{M}$)

Input: A submodule $\mathbf{N} \subseteq \mathbf{M}$
Output: A primary decomposition of $\mathbf{N}$ in $\mathbf{M}$

- Compute $L := \{\mathfrak{p}_1, \ldots, \mathfrak{p}_k\}$, the list of isolated primes of $\mathbf{M}/\mathbf{N}$ as isolated primes of $\operatorname{Ann} \mathbf{M}/\mathbf{N}$.
- For $i := 1, \ldots, k$ compute polynomials $f_i \in \mathbf{R}$ separating $L \backslash \{\mathfrak{p}_i\}$ from $\mathfrak{p}_i$.
- For $i := 1, \ldots, k$ compute the quasi-primary components $\mathbf{N}_i := \mathbf{N} :_{\mathbf{M}} (f_i)^\infty$ as stable quotients and integers e_i such that $f_i^{e_i} \cdot \mathbf{N}_i \subseteq \mathbf{N}$.
- return
  $\bigcup_i$ **PrimDecB**$(\mathbf{N}_i, \mathbf{M}, \mathfrak{p}_i) \cup$ **PrimDecA**$(\mathbf{N}' := \mathbf{N} + (f_1^{e_1}, \ldots, f_k^{e_k})\mathbf{M}, \mathbf{M})$

Lemma 2.2.5 gives us a procedure for the decomposition of quasi-primary ideals.

Algorithm 2.2.6

PrimDecB($\mathbf{N}$,$\mathbf{M}$,$\mathfrak{p}$)

Input: A quasi-primary submodule $\mathbf{N} \subseteq \mathbf{M}$, such that $\mathbf{M}/\mathbf{N}$ has a unique isolated prime $\mathfrak{p}$
Output: A primary decomposition of $\mathbf{N}$ in $\mathbf{M}$

- Find a maximal relative independent set $\{x_\nu, \nu \in V\}$ for $\mathbf{N} \subseteq \mathbf{M}$.
- Compute a Gröbner basis B of $\mathbf{N}$ with respect to an inverse module term block order with respect to V.
- Change to $\widetilde{\mathbf{R}}$, extract a minimal Gröbner basis $B' \subseteq B$ of $\widetilde{\mathbf{N}}$ and compute $c \in \mathbf{R}$, the product of the leading coefficients of the elements of B' regarded as polynomial vectors in $\widetilde{\mathbf{F}}$.
- Compute $\mathbf{N}' := \mathbf{N} :_{\mathbf{M}} (c)^\infty$ and an integer e such that $c^e \mathbf{N}' \subseteq \mathbf{N}$.
- return $\{(\mathbf{N}', \mathfrak{p}\} \cup$ **PrimDecA**$(\mathbf{N} + c^e \cdot \mathbf{M}, \mathbf{M})$.

Using these results we can formulate the following procedures in the language of Singular:

2.2.1 Minimal Primary Decomposition

The primary decomposition computed by the algorithm need not to be minimal. We can obtain a primary decomposition with pairwise different primes, if we collect all the $\mathfrak{p}$-primary components with the same $\mathfrak{p}$ and substitute them by their intersection. But this decomposition may not be minimal at all. The set of the computed primes may contain superfluous primes.

Some kind of minimality test is given by the following proposition.

Proposition 2.2.6 *Let* $\{(\mathbf{N}_i, \mathbf{M}, \mathfrak{p}_i) : i = 1, \ldots, m\}$ *be a collection of* $\mathfrak{p}_i$*-primary modules* $\mathbf{N}_i \subseteq \mathbf{M}$*, with* $\mathfrak{p}_i \neq \mathfrak{p}_j, \forall i \neq j$*, such that* $\mathbf{N} = \bigcap \mathbf{N}_i$ *is an possibly redundant primary decomposition of* $\mathbf{N}$ *in* $\mathbf{M}$*, with pairwise different primes* $\mathfrak{p}_i$*.*

Let $f \in \mathbf{R}$ *separate* $\{\mathfrak{p}_j \not\subseteq \mathfrak{p}_i\}$ *and compute* $\mathbf{M}_i := \mathbf{N} :_{\mathbf{M}} (f)^\infty$*.*

Then $\mathfrak{p}_i \notin \mathrm{Ass}\,(\mathbf{M}/\mathbf{N})$ *if and only if* $\mathbf{M}_i = \mathbf{M}_i :_{\mathbf{M}} \mathfrak{p}_i^\infty$*.*

Proof: f separates $\{\mathfrak{p}_j \not\subseteq \mathfrak{p}_i\}$ from $\mathfrak{p}_i$. Hence by Lemma 1.3.2 $\mathbf{M}_i = \mathbf{N} :_{\mathbf{M}} (f)^\infty = \bigcap_{\{\mathfrak{p}_j \subseteq \mathfrak{p}_i\}} \mathbf{N}_j$. Again by Lemma 1.3.2 we conclude that $\mathbf{M}_i : \mathfrak{p}_i^\infty = \bigcap_{\{\mathfrak{p}_j \subseteq \mathfrak{p}_i\}} \{\mathbf{N}_j : \mathfrak{p}_i \not\subseteq \mathfrak{p}_j\} = \bigcap_{\{\mathfrak{p}_j \not\subseteq \mathfrak{p}_i\}} \mathbf{N}_j$. Therefore $\mathfrak{p}_i$ is a superfluous prime if and only if $\mathbf{M}_i = \mathbf{M}_i : \mathfrak{p}_i^\infty$. □

We can write this as a procedure in SINGULAR as follows:

Chapter 3

Implementation of the Algorithms

In this chapter we compare different implementations of the algorithms decribed in Chapter 2. First we discuss various improvements of the algorithm based on Gianni, Trager and Zacharias, then we give detailed timing examples. Based an these we compare the implementations of the Algorithms 2.1.4 and 2.2.5 given in Section 2.1.3 and 2.2.1 on the one hand and the additions of Section 3.1 on the other.

3.1 Improvements of the Algorithm

In the algorithm of Gianni, Trager and Zacharias we use random coordinate change to obtain modules in general position. But random coordinate change destroys sparseness and usually makes Gröbner basis computation very difficult. So to make this algorithm really efficient, it is necessary to do some preprocessing to avoid random coordinate changes whenever possible. We can generalize most of the tools known from the primary decomposition of ideals to the case of modules. To illustrate the effect of these enhancements easy examples are given and computed on the following test system: a Linux-PC with AMD K6-III processor (400MHz), running SINGULAR 1-3-8.

3.1.1 Splitting of a Module

In this section we compute submodules (not necessarily primary) $\mathbf{N}_1, \ldots, \mathbf{N}_k$ of $\mathbf{M}$, such that $\mathbf{N} = \mathbf{N}_1 \cap \cdots \cap \mathbf{N}_k$ and the primary decomposition of the $\mathbf{N}_i$ is easier to compute than the primary decomposition of $\mathbf{N}$. First we formulate a module version of the splitting tools given in [DGP, Chapter 2]:

Lemma 3.1.1 (Splitting Tools) *Let* $\mathbf{N}$ *be a submodule of* $\mathbf{M}$.

1. $\mathbf{N} = (\mathbf{N} :_{\mathbf{M}} f^e) \cap (N + f^e \cdot \mathbf{M})$ *if* $\mathbf{N} :_{\mathbf{M}} f^e = \mathbf{N} :_{\mathbf{M}} f^{e+1}$ *for some* $f \in \mathbf{R}$.
2. $\mathbf{N} = (\mathbf{N} + f \cdot \mathbf{M}) \cap (\mathbf{N} + f \cdot \mathbf{M})$ *if* $f \cdot g \in \operatorname{Ann} \mathbf{M}/\mathbf{N}$ *and* $(f, g) = (1)$.

Proof:

1. We prove analogously to the proof of Lemma 1.5.8:

   Since $\mathbf{N} \subseteq (\mathbf{N} :_{\mathbf{M}} f^e)$ and $\mathbf{N} \subseteq (\mathbf{N} + f^e \cdot \mathbf{M})$, we only have to show the inclusion $(\mathbf{N} :_{\mathbf{M}} f^e) \cap (N + f^e \cdot \mathbf{M}) \subseteq \mathbf{N}$. Let $x \in (\mathbf{N} :_{\mathbf{M}} f^e) \cap (N + f^e \cdot \mathbf{M})$. Then $x = n + f^e m$, with $n \in \mathbf{N}, m \in \mathbf{M}$. Since x is also contained in $\mathbf{N} : f^\infty$, we have that $f^e x \in \mathbf{N}$ and therefore $f^e x = f^e n + f^{2e} m \in \mathbf{N}$. Hence $m \in \mathbf{N} : f^\infty = \mathbf{N} : f^e$, which means that $f^e m \in \mathbf{N}$, and finally $x = n + f^e m \in \mathbf{N}$.

2. Again we only need to show $(\mathbf{N}+f\cdot\mathbf{M})\cap(\mathbf{N}+g\cdot\mathbf{M}) \subseteq \mathbf{N}$. Let $u, v \in \mathbf{R}$ with $u\cdot f+v\cdot g = 1$, and let $x \in (\mathbf{N}+f\cdot\mathbf{M})\cap(\mathbf{N}+g\cdot\mathbf{M})$. Then there exist $n_1, n_2 \in \mathbf{N}$ and $m_1, m_2 \in \mathbf{M}$, such that $x = n_1 + f \cdot m_1 = n_2 + g \cdot m_2$. Hence we can write:

$$\begin{aligned} f \cdot m_1 \cdot 1 &= f \cdot m_1 \cdot (u \cdot f + v \cdot g) \\ &= (n_2 - n_1 + g \cdot m_2)(u \cdot f + v \cdot g) \\ &= (n_2 - n_1) \cdot u \cdot f + (f \cdot g)(v \cdot m_1 + u \cdot m_2) \end{aligned}$$

   Since $f \cdot g \in \operatorname{Ann} \mathbf{M}/\mathbf{N}$ the last term is in $\mathbf{N}$ and therefore $f \cdot m_1$. Thus $x = n_1 + f \cdot m_1 \in \mathbf{N}$.

□

We can now factorize generators of $\operatorname{Ann} \mathbf{M}/\mathbf{N}$ and split the module as often as possible, before starting the algorithm. If the condition $(f_1, \ldots, f_k) = (1)$ is not fulfilled for the factors $f_1, \ldots, f_k$ of f, we can still apply the second tool to a suitable power of f_1. Using Lemma 3.1.1 we produce as many reducible modules as possible. This leads to the following preprocessing algorithm (cf. [DGP, Algorithm 9]):

Algorithm 3.1.7

Split($\mathbf{N}$)

Input:	A zero-dimensional submodule $\mathbf{N} \subseteq \mathbf{M} = \mathbb{K}[x_1, \ldots, x_n]^s$
Output:	Two sets of of modules, Primary $= \{(\mathbf{N}_1, \mathfrak{p}_1), \ldots, (\mathbf{N}_r, \mathfrak{p}_k)\}$ and Rest $= \{\mathbf{M}_1, \ldots, \mathbf{M}_k\}$, such that $\mathbf{N} = (\bigcap \mathbf{N}_i) \cap (\bigcap \mathbf{M}_i)$, $\mathbf{N}_i$ primary with associated prime $\mathfrak{p}_i$.

- Let Primary := $\emptyset$ and Rest := $\emptyset$.

- Factorize all generators of $\operatorname{Ann}\mathbf{M}/\mathbf{N}$ and split $\mathbf{N}$ and the resulting modules as often as possible.
- For each module obtained this way compute a Gröbner basis with respect to $>_{lex}$ induced by $x_1 > \cdots > x_n$
- Test whether the modules are primary and in general position to x_n: Put detected primary modules to Primaryand the other modules to Rest
- return Primary, Rest

Since we do not need to decompose primary modules, it is useful to test for primarity after every step. To avoid the difficult primary test of Lemma 2.1.4 we formulate a few lemmas, which help us to recognize some kinds of primary submodules. By Lemma 1.4.1 we only need to check the annihilator of a zero-dimensional module for primarity.

The following lemma decribes the prime test for zero-dimensional ideals:

Lemma 3.1.2 (Prime Test) *A zero-dimensional ideal* $\mathbf{I}$ *is prime and in general position with respect to* $>_{lex}$ *induced by* $x_1 > \cdots > x_n$ *if and only if for a corresponding minimal Gröbner basis* G *with* $\{g\} = G \cap \mathbb{K}[x_n]$*, we have* $\deg(g) = \dim_{\mathbb{K}} \mathbb{K}[x_1, \ldots, x_n]/\mathbf{I}$ *and* g *is irreducible.*

Proof: If $\mathbf{I}$ is in general position and prime then $\mathbf{I} = (x_1 - g_1(x_n), \ldots, x_{n-1} + g_{n-1}(x_n), g_n(x_n))$, with $g_n(x_n)$ irreducible and $k = \deg(g_n(x_n))$. Hence $G \cap \mathbb{K}[x_n] = \{g_n(x_n)\}$ and the elements of $\mathbb{K}[x_1, \ldots, x_n]/\mathbf{I}$ (considered as a $\mathbb{K}$-vector space) are generated by $1, x_n, x_n^2, \ldots, x_n^{k-1}$. Therefore $\dim_{\mathbb{K}} \mathbb{K}[x_1, \ldots, x_n]/\mathbf{I} = \deg(g_n(x_n))$.

Conversely let g be such that $\deg(g(x_n)) = \dim_{\mathbb{K}} \mathbb{K}[x_1, \ldots, x_n]/\mathbf{I}$ with g irreducible of degree k. Then we have an inclusion of the vector spaces $\langle 1, x_n, \ldots, x_n^{k-1}\rangle \subseteq \mathbb{K}[x_1, \ldots, x_n]/\mathbf{I}$ with $k = \dim_{\mathbb{K}} \langle 1, x_n, \ldots, x_n^{k-1}\rangle \leq \dim_{\mathbb{K}} \mathbb{K}[x_1, \ldots, x_n]/\mathbf{I} = k$. Hence the vector spaces are equal and therefore $x_i \in \langle 1, x_n, \ldots, x_n^{k-1}\rangle \bmod \mathbf{I}$ for all $i = 1, \ldots, n-1$. Thus there exists polynomials $g_1, \ldots, g_{n-1} \in \mathbb{K}[x_n]$ such that $x_i = g_i(x_n) \bmod \mathbf{I}$. Hence the ideal $\mathfrak{m} = (x_1 - g_1(x_n), \ldots, x_{n-1} - g_{n-1}(x_n), g(x_n))$ is contained in $\mathbf{I}$. Since $\mathfrak{m}$ is maximal, we have $\mathbf{I} = \mathfrak{m}$ and therefore $\mathbf{I}$ is prime and in general position. □

Corollary 3.1.3 *Let* $\mathbf{N}$ *be a zero-dimensional submodule of* $\mathbf{M}$*,* $\mathfrak{a} = \operatorname{Ann}\mathbf{M}/\mathbf{N}$ *and* g *be a polynomial in* $\mathbb{K}[x_n]$ *such that* $(g) = \mathfrak{a} \cap \mathbb{K}[x_n]$ *with the factorization* $g = f_1^{e_1} \cdot \ldots \cdot f_l^{e_l}$*. If* $\deg(g(x_n)) = \dim_{\mathbb{K}} \mathbb{K}[x_1, \ldots, x_n]/\mathfrak{a}$ *then* $\mathbf{N} = (\mathbf{N} + f_1^{e_1} \cdot \mathbf{M}) \cap \ldots \cap (\mathbf{N} + f_l^{e_l} \cdot \mathbf{M})$ *is a (minimal) primary decomposition of* $\mathbf{N}$ *with associated primes* $\mathfrak{p}_i = (\mathfrak{a}, f_i)$*.*

Proof: By Lemma 3.1.1(1) we get $\mathbf{N} = (\mathbf{N} + f_1^{e_1} \cdot \mathbf{M}) \cap \cdots \cap (\mathbf{N} + f_l^{e_l} \cdot \mathbf{M})$. The annihilator of $\mathbf{M}/(\mathbf{N} + f_i^{e_i} \cdot \mathbf{M})$ is $\mathfrak{a} + (f_i^{e_i})$ for $i = 1, \ldots, l-1$. As in the proof

of Lemma 3.1.2 we can find polynomials $g_{i_1}, \ldots, g_{i_{n-1}}$ for $i = 1, \ldots, l-1$, such that $(x_1 - g_{i_1}(x_n), \ldots, x_{n-1} - g_{i_{n-1}}(x_n), f_i^{e_i}(x_n)) \subseteq \mathfrak{a} + (f_i^{e_i}) \subseteq \mathfrak{p}_i$. Hence $\mathfrak{a} + (f_i^{e_i})$ is primary with associated prime $\mathfrak{p}_i$.

