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

Signature standard bases over principal ideal rings

Adrian Popescu

1. Gutachter: Prof. Dr. Gerhard Pfister
2. Gutachter: Prof. Dr. Martin Kreuzer

Vollzug der Promotion: 28 September 2016

D 386

Abstract

By using Gröbner bases of ideals of polynomial algebras over a field, many implemented algorithms manage to give exciting examples and counter examples in Commutative Algebra and Algebraic Geometry. Part A of this thesis will focus on extending the concept of Gröbner bases and Standard bases for polynomial algebras over the ring of integers and its factors $\mathbb{Z}_m[x]$. Moreover we implemented two algorithms for this case in SINGULAR which use different approaches in detecting useless computations, the classical Buchberger algorithm and a F5 signature based algorithm.

Part B includes two algorithms that compute the graded Hilbert depth of a graded module over a polynomial algebra R over a field, as well as the depth and the multigraded Stanley depth of a factor of monomial ideals of R. The two algorithms provide faster computations and examples that lead B. Ichim and A. Zarojanu to a counter example of a question of J. Herzog. A. Duval, B. Goeckner, C. Klivans and J. Martin have recently discovered a counter example for the Stanley Conjecture. We prove in this thesis that the Stanley Conjecture holds in some special cases.

Part C explores the General Neron Desingularization in the frame of Noetherian local domains of dimension 1. We have constructed and implemented in SINGULAR and algorithm that computes a strong Artin Approximation for Cohen$-$Macaulay local rings of dimension 1.

Für Polynomringe über Körpern sind Gröbnerbasen ein wichtiges Hilfsmittel für Berechnungen in der kommutativen Algebra und der algebraischen Geometrie. Teil A dieser Dissertation behandelt die Verallgemeinerung des Konzepts der Gröbnerbasis und Standardbasis für Polynomringen über dem Ring der ganzen Zahlen und seinen Faktorringen. Hierbei haben wir theoretische Aspekte untersucht und eine Vielzahl neuer Strategien entwickelt. Darauf aufbauend haben wir zwei verschiedene Algorithmen im Computeralgebrasystem SINGULAR implementiert, die sich durch ihre Kriterien zum Finden unnötiger Berechnungen grundlegend unterscheiden, den Buchberger Algorithmus und den signaturbasierte F5 Algorithmus.

Der Teil B beinhaltet zwei Algorithmen zum Berechnen der graduierten Hilberttiefe von graduierten Moduln über einer polynomialen Algebra R über einem Körper, sowie der Tiefe und der multigraduierten Stanleytiefe eines Faktors eines monomialen Ideals von R. Die beiden Algorithmen bieten schnellere Berechnungsverfahren, die zu Beispielen geführt haben, die B. Ichim und A. Zarojanu nutzen konnten ein Gegenbeispiel für eine Frage von J. Herzog zu finden. Vor Kurzem erst haben A. Duval, B. Goeckner, C. Klivans und J. Martin ein Gegenbeispiel zur Stanley Vermutung gefunden. Wir zeigen in dieser Arbeit, dass die Stanley Vermutung in speziellen Fällen richtig ist.

Der Teil C ist der allgemeinen Neron Desingularisierung im Fall von Noetherschen, lokalen Bereichen der Dimension 1 gewidmet. Hier wurde ein Algorithmus zur Berechnung der Artinsche Approximation für lokale Cohen−Macaulay Ringe der Dimension 1 entwickelt.

ACKNOWLEDGMENTS

I would like to thank Gerhard Pfister for being much more than a professor; Christian Eder for always having time for my standard bases questions and for working on them together; Hans Schönemann for been able to answer all my SINGULAR -related questions; Wolfram Decker and Christoph Lossen for finding financial support without which this thesis would not have been possible; Anne Frühbis-Krüger for the help in developing new strategies used in solving integer related bugs, Jakob Kröker for sending many bugs regarding the standard bases over rings and to Petra Bäsell for helping me in any other problem.

I would also like to thank Natalie for spending lots of time correcting my English mistakes and special thanks to my parents, Adriana and Dorin, who supported and motivated me.

Danke :)

Preface

This thesis is structured into three parts that approach different branches in Computer Algebra and Commutative Algebra and Programming in SINGULAR 's sources. The preface will provide the reader a brief history of the main concepts used in each of these parts and an overall description of this papers structure.

PART A – *Standard Bases over Rings*

The notion of Gröbner bases was first introduced in 1965 by Buchberger in his PhD thesis [Bu]. Independently, Grauert and Hironaka introduced the notion of standard bases, a generalization of the Gröbner bases. This concept proves to be useful in a wide range of fields like Commutative Algebra, Singularity Theory and Algebraic Geometry. They are used in solving system of polynomial equations, ideal intersection problems and even resolving Sudoku puzzles. The standard bases serve as a stepping stone of a relatively new mathematical field: Computer Algebra. Although the main idea of Buchberger's algorithm still remains the same, many new strategies and criteria were developed to speed up the algorithm.

O. Wienand has modified the Gröbner basis algorithm to work well over base rings that are not necessarily fields and may have zero divisors. The author has extended Wienand's implementation of the algorithm in SINGULAR also for local and mixed orderings in his Master thesis. Here we review the algorithm and add some strategies to speed up the computations, especially when working over the integers.

In 2002 Faugère introduced the F5 algorithm. A lot of the computations in Buchberger's algorithm give no new information for the standard basis. The F5 algorithm provides strategies to detect and eliminate some of these useless computations by using signatures. A very interesting survey on the F5 algorithms and variants can be found in [EF]. C. Eder studied and implemented a variant of F5 in SINGULAR . This paper provides a generalization of this algorithm over rings that are not necessarily fields.

PART B – *Depth, Stanley Depth and Hilbert Depth*

In 1982, Richard Stanley has conjectured an upper bound for the depth of a multigraded module (see [St]). Nowadays this is known as the Stanley Conjecture and it surprisingly connects two notions: depth – a homological invariant and the Stanley depth – a combinatorial concept defined by Stanley Decompositions. At a first glance no connection between these can be established, but after a closer look similarities can be established: if the Stanley depth is 0, then the depth is 0. Rinaldo published an algorithm that computes the Stanley depth of a quotient of monomial ideals and implemented it in [CoCoA]. After the introduction of the canonical form we were able to speed up this algorithm (see [AP2]).

The conjecture has been open for more than 30 years and in the last 10 years many papers prove particular cases of it. However, in 2016 a counterexample was published in [DuGo]. A survey concerning the Stanley depth has been published by J. Herzog; before this counterexample was found and contains an overview of most of the known results up to 2011 (see [H]).

Stanley decompositions prove to be very useful in describing finitely generated graded algebras, e.g. rings of invariants under some group action (see Sturmfels and White [SW]), and in applications of the normal form theory for systems of differential equations with nilpotent linear part (see Murdock and Sanders [MS]). Another astonishing result concerning the Stanley conjecture is the following result from [HSY] stating that if the conjecture holds for the Stanley-Reisner of a certain simplicial complex then it is partitionable.

Herzog, Vlădoiu and Zheng proved in [HVZ] that the module structure is already given by the Hilbert series and therefore the Stanley depth may be computed from this. Based on this idea, Bruns, Krattenthaler and Uliczka introduced in [BKU] the Hilbert depth – a similar notion to the Stanley depth. We introduce a first algorithm that computes the Hilbert depth and implemented it in SINGULAR .

PART C – *Constructive General Neron Desingularization*

The Implicit Mapping Theorem plays an important role in Singularity Theory and Analytic Geometry. A Noetherian local ring (R, m) has the Artin Approximation Property if *each finite system of polynomials equations over R has a solution in R iff it has a solution in the completion $\hat{R}$.* This can be formulated as follows: for every $c \in \mathbb{N}$, each solution of this system $\hat{y}$ from $\hat{R}$ can be approximated with a solution y in R modulo m^c. M. Artin proved in 1969 that the algebraic power series over a field has the Artin Approximation Property (see [Artin1]). In 1986, D. Popescu proved that each excellent Henselian local ring has this property (see [DP2]) by using the existence of a so-called General Neron Desingularization. D. Popescu and the author gave an algorithmic proof of the existence of a General Neron Desingularization in the frame of one dimensional local domains (see [APDP2]) and implemented this algorithm in SINGULAR . A generalization of this algorithm was later published by G. Pfister and D. Popescu (see [PfPo2]).

Many applications of the Artin Approximation Property in Singularity theory and Commutative Algebra have been presented in a series of conferences hosted in Luminy, Marseilles, France organized by H. Hauser and G. Rond at the beginning of 2015.

THE STRUCTURE

In ***Chapter A:1*** we introduce the reader to a series of basic definitions and notations from the standard bases theory together with the classic Buchberger Algorithm to compute a Gröbner basis. We continue by expanding the above mentioned concepts to the case when the ground ring is a principal ideal ring that is not a field (as an extension of O. Wienands PhD Thesis [W]). This chapter also provides a description of several strategies that will decrease the run time and memory of the algorithm.

Chapter A:2 consists of Faugère's F5 algorithm, optimized and implemented in SINGULAR by C. Eder over fields, and a generalization of the strategy for principal ideal rings. We also describe the difficulties that arise on this approach together with a couple of ideas that will overcome these obstacles. At the very end we compare the Buchberger Algorithm with this version of F5 over the ring of integers on some random examples.

Chapter B:1 introduces the basic definitions and notations used throughout Part B and explores in details concepts like the depth of a module, Hilbert Series and Stanley and Hilbert decomposition.

Chapter B:2 consists of a procedure that computes the Hilbert depth of a module. This is the first in a line of algorithms that recently appeared in order to compute hdepth. This chapter was published in the Journal of Symbolic Computation [AP2].

In ***Chapter B:3*** we take an in depth look of the Stanley Conjecture. After a short introduction of the conjecture, we present some new results in which the conjecture holds. This chapter was partially published in several papers (see [AP1], [AP3] and [APDP1]).

Chapter B:4 introduces the new notion of canonical form that is later used to optimize Rinaldo's algorithm for the Stanley depth computation. This chapter was published in Bulletin Mathématique de la Société des Sciences Mathématiques de Roumanie ([AP3]).

Part C starts with a small introduction on Artin Approximation and the General Neron Desingularization. It also contains an algorithmic proof in the case of Noetherian local domains of dimension 1. The algorithm for this case was implemented in SINGULAR . This part has also been published (see [APDP2]).

At the very end the ***Appendix*** contains several important examples used throughout the thesis together with a part exploring the source code of SINGULAR on page 143.

Part A

Standard bases over rings

Chapter A:1

Buchberger's Algorithm

In this chapter we introduce the basic notions for standard bases and the additional strategies needed when the polynomial algebra is defined over a ring that is not necessarily a field.

A:1.1 Basic definitions and notations

Definition A:1.1.1.

Let $\mathrm{Mon}(x_1, \ldots, x_n)$ be the set of monomials in $n \in \mathbb{N}$ variables $x_1, \ldots, x_n$. We denote by x the set of all variables and for $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ we set $x^\alpha := x_1^{\alpha_1} \cdot \ldots \cdot x_n^{\alpha_n}$.

A ***monomial ordering*** $<$ is a total ordering on $\mathrm{Mon}(x)$ satisfying the following property

$$x^\alpha < x^\beta \implies x^\gamma \cdot x^\alpha < x^\gamma \cdot x^\beta \quad \text{for all } \alpha, \beta, \gamma \in \mathbb{N}^n.$$

A monomial ordering is called ***global*** if $1 = x^0 \leq x^\alpha$ for all $\alpha \in \mathbb{N}^n$.

A monomial ordering is called ***local*** if $1 = x^0 \geq x^\alpha$ for all $\alpha \in \mathbb{N}^n$.

A monomial ordering is called ***mixed*** if there is $\alpha, \beta \in \mathbb{N}^n$ such that $x^\alpha < 1 < x^\beta$.

Definition A:1.1.2.

Let $<$ be a monomial ordering on $\mathrm{Mon}(x)$, R be a ring and $f \in R[x] := R[x_1, \ldots, x_n]$ a polynomial. We can write f in a unique way

$$f = \sum_{i=1}^{L} c_i \cdot x^{\alpha^{(i)}},$$

$0 \neq c_i \in R$ and $\alpha^{(i)} \in \mathbb{N}^n$ for $1 \leq i \leq L$ such that $x^{\alpha^{(1)}} < \ldots < x^{\alpha^{(L)}}$. We define

- the ***length*** of f denoted by $\ell(f) = L$,
- the ***leading term*** denoted by $\mathrm{LT}(f) = c_L \cdot x^{\alpha^{(L)}}$,
- the ***leading coefficient*** denoted by $\mathrm{LC}(f) = c_L$,
- the ***leading monomial*** denoted by $\mathrm{LM}(f) = x^{\alpha^{(L)}}$,
- the ***tail*** of f denoted by $\mathrm{tail}(f) = f - \mathrm{LT}(f)$,

- the ***degree*** of f denoted by $\deg(f) = \max\limits_{1 \leq i \leq L} \left\{\alpha_1^{(i)} + \ldots + \alpha_n^{(i)}\right\}$ and
- the ***ecart*** of f denoted by $\mathrm{ecart}(f) = \deg(f) - \deg\big(\mathrm{LT}(f)\big)$.

In the following we define some common monomial orderings and provide a small example to see the differences between them.

Definition A:1.1.3.

- ***negative degree reverse lexicographical ordering*** (called `ds` in Singular). This *local ordering* is defined on $R[x]$ as follows:

$$x^\alpha <_{\texttt{ds}} x^\beta \iff \begin{array}{l} \sum\limits_{i=1}^{n} \alpha_i =: \deg(x^\alpha) > \deg(x^\beta) \text{ or} \\ \deg(x^\alpha) = \deg(x^\beta) \text{ and } \exists 1 \leq i \leq n \text{ s.t. } \alpha_n = \beta_n, \ldots, \alpha_{i+1} = \beta_{i+1}, \alpha_i > \beta_i. \end{array} \tag{A:1.1}$$

- ***negative lexicographical ordering*** (called `ls` in Singular). This *local ordering* is defined on $R[x]$ as follows:

$$x^\alpha <_{\texttt{ls}} x^\beta \iff \exists\, 1 \leq i \leq n \text{ s.t. } \alpha_1 = \beta_1, \ldots, \alpha_{i-1} = \beta_{i-1}, \alpha_i > \beta_i. \tag{A:1.2}$$

- ***degree reverse lexicographical ordering*** (called `dp` in Singular). This *global ordering* is defined on $R[x]$ as follows:

$$x^\alpha <_{\texttt{dp}} x^\beta \iff \begin{array}{l} \sum\limits_{i=1}^{n} \alpha_i =: \deg(x^\alpha) < \deg(x^\beta) \text{ or} \\ \deg(x^\alpha) = \deg(x^\beta) \text{ and } \exists 1 \leq i \leq n \text{ s.t. } \alpha_n = \beta_n, \ldots, \alpha_{i+1} = \beta_{i+1}, \alpha_i < \beta_i. \end{array} \tag{A:1.3}$$

- ***lexicographical ordering*** (called `lp` in Singular). This *global ordering* is defined on $R[x]$ as follows:

$$x^\alpha <_{\texttt{lp}} x^\beta \iff \exists\, 1 \leq i \leq n \text{ s.t. } \alpha_1 = \beta_1, \ldots, \alpha_{i-1} = \beta_{i-1}, \alpha_i < \beta_i. \tag{A:1.4}$$

- ***weighted orderings***. All the above degree orderings can be enhanced by introducing weights. For a vector $w = \big(w_1, \ldots, w_n\big) \in \mathbb{Z}^n$ we define the weighted degree of x^α by

$$\text{w-deg}(x^\alpha) = w_1\alpha_1 + \ldots + w_n\alpha_n.$$

We define the ***weighted reverse lexicographic ordering***, denoted by $\texttt{wp}(w_1, \dots, w_n)$, by replacing deg by w-deg in the Equation A:1.3.

Similarly we define the ***weighted negative reverse lexicographic ordering***, denoted by $\texttt{ws}(w_1, \dots, w_n)$, by changing Equation A:1.1.

EXAMPLE A:1.1.4. Consider $f = xyz + x^2 + z^2 + y \in R[x, y, z]$. Below we write f in its unique form for each of the monomial orderings from Definition A:1.1.3.

$$\begin{aligned}
<_{\texttt{dp}}&: \; xyz + x^2 + z^2 + y \\
<_{\texttt{lp}}&: \; x^2 + xyz + y + z^2 \\
<_{\texttt{ls}}&: \; z^2 + y + xyz + x^2 \\
<_{\texttt{ds}}&: \; y + x^2 + z^2 + xyz
\end{aligned}$$

We notice that in this example, each of the 4 monomials can be seen as the leading term depending on the monomial ordering.

REMARK A:1.1.5. Unfortunately, this is not true in general. Consider $f = x^2 + xy + y^2 \in R[x, y]$. There is no monomial ordering such that xy is the leading monomial of f.

Proof.
Let $<$ be a monomial ordering. There are two cases:

- $x < y$. In this case multiplying by x, respectively y gives us that $\mathrm{LT}(f) = y^2$.
- $x > y$. Similarly, $\mathrm{LT}(f) = x^2$. □

DEFINITION A:1.1.6.
Let $I \subseteq R[x]$ be an ideal.
The ***leading ideal*** of I is $\mathrm{L}_<(I) = \langle \{\mathrm{LT}(f) \mid f \in I\} \rangle$.
The ***leading monomial ideal*** of I is $\mathrm{LM}_<(I) = \langle \{\mathrm{LM}(f) \mid f \in I\} \rangle$.
The leading ideal for a set of polynomials $S := \{f_1, \dots, f_m\}$ is defined to be the leading ideal of the ideal generated by S.
From now on, if the monomial ordering is clear from the context, we will only write $\mathrm{L}(I)$ (respectively $\mathrm{LM}(I)$) instead of $\mathrm{L}_<(I)$ (respectively $\mathrm{LM}_<(I)$).

REMARK A:1.1.7. When R is a field it is easy to see that $\mathrm{L}_<(I) = \mathrm{LM}_<(I)$ for any ideal $I \subset R[x]$. This is not true over $R[x]$ if R is a ring that is not a field. For instance over $\mathbb{Z}[x]$, $\mathrm{L}_<\big(\langle 2x \rangle\big) = \langle 2x \rangle \subsetneq \langle x \rangle = \mathrm{LM}_<\big(\langle 2x \rangle\big)$. In general $\mathrm{L}_<(I) \subseteq \mathrm{LM}_<(I)$, but the converse does not always hold for rings that are no fields, as shown in the example above. As one would expect, the leading coefficients play an important role.

Throughout Part A when speaking about rings, we will refer in fact to rings that are no fields.

DEFINITION A:1.1.8 (Standard Bases)**.**

A ***standard basis*** of an ideal $I \subset R[x]$ with respect to a fixed monomial ordering $<$ is a finite set $S \subset I$ such that $\mathrm{L}(I) = \mathrm{L}(S)$.

S is a ***strong standard basis*** if additionally it satisfies the following property:

$$\forall f \in I, \exists\, g \in S \text{ such that } \mathrm{LT}(g) \mid \mathrm{LT}(f).$$

REMARK A:1.1.9. Note that in the case of fields all standard bases S are strong since we can always set the leading coefficient of a polynomial $g \in S$ to be 1 by multiplying g with its inverse $\mathrm{LC}(g)^{-1}$. Over rings this is not true, as shown in the following example.

EXAMPLE A:1.1.10. Consider the ideal $I = \langle 4x, 13x \rangle = \langle x \rangle$ over $\mathbb{Z}[x]$. It is easy to see that $S = \{4x, 13x\}$ is a standard basis for I. Take $f = 2x \in I$ and since $4x$ nor $13x$ divides f, S is not a strong standard basis of I. A strong standard basis for I is $S' = \{x\}$.

DEFINITION A:1.1.11.

A ring R is a ***principal ideal ring*** if each ideal I of R can be generated using a single element.

DEFINITION A:1.1.12.

Let $I \subset R$ be an ideal. The ***annihilator*** (or ***annulator***) of I, denoted by $\mathrm{Ann}(I)$, is

$$\mathrm{Ann}(I) = \{c \in R \mid c \cdot m = 0, \forall m \in I\}.$$

For an element $c \in R$, we will shortly denote by $\mathrm{Ann}(c) = \mathrm{Ann}\left(\langle c \rangle\right)$.

We extend the previous definitions and notations over modules. Let $I = \langle f_1, \ldots, f_s \rangle$ an ideal in $R[x]$ and $R[x]^s$ be the free $R[x]$ module $\bigoplus_{i=1}^{s} R[x] \cdot \varepsilon_i$, where ε_i is the canonical basis.

DEFINITION A:1.1.13.

A ***monomial*** in the module case is an element of the form $m \cdot \varepsilon_i = \big(0, \ldots, 0, m, 0, \ldots, 0\big) \in R[x]^s$ where m is a monomial in $R[x]$ and it is located on the $i-$th component while all the others components are 0.

DEFINITION A:1.1.14.

A ***module monomial ordering*** (or simply called ***module ordering***) is a total ordering $\prec$ on the set of monomials of $R[x]^s$ compatible with the monomial ordering $<$ on $R[x]$, i.e.

- ○ $m_\alpha < m_\beta \implies m_\alpha \cdot \varepsilon_i \prec m_\beta \cdot \varepsilon_i$,
- ○ $m_\alpha \cdot \varepsilon_i \prec m_\beta \cdot \varepsilon_j \implies m_\alpha m_\gamma \cdot \varepsilon_i \prec m_\beta m_\gamma \cdot \varepsilon_j$,

for all monomials m_α, m_β, m_γ in $R[x]$ and for each $i, j = 1, \dots, s$.

Definition A:1.1.2 simply extends to the module case.

There are 4 canonical ways to naturally extend the ring ordering $<$ to a module ordering by regarding the components as shown in the example below.

Example A:1.1.15 (Module orderings)**.** We define the following module orderings:

- $\prec_{(c,<)}$
$$m_\alpha \cdot \varepsilon_i \prec_{(c,<)} m_\beta \cdot \varepsilon_j \quad\Longleftrightarrow\quad \begin{array}{l} i > j \text{ or} \\ i = j \text{ and } m_\alpha < m_\beta. \end{array}$$
- $\prec_{(C,<)}$
$$m_\alpha \cdot \varepsilon_i \prec_{(C,<)} m_\beta \cdot \varepsilon_j \quad\Longleftrightarrow\quad \begin{array}{l} i < j \text{ or} \\ i = j \text{ and } m_\alpha < m_\beta. \end{array}$$
- $\prec_{(<,c)}$
$$m_\alpha \cdot \varepsilon_i \prec_{(<,c)} m_\beta \cdot \varepsilon_j \quad\Longleftrightarrow\quad \begin{array}{l} m_\alpha < m_\beta \text{ or} \\ m_\alpha = m_\beta \text{ and } i > j. \end{array}$$
- $\prec_{(<,C)}$
$$m_\alpha \cdot \varepsilon_i \prec_{(<,C)} m_\beta \cdot \varepsilon_j \quad\Longleftrightarrow\quad \begin{array}{l} m_\alpha < m_\beta \text{ or} \\ m_\alpha = m_\beta \text{ and } i < j. \end{array}$$

Definition A:1.1.16 (syzygy)**.**

Let $I = \langle f_1, \dots, f_s \rangle$ be an ideal in $R[x]$. Consider the map

$$\begin{tikzcd} \pi : \bigoplus_{i=1}^{s} R[x] \cdot \varepsilon_i \arrow[r] & R[x] \\ \varepsilon_i \arrow[r, mapsto] & f_i \end{tikzcd}$$

A ***syzygy*** is an element $s \in \ker \pi$.

A:1.2 Buchberger's algorithm blueprint

This section presents a simple Buchberger algorithm blueprint for computing standard bases in polynomial rings over principal ideal rings. For this we introduce some more basic notions.

In the following, let R be a ring. Usually we will take R to be $\mathbb{Z}$ or $\mathbb{Z}_r := {}^{\mathbb{Z}}/_{r\mathbb{Z}}$ where $r \in \mathbb{Z}$ is not a prime number. Let $f, g \in R[x]$ be two polynomials such that

$$f = c_f \cdot m_f + \dots$$

$$g = c_g \cdot m_g + \dots$$

with $\mathrm{LT}(f) = c_f \cdot m_f$ and $\mathrm{LT}(g) = c_g \cdot m_g$.

Definition A:1.2.1.

We define several important polynomials used in the strong standard bases theory.