□

For some purposes it is be useful to consider the following lemma instead of Lemma 3.1.2:

Lemma 3.1.4 *Let* $\mathbf{I}$ *be a zero-dimensional ideal and* G *be a minimal Gröbner basis with respect to* $>_{lex}$*. If* g *is an irreducible element of* G *with* $\deg(g) = \dim_{\mathbb{K}} \mathbb{K}[x_1, \ldots, x_n]/\mathbf{I}$ *then* $\mathbf{I}$ *is prime.*

Proof: Since the G is a minimal Gröbner basis, $m \notin \mathrm{Lt}(\mathbf{I})$ for all monomials m with $m | \mathrm{Lt}(g)$. Hence we have an inclusion of vector spaces: $\langle \{m \in \mathbb{K}[x_1, \ldots, x_n] : m | \mathrm{Lt}\, g\} \rangle \in \mathbb{K}[x_1, \ldots, x_n]/\mathbf{I}$. Let k be the degree of g and assume that there exist $x_i, x_j, i \neq j$ such that $x_j, x_j | \mathrm{Lt}(g)$. Then $\#\{m : m | \mathrm{Lt}\, g\} \cup \{1\} > k$, which is a contradiction to $\dim_{\mathbb{K}} \mathbb{K}[x_1, \ldots, x_n]/\mathbf{I} = k$. Therefore $\mathrm{Lt}(g) = a x_i^k$ for some ring variable x_i and $\mathbb{K}[x_1, \ldots, x_n]/\mathbf{I} = \left\langle 1, x_i, \ldots, x_i^{k-1} \right\rangle$. Thus there exist polynomials $g_j \in \mathbb{K}[x_i]$ such that $\mathbf{I} = (x_1 - g_1(x_i), \ldots, x_{i-1} - g_{i-1}(x_i), x_{i+1} - g_{i+1}(x_i), x_{n-1} - g_{n-1}(x_i), g)$ and therefore $\mathbf{I}$ is prime. □

Another check for primary modules is based on a test for homogeneous ideals:

Lemma 3.1.5 *Let* $\mathbf{I}$ *be a zero-dimensional ideal of* $\mathbf{R}$*. If there exists a homogeneous ideal* $\mathbf{J}$ *such that* $\mathbf{I} \subseteq \mathbf{J} \subseteq \sqrt{\mathbf{I}}$ *then* $\mathbf{I}$ *is primary with associated prime* $(x_1, \ldots, x_n)$*.*

Proof: $\mathbf{J}$ is also zero-dimensional, hence there exist $g_i \in \mathbf{J} \cap \mathbb{K}[x_i]$ for all $i = 1, \ldots, n$. We can write $g_i = a_k x_i^k + \cdots + a_0$ with $a_j \in \mathbf{R}$ for $j = 0, \ldots, k$. Since $\mathbf{J}$ is a homogenous ideal, the homogeneous components $a_j x_i^{e_j}$ of g_i are contained in $\mathbf{J}$. Therefore $x_i \in \sqrt{\mathbf{J}} = \sqrt{\mathbf{I}}$ for all $i = 1, \ldots, n$. Thus $\sqrt{\mathbf{I}} = (x_1, \ldots, x_n)$ and $\mathbf{I}$ is primary. □

Now, we can use the Lemmas 3.1.2 and 3.1.5 to formulate the following a priori test for primarity.

Algorithm 3.1.8

primTest($\mathbf{I}$, $\mathbf{J}$,f)

Input: Two zero-dimensional ideals $\mathbf{I}$, $\mathbf{J}$ in $\mathbb{K}[x_1, \ldots, x_n]$ such that $\mathbf{I} \subseteq \mathbf{J} \subseteq \sqrt{\mathbf{I}}$ and a polynomial $f \in \mathbf{I}$.

Output: If $\deg(f) = \dim_{\mathbb{K}} \mathbb{K}[x_1, \ldots, x_n]/\mathbf{I}$ then $\mathbf{I}$ is returned, else if $\mathbf{I}$ or $\mathbf{J}$ is homogeneous $\sqrt{\mathbf{I}}$, otherwise (0)

- Let $\mathfrak{p} := (0)$ the zero-ideal in $\mathbb{K}[x_1, \ldots, x_n]$.

- if $\deg(f) = \dim_{\mathbb{K}} \mathbb{K}[x_1, \ldots, x_n]/\mathbf{I}$ then $\mathfrak{p} := \mathbf{I}$
  else
  - if $\mathbf{I}$ or $\mathbf{J}$ is homogeneous then $\mathfrak{p} := (x_1, \ldots, x_n)$.
- return $\mathfrak{p}$.

In the language of Singular we obtain:

Using the *split* and the *primTest* algorithm, we can write the following procedure:

Remark Since $\mathbf{N}$ may be in general position with respect to x_n by itself, it is useful to test whether the obtained modules are primary and in general position after the first splitting (cf. Appedix A, procedure `preComp`, line 1158).

Example

Let $\mathbf{R} = \mathbb{Z}_{32003}[x, y, z, u, v]$, $p = x^4 + y^4 + z^4 + u^4 + v^4 + (x + y + z + u + v)^4$ be a polynomial in $\mathbf{R}$ and $\mathbf{I} = \text{JacobiIdeal}(p)$. The computation time on the test system without *splitting* is 38 seconds, but only 5 seconds otherwise.

3.1.2 Simplification of a Module

Let $\mathfrak{a}$ be an ideal in the ring $\mathbb{K}[x_1, \ldots, x_n]$. If $\mathfrak{a}$ contains a linear polynomial l with leading monomial x_i (with respect to a suitable ordering), we can reduce the number of generators and the number of ring variables and decompose $\mathfrak{a}' = \mathfrak{a}/(l) \in \mathbb{K}[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$ instead of $\mathfrak{a}$. A primary decomposition of $\mathfrak{a}/(l) = \mathfrak{q}_1 \cap \cdots \cap \mathfrak{q}_k$ leads to a primary decomposition of $\mathfrak{a}$ namely $\mathfrak{a} = (\mathfrak{q}_1, l) \cap \cdots \cap (\mathfrak{q}_k, l)$.

We will generalize this fact to the case of submodules of $\mathbf{M}$ using the following lemma:

Lemma 3.1.6 *Let $\mathbf{N}$ be a submodule of $\mathbf{M}$ and $\operatorname{Ann} \mathbf{M}/\mathbf{N}$ contains a linear polynomial l with leading monomial x_i (with respect to a suitable ordering). If $\mathbf{N}_1 \cap \cdots \cap \mathbf{N}_k$ is a minimal primary decomposition of $\mathbf{N}' = \mathbf{N}/(l \cdot \mathbf{M})$ in $\mathbf{M}' = \mathbf{M} \cap \mathbb{K}[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]^s$ then $\mathbf{N} = (\mathbf{N}_1 + l \cdot \mathbf{M}) \cap \cdots \cap (\mathbf{N}_k + l \cdot \mathbf{M})$ is a minimal primary decomposition of $\mathbf{N}$.*

Proof: Since $l \in \operatorname{Ann} \mathbf{M}/\mathbf{N}$, the module $l \cdot \mathbf{M}$ is a submodule of $\mathbf{N}$ and hence $\mathbf{N} = (\mathbf{N}' + l \cdot \mathbf{M})$ and $\mathbf{N} = (\mathbf{N}_1 + l \cdot \mathbf{M}) \cap \cdots \cap (\mathbf{N}_k + l \cdot \mathbf{M})$. Since $\mathbf{N}_i$ is a module in $\mathbb{K}[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]^s$ we can write $\mathbf{M}'/\mathbf{N}_i = (\mathbf{M}/l \cdot \mathbf{M})/((\mathbf{N}_i + l \cdot \mathbf{M})/l \cdot \mathbf{M})$ and the latter is isomorphic to $\mathbf{M}/(\mathbf{N}_i + l \cdot \mathbf{M})$ by the *Second Isomophism Theorem*. By Lemma 1.1.5 we may assume $\operatorname{Ass} \mathbf{M}'/\mathbf{N}_i = \operatorname{Ass} \mathbf{M}/(\mathbf{N}_i + l \cdot \mathbf{M})$ and therefore $(\mathbf{N}_i + l \cdot \mathbf{M})$ is primary. The

minimality of the decomposition in $\mathbf{M}$ follows from the minimality in $\mathbf{M}'$. □

This leads to the following algorithm:

Algorithm 3.1.9

Simplification($\mathbf{N}$)

Input: A submodule $\mathbf{N} \subseteq \mathbf{M} = \mathbb{K}[x_1, \ldots, x_n]^s$
Output: A minimal primary decomposition of $\mathbf{N} \subseteq \mathbf{M}$

- Let $\operatorname{Ann} \mathbf{M}/\mathbf{N} = (g_1, \ldots, g_k)$, $\mathbf{I} = \emptyset$
- for $i := 1, \ldots, k$
  - for $j := 1, \ldots, n$
    if $\deg(g_i) = 1$ and $\operatorname{Lt}(g_i) = x_j$ then $\Sigma = \Sigma \cup \{g_i\}$
- if $\#(\Sigma) = n$ then return $\{\mathbf{N}, \operatorname{Ann} \mathbf{M}/\mathbf{N}\}$.
- if $\#(\Sigma) > 0$ then then map κ is defined by:
  $x_\nu \mapsto x_\nu - \frac{1}{\operatorname{lc}(g_\mu)} \cdot g_\mu$, if $x_\nu = \operatorname{Lt}(g_\mu)$ for a $g_\mu \in \Sigma$
  $x_\nu \mapsto x_\nu$, else.
- if $\#(\Sigma) = 0$ then κ is the identity map.
- Let $\{\mathbf{N}_1, \mathfrak{p}_1\}, \ldots, \{\mathbf{N}_m, \mathfrak{p}_m\}$ be a minimal primary decomposition of $\kappa(\mathbf{N})$.
- return $\{\mathbf{N}_1 + \Sigma \cdot \mathbf{M}, \mathfrak{p}_1\}, \ldots, \{\mathbf{N}_m + \Sigma \cdot \mathbf{M}, \mathfrak{p}_m\}$.

Now we can add the following lines at the beginning of the procedure of section 2.1.3. If we use this addition in every iteration of the procedure, this addition would cause an infinite loop. Hence we may only use this in the first iteration of the algorithm. We can be sure about this by checking for the non-existence of the module *check* (given by the optional parameter #).

The Case of One Variable

Since we are reducing the number of ring variables successively by one as much as possible, we may reach the special case, when $\mathbf{R}$ is a polynomial ring with one variable. In this case it would be useful to have special procedure. Since ideals in the ring $\mathbb{K}[x_1]$ are trivially zero-dimensional and in general position with respect to x_1, we can skip the reduction to dimension zero, random coordinate change and the primarity test, and formulate the following algorithm:

Algorithm 3.1.10

decOneVar(N)

Input: A submodule $\mathbf{N} \subseteq \mathbf{M} = \mathbb{K}[x_1]^s$
Output: The primary decomposition of $\mathbf{N}$ in $\mathbf{M}$

- Let $\mathbf{I} := \operatorname{Ann} \mathbf{M}/\mathbf{N}$ and G such that $(G) := \mathbf{I}$
- factorize $G = g_1^{\nu_1} \cdot \ \ldots \ \cdot g_t^{\nu_t}$
- for $i = 1$ to t do

  $h_i := \prod_{j \neq i} g_j^{\nu_j}$
  $\mathbf{N}_i := \mathbf{N} : h_i, \mathfrak{p}_i := (\mathbf{I}, g_i)$
  Result = Result $\cup \{(\mathbf{N}_i, \mathfrak{p}_i)\}$
- return Result

This in implemented in the following procedure:

Example

The following easy example demonstrates, how useful these subalgorithms are: Let $\mathbf{I} = (x_1 - 1, \ldots, x_{n-2} - 1, g(x_{n-1}), x_n - 1) \subseteq \mathbf{R}^1$ be an ideal with $g \in \mathbb{K}[x_{n-1}]$ a polynomial, which is reducible.

By Lemma 3.1.6 we only need to decompose the ideal $\mathbf{I}' = (g(x_{n-1})) \subseteq \mathbb{K}[n_{n-1}]$. Using the decomposition for one variable we immediately get the primary decomposition: $\mathbf{I} = ((g_1) + \mathfrak{a}) \cap \cdots \cap ((g_k) + \mathfrak{a})$ with $g = g_1 \cdot \ \ldots \ \cdot g_k$ the factorization of g and $\mathfrak{a} = (x_1 - 1, \ldots, x_{n-2} - 1, x_n - 1)$.

Using the test system (take e. g. $n = 11$ and $g = (x_{n-1}^3 - x_{n-1})^4$) the computation lasts for two minutes without the subalgorithm, but doesn't need any recognizable time with the subalgorithm.

Remark Obviously the ideal $\mathbf{I}$ is in general position with respect to x_{n-1}, but the algorithm tries to find a linear coordinate change such that the obtained module is in general position with respect to x_n. But this is very hard to compute in this case.

3.1.3 Minimal Saturation

In section 1.5 we use saturation to compute the recontractions of modules. We have seen that we can find a polynomial $c \in \mathbf{R}$ such that $\mathbf{N} = \mathbf{N}' \cap (\mathbf{N} + c^e \cdot \mathbf{M})$ with $\mathbf{N}' := \widetilde{\mathbf{N}} \cap \mathbf{M} = \mathbf{N} :_{\mathbf{M}} (c)^\infty$ and e is an integer such that $c^e \cdot \mathbf{N}' \subseteq \mathbf{N}$. But c^e may be of large degree and so the decomposition of $(\mathbf{N} + c^e \cdot \mathbf{M})$ will become very diffithecult. Therefore we need to compute a factor g of c^e such that g is of minimal degree with $\mathbf{N} :_{\mathbf{M}} (c)^\infty = \mathbf{N} :_{\mathbf{M}} (g)$. Then $\mathbf{N} = \mathbf{N}' \cap (\mathbf{N} + g \cdot \mathbf{M})$.

We can compute g with the following procedures:

3.1.4 Minimal Standard basis

We know from Theorem 1.5.5, that a Gröbner basis B of the module $\mathbf{N}$ is also a Gröbner basis of the extension module $\widetilde{\mathbf{N}}$. But B is usually not a minimal Gröbner basis. But additional generators make computation of the primary decomposition difficult. The following lemma gives the idea for a procedure, which deletes superfluous elements from a standard basis.

Lemma 3.1.7 *Let* $\mathbf{N}$ *be a submodule of* $\mathbf{M}$ *and* B *be a Gröbner basis of* $\mathbf{N}$ *with respect to a well-ordering.* g *is a superfluous element of the Gröbner basis if and only if there exists an element* $f \in B$ *such that* $\mathrm{Lt}(f) | \mathrm{Lt}(g)$.

Proof: Let g be a superfluous element. Hence $\mathrm{Lt}(g) \in (\mathrm{Lt}(G \backslash \{g\}))$ and therefore there exists $f \in G$ such that $\mathrm{Lt}(f) | \mathrm{Lt}(g)$.

Conversely let $\mathrm{Lt}(f) = a_1 x^{\underline{\mu}} e_\nu$, $\mathrm{Lt}(g) = x^{\underline{\nu}} x^{\underline{\mu}} e_\nu$ and $f = a_1 x^{\underline{\mu}} + r_1, g = a_2 x^{\underline{\nu}} x^{\underline{\mu}} + r_2$. Since we are using a well-ordering, we can choose r_1, r_2 such that $\mathrm{Lt}(f)$ resp. $\mathrm{Lt}(g)$ do not divide the terms of r_1, resp. r_2. Then $s := x^{\underline{\nu}} f - g = x^{\underline{\nu}} r_1 - r_2 \in (B)$ by *Buchberger's Criterion for Submodules* (e. g. [CLO2, Theorem 2.9]). Since terms of s are not divisble by $\mathrm{Lt}(g)$ we have $s \in (B \backslash \{g\})$ and therefore $g = x^{\underline{\nu}} f - s \in (B \backslash \{g\})$. Hence $(B) = (B \backslash \{g\})$. Furthermore we know that $\mathrm{Lt}(g) = \mathrm{Lt}(f)$ and hence $(\mathrm{Lt}(B)) = (\mathrm{Lt}(B \backslash \{g\}))$. □

Using this test we can formulate an algorithm to extract a minimal Gröbner bases from a given Gröbner basis. If the basis elements f and g have the same leading term, we have two choices. To make the further computation easier we will keep the element with the smaller number of terms. This motivates to formulate the following algorithm:

Algorithm 3.1.11

ClearStandardBasis(G)

Input: A Gröbner basis $G = \{g_1, \ldots, g_k\}$, such that $\mathrm{Lt}(g_1) \leq \cdots \leq \mathrm{Lt}(g_k)$
Output: A minimal Gröbner basis $G' \subseteq G$

- Let $G' = G$.
- for (g_i, g_j) with $g_i, g_j \in G'$ such that $i < j$
  - Let $m_i \cdot e_{\nu_i} = \mathrm{Lt}(g_i)$ and $m_j \cdot e_{\nu_j} = \mathrm{Lt}(g_j)$.
  - If $\nu_i = \nu_j$ and $m_i | m_j$ then
    - if $\mathrm{Lt}(g_i) = \mathrm{Lt}(g_j)$ and g_i, g_j have the same number of terms. then $G' = G' \backslash \{g_i\}$
      else $G' = G' \backslash \{g_j\}$
- return G'.

In the language of Singular we obtain:

Example

The following examples illustrates, how useful the extraction of a minimal Gröbner basis is:

Let $\mathbf{R} = \mathbb{Z}_{32003}[a,b,c,d,e,f]$ and $\mathbf{I} \in \mathbf{R}$ be an ideal such that $\mathbf{I} = (a^2cdf^2, b^2cdf^2, a^2bdf^2, b^3df^2, a^3df^2, ab^2df^2, a^2cde, b^2cde, a^2bde, b^3de, a^3de, ab^2de, a^2cd^2, b^2cd^2, a^2bd^2, b^3d^2, a^3d^2, ab^2d^2, a^2c^2f^2, b^2c^2f^2, a^2bcf^2, b^3cf^2, a^3cf^2, ab^2cf^2, a^2b^2f^2, b^4f^2, a^3bf^2, ab^3f^2, a^4f^2)$. $\mathbf{I}$ has five primary components of dimension 4, 4, 3, 3, resp. 2 and corresponding associated primes $\mathfrak{p}_1 \subset \mathfrak{p}_3$ and $\mathfrak{p}_2 \subset \mathfrak{p}_4 \subset \mathfrak{p}_5$. The plain version of the algorithm needs 2.5 seconds to compute the primary decomposition. If we add the procedure above, the computation time is 1.2 seconds. This is caused by the fact, that a Gröbner basis of $\mathbf{I}$ in $\mathbf{R}$ considered as a denomiator-free Gröbner basis of $\widetilde{\mathbf{I}}$ in $\widetilde{\mathbf{R}}$, has many redundant generators. E. g. if we take the maximal independent set $V = \{a,b,c,e\}$ then $\widetilde{\mathbf{I}} = (df^2, d, f^2)$.

3.1.5 Maximal Independent Sets

In section 1.5 the dimension of the submodule $\mathbf{N}$ of $\mathbf{M}$ is reduced to zero by using the extension $\widetilde{\mathbf{N}}$ of $\mathbf{N}$ defined by the ring map $\mathbb{K}[x_1,\ldots,x_n] \hookrightarrow \mathbb{K}(x_\nu, \nu \in V)[x_\mu, \mu \notin V]$ for a maximal independent set V. To get a complete primary decomposition we decompose $\mathbf{N}$ into the equidimensional module $\mathbf{N}' = \widetilde{\mathbf{N}} \cap \mathbf{M}$ and another module $\mathbf{N}''$. If $\mathbf{N}$ is a quasi-primary module then $\mathbf{N}''$ is of lower relative dimension by Lemma 2.2.5. But we cannot be sure about this in general. Indeed, if the $\dim \mathbf{M}/\mathbf{N}'' = \dim \mathbf{M}/\mathbf{N}$ all maximal independent sets of $\mathbf{N}''$ are maximal independent sets of $\mathbf{N}$, but V is not a maximal independent set of $\mathbf{N}''$ any more. It is easier to compute all maximal independent sets (and even the independent sets, which cannot be enhanced) in the first step of the algorithm than computing a maximal independent set at each step of the algorithm. We can enhance the algorithm of section 2.1.3 as follows:

Algorithm 3.1.12

GTZallInd(N,M)

Input: A submodule $\mathbf{N} \subseteq \mathbf{M}$
Output: The primary decomposition von $\mathbf{N}$ in $\mathbf{M}$

- Let $\mathcal{V}$ the set of all maximal independent sets of $\operatorname{Ann} \mathbf{M}/\mathbf{N}$, $\mathbf{N}' = \mathbf{N}$ and result= $\emptyset$.
- While $\dim(\mathbf{N}') = \dim(\mathbf{N})$
  - Take $V \in \mathcal{V}$ and denote $\{x_\nu, \nu \in V\}$ by v.
  - Change ring to $\mathbb{K}(v)[x \backslash v]$.

- Compute a Gröbner Basis $B = \{G_1, \ldots, G_s \in (\mathbb{K}[x_1, \ldots, x_n])^s\}$ of $\mathbf{N}'$ (in $\mathbb{K}(v)[x \backslash v]$).
- $h := \prod \mathrm{lc}(G_i)$
- Compute e such that $(B) : h^e = (B) : h^{e+1}$
- $\{(\mathbf{N}_i, \mathfrak{p}_i)\} :=$ **ZeroDecMod**$(\mathbf{N}')$
- Result = Result $\cup \{(\mathbf{N}_i \cap \mathbf{M}, \mathfrak{p}_i \cap \mathbb{K}[x_1, \ldots, x_n])\}$
- $\mathbf{N}' = \mathbf{N}' + h^e \cdot \mathbf{M}$ and $V = V \backslash \{v\}$

- Result = Result $\cup$ **GTZallInd**$(\mathbf{N}')$
- return Result

After using all maximal independent sets, we can also use the independent sets, which cannot be enhanced, to reduce $\mathbf{N}'$ to a zero-dimensional module (cf. procedure `GTZopt` in Appendix A, line 572).

Example

Let $\mathbf{R} = \mathbb{Q}[x_1, \ldots, x_{10}, y_1, \ldots, y_{10}]$ and consider the module $\mathbf{N}$

$$\mathbf{N} = \left(\begin{pmatrix} x_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} 0 \\ \vdots \\ 0 \\ x_{10} \end{pmatrix}, \begin{pmatrix} y_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} 0 \\ \vdots \\ 0 \\ y_{10} \end{pmatrix} \right) \subseteq \mathbf{R}^{10}.$$

The module $\mathbf{N}$ is of dimension 18 and has ten maximal independent sets $V_i = \{x_1, \ldots, x_{10}, y_1, \ldots, y_{10}\} \backslash \{x_i, y_i\}, \forall i = 1, \ldots, 10$ and decomposes into 10 primary modules:

$$(e_j, (x_i, y_i) \cdot e_i, j \neq i), \forall i = 1, \ldots, 10, \text{ with } e_\nu \text{ the } \nu^{\text{th}} \text{ unit vector}$$

If we compute only one maximal independent set in the first step of the algorithm, the computation time using the test system is 37 seconds. If we compute and save all maximal independent sets in the beginning, the time is 22 seconds.

3.2 Comparison of the Implementations

In this section we compare the implementations of the algorithms, discussed in Chapter 2 and the last section. The comparison is based on a series of 11 suitable examples. They are computed again on a Linux-PC (with AMD K6-III 400MHz processor) running Singular 1-3-8. Some of the examples are ideals of $\mathbb{K}[x_1, \ldots, x_n]$, which had been used to test the primary decomposition of ideals. In this case we can compare the timing results

with timings using procedures of the Singular-Library *primdec.lib*. We can generate examples of modules $\mathbf{N} \subseteq \mathbf{R}^s$ for $s > 1$ using ideal examples: $\mathbf{N} = \mathbf{I} \times \mathbf{J} \subseteq \mathbf{R}^2$ and $\mathbf{N} = \left(\mathfrak{a} \cdot \mathbf{R}^2 + \mathbf{I} \times \mathbf{J}\right) \subseteq \mathbf{R}^2$, for some ideals $\mathfrak{a}, \mathbf{I}, \mathbf{J} \subseteq \mathbf{R}$ Finally we derive examples from matrices with mathematical or physical meaning.

Examples

1. Let $\mathbf{R} = \mathbb{Z}_{32003}[a,b,c,d,e,f]$ and $\mathbf{I} \in \mathbf{R}$ be an ideal such that $\mathbf{I} = (a^2cdf^2, b^2cdf^2, a^2bdf^2, b^3df^2, a^3df^2, ab^2df^2, a^2cde, b^2cde, a^2bde, b^3de, a^3de, ab^2de, a^2cd^2, b^2cd^2, a^2bd^2, b^3d^2, a^3d^2, ab^2d^2, a^2c^2f^2, b^2c^2f^2, a^2bcf^2, b^3cf^2, a^3cf^2, ab^2cf^2, a^2b^2f^2, b^4f^2, a^3bf^2, ab^3f^2, a^4f^2)$

   $\mathbf{I}$ has five primary components of dimension 4, 4, 3, 3, resp. 2 and corresponding associated primes $\mathfrak{p}_1 \subset \mathfrak{p}_3$ and $\mathfrak{p}_2 \subset \mathfrak{p}_4 \subset \mathfrak{p}_5$.

2. $\mathbf{R} = \mathbb{Z}_{32003}[a,b]$, $\mathbf{N} = \left(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} a \\ -a^2 \end{pmatrix}\right)$

   $\mathbf{N}$ is equidimensional of dimension 5 and has two components:

$$\left(\begin{pmatrix} a \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right), \left(\begin{pmatrix} a^2+b \\ 0 \end{pmatrix}, \begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} -1 \\ a \end{pmatrix}\right)$$

3. $\mathbf{R} = \mathbb{Z}_{32003}[x,y,z,u,v]$, $p = x^4+y^4+z^4+u^4+v^4+(x+y+z+u+v)^4$, $\mathbf{I} =$ JacobiIdeal(p). It decomposes into 15 zero-dimensional components and one embedded component of dimension 1.

4. $\mathbf{R} = \mathbb{Q}[x,y,z]$, $\mathbf{I} = (x^2y, xz^2, y^2z, z^3)$ (cf. [Grä]). $\mathbf{I}$ has three primary components of dimension 1, 1, resp. 0 with corresponding associated primes $\mathfrak{p}_1 \subset \mathfrak{p}_3$ and $\mathfrak{p}_2 \subset \mathfrak{p}_3$.

5. $\mathbf{R} = \mathbb{Q}[x,y,z]$, $\mathbf{J} = (xy, y^2, yz)$ (cf. [Grä]) $\mathbf{J}$ has two primary components of dimension 2, resp. 0, with embedded associated primes

6. $\mathbf{I} \times \mathbf{J} = (x^2y, xz^2, y^2z, z^3) \times (xy, y^2, yz) \subseteq \mathbb{Q}[x,y,z]^2$. $\mathbf{N}$ has four primary components of dimension 2, 1, 1, resp. 0, with corresponding associated primes $\mathfrak{p}_1 \subseteq \mathfrak{p}_3 \subseteq \mathfrak{p}_4$ and $\mathfrak{p}_2 \subseteq \mathfrak{p}_4$.

7. $\mathbf{R} = \mathbb{Z}_{32003}[a,b,c,d,e,f]$, $\mathbf{N} = (de+cf, be+af, bcf-adf, f^3g+e^2f, cdf^2g-c^2ef, adf^2g-acef, abf^2g-a^2ef, cd^2fg+c^3f, ad^2fg+ac^2f, abdfg+a^2cf, ab^2fg+a^3f) \times (de+cf, be+af, bcf-adf, f^3g+e^2f, d^2f^2g+c^2f^2, bdf^2g+acf^2, b^2f^2g+a^2f^2) \subseteq \mathbb{Z}_{32003}[a,b,c,d,e,f]^2$.

   $\mathbf{N}$ has six primary components of dimension 5, 4, 4, 4, 3, 3 with corresponding associated primes $\mathfrak{p}_1 \subseteq \mathfrak{p}_4 \subseteq \mathfrak{p}_5$, $\mathfrak{p}_2 \subseteq \mathfrak{p}_5$ and $\mathfrak{p}_1 \subseteq \mathfrak{p}_6$.