- The $s-$***polynomial*** (or ***s−pair***) of f and g, denoted by

$$\text{s-poly}(f,g) = \frac{\mathrm{lcm}(c_f,c_g)\,\mathrm{lcm}\,(m_f,m_g)}{c_f m_f}\cdot f - \frac{\mathrm{lcm}(c_f,c_g)\,\mathrm{lcm}\,(m_f,m_g)}{c_g m_g}\cdot g.$$

  Note that the leading terms will cancel out.

- Since R is a principal ideal ring, we have that $\langle c_f, c_g\rangle = \langle c\rangle$. Let $c = d_f \cdot c_f + d_g \cdot c_g$. The ***strong polynomial*** (or the ***gcd polynomial***, ***gcd−pair***, ***strong pair***) of f and g denoted by

$$\text{gcd-poly}(f,g) = \frac{d_f\,\mathrm{lcm}(m_f,m_g)}{m_f}\cdot f + \frac{d_g\,\mathrm{lcm}(m_f,m_g)}{m_g}\cdot g.$$

  Notice that $\mathrm{LT}\left(\text{gcd-poly}(f,g)\right) = c\cdot\mathrm{lcm}(m_f,m_g)$.

- In case of rings with zero divisors, the ***extended s−polynomial*** of f denoted by

$$\text{ext-poly}(f) = \mathrm{Ann}(c_f)\cdot f = \mathrm{Ann}(c_f)\cdot\mathrm{tail}(f).$$

  Note that if $\mathrm{LC}(f)$ is not a zero divisor then $\text{ext-poly}(f) = 0$.

Remark A:1.2.2. Note that in the definition of gcd−pairs d_f and d_g are not uniquely determined. For instance, $\langle 3,5\rangle = \langle 1\rangle$ and we can consider several d_f and d_g:

$$\begin{array}{rcrrcrl} 2 & \cdot & 3+ & (-1) & \cdot & 5 & =1\\ (-3) & \cdot & 3+ & 2 & \cdot & 5 & =1\\ 7 & \cdot & 3+ & (-4) & \cdot & 5 & =1 \end{array}$$

However, it is easy to prove that we can randomly choose one of them, since they will give dependent polynomials with respect to the s−polynomial. For this, see the next lemma.

Lemma A:1.2.3.

Let two polynomials $f, g \in R[x]$,

$$f = c_f \cdot m_f + \dots,$$

$$g = c_g \cdot m_g + \dots,$$

such that $\langle c\rangle = \langle c_f, c_g\rangle$. Let $d_f \neq d'_f, d_g \neq d'_g$ such that

$$d_f \cdot c_f + d_g \cdot c_g = c = d'_f \cdot c_F + d'_g \cdot c_g.$$

Denote by P and Q the gcd−polynomial constructed with d_f,d_g, respectively d'_f and d'_g. Then $P = Q + \alpha\cdot\text{s-poly}(f,g)$ for an $\alpha \in R$.

Proof.

By definition,

$$\text{s-poly}(f,g) = \frac{L}{c_f} \cdot \frac{\operatorname{lcm}(m_f, m_g)}{m_f} \cdot f - \frac{L}{c_g} \cdot \frac{\operatorname{lcm}(m_f, m_g)}{m_g} \cdot g,$$

where $L = \operatorname{lcm}(c_f, c_g)$,

$$P = d_f \cdot \frac{\operatorname{lcm}(m_f, m_g)}{m_f} \cdot f + d_g \cdot \frac{\operatorname{lcm}(m_f, m_g)}{m_g} \cdot g,$$

and

$$Q = d'_f \cdot \frac{\operatorname{lcm}(m_f, m_g)}{m_f} \cdot f + d'_g \cdot \frac{\operatorname{lcm}(m_f, m_g)}{m_g} \cdot g.$$

One can easily see that

$$P - Q = (d_f - d'_f) \cdot \frac{\operatorname{lcm}(m_f, m_g)}{m_f} \cdot f + (d_g - d'_g) \cdot \frac{\operatorname{lcm}(m_f, m_g)}{m_g} \cdot g.$$

Using the choice of d_f, d'_f, d_g and d'_g, we know that

$$(d_f - d'_f) \cdot c_f = -(d_g - d'_g) \cdot c_g,$$

and therefore

$$\frac{d_f - d'_f}{\frac{L}{c_f}} = -\frac{d_g - d'_g}{\frac{L}{c_g}} =: \alpha.$$

This is the α we needed and hence we are done. □

Example A:1.2.4. Consider two polynomials $f = 5x^3 + 5xy + 3y$ and $g = 3y^2 + 2x$ in $\mathbb{Z}_{10}[x, y]$ and the dp (degree reverse lexicographical) ordering. Then

- $\text{s-poly}(f,g) = 3y^2 \cdot f - 5x^3 \cdot g = 5xy^3 + 9y^3$,
- $\text{gcd-poly}(f,g) = -y^2 \cdot f + 2x^3 \cdot g = x^3y^2 + 4x^4 + 5xy^3 + 7y^3$,
- $\text{ext-poly}(f) = 2 \cdot f = 6y$ and
- $\text{ext-poly}(g) = 0 \cdot g = 0$.

Remark A:1.2.5. Note that if $\langle c_f, c_g \rangle = \langle c_f \rangle$, then we can take $d_f = 1$ and $d_g = 0$. Therefore $\text{gcd-poly}\,(f,g) = m \cdot f$, for the monomial $m = \dfrac{\operatorname{lcm}(m_f, m_g)}{m_f}$. In other words if one of the leading coefficients divides the other, then the gcd-poly is equal to a multiple of the initial polynomial and, as we will later see, it brings no new information. We call such a gcd$-$polynomial ***redundant***.

DEFINITION A:1.2.6.

Let $S = \{f \in R[x] \mid \mathrm{LM}(f) = 1 \text{ and } \mathrm{LC}(f) \text{ is a unit in } R\}$. Denote by

$$R[x]_< := S^{-1}R[x].$$

Note that if $<$ is a global ordering, $R[x]_< = R[x]$ and when $<$ is a local or mixed ordering, $R[x]_< = R[x]_{\langle x \rangle}$.

DEFINITION A:1.2.7.

Let $\mathcal{G}$ be the set of all finite subsets $G \subset R[x]_<$. The map

$$\begin{tikzcd}
\mathrm{NF}: & R[x]_< \times \mathcal{G} \arrow[r] & R[x] \\
& (f, G) \arrow[r, mapsto] & \mathrm{NF}(f \mid G)
\end{tikzcd}$$

is called a ***weak normal form*** if for each $f \in R[x]_<$ and each finite set of polynomials $G \subset R[x]_<$, the following properties hold:

- $\mathrm{NF}(0 \mid G) = 0$,
- if $\mathrm{NF}(f \mid G) \neq 0$, then $\mathrm{LT}\big(\mathrm{NF}(f \mid G)\big) \notin \mathrm{L}(G)$ and
- for $f \neq 0$, there exists a unit $u \in R[x]_<$ such that either $u \cdot f = \mathrm{NF}(f \mid G)$ or the remainder $r = u \cdot f - \mathrm{NF}(f \mid G)$ has a ***standard representation*** w.r.t. G, that is

  $$r = \sum_{g \in G} c_g \cdot g, \text{ where } c_g \in R,$$

  $\mathrm{LM}(r) \geq \mathrm{LM}(c_g) \cdot \mathrm{LM}(g)$ for all g s.t. $c_g \cdot g \neq 0$.

A weak normal form is simply called a ***normal form*** if one can always choose $u = 1$ in the standard representation.

An algorithm to compute the weak normal form over fields can be found in the SINGULAR book [GP, Algorithm 1.7.6]. The algorithm will look as below, if the monomial ordering is *global*.

```
Algorithm reduce(poly f, ideal G)
Input: $f \in R[x]$ and a finite subset $G \subset R[x]$ on which we consider a global ordering
Output: $h \in R[x]$, a weak normal form of $f$ w.r.t. $G$
1: $h = f$;
2: while exists $g \in G$ s.t. $\mathrm{LT}(g) \mid \mathrm{LT}(h)$ do
3:     $h = \text{s-poly}(h, g)$;
4: return $h$
```

Note that on line 3 of the algorithm, with each s-poly computation, the leading monomial of h will decrease. This algorithm only works for global orderings. If the orderings are local or mixed, we need to also consider the ecart of h. This is explained in Section A:1.5.

The normal form plays an important role in the standard basis computation because of Buchberger's Criterion described below.

LEMMA A:1.2.8 (Buchberger Criterion,[GP, Theorem 1.7.3]).

Let I and ideal in $R[x]$ and a set $G = \{g_1, \dots, g_s\} \subset I$. Let $\mathrm{NF}(\cdot|G)$ be a weak normal form on $R[x]$ with respect to G. Then the following are equivalent:

1. *G is a standard basis of I*
2. $\mathrm{NF}(f \mid G) = 0$ *for all* $f \in I$
3. *each $f \in I$ has a standard representation with respect to G*
4. *G generates I and* $\mathrm{NF}\left(\text{s-poly}(g_i, g_j) \mid G\right) = 0$ *for* $1 \leq i, j \leq s$.

REMARK A:1.2.9. Note that the elements of a standard basis generate not just the leading ideal but the ideal itself. Therefore, one can view the standard bases as a set of generators of an ideal with special properties.

We are now ready to show the algorithm's blueprint.

ALGORITHM A:1.2.10 (Buchberger's blueprint).

Let R be a principal ideal ring, $<$ a monomial ordering on $R[x]$ and $I = \langle f_1, \dots, f_r \rangle \subset R[x]$ be an ideal. The following algorithm returns a strong standard basis $S = \{g_1, \dots, g_t\}$ of I.

```
1: $S = \{f_1, \dots, f_r\}$;
2: $L = \{\text{s-poly}(f_i, f_j) \mid i < j\} \cup \{\text{gcd-poly}(f_i, f_j) \mid i < j\} \cup \{\text{ext-poly}(f_i)\}$;
3: while $L \neq \emptyset$ do
4:     choose $h \in L$ and reduce it with $S$;
5:     if $h \neq 0$ then
6:         $S = S \cup \{h\}$;
7:         $L = L \cup \{\text{s-poly}(g, h) \mid g \in S\} \cup \{\text{gcd-poly}(g, h) \mid g \in S\} \cup \{\text{ext-poly}(h)\}$;
8: return $S$
```

The set L will be called the pair list. Through the reduction on line 4 we understand computing a normal form as in the algorithm described earlier.

REMARK A:1.2.11. When working with a polynomial ring over a field, we just add of the s$-$polynomials in line 7 of the algorithm. If R is a ring, the following natural questions arise:

Why do we need extended s−polynomials?

Let $I = \langle f \rangle \subset \mathbb{Z}_{10}[x, y]$ be an ideal, where $f = 5x^3 + 5xy + 3y$ and consider the dp ordering. Algorithm A:1.2.10 without adding the extended s-polynomial yields $S = \{f\}$ as a strong standard basis with $\mathrm{L}(S) = \langle 5x^3 \rangle$. But since I is an ideal and $f \in I$, then $2f \in I$ implying $\mathrm{LT}(2f) = 6y \in \mathrm{L}(I) = \mathrm{L}(S)$. Therefore, S should also contain $2f$, and hence the extended s−polynomial of f, $\text{ext-poly}(f)$. This is the reason why we need to add $\text{ext-poly}(f)$ in the case of zero divisors.

Why do we need strong / gcd polynomials?

Example A:1.1.10 gives us a quick answer. If we compute the strong standard basis with Algorithm A:1.2.10 without considering the strong pairs, then we would get $S = \langle 4x, 13x \rangle$ as a strong standard basis, which is false. If we add $\text{gcd-poly}\,(4x, 13x) = x$ to S, then we get a correct result. This is the reason why we need to add $\text{gcd-poly}(f, g)$ in the case of rings to obtain a strong standard basis. We need a strong standard basis since most of the useful properties of a standard basis over fields come from the fact that it is also a strong standard basis.

In the last couple of years we improved existing strategies, developed and implemented new tricks in ring standard bases computations in the SINGULAR source code. In the next sections, we present some of them.

A:1.3 ALL vs. JUST

In this section let $R = \mathbb{Z}$. Looking back at Remark A:1.2.5 we see that we do not always need to construct gcd-polys. D. Lichtblau proved in [Li, Theorem 2] that in Algorithm A:1.2.10 is in fact enough to take for a pair (g, h) either the s-poly or the gcd-poly. Note line 7 of the pseudo-code where we add $\text{s-poly}(g, h)$ and $\text{gcd-poly}(g, h)$ to the pair list L. Instead of this, if $\text{gcd-poly}(g, h)$ is not redundant then we only add $\text{gcd-poly}(g, h)$ and if $\text{gcd-poly}(g, h)$ is redundant then we only add $\text{s-poly}(g, h)$.

We name the usual strategy in which we consider all pairs by ALL and the one in which we consider just one pair simply by JUST.

At a first glance fewer pairs could be interpreted as a faster computation and less used memory. This turns out to be wrong. We have compared the timings of the two different strategies for random examples.

In most cases, ALL strategy was faster than JUST. We found examples for which ALL was 38 000 times faster than JUST and examples where JUST was 2 300 times faster than ALL. In the tables below we present the different timings we obtained for our examples. All of the timings are represented in milliseconds. The input ideals can be found in the Appendix on page 131.

	ALL time	JUST time	Factor
Example A:1	6 630	251 481 340	38 000
Example A:2	40	63 740	1 600
Example A:3	51 570	6 134 060	119
Example A:4	4 400	522 340	118
Example A:5	250	19 080	76
Example A:6	4 110	271 460	66
Example A:7	56 720	2 676 080	47
Example A:8	190	8 340	44
Example A:9	1 131 470	39 694 950	35
Example A:10	110	3 020	27

Table 1.1: Best examples for ALL strategy

	ALL time	JUST time	Factor
Example B:1	1 696 700	730	2 324
Example B:2	252 950	450	562
Example B:3	4 090	30	136
Example B:4	35 160	500	70
Example B:5	3 690	110	33

Table 1.2: Best examples for JUST strategy

All these examples where randomly generated. We ran the following file in the two versions of SINGULAR and computed the timings for the two strategies:

```
system("--ticks-per-sec",1000); //sets timer to ms
ring r = integer,(x,y,z),dp;    //defines the ring
ideal i = ...;                  //defines the ideal
int t = timer;                  //sets the timer
ideal g = std(i);               //computes the standard basis
"Time:",timer-t;                //prints the timer difference
```

As seen in the tables, it is not possible to know from the input whether ALL or JUST strategy will be faster in computing the standard bases. All the examples we found where JUST strategy was faster, have as a third generator a pure monomial. But this seems to be randomly as some examples that are faster with ALL strategy also have as a third generator a monomial (see Examples A:1, A:4 and A:10 on page 131).

We searched for such examples over finite rings, like $\mathbb{Z}_{2^{100}}$ or $\mathbb{Z}_{10^{200}}$ but we were unable to find interesting examples over these finite rings. The reason why we found examples over $\mathbb{Z}$ is that here there are no bounds for coefficients. In order to give an idea of how huge the coefficients can be in the standard bases computation over $\mathbb{Z}$, we computed Example B:1 over $\mathbb{Z}_{10^{200}}$ with the ALL strategy and got the result after 8 seconds; over $\mathbb{Z}_{10^{1000}}$ we obtained the result after 648 seconds (10 minutes) and over $\mathbb{Z}$ we got the result after almost 30 hours. One can imagine how big the coefficients are when computing over $\mathbb{Z}[x]$. This is the main reason why in some cases, the computation of a standard basis over $\mathbb{Z}$ will be slow.

A:1.4 Huge coefficients over the integers

One of the most common problems when computing over $\mathbb{Z}[x]$ is that the coefficients will rapidly increase. If during the algorithm we have two polynomials with big coefficients, when computing their s−polynomial (or strong polynomials) we multiply them with the corresponding monomials, and hence the coefficients will become larger. This will slow down the algorithm and increase the memory usage. In order to keep the coefficients as small as possible we have implemented some easy tricks.

SINGULAR 's polynomial reduction procedure `reduce`(f,g) only reduces f by g if $\mathrm{LT}(g)$ divides $\mathrm{LT}(f)$. So, if f and g are equal to

$$f = 78\,234\,678\,956\,783\,785\,656\,788\,689 \cdot x^{13}y^{19} + 987\,237\,527\,429\,289 \cdot x^5y^{11}$$

$$g = 1\,024 \cdot x^2y^{10},$$

then the result of `reduce`(f,g) will be equal to f. But that is a huge waste of memory since we could simply use f `mod` $g = x^{13}y^{19} - 423 \cdot x^5y^{11}$ afterwards. Not only this affects the coefficients of f but will also cause smaller coefficients of $\text{s-poly}(f,-)$ and $\text{gcd-poly}(f,-)$.

We examined several examples for which SINGULAR hanged before, most of the time because it ran out of memory, and tested them against the new strategies with positive results. Despite that, the most interesting one of them (we will refer to this as *Example 18* – can be found in the Appendix on page 133), the resulting standard basis was *small* enough (9 polynomials with maximum length 3 and 2 digit coefficients) and even contained a constant. After checking *Example 18* we observed that it took a lot of time until a monomial (or constant) would have entered the partial standard basis (line 6 in Algorithm A:1.2.10). As before, this gave rise to big coefficients filling the memory up. With the help of G. Pfister and D. Popescu, we implemented a way to access a constant (or a monomial) early in the algorithm. This we explain in the following.

ALGORITHM A:1.4.1 (preIntegerCheck)**.**

Let $I = \langle f_1, \ldots, f_r \rangle$ be an ideal in $\mathbb{Z}[x]$ and $<$ a fixed monomial ordering. We use the following idea to get access to a constant or a monomial (if it exists) from I. This is useful when computing over $\mathbb{Z}$ because adding this constant to the generating system of I before starting the standard basis algorithm will keep the coefficients small.

The main idea is described as follows: we want to compute a standard basis over $\mathbb{Z}[x]$. We will first compute the standard basis over $\mathbb{Q}[x]$. If the result is 1, then we know there is a constant in the ideal and we can get access to it by using some Singular tricks.

Algorithm preIntegerCheck(ideal I)

```
J = {1, f_1, ..., f_r}
compute $S$, a standard basis of $I$ over $\mathbb{Q}[x]$
compute the syzygies $Z \subset \bigoplus_{i=0}^{r} \mathbb{Q}[x] \cdot \varepsilon_i$ of $J$
if $S = \langle 1 \rangle$ then
    search in $Z$ for a syzygy where the 0 component consists of just a constant $c \cdot \varepsilon_0$
    return $I \cup \{c\}$
else
    search in $Z$ for a syzygy where the 0 component consists of just a term $c \cdot m \cdot \varepsilon_0$
    if such monomial is found then
        return $I \cup \{c \cdot m\}$
return $I$
```

Note that Algorithm A:1.4.1 is very costly and in the case when the algorithm doesn't find a constant or monomial, it increases the standard basis run-time without bringing any new information. However in the other cases, the algorithms run-time will improve because we have a bound on the coefficients. This is very useful over $\mathbb{Z}$ since the biggest problem proves to be the size of the coefficients that appear during the standard basis computation.

Example A:1.4.2. Let $I = \langle x+4, xy+9, x-y+8 \rangle$ be an ideal in $\mathbb{Z}[x,y]$. We compute the syzygies $Z \subset \bigoplus_{i=0}^{3} \mathbb{Q}[x,y]$ of $J = \{1, I\} =: \{1, f_1, f_2, f_3\}$. The result of syz$(J)$ in Singular is

```
_[1]=[7,-x-4,1,x]
_[2]=[y-4,-1,0,1]
_[3]=[x+4,-1]
```

Indeed,

$$7 \cdot 1 + (-x-4) \cdot f_1 + 1 \cdot f_2 + x \cdot f_3 = 0.$$

Since the standard basis of I over $\mathbb{Q}[x,y]$ is 1, it is expected that one of the syzygies has on the first component a constant $-$ in this case it is the first syzygy. We continue by computing the standard basis of $\langle 7, I \rangle$ over $\mathbb{Z}[x,y]$.

This was a very easy example. For a more interesting one we refer to the Example 18 found in the Appendix at page 133.

A:1.5 Strong pairs in the reduction procedure

During testing, another interesting example came to our attention. Recall the definition of a normal form A:1.2.7.

Suppose that we want to reduce a polynomial $f \in R[x]$ with respect to a set of polynomials G and assume that we have a *local* or *mixed* monomial ordering. The first try to extend the field case algorithm to the ring case for the local orderings would look like the following

```
1: $h = f$
2: $T = G$
3: **while** $h \neq 0$ and there is a $g \in T$ with $\mathrm{LT}(g) | \mathrm{LT}(h)$ **do**
4:    choose such $g$ with minimal ecart
5:    **if** $\mathrm{ecart}(g) > \mathrm{ecart}(h)$ **then**
6:       $T = T \cup \{h\}$
7:    $h = \text{s-poly}(h, g)$
8: **return** $h$
```

The first obvious thing that had to be changed over rings was in line 3, where instead of checking the divisibility of the leading monomials, one has to consider the leading terms. But this is not enough, at least in the case of local and mixed orderings as shown in Example A:1.5.1.

EXAMPLE A:1.5.1. Let $\mathbb{Z}[x, y]$ with the local monomial ordering `ds`. Consider the ideal

$$I = \left\langle 6 + y + x^2, 4 + x \right\rangle .$$

Then a standard basis of I with respect to `ds` is

$$S = \left\{ 2 - x + y + x^2, x - 2y - x^2 - xy - x^3 \right\} .$$

The `reduce` procedure from SINGULAR computes a weak normal form and it would be expected to see that `reduce`$(4 + x, S) = 0$. This was not the case: after running out of memory SINGULAR crashed. We explain the reason for that in the following.

We print the first steps of the above described algorithm for $f = 4+x$ and $G = S$. On the arrows we print the polynomial with which we reduced and we put a box around the polynomial if we had to reduce with a bigger ecart as in line 6 of the algorithm (and hence we add it to T).

$$T = \left\{ 2 - x + y + x^2, x - 2y - x^2 - xy - x^3 \right\}$$

$$\begin{tikzcd}
\boxed{4+x} \arrow[d, "2-x+y+x^2"] \\
\boxed{3x-2y-2x^2} \arrow[d, "x-2y-x^2-xy+x^3"] \\
4y+x^2+3xy+3x^3 \arrow[d, "2-x+y+x^2"] \\
x^2+5xy-2y^2+3x^3-2x^2y \arrow[d, "x-2y-x^2-xy-x^3"] \\
7xy-2y^2+4x^3-x^2y+x^4 \arrow[d, "x-2y-x^2-xy-x^3"] \\
12y^2+4x^3+6x^2y+7xy^2+x^4+7x^3 \arrow[d, dotted] \\
\ldots
\end{tikzcd} \tag{A:1.5}$$

At the end of the displayed steps, T will be

$$T = \left\{2-x+y+x^2, x-2y-x^2-xy-x^3, 4+x, 3x-2y-2x^2\right\}.$$

In order to give an idea of how big the example is, after 100 reduction steps we obtain:

$$\begin{aligned} h_{100} = \; & -1\,166\,313\,310\,086\,309 \cdot x^4y^{12} + 444\,754\,080\,892\,792 \cdot x^3y^{13} - 2\,213\,835\,453\,852 \cdot x^2y^{14} \\ & +112\,603\,082\,300\,511 \cdot x^7y^{10} - 720\,925\,840\,590\,079 \cdot x^6y^{11} + 430\,571\,412\,704\,270 \cdot x^5y^{12}, \end{aligned}$$

after 200 reduction steps we get

$$\begin{aligned} h_{200} = \; & 3\,637\,133\,524\,532\,445\,205\,884\,946\,278 \cdot x^5y^{26} - 2\,575\,374\,732\,708\,631\,350\,945\,040\,914 \cdot x^4y^{27} \\ & +835\,831\,707\,443\,157\,464\,048\,106\,896 \cdot x^3y^{28} - 1\,065\,123\,810\,503\,984\,516\,294\,535\,627 \cdot x^2y^{29} \\ & +173\,721\,443\,393\,369\,916\,682\,497\,810 \cdot xy^{30} + 1\,798\,249\,423\,092\,570\,138\,839\,547\,003 \cdot x^7y^{25} \\ & -1\,198\,832\,948\,728\,380\,092\,559\,698\,002 \cdot x^6y^{26}, \end{aligned}$$

and after 30 seconds

$$\begin{aligned} h_{30\text{ sec}} = {} & 3\,913\,503\,459\,539\,899\,164\,871\,865\,014\,196\,288\,690\,571\,138\,467\,618\,789\,418\,184\,838\,401\,337\,877 \cdot x^7y^{66} \\ & -7\,977\,092\,659\,639\,617\,123\,404\,184\,624\,123\,573\,036\,579\,747\,268\,029\,770\,715\,047\,606\,399\,387\,680 \cdot x^6y^{67} \\ & +53\,248\,176\,383\,593\,860\,383\,685\,894\,608\,357\,176\,403\,606\,618\,276\,029\,538\,211\,550\,919\,072\,921\,382 \cdot x^5y^{68} \\ & -274\,849\,303\,713\,564\,778\,057\,611\,697\,931\,914\,924\,087\,132\,281\,903\,387\,330\,927\,889\,415\,586\,742\,531 \cdot x^4y^{69} \\ & +1\,479\,142\,607\,536\,352\,118\,229\,469\,735\,534\,158\,627\,248\,215\,618\,509\,941\,658\,782\,249\,670\,418\,449\,889 \cdot x^3y^{70} \\ & +1\,479\,142\,607\,536\,352\,118\,229\,469\,735\,534\,158\,627\,248\,215\,618\,509\,941\,658\,782\,249\,670\,418\,449\,889 \cdot x^2y^{71} \\ & +4\,273\,779\,148\,510\,141\,908\,674\,250\,546\,167\,872\,417\,997\,942\,162\,329\,880\,059\,847\,103\,655\,340\,794\,093 \cdot xy^{72} \\ & -1\,917\,640\,585\,348\,831\,342\,301\,240\,093\,900\,677\,015\,082\,052\,593\,613\,273\,906\,209\,784\,441\,919\,747\,994 \cdot y^{73} \\ & -8\,102\,746\,632\,854\,636\,451\,624\,398\,571\,806\,695\,014\,588\,968\,214\,137\,281\,517\,130\,672\,630\,390 \cdot x^{12}y^{62} \\ & +137\,341\,917\,777\,610\,511\,092\,869\,801\,797\,066\,794\,306\,699\,920\,105\,225\,395\,995\,610\,075\,468\,198 \cdot x^{11}y^{63} \\ & -540\,404\,286\,129\,841\,284\,758\,689\,195\,319\,879\,351\,691\,659\,685\,294\,164\,765\,559\,251\,450\,330\,567 \cdot x^{10}y^{64} \\ & +397\,726\,261\,007\,869\,923\,948\,576\,228\,831\,933\,897\,595\,235\,015\,998\,652\,517\,116\,071\,021\,642\,120 \cdot x^9y^{65} \\ & +2\,356\,138\,563\,161\,310\,566\,373\,010\,452\,267\,195\,402\,915\,889\,568\,716\,606\,153\,637\,319\,213\,421\,046\,099 \cdot x^3y^{71} \\ & -1\,917\,640\,585\,348\,831\,342\,301\,240\,093\,900\,677\,015\,082\,052\,593\,613\,273\,906\,209\,784\,441\,919\,747\,994 \cdot x^2y^{72}. \end{aligned}$$

After less than minute we obtain polynomials whose coefficients are already one page long.