8. $\mathbf{R} = \mathbb{Q}[x,y,z]$, $\mathbf{N} = (xy^2z^2 - xy^2z + xyz^2 - xyz, x^2yz^2 - x^2yz, xy^3z + xy^2z, x^2y^2z, xy^4 - xy^2, x^2y^3 - x^2y^2, x^3y^2 + x^3yz - x^3y, x^4y - 2x^3yz + 3x^3y + 3x^2y^2 - 16001xy^3 - 16001xy^2, x^4z^3 - 16001x^4yz - x^4z^2 + 2x^3z^3 + 4x^2yz^3 + 16001x^4y - 16000x^3yz - 2x^3z^2 - 4x^2yz^2 + x^2z^3 + 16000x^3y + 16000x^2y^2 - 8001xy^3 - x^2z^2 - 8001xy^2, x^5z^2 + 8000x^4yz + 2x^4z^2 - 8000x^4y - 8003x^3yz + x^3z^2 - 12x^2yz^2 + 8003x^3y + 8003x^2y^2 - 4000xy^3 + 12x^2yz - 4000xy^2, x^7z + 2x^6z - 2x^4z - x^3z) \times (xy^3z + xy^2z, x^2y^2z, xy^4 - xy^2, x^2y^3 - x^2y^2, xy^2z^3 - xy^2z^2 + xyz^3 - xyz^2, x^2yz^3 - x^2yz^2, x^4yz - x^3yz^2 + 2x^3yz - x^2yz^2 + x^2yz, x^5z + x^4z^2 + x^4z + 2x^3z^2 - x^3z + x^2z^2 - x^2z, x^4y^2 + 3x^3y^2 + 3x^2y^2 - 16001xy^3 - 16001xy^2, x^4z^3 - x^4z^2 + 2x^3z^3 + 4x^2yz^3 - 2x^3z^2 - 4x^2yz^2 + x^2z^3 - x^2z^2, x^6y + 3x^5y + 3x^4y + x^3y)$

   $\mathbf{N}$ has one primary component of dimension 2, four of dimension 1 and five of dimension 0 with corresponding associated primes $\mathfrak{p}_1 \subseteq \mathfrak{p}_3$; $\mathfrak{p}_1 \subseteq \mathfrak{p}_4 \subseteq \mathfrak{p}_8, \mathfrak{p}_9$; $\mathfrak{p}_2 \subseteq \mathfrak{p}_7, \mathfrak{p}_9, \mathfrak{p}_{10}$; $\mathfrak{p}_5 \subseteq \mathfrak{p}_6, \mathfrak{p}_{10}$.

9. The tensor product of two vectors $\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$ and $\begin{pmatrix} b_1 \\ b_2 \\ b_2 \end{pmatrix}$ is defined by the matrix $\begin{pmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{pmatrix}$.

   This motivates to decompose the module

$$\mathbf{N} = \left(\begin{pmatrix} a_1b_1 \\ a_2b_1 \\ a_3b_1 \end{pmatrix}, \begin{pmatrix} a_1b_2 \\ a_2b_2 \\ a_3b_2 \end{pmatrix}, \begin{pmatrix} a_1b_3 \\ a_2b_3 \\ a_3b_3 \end{pmatrix} \right) = \left(\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \right) \cap (b_1, b_2, b_3) \cdot \mathbf{R}^3$$

10. The inertial tensor of a point mass (of mass 1) at the point (x,y,z) motivates to consider the module

$$\mathbf{N} = \left(\begin{pmatrix} y^2+z^2 \\ xy \\ xz \end{pmatrix}, \begin{pmatrix} xy \\ x^2+z^2 \\ yz \end{pmatrix}, \begin{pmatrix} xz \\ yz \\ x^2+y^2 \end{pmatrix} \right).$$

    $\mathbf{N}$ has three primary components of dimension 2 with associated primes $(x), (y), (z)$.

11. If we identify a_1, a_2, a_3 with a, b, c and b_1, b_2, b_3 with x, y, z we can intersect the modules of Example 9 and 10:

$$\mathbf{N} = \left(\begin{pmatrix} x^3y^2z^2c \\ x^2y^3z^2c \\ x^2y^2z^3c \end{pmatrix}, \begin{pmatrix} x^3y^2z^2b \\ x^2y^3z^2b \\ x^2y^2z^3b \end{pmatrix}, \begin{pmatrix} x^3y^2z^2a \\ x^2y^3z^2a \\ x^2y^2z^3a \end{pmatrix} \right)$$

In the table below we give the timings (in seconds) for the examples above computed on the test system. "∞" means that the computation was stopped after 600 seconds. In the first rows the timings for the computation of a minimal primary decomposition by different implementations are given.

		1	2	3	4
a)	module SY	3.80	0.16	6.12	0.18
b)	SY mit fact Groebner	2.33	0.18	6.34	0.12
c)	not opt GTZ	2.51	0.04	38.04	0.08
d)	+ clrSBmod	1.17	0.06	34.42	0.09
e)	+ precomp (split)	2.71	0.05	5.23	0.08
f)	+ minsatmod	2.80	0.06	38.19	0.10
g)	+ Simpl	4.71	0.05	38.10	0.10
h)	+ all max indep	1.23	0.06	36.03	0.10
i)	opt GTZ	1.20	0.06	4.12	0.10
j)	ideal by GTZ	1.30	-	1.69	0.10
k)	ideal by SY	0.55	-	13.13	0.10
l)	minAssPrime(ann)	1.63	0.06	1.90	0.10
m)	with fact Groebner	0.08	0.08	1.55	0.04

	5	6	7	8	9	10	11
a)	0.26	0.38	18.66	1.46	1.40	0.58	∞
b)	0.19	0.23	18.74	1.06	1.41	0.51	16.87
c)	0.06	0.16	3.48	8.99	1.60	0.21	261.38
d)	0.06	0.22	1.30	11.53	0.90	0.22	3.30
e)	0.07	0.18	3.84	11.02	1.80	0.20	∞
f)	0.07	0.26	1.11	11.42	0.50	0.33	3.12
g)	0.09	0.24	6.97	10.19	3.14	0.31	287.12
h)	0.08	0.19	0.98	12.70	0.40	0.24	3.26
i)	0.07	0.21	1.00	8.99	0.43	0.25	2.78
j)	0.10	-	-	-	-	-	-
k)	0.09	-	-	-	-	-	-
l)	0.10	0.18	-	0.47	-	0.11	0.43
m)	0.02	0.04	-	0.07	-	0.05	0.54

In three (of eleven) examples the algorithm based on the approach of Shimoyama and Yokoyama is faster than the plain algorithm based on Gianni, Trager and Zacharias, in particular if the factorizing Gröbner basis algorithm is used (cf. [Grä]) to compute the minimal associated primes. The factorizing Gröbner algorithm is usually very efficient (but can be very time-consuming in a few cases, cf. [DGP]). But if we add the improvements of Section 3.1 to the plain algorithm, usually the resulting procedure becomes faster, even faster than the procedure based on Shimoyama/Yokoyama. But the examples show, that there is no unique strategy for the computation of the primary decomposition of modules. This coincides with the results known from the primary decomposition of ideals. (cf. [DGP]).

Appendix A

The Singular Library
modec.lib

Table of Procedures

```
// $Id: modec.lib $
////////////////////////////////////////////////////////////////////////////
// modec.lib                                                                //
// algorithms for primary decomposition for modules based on                //
// the algorithms of Gianni, Trager and Zacharias and                       //
// Shimoyama and Yokoyama (generalization suggested by                      //
// the ideas of Hans-Gert Gräbe, Leipzig  )                                 //
// using elments of primdec.lib                                            //
// written by Alexander Dreyer                                              //
//                                                                          //
////////////////////////////////////////////////////////////////////////////

version="$Id: modec.lib,v 1.00 2001/04/26 dreyer Exp $";

info="
LIBRARY: modec.lib   PROCEDURES FOR PRIMARY DECOMPOSITION OF MODULES
AUTHORS:  Alexander Dreyer, dreyer@mathematik.uni-kl.de; adreyer@web.de

PROCEDURES:
  sep(l);                      computes a list of separators of prime ideals
  PrimdecA(N[,i]);             (not necessarily minimal) primary decomposition
                               via Shimoyama/Yokoyama (suggested by Graebe)
  PrimdecB(N, p);              (not necessarily minimal) primary decomposition
                               for pseudo-primary ideals
  modDec(N[, i]);               minimal primary decomposition
                               via Shimoyama/Yokoyama (suggested by Graebe)
  zeroMod(N[,check]);          minimal zero-dimensional primary decomposition
                               via Gianni, Trager and Zacharias
  GTZmod(N[,check]);           minimal primary decomposition
                               via Gianni, Trager and Zacharias
  dec1var(N[,check[, ann]]);   primary decomposition for one variable
  annil(N);                    the annihilator of M/N in the basering
  splitting(N[,check[, ann]]); splitting to simpler modules
  primTest(i[, p]);           tests whether i is prime or homogeneous
  preComp(N, check[, ann]);    enhanced Version of splitting
  indSet(i);                   lists with varstrings of(in)dependend variables
  GTZopt(N[,check[, ann]]);    a faster version of GTZmod
  zeroOpt(N[,check[, ann]]);   a faster version of zeroMod
  clrSBmod(N);                 extracts an minimal SB from a SB
  minSatMod(N, I);             minimal saturation of N w.r.t. I
  specialModulesEqual(N1,N2);  checks for equality of standard bases of modules
                               if N1 is contained in N2 or vice versa
  stdModulesEqual(N1,N2);      checks for equality of standard bases
  ModulesEqual(N1,N2);         checks for equality of modules
  getData(N, l[,i]);           extracts oldData and computes the remaining data

REMARK:
These procedures are implemented to be used in characteristic 0.
@*They also work in positive characteristic >> 0.
@*In small characteristic and for algebraic extensions, the
procedures cia Gianni Trage, Zacharias may not terminate.
 ";

LIB "primdec.lib";
LIB "ring.lib";
LIB "standard.lib";


////////////////////////////////////////////////////////////////////////////
// sep(l)
// computes a list of separators for a list of prime ideals
// it in a generalization of parts of pseudo_prim_dec_i from primdec.lib
```

```
////////////////////////////////////////////////////////////////////////////

proc sep (list @L)
"USAGE:   sep(l); list l of prime ideals
RETURN:  list sepList;
         a list of separators of the prime ideals in l,
         i.e. polynomials p_ij, s.th. p_ij is in l[j],
         for all l[j] not contained in l[i]
         but p_ij is not in l[i]
EXAMPLE: example sep; shows an example
"

{
  ideal p_ij;                   // generating p_ij in @L[@j], not in @L[@i]
  list f_i;                     // generating f_i NOT in @L[@i], but in all @L[@j]
  int @i,@j,@k;
  int sizeL=size(@L);

  for(@i=1;@i<=sizeL;@i++)
  {
    p_ij=0;
    @L[@i]=std(@L[@i]);
    for(@j=1;@j<=sizeL;@j++)            // compute the seperator sep_i
                                        // of the i-th component
                                        // f_i separates {Pj not incl in Pi} from Pi
    {
      if (@i!=@j)                       // searching for g: not in @L[@i], but @L[@j]
      {
        for(@k=1;@k<=ncols(@L[@j]);@k++)
        {
          if(NF(@L[@j][@k],@L[@i],1)!=0)
          {
            p_ij=p_ij+@L[@j][@k];
            break;
          }
        }
      }
    }
  f_i=f_i+list(lcm(p_ij));
  }
  return(f_i);
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,(x,y,z),dp;
  ideal i=(x2y,xz2,y2z,z3);
  list l=minAssPrimes(i,1);
  list sepL=sep(l);
  sepL;
}

////////////////////////////////////////////////////////////////////////////
// PrimdecA(N[,i])
// computes a primary decomposition, not necessarily minimal,
// using a generalization of the algorithm of Schimoyama/Yokoyama,
// suggested by Hans-Gerd Graebe, Leipzig
// [Hans-Gert Graebe, Minimal Primary Decompostion
//  and Factorized Groebner Bases, AAECC 8, 265-278 (1997)]
////////////////////////////////////////////////////////////////////////////


proc PrimdecA(module @N, list #)
```

```
"USAGE:   PrimdecA (N[, i]); module N, int i
RETURN:  list l
         a (not necessarily minimal) primary decomposition of N
         computed by a generalized version of
         the algorithm of Schimoyama/Yokoyama,
         if i=1 is given, the factorizing Groebner is used
         to compute the isolated primes.
EXAMPLE: example PrimdecA; shows an example
"

{
  module @M=freemodule(nrows(@N));        // @M=basering^k
  ideal ann=annil(@N);                    // the annihilator of @N
  ////////////////////////////////////////////////////////////////
  // in the trivial case we avoid to compute anything
  ////////////////////////////////////////////////////////////////
  if (ann[1]==1)
  {
    return(list());
  }
  ////////////////////////////////////////////////////////////////
  // Computation of the Associated Primes
  ////////////////////////////////////////////////////////////////
  if (ann[1]==0)
  {
    list pr=list(ideal(0));
  }
  else
  {
    list pr = minAssPrimes(ann, #);
    // causes message  "/ ** redefining @res **" if #[1]=1
  }
  list sp, pprimary;     // the separators and the pseudo-primary modules
  int @i;
  ideal rest;            // for the computation of the remaining components
  sp=sep(pr);
  int sizeSp=size(sp);   // the number of separators

  ////////////////////////////////////////////////////////////////
  // Computation of the pseudo-primary modules
  // and an ideal rest s.th. @N is the intersection of the
  // pseudo-primary modules and @N+rest*@M
  ////////////////////////////////////////////////////////////////
  for(@i=1;@i<=sizeSp;@i++)
  {
    pprimary=pprimary+list(sat(@N,sp[@i]));
    rest=rest+sp[@i]^pprimary[@i][2];
  }
  list result;           // a primary decomposition of @N

  ////////////////////////////////////////////////////////////////
  // Extraction of the pseudo-primary modules
  ////////////////////////////////////////////////////////////////
  for (@i=1;@i<=size(pprimary);@i++)
  {
   result=result+PrimdecB(pprimary[@i][1],pr[@i]);
  }

  ////////////////////////////////////////////////////////////////
  // Computation of remaining components
  ////////////////////////////////////////////////////////////////
  result=result+PrimdecA(@N+rest*@M);
```

```
  return(result);
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,(x,y,z),dp;
  module N=x*gen(1)+ y*gen(2),
           x*gen(1)-x2*gen(2);
  list l=PrimdecA(N);
  l;
}


////////////////////////////////////////////////////////////////////////////
// PrimdecB(N, p)
// computes a primary decomposition, not necessarily minimal,
// of a pseudo-primary module N with isolated prime p
// it is based on extraction in primdec.lib
////////////////////////////////////////////////////////////////////////////

proc PrimdecB(module @N, ideal isoPrim)
"USAGE:   PrimdecB (N, p); pseudo-primary module N, isolated prime ideal p
RETURN:  list l
         a (not necessarily minimal) primary decomposition of N
EXAMPLE: example PrimdecB; shows an example
"

{
  module @M=freemodule(nrows(@N));        // @M=basering^k
  ideal ann=annil(@N);                    // the annihilator of @N

  ////////////////////////////////////////////////////////////////
  // the not-that-trivial case of ann==0
  ////////////////////////////////////////////////////////////////
  if(size(ann)==0)
  {
    def BAS=basering;
    execute("ring Rloc=("+charstr(basering)+","+varstr(basering)+"),
                                  dummy,("+ordstr(basering)+");");
    module @N=imap(BAS, @N);
    poly @q=prepareSat(@N);

    setring BAS;
    poly @q=imap(Rloc, @q);
    list satu=sat(@N,@q);
    if(satu[2]==0)
    {
      return(list(list(@N,ideal(0))));
    }
    else
    {
      return(list(list(satu[1],ideal(0)))+ PrimdecA(@N+(@q^satu[2])*@M));
    }
  }

  ////////////////////////////////////////////////////////////////
  // Extraction of the isolated component @N' and
  // searching for a polynomial @f of minimal degree
  // s.th. @N=intersect(@N', @N+@f*@M)
  ////////////////////////////////////////////////////////////////
  list indSets=indepSet(ann,0);
  poly @f;
```

```
  if(size(indSets)!=0)          //check, whether dim isoPrim !=0
250 {
    intvec indVec;              // a maximal independent set of variables
                                // modulo isoPrim
    string @U;                  // the independent variables
    string @A;                  // the dependent variables
    int @j,@k;
    int szA;                    //  the size of @A
    int degf;
    ideal @g;
    list polys;
260 int sizePolys;
    list newPoly;

    ////////////////////////////////////////////////////////////
    // the preparation of the quotient ring
    ////////////////////////////////////////////////////////////
    def BAS=basering;
    for (@k=1;@k<=size(indSets);@k++)
    {
      indVec=indSets[@k];
270   for (@j=1;@j<=nvars(BAS);@j++)
      {
        if (indVec[@j]==1)
        {
          @U=@U+varstr(@j)+",";
        }
        else
        {
          @A=@A+varstr(@j)+",";
          szA++;
280     }
      }

      @U[size(@U)]=")";           // we compute the extractor (w.r.t. @U)
      execute("ring RAU="+charstr(basering)+",("+@A+@U+",(C,dp("+string(szA)+"),dp);");
      module @N=std(imap(BAS,@N));
                                  // this is also a standard basis in (R[U])[A]
      @A[size(@A)]=")";
      execute("ring Rloc=("+charstr(basering)+","+@U+",("+@A+",(C,dp);");
      ideal @N=imap(RAU,@N);
290   ideal @h;
      for(@j=ncols(@N);@j>=1;@j--)
      {
        @h[@j]=leadcoef(@N[@j]);  // consider I in (R(U))[A]
      }
      setring BAS;
      @g=imap(Rloc,@h);
      kill RAU,Rloc;
      @U="";
      @A="";
300   szA=0;
      @f=lcm(@g);
      newPoly[1]=@f;
      polys=polys+newPoly;
      newPoly=list();
    }
    @f=polys[1];
    degf=deg(@f);
    sizePolys=size(polys);
    for (@k=2;@k<=sizePolys;@k++)
310 {
```

```
      if (deg(polys[@k])<degf)
      {
        //Wählt das poly mit dem geringsten Grad.
        @f=polys[@k];
        degf=deg(@f);
      }
    }
  }
  else
  {
    @f=1;
  }
  if(@f!=1)
  {
    list satu = minSatMod(@N,@f);
    return(list(list(satu[1],isoPrim))+ PrimdecA(@N+satu[2]*@M));
    //    list satu = sat(@N,@f);
    //return(list(list(satu[1],isoPrim))+ PrimdecA(@N+@f^satu[2]*@M));
  }
  else
  {
    return(list(list(@N,isoPrim)));
  }
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,(x,y,z),dp;
  module N=y*gen(1),y2*gen(2),yz*gen(2),yx*gen(2);
  ideal p=y;
  list l=PrimdecB(N,p);
  l;
}

////////////////////////////////////////////////////////////////////////////
// modDec(N[,i])
// extracts a minimal primary decomposition from the
// primary decomposition computed by PrimdecA(N[, i])
// using a Minimality Test suggested by Hans-Gerd Graebe, Leipzig
////////////////////////////////////////////////////////////////////////////

proc modDec(module @N, list #)
"USAGE:   modDec (N[, i]); module N, int i
RETURN:  list l
         a minimal primary decomposition of N
         computed by an generalized version of
         the algorithm of Schimoyama/Yokoyama,
         if i=1 is given, the factorizing Groebner is used
EXAMPLE: example modDec; shows an example
"
{
  list prim = PrimdecA(@N, #);
  int @i,@j,@k,@l;
  int sizePrim=size(prim);

  ////////////////////////////////////////////////////////////
  // the trivial case
  ////////////////////////////////////////////////////////////
  if (sizePrim==0)
  {
    return(list(list(@N, ideal(1))));
  }
```

```
  ////////////////////////////////////////////////////////////
  // collect primary components with the same associated prime
  // and substitute them by their intersection
  ////////////////////////////////////////////////////////////
  for(@i=1;@i<=sizePrim;@i++)
  {
    for(@j=@i+1;@j<=sizePrim;@j++)
    {
      if (@j!=@i)
      {
        if (idealsEqual(prim[@i][2],prim[@j-@l][2])==1)
        {
          prim[@i][1]=intersect(prim[@i][1],prim[@j-@l][1]);
          prim=delete(prim,@j-@l);
          @l++;
          sizePrim--;
        }
      }
    }
  }

  ////////////////////////////////////////////////////////////
  // minimality test suggested by Graebe:
  // if f separates the prime p from the primes not contained in p,
  // then p is not in Ass(M/N) <=> sat(N,f)=sat(sat(N,f),p)
  ////////////////////////////////////////////////////////////
  list tempList;
  for(@i=1;@i<=sizePrim;@i++)
  {
    tempList=tempList+list(prim[@i][2]);
  }
  list sepPrimes=sep(tempList);   // the separators of the primes
  tempList=list();
  for(@i=1;@i<=sizePrim;@i++)     // compute sat(N,f), for all separators f
  {
    tempList=tempList+list(sat(@N,sepPrimes[@i])[1]);
  }
  module testMod;
  @k=0;
  for(@i=1;@i<=sizePrim;@i++)
  {
    testMod=sat(tempList[@i],prim[@i-@k][2])[1];  // computes sat(sat(N,f),p)
    if (size(NF(testMod,std(tempList[@i]),1))==0) // tests if equal to sat(N,f)
    {
      prim=delete(prim,@i-@k);                    // if yes: the component
      @k++;                                       // is superfluous
    }
  }

  return(prim);
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,(x,y,z),dp;
  module N=x*gen(1)+ y*gen(2),
           x*gen(1)-x2*gen(2);
  list l=modDec(N);
  l;
}

////////////////////////////////////////////////////////////////////////////
// zeroMod (N[, check])
```

```
// computes a minimal primary decomposition of a zero-dimensional Module
// using a generalized version of the algorithm of
// Gianni, Trager and Zacharias, suggested by Alexander Dreyer
// [Diploma Thesis, University of Kaiserslautern, 2001]
////////////////////////////////////////////////////////////////////////////
440
proc zeroMod (module @N, list #)
"USAGE:   zeroMod (N[, check]); zero-dimensional module N[, module check]
RETURN:  list l
         the minimal primary decomposition of a zero-dimensional module N,
         computed by a gernalized version of the algorithm
         of Gianni, Trager and Zacharias
NOTE:    if the parameter check is given, only components
         not containing check are computed
EXAMPLE: example zeroMod; shows an example
450 "
{
  ///////////////////////////////////////////////////////////////
  // the module check is needed to compute a minimal decomposition
  // components containing check are ignored
  ///////////////////////////////////////////////////////////////
  if (size(#)>0)
  {
    module check=#[1];
    if (size(NF(check,std(@N),1))==0)
460 {
      return(list());
    }
  }
  else
  {
    module check=freemodule(nrows(@N));
  }

  ///////////////////////////////////////////////////////////////
470 // the ordering is changed to lex
  ///////////////////////////////////////////////////////////////
  def BAS = basering;
  changeord("@R","lp");
  module @N=fetch(BAS,@N);
  int nVar=nvars(@R);
  module @M=freemodule(nrows(@N));     // @M=basering^k
  ideal ann=std(quotient(@N,@M));      // the annihilator of @M/@N
  int @k;
  list result, rest;
480 ideal primary, prim;
  module primMod;

  ///////////////////////////////////////////////////////////////
  // the random coordnate change and its inverse
  ///////////////////////////////////////////////////////////////
  option(redSB);
  ideal prepMap=maxideal(1);
  prepMap[nVar]=0;
  prepMap[nVar]=(random(100,1,nVar)*transpose(prepMap))[1,1]+var(nVar);
490 map phi=@R,prepMap;
  prepMap[nVar]=2*var(nVar)-prepMap[nVar];
  map invphi=@R,prepMap;

  ideal @j=std(phi(ann));

  list fac=factorize(@j[1],2);   // factorization of the 1st elt. in Ann(@N)
```

```
//////////////////////////////////////////////////////////////
// Case: 1st element irreducible
//////////////////////////////////////////////////////////////
if(size(fac[2])==1)
{
  prim=primaryTest(@j,fac[1][1]);
  prim=invphi(prim);
  setring BAS;
  @N=std(@N);
  ideal prim=std(imap(@R,prim));
  kill @R;
  if(prim!=0)
  {
    return(list(list(@N,prim)));
  }
  else
  {
    return(zeroMod(@N,check));
  }
 }

//////////////////////////////////////////////////////////////
// Computation of the - hopefully primary - modules
// their annihilators and associated primes
//////////////////////////////////////////////////////////////
poly @p, @h;
module check;
for (@k=1;@k<=size(fac[1]);@k++)
{
  @p=fac[1][@k]^fac[2][@k];
  @h=@j[1]/@p;
  primMod=std(quotient(phi(@N),@h));
  check=imap(BAS,check);
  check=phi(check);
  if (size(NF(check,primMod,1))>0)
  {
    primary=std(@j+@p);
    // test if the modules were primary and in general position
    prim=primaryTest(primary,fac[1][@k]);
    if (prim==0)
    {
      rest[size(rest)+1]=invphi(primMod);
    }
    else
    {
      result[size(result)+1]=list(std(invphi(primMod)),std(invphi(prim)));
    }

  }
}

//////////////////////////////////////////////////////////////
// the bad cases
//////////////////////////////////////////////////////////////

for (@k=1; @k<=size(rest);@k++)
{
  result = result+zeroMod(rest[@k],invphi(check));
}

option(noredSB);
```

```
  setring BAS;
  list result=imap(@R, result);
  kill @R;
  return(result);
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,z,dp;
  module N=z*gen(1),(z-1)*gen(2),(z+1)*gen(3);
  list l=zeroMod(N);
  l;
}


////////////////////////////////////////////////////////////////////////////
// GTZmod (N[, check])
// computes a minimal primary decomposition of N
// using a generalized version of the algorithm of
// Gianni, Trager and Zacharias, suggested by Alexander Dreyer
// [Diploma Thesis, University of Kaiserslautern, Germany, 2001]
////////////////////////////////////////////////////////////////////////////

proc GTZmod (module @N, list #)
"USAGE:   GTZmod (N[, check]); module N[, module check]
RETURN:  list l
         the minimal primary decomposition of the module N,
         computed by a gernalized version of the algorithm
         of Gianny, Trager and Zacharias
NOTE:    if the parameter check is given, only components
         not containing check are computed
EXAMPLE: example GTZmod; shows an example
"
{
  if (size(@N)==0)
  {
    return(list(@N,ideal(0)));
  }

  ////////////////////////////////////////////////////////////
  // the module check is needed to compute a minimal decomposition
  // components containing check are ignored
  ////////////////////////////////////////////////////////////
  if (size(#)>0)
  {
    module check=#[1];
    if (size(NF(check,std(@N),1))==0)
    {
      return(list());
    }
  }
  else
  {
    module check= freemodule(nrows(@N));
  }

  module @M=freemodule(nrows(@N));
  def BAS = basering;
  int @j;
  int nVar=nvars(BAS);
  int @k;
  string @U;                      // the independent variables
  string @A;                      // the dependent variables
```

```
@N=std(@N);
ideal ann=std(quotient(@N,@M));  // the annihilator of @M/@N


////////////////////////////////////////////////////////////////
// the trivial and the zero-dimensional case
////////////////////////////////////////////////////////////////
int Ndim=dim(@N);
if ((Ndim==0)||(Ndim==-1))
{
  return(zeroMod(@N, check));
}

////////////////////////////////////////////////////////////////
// the not-that-trivial case of ann==0
////////////////////////////////////////////////////////////////
if(size(ann)==0)
{
  execute("ring Rloc=("+charstr(basering)+","+varstr(basering)+"),
                                    dummy,("+ordstr(basering)+");");
  module @N=imap(BAS, @N);
  poly @q=prepareSat(@N);

  setring BAS;
  poly @q=imap(Rloc, @q);
  list satu=sat(@N,@q);
  if(satu[2]==0)
  {
    return(list(list(@N,ideal(0))));
  }
  else
  {
    check=intersect(check,satu[1]);
    return(list(list(satu[1],ideal(0)))+GTZmod(@N+(@q^satu[2])*@M,check));
  }
}

////////////////////////////////////////////////////////////////
// the preparation of the quotient ring
////////////////////////////////////////////////////////////////