Together with Christian Eder and Anne Frühbis-Krüger we developed the following strategy. In line 6 of Algorithm A:1.5.2, instead of adding just the reducer h, we also add all the gcd-polys of h with $g \in T$.

Looking back at the reductions presented in Equation A:1.5 we see that already at the first step, one had to reduce with a bigger ecart and we added $4 + x$ to T. Now we also have to add $\text{gcd-poly}(4+x, 2-x+y+x^2)$ and $\text{gcd-poly}(4+x, x-2y-x^2-xy-x^3)$. Luckily in this case none of the pairs gives us something new. In the second step we also reduced with a bigger ecart and we have to add to T the new polys

$$\text{gcd-poly}(3x - 2y - 2x^2, 4 + x) = x + 2y + 3x^2$$

and $\text{gcd-poly}(3x - 2y - 2x^2, 2 - x + y + x^2) = x - 2y - x^2 - xy - x^3$ (it is already in T). Below we print again the reduction of $4 + x$ with S from Equation A:1.5 with this new strategy. With the dashed lines we display the new changes:

$$T = \left\{2 - x + y + x^2, x - 2y - x^2 - xy - x^3\right\},$$

$$\begin{aligned} &\boxed{4+x} \xrightarrow{2-x+y+x^2} \boxed{3x-2y-2x^2} \qquad T = T \cup \left\{ \boxed{x+2y+3x^2} \right\} \\ &\xrightarrow{x-2y-x^2-xy+x^3} 4y+x^2+3xy+3x^3 \\ &\xrightarrow{2-x+y+x^2} x^2+5xy-2y^2+3x^3-2x^2y \\ &\overset{x+2y+3x^2}{\dashrightarrow} 3xy-2y^2-2x^2y \\ &\overset{3x-2y-2x^2}{\dashrightarrow} 0 \end{aligned} \tag{A:1.6}$$

At the end of the reduction

$$T = \left\{2 - x + y + x^2, x - 2y - x^2 - xy - x^3, 4 + x, 3x - 2y + 3x^2, x + 2y + 3x^2\right\}.$$

Algorithm A:1.5.2.

So, it is clear now that this is the improved algorithm of reduction over a principal ideal ring with a local / mixed ordering.

```
Algorithm reduce(poly $f$, ideal $G$)
Input: $f \in R[x]$ and a finite subset $G \subset R[x]$
Output: $h \in R[x]$, a weak normal form of $f$ w.r.t. $G$
$h = f$
$T = G$
while $h \neq 0$ and there is a $g \in T$ with $\mathrm{LT}(g) | \mathrm{LT}(h)$ do
   choose such $g$ with minimal ecart
   if $\mathrm{ecart}(g) > \mathrm{ecart}(h)$ then
      $T = T \cup \{h\} \cup \{\text{gcd-poly}(h, T_i) \mid T_i \in T\}$
   $h = \text{s-poly}(h, g)$
return $h$
```

Chapter A:2

Signature standard bases over the integers

This chapter is a joint work with Christian Eder. Here we extend a variant of the F5 Algorithm (see [F]) by first considering the field case as a foundation and then generalizing it for rings.

Looking at Buchberger's Algorithm for computing Gröbner bases in Algorithm A:1.2.10, we see that one computes many s-polynomials and not all of these bring new information to our standard basis. In fact, just a small percentage of these computations will end with a non-zero polynomial. F5 uses several criteria to detect some of these useless computation. To be more specific, this criteria detects if some s$-$polynomial will reduce to 0 before computing. The idea proves to be very useful because it doesn't just avoid constructing new s$-$polynomials but also prevents the reduction to 0 of these useless pairs $-$ which is a very time consuming process.

Christian Eder has successfully implemented in Singular a variant of F5 for fields by using the Syzygy Criterion, Rewrite Criterion and F5C (see [EP]).

The following sections consist of the problems that arise over principal ideal rings and their resolution. At the end the resulted algorithm proves to be not as optimal as in the field case, but we still managed to find some interesting examples.

Throughout this chapter we have to think of the F5 algorithm as being the Gröbner basis algorithm presented in Algorithm A:1.2.10 to which we add signatures for each polynomial in the computation.

A:2.1 Definitions and notations

Faugère had the idea (see [F]) to add to each polynomial a so-called *signature* $-$ a module monomial $-$ in order to detect irrelevant s$-$polynomials before even constructing them. This will theoretically increase the memory usage (with the size of a module monomial for each polynomial occurring in the computations), but the running time will decrease in most of the field examples.

As before, let R be a ring, $R[x]$ the polynomial ring in n variables $x_1, \ldots, x_n$ shortly denoted by x and $I = \langle f_1, \ldots, f_m \rangle$ a finitely generated ideal in $R[x]$. Let $R[x]^m$ be the free module generated by $\varepsilon_1, \ldots, \varepsilon_m$ and π the projection

$$\begin{aligned} \pi : R[x]^m &\longrightarrow R[x] \\ \varepsilon_i &\longmapsto f_i \end{aligned} \tag{A:2.1}$$

REMARK A:2.1.1.

1. For each element f in I, there is a element $F \in R[x]^m$ such that $\pi(F) = f$. This is easily proven by taking the linear combination $f = \sum_{i=1}^{m} p_i \cdot f_i$ and lifting it to get $F = \sum_{i=1}^{m} p_i \cdot \varepsilon_i$. Note that F is not necessarily unique, since f can be represented in multiple linear combinations.
2. The elements of $\ker(\pi)$ are syzygies of I.

In this chapter we use as the module ordering $\prec := (C, <)$ (see Example A:1.1.15) for a global monomial ordering $<$ on $R[x]$. We extend the module ordering $\prec$ to module monomials with coefficients:

$$c_1 \cdot x^{\alpha_1} \prec c_2 \cdot x^{\alpha_2} \iff \begin{array}{l} x^{\alpha_1} \prec x^{\alpha_2} \text{ or} \\ x^{\alpha_1} = x^{\alpha_2} \text{ and } |c_1| < |c_2|, \end{array}$$

where by $|c|$ we denote the absolute value for $c \in R$.

Our purpose is to compute a Gröbner basis for I.

DEFINITION A:2.1.2 (Signatures).

Let $f \in R[x]$ be a polynomial that appears in the Gröbner basis algorithm (Algorithm A:1.2.10). We enhance f by giving it a ***signature***, denoted by $\mathrm{sig}(f)$. This is a module monomial with coefficient defined as follows:

- for an input polynomial f_i, we set

$$\mathrm{sig}(f_i) = 1 \cdot \varepsilon_i,$$

- for a s$-$polynomial of $f, g \in R[x]$ denoted by

$$p := \text{s-poly}(f, g) = c_f m_f \cdot f - c_g m_g \cdot g,$$

  where $c_f, c_g \in R$ and m_f, m_g are monomials in $R[x]$ corresponding to the s-poly computation, we define the signature as

$$\mathrm{sig}(p) = \mathrm{LT}_{\prec}(c_f m_f \cdot \mathrm{sig}(f) - c_g m_g \cdot \mathrm{sig}(g)),$$

- we define it similarly for the gcd$-$polynomials.

The details concerning the signature of the extended s$-$polynomials will follow in Section A:2.5. For now, let us consider $R = \mathbb{Z}$.

In the following example we consider Buchberger's blueprint (Algorithm A:1.2.10) while adding these signature. It is very important for the termination and future criteria to consider each time the polynomial with the smallest signature w.r.t. $\prec$. In fact, we assume that at each step, the partial standard basis is a standard basis up to the current signature. This means that if we added to our partial standard basis S a polynomial f_i, all polynomials p with $\operatorname{sig}(p) \prec \operatorname{sig}(f_i)$ will reduce to 0 with respect to S. In other words, the signature of each newly added polynomial to the standard basis will be greater (or equal) to the previous one. This is crucial for the criteria, without which the algorithm will significantly slow down. We revisit this subject in Section A:2.3.

When working with signatures, the first problem that arises is in the reduction step. We defined the signature for the initial elements, the s$-$polynomials and the gcd$-$polynomials. Because of the need to have increasing signature, the reduce procedure from Algorithm A:1.5.2 will have to be updated.

ALGORITHM A:2.1.3 (reduce procedure for signature algorithm)**.**

Let $f \in R[x]$ be a polynomial with signature $\operatorname{sig}(f)$ and the set $G = \{g_1, \ldots, g_s\} \subset R[x]$ with the corresponding signatures $\operatorname{sig}(G) = \{\operatorname{sig}(g_1), \ldots, \operatorname{sig}(g_s)\}$. Then the reduction procedure of f with respect to G is the following:

Algorithm reduce(poly f, ideal G)

```
1: $h = f$;
2: $\operatorname{sig}(h) = \operatorname{sig}(f)$;
3: while exists $g \in G$ s.t. $\operatorname{LT}(g) \mid \operatorname{LT}(h)$ do
4:     find $c \in R$ and the monomial $m \in R[x]$ such that $\text{s-poly}(h, g) = h - cm \cdot g$;
5:     if $\operatorname{sig}(h) \succ cm \cdot \operatorname{sig}(g)$ then
6:         $h = \text{s-poly}(h, g)$;
7: return $h$;
```

It is trivial to see that c and m from lines 4 and 5 are equal to $c = \dfrac{\operatorname{LC}(h)}{\operatorname{LC}(g)}$ respectively $m = \dfrac{\operatorname{LM}(h)}{\operatorname{LM}(g)}$.

REMARK A:2.1.4. Note that we only allow the reductions that preserve the signature of the pair h. If the reduction changes the signature, then either the signature vanishes or we have a decrease in signature contradicting our assumption. We explain this in detail.

Firstly, on line 5 of the algorithm we do not allow reductions if $\operatorname{sig}(h) \prec cm \cdot \operatorname{sig}(g)$. This is a consequence of the fact that we need to have the partial standard basis a standard basis up to the current signature. Assume that we have an unreduced element p. At this point we know that S is a standard basis up to $\operatorname{sig}(p)$. During the reduction process we find a $g \in G$ such that $\operatorname{LT}(g) \mid \operatorname{LT}(p)$ and $\operatorname{sig}(p) \prec \dfrac{\operatorname{LT}(p)}{\operatorname{LT}(g)} \cdot \operatorname{sig}(g)$. If we allow this reduction than we would add to the partial standard basis the polynomial $p_{\text{red}} := p - \dfrac{\operatorname{LT}(p)}{\operatorname{LT}(g)} \cdot g$ with a signature $\operatorname{sig}(p_{\text{red}}) \succ \operatorname{sig}(p)$. We add the new s$-$polynomials and gcd−polynomials (with signature greater or equal than $\operatorname{sig}(p')$). It may happen that there is already in the pair list L an earlier element s with $\operatorname{sig}(p) \preceq \operatorname{sig}(s) \prec \operatorname{sig}(p')$. If we cannot reduce this element it would be added to the partial standard basis and contradict the assumption of our algorithm − that the signature of the new element has to be greater or equal than the signature of the previous element.

Secondly, it may happen that the signatures on line 5 are equal. Then the signatures will cancel out and our new element will not have a proper signature. We call this a ***sigdrop***.

The first idea we had to overcome this second case was instead of considering just the leading term of the module element (i.e. the signature) we have stored the whole module element. We have implemented this strategy in SINGULAR and we noticed that this strategy proves to be ineffective. For once, it is used much more memory − instead of saving just the leading term, we save the whole module element (which can be arbitrary big). Moreover, as in the previous case, the assumption of the partial standard basis being a standard basis up to the current signature would be violated. Indeed, let the module element corresponding to h be

$$H := cm \cdot \operatorname{sig}(g) + c_1 m_1 \cdot \varepsilon_i + \ldots \in R[x]^m$$

and the module element corresponding to g be

$$G := \operatorname{sig}(g) + c_2 m_2 \cdot \varepsilon_j + \ldots \in R[x]^m.$$

Then the module element corresponding to the reduced h, denoted by $h' = h - c \cdot g$, is

$$H' = c_2 c m_2 m \cdot \varepsilon_j + c_1 m_1 \cdot \varepsilon_i + \ldots.$$

Assume now that $\operatorname{sig}(h') = c_2 c m_2 m \cdot \varepsilon_j \prec \operatorname{sig}(h)$ and we add h' to the partial standard basis and build the s−pairs. The next element that will be added to S (or even this one) may have a smaller signature as the previous one, contradicting our assumption.

We show what happens in the first couple of steps in Example A:2.1.5.

EXAMPLE A:2.1.5. Let $I = \langle 5y^2, 6xy + 4y^2 + 2x \rangle \subset \mathbb{Z}[x, y]$. We illustrate how the first 4 polynomials are added to the partial standard basis. The first step is to add the polynomial f_1 to the partial standard basis S:

$$f_1 = 5y^2 \qquad \Big| \qquad \operatorname{sig}(f_1) = \varepsilon_1 \ .$$

In the next step we add f_2

$$f_2 = 6xy + 4y^2 + 2x \qquad \Big| \qquad \operatorname{sig}(f_2) = \varepsilon_2 \ .$$

At this point we have to add to the pair list L $\text{s-poly}(f_1, f_2)$ and $\text{gcd-poly}(f_1, f_2)$:

$$\begin{array}{rcrcrl} \text{s-poly}(f_1, f_2) & = & -6 \cdot f_1 & + & 5y \cdot f_2 & , \ \operatorname{sig} = 5y \cdot \varepsilon_2 \\ \text{gcd-poly}(f_1, f_2) & = & -1 \cdot f_1 & + & y \cdot f_2 & , \ \operatorname{sig} = y \cdot \varepsilon_2 \end{array}$$

Since the signature of the gcd$-$polynomial is smaller, $\text{gcd-poly}(f_1, f_2)$ will be considered in the next step:

$$f_3 := \text{gcd-poly}(f_1, f_2) = xy^2 + 4y^3 + 2xy \quad \Big| \quad \text{sig}(f_3) = y \cdot \varepsilon_2 \ .$$

Hence we have to add $\text{s-poly}(f_3, f_1)$, $\text{gcd-poly}(f_3, f_1)$, $\text{s-poly}(f_3, f_2)$ and $\text{gcd-poly}(f_3, f_2)$ to the pair list L. Since both of the gcd$-$polynomials are redundant we only need to add the polynomials below to the pair list:

$$\begin{array}{lcclcl} \text{s-poly}(f_3, f_1) & = & 5 \cdot f_3 & - & x \cdot f_1 & , \ \text{sig} = 5y\varepsilon_2 \\ \text{s-poly}(f_3, f_2) & = & 6 \cdot f_3 & - & y \cdot f_2 & , \ \text{sig} = 6y \cdot \varepsilon_2 - y \cdot \varepsilon_2 = 5y \cdot \varepsilon_2 \end{array} \tag{A:2.2}$$

At this point the pair list L contains three pairs, all having the same signature $5y \cdot \varepsilon_2$. In fact all three polynomials are equal to $20y^3 + 10xy$ and therefore it doesn't actually matter which we consider first. After we reduce it with f_1, we add it to the partial standard basis:

$$f_4 := \text{s-poly}(f_2, f_3) = 10xy \qquad \Big| \quad \text{sig}(f_4) = 5y \cdot \varepsilon_2 \ .$$

REMARK A:2.1.6. When working over a field, the signature of a s$-$polynomial $\text{s-poly}(f, g)$ is set to be $\max\{c_f m_f \cdot \text{sig}(f), c_g m_g \cdot \text{sig}(g)\}$. Unfortunately, this isn't true over rings (see Equation A:2.2 from Example A:2.1.5). The reason is that over fields we do not need coefficients in the signature since we can always set the leading coefficient to be 1 by multiplying with it's inverse.

A:2.2 Criteria: Syzygy, Rewrite and F5C

This section contains several criteria used over fields in SINGULAR 's source code and how we can extend them to rings.

A very useful criteria over fields is the syzygy criterion. This will take advantage of the already zero$-$reductions and delete further pairs. The proof over fields extends automatically over rings. The difficult problem over rings is to assure that the partial standard basis is a standard basis up to the current signature. We show the difficulty in Example A:2.2.19.

LEMMA A:2.2.1 (Syzygy Criterion)**.**

Assume that in Algorithm A:2.1.3 we have reduced a polynomial s to 0 with $\text{sig}(s) = cm \cdot \varepsilon_i \neq 0$. Let f a polynomial from the pair set L with $\text{sig}(f) \succ \text{sig}(s)$. If $\text{sig}(s) \mid \text{sig}(f)$, then f will reduce to 0.

Proof.

Since $\operatorname{sig}(s) \mid \operatorname{sig}(f)$ it implies $\operatorname{sig}(f) = cm \cdot \operatorname{sig}(s)$. Denote the modules elements corresponding to s and f by $S = \operatorname{sig}(s) + \ldots$ respectively $F = \operatorname{sig}(f) + \ldots$ and let the module element $P := F - cm \cdot S$ with $\mathrm{LT}(P) \prec \operatorname{sig}(f)$. Because the partial standard basis is a standard basis up to $\operatorname{sig}(f)$ and since $\mathrm{LT}(P)$ is strictly smaller than $\operatorname{sig}(f)$, we know that $\pi(P)$ will reduce to 0 via the partial standard basis. Since S is a syzygy, $\pi(P) = \pi(F - cm \cdot S) = \pi(F) = f$. Therefore we proved that f reduces to 0. □

REMARK A:2.2.2. Note that we can use the Syzygy Criterion when building the s−polynomial or the gcd−polynomial. We can first compute it's signature and if we find a syzygy that divides this signature, then we can delete the pair.

REMARK A:2.2.3. The Syzygy Criterion proves to be very useful, as we already know some initial syzygies. Assume in our partial standard basis we have two elements f_1 and f_2, with $\operatorname{sig}(f_1) = \varepsilon_1$ and $\operatorname{sig}(f_2) = \varepsilon_2$. In this case a natural syzygy is $\mathrm{LT}(f_1) \cdot \varepsilon_2$. This is because $\pi(f_1 \cdot \varepsilon_2 - f_2 \cdot \varepsilon_1) = f_1 \cdot f_2 - f_2 \cdot f_1 = 0$.

In general, if the polynomials f_i have the signatures $\operatorname{sig}(f_i) = \varepsilon_i$ for $1 \leq i \leq n$, then we can add to the initial syzygies the following set

$$\{\mathrm{LT}(f_j) \cdot \varepsilon_k \mid j = 1, \ldots, n-1 \text{ and } j < k \leq n\}.$$

We explain this in the following example.

EXAMPLE A:2.2.4. This example represents a continuation of Example A:2.1.5. Denote by Syz the set consisting in the leading terms of the found syzygies. At this point we have the initial syzygy (see Remark A:2.2.3):

$$\mathrm{Syz} = \left\{5y^2 \cdot \varepsilon_2\right\}.$$

Recall the first steps from Example A:2.1.5:

$$\begin{array}{l|l}
f_1 = 5y^2 & \operatorname{sig}(f_1) = \varepsilon_1 \\
f_2 = 6xy + 4y^2 + 2x & \operatorname{sig}(f_2) = \varepsilon_2 \\
f_3 := \text{gcd-poly}(f_1, f_2) = xy^2 + 4y^3 + 2xy & \operatorname{sig}(f_3) = y \cdot \varepsilon_2 \\
f_4 := \text{s-poly}(f_2, f_3) = 10xy & \operatorname{sig}(f_4) = 5y \cdot \varepsilon_2
\end{array}$$

We previously added f_4 and the pair list L consists in the new $\text{s-poly}(f_4, f_i)$ and $\text{gcd-poly}(f_4, f_i)$ for $i = 1, 2, 3$:

$$\begin{array}{rcrcrcl}
\text{gcd-poly}(f_4, f_2) & = & -2 \cdot f_2 & + & f_4 & , & \operatorname{sig} = 5y \cdot \varepsilon_2 \\
\text{s-poly}(f_4, f_1) & = & -10xy \cdot f_1 & + & 5y^2 \cdot f_4 & , & \operatorname{sig} = 25y^3 \cdot \varepsilon_2 \\
\text{s-poly}(f_4, f_2) & = & -5 \cdot f_2 & + & 3 \cdot f_4 & , & \operatorname{sig} = 15y \cdot \varepsilon_2 \\
\text{s-poly}(f_4, f_3) & = & -10 \cdot f_3 & + & y \cdot f_4 & , & \operatorname{sig} = 5y^2 \cdot \varepsilon_2
\end{array}$$

The first element deleted with the syzygy criterion is $\text{s-poly}(f_4, f_1)$, since it has signature $\operatorname{sig} = 25y^3 \cdot \varepsilon_2$ and we have one syzygy with the leading term dividing the signature. In this case the $\text{s-poly}(f_4, f_1)$ is trivially already 0.

However, we cannot delete $\text{s-poly}(f_4, f_3)$ that has signature $\text{sig} = 5y^2 \cdot \varepsilon_2$ because we in the proof we needed that the syzygy's signature is strictly smaller than the signature of the considered pair.

The pair with the smallest signature in the pair list is $\text{gcd-poly}(f_4, f_2)$ and hence we add it to the partial standard basis:

$$f_5 := \text{gcd-poly}(f_2, f_4) = -2xy - 8y^2 - 4x \mid \text{sig}(f_5) = 5y \cdot \varepsilon_2 \quad .$$

With f_5 being added to the partial standard basis, we have to add the corresponding new pairs to the pair list:

$$\begin{array}{rcrcrcl}
\text{gcd-poly}(f_1, f_5) & = & x \cdot f_1 & + & 2y \cdot f_5 & , & \text{sig} = 10y^2 \cdot \varepsilon_2 \\
\text{s-poly}(f_1, f_5) & = & 2x \cdot f_1 & + & 5y \cdot f_5 & , & \text{sig} = 25y^2 \cdot \varepsilon_2 \\
\text{s-poly}(f_2, f_5) & = & f_2 & + & 3 \cdot f_5 & , & \text{sig} = 15y \cdot \varepsilon_2 \\
\text{s-poly}(f_3, f_5) & = & 2 \cdot f_3 & + & y \cdot f_5 & , & \text{sig} = 5y^2 \cdot \varepsilon_2 \\
\text{s-poly}(f_4, f_5) & = & f_4 & + & 5 \cdot f_5 & , & \text{sig} = 30y \cdot \varepsilon_2
\end{array}$$

Note that we can already delete $\text{gcd-poly}(f_1, f_5)$ and $\text{s-poly}(f_1, f_5)$ because of the syzygy criterion.