intvec indVec=indepSet(ann);
int szA;
for (@k=1;@k<=size(indVec);@k++)
{
  if (indVec[@k]==1)
  {
    @U=@U+varstr(@k)+",";
  }
  else
  {
    @A=@A+varstr(@k)+",";
    szA++;
  }
}
@U[size(@U)]=")";             // we compute the extractor (w.r.t. @U)
execute("ring RAU="+charstr(basering)+",("+@A+@U+",(C,dp("+string(szA)+"),dp);");
module @N=std(imap(BAS,@N));  // this is also a standard basis in (R[U])[A]
@A[size(@A)]=")";
execute("ring Rloc=("+charstr(basering)+","+@U+",("+@A+",(C,dp);");
module @N=imap(RAU,@N);
kill RAU;
```

```
  ////////////////////////////////////////////////////////////
  // the zero-dimensional decomposition
  ////////////////////////////////////////////////////////////
  list qprim=zeroMod(@N,imap(BAS,check));

  ////////////////////////////////////////////////////////////
  //  preparation for saturation
690 ////////////////////////////////////////////////////////////
  poly @q=prepareSat(@N);
  if (size(qprim)==0)
  {
    setring BAS;
    poly @q=imap(Rloc,@q);
    kill Rloc;
    @q=@q^sat(@N,@q)[2];
    if (deg(@q)>0)
    {
700   return(GTZmod(@N+@q*@M,check));
    }
    else
    {
      return(list());
    }
  }

  list @p;
  for (@k=1;@k<=size(qprim);@k++)
710 {
    @p[@k]=list(prepareSat(qprim[@k][1]),prepareSat(qprim[@k][2]));
  }

  ////////////////////////////////////////////////////////////
  // compute the recontractions
  // back in the original ring
  ////////////////////////////////////////////////////////////
  setring BAS;
  list @p=imap(Rloc,@p);
720 list qprim=imap(Rloc,qprim);
  poly @q=imap(Rloc,@q);
  kill Rloc;
  for(@k=1;@k<=size(qprim);@k++)
  {
    qprim[@k]=list(sat(qprim[@k][1],@p[@k][1])[1],
                   sat(qprim[@k][2],@p[@k][2])[1]);
    check=intersect(check,qprim[@k][1]);
  }
  @q=@q^sat(@N,@q)[2];
730 if (deg(@q)>0)
  {
    qprim=qprim+GTZmod(@N+@q*@M,check);
  }

  return(qprim);
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,(x,y,z),dp;
740 module N=x*gen(1)+ y*gen(2),
           x*gen(1)-x2*gen(2);
  list l=GTZmod(N);
  l;
}
```

```
proc prepareSat(module @N, list #)

{
  int @k;
  poly @p=leadcoef(@N[1]);
  for (@k=2;@k<=size(@N);@k++)
  {
    @p=lcm_chr(@p,leadcoef(@N[@k]));
    //    @p=@p*leadcoef(@N[@k]);
  }
  return(@p);
}

proc lcm_chr(poly @i, poly @j)

{
  def LBAS = basering;
  if (npars(basering)==0)
  {
    string strg="";
  }
  else
  {
     if (nvars(basering)==0)
     {
       string strg=parstr(basering);
     }
     else
     {
       string strg=parstr(basering)+",";
     }
  }
  execute("ring PRing="+string(char(basering))+",("+strg+varstr(basering)+"),dp");
  ideal @a=ideal(imap(LBAS,@i),imap(LBAS,@j));
  poly @p=lcm(@a);
  setring LBAS;
  poly @p=imap(PRing,@p);
  kill PRing;
  return(@p);
}

////////////////////////////////////////////////////////////////////////
// The optimized procedures and procdures needed for this optimization
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
// testit (N, l)
// a small procedure, which checks whether
// N=intersect(l[1][1],...,l[size(l)][1])
// and whether annil(l[i][1]) is primary
// Just for testing the procedures.
////////////////////////////////////////////////////////////////////////////

proc testit (module N, list #)
"USAGE:   testit (N, l); module N, list l
EXAMPLE: example testit; shows an example
"
{
  if (size(#)==0)
  {
```

```
    return()
  }
  int i;
  list l=#;
  module nn=freemodule(nrows(N));
  module M=freemodule(nrows(N));

  for(i=1;i<=size(l);i++)
  {
    nn=intersect(nn,l[i][1]);
    if ((size(decomp(quotient(l[i][1],M)))>2)&&(size(l[i][2])>0))
    {
     "nicht primary obwohl erkannt!";
     l[i];std(quotient(l[i][1],M));std(radical(quotient(l[i][1],M)));
      pause();
     }
  }
  int j,k;
  j=size(NF(nn,std(N),1));
  k=size(NF(N,std(nn),1));
  if ((j!=0)||(k!=0))
  {
    "testit fehler!!!";
    pause();
  }
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,(x,y,z),dp;
  module N=x*gen(1)+ y*gen(2),
           x*gen(1)-x2*gen(2);
  list l=GTZmod(N);
  testit(N,l);
}

////////////////////////////////////////////////////////////////////////////
// annil (N)
// computes the annihilator of M/N in the basering
////////////////////////////////////////////////////////////////////////////

proc annil (module N)
"USAGE:   annil(N);  modul N
RETURN:  ideal ann=std(quotient(N,freemodule(nrows(N))));
         the annihilator of M/N in the basering
NOTE:    ann is a std basis in the basering
EXAMPLE: example annil; shows an example
"
{
 intvec optionsVec=option(get);
 option (returnSB);
 ideal ann=quotient(N,freemodule(nrows(N)));
 attrib (ann, "isSB",1);
 option (set,optionsVec);
 return(ann);
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,(x,y,z),dp;
  module N=x*gen(1), y*gen(2);
  ideal ann=annil(N);
  ann;
}
```

```
870 ////////////////////////////////////////////////////////////////////////////
    //  splitting(N[,check[, ann]])
    //  INPUT:  a zero-dimensional module N, module check, ideal ann=annil(N)
    //          splitting computes an list of modules
    //          using the factorization of the elements of annil(N)
    //          s.th. N is equal to the intersections of these modules
    //          A prim test is used to check if the modules are primary
    //  OUTPUT: (l, check)
    ////////////////////////////////////////////////////////////////////////////

880 proc splitting (module @N, list #)
    "USAGE:   splitting(N[,check[, ann]]);  modul N, module check, ideal ann
    RETURN:  (l, check) list l, module check
             the elements of l consists of a triple with
             [1] of type module [2] and [3] of type ideal
             s.th. the intersection of the modules is equal to the
             zero-dimensional module N, furthermore l[j][3]=annil(l[j][1])
             if l[j][2]!=0 then the module l[j][1] is primary
                with associated prime l[j][2],
                and check=intersect(check, l[j][1]) is computed
890 NOTE:    if the parameter check is given, only components not containing
             check are computed; if ann is given, ann is used instead of annil(N)
    EXAMPLE: example splitting; shows an example
    "
    {
      ideal ann; module check, @M; int checked;
      (ann, check, @M,checked)=getData(@N,#);
      if(checked)
      {
        return(list());
900   }
      if(size(#)>=3)
      {
        ideal splitPrime=#[3];
      }
      else
      {
        ideal splitPrime=ann;
      }

910   list fact, result, splitTemp;
      int @i,@k,@j,szFact;
      for (@i=1;@i<=size(ann);@i++)
      {
        fact=factor(ann[@i]);
        szFact=size(fact[2]);
        // if the element is the power of an irreducible element
        if(szFact==1)
        {
          if(vdim(ann)==deg(ann[@i]))
920       {
            splitPrime=interred(splitPrime+ideal(fact[1][1]));
            result=result+list(list(@N,splitPrime,ann,splitPrime));
          }
          else
          {
            splitPrime=interred(splitPrime+ideal(fact[1][1]));
            if (homog(splitPrime))
            {
               result=result+list(list(@N,maxideal(1),ann,splitPrime));
930         }
```

```
    }
  }
  else
  {
    if(gcdTest(fact[1]))       // Case: (f1,...,fk)=(1)
    {
      (splitTemp, check)=sp1(@N,fact,check,ann,splitPrime);
      result=result+splitTemp;
    }
    else
    {
                              // if the element is not irreducible
      (splitTemp, check)=sp2(@N,fact[1][1],check,ann,splitPrime);
      result=result+splitTemp;
    }
  }
}
@i=1;@k=size(result);

////////////////////////////////////////////////////////////////
// delete multiple Modules
////////////////////////////////////////////////////////////////
while (@i<=@k)
{
  @j=1;
  while(@j<=@i-1)
  {
    if (stdModulesEqual(result[@j][1],result[@k][1]))
    {
      result=delete(result,@i);
      @k--;@i--;break;
    }
    @j++;
  }
  @i++;
}
list rest;
@i=1;@k=size(result);

////////////////////////////////////////////////////////////////
// if not primary then split the obtained modules once again
////////////////////////////////////////////////////////////////
while (@i<=@k)
{
  if (size(result[@i][2])==0)
  {
    rest=rest+list(list(result[@i][1],result[@i][3],result[@i][4]));
    result=delete(result,@i);
    @k--;@i--;
  }
  else
  {
    check=intersect(check,result[@i][1]);
  }
  @i++;
}
for(@i=1;@i<=size(rest);@i++)
{
  (splitTemp,check)=splitting(rest[@i][1],check,rest[@i][2],rest[@i][3]);
  result=result+splitTemp;
}
```

```
  if (size(result)==0)
  {
    result=list(list(@N,ideal(0),ann,ann));
  }
  return(result, check);
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,z,lp;
  module N=z*gen(1), (z+1)*gen(2);
  N=std(N);
  list l; module check;
  (l, check)=splitting(N);
  l;
  check;
}

  ////////////////////////////////////////////////////////////
  // sp1: splits a module as follows
  // (N+f*g*M)=intersect((N+f*M),(N+g*M)) if (f,g)=(1)
  ////////////////////////////////////////////////////////////

static proc sp1(module @N,list fact,list #)
{
  ideal ann; module check, @M; int @i;
  (ann, check, @M, @i)=getData(@N, #);
  if(size(#)>=3)
  {
    ideal splitPrime=#[3];
  }
  else
  {
    ideal splitPrime=ann;
  }
  list pr;
  module splitMod;
  ideal splitAnn, prim, tempPrime;
  for(@i=1;@i<=size(fact[2]);@i++)
  {
    splitMod=std(@N+(fact[1][@i]^fact[2][@i])*@M);
    if(size(NF(check,splitMod,1))>0)
    {
      splitAnn=std(ann,(fact[1][@i]^fact[2][@i]));
      tempPrime=interred(splitPrime+ideal(fact[1][@i]));
      prim=primTest(splitAnn,fact[1][@i],tempPrime);
      pr=pr+list(list(splitMod,prim,splitAnn,tempPrime));
      if (size(prim)>0)
      {
        check=intersect(check,splitMod);
      }
    }
  }
  return (pr, check);
}

  ////////////////////////////////////////////////////////////
  // sp2: splits a module as follows
  // N=intersect((N:f),(N+f*M))
  ////////////////////////////////////////////////////////////

static proc sp2(module @N,  poly p, list #)
{
```

```
  ideal ann; module check, @M; int @i;
  (ann, check, @M, @i)=getData(@N, #);
  if(size(#)>=3)
  {
    ideal splitPrime=#[3];
  }
  else
  {
    ideal splitPrime=ann;
  }
  list fact=sat(@N, p);
  list splitList;
  ideal splitAnn, prim, tempPrime;
  if (fact[2]>0)
  {
   module n1=std(@N+(p^fact[2]*@M));
   module n2=fact[1];
   if (size(NF(check,n1,1))>0)
   {
    splitAnn=std(ann+ideal(p^fact[2]));
    tempPrime=interred(splitPrime+ideal(p));
    prim=primTest(tempPrime);
    splitList=list(list(n1, prim, splitAnn,tempPrime));
    if(size(prim)>0)
    {
      check=intersect(check, n1);
    }
   }
   if(size(NF(check,n2,1))>0)
   {
    splitAnn=annil(n2);
    prim=primTest(splitAnn);
    splitList=splitList+list(list(n2,prim,splitAnn,splitAnn));
    if(size(prim)>0)
    {
    check=intersect(check, n2);
    }
   }
   return(splitList, check);
  }
  else
  {
   return (list(list(@N,ideal(0),ideal(0))), check);
  }
}

////////////////////////////////////////////////////////////////////////////
//  primTest(i[, p])
//  tests whether i is prime or homogeneous
//  is both cases radical(i) is returned
////////////////////////////////////////////////////////////////////////////

proc primTest(ideal id, list #)
"USAGE:   primTest(i[, p[, j]]); a zero-dimensional ideal i,
          irreducible poly p in i
RETURN:  if i vdim(i)=deg(f) then i is returned
         else if i or j is homogeneous radical(i)
              else (0)
EXAMPLE: example primTest; shows an example
"
{
  ideal tempPrime;
```

```
  int testTempPrime;
  if (size(#)>0)
  {
    poly @p=#[1];
    if(size(#)>1)
    {
      tempPrime=#[2];
      testTempPrime=1;
    }
  }
  else
  {
    poly @p=0;

  }
  ideal prim=ideal(0);
  if((size(#)>0)&&(vdim(id)==deg(@p)))
  {
    prim=id;
  }
  else
  {
    if ((homog(id))||((testTempPrime)&&(homog(tempPrime))))
    {
      prim=maxideal(1);
    }
  }
  return (prim);
}
example
{ "EXAMPLE:"; echo=2;
  ring r=0,(x,y,z),lp;
  ideal i=x+1,y-1,z;
  i=std(i);
  ideal primId=primTest(i,z);
  primId;

  i=x,z2,yz,y2;
  i=std(i);
  primId=primTest(i);
  primId;
}

////////////////////////////////////////////////////////////////////////////
// preComp(N, check[, ann])
// preComp is an enhanced version of splitting,
// but before computing splitting the first element of std(annil(N))
// is factorized and the obtained modules are tested for primarity
////////////////////////////////////////////////////////////////////////////

proc preComp (module @N, list #)
"USAGE:   preComp(N,check[, ann]);  modul N, module check, ideal ann
RETURN:  (l, check) list l, module check
         the elements of l consists of a triple with
         [1] of type module [2] and [3] of type ideal
         s.th. the intersection of the modules is equal to the
         zero-dimensional module N, furthermore l[j][3]=annil(l[j][1])
         if l[j][2]!=0 then the module l[j][1] is primary
            with associated prime l[j][2],
            and check=intersect(check, l[j][1]) is computed
NOTE:    only components not containing check are computed;
         if ann is given, ann is used instead of annil(N)
```

```
EXAMPLE: example preComp; shows an example
"
{
  def BAS=basering;
  changeord("@R","C,lp");
  module @N=std(imap(BAS,@N));
  ideal ann; module check, @M; int @k;
  (ann, check, @M, @k)=getData(@N,imap(BAS,#),1);
  list act,primary;
  ideal primid,helpid;
  module primmod;

  ////////////////////////////////////////////////////////////////
  // the first element of the standardbase is factorized
  ////////////////////////////////////////////////////////////////
  if(deg(ann[1])>0)
  {
    act=factor(ann[1]);
  }
  else
  {
    setring BAS;
    module check=imap(@R,check);
    kill @R;
    return(list(), check);
  }

  ////////////////////////////////////////////////////////////////
  // with the factors new modules are created
  // (hopefully the primary decomposition)
  ////////////////////////////////////////////////////////////////
  if(size(act[1])>1)               // Case: act[1] not irreducible
  {
     for(@k=1;@k<=size(act[1]);@k++)
     {
       primmod=std(@N+(act[1][@k]^act[2][@k])*@M);
       if (size(NF(check,primmod,1))>0)
       {
         primid=std(ann,act[1][@k]^act[2][@k]);
         if((act[2][@k]==1)&&(vdim(primid)==deg(act[1][@k])))
         {
           primary = primary+list(list(primmod,primid,primid));
         }
         else
         {
           helpid=primid;
           primid=primaryTest(primid,act[1][@k]);
           primary = primary+list(list(primmod,primid,helpid));
         }
       }
       if (size(primid)>0)
       {
         check=intersect(check, primmod);
       }
     }
   }
   else                            // Case: act[1]  irreducible
   {
     primid=ann;
     primmod=@N;

     if (size(NF(check,primmod,1))>0)
```

```
    {
      if((act[2][1]==1)&&(vdim(primid)==deg(act[1][1])))
      {
        primary = primary+list(list(primmod,primid,primid));
      }
      else
      {
        primid = primaryTest(primid,act[1][1]);
        primary = primary+list(list(primmod,primid,ann));
      }
      if (size(primid)>0)
      {
        check=intersect(check,primmod);
      }
    }
  }

  if (size(primary)==0)
  {
    setring BAS;
    module check=imap(@R,check);
    kill @R;
    return(list(), check);
  }

  ////////////////////////////////////////////////////////////
  // the modules which are not primary are splitted
  ////////////////////////////////////////////////////////////
  list splitTemp;
  int sz=size(primary);
  @k=1;
  while (@k<=sz)
  {
    if (size(primary[@k][2])==0)
    {
      (splitTemp, check)=splitting(primary[@k][1],check,primary[@k][3]);
      primary = delete(primary, @k)+splitTemp;
      @k--;sz--;
    }
    @k++;
  }

  setring BAS;
  list primary=imap(@R,primary);
  module check=imap(@R,check);
  kill @R;
  return(primary,check);
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,z,lp;
  module N=z*gen(1), (z+1)*gen(2);
  N=std(N);
  list l; module check;
  (l, check)=preComp(N,freemodule(2));
  l;
  check;
}

////////////////////////////////////////////////////////////////////////////
// indSet(i)
// based on independendSet from primdec.lib
```

```
////////////////////////////////////////////////////////////////////////////

proc indSet (ideal @j)
"USAGE:   indSet(i); i ideal
RETURN:  list with two entrees
         both are lists of new varstrings with the dependend variables
         the independent set, the ordstring with the corresp. block ordering,
         and the integer where the independent set starts in the varstring
NOTE:    the first entry gives the strings for all maximal independend sets
         the second gives the strings for the independend sets,
         which cannot be enhanced
EXAMPLE: example indSet; shows an example
"
{
   int n,k,di;
   int jdim=dim(@j);
   list maxind, rest,hilf;
   string var1,var2;
   list v=indepSet(@j,1);

   for(n=1;n<=size(v);n++)
   {
      di=0;
      var1="";
      var2="";
      for(k=1;k<=size(v[n]);k++)
      {
         if(v[n][k]!=0)
         {
            di++;
            var2=var2+string(var(k))+",";
         }
         else
         {
            var1=var1+string(var(k))+",";
         }
      }
      if(di>0)
      {
         var1=var1[1..size(var1)-1];
         var2=var2[1..size(var2)-1];
         hilf[1]=var1;
         hilf[2]=var2;
         hilf[3]="(C,dp("+string(nvars(basering)-di)+"),dp)";
         //"lp("+string(nvars(basering)-di)+"),dp("+string(di)+")";
         hilf[4]=di;
         if(di==jdim)
         {
          maxind=maxind+list(hilf);
         }
         else
         {
          rest=rest+list(hilf);
         }
      }
       else
       {

        if(jdim==0)
         {
          maxind=maxind+list(varstr(basering),"dummy",ordstr(basering),0);
         }
```

```
            else
            {
             rest=rest+list(varstr(basering),"dummy",ordstr(basering),0);
            }
            resu[n]=varstr(basering),ordstr(basering),0;
1370      }
   }
   return(list(maxind,rest));
}
example
{ "EXAMPLE:"; echo = 2;
   ring s1=(0,x,y),(a,b,c,d,e,f,g),lp;
   ideal i=ea-fbg,fa+be,ec-fdg,fc+de;
   i=std(i);
   list  l=indSet(i);
1380  l;
}

////////////////////////////////////////////////////////////////////////////
// GTZopt(N[,check[, ann]])
// a faster version of GTZMod
////////////////////////////////////////////////////////////////////////////

proc GTZopt (module @N, list #);
"USAGE:   GTZopt (N[, check]); module N[, module check]
1390 RETURN:  list l
         the minimal primary decomposition of the module N,
         computed by a generalized and optimized version of
         the algorithm of Gianny, Trager and Zacharias
NOTE:    if the parameter check is given, only components
         not containing check are computed
EXAMPLE: example GTZmod; shows an example
"
{
  @N=std(@N);
1400  if (size(@N)==0)
  {
    return(list(@N,ideal(0)));
  }

  ////////////////////////////////////////////////////////////////
  // the module check is needed to compute a minimal decomposition
  // components containing check are ignored
  ////////////////////////////////////////////////////////////////
  ideal ann; module check, @M; int checked;
1410  (ann, check, @M, checked)=getData(@N, #);
  if (checked)
  {
    return(list());
  }

  ////////////////////////////////////////////////////////////////
  // if ann is zero-dimensional and homogeneous
  // then it is primary with associated prime maxideal(1)
  ////////////////////////////////////////////////////////////////
1420  if((homog(ann)==1)&&(dim(ann)==0))
  {
    return(list(list(@N,maxideal(1))));
  }

  ////////////////////////////////////////////////////////////////
  // the not-that-trivial case of ann==0
```

```
////////////////////////////////////////////////////////////
def BAS = basering;
if(size(ann)==0)      //check, whether ann=0
{
 execute("ring Rloc=("+charstr(basering)+","+varstr(basering)+"),
                                  dummy,("+ordstr(basering)+");");
 module @N=clrSBmod(imap(BAS, @N));
 module @M=freemodule(nrows(@N));
 poly @q=prepareSat(@N);

 setring BAS;
 poly @q=imap(Rloc, @q);
 list satu=sat(@N,@q);

 if(satu[2]==0)
 {
   return(list(list(@N,ideal(0))));
 }
 else
 {
  check=intersect(check,satu[1]);
  return(list(list(satu[1],ideal(0)))+GTZopt(@N+(@q^satu[2])*@M,check));
 }
}

int @k1, @k2, @k3, @k4;   // the indices for nested for/while loops
int nVar=nvars(BAS);
int Ndim=dim(@N);

////////////////////////////////////////////////////////////////
// Simplification of the modules using
// N=N/(a*x_i+b)*M+(a*x_i+b)*M, for (a*x_i+b) in ann
////////////////////////////////////////////////////////////////
if (size(#)==0)
{
  ideal fried;
  @k2=size(ann);
  for(@k1=1;@k1<=@k2;@k1++)
  {
    if(deg(lead(ann[@k1]))==1)
    {
      fried[size(fried)+1]=ann[@k1];
    }
  }
  if(size(fried)==nVar)
  {
     return(list(list(@N, ann)));
  }
  if(size(fried)>0)
  {
     string newva;
     string newma;
     for(@k1=1;@k1<=nVar;@k1++)
     {
        checked=0;
        for(@k2=1;@k2<=size(fried);@k2++)
        {
           if(leadmonom(fried[@k2])==var(@k1))