For the next step, we consider the pair with smallest signature in L. In our case $15y \cdot \varepsilon_2$ and after we reduce it with f_1, we add it to the partial standard basis.

$$f_6 := \text{s-poly}(f_2, f_5) = -10x \qquad\qquad \mid \text{sig}(f_5) = 15y \cdot \varepsilon_2 \; .$$

Note that all the new pairs will be deleted by the syzygy criterion:

$$\begin{array}{rcrcrcl}
\text{gcd-poly}(f_2, f_6) & = & 2 \cdot f_2 & + & y \cdot f_6 & , & \text{sig} = 15y^2 \cdot \varepsilon_2 \\
\text{s-poly}(f_1, f_6) & = & 10x \cdot f_1 & + & 5y^2 \cdot f_6 & , & \text{sig} = 75y^3 \cdot \varepsilon_2 \\
\text{s-poly}(f_2, f_6) & = & 5 \cdot f_2 & + & 3y \cdot f_6 & , & \text{sig} = 45y^2 \cdot \varepsilon_2 \\
\text{s-poly}(f_3, f_6) & = & 10 \cdot f_3 & + & y^2 \cdot f_6 & , & \text{sig} = 15y^3 \cdot \varepsilon_2 \\
\text{s-poly}(f_4, f_6) & = & f_4 & + & y \cdot f_6 & , & \text{sig} = 15y^2 \cdot \varepsilon_2 \\
\text{s-poly}(f_5, f_6) & = & -5 \cdot f_5 & + & y \cdot f_6 & , & \text{sig} = 15y^2 \cdot \varepsilon_2
\end{array}$$

Now the pair list consists of:

$$L = \{\text{s-poly}(f_3, f_5), \text{s-poly}(f_4, f_5), \text{s-poly}(f_2, f_4), \text{s-poly}(f_3, f_4)\} \, .$$

But all of these will reduce to 0.

Remark A:2.2.5. Over fields, it can be easily proven that each signature appears only once, i.e. if there are two pairs having the same signature, then we can randomly pick one of them. This is not true over rings, as shown in Example A:2.2.4: f_4 and f_5 have the same signature. We need f_5 in our standard basis. In this case f_5 is the gcd$-$polynomial of f_4 and f_2. One suspects that this may happen just for the gcd$-$pairs. The next examples illustrates that this is false. Also completely independent pairs can have the same signature over integers.

Example A:2.2.6. Let $I = \langle 4x^4, 4x^3 + 2x^2 + 4x \rangle \subset \mathbb{Z}[x]$ be an ideal. These are all polynomials added to S in the signature based algorithm:

$f_1 = 4x^4$	$\text{sig} = \varepsilon_1$
$f_2 = 4x^3 + 2x^2 + 4x$	$\text{sig} = \varepsilon_2$
$f_3 = \text{s-poly}(f_1, f_2) = 2x^3 + 4x^2$	$\text{sig} = x \cdot \varepsilon_2$
$f_4 = \text{s-poly}(f_2, f_3) = 6x^2 - 4x$	$\text{sig} = 2x \cdot \varepsilon_2$
$f_5 = \text{s-poly}(f_1, f_3) = -4x^2 - 8x$	$\text{sig} = 2x^2 \cdot \varepsilon_2$
$f_6 = \text{s-poly}(f_3, f_4) = -16x^2$	$\text{sig} = 2x^2 \cdot \varepsilon_2$
$f_7 = \text{s-poly}(f_2, f_4) = -10x^2 - 4x$	$\text{sig} = 2x^2 \cdot \varepsilon_2$
$f_8 = \text{gcd-poly}(f_4, f_5) = 2x^2 - 12x$	$\text{sig} = 2x^2 \cdot \varepsilon_2$
$f_9 = \text{s-poly}(f_2, f_4) = -14x^2 - 12x$	$\text{sig} = 4x^2 \cdot \varepsilon_2$
$f_{10} = \text{gcd-poly}(f_4, f_9) = -2x^2 - 20x$	$\text{sig} = 4x^2 \cdot \varepsilon_2$
$f_{11} = \text{s-poly}(f_4, f_5) = -32x$	$\text{sig} = 6x^2 \cdot \varepsilon_2$

Note that f_5, f_6, f_7 and f_8 have the same signature.

After we complete reduce the standard basis we obtain $G = \{2x^2 - 12x, 32x\} = \{f_8, -f_{11}\}$ and see that the standard basis consists just of the s-poly and gcd-poly of the same two polynomials. This is unusual and shows once again that we need both pairs in the signature based algorithm even if they may have the same signature.

A method for the F5 algorithm that we can extend with ease to the ring case is the F5C strategy developed by Eder and Perry in [EP]. This is a step in the incremental approach in computing standard basis when using the position over terms ordering $(C, <)$. We present this in the following algorithm.

Algorithm A:2.2.7 (F5C).

Let $I = \langle f_1, \dots, f_n \rangle \subset R[x]$ be an ideal. We can compute a standard basis S of I incrementally by using this algorithm.

```
Algorithm F5C($f_1, \dots, f_n$)
S = {f_1, ext-poly(f_1)}; //partial standard basis
L = ∅; //pair list
i=2;
while i < n do
    L = {s-poly(f_i, g) | g ∈ S} ∪ {gcd-poly(f_i, g) | g ∈ S} ∪ {ext-poly(f_i)};
    compute the standard basis of S ∪ {f_i} by starting F5 with the pair-set L;
    completely reduce S without considering signatures;
    i = i + 1;
return G;
```

By *completely reduce* it is understood that we only consider those elements from the standard basis whose leading terms are not divisible by other leading terms (i.e.

a minimal standard basis is computed). As stated before the signatures block some of the reductions and for instance we may obtain at the end of F5 the following two polynomials: $10x^2y$ and $2xy$ with some signatures that will block the zero-reduction. It is clear that the first one brings no new information for the standard basis.

Because of line 7, many polynomials will be deleted from S and therefore fewer pairs will be considered in the next loop of the algorithm.

We continue by explaining the algorithm in detail. After computing the standard basis S on line 6 with F5, we reduce it to S^{red} without considering the signatures. After this step the old signatures become unusable and therefore we need to reset them.

Assume that we are in the $i-$th loop of the algorithm. Denote the previous S from line 5 by $S_{i-1} = \{g_1, \dots, g_n\}$. Then we would add f_i to S_{i-1} and compute a signature standard basis of $S_{i-1} \cup \{f_i\}$ by adding just the pairs as seen in line 5. At the end of this computation the current partial standard basis will be $S_i = \{g_1, \dots, g_n, f_i, h_1, \dots, h_s\}$. After the reduction from line 7, $S_i^{\text{red}} = \{g'_1, \dots, g'_t\}$ and we set $S = S_i^{\text{red}}$ with the reseted signatures $\operatorname{sig}(g'_i) = \varepsilon_i$.

We show how the strategy works in the following example.

EXAMPLE A:2.2.8. We continue with an enhanced version of Example A:2.2.6. We add to the ideal I the polynomial $14x^5 + 3x$, and we compute the standard basis of

$$I = \left\langle 4x^4, 4x^3 + 2x^2 + 4x, 14x^5 + 3x \right\rangle = \langle F_1, F_2, F_3 \rangle .$$

This new element gets a new signature ε_3, and hence the order of the polynomials from Example A:2.2.6 will remain unmodified. Next we show the differences caused by F5C. We print a table with the standard basis S and the pair list L. On the left column is the next step without F5C and on the right side we can see the benefits of F5C.

	without F5C	with F5C
S	$f_1 = F_1 = 4x^4$ sig $= \varepsilon_1$ $f_2 = F_2 = 4x^3 + 2x^2 + 4x$ sig $= \varepsilon_2$ $f_3 = 2x^3 + 4x^2$ sig $= x \cdot \varepsilon_2$ $f_4 = 6x^2 - 4x$ sig $= 2x \cdot \varepsilon_2$ $f_5 = -4x^2 - 8x$ sig $= 2x^2 \cdot \varepsilon_2$ $f_6 = -16x^2$ sig $= 2x^2 \cdot \varepsilon_2$ $f_7 = -10x^2 - 4x$ sig $= 2x^2 \cdot \varepsilon_2$ $f_8 = 2x^2 - 12x$ sig $= 2x^2 \cdot \varepsilon_2$ $f_9 = -14x^2 - 12x$ sig $= 4x^2 \cdot \varepsilon_2$ $f_{10} = -2x^2 - 20x$ sig $= 4x^2 \cdot \varepsilon_2$ $f_{11} = -32x$ sig $= 6x^2 \cdot \varepsilon_2$ $f_{12} = F_3 = 14x^5 + 3x$ sig $= \varepsilon_3$	$f_1 = 2x^2 - 12x$ sig $= \varepsilon_1$ $f_2 = -32x$ sig $= \varepsilon_2$ $f_3 = F_3 = 14x^5 + 3x$ sig $= \varepsilon_3$

From this we can immediately see the advantages of F5C even for such a small example. Instead of adding 11 s−polynomials and 7 gcd−polynomials to the pair set L we only add 2 s-poly and 1 gcd-poly.

Remark A:2.2.9. Looking back at Example A:2.2.8 we see another advantage of the F5C: better syzygies are added.

In the run without F5C, after adding F_3 to our standard basis, we have the initial syzygies:

$$\mathrm{Syz}_{\text{no F5C}} = \left\{4x^4 \cdot \varepsilon_2, 4x^4 \cdot \varepsilon_3, 4x^3 \cdot \varepsilon_3\right\},$$

and in the variant with F5C we obtain

$$\mathrm{Syz}_{\text{with F5C}} = \left\{2x^2 \cdot \varepsilon_2, 2x^2 \cdot \varepsilon_3, 32x \cdot \varepsilon_3\right\}.$$

One can immediately see that $\mathrm{Syz}_{\text{with F5C}}$ are better since they delete more pairs than $\mathrm{Syz}_{\text{no F5C}}$.

An easy trick in optimizing F5 over rings that we have implemented is the following.

Lemma A:2.2.10 (Gcd Pair Replace)**.**

Assume that in the F5 algorithm we consider a pair p *with signature* $\mathrm{sig}(p)$*. Let* $S = \{f_1, \ldots, f_s\}$ *be the partial standard basis up to this step. If there exists a polynomial* $f \in S$ *such that* $\text{gcd-poly}(p, f)$ *has signature equal to* $\mathrm{sig}(p)$*, then we can replace* p *with* $\text{gcd-poly}(p, f)$.

Proof.

First of all, note that if such an f_i exists and we would add p to S, then $\text{gcd-poly}(p, f_i)$ would be the pair considered in the next step in the algorithm because it has the same signature as p. Since $\text{gcd-poly}(p, f)$ has the same signature as p, then $\text{gcd-poly}(p, f) = p - cm \cdot f$, for $c \in R$ and m a monomial in $R[x]$.

Next we prove that all s−polynomials and gcd−polynomials q between p and another element $g \in S$ can be reduced using $\text{gcd-poly}(p, f)$ to 0. Then q could be written as

$$q := c_p m_p \cdot p - c_g m_g \cdot g.$$

But we can also consider the pair between g and $\text{gcd-poly}(p, f)$ of the form

$$q' := c_p m_p \cdot \text{gcd-poly}(p, f) - c_g m_g \cdot g = c_p m_p \cdot (p - cm \cdot f) - c_g m_g \cdot g = q - c_p c m_p m \cdot f.$$

Note that it is obvious that we can consider only q'. □

This idea can be integrated in the reduction procedure for the signature based algorithm F5 (Algorithm A:2.1.3).

ALGORITHM A:2.2.11.

The following procedure will search if for a polynomial p there exists $f \in S$ as in Lemma A:2.2.10.

```
Algorithm sbaCheckGcdReplace(poly $p$, ideal $S$)
for $f \in S$ do
    if $\mathrm{sig}\left(\text{gcd-poly}(p, f)\right) = \mathrm{sig}(p)$ then
        return $\text{gcd-poly}(p, f)$
return $p$
```

Now the reduction procedure for the signature algorithm (Algorithm A:2.1.3) can be updated to the following.

- On the first line, we replace f by sbaCheckGcdReplace(f, S)
- Before returning h, (i.e. no more reductions found), we replace h by sbaCheckGcdReplace(h,S), and if this is a new polynomial, try again the reduction procedure.

In fact, we can add to the reduction procedure the strategy presented in Section A:1.4 where pure monomials were used to shorten the coefficients of polynomials. Because we are working with signatures, we only allow the reductions where the signature will not change.

The following example illustrates the benefits of the strategy presented in Lemma A:2.2.10.

EXAMPLE A:2.2.12. Let $I = \langle 28y^2, 16x^2 + 7xy \rangle =: \{F_1, F_2\} \subset \mathbb{Z}[x, y]$ be an ideal. The box contains the element that vanishes during the run of the algorithm with the strategy from Lemma A:2.2.10 enabled.

$$\begin{array}{l|l}
f_1 = F_1 = 28y^2 & \mathrm{sig} = \varepsilon_1 \\
f_2 = F_2 = 16x^2 + 7xy & \mathrm{sig} = \varepsilon_2 \\
f_3 = \text{gcd-poly}(f_1, f_2) = 4x^2y^2 + 14xy^3 & \mathrm{sig} = 2y^2 \cdot \varepsilon_2 \\
\boxed{f_4 = \text{s-poly}(f_1, f_2) = 21xy^3} & \boxed{\mathrm{sig} = 7y^2 \cdot \varepsilon_2} \\
f_5 = \text{gcd-poly}(f_4, f_1) = -7xy^3 & \mathrm{sig} = 7y^2 \cdot \varepsilon_2 \\
f_6 = \text{s-poly}(f_1, f_3) = 14xy^3 & \mathrm{sig} = 14y^2 \cdot \varepsilon_2 \\
f_7 = \text{gcd-poly}(f_4, f_3) = x^2y^3 & \mathrm{sig} = 7xy^2 \cdot \varepsilon_2 \\
f_8 = \text{gcd-poly}(f_2, f_6) = -2x^2y^3 & \mathrm{sig} = 14xy^2 \cdot \varepsilon_2 \\
f_9 = \text{gcd-poly}(f_4, f_2) = -x^2y^3 & \mathrm{sig} = 21xy^2 \cdot \varepsilon_2
\end{array}$$

In this case the only element that can be replaced is f_4, and therefore all pairs with f_4 are irrelevant. This is a small example, in larger examples the benefits of this approach by shortening the pair list L and the partial standard basis S are more obvious. This example also illustrates the main disadvantage in working with signatures: when using the normal Buchberger Algorithm, instead of f_7, f_8, f_9 we would simply have f_7, but with the F5 Algorithm, we cannot reduce f_8 with f_7. If we allow this reduction the result would look like

$$f_8 + 2 \cdot f_7$$

with signature $28xy^2 \cdot \varepsilon_2$, and hence the signature would increase, contradicting our assumption. For the same reason we cannot reduce f_9 with f_7.

REMARK A:2.2.13. An idea that comes in mind to avoid adding irrelevant elements to the partial standard bases such as f_8, f_9 in the previous example would be to reduce always reduce them without considering the signature.

Another important criteria that deletes useless pairs in the field case is the Rewrite Criterion. In this we use the signature of the polynomial that have been already added to the partial standard basis. The proof in the field case can be found in [EP].

LEMMA A:2.2.14 (Rewrite Criterion)**.**
Let $S = \{f_1, \dots, f_s\}$ be a partial signature standard basis and $p := \text{s-poly}(f_i, f_j)$. Assume that there exists a $f_k \in S$ such that $\text{sig}(f_k) \mid \text{sig}(p)$ and $i, j \leq k$. Then p contains no new information since we can rewrite it using the actual standard basis. In other words, we can delete this pair from L.

The proof uses the fact that the signature of a s−polynomial $\text{s-poly}(f, g)$ is either a multiple of $\text{sig}(f)$ or a multiple of $\text{sig}(g)$. But over rings, as seen in Remark A:2.1.6, this does not hold, and will cause the deletion of critical pairs needed by the algorithm as shown in the following example.

EXAMPLE A:2.2.15. Let $I = \langle 8y^2, 4x^2y + 5y^3 + 7y^2 + 9x \rangle$ be an ideal in $\mathbb{Z}[x, y]$ with the dp ordering. Assume that we use the Rewrite Criterion also over rings. In the following we show each polynomial that is added to the standard basis.

$$\begin{array}{l|l}
f_1 = 8y^2 & \text{sig} = \varepsilon_1 \\
f_2 = 4x^2y + 5y^3 + 7y^2 + 9x & \text{sig} = \varepsilon_2 \\
f_3 = \text{gcd-poly}\big(\text{s-poly}(f_1, f_2), f_1\big) = 2y^4 + 14y^3 + 18xy & \text{sig} = 2y \cdot \varepsilon_2 \\
f_4 = \text{s-poly}(f_3, f_1) = 72xy & \text{sig} = 8y \cdot \varepsilon_2 \\
f_5 = \text{gcd-poly}\big(\text{s-poly}(f_2, f_4), f_1\big) = -2y^3 - 126y^2 - 162x & \text{sig} = 8xy \cdot \varepsilon_2 \\
f_6 = \text{s-poly}(f_1, f_5) = -648x & \text{sig} = 32xy \cdot \varepsilon_2 \\
f_7 = \text{gcd-poly}\big(\text{s-poly}(f_2, f_3), f_3\big) = -y^6 + 28x^2y^3 + 5y^5 + 36x^3y + 27xy^3 & \text{sig} = 4x^2y \cdot \varepsilon_2
\end{array}$$

The next critical pair that should go in the standard basis would be

$$f_8 = \text{s-poly}(f_2, f_5) = 5y^5 - 252x^2y^2 + 7y^4 - 324x^3 + 9xy^2 \quad \Big| \; \text{sig} = 16x^3y \cdot \varepsilon_2,$$

but this is deleted by the Rewrite Criterion: $\mathrm{sig}(f_7) \mid \mathrm{sig}(f_8)$, $2 < 7$ and $5 < 7$; and therefore leads to an incorrect standard basis, instead of the correct one:

$$\begin{aligned}
&648 \cdot x,\\
&8 \cdot y^2,\\
&72 \cdot xy,\\
&2 \cdot y^3 + 126 \cdot y^2 + 162 \cdot x,\\
&4 \cdot x^2y + 5 \cdot y^3 + 7 \cdot y^2 + 9 \cdot x,\\
&y^5 - 324 \cdot x^3 - 315 \cdot xy^2 - 3\,969 \cdot y^3 - 5\,103 \cdot xy.
\end{aligned}$$

Because of the deleted polynomial, we cannot reconstruct the last polynomial, which would be $\text{gcd-poly}(f_8, f_3)$ with $\mathrm{sig} = 16x^3y \cdot \varepsilon_2$.

Unfortunately we were unable to develop a version of Rewrite Criterion that would correctly function over rings.

We are now ready to continue the idea from Remark A:2.1.4 concerning the sigdrop. When working over a field, Rewrite Criterion deletes the pairs before they would come into a situation where the signature would vanish. Therefore in the case of fields, there are no signature drops.

REMARK A:2.2.16. Recall that in the field case, all the polynomials and signature have the leading coefficient 1.

In fact, the field version of the Rewrite Criterion is the following:

Let $f_i, f_j \in S$ be two polynomials in the partial standard basis, $m_i = \mathrm{LM}(f_i)$, $m_j = \mathrm{LM}(f_j)$, $L = \mathrm{lcm}(m_i, m_j)$ such that

$$\text{s-poly}(f_i, f_j) = \frac{L}{m_i} \cdot f_i - \frac{L}{m_j} \cdot f_j =: \lambda_i \cdot f_i - \lambda_j \cdot f_j.$$

If there is an f_k with $k \geq i, j$ such that $\mathrm{sig}(f_k) \mid \lambda_i \cdot \mathrm{sig}(f_i)$ or $\mathrm{sig}(f_k) \mid \lambda_j \cdot \mathrm{sig}(f_j)$, then we can delete $\text{s-poly}(f_i, f_j)$.

LEMMA A:2.2.17.

Over fields, the signature of a s−polynomial will never vanish.

Proof.

Assume we have a s−polynomial for which the signature vanishes. Using the notation from the previous remark,

$$\text{s-poly}(f_i, f_j) = \lambda_i \cdot f_i - \lambda_j \cdot f_j,$$

and

$$\lambda_i \cdot \mathrm{sig}(f_i) = \lambda_j \cdot f_j.$$

Assume that $\mathrm{sig}(f_i) \prec \mathrm{sig}(f_j)$ (i.e. $i < j$). Then we can apply the Rewrite Criterion for f_i and delete the pair. □

In the ring case this does not hold, as the following easy example reveals.

Example A:2.2.18 (Vanishing signature)**.** Consider the ideal generated by $F_1 = 7x^2y + 2x$ and $F_2 = 8xy^2 + 5xy + 2y^2$. In the following we print the next 10 steps of the algorithm.

$$\begin{array}{l|l}
f_1 = F_1 = 8xy^2 + 5xy + 2y^2 & \text{sig} = \varepsilon_1 \\
f_2 = F_2 = 7x^2y + 2x & \text{sig} = \varepsilon_2 \\
f_3 = \text{gcd-poly}(f_1, f_2) = -x^2y^2 - 5x^2y - 2xy^2 + 2xy & \text{sig} = y \cdot \varepsilon_2 \\
f_4 = \text{gcd-poly}\left(\text{s-poly}(f_2, f_1), f_1\right) = 2xy^2 + 26xy + 4y^2 + 10x & \text{sig} = 8y \cdot \varepsilon_2 \\
f_5 = \text{s-poly}(f_4, f_1) = 99xy + 14y^2 + 40x & \text{sig} = 32y \cdot \varepsilon_2 \\
f_6 = \text{gcd-poly}(f_4, f_5) = xy^2 + 14y^3 - 1234xy - 196y^2 - 490x & \text{sig} = 32y^2 \cdot \varepsilon_2 \\
f_7 = \text{s-poly}(f_4, f_5) = 28y^3 - 2494xy - 396y^2 - 990x & \text{sig} = 64y^2 \cdot \varepsilon_2 \\
f_8 = \text{s-poly}(f_3, f_4) = 16x^2y + 10x^2 + 4xy & \text{sig} = 8xy \cdot \varepsilon_2 \\
f_9 = \text{gcd-poly}\left(\text{gcd-poly}_{\text{red}}(f_4, f_2), f_2\right) = -x^2y + 30x^2 - 186xy - 28y^2 - 94x & \text{sig} = 24xy \cdot \varepsilon_2
\end{array}$$

The next element that we would add to S is

$$f_{10} = \text{gcd-poly}(f_5, f_2) = x^2y + 14xy^2 + 40x^2 - 28x$$

with $\text{sig} = 32xy \cdot \varepsilon_2$. But, when building the pair

$$\text{s-poly}(f_{10}, f_6) = x \cdot f_6 - y \cdot f_{10}$$

we see that $x \cdot \text{sig}(f_6) = 32xy^2 \cdot \varepsilon_2 = y \cdot \text{sig}(f_{10})$. Therefore the signature vanishes and we have to stop the algorithm.

The example below represents another case when we have to terminate the algorithm.

Example A:2.2.19 (Decreasing Signature)**.** Consider over $\mathbb{Z}[x, y]$ the ideal I generated by $F_1 = 5x^2y + 8y^2$ and $F_2 = 6x^3 + 7xy + y$. In the following we print the first 8 steps in the algorithm.

$$\begin{array}{l|l}
f_1 = F_1 = 5x^2y + 8y^2 & \text{sig} = \varepsilon_1 \\
f_2 = F_2 = 6x^3 + 7xy + y & \text{sig} = \varepsilon_2 \\
f_3 = \text{gcd-poly}(f_1, f_2) = x^3y - xy^2 + y^2 & \text{sig} = y \cdot \varepsilon_2 \\
f_4 = \text{s-poly}(f_1, f_2) = -13xy^2 + 5y^2 & \text{sig} = 5y \cdot \varepsilon_2 \\
f_5 = \text{gcd-poly}(f_1, f_4) = -x^2y^2 + 10xy^2 + 40y^3 & \text{sig} = 10xy \cdot \varepsilon_2 \\
f_6 = \text{gcd-poly}\left(\text{s-poly}(f_1, f_4), f_4\right) = -xy^2 + 104y^3 + 10y^2 & \text{sig} = 25xy \cdot \varepsilon_2 \\
f_7 = \text{s-poly}(f_1, f_5) = 50xy^2 + 208y^3 & \text{sig} = 50xy \cdot \varepsilon_2 \\
f_8 = \text{s-poly}(f_4, f_5) = 125xy^2 + 520y^3 & \text{sig} = 125xy \cdot \varepsilon_2
\end{array}$$

When we try to add f_8 to our standard basis, by trying to build $\text{gcd-poly}(f_7, f_8)$ we notice that the signature decreases:

$$\begin{aligned}\text{gcd-poly}(f_7, f_8) &= -2 \cdot f_7 + f_8 = 25xy^2 + 104y^3,\\ \text{sig}\left(\text{gcd-poly}(f_7, f_8)\right) &= 25xy \cdot \varepsilon_2 \prec 125xy \cdot \varepsilon_2 = \text{sig}(f_8).\end{aligned}$$

This decreasing signature, from now on referred to as a sigdrop, will cause us to end the algorithm, because we would contradict the assumption that the partial standard basis is a standard basis up to the current signature.

A:2.3 SigDrops

In the previous section we have seen that over rings, sigdrops occur and that they lead to the termination of the algorithm since otherwise it could return an incorrect result. This section describes what can be done after a sigdrop occurs.

We give a trivial example that happens often in a signature standard basis computation over rings. Assume that we added to S a polynomial $f_p := \text{s-poly}(f_i, f_j)$ with signature $\text{sig}(f_p)$ and that in the next step we consider the same polynomial with the same signature but from a different pair $\text{s-poly}(f_k, f_l) = f_p$. We do not want to add the same polynomial with the same signature in the partial standard basis, but using the reduction procedure shown in Algorithm A:2.1.3, we see that we cannot reduce $\text{s-poly}(f_k, f_l)$ by f_p since they have the same signature and it would vanish causing a sigdrop. A solution to this would be to also let reduction when the signatures are equal and whenever we have sigdrops (caused by vanishing signature as in Example A:2.2.18 or a decreased signature as in Example A:2.2.19), we reduce the polynomial with the reduction procedure from the normal Buchberger Algorithm without considering the signatures. If after this reduction the polynomial is 0, we can continue with the algorithm – since it was an irrelevant element. However, if after the reduction the element is still non zero, then we have to terminate the algorithm.

After these sigdrops, we have to restart the algorithm with the partial standard basis but with reinitialized new signatures. As we saw sigdrops can happen either in the reduction step or when building the s–polynomials and gcd–polynomials.

Consider that we run the signature based algorithm for the input ideal $I_1 = \{F_1, \ldots, F_m\}$.

Having the F5C strategy in mind, the partial standard basis S_1 will look

$S_1 =$

f_1	sig $= \varepsilon_1$
f_2	sig $= \varepsilon_2$
$\cdots$	
f_s	sig $= \varepsilon_s$
f_{s+1}	sig $= c_1 m_1 \cdot \varepsilon_s$
f_{s+2}	sig $= c_2 m_2 \cdot \varepsilon_s$
$\cdots$	
f_{s+t}	sig $= c_t m_t \cdot \varepsilon_s$

Assume for the moment that the sigdrop was caused in the easier case – in the reduction step – and that after trying to reduce the polynomial f_{s+t+1} without considering signatures, it still remains nonzero.

Looking at F5C (or using the idea behind the incremental computation) it is clear that $f_s = F_u \in I_1$.

A natural way to continue after this sigdrop is to start again the algorithm with the new input $I_2 = \{f_1, \ldots, f_{s+t+1}, F_{u+1}, \ldots, F_m\}$ with new signatures as in F5C and delete most of the elements in the pair list L_1 – since the signatures will change and we will have to also recompute the signatures of the pairs. The only pairs from L_1 that we do not delete are the unused initial polynomials $F_{u+1}, \ldots, F_m$. These elements have to go again as initial elements.

Remark A:2.3.1. Note that we can already add $f_1, \ldots, f_{s-1}$ to the partial standard basis S_2 since in the previous run all of the pairs between them were either added to S_1 or reduced to 0. This holds due to our module ordering $(C, <)$ – all the remaining pairs in L_1 have the signature non zero on the s component, and hence it originally had to come from a pair with f_s.

Now we assume that the sigdrop was caused in a pair computation p (s-poly or gcd-poly) between the newly added polynomial f_{s+t} and an f_i from the partial standard basis. In this case, we have to add to I_2 also p.

The hard step is to choose an ordering for the elements of I_2. This ordering affects the new signatures of the polynomials. If we sort them simply by increasing ring monomial ordering, as one normally does at a beginning of the signature algorithm, then it may be possible that we would get to the same sigdrop, and hence the algorithm will be stuck in an infinite loop. We show in detail such an example.

Example A:2.3.2. Consider the ideal I generated by $F_1 = 5y^3 + 8y^2 + 6y$ and $F_2 = 8x^2y + 9x^2$ in $\mathbb{Z}[x, y]$ with the `dp` ordering. We print the first steps in the following.

$$\begin{array}{l|l}
f_1 = F_1 = 5y^3 + 8y^2 + 6y & \text{sig} = \varepsilon_1 \\
f_2 = F_2 = 8x^2y + 9x^2 & \text{sig} = \varepsilon_2 \\
f_3 = \text{gcd-poly}(f_1, f_2) = x^2y^3 - 6x^2y^2 - 18x^2y & \text{sig} = 2y^2 \cdot \varepsilon_2 \\
f_4 = \text{s-poly}(f_1, f_2) = -19x^2y^2 + 54x^2 & \text{sig} = 5y^2 \cdot \varepsilon_2 \\
f_5 = \text{gcd-poly}\big(\text{s-poly}(f_1, f_3), f_2\big) = 2x^2y^2 - 51x^2y & \text{sig} = 10y^2 \cdot \varepsilon_2 \\
f_6 = \text{gcd-poly}\big(\text{s-poly}(f_2, f_3), f_2\big) = -x^2y^2 - 81x^2y & \text{sig} = 15y^2 \cdot \varepsilon_2 \\
f_7 = \text{s-poly}(f_2, f_4) = 171x^2y + 432x^2 & \text{sig} = 40y^2 \cdot \varepsilon_2 \\
f_8 = \text{s-poly}(f_2, f_5) = -213x^2y & \text{sig} = 40y^2 \cdot \varepsilon_2
\end{array}$$

The next element considered is

$$\text{gcd-poly}(f_4, f_5) = -x^2y^2 - 459x^2y + 54x^2 \;\Big|\; \text{sig} = 95y^2 \cdot \varepsilon_2.$$

We would like to reduce this by f_6 to $-378x^2y + 54x^2$ which has signature $80y^2 \cdot \varepsilon_2$. This would cause a sigdrop since the signature decreased. After reducing as much as we can with the reduction procedure without signatures, we obtain $f_9 := -165x^2y + 54x^2$. Then we have to terminate the algorithm, add this polynomial and after sorting S_1 by the monomial ordering we start the next run with

$I_2 =$

$f_1 = 5y^3 + 8y^2 + 6y$	sig $= \varepsilon_1$
$f_2 = 8x^2y + 9x^2$	sig $= \varepsilon_2$
$f_8 = 213x^2y$	sig $= \varepsilon_3$
$f_9 = 165x^2y - 54x^2$	sig $= \varepsilon_4$
$f_7 = 171x^2y + 432x^2$	sig $= \varepsilon_5$
$f_6 = x^2y^2 + 81x^2y$	sig $= \varepsilon_6$
$f_5 = 2x^2y^2 - 51x^2y$	sig $= \varepsilon_7$
$f_4 = 19x^2y^2 - 54x^2$	sig $= \varepsilon_8$
$f_3 = x^2y^3 - 6x^2y^2 - 18x^2y$	sig $= \varepsilon_9$

Using the strategy described in Remark A:2.3.1, f_1 can be automatically added to S_2.

One can immediately see that in the second run we would build exactly the same pairs as in the first run because of the incremental approach (we would first compute a standard basis for the first two elements). Hence this would result in an infinite loop: the second run will end with the exact same sigdrop and so will the further ones.

From this example, we can immediately see what the problem that causes this infinite loop is. We have to put the element that caused the sigdrop on the first position (after the elements that will be automatically added to the partial standard basis as described in Remark A:2.3.1). If we do this, the algorithm will end with the correct standard basis: the first run will be the exactly the same, but we start the second run with:

$I_2 =$

$f_1 = 5y^3 + 8y^2 + 6y$	sig $= \varepsilon_1$
$f_9 = 165x^2y - 54x^2$	sig $= \varepsilon_2$
$f_2 = 8x^2y + 9x^2$	sig $= \varepsilon_3$
$f_3 = x^2y^3 - 6x^2y^2 - 18x^2y$	sig $= \varepsilon_4$
$f_4 = -19x^2y^2 + 54x^2$	sig $= \varepsilon_5$
$f_5 = 2x^2y^2 - 51x^2y$	sig $= \varepsilon_6$
$f_6 = -x^2y^2 - 81x^2y$	sig $= \varepsilon_7$
$f_7 = 171x^2y + 432x^2$	sig $= \varepsilon_8$
$f_8 = -213x^2y$	sig $= \varepsilon_9$

We print what happens in the second run in the following:

$$
\begin{array}{l|l}
g_1 = f_1 = 5y^3 + 8y^2 + 6y & \text{sig} = \varepsilon_1 \\
g_2 = f_9 = 165x^2y - 54x^2 & \text{sig} = \varepsilon_2 \\
g_3 = \text{s-poly}(g_1, g_2) = -318x^2y^2 - 198x^2y & \text{sig} = y^2 \cdot \varepsilon_2 \\
g_4 = \text{gcd-poly}(g_1, g_2) = 3x^2y^2 - 4\,230x^2y & \text{sig} = 14y^2 \cdot \varepsilon_2 \\
g_5 = \text{s-poly}(g_2, g_3) = -16\,614x^2y & \text{sig} = 55y^2 \cdot \varepsilon_2 \\
g_6 = \text{gcd-poly}(g_5, g_2) = 3x^2y - 70\,686x^2 & \text{sig} = 715y^2 \cdot \varepsilon_2 \\
g_7 = \text{s-poly}(g_5, g_2) = -299\,052x^2 & \text{sig} = 3\,025y^2 \cdot \varepsilon_2 \\
g_8 = \text{gcd-poly}(g_3, g_1) = -x^2y^3 + 620x^2y^2 + 17\,954\,244x^2 & \text{sig} = 2y^3 \cdot \varepsilon_2 \\
g_9 = \text{gcd-poly}(g_6, g_1) = x^2y^3 - 8\,468x^2y^2 - 141\,372x^2 & \text{sig} = 28y^3 \cdot \varepsilon_2
\end{array}
$$

At this point we are done with elements with signature on the second component, so we can apply F5C and obtain

$$
\begin{array}{l|l}
h_1 = g_7 = 299\,052x^2 & \text{sig} = \varepsilon_1 \\
h_2 = g_1 = 5y^3 + 8y^2 + 6y & \text{sig} = \varepsilon_2 \\
h_3 = g_6 = 3x^2y - 70\,686x^2 & \text{sig} = \varepsilon_3 \\
h_4 = g_8^{\text{red}} = x^2y^3 - 620x^2y^2 - 11\,124x^2 & \text{sig} = \varepsilon_4 \\
h_5 = \text{gcd-poly}(f_2, h_3) = -x^2y + 212\,067x^2 & \text{sig} = \varepsilon_5 \\
h_6 = \text{s-poly}(h_3, h_5) = 266\,463x^2 & \text{sig} = 3 \cdot \varepsilon_5 \\
h_7 = \text{gcd-poly}(h_1, h_6) = 1\,917x^2 & \text{sig} = 165 \cdot \varepsilon_5
\end{array}
$$

At this point we are done with elements that have the signature in the third component, so we can apply F5C. We end with

$$
\begin{array}{l|l}
s_1 = h_7 = 1\,917x^2 & \text{sig} = \varepsilon_1 \\
s_2 = h_2 = 5y^3 + 8y^2 + 6y & \text{sig} = \varepsilon_2 \\
s_3 = h_5^{\text{red}} = x^2y - 1\,197x^2 & \text{sig} = \varepsilon_3
\end{array}
$$

All the others polynomials that we have to consider, $f_3, \dots, f_8$, will reduce to 0 with respect to $S = \{s_1, s_2, s_3\}$. So, the signature based algorithm will end with the correct standard basis. In this example we needed two runs of the algorithm to complete the algorithm.

In general two problems may appear. The first issue that can arise is that if we would not allow all the reductions, we would add a lot of useless polynomials and build pairs between them. If we would not allow sigdrops in the reduction step, and just reduce when the signature remains the same, then one easily can find examples where we actually have the real standard basis in the partial standard basis, but we cannot reduce all the pairs.

The second issue that may appear is that after a sigdrop, we put the element that caused it at the beginning. It may happen that some pairs with this element would reduce with an element that we had in the previous loop in the partial standard basis, but now it appears later and hence we do not have access to them yet.

In the following we display some examples. On the first column the standard basis is displayed (computed with `std` from SINGULAR) and the next columns will correspond to the different signature algorithm runs. The symbol “√ ” signifies that the polynomial has been added to the partial standard basis in the signature based algorithm and a dot “.” if we added to S an element with the same leading monomial but different coefficient (for instance $13x^2 + 2xy$ instead of $x^2 - y$).

EXAMPLE A:2.3.3. For the previous example this is how the table looks like:

	Run 1	Run 2
$1\,917x^2$		..√
$5y^3 + 8y^2 + 6y$	√	√
$x^2y - 1\,197x^2$	....	...√

EXAMPLE A:2.3.4. Let $I = \langle 10xy^2 + 9y, 5x^2y + x^2 + 9xy\rangle \subset \mathbb{Z}[x, y]$ be an ideal. The standard basis of I computed with the command `std` in SINGULAR is $S = \{s_1, \ldots, s_5\}$, where

$$\begin{aligned} s_1 &= 18xy + 162y^2 - 81y \\ s_2 &= 2x^2 + 63xy + 81y \\ s_3 &= 810y^3 - 405y^2 - 81y \\ s_4 &= xy^2 + 3564y^3 + 9xy - 1701y^2 - 396y \\ s_5 &= x^2y - 2916y^3 - 7x^2 - 225xy + 1458y^2 \end{aligned}$$

With the signature based algorithm we find 5 sigdrops, so we have to run it 6 times. The corresponding table is

	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6
$18xy + \ldots$	√			.	....	..√
$2x^2 + \ldots$	√			...√.....	..√.	..√...
$810y^3 + \ldots$		..	..	......√	√	√
$xy^2 + \ldots$	...√	.	.	..	..√	√
$x^2y + \ldots$	...√	.	.	...........√	.....√	..√

Note that in the first run, we already had 4 out of 5 polynomials in the partial standard basis. But because of a sigdrop, we had to restart. Without this sigdrop, in the next 2 steps we would have been able to find also the last polynomial $810y^3 + \ldots$. Because of this we had to restart and each time we added bigger polynomials to S. We needed to do 5 extra runs to find the correct standard basis.

Also note that in Run 4, after the polynomial $2x^2 + \ldots$ has been added, we still add polynomials of the form $2 \cdot cx^2 + \ldots$ to the partial standard basis because we are not able to reduce them (due to signatures).

These problems happen quite often and a technique to overcome this is to combine the signature algorithm with the normal Buchberger Algorithm: we let the signature algorithm to run up to maximal N times, and afterwards we use the normal standard basis algorithm on the intermediate result in order to obtain a Gröbner basis.

A:2.4 Timings

This section contains some of the most interesting examples over $\mathbb{Z}$ that finish faster with the signatures algorithm compared to the normal algorithm. For all these examples we set $N = 1$, i.e. after the first sigdrop we use the Buchberger Algorithm. The input ideals for these examples can be found in the Appendix at page 137. These examples were randomly generated. The first column contains the time needed by the classical standard basis algorithm implemented in SINGULAR (STD) and the second column holds the timing for signature based algorithm (SBA). The times are displayed in milliseconds.

	STD Time	SBA Time	Factor
Example C:1	10 430	600	17
Example C:2	19 390	100	193
Example C:3	48 250	400	120
Example C:4	83 470	20	4 173
Example C:5	22 702 640	5 958 520	4
Example C:6	112 230	570	196
Example C:7	3 828 960	48 700	78
Example C:8	76 230	30	2 541
Example C:9	403 480	5 030	80
Example C:10	9 393 780	10 360	906
Example C:11	21 792 210	34 640	629
Example C:12	1 904 710	170	11 204
Example C:13	21 198 950	129 420	163
Example C:14	1 563 870	92 490	16
Example C:15	43 520	140	310
Example C:16	854 240	119 980	7
Example C:17	10 684 360	185 090	57

The above timings where computed by using the following script in SINGULAR :

```
system("--ticks-per-sec",1000); //sets timer to ms
ring r = integer,(x,y,z),dp;
ideal i = ...;
int tstd = timer;              //will store the timing for std
ideal std_i = std(i);
tstd = timer - tstd;
int tsba = timer;              //will store the timing for sba
ideal sba_i = sba(i,1,0);
tsba = timer - tsba;
```

In these examples SBA proves to perform much better. The reason for this is that until the sigdrop, we get access to a polynomial that without signatures will take much more pairs to obtain. After the sigdrop we continue with STD.

However, in the case when we set $N = 1$ (just one SBA run is allowed), most examples will finish almost equally fast for the two algorithms. If we set $N > 1$, then SBA will become much slower than the STD algorithm for reasons illustrated in Example A:2.3.4.

A:2.5 Finite rings $- \mathbb{Z}_m$

In the previous sections we have considered $R = \mathbb{Z}$. The strategies, however, hold for $\mathbb{Z}_m$, for a $m \in \mathbb{N}$, as well, provided we set also the signature for an extended s$-$polynomial. There are two different natural ways to define them.

The first one would be to consider them as completely new elements and assign them a new signature.

Definition A:2.5.1.

For each extended s$-$polynomial of a polynomial f we enlarge our module with one more component ε_{m+1} and we set

$$\operatorname{sig}\big(\text{ext-poly}(f)\big) = 1 \cdot \varepsilon_{m+1}.$$

The second one, takes into consideration the definition of a extended s$-$polynomial.

Definition A:2.5.2.

We set the signature of an extended s$-$polynomial of a polynomial $f = c_f \cdot m_f + \ldots$ by

$$\operatorname{sig}\big(\text{ext-poly}(f)\big) = \operatorname{Ann}(c_f) \cdot \operatorname{sig}(f).$$

Over $\mathbb{Z}_m[x]$ we will often have sigdrops since we take also coefficients in signatures in consideration, this signature may become 0 as the next example shows. Because of this, the algorithm does not perform well for ideals in $\mathbb{Z}_m[x]$.

Example A:2.5.3. Let $R = \mathbb{Z}_6[x]$, $f = 3x$ with $\operatorname{sig}(f) = 3x \cdot \varepsilon_2$ and $g = 4x^3 + 3x$ with $\operatorname{sig}(g) = x \cdot \varepsilon_2$. We want to compute $\operatorname{sig}\big(\text{s-poly}(f, g)\big)$. Then

$$\text{s-poly}(f, g) = 4x^2 \cdot f - 3 \cdot g = 3x,$$

and therefore

$$\operatorname{sig}\big(\text{s-poly}(f, g)\big) = \operatorname{LT}(4x^2 \cdot 3x \cdot \varepsilon_2 + 3 \cdot x \cdot \varepsilon_2) = \operatorname{LT}(3x \cdot \varepsilon_2) = 3x \cdot \varepsilon_2.$$

Note that signature of the s$-$polynomial has dropped (since the coefficient of x^3 was $12 \equiv 0 \bmod 6$) and this would cause the algorithm to end.

For both definitions (Definition A:2.5.1 and Definition A:2.5.2) one can find examples where the vanishing signature or a decreasing signature cause a sigdrop ending the algorithm.

EXAMPLE A:2.5.4. In this example we use Definition A:2.5.2. Let $I = \langle x^5, x^4 + x^3 + 5x^2 + 3x \rangle \subset \mathbb{Z}_6[x]$ be an ideal. We show in the following the first 4 steps of the algorithm.

$$\begin{array}{l|l} f_1 = x^5 & \text{sig} = \varepsilon_1 \\ f_2 = x^4 + x^3 + 5x^2 + 3x & \text{sig} = \varepsilon_2 \\ f_3 = \text{s-poly}(f_1, f_2) = 4x^3 + 4x^2 + 3x & \text{sig} = x \cdot \varepsilon_2 \\ f_4 = \text{ext-poly}(f_3) = 3x & \text{sig} = 3x \cdot \varepsilon_2 \end{array}$$

Since $\text{Ann}(4) = 3$, $\text{ext-poly}(f_3) = 3 \cdot f_3 = 3x$ with $\text{sig} = 3x \cdot \varepsilon_2$.

When we try to add the s$-$polynomial between f_4 and f_3,

$$\text{s-poly}\left(f_4, f_3\right) = 4x^2 \cdot f_4 - 3 \cdot f_3,$$

we see that the signature is $\text{LT}(12x^3 \cdot \varepsilon_2 - 3x \cdot \varepsilon_2) = 3x \cdot \varepsilon_2$, since the first bigger term would vanish since we are working over $\mathbb{Z}_6[x]$. However, this will not lead to the termination of the algorithm despite the decreasing signature, because $\text{sig}\left(\text{s-poly}(f_3, f_4)\right) = 3x \cdot \varepsilon_2 = \text{sig}(f_4)$, which was the current signature. The algorithm continues by adding the last polynomial to the standard basis:

$$\begin{array}{l|l} f_5 = \text{s-poly}(f_2, f_3) = x^2 & \text{sig} = x^2 \cdot \varepsilon_2, \end{array}$$

EXAMPLE A:2.5.5. If we compute the above example with the definition of the signature of an extended s$-$polynomial as in Definition A:2.5.1, then we obtain a sigdrop, causing the termination of the algorithm:

$$\begin{array}{l|l} f_1 = x^5 & \text{sig} = \varepsilon_1 \\ f_2 = x^4 + x^3 + 5x^2 + 3x & \text{sig} = \varepsilon_2 \\ f_3 = \text{s-poly}(f_1, f_2) = 4x^3 + 4x^2 + 3x & \text{sig} = x \cdot \varepsilon_2 \end{array}$$

We add to the pair list L, the extended s$-$polynomial $\text{ext-poly}(f_3)$ with signature ε_3, so this element will be considered after all the other pairs.

The next element that will be added to S is

$$\begin{array}{l|l} f_4 = \text{s-poly}(f_2, f_3) = x^2 & \text{sig} = x^2 \cdot \varepsilon_2, \end{array}$$

In the next step we consider after the reduction $\text{s-poly}(f_1, f_3) = 5x^3 + 4x^2$ with $\text{sig} = x^3 \cdot \varepsilon_2$. Using the strategy presented in Lemma A:2.2.10, we replace this with $\text{gcd-poly}\left(\text{s-poly}(f_1, f_3), f_3\right) = x^3 + 3x$ with $\text{sig} = x^3 \cdot \varepsilon_2$. After reducing it with f_4, we obtain the polynomial $3x$ with no signature, causing the algorithm to terminate.

Part B

Depth, Stanley Depth and Hilbert Depth

Introduction

Part B starts recalling some preliminaries on Cohen−Macaulay Modules, Hilbert Functions and Hilbert Polynomials. Chapter 2 introduces the multigraded Stanley depth and the graded Hilbert depth (shortly sdepth_n and hdepth_1) presenting also the Stanley Conjecture. Afterwards we present an implemented algorithm in Singular that computes the Hilbert Depth published in Journal Symbolic Computation ([AP2]). Several computational experiments are included. In particular, these show that if $R = K[x_1, \ldots, x_6]$, for a field K, and $m = \langle x_1, \ldots, x_6 \rangle$, then $\operatorname{hdepth}_1(R \oplus m) = 4 > 3 = \operatorname{hdepth}_1(m)$ (see Remark B:2.2.4). This hints that it might holds also $\operatorname{sdepth}_n(R \oplus m) > \operatorname{sdepth}_n(m)$, which is a negative solution for a question of J. Herzog. This is indeed the case as it was stated by B. Ichim and A. Zarojanu in [IZ].