           {
              checked=1;
              break;
           }
```

```
            }
            if(checked==0)
            {
              newva=newva+string(var(@k1))+",";
              newma=newma+string(var(@k1))+",";
            }
            else
            {
              newma=newma+string(var(@k1)-((1/leadcoef(fried[@k2]))*fried[@k2]))+",";
            }
          }
          newva[size(newva)]=")";
          newma[size(newma)]=";";
          execute("ring @deirf=("+charstr(BAS)+"),("+newva+",(C,lp);");
          execute("map @kappa=BAS,"+newma);
          ideal @j = @kappa(ann);
          module @N = @kappa(@N);
          @N=simplify(@N,2);
          @j=simplify(@j,2);
          list pr=GTZopt(@N,freemodule(nrows(@N)),@j);
          setring BAS;
          list pr=imap(@deirf,pr);

          for(@k1=1;@k1<=size(pr);@k1++)
          {
            pr[@k1][1]=std(pr[@k1][1]+fried*@M);
            pr[@k1][2]=std(pr[@k1][2]+fried);
          }
          return(pr);
        }
      }

      ////////////////////////////////////////////////////////////////
      // the trivial case
      ////////////////////////////////////////////////////////////////
      if(Ndim==-1)
      {
        return(list(list(@N,ideal(1))));
      }

      ////////////////////////////////////////////////////////////////
      // the case of one variable
      ////////////////////////////////////////////////////////////////
      if(nVar==1)
      {
        return(dec1var(@N));
      }

      ////////////////////////////////////////////////////////////////
      // the zerodimensional case
      ////////////////////////////////////////////////////////////////
      if (Ndim==0)
      {
        return(zeroOpt(@N, check, ann));
      }

      ////////////////////////////////////////////////////////////////
      // the preparation of the quotient ring
      ////////////////////////////////////////////////////////////////
      list result;
      list indep =indSet(ann);
      poly @q;
```

```
list @p,primary;
ideal @h;
int szIndep;
for (@k1=1;@k1<=2;@k1++)
{
  szIndep=size(indep[@k1]);
  for (@k2=1;@k2<=szIndep;@k2++)
  {
    execute("ring RAU=("+charstr(basering)+"),("+indep[@k1][@k2][1]+
                ","+indep[@k1][@k2][2]+"),"+indep[@k1][@k2][3]+";");
    module @N=std(imap(BAS,@N)); // the standard basis in (R[U])[A]
    execute("ring Rloc=("+charstr(basering)+","+indep[@k1][@k2][2]+"),
                                    ("+indep[@k1][@k2][1]+"),(C,dp);");
    module @N=imap(RAU,@N); //std in lokalisierung
    @N=clrSBmod(@N);
    kill RAU;

    ////////////////////////////////////////////////////////////
    // the zero-dimensional decomposition
    ////////////////////////////////////////////////////////////
    list qprim, preList;
    module check=imap(BAS, check);
    (preList,check)=preComp(@N,check);
    for (@k3=1; @k3<=size(preList); @k3++)
    {
      if(size(preList[@k3][2])>0)
      {
        qprim=qprim+list(list(preList[@k3][1],preList[@k3][2]));
      }
      else
      {
        checked=size(qprim);
        qprim=qprim+zeroOpt(preList[@k3][1], check, preList[@k3][3]);
        for(@k4=checked+1;@k4<=size(qprim);@k4++)
        {
          check=intersect(check, qprim[@k4][1]);
        }
      }
    }  // end of for(@k3...)
    kill preList;

    ////////////////////////////////////////////////////////////
    // Preparation of the saturation of @N
    ////////////////////////////////////////////////////////////
    //poly pp=prepareSat(@N);
    ideal @h2;
    for (@k3=1;@k3<=size(@N);@k3++)
    {
      @h2[@k3]=leadcoef(@N[@k3]);
    }

    if (size(qprim)==0)     // there aren't any new components
    {
      setring BAS;
      check=imap(Rloc,check);
      @h=imap(Rloc,@h2);
      @q=minSatMod(imap(BAS,@N),@h)[2];
      // @q=imap(Rloc,pp)^sat(imap(BAS,@N),imap(Rloc,pp))[2];
      kill Rloc;
      if (deg(@q)>0)
      {
        @N=std(@N+@q*@M);
```

```
      ann=std(ideal(ann+@q));
      kill qprim;
    }
  }
  else                    // there are new components
  {
    ////////////////////////////////////////////////////////////////
    // Preparation of the saturation of qprim
    ////////////////////////////////////////////////////////////////
    list @p2;
    for (@k3=1;@k3<=size(qprim);@k3++)
    {
      @p2[@k3]=list(prepareSat(qprim[@k3][1]),prepareSat(qprim[@k3][2]));
    }


    ////////////////////////////////////////////////////////////////
    // compute the recontractions
    // back in the original ring
    ////////////////////////////////////////////////////////////////
    setring BAS;
    @p=imap(Rloc,@p2);
    primary=imap(Rloc,qprim);
    @h=imap(Rloc,@h2);
    kill Rloc;
    for(@k3=1;@k3<=size(primary);@k3++)
    {
      primary[@k3]=list(sat(primary[@k3][1],@p[@k3][1])[1],
                        sat(primary[@k3][2],@p[@k3][2])[1]);
      check=intersect(check,primary[@k3][1]);
    }
    @q=minSatMod(imap(BAS,@N),@h)[2];
    result=result+primary;
    if (deg(@q)>0)
    {
      @N=std(@N+@q*@M);
      ann=std(ideal(ann+@q));
    }
  }                       // end of else
  if ((@k1==1)&&(@k2<szIndep)&&(Ndim>dim(ann)))
  {
    break;
  }
 }
}
return(result+GTZopt(@N,check,ann));
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,(x,y,z),dp;
  module N=x*gen(1)+ y*gen(2),
           x*gen(1)-x2*gen(2);
  list l=GTZopt(N);
  l;
}


////////////////////////////////////////////////////////////////////////////
// dec1var(N[,check[, ann]])
// primary decompostion for a ring with one variable
////////////////////////////////////////////////////////////////////////////
```

```
proc dec1var (module @N, list #);
"USAGE:   dec1var (N); zero-dimensional module N[, module check]
RETURN:  list l
         the minimal primary decomposition of a submodule N of R^s
         if nvars(R)=1
NOTE:    if the parameter check is given, only components
         not containing check are computed
EXAMPLE: example zeroMod; shows an example
"
{
  ideal ann; module @M, check; int checked;
  (ann, check, @M, checked)=getData(@N, #);

  list fac = factorize(ann[1],2);
  if(size(fac[2])==1)
  {
   return(list(list(@N,ann)));
  }
  // comp of the primary modules, the primary ideals and the primes
  poly @h;
  module primod;
  list result;
  int @k;
  for (@k=1;@k<=size(fac[1]);@k++)
  {
    @h=ann[1]/(fac[1][@k]^fac[2][@k]);
    result =result+list(list(std(quotient(@N,@h)), std(ann,fac[1][@k])));
  }
    return(result);
}


////////////////////////////////////////////////////////////////////////////
// zeroOpt(N[,check[, ann]])
// a faster version of zeroMod
////////////////////////////////////////////////////////////////////////////

proc zeroOpt (module @N, list #)
"USAGE:   zeroOpt (N[, check]); zero-dimensional module N[, module check]
RETURN:  list l
         the minimal primary decomposition of a zero-dimensional module N,
         computed by a generalized and optimized version of the algorithm
         of Gianny, Trager and Zacharias
NOTE:    if the parameter check is given, only components
         not containing check are computed
EXAMPLE: example zeroMod; shows an example
"
{
  @N=interred(@N);
  attrib(@N,"isSB",1);

  ////////////////////////////////////////////////////////////
  // the module check is needed to compute a minimal decomposition
  // components containing check are ignored
  ////////////////////////////////////////////////////////////
  ideal ann; module @M, check; int checked;
  (ann, check, @M, checked)=getData(@N, #);

  if (checked)
  {
    return(list());
  }
```

```
////////////////////////////////////////////////////////////
// the ordering is changed to lex
////////////////////////////////////////////////////////////
def BAS = basering;
changeord("@R","C,lp");
module @N=std(imap(BAS,@N));
module @M=imap(BAS,@M);
ideal ann=std(imap(BAS,ann));
module check=imap(BAS,check);

////////////////////////////////////////////////////////////
if(vdim(ann)==deg(ann[1]))           // if ann ist prime
{
  list fact=factor(ann[1]);
  int k;ideal id;list result;
  module hilf;
  for(k=1;k<=size(fact[1]);k++)
  {
    id=ann;
    hilf=std(@N+(fact[1][k]^fact[2][k])*@M);
    id[1]=fact[1][k];
    if(size(NF(check,hilf,1))>0)
    {
      result=result+list(list(hilf,interred(id)));
    }
  }
  setring BAS;
  list result=imap(@R, result);
  kill @R;
  return(result);

}


if (homog(ann))                      // if ann is homogeneous
{                                    // then radical(ann)=maxideal(1)
  if(size(NF(check,@N,1))>0)
  {
    setring BAS;
    kill @R;
    return (list(list(@N,maxideal(1))));
  }
  else
  {
    setring BAS;
    kill @R;
    return(list());
  }
}

////////////////////////////////////////////////////////////
// the random coordnate change and its inverse
// furthermore the module is simplified using
// N=N/(a*x_i+b)+(a*x_i+b)*M, for a*x_i+b in ann
////////////////////////////////////////////////////////////
int nVar=nvars(@R);
int @k, @k1;
list result, rest;
ideal primary, prim;
module primmod;
ideal fried;
intvec optionsVec=option(get);
```

```
option(redSB);
ideal prepMap = randomLast(100);
ideal prepInv=maxideal(1);
for(@k=1;@k<=size(ann);@k++)
{
  if(deg(lead(ann[@k]))==1)
  {
    fried[size(fried)+1]=ann[@k];
  }
}
if(size(fried)==nVar)
{
  return(list(list(@N, ann)));
}
if(size(fried)>0)
{
  for(@k=1;@k<@k1;@k1++)
  {
    for(@k1=1;@k1<=size(fried);@k1++)
    {
      if(leadmonom(fried[@k1])==var(@k))
      {
        prepMap[@k]=var(@k)+((1/leadcoef(fried[@k1]))*(var(@k)-fried[@k1]));
        prepMap[nVar]=subst(prepMap[nVar],var(@k),0);
        prepInv[@k]=fried[@k1];
      }
    }
  }
}
map phi=@R,prepMap;
prepInv[nVar]=2*var(nVar)-prepMap[nVar];
map invphi=@R,prepInv;
ideal @j=std(phi(ann));         // factorization of the 1st elt. in Ann(@N)
list fac = factorize(@j[1],2);

//////////////////////////////////////////////////////////////
// Case: 1st element irreducible
//////////////////////////////////////////////////////////////
if(size(fac[2])==1)
{
  prim=primaryTest(@j,fac[1][1]);
  prim=invphi(prim);
  setring BAS;
  @N=std(@N);
  ideal prim =imap(@R,prim);
  kill @R;
  if(prim!=0)
  {
    return(list(list(@N,prim)));
  }
  else
  {
    return(zeroOpt(@N,check));
  }
}

//////////////////////////////////////////////////////////////
// Computation of the - hopefully primary - modules
// their annihilators and associated primes
//////////////////////////////////////////////////////////////
poly @p, @h;
for (@k=1;@k<=size(fac[1]);@k++)
```

```
  {
    @p=fac[1][@k]^fac[2][@k];
    @h=@j[1]/@p;
    primmod=std(quotient(phi(@N),@h));
    check=phi(check);
    if (size(NF(check,primmod,1))>0)
    {
      primary=std(@j+@p);
      // test if the modules were primary and in general position
      prim=primTest(primary,fac[1][@k]);
      if(size(prim)==0)
      {
        prim=primaryTest(primary,fac[1][@k]);
      }
      if (prim==0)
      {
        rest[size(rest)+1]=invphi(primmod);
      }
      else
      {
        result[size(result)+1]=list(std(invphi(primmod)),std(invphi(prim)));
      }
    }
  }

  ////////////////////////////////////////////////////////////////
  // the bad cases
  ////////////////////////////////////////////////////////////////
  for (@k=1; @k<=size(rest);@k++)
  {
    result = result+zeroOpt(rest[@k],check);
  }
  option (set,optionsVec);
  if(size(result)==0)
  {
    setring BAS;
    kill @R;
    return(list());
  }
  setring BAS;
  list result=imap(@R, result);
  kill @R;

  return(result);
}
example
{ "EXAMPLE:"; echo = 2;
  ring r=0,z,dp;
  module N=z*gen(1),(z-1)*gen(2),(z+1)*gen(3);
  list l=zeroOpt(N);
  l;
}

////////////////////////////////////////////////////////////////////////////
// clrSBmod(N)
// Generalization of clearSB from primdec.lib
////////////////////////////////////////////////////////////////////////////

proc clrSBmod (module @N)
"USAGE:   clrSBmod(N); N module which is SB ordered by monomial ordering
RETURN:  module = minimal SB
EXAMPLE: example clrSBmod; shows an example
```

```
"
{
  int @k,@j;
  list Nsizes;
  for (@k=1;@k<=size(@N);@k++)
  {
    Nsizes[@k]=size(@N[@k]);
  }
  module leadVec;
  int szN=size(@N);
  @j=0;
  while(@j<szN-1)
  {
    @j++;
    if(deg(@N[@j])>0)
    {
      leadVec=lead(@N[@j]);
      attrib(leadVec,"isSB",1);
      for(@k=@j+1;@k<=szN;@k++)
      {
        if(size(NF(lead(@N[@k]),leadVec,1))==0)
        {
          if((leadexp(leadVec[1])!=leadexp(@N[@k]))||(Nsizes[@j]<=Nsizes[@k]))
          {
            @N[@k]=0;
          }
          else
          {
            @N[@j]=0;
             break;
          }
        }
      }
    }
  }
  return(simplify(@N,2));
}
example
{ "EXAMPLE:"; echo = 2;
   ring  r = (0,a,b),(x,y,z),dp;
   module N1=ax2+y,a2x+y,bx;
   module N2=clrSBmod(N1);
   N2;
}


////////////////////////////////////////////////////////////////////////////
// minSatMod(N, id)
// Generalization of minsat from primdec.lib
////////////////////////////////////////////////////////////////////////////

proc minSatMod(module Nnew, ideal @h)
"USAGE:   minSatMod(N, I); module N, ideal I
RETURN:  list with 2 elements:
         [1]=sat(N,product(I))[1],
         [2]=p, the polynomial of minimal degree s.th. [1]=quotient(N,p)
EXAMPLE: example minSatMod; shows an example
"
{
   int @i,@k;
   poly @f=1;
   module Nold;
```

```
   ideal fac;
   list quotM,@l;

   for(@i=1;@i<=ncols(@h);@i++)
   {
1990  if(deg(@h[@i])>0)
      {
         fac=fac+factorize(@h[@i],1);
      }
   }
   fac=simplify(fac,4);
   if(size(fac)==0)
   {
      @l=Nnew,1;
      return(@l);
2000  }
   fac=sort(fac)[1];
   for(@i=1;@i<=size(fac);@i++)
   {
      @f=@f*fac[@i];
   }
   quotM[1]=Nnew;
   quotM[2]=fac;
   quotM[3]=@f;
   @f=1;
2010  while(specialModulesEqual(Nold,quotM[1])==0)
   {
      if(@k>0)
      {
         @f=@f*quotM[3];
      }
      Nold=quotM[1];
      quotM=quotMinMod(quotM);
      @k++;
   }
2020  @l=quotM[1],@f;
   return(@l);
}
example
{ "EXAMPLE:"; echo = 2;
   ring  r = 0,(x,y,z),dp;
   module N=xy*gen(1);
   ideal h=yz,z2;
   list l=minSatMod(N,h);
   l;
2030 }

////////////////////////////////////////////////////////////////////////////
// quotMinMod(N, fac, f)
// Generalization of quotMin from primdec.lib
////////////////////////////////////////////////////////////////////////////

proc quotMinMod(list tsil)
{
   int @i,@j,action;
2040  module verg;
   list @l;
   poly @g;
   intvec optionsVec;

   module laedi=tsil[1];
   ideal fac=tsil[2];
```

```
   poly @f=tsil[3];
   optionsVec=option(get);
   option(returnSB);
   module star=quotient(laedi,@f);
   option(set,optionsVec);
   if(specialModulesEqual(star,laedi))
   {
      @l=star,fac,@f;
      return(@l);
   }

   action=1;
   while(action==1)
   {
      if(size(fac)==1)
      {
         action=0;
         break;
      }
      for(@i=1;@i<=size(fac);@i++)
      {
        @g=1;
        verg=laedi;

         for(@j=1;@j<=size(fac);@j++)
         {
            if(@i!=@j)
            {
               @g=@g*fac[@j];
            }
         }
         optionsVec=option(get);
         option(returnSB);
         verg=quotient(laedi,@g);
         option(set,optionsVec);
         if(specialModulesEqual(verg,star)==1)
         {
            @f=@g;
            fac[@i]=0;
            fac=simplify(fac,2);
            break;
         }

         if(@i==size(fac))
         {
            action=0;
         }
      }
   }
   @l=star,fac,@f;
   return(@l);
}

////////////////////////////////////////////////////////////////////////////
// specialModulesEqual(N1,N2)
// Generalization of specialIdealsEqual from primdec.lib
////////////////////////////////////////////////////////////////////////////

proc specialModulesEqual( module k1, module k2)
"USAGE:   specialModulesEqual(N1, N2) N1, N2 standard bases of modules,
         s.th. N1 is contained in N2 or vice versa
RETURN:  int i
```

```
            if (N1==N2) then i=1
            else i=0
EXAMPLE: example specialModulesEqual; shows an example
"
{
   int @j;

   if(size(k1)==size(k2))
   {
      for(@j=1;@j<=size(k1);@j++)
      {
         if(leadexp(k1[@j])!=leadexp(k2[@j]))
         {
            return(0);
         }
      }
      return(1);
   }
   return(0);
}
example
{ "EXAMPLE:"; echo = 2;
   ring  r = 0,(x,y,z),dp;
   module N1=x*freemodule(2);
   module N2=xy*freemodule(2);
   int i=specialModulesEqual(N1,N2);
   i;

   N2=N1;
   i=specialModulesEqual(N1,N2);
   i;
}

////////////////////////////////////////////////////////////////////////////
// sat2mod(N,i)
// Generalization of sat2 from primdec.lib
////////////////////////////////////////////////////////////////////////////

proc sat2mod (module id, ideal h1)
"USAGE:   sat2mod(id,j);  id ideal, j polynomial
RETURN:  saturation of id with respect to j (= union_(k=1...) of id:j^k)
NOTE:    result is a std basis in the basering
"
{
   int @k,@i;
   def @P= basering;
   if(ordstr(basering)[1,2]!="dp")
   {
      execute("ring @Phelp=("+charstr(@P)+"),("+varstr(@P)+"),(C,dp);");
      module inew=std(imap(@P,id));
      ideal  @h=imap(@P,h1);
   }
   else
   {
      ideal @h=h1;
      module inew=std(id);
   }
   ideal fac;

   for(@i=1;@i<=ncols(@h);@i++)
   {
     if(deg(@h[@i])>0)
```

```
      {
         fac=fac+factorize(@h[@i],1);
      }
   }
   fac=simplify(fac,4);
   poly @f=1;
   if(deg(fac[1])>0)
   {
      module iold;

      for(@i=1;@i<=size(fac);@i++)
      {
        @f=@f*fac[@i];
      }
      intvec optionsVec=option(get)
      option(returnSB);
      while(specialModulesEqual(iold,inew)==0 )
      {
         iold=inew;
         if(deg(iold[size(iold)])!=1)
         {
            inew=quotient(iold,@f);
         }
         else
         {
            inew=iold;
         }
         @k++;
      }
      option(set,optionsVec);
      @k--;
   }

   if(ordstr(@P)[1,2]!="dp")
   {
      setring @P;
      module inew=std(imap(@Phelp,inew));
      poly @f=imap(@Phelp,@f);
   }
   list L =inew,@f^@k;
   return (L);
}

////////////////////////////////////////////////////////////////////////////
// stdModulesEqual(N1,N2)
// Generalization of stdIdealsEqual from primdec.lib
////////////////////////////////////////////////////////////////////////////

proc stdModulesEqual(module k1, module k2)
"USAGE:   stdModulesEqual(N1, N2) N1, N2 standard bases of modules,
RETURN:  int i
         if (N1==N2) then i=1
         else i=0
EXAMPLE: example stdModulesEqual; shows an example
"
{
   int @j;

   if(size(k1)==size(k2))
   {
      for(@j=1;@j<=size(k1);@j++)
      {
```

```
            if(leadexp(k1[@j])!=leadexp(k2[@j]))
            {
               return(0);
            }
         }
         attrib(k2,"isSB",1);
         if(size(reduce(k1,k2,1))==0)
         {
            return(1);
         }
      }
      return(0);
}
example
{ "EXAMPLE:"; echo = 2;
   ring  r = 0,(x,y,z),dp;
   module N1=x*freemodule(2);
   module N2=xy*freemodule(2);
   int i=stdModulesEqual(N1,N2);
   i;

   N2=N1;
   i=stdModulesEqual(N1,N2);
   i;
}

////////////////////////////////////////////////////////////////////////////
// ModulesEqual(N1,N2)
// Generalization of IdealsEqual from primdec.lib
////////////////////////////////////////////////////////////////////////////

proc modulesEqual( module @k, module @j)
"USAGE:   modulesEqual(N1, N2) N1, N2 modules,
RETURN:  int i
         if (N1==N2) then i=1
         else i=0
EXAMPLE: example modulesEqual; shows an example
"
{
   return(stdModulesEqual(std(@k),std(@j)));
}
example
{ "EXAMPLE:"; echo = 2;
   ring  r = 0,(x,y,z),dp;
   module N1=x*freemodule(2);
   module N2=xy*freemodule(2);
   int i=stdModulesEqual(N1,N2);
   i;

   N2=N1;
   i=modulesEqual(N1,N2);
   i;
}

proc getData (module @N, list oldData, list #)
"USAGE:   getData(N, l[, noCheck]);  module N, list l[, int noCheck]
RETURN:  (ann, check, M, checked)
         ideal ann, module check, M, int checked

         if l[1] is contained in N [and noCheck is not given]
            then checked=1, ann=ideal(0), check=0, M=0;
         else checked=0, M=freemodule(nrows(N)); check=l[1]
```

```
            (resp. check=M if l is an empty list) and
            if size(l)>1 then  ann=l[2] else ann is the annihilator of M/N.

NOTE:    ann is a std basis in the basering
EXAMPLE: example getData; shows an example
"
{
  if (size(oldData)>0)
  {
    if ((size(#)==0)&&(size(NF(oldData[1],@N,1))==0))
    {
      return(ideal(0), 0 ,  0, 1);
    }
    module @M=freemodule(nrows(@N));
    if (size(oldData)>1)
    {
      ideal ann=oldData[2];
      attrib(ann,"isSB",1);
    }
    else
    {
      ideal ann=annil(@N);
    }
  }
  else
  {
   module @M=freemodule(nrows(@N));
   oldData[1]=@M;
   ideal ann=annil(@N);
  }
 return(ann, oldData[1], @M, 0);
}
example
{ "EXAMPLE:"; echo = 2;
   ring  r = 0,(x,y,z),lp;
   module N=x*gen(1),y*gen(2);
   ideal ann; module check, M; int checked; list l;
   (ann, check, M, checked)=getData(N,l);
   ann; check; M; checked;

   l=list(check,ann);
   (ann, check, M, checked)=getData(N,l);
   ann; check; M; checked;

   l=list(N);
   (ann, check, M, checked)=getData(N,l);
   ann; check; M; checked;
}
```