Chapter 3 starts recalling some cases when the multigraded Stanley Conjecture holds obtained by D. Popescu, A. Zarojanu and Y.H. Shen ([DP5], [PZ],[DP6], [Sh]) in the frame of monomial squarefree ideals. Unfortunately, the multigraded Stanley Conjecture does not hold in general as [DuGo] shows (see here Example B:3.1.8). The computer algebra system Normaliz ([BIS]) uses the multigraded Stanley decomposition to compute the Hilbert Series. The case when the multigraded Stanley Conjecture fails makes difficult to find the best multigraded Stanley decomposition and so the computation by Normaliz is harder.

Theorem B:3.2.13 (see [AP1]) shows that the Stanley Conjecture holds for intersections of three monomial prime ideals. The proof uses a special decomposition extended later by others (see [DP7], [HPV]) under the name of splitting of variables and used for example in [RTU] and [Ta].

Let $J \subsetneq I \subset R = K[x_1, \ldots, x_n]$ be monomial square free ideals generated by monomials of degree $\geq d$. Then $\operatorname{depth} \dfrac{I}{J} \geq d$. If among the minimal set of monomial generators of I there exist at most 3 squarefree monomials of degree d and $\operatorname{sdepth}_n \dfrac{I}{J} = d + 1$ we show that $\operatorname{depth} \dfrac{I}{J} \leq d + 1$ (see Theorem B:3.1.7).

Let $G(I),\ G(J)$ be the sets of minimal monomial generators of I, respectively J, where $J \subsetneq I \subset R$ and $d_i,\ i \in \{1, \ldots, n\}$ the maximum degree of x_i which appears in a monomial from $G(I) \cup G(J)$. We say that ${}^{I}\!/_{J}$ is in the *canonical form* if for any $i \in \{1, \ldots, n\}$ and $t < d_i$, there exist a monomial from $G(I) \cup G(J)$ of degree t in x_i. In Chapter 4 we introduced a factor of monomial ideals $\overline{{}^{I}\!/_{J}}$ associated to ${}^{I}\!/_{J}$ − the canonical form. We have $\operatorname{sdepth}_n {}^{I}\!/_{J} = \operatorname{sdepth}_n \overline{{}^{I}\!/_{J}}$ by [IKM, Proposition 5.1] (see here Proposition B:4.0.1) and $\operatorname{depth}_n {}^{I}\!/_{J} = \operatorname{depth}_n \overline{{}^{I}\!/_{J}}$ by Theorem B:4.1.11. Using the canonical form, we were able to extend Rinaldo's Stanley depth algorithm to optimize the computation of depth and sdepth of ${}^{I}/_{J}$ using its canonical form. Example B:4.1.13 shows that the algorithm is much quicker than the other known ones.

Chapter B:1

Definitions and Notations

In this chapter we give the definitions and notations that we use throughout this part. We also try to present examples to better understand notions like Stanley and Hilbert depth.

B:1.1 Cohen$-$Macaulay Modules

In this section we present some connections between the Hilbert functions and Cohen$-$Macaulay rings.

Definition B:1.1.1.
We say that an ideal $Q \subset R$ is ***primary*** if $x \cdot y \in Q$, then $x \in Q$ or $y^n \in Q$ for a certain $n \in \mathbb{N}$. If Q is a primary ideal, then the radical of Q is necessarily a prime ideal P, and this ideal is called the associated prime ideal of Q. In this situation, Q is said to be $P-$primary.

Definition B:1.1.2.
An $R-$module M is ***flat*** if the tensor product functor $M \otimes_R -$ is exact. For example the projective modules, in particular the free ones, are flat modules.

Definition B:1.1.3.
Let R be a ring and M a $R-$module. A sequence $x := x_1, \ldots, x_n,\ x_i \in R\ \forall i$ is called a $M-$***regular sequence*** if

- x_i is not a zero divisor in $\dfrac{M}{(x_1, \ldots, x_{i-1})}$ for $i = 1, \ldots, n$
- $\dfrac{M}{xM} \neq 0.$

In this case n is called ***the length of*** x.

Remark B:1.1.4. If R is a polynomial ring over a field then a regular sequence x of R is the algebraic analogue of the geometric notion of the complete intersection. More precisely, ${}^{R}/_{(x)}$ defines a complete intersection if and only if x is regular.

Definition B:1.1.5.

Let (R, m) be a Noetherian local ring with m the maximal ideal. The length of a maximal $M-$regular sequence $x := x_1, \ldots, x_n$, for $x_i \in m \; \forall i$ is called the ***depth*** of M.

Remark B:1.1.6. If M is a finite $R-$module then one can define depth by using some notions from Homological Algebra [BH, Theorem 1.2.5]:

$$\operatorname{depth} M = \inf \left\{ n \;\middle|\; \operatorname{Ext}^n \left(\frac{R}{m}, M \right) \neq 0 \right\}$$

Proposition B:1.1.7 ([BH, Propositions 1.2.12, 1.2.13])**.**

Let (R, m) be a local Noetherian ring and M a finite generated $R-$module. Then $\operatorname{depth} M \leq \dim M$. Moreover, $\operatorname{depth} M \leq \dim {}^{R}/_{P}$ for all $P \in \operatorname{Ass} M$.

Definition B:1.1.8.

Let (R, m) be a local Noetherian ring, and M a $R-$module. M is called ***Cohen $-$ Macaulay module*** if $\dim M = \operatorname{depth} M$ or if $M = 0$.

A Noetherian ring R is called a ***Cohen$-$Macaulay ring*** if (R, m) is Cohen $-$ Macaulay for any maximal ideal $m \in$ Max(R).

B:1.2 Hilbert Function and Hilbert Polynomial

The Hilbert function of a graded module associates to an integer n the dimension of the $n-$th graded part of a given module. For sufficiently large n, the values of this function can be given by a polynomial, the Hilbert polynomial. To show this, we use the Hilbert$-$Poincaré series, a formal power series in t with coefficients being values of the Hilbert function.

Throughout this chapter let K be a field.

Definition B:1.2.1.

A ***graded ring*** R is a ring together with a direct sum decomposition

$$R = \bigoplus_{i \geq 0} R_i,$$

where the R_i are abelian groups satisfying $R_i R_j \subset R_{i+j}$ for any $i, j \geq 0$.

A ***graded $K-$algebra*** is a $K-$algebra which is a graded ring such that R_i is a $K-$vector space for all $i \geq 0$ and $R_0 = K$.

The R_i are called the ***homogeneous components*** and the elements of R_i are called ***homogeneous element of degree*** i.

A $R-$module M, together with a direct sum decomposition

$$M = \bigoplus_{i \geq 0} M_i$$

into abelian groups is called a ***graded*** $R-$***module*** if $R_iM_j \subset M_{i+j}$ for all $i \geq 0$ and $j \in \mathbb{Z}$.

Let $M = \bigoplus\limits_{i\geq 0} M_i$ be a graded $R-$module and define $M(d) := \bigoplus\limits_{i\in\mathbb{Z}} M(d)_i$ with $M(d)_i := M_{d+i}$. Then $M(d)$ is a graded $R-$module and it is called the $d-$***th shift*** of M.

A submodule $N \subset M$ is called a ***graded submodule*** if it is generated by homogeneous elements, or equivalently, if N is graded with the induced grading, i.e. $N = \bigoplus\limits_{i\in\mathbb{Z}}(M_i \cap N)$.

Let $R = \bigoplus\limits_{i\geq 0} R_i$ be a graded ring, $M = \bigoplus\limits_{i\in\mathbb{Z}} M_i$ and $N = \bigoplus\limits_{i\in\mathbb{Z}} N_i$ be two graded $R-$modules. A homomorphism $\varphi : M \to N$ is called ***graded of degree*** d if $\varphi(M_i) \subset N_{i+d}$ for all i. If φ is graded of degree zero, we call φ just graded.

Definition B:1.2.2.

Let $R = \bigoplus\limits_{i\geq 0} R_i$ be a Noetherian graded $K-$algebra and let $M = \bigoplus\limits_{i\in\mathbb{Z}} M_i$ be a finitely generated graded $R-$module.

The ***Hilbert function*** $\mathrm{H}_M : \mathbb{Z} \longrightarrow \mathbb{Z}$ of M is defined by

$$\mathrm{H}_M(n) := \dim_K(M_n).$$

The ***Hilbert-Poincaré series*** HP_M of M is defined by

$$\mathrm{HP}_M(t) := \sum_{i\in\mathbb{Z}} \mathrm{H}_M(i) \cdot t^i \ \in \mathbb{Z}\llbracket t\rrbracket \left[t^{-1}\right].$$

Note that the definition of H_M and HP_M depend only on the graded structure of M. Hence if $\varphi : R' \longrightarrow R$ is a graded $K-$algebra map, then it does not matter whether we consider M as a $R-$module or as a $R'-$module.

The next results are proven in [BH, Chapter 4].

Lemma B:1.2.3.

Let $R = \bigoplus\limits_{i\geq 0} R_i$ be a Noetherian graded $K-$algebra and let M be a finitely generated graded $R-$module.

1. *Let $N \subset M$ be a graded submodule. Then*

$$\mathrm{H}_M(n) = \mathrm{H}_N(n) + \mathrm{H}_{M/N}(n)$$

for all n. In particular $\mathrm{HP}_M(t) = \mathrm{HP}_N(t) + \mathrm{HP}_{M/N}(t)$.

2. *Let d be an integer. Then*

$$\mathrm{H}_{M(d)}(n) = \mathrm{H}_M(n+d)$$

for all n. In particular $\mathrm{HP}_{M(d)}(t) = t^{-d} \cdot \mathrm{HP}_M(t)$.

3. *Let d be a non-negative integer, let $f \in R_d$ and let $\varphi : M(-d) \longrightarrow M$ be defined by $\varphi(m) := f \cdot m$. Then $\mathrm{Ker}(\varphi)$ and $\mathrm{Coker}(\varphi)$ are graded ${}^{R}/_{\langle f\rangle}-$module with the induced gradings and*

$$\mathrm{H}_M(n) - \mathrm{H}_M(n-d) = \mathrm{H}_{\mathrm{Coker}(\varphi)}(n) - \mathrm{H}_{\mathrm{Ker}(\varphi)}(n-d)$$

for all n. In particular $\mathrm{HP}_M(t) - t^d \cdot \mathrm{HP}_M(t) = \mathrm{HP}_{\mathrm{Coker}(\varphi)}(t) - t^d \cdot \mathrm{HP}_{\mathrm{Ker}(\varphi)}(t)$.

THEOREM B:1.2.4.

Let $R = \bigoplus_{i\geq 0} R_i$ be a graded $K-$algebra and assume that R is generated, as $K-$algebra, by $x_1, \ldots, x_r \in R_1$. Then for any finitely generated graded $R-$module $M = \bigoplus_{i\geq 0} M_i$,

$$\mathrm{HP}_M(t) = \frac{Q(t)}{(1-t)^r}, \text{ for some } Q(t) \in \mathbb{Z}[t]. \tag{B:1.1}$$

With the notations of Theorem B:1.2.4, canceling all common factors from the equation (B:1.1) we obtain

$$\mathrm{HP}_M(t) = \frac{G(t)}{(1-t)^s}, \quad 0 \leq s \leq r, \quad G(t) = \sum_{i=0}^{d} g_i t^i \in \mathbb{Z}[t], \tag{B:1.2}$$

such that $g_d \neq 0$ and $G(1) \neq 0$, that is s is the pole order of $\mathrm{HP}_M(t)$.

THEOREM B:1.2.5.

Let R be a polynomial ring over a field and $M \neq 0$ be a finite graded $R-$module of dimension d. Assume that M is Cohen$-$Macaulay. Let $G_M(t)$ be as in Equation (B:1.2), $G_M(t) = \sum_i g_i t^i$. Then $g_i \geq 0$ for all i.

DEFINITION B:1.2.6.

Let $R = \bigoplus_{i\geq 0} R_i$ be a Noetherian graded $K-$algebra, and let $M = \bigoplus_{i\geq 0} M_i$ be a finitely generated graded $R-$module.

- The polynomials $Q(t)$ and $G(t)$ defined in (B:1.1) and (B:1.2) are called the ***first*** and the ***second Hilbert series*** of M.
- Let d be the degree of the second Hilbert series $G(t)$ and let s be the pole order of the Hilbert-Poincaré series $\mathrm{HP}_M(t)$ at $t = 1$. Then

$$\mathrm{P}_M(n) := \sum_{i=0}^{d} g_i \cdot \left(\begin{array}{c} s-1+n-i \\ s-1 \end{array} \right) \in \mathbb{Q}[n], \tag{B:1.3}$$

where $\binom{n}{k} = 0$ for $k < 0$, is called the ***Hilbert polynomial*** of M.

Definition B:1.2.7.

Recall Equation (B:1.2). We define the ***multiplicity*** of the module M as

$$e(M) = G(1).$$

One can prove that $e(M) > 0$ $\big($ see [BH] $\big)$.

Definition B:1.2.8.

Let R be a ring and M be a $R-$module. The ***dimension*** of M is defined by $\dim M = \dim \frac{R}{\operatorname{Ann}(M)}$.

The ***length*** of M, denoted by $\ell(M)$, is defined to be the largest number n such that there exists a chain of submodules of length n, i.e. $N_0 \subsetneq N_1 \subsetneq \ldots \subsetneq N_n$, for N_i submodules of M.

Remark B:1.2.9. Note that in the case of the Hilbert function definition (Definition B:1.2.2), M_n were seen as $K-$vector spaces and then $\ell(M_n)$ is the dimension of the vectorial space M_n over K.

Definition B:1.2.10.

An ***associated prime*** of a module M over a ring R is a prime ideal of R of the form $\operatorname{Ann}_R z$ for some nonzero $z \in M$. The set of all associated prime ideals is denoted by $\operatorname{Ass} M$.

Theorem B:1.2.11 $\big($Hilbert's Theorem$\big)$.

Let M be a finite graded $R-$module of dimension d. Then $\mathrm{HP}_M(n)$ is of polynomial type of degree $d-1$.

Remark B:1.2.12. Theorem B:1.2.11 tells us that the s from (B:1.2) is in fact $\dim M$:

$$\mathrm{HP}_M(t) = \frac{G(t)}{(1-t)^{\dim M}}.$$

Recall that $e(M) = G(1) > 0$.

Proposition B:1.2.13.

With the above assumptions, P_M is a polynomial in n with rational coefficients, of degree $s-1$, and satisfies $\mathrm{P}_M(n) = \mathrm{H}_M(n)$ for $n \geq d$. Moreover, there exists $a_i \in \mathbb{Z}$ such that

$$\mathrm{P}_M(n) = \sum_{i=0}^{s-1} a_i \cdot \binom{n}{i} = \frac{a_{s-1}}{(s-1)!} \cdot n^{s-1} + \text{lower terms in } n,$$

where $a_{s-1} = G(1) > 0$.

CHAPTER B:2

HILBERT DEPTH

B:2.1 Hilbert Series and Hilbert Depth

The results of this chapter are published in [AP2].

Let K be a field and $R = K[x_1 \ldots, x_n]$ be the polynomial algebra over K in n variables. On R consider the following two grading structures:

- the $\mathbb{Z}$ $-$grading in which each x_i has degree 1 and
- the multigraded structure, i.e. the $\mathbb{Z}^n$ $-$grading, in which each x_i has degree the $i-$th vector e_i of the canonical basis.

DEFINITION B:2.1.1.

After Bruns-Krattenthaler-Uliczka [BKU] (see also [Sh]), a ***Hilbert decomposition*** of a $\mathbb{Z}$ $-$graded $R-$module M is a finite family

$$\mathcal{H} = (R_i, s_i)_{i \in I}$$

in which $s_i \in \mathbb{Z}$ and R_i is a $\mathbb{Z}$ $-$graded $K-$algebra retract of R for each $i \in I$ such that

$$M \cong \bigoplus_{i \in I} R_i(-s_i)$$

as a graded $K-$vector space.

The ***Hilbert depth*** of $\mathcal{H}$ denoted by $\operatorname{hdepth}_1 \mathcal{H}$ is the depth of the $R-$module $\bigoplus_{i \in I} R_i(-s_i)$.

The ***Hilbert depth*** of M is defined as

$$\operatorname{hdepth}_1(M) = \max\{\operatorname{hdepth}_1 \mathcal{H} \mid \mathcal{H} \text{ is a Hilbert decomposition of } M\}.$$

We set $\operatorname{hdepth}_1(0) = \infty$.

It is known that $\operatorname{hdepth}_1(M) \geq \operatorname{depth}(M)$ (see [BKU]).

DEFINITION B:2.1.2.

A Laurent series in $\mathbb{Z} \llbracket t, t^{-1} \rrbracket$ is called ***positive*** if it has only non-negative coefficients.

Theorem B:2.1.3 (Uliczka [Uli]).

$\operatorname{hdepth}_1(M) = \max\{e \mid (1-t)^e \operatorname{HP}_M(t)$ is positive$\}$, *where* $\operatorname{HP}_M(t)$ *is the Hilbert – Poincaré series of* M.

If M is a multigraded $\mathbb{Z}^n-$module, then one can define $\operatorname{hdepth}_n(M)$ as above by considering the $\mathbb{Z}^n-$grading instead of the standard one. There exists an algorithm for computing the hdepth_n of a finitely generated multigraded module M over the standard multigraded polynomial ring $K[x_1,\ldots,x_n]$ in Ichim and Moyano-Fernández's paper [IM] (see also [IZ]).

The main purpose of this chapter is to provide an algorithm to compute $\operatorname{hdepth}_1(M)$, where M is a graded $R-$module (see Algorithm B:2.1.7). We mention that later another such algorithm appeared in [BMU] and containing ours at the references.

Definition B:2.1.4.

A ***Stanley decomposition*** (see [St]) of a $\mathbb{Z}-$graded (resp. $\mathbb{Z}^n-$graded) $R-$module M is a finite family

$$\mathcal{D} = (R_i, u_i)_{i\in I}$$

in which u_i are homogeneous elements of M and R_i is a graded (resp. $\mathbb{Z}^n-$graded) $K-$algebra retract of R for each $i \in I$ such that $R_i \cap \operatorname{Ann}(u_i) = 0$ and

$$M = \bigoplus_{i\in I} R_i u_i$$

as a graded $K-$vector space.

The ***Stanley depth*** of $\mathcal{D}$ denoted by $\operatorname{sdepth}\mathcal{D}$ is the depth of the $R-$module $\bigoplus\limits_{i\in I} R_i u_i$.

The ***Stanley depth*** of M is defined as

$$\operatorname{sdepth}(M) = \max\{\operatorname{sdepth}\mathcal{D} \mid \mathcal{D} \text{ is a Stanley decomposition of } M\}.$$

We set $\operatorname{sdepth}(0) = \infty$.

The ***Stanley Conjecture*** claims that $\operatorname{sdepth}(M) \geq \operatorname{depth}(M)$.

We talk about $\operatorname{sdepth}_1(M)$ and $\operatorname{sdepth}_n(M)$ if we consider the $\mathbb{Z}-$grading respectively the $\mathbb{Z}^n-$grading of M. The Hilbert depth of M is greater than the Stanley depth of M and can be strictly greater (an example can be found in [BKU]).

Herzog posed the following question (see also [BGS, Problem 1.67]):

Is $\operatorname{sdepth}_n(R \oplus m) = \operatorname{sdepth}_n(m)$, where m is the maximal ideal in R?

Since we implemented an algorithm to compute hdepth_1, we have tested whether $\operatorname{hdepth}_1(R\oplus m) = \operatorname{hdepth}_1(m)$ and as a consequence when $\operatorname{sdepth}_n(R\oplus m) = \operatorname{sdepth}_n(m)$.

Proposition B:2.2.6 says that Herzog's question holds for $n \in \{1,\ldots,5,7,9,11\}$, but Remark B:2.2.4 says that for $n = 6$ it holds $\operatorname{hdepth}_1(R \oplus m) > \operatorname{hdepth}_1 m$, which is a sign that in this case $\operatorname{sdepth}_n(R \oplus m) > \operatorname{sdepth}_n m$ and so Herzog's question could have a negative answer for $n = 6$. This is indeed the case as it was shown later by Ichim and Zarojanu in [IZ].

We introduce an algorithm which computes hdepth_1 (Algorithm B:2.1.7) and prove its correctness (Theorem B:2.1.8). In the next section we provide some examples and some results related to [BGS, Problem 1.67].

Remark B:2.1.5. The algorithm presented in this section is based on Theorem B:2.1.3 and at a first glance it might look trivial. The difficulty lies in the fact that it is not clear how many coefficients of the infinite Laurent series have to be checked for positivity. This paper provides a bound up to which it suffices to check.

The algorithm which we construct requires the module M as the input. In fact we only need the polynomial $G(t)$ from Equation (B:1.2) and the dimension of M.

Definition B:2.1.6.

Let $p(t) = \sum\limits_{i=0}^{\infty} a_i \cdot t^i \in \mathbb{Z}\,[\![t]\!]$ be a formal power series. By jet$_j(p)$ we understand the polynomial till order j, jet$_j(p) = \sum\limits_{i=0}^{j} a_i \cdot t^i$.

Algorithm B:2.1.7.

We now present the algorithm that computes the hdepth_1 of a $\mathbb{Z}-$graded module `M`. The algorithm uses the following procedures which can easily be constructed in any computer algebra system:

- ○ `inverse(poly p, int bound)`: computes the inverse of a power series `p` till the order `bound`,
- ○ `hilbconstruct(module M)`: computes the second Hilbert series of the module `M` - a way to do this in Singular is to use the already built-in function `hilb(module M, 2)` which returns the list of coefficients of the second Hilbert series and construct the series,
- ○ `positive(poly f)`: returns `1` if f has all the coefficients non-negative and `0` else,
- ○ `sumcoef(poly f)`: returns the sum of the coefficients of `f`,
- ○ `jet(poly p, int j)`: returns the jet$_{\mathtt{j}}$ `p`. This procedure is already implemented in Singular and
- ○ `dim(module M)`: returns the dimension of `M`. This procedure is already implemented in Singular .

Below we give the algorithm `hdepth(poly g, int dim__M)`. Hence in order to compute hdepth_1 `M`, one considers `g(t) = hilbconstruct( M )` and `dim__M = dim(M)`.

Algorithm hdepth_1 (poly g, int `dim__M`)

Input:

- a polynomial $\mathsf{g}(t) \in \mathbb{Z}[t]$ (equal to the second Hilbert series)
- an integer `dim__M` $= \dim M$

Output:

- hdepth_1`M`

```
if positive($g$) = 1 then
   return dim__M;
poly $f$ = g;
int $c$, $d$, $\beta$;
$\beta = \deg(\mathsf{g})$;
for $d$ = dim__M to $d = 0$ do
   $d = d - 1$;
   $f$ = jet( g·inverse( $(1-t)^{\mathtt{dim\_\_M}-d}$, $\beta$ ) );
   if positive($f$) = 1 then
      return $d$;
   $c$ = sumcoef($f$);
   if $c < 0$ then
      while $c < 0$ do
         $\beta = \beta + 1$;
         $f$ = jet( g · inverse( $(1-t)^{\mathtt{dim\_\_M}-d}$, $\beta$ ) );
         $c$ = sumcoef($f$);
```

THEOREM B:2.1.8.

Given a $\mathbb{Z}$ –graded module M, Algorithm B:2.1.7 correctly computes

$$\max \{n \mid (1-t)^n \cdot \mathrm{HP}_M(t) \text{ is positive } \} \tag{B:2.1}$$

where $\mathrm{HP}_M(t) = \dfrac{G(t)}{(1-t)^{\dim M}}$ is the Hilbert-Poincaré series of M. Hence, by Theorem B:2.1.3, the algorithm computes the Hilbert depth of a module M for $\mathtt{g} = G(t)$ and `dim__M`$= \dim M$.

Proof.

Note that $G(1)$ is the multiplicity of the module M and hence $G(1) > 0$.

Assume that $M \neq 0$. Denote the bound β at the end of the loop where $d = i$ by β_i. In order to prove this theorem one has to show the following two claims:

- the maximum from (B:2.1) does not exceed $\dim M$,

- after the bound β_i degree, the coefficients are non-negative.

For the first part consider $G(t) = \sum_{\mu=0}^{g} a_\mu \cdot t^\mu$. Note that

$$(1-t)^{\dim M+1} \cdot \mathrm{HP}_M(t) = (1-t) \cdot G(t) = a_0 + (a_1 - a_0) \cdot t + \ldots + (a_g - a_{g-1}) \cdot t^g - a_g \cdot t^{g+1}.$$

If all coefficients would be non-negative, we would obtain

$$0 \geq a_g \geq a_{g-1} \geq a_{g-2} \geq \ldots \geq a_2 \geq a_1 \geq a_0 \geq 0$$

which implies that $G(t) = 0$. This will lead to a contradiction with $M \neq 0$. The same holds for $(1-t)^{\dim M+\alpha} \cdot \mathrm{HP}_M(t)$ by considering $(1-t)^{\dim M+\alpha-1} \cdot \mathrm{HP}_M(t)$ instead of $G(t)$, where $\alpha \geq 0$. Thus the maximum from (B:2.1) is smaller or equal than $\dim M$.

Note that if $G(t)$ already has all the coefficients non-negative, then the algorithm stops by returning $\dim M$, and the result is correct since in this case $\mathrm{hdepth}_1 M = \dim M$.

For the second part we need to show that at each step i the coefficient of the term of order β_i in $\dfrac{G(t)}{(1-t)^{\dim M - i}}$ is non-negative and the coefficients of the terms of higher order are increasing (and hence non-negative). Apply induction on i. For the first step, $d = \dim M - 1$, $f = \dfrac{G(t)}{(1-t)}$ and all the coefficients of the terms of order $\geq \beta_{\dim M - 1} = \deg G(t)$ are equal to the sum of the coefficients $G(1) > 0$. For the general step i, assume that at the beginning of loop $d = i$, we started with $\dfrac{G(t)}{(1-t)^{\dim M - i}} = \sum_{\mu=0}^{\infty} b_\mu \cdot t^\mu$ which satisfied all the desired properties by induction: the bound β_i was increased (if required), such that the coefficient sum $c_i := \sum_{\mu=0}^{\beta_i} b_\mu > 0$ and all coefficients of higher order terms are non-negative, i.e. $b_\mu \geq 0$ for $\mu \geq \beta_{i-1}$. We now consider the next step, $d = i - 1$, and compute the new f as in line 9 of the algorithm. In order to check that the coefficients of the terms of order higher than the bound β_i are non-negative. We have:

$$\frac{G(t)}{(1-t)^{\dim M - (i-1)}} = \overbrace{b_0 + (b_0 + b_1) \cdot t + \ldots + \underbrace{\left(\sum_{\mu=0}^{\beta_i} b_\mu \right)}_{c_i > 0} \cdot t^{\beta_i}}^{= \mathrm{jet}_{\beta_i}} + (c_i + b_{\beta_i+1}) \cdot t^{\beta_i+1} + \ldots$$

By induction, $0 < b_{\beta_i} \leq b_{\beta_i+1} \leq b_{\beta_i+2} \leq \ldots$ and since $c_i > 0$ we obtain $c_i + b_{\beta_i+\nu} > 0$ for $\nu \geq 0$.

The termination of the algorithm is trivial since we know that in the last loop we would consider $\dfrac{G(t)}{(1-t)^{\dim M}} = \mathrm{HP}_M(t)$ which is positive by the definition, and hence it will return $\mathrm{hdepth}_1 M = 0$. □

Remark B:2.1.9. The maximum from the statement of [Uli, Theorem 3.2] (see here Theorem B:2.1.3) is always smaller than $\dim M$. This was not shown in Uliczka's proof and it has to be proved in Theorem B:2.1.8.

B:2.2 Computational Experiments

The following examples illustrate the usage of the implementation of the algorithm in Singular , which can be found in the Section B:2.3. Note that in the outputs we print exactly the jet we considered in our computations followed by "+. . . ".

Example B:2.2.1. Let $I = (x) \cap (y_1, \ldots, y_5)$ be an ideal in $\mathbb{Q}[x, y_1, \ldots, y_5]$. In order to compute $\operatorname{hdepth}_1(I)$ we type the following in Singular after loading the library `hdepth.lib`.

```
ring R=0,(x,y(1..5)),ds;
ideal i=intersect(x,ideal(y(1..5)));
module m=i;
"dim M = ",dim(m);
//  dim M = 5
hdepth( hilbconstruct( m ), dim(m) );
//  G(t)= 1+t-4t2+6t3-4t4+t5
//  G(t)/(1-t)^ 1 = 1+2t-2t2+4t3+t5 +...
//  G(t)/(1-t)^ 2 = 1+3t+t2+5t3+5t4+6t5 +...
//  hdepth= 3
```

Example B:2.2.2. Consider a module M for which $\operatorname{HP}_M(t) = \dfrac{2 - 3t - 2t^2 + 2t^3 + 4t^4}{(1-t)^{\dim M}}$.

Denote by `dim__M` the dimension of M.

```
ring R = 0, t, ds;
poly g = 2-3*t-2*t^2+2*t^3+4*t^4;
hdepth( g, dim__M);
// G(t)= 2-3t-2t2+2t3+4t4
// G(t)/(1-t)^ 1 = 2-t-3t2-t3+3t4+3t5 +...
// G(t)/(1-t)^ 2 = 2+t-2t2-3t3+3t5 +...
// G(t)/(1-t)^ 3 = 2+3t+t2-2t3-2t4+t5 +...
// G(t)/(1-t)^ 4 = 2+5t+6t2+4t3+2t4+3t5 +...
```

Hence, it results $\operatorname{hdepth}_1 M = \dim M - 4$.

As seen in the proof, we had to increase our bound if the coefficient sum was ≤ 0. Note that in this example, the coefficient sum of $\operatorname{jet}_4 \left(\dfrac{G(t)}{(1-t)} \right)$ is zero and thus we increase the bound to 5 (the coefficient sum of the jet_5 will be equal to $3 > 0$).

EXAMPLE B:2.2.3. Consider $R = K[x_1, \dots, x_n]$ for $n \in \{4, 5, \dots, 19\}$ and m the maximal ideal. We computed $\operatorname{hdepth}_1 m$, $\operatorname{hdepth}_1(R \oplus m)$, …, $\operatorname{hdepth}_1(R^6 \oplus m)$ and $\operatorname{hdepth}_1(R^{100} \oplus m)$. We obtain the following results:

n	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$\operatorname{hdepth}_1(m)$	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10
$\operatorname{hdepth}_1(R \oplus m)$	2	3	4	4	5	5	6	6	7	8	8	9	9	10	11	11
$\operatorname{hdepth}_1(R^2 \oplus m)$	3	3	4	4	5	6	6	7	8	8	9	10	10	11	11	12
$\operatorname{hdepth}_1(R^3 \oplus m)$	3	3	4	5	5	6	7	7	8	9	9	10	10	11	12	12
$\operatorname{hdepth}_1(R^4 \oplus m)$	3	3	4	5	6	6	7	8	8	9	9	10	11	11	12	12
$\operatorname{hdepth}_1(R^5 \oplus m)$	3	4	4	5	6	6	7	8	8	9	10	10	11	11	12	13
$\operatorname{hdepth}_1(R^6 \oplus m)$	3	4	4	5	6	7	7	8	8	9	10	10	11	11	12	13
$\operatorname{hdepth}_1(R^{100} \oplus m)$	3	4	5	6	7	8	8	9	10	11	11	12	13	13	14	15

Figure 2.1: $\operatorname{hdepth}_1(R^i \oplus m)$ computations

REMARK B:2.2.4. Note that for $n = 6$ we have $\operatorname{hdepth}_1(R \oplus m) = 4 > 3 = \operatorname{hdepth}_1 m$. This is a sign that in this case $\operatorname{sdepth}_n(R \oplus m) > \operatorname{sdepth}_n(m)$ and so Herzog's question could have a negative answer for $n = 6$. The difference $\operatorname{hdepth}_1(R \oplus m) - \operatorname{hdepth}_1(m)$ can be greater than 1 as one can see for $n = 18$.

Note that $\operatorname{hdepth}_1(R^s \oplus m) - \operatorname{hdepth}_1 m$ increases when s and n increase. For example $\operatorname{hdepth}_1(R^{100} \oplus m) - \operatorname{hdepth}_1(m) = 5$ for $s = 100$ and $n = 19$.

LEMMA B:2.2.5.

Let $n \in \mathbb{N}$ be such that $\operatorname{hdepth}_1 m = \operatorname{hdepth}_1(R \oplus m)$. Then

$$\operatorname{sdepth}_n m = \operatorname{sdepth}_n(R \oplus m).$$

Proof.

By [BKU] and [BiHo] we have $\operatorname{hdepth}_1 m = \left\lceil \frac{n}{2} \right\rceil = \operatorname{sdepth}_n m$. It is enough to see that the following inequalities hold:

$$\operatorname{hdepth}_1 m = \operatorname{sdepth}_n m \leq \operatorname{sdepth}_n(R \oplus m) \leq \operatorname{hdepth}_n(R \oplus m) \leq \operatorname{hdepth}_1(R \oplus m).$$

□

PROPOSITION B:2.2.6.

If $n \in \{1, \dots, 5, 7, 9, 11\}$ then $\operatorname{sdepth}_n m = \operatorname{sdepth}_n(R \oplus m)$, that is Herzog's question has a positive answer.

Proof.

Note that $\operatorname{hdepth}_1 m = \operatorname{hdepth}_1(R \oplus m)$ for n as above and apply Lemma B:2.2.5.

□

B:2.3 The hdepth procedure

As stated before, Algorithm B:2.1.7 was implemented as a procedure for the computer algebra system SINGULAR [Singular]. This procedure was used in order to obtain the results from Figure B:2.2.3. The additional procedures which have been used were defined in Algorithm B:2.1.7. In addition, we printed some information which we find useful for understanding the algorithm.

```
proc hdepth(poly g, int dim__M)
{
  int d;
  ring T = 0,t,ds;
  "G(t)=",g;
  if(positiv(g)==1)
    return("hdepth=",dim__M);
  poly f=g;
  number ag;
  int c1, bound;
  bound = deg(g);
  for(d = dim__M; d>=0; d--)
  {
    f = jet( g*inverse( (1-t)^(dim__M-d),bound ) , bound );
    if(positiv(f) == 1)
    {
      "G(t)/(1-t)^",dim__M-d,"=",f,"+...";
      "hdepth=",d;
      return();
    }
    c1=sumcoef(f);
    if(c1<=0)
    {
      while(c1<0)
      {
        bound = bound + 1;
        f =  jet( g*inverse( (1-t)^(dim__M-d),bound), bound);
        c1 = sumcoef(f);
      }
      "G(t)/(1-t)^",dim__M-d,"=",g,"+...";
    }
  }
}
```

Chapter B:3

The Stanley Conjecture

B:3.1 The present stay of Stanley's Conjecture and its impact in Commutative Algebra

For more than 30 years the Stanley Conjecture was a dream for many people working in combinatorics and commutative algebra. Many people believed that this conjecture holds and tried to prove directly some of its consequences. For example in this way a lower bound of depth given by Lyubeznik [Ly] was extended by Herzog et al. [HPV] for sdepth.

Some numerical upper bounds of sdepth give also upper bounds of depth, which are independent of char K. More precisely, let $S = K[x_1, \ldots, x_n]$, $n \in \mathbb{N}$, be a polynomial ring over a field K and $I \supsetneq J$ two squarefree monomial ideals of S. Then denote by $\rho_j(I \setminus J)$ for the number of all squarefree monomials of degrees j in $I \setminus J$.

Theorem B:3.1.1 ([DP6, Theorem 1.3])**.**
Assume that $\operatorname{depth}_S(I/J) \geq t$, *where* $t \in \mathbb{Z}$ *such that* $d \leq t < n$. *If* $\rho_{t+1}(I \setminus J) < \alpha_t := \sum_{i=0}^{t-d} (-1)^{t-d+i} \rho_{d+i}(I \setminus J)$, *then* $\operatorname{depth}_S(I/J) = t$ *independently of the characteristic of* K.

An extension is given below.

Theorem B:3.1.2 (Shen [Sh, Theorem 2.4])**.**
Assume that $\operatorname{depth}_S(I/J) \geq t$, *where* t *is an integer such that* $d \leq t < n$. *If for some* k *with* $d + 1 \leq k \leq t + 1$ *it holds* $\rho_k(I \setminus J) < \sum_{j=d}^{k-1} (-1)^{k-j+1} \binom{t+1-j}{k-j} \rho_j(I \setminus J)$, *then* $\operatorname{depth}_S(I/J) = t$ *independently of the characteristic of* K.

Shen's proof is very short, based on a strong tool, namely the Hilbert depth considered by Bruns-Krattenhaler-Uliczka [BKU] (see also [Uli], [IM]) that was introduced in Section B:2.1.

Let r be the number of the squarefree monomials of degrees d of I and B (resp. C) be the set of the squarefree monomials of degrees $d+1$ (resp. $d+2$) of $I \setminus J$. Set $s = |B|$,

$q = |C|$. Theorems B:3.1.1 and B:3.1.2 lead to the following:

Theorem B:3.1.3 (D. Popescu, [DP2])**.**
If $s > q + r$ *then* $\operatorname{depth}_S I/J \leq d + 1$.

Theorem B:3.1.4 (Shen, [Sh])**.**
If $s < 2r$ *then* $\operatorname{depth}_S I/J \leq d + 1$.

If $r > s$ then Theorem B:3.1.1 says that $\operatorname{depth}_S I/J = d$, namely the minimum possible.

Theorem B:3.1.5 (D. Popescu, [DP5, Theorem 4.3])**.**
If $\operatorname{sdepth}_S I/J = d$*, then* $\operatorname{depth}_S I/J = d$*, that is Stanley's Conjecture holds in this case.*

The next step in proving Stanley's Conjecture is the following weaker case.

Conjecture B:3.1.6.
Suppose that $I \subset S$ *is minimally generated by some squarefree monomials* $f_1, \ldots, f_k$ *of degrees* d*, and a set* H *of squarefree monomials of degrees* $\geq d+1$*. If* $\operatorname{sdepth}_S I/J = d+1$*, then* $\operatorname{depth}_S I/J \leq d + 1$.

A partial answer is the following theorem which is the subject of Section B:3.3.

Theorem B:3.1.7 (A. Popescu and D. Popescu [APDP1, Theorem 0.6])**.**
Let C *be the set of the squarefree monomials of degree* $d+2$ *of* $I \setminus J$*. Conjecture B:3.1.6 holds in each of the following two cases:*

1. $r \leq 3$,

2. $r = 4$, $E = \emptyset$ *and there exists* $c \in C$ *such that* $\operatorname{supp} c \not\subset \bigcup_{i=1}^{4} \operatorname{supp} f_i$.

However, Conjecture B:3.1.6 is false in general as shows an example similar to the one given in [DuGo].

Example B:3.1.8 (Stanley Conjecture Counterexample)**.** Let $n = 6$, $r = 5$, $E = \emptyset$, $I = (yt, yz, xy, xu, xv) \subset K[x, y, z, t, u, v]$, $J = (ytv, xtv, yzv, xtu)$. Then

$$\operatorname{sdepth}_S I/J < \operatorname{depth}_S I/J.$$

We should point out that the examples from [DuGo] do not say that Stanley's Conjecture is false in the frame of ideals. On the other hand, as we saw, there exist many cases in which Stanley's Conjecture holds. We should mention also that Katthän proved in

[Ka] that if J is an ideal (not necessarily squarefree) generated by at most 6 generators then Stanley's Conjecture holds for S/J.

Why is important to know the cases when Stanley's Conjecture holds? We recall that if this is the case for the Stanley-Reisner of a certain simplicial complex then this is partitionable (see [HSY]). Another reason is explained in [BIS]. More precisely, the computer algebra system NORMALIZ uses Stanley decompositions to compute the Hilbert series. Thus if Stanley's Conjecture holds, then we have an idea about how "good" could this decomposition be, i.e. how big could its sdepth be. We should recall that here we speak about *multigraded* Stanley's Conjecture which sometime could fail, but the graded Stanley's Conjecture always holds (see [BKU, Theorem 2.5]). The fact that multigraded Stanley's Conjecture fails sometimes implies that it is harder to find the best existing multigraded Stanley decomposition in order to compute the multigraded Hilbert series.

B:3.2 The Stanley Conjecture on intersections of three monomial prime ideals.

This section contains results published in [AP1]. These ideas were often used in papers like [HPV], [RTU] and [Ta]. Let $S = K[x_1, \ldots, x_n]$, I be an ideal in $K[x_1, \ldots, x_r] = S'$ and J be an ideal in $K[x_{r+1}, \ldots, x_n] = S''$, where $1 < r < n$. The following two lemmas are elementary, their proofs being suggested by [Vi, Theorem 2.2.21] and [PQ, Lemma 4.1]. We include these for the sake of our completeness.

LEMMA B:3.2.1.
With the notations as above, the following equality holds:

$$\operatorname{depth}_S(IS \cap JS) = \operatorname{depth}_{S'}(I) + \operatorname{depth}_{S''}(J).$$

Proof.
Let us consider the following exact sequence of S-modules:

$$0 \to \frac{S}{IS \cap JS} \to \frac{S}{IS} \oplus \frac{S}{JS} \to \frac{S}{IS + JS} \to 0.$$

By [Vi, Theorem 2.2.21] it follows that

$$\operatorname{depth}_S\left(\frac{S}{IS + JS}\right) = \operatorname{depth}_{S'}\left(\frac{S'}{I}\right) + \operatorname{depth}_{S''}\left(\frac{S''}{J}\right).$$

But since $\operatorname{depth}(S/IS) = \operatorname{depth}_{S'}(S'/I) + (n-r)$ and $\operatorname{depth}(S/JS) = \operatorname{depth}_{S''}(S''/J) + r$, we obtain $\operatorname{depth}\left(\frac{S}{IS} \oplus \frac{S}{JS}\right) > \operatorname{depth}\left(\frac{S}{IS + JS}\right)$. Applying Depth Lemma to the above exact sequence yields

$$\operatorname{depth}_S\left(\frac{S}{IS\cap JS}\right)=\operatorname{depth}_S\left(\frac{S}{IS+JS}\right)+1=\operatorname{depth}_{S'}\left(\frac{S'}{I}\right)+\operatorname{depth}_{S''}\left(\frac{S''}{J}\right)+1.$$

We are done since

$$\operatorname{depth}_S\left(IS\cap JS\right)=\operatorname{depth}_S\ \left(\frac{S}{IS+JS}\right)+2=\operatorname{depth}_{S'}(I)+\operatorname{depth}_{S''}(J).$$

□

The following result is analogous to [Ra, Theorem 3.1].

Lemma B:3.2.2.
With the notations from the previous lemma, the following inequality yields:

$$\operatorname{sdepth}_S(IS\cap JS)\geq\operatorname{sdepth}_{S'}(I)+\operatorname{sdepth}_{S''}(J).$$

Proof.
Let $\mathcal{M}(I)$ be the monomials from the ideal I and consider two polynomials $f\in\mathcal{M}(I)$ and $g\in\mathcal{M}(J)$. The correspondence $(f,g)\mapsto f\cdot g$ defines an injective map $\varphi:\mathcal{M}(I)\times\mathcal{M}(J)\to\mathcal{M}(IS\cap JS)$. If $w\in\mathcal{M}(IS\cap JS)$, then $w=f\cdot g$ for some monomials $f\in S'$ and $g\in S''$; hence $f\cdot g\in IS$, therefore $f\in IS\cap S'=I$ since the variables of g are free over S/IS. Similarly we obtain $g\in J$, and so φ becomes surjective since $w=\varphi(f,g)$.

Let $\mathcal{D}'$ be a Stanley decomposition of I,

$$\mathcal{D}':I=\bigoplus_{i=1}^{s}u_iK[Z_i],$$

with $\operatorname{sdepth}_{S'}(\mathcal{D}')=\operatorname{sdepth}_{S'}(I)$, and let $\mathcal{D}''$ be a Stanley decomposition of J

$$\mathcal{D}'':J=\bigoplus_{j=1}^{t}v_jK[T_i],$$

with $\operatorname{sdepth}_{S''}(\mathcal{D}'')=\operatorname{sdepth}_{S''}(J)$. $\mathcal{D}'$ and $\mathcal{D}''$ induce a Stanley decomposition over $IS\cap JS$ because φ is bijective:

$$\widetilde{\mathcal{D}}:IS\cap JS=\bigoplus_{j=1}^{t}\bigoplus_{i=1}^{s}u_iv_jK[Z_i\cup T_j].$$

Therefore,

$$\begin{aligned}\operatorname{sdepth}_S(IS\cap JS)\quad&\geq\operatorname{sdepth}_S(\widetilde{\mathcal{D}})=\min_{i,j}(|Z_i|+|T_j|)\geq\min_i|Z_i|+\min_j|T_j|=\\&=\operatorname{sdepth}_{S'}(\mathcal{D}')+\operatorname{sdepth}_{S''}(\mathcal{D}'')=\operatorname{sdepth}_{S'}(I)+\operatorname{sdepth}_{S''}(J).\end{aligned}$$

□

Remark B:3.2.3. The inequality of the above lemma can be strict as happens in the case when $n = 4$, $r = 2$, $I = (x_1, x_2)$, $J = (x_3, x_4)$. Indeed, then $\mathrm{sdepth}_{S'}(I) = 1$, $\mathrm{sdepth}_{S''}(J) = 1$, and $\mathrm{sdepth}_S(IS \cap JS) = 3 > 2 = \mathrm{sdepth}_{S'}(I) + \mathrm{sdepth}_{S''}(J)$, as shows the Stanley decomposition

$$\begin{aligned} IS \cap JS = \ & x_1x_3K[x_1, x_3, x_4] \oplus x_1x_4K[x_1, x_2, x_4] \oplus \\ & x_2x_3K[x_1, x_2, x_3] \oplus x_2x_4K[x_2, x_3, x_4] \oplus x_1x_2x_3x_4S. \end{aligned}$$

Theorem B:3.2.4.

Let $r_i \in \mathbb{N}$ such that $0 = r_0 < r_1 < r_2 < \ldots < r_s = n$ and $S = K[x_1, \ldots, x_n]$. Denote by $P_1 = (x_1, \ldots, x_{r_1})$, $P_2 = (x_{r_1+1}, \ldots, x_{r_2}), \ldots, P_s = (x_{r_{s-1}+1}, \ldots, x_{r_s})$ and $I = \bigcap_{i=1}^{s} P_i$. Then,

$$\mathrm{sdepth}\ (I) \geq \mathrm{depth}\ (I) = s.$$

In particular Stanley's Conjecture holds in this case.

Proof.

Denote by $S^i = K[x_{r_{i-1}+1}, \ldots, x_{r_i}]$. Since $\mathrm{depth}_{S^i}\left(\dfrac{S^i}{P_i \cap S^i}\right) = 0$ we obtain $\mathrm{depth}_{S^i}(P_i \cap S^i) = 1$. By Lemma B:3.2.1 and recurrence we obtain $\mathrm{depth}(I) = s$. Applying Lemma B:3.2.2, by recurrence and [Sh] it results that

$$\mathrm{sdepth}_S\ (I) \geq \sum_{i=1}^{s} \mathrm{sdepth}_{S^i}(P_i \cap S^i) = \sum_{i=1}^{s} \left\lceil \frac{r_i - r_{i-1}}{2} \right\rceil \geq s,$$

where $\lceil a \rceil$ is the lowest integer number greater or equal to $a \in \mathbb{R}$. □

Remark B:3.2.5. If $s = 2$, then Ishaq [Is, Corollary 2.9, 2.10] proved that $\mathrm{sdepth}(I) = \left\lceil \dfrac{n+1}{2} \right\rceil$ if either n is odd, or n is even but r_1 is odd, and $\dfrac{n}{2} \leq \mathrm{sdepth}(I) \leq \dfrac{n}{2} + 1$ if n and r_1 are even.

We need the following two lemmas:

Lemma B:3.2.6 ([PQ, Lemma 4.3])**.**

Let $S = K[x_1, \ldots, x_n]$ and consider the ideals $Q = (x_1, \ldots, x_t)$ and $Q' = (x_{r+1}, \ldots, x_n)$ where $1 \leq r \leq t < n$. Then

$$\mathrm{sdepth}(Q \cap Q') \geq \left\lceil \frac{r}{2} \right\rceil + \left\lceil \frac{n-t}{2} \right\rceil.$$

Lemma B:3.2.7 ([HVZ, Lemma 3.6])**.**

Let $I \subset S = K[x_1, \ldots, x_n]$ be an ideal and $S' = S[x_{n+1}, \ldots, x_t]$, where $t > n$. Then,

$$\mathrm{sdepth}_{S'}(IS') = \mathrm{sdepth}_S(I) + (t - n).$$

PROPOSITION B:3.2.8.