Bibliography

[AM] M.F. Athiyah, I.G. MacDonald: Introduction to Commutative Algebra, Addison-Wesley (1994)

[CLO1] Cox, Little, O'Shea: Ideals, Varieties and Algorithms, Undergraduate Texts in Mathematics, Springer-Verlag (1997)

[CLO2] Cox, Little, O'Shea: Using Algebraic Geometry, Graduate Texts in Mathematics 185, Springer-Verlag (1998)

[DGP] Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister: Algorithms and Comparisons, in: Greuel, G.-M.; Matzat, B. H.; Hiß, G. (Eds.), Algorithmic Algebra and Number Theory, 187-220, Springer-Verlag, Heidelberg (1998).

[EH] David Eisenbud and Joe Harris: The geometry of schemes, Graduate Texts in Mathematics 197, Springer-Verlag, New York, 1999.

[Eis] David Eisenbud: Commutative Algebra with a View Toward Algebraic Geometry, Graduate Texts in Mathematics 150, Springer-Verlag (1995).

[GP] Gert-Martin Greuel, Gerhard Pfister: A SINGULAR Introduction to Commutative Algebra, to appear in: Mathematics - Monograph (English), Springer-Verlag

[Grä] Hans-Gert Gräbe: Minimal Primary Decomposition and Factorized Gröbner Bases, J. AAECC 8, 265 - 278 (1997)

[GTZ] Gianni, Trager, Zacharias: Gröbner Bases and Primary Decomposition of Polynomial Ideals, J. Symbolic Computation 6, 149-167 (1988)

[Rut] Elisabeth W. Rutman: Gröbner bases and Primary Decomposition of Modules, J. Symbolic Computation 14(5), 483-504 (1992)

[SY] Shimoyama, T.; Yokoyama, K. . Localization and Primary Decomposition of Polynomial ideals. J. Symb. Comp. 22, 247-277 (1996)

Statement

Hereby I assure that I wrote this thesis by myself and that I have exclusivesly used the indicated literature and resources.

Kaiserslautern, August 25, 2003

Alexander Dreyer