Let P_1,P_2,P_3 be three non-zero monomial prime ideals of S not included one in the other such that $\sum_{i=1}^{3} P_i = (x_1, \ldots, x_n)$. Let I=$P_1 \cap P_2 \cap P_3$.. Then

$$\text{depth } (I) = \begin{cases} 3, \text{ if } P_i \not\subset P_j + P_k \text{ for any different } i,j,k \in \{1,2,3\} \\ n + 2 - \max\{\mathrm{ht}(P_i + P_j), \mathrm{ht}(P_i + P_k)\}, \text{if } P_i \subset P_j + P_k \ . \end{cases}$$

Proof.

Consider the following exact sequence of S-modules:

$$0 \to \frac{S}{P_1 \cap P_2} \to \frac{S}{P_1} \oplus \frac{S}{P_2} \to \frac{S}{P_1 + P_2} \to 0. \qquad \text{(B:3.1)}$$

We have

$$\text{depth } \left(\frac{S}{P_1 + P_2}\right) = n - \mathrm{ht}(P_1 + P_2)$$

and

$$\text{depth } \left(\frac{S}{P_1} \oplus \frac{S}{P_2}\right) = \min\left\{\text{depth } \left(\frac{S}{P_1}\right), \text{depth } \left(\frac{S}{P_2}\right)\right\} = n - \max\left\{\mathrm{ht}(P_1), \mathrm{ht}(P_2)\right\}.$$

By hypotheses we know that $\mathrm{ht}(P_1 + P_2) > \max\{\mathrm{ht}(P_1), \mathrm{ht}(P_2)\}$ and hence $n - \mathrm{ht}(P_1 + P_2) < n - \max\{\mathrm{ht}(P_1), \mathrm{ht}(P_2)\}$. This leads to $\text{depth}(\frac{S}{P_1+P_2}) < \text{depth}(\frac{S}{P_1} \oplus \frac{S}{P_2})$ and by applying Depth Lemma to B:3.1 we obtain

$$\text{depth}(\frac{S}{P_1 \cap P_2}) = n - \mathrm{ht}(P_1 + P_2) + 1.$$

There are two cases:
Case 1: $P_i \not\subset P_j + P_k$ for any different $i,j,k \in \{1,2,3\}$.

Let us consider the following two exact sequences of S-modules:

$$0 \to \frac{S}{I} \to \frac{S}{P_1 \cap P_2} \oplus \frac{S}{P_3} \to \frac{S}{P_3 + (P_1 \cap P_2)} \to 0, \qquad \text{(B:3.2)}$$

$$0 \to \frac{S}{(P_1 + P_3) \cap (P_2 + P_3)} \to \frac{S}{(P_1 + P_3)} \oplus \frac{S}{(P_2 + P_3)} \to \frac{S}{P_1 + P_2 + P_3} \to 0. \quad \text{(B:3.3)}$$

By the hypothesis of this case, we have $\text{depth}(\frac{S}{P_1+P_3}) > 0$ and $\text{depth}(\frac{S}{P_2+P_3}) > 0$. Applying Depth Lemma to (B:3.3) we obtain $\text{depth}\left(\frac{S}{(P_1 + P_3) \cap (P_2 + P_3)}\right) = 1$ because $P_1 + P_2 + P_3$ is the maximal ideal. But $(P_1 + P_3) \cap (P_2 + P_3) = P_3 + (P_1 \cap P_2)$ and therefore $\text{depth}\left(\frac{S}{P_3 + (P_1 \cap P_2)}\right) = 1$. Using again the hypothesis of this case in (B:3.2) we can say that $\text{depth}\left(\frac{S}{P_3}\right) > 1$ and $\text{depth}\left(\frac{S}{P_1 \cap P_2}\right) > 1$. By Depth Lemma applied to

(B:3.2) we have $\mathrm{depth}(S/I) = 2$. Thus $\mathrm{depth}\ (I) = 3$.
Case 2: There exist different $i, j, k \in \{1, 2, 3\}$ such that $P_i \subset P_j + P_k$.

After a possible renumbering of $(P_i)_{1 \le i \le 3}$ we may suppose that $P_1 \subset P_2 + P_3$. Note that $P_1 = P_1 \cap P_2 + P_1 \cap P_3$. Consider the next exact sequence of S-modules:

$$0 \to \frac{S}{I} \to \frac{S}{(P_1 \cap P_2)} \oplus \frac{S}{P_1 \cap P_3} \to \frac{S}{P_1} \to 0. \tag{B:3.4}$$

Note that $\mathrm{depth}\left(\frac{S}{P_1 \cap P_2}\right)$ and $\mathrm{depth}\left(\frac{S}{P_1 \cap P_3}\right)$ are smaller or equal than $\dim\left(\frac{S}{P_1}\right)$ (see [BH]). We prove that

$$\mathrm{depth}\left(\frac{S}{I}\right) = \min\left\{\mathrm{depth}\left(\frac{S}{P_1 \cap P_2}\right), \mathrm{depth}\left(\frac{S}{P_1 \cap P_3}\right)\right\}.$$

If

$$\begin{array}{ccc} \dim\left(\frac{S}{P_1}\right) & > & \min\left\{\mathrm{depth}\left(\frac{S}{P_1 \cap P_2}\right), \mathrm{depth}\left(\frac{S}{P_1 \cap P_3}\right)\right\} \\ \| & & \| \\ n - \mathrm{ht}(P_1) & & n + 1 - \max\left\{\mathrm{ht}(P_1 + P_2), \mathrm{ht}(P_1 + P_3)\right\} \end{array}$$

then we are done by Depth Lemma applied to (B:3.4).

Otherwise, $n - \mathrm{ht}(P_1) = n + 1 - \max\{\mathrm{ht}(P_1 + P_2), \mathrm{ht}(P_1 + P_3)\}$ and applying again Depth Lemma we obtain that $\mathrm{depth}\left(\frac{S}{I}\right) \ge \min\left\{\mathrm{depth}\left(\frac{S}{P_1 \cap P_2}\right), \mathrm{depth}\left(\frac{S}{P_1 \cap P_3}\right)\right\}$, the inequality being equality since

$$\mathrm{depth}\left(\frac{S}{I}\right) \le \dim\left(\frac{S}{P_1}\right) = n - \mathrm{ht}(P_1) = \min\left\{\mathrm{depth}\left(\frac{S}{P_1 \cap P_2}\right), \mathrm{depth}\left(\frac{S}{P_1 \cap P_3}\right)\right\}.$$

Thus we have proved that

$$\mathrm{depth}(S/I) = n + 1 - \max\{\mathrm{ht}(P_1 + P_2), \mathrm{ht}(P_1 + P_3)\}$$

and so

$$\mathrm{depth}\ (I) = n + 2 - \max\{\mathrm{ht}(P_1 + P_2), \mathrm{ht}(P_1 + P_3)\}.$$

□

The next lemma presents a decomposition of the above I as a direct sum of its linear subspaces. These subspaces are “simpler” monomial ideals, for which we already know “good” Stanley decompositions. Substituting them in the above direct sum we get some ***special Stanley decompositions*** where it is easier to lower bound their Stanley depth.

We may suppose after a possible renumbering of variables that $P_1 = (x_1, \dots, x_r)$. Let us denote the following:

b_2 – the number of variables from $\{x_i|1 \leq i \leq r\}$ for which $x_i \in P_2$,
b_3 – the number of variables from $\{x_i|1 \leq i \leq r\}$ for which $x_i \in P_3$,
b_1 – the number of variables from $\{x_i|1 \leq i \leq r\}$ for which $x_i \in P_2 \cup P_3$,
a_{23} – the number of variables from $\{x_i|1 \leq i \leq r\}$ for which $x_i \in P_2 \setminus P_3$,
a_{32} – the number of variables from $\{x_i|1 \leq i \leq r\}$ for which $x_i \in P_3 \setminus P_2$,
c – the number of variables from $\{x_i|r+1 \leq i \leq n\}$ for which $x_i \in P_2 \cap P_3$,

$$\begin{aligned}
A &= \left\lceil \frac{a_{32}}{2} \right\rceil + \left\lceil \frac{\operatorname{ht}(P_2) - b_2}{2} \right\rceil + n - a_{32} - \operatorname{ht}(P_2),\\
B &= \left\lceil \frac{a_{23}}{2} \right\rceil + \left\lceil \frac{\operatorname{ht}(P_3) - b_3}{2} \right\rceil + n - a_{23} - \operatorname{ht}(P_3),\\
C &= \left\lceil \frac{r - b_1}{2} \right\rceil + \left\lceil \frac{\operatorname{ht}(P_2) - b_2 - c}{2} \right\rceil + \left\lceil \frac{\operatorname{ht}(P_3) - b_3 - c}{2} \right\rceil,\\
S' &= K\big[\{x_i|1 \leq i \leq r, x_i \notin P_3\}\big],\\
S'' &= K\big[\{x_i|1 \leq i \leq r, x_i \notin P_2\}\big],\\
\widetilde{S} &= K\big[\{x_i|1 \leq i \leq r, x_i \notin P_2 + P_3\}, x_{r+1}, \ldots, x_n\big].
\end{aligned}$$

Lemma B:3.2.9.

Let $S = K[x_1, \ldots, x_n]$, P_1, P_2, P_3 be three non-zero monomial prime ideals not included one in the other such that $\sum_{i=1}^{3} P_i = (x_1, \ldots, x_n)$ and $I = P_1 \cap P_2 \cap P_3$. The next sum is a direct sum of linear subspaces of I:

$$I = I_1 \oplus I_2 \oplus I_3 \oplus I_4,$$

where:

$$\begin{aligned}
I_1 &= \big(I \cap K[x_1, \ldots, x_r]\big)S,\\
I_2 &= (P_2 \cap S')S'[x_{r+1}, \ldots, x_n] \cap \big(P_3 \cap S'[x_{r+1}, \ldots, x_n]\big),\\
I_3 &= (P_3 \cap S'')S''[x_{r+1}, \ldots, x_n] \cap \big(P_2 \cap S''[x_{r+1}, \ldots, x_n]\big),\\
I_4 &= I \cap \widetilde{S}.
\end{aligned}$$

Proof.

Note that $I \supseteq I_1 + I_2 + I_3 + I_4$ is obvious because every $I_i \subseteq I$. Conversely, let a be a monomial from I. If $a \notin I_1$, then we have the next *three disjoint cases*:

Case 1. $a \notin \big(P_2 \cap K[x_1, \ldots, x_r]\big)S$ but $a \in \big(P_3 \cap K[x_1, \ldots, x_r]\big)S$.

Let $a = uv$, where $u \in K[x_1, \ldots, x_r]$ and $v \in K[x_{r+1}, \ldots, x_n]$ monomials. From the hypothesis of this case we get that $u \notin \big(P_2 \cap K[x_1, \ldots, x_r]\big)$. But P_2 is a prime ideal, so it follows that $v \in P_2$, which leads us to $a \in I_3$.

Case 2. $a \in \big(P_2 \cap K[x_1, \ldots, x_r]\big)S$ but $a \notin \big(P_3 \cap K[x_1, \ldots, x_r]\big)S$.

This case is similar with *Case 1*.

Case 3. $a \notin \big(P_2 \cap K[x_1, \ldots, x_r]\big)S$ and $a \notin \big(P_3 \cap K[x_1, \ldots, x_r]\big)S$.

Let $a = uv$, where $u \in K[x_1, \ldots, x_r]$ and $v \in K[x_{r+1}, \ldots, x_n]$ monomials. From the hypothesis of this case we get that $u \notin P_2 \cap K[x_1, \ldots, x_r]$ and $u \notin P_3 \cap K[x_1, \ldots, x_r]$. Thus $v \in P_2 \cap P_3 \cap K[x_{r+1}, \ldots, x_n]$ because P_2 and P_3 are prime ideals. Hence $a \in I_4$ since $u \in P_1$.

Because the cases are disjoint we have obtained that the sum $I = I_1 + I_2 + I_3 + I_4$ is direct. □

PROPOSITION B:3.2.10.

Let P_1, P_2, P_3 be three non-zero prime monomial ideals of S such that there exists no inclusion between any two of them, $\sum_{i=1}^{3} P_i = (x_1, \ldots, x_n)$ and set $I = P_1 \cap P_2 \cap P_3$. With the above notations we have:

$$\operatorname{sdepth}\,(I) \geq \begin{cases} \min\{A, B, C\}, & \text{if } P_i \not\subset P_j + P_k \text{ for any different } i, j, k \in \{1, 2, 3\} \\ \min\{A, B\}, & \text{if } P_1 \subset P_2 + P_3. \end{cases}$$

The proof follows from Lemma B:3.2.9, but first we explain the idea in the following example:

EXAMPLE B:3.2.11. Let $S = K[x_1, x_2, x_3, x_4]$, $P_1 = (x_1, x_2)$, $P_2 = (x_2, x_3, x_4)$ and $P_3 = (x_1, x_3)$. Then $I = P_1 \cap P_2 \cap P_3 = (x_1x_2, x_1x_3, x_1x_4, x_2x_3)$ and the following Stanley decomposition of I is given by Lemma B:3.2.9:

$$I = (x_1x_2)\ K[x_1, x_2, x_3, x_4] \oplus (x_2x_3)\ K[x_2, x_3, x_4] \oplus (x_1x_3, x_1x_4)\ K[x_1, x_3, x_4].$$

Note that the last term of Lemma B:3.2.9 does not appear in this example since $P_1 \subset P_2 + P_3$. We see that the first and the third term in the sum are principal ideals. Therefore $\operatorname{sdepth}\big((x_1x_2)K[x_1, x_2, x_3, x_4]\big) = 4$ and $\operatorname{sdepth}\big((x_2x_3)K[x_2, x_3, x_4]\big) = 3$. As for the third term we use Lemma B:3.2.6, so

$$\operatorname{sdepth}\big((x_1) \cap (x_3, x_4)\ K[x_1, x_3, x_4]\big) = \left\lceil \frac{1}{2} \right\rceil + \left\lceil \frac{2}{2} \right\rceil = 2.$$

Thus $\operatorname{sdepth}\ (I) \geq \min\{4, 2, 3\} = 2$. The same statement follows from Proposition B:3.2.10 because in this case $b_2 = 1, b_3 = 1, b_1 = 2, a_{23} = 1, a_{32} = 1, c = 1, A = 2$ and $B = 3$. Therefore $\operatorname{sdepth}(I) \geq \min\{A, B\} = 2$. Note that $\operatorname{depth}\ (I) = 2$ by Proposition B:3.2.8. Therefore Stanley's Conjecture holds for this example.

Proof of Proposition B:3.2.10.

From Lemma B:3.2.9 we have the direct sum of spaces $I = I_1 \oplus I_2 \oplus I_3 \oplus I_4$ where:

$$\begin{array}{rcc} I_1 & = & \big(I \cap K[x_1, \dots, x_r]\big)S, \\ I_2 & = & (P_2 \cap S')S'[x_{r+1}, \dots, x_n] \cap \big(P_3 \cap S'[x_{r+1}, \dots, x_n]\big), \\ I_3 & = & (P_3 \cap S'')S''[x_{r+1}, \dots, x_n] \cap \big(P_2 \cap S''[x_{r+1}, \dots, x_n]\big), \\ I_4 & = & I \cap \widetilde{S}. \end{array}$$

Then $\operatorname{sdepth}\,(I) \geq \min_{1 \leq i \leq 4} \operatorname{sdepth}(I_i)$. Note that $\operatorname{sdepth}_S\,(I_1)$ is greater or equal than the number of free variables which is $n - r = \dim(S/P_1) \geq \operatorname{sdepth}\,(I)$ by [Artin1] (so I_1 can be always omitted for sdepth computation). By Lemma B:3.2.2 we have

$$\begin{aligned} \operatorname{sdepth}_{S'[x_{r+1},\dots,x_n]}\,(I_2) \; & \geq \operatorname{sdepth}_{S'}(P_2 \cap S') + \operatorname{sdepth}_{K[x_{r+1},\dots,x_n]}\big(P_3 \cap K[x_{r+1}, \dots, x_n]\big) = \\ & = \left(\left\lceil \tfrac{a_{23}}{2} \right\rceil + r - b_3 - a_{23}\right) + \left(\left\lceil \tfrac{\operatorname{ht}(P_3)-b_3}{2} \right\rceil + (n-r) - (\operatorname{ht}(P_3) - b_3)\right) \\ & = \left\lceil \frac{a_{23}}{2} \right\rceil + \left\lceil \frac{\operatorname{ht}(P_3) - b_3}{2} \right\rceil + n - a_{23} - \operatorname{ht}(P_3) = \\ & = B, \end{aligned}$$

by also applying [Sh](see [Ci]). Similarly we get

$$\operatorname{sdepth}_{S''[x_{r+1},\dots,x_n]}\,(I_3) \geq \left\lceil \frac{a_{32}}{2} \right\rceil + \left\lceil \frac{\operatorname{ht}(P_2) - b_2}{2} \right\rceil + n - a_{32} - \operatorname{ht}(P_2) = A.$$

Finally,

$$\operatorname{sdepth}_{\widetilde{S}}\,(I_4) \geq \left\lceil \frac{r - b_1}{2} \right\rceil + \operatorname{sdepth}_{K[x_{r+1},\dots,x_n]}\big(P_2 \cap P_3 \cap K[x_{r+1}, \dots, x_n]\big).$$

There are two cases:

Case 1. If $P_i \not\subset P_j + P_k$ for any different $i, j, k \in \{1, 2, 3\}$

Note that $P_2 \cap K[x_{r+1}, \dots, x_n] \not\subset P_3 \cap K[x_{r+1}, \dots, x_n]$ (otherwise it would result that $P_2 \subset P_1 + P_3$, *contradicting* the hypothesis of this case). In the same idea $P_3 \cap K[x_{r+1}, \dots, x_n] \not\subset P_2 \cap K[x_{r+1}, \dots, x_n]$. By applying Lemma B:3.2.6 for $r = \operatorname{ht}(P_2) - b_2 - c$ and $n - t = \operatorname{ht}(P_3) - b_3 - c$ we get

$$\operatorname{sdepth}_{K[x_{r+1},\dots,x_n]}\big(P_2 \cap P_3 \cap K[x_{r+1}, \dots, x_n]\big) \geq \left\lceil \frac{\operatorname{ht}(P_2) - b_2 - c}{2} \right\rceil + \left\lceil \frac{\operatorname{ht}(P_3) - b_3 - c}{2} \right\rceil.$$

Note that there are no free variables above. Therefore

$$\operatorname{sdepth}_{\widetilde{S}}\,(I_4) \geq \left\lceil \frac{r - b_1}{2} \right\rceil + \left\lceil \frac{\operatorname{ht}(P_2) - b_2 - c}{2} \right\rceil + \left\lceil \frac{\operatorname{ht}(P_3) - b_3 - c}{2} \right\rceil = C.$$

Consequently, it follows

$$\operatorname{sdepth}\,(I) \geq \min\{A, B, C\}.$$

Case 2. If $P_1 \subset P_2 + P_3$

Note that in this case $\widetilde{S} = K[x_{r+1}, \dots, x_n]$ and $P_1 \cap \widetilde{S} = 0$. Thus I_4 does not appear in the Stanley decomposition of I given by Lemma B:3.2.9. Hence $\operatorname{sdepth}\,(I) \geq \min\{A, B\}$.

If one $I_i = 0$, we consider in both cases that its corresponding integer from $\{A, B, C\}$ will not appear in the sdepth formula. □

Remark B:3.2.12. *In the notations and hypotheses of Proposition B:3.2.10, let $\widehat{S} = S[x_{n+1}, \dots, x_t]$ for some $t > n$. Then*

$$\operatorname{sdepth}_{\widehat{S}}\,(I\widehat{S}) \geq \begin{cases} \min\{A, B, C\} + (t-n), & \text{if } P_i \not\subset P_j + P_k \text{ for any different } i, j, k \\ \min\{A, B\} + (t-n), & \text{if } P_1 \subset P_2 + P_3, \end{cases}$$

by the Lemma B:3.2.7 and the Proposition B:3.2.10.

Theorem B:3.2.13.

Let P_1, P_2 and P_3 be three non-zero prime monomial ideals of S not included one in the other and set $I = P_1 \cap P_2 \cap P_3$. Then,

$$\operatorname{sdepth}\,(I) \geq \operatorname{depth}\,(I).$$

Proof.

By [HVZ] it is enough to suppose the case when $\sum_{i=1}^{3} P_i = (x_1, \dots, x_n)$. As in Proposition B:3.2.8 and Proposition B:3.2.10 there are two cases:

Case 1. If $P_i \not\subset P_j + P_k$ for any different $i, j, k \in \{1, 2, 3\}$.

By Propositions B:3.2.8 and B:3.2.10 we have $\operatorname{depth}(I) = 3$ and $\operatorname{sdepth}\,(I) \geq \min\{A, B, C\}$. We have to prove that $A \geq 3$, $B \geq 3$ and $C \geq 3$.

By hypothesis of *Case 1* we have $n \geq 3$ and $A \geq \lceil \frac{1}{2} \rceil + \lceil \frac{1}{2} \rceil + 1 = 3$. Similarly we get $B \geq 3$. Also it follows $C \geq \lceil \frac{1}{2} \rceil + \lceil \frac{1}{2} \rceil + \lceil \frac{1}{2} \rceil = 3$. Therefore, $\operatorname{sdepth}\,(I) \geq \operatorname{depth}(I)$.

Case 2. There exist different $1 \leq i \leq j \leq k \leq 3$ such that $P_i \subset P_j + P_k$.

After a possible renumbering of $(P_i)_{1 \leq i \leq 3}$ we may suppose that $P_1 \subset P_2 + P_3$. In this case $\operatorname{depth}(I) = n + 2 - \max\{\operatorname{ht}(P_1 + P_2), \operatorname{ht}(P_1 + P_3)\}$ and $\operatorname{sdepth}\,(I) \geq \min\{A, B\}$ by Propositions B:3.2.8 and B:3.2.10. Since $P_1 \subset P_2 + P_3$ we have

$$a_{32} + \operatorname{ht}(P_2) = \operatorname{ht}(P_1 + P_2).$$

As the P_i's are not included one in the other we get

$$\left\lceil \frac{a_{32}}{2} \right\rceil + \left\lceil \frac{\operatorname{ht}(P_2) - b_2}{2} \right\rceil + n - (a_{32} + \operatorname{ht}(P_2)) \geq 1 + 1 + n - \operatorname{ht}(P_1 + P_2),$$

thus $A \geq \operatorname{depth}(I)$. Similarly it will result that $B \geq \operatorname{depth}(I)$.

In conclusion we proved that

$$\operatorname{sdepth}\,(I) \geq \operatorname{depth}\,(I).$$

□

Next we express the integers A, B, C only in terms of heights of P_i, thus independently of the numbering of the variables.

Proposition B:3.2.14.

With the notations above, the following holds:

$$\begin{aligned}A &= \left\lceil\frac{3n-\mathrm{ht}(P_1+P_2)-\mathrm{ht}(P_2+P_3)-\mathrm{ht}(P_2)}{2}\right\rceil+\left\lceil\frac{\mathrm{ht}(P_1+P_2)-\mathrm{ht}(P_1)}{2}\right\rceil,\\ B &= \left\lceil\frac{3n-\mathrm{ht}(P_1+P_3)-\mathrm{ht}(P_2+P_3)-\mathrm{ht}(P_3)}{2}\right\rceil+\left\lceil\frac{\mathrm{ht}(P_1+P_3)-\mathrm{ht}(P_1)}{2}\right\rceil,\\ C &= \left\lceil\frac{n-\mathrm{ht}(P_2+P_3)}{2}\right\rceil+\left\lceil\frac{n-\mathrm{ht}(P_1+P_3)}{2}\right\rceil+\left\lceil\frac{n-\mathrm{ht}(P_1+P_2)}{2}\right\rceil.\end{aligned}$$

Proof.

By definition we have

$$\begin{aligned} r &= \mathrm{ht}(P_1),\\ b_2 &= \mathrm{ht}(P_1)+\mathrm{ht}(P_2)-\mathrm{ht}(P_1+P_2),\\ b_3 &= \mathrm{ht}(P_1)+\mathrm{ht}(P_3)-\mathrm{ht}(P_1+P_3),\\ b_1 &= \mathrm{ht}(P_1)+\mathrm{ht}(P_2+P_3)-n,\\ a_{23} &= \mathrm{ht}(P_1+P_3)+\mathrm{ht}(P_2+P_3)-\mathrm{ht}(P_3)-n,\\ a_{32} &= \mathrm{ht}(P_1+P_2)+\mathrm{ht}(P_2+P_3)-\mathrm{ht}(P_2)-n,\\ c &= \mathrm{ht}(P_1+P_2)+\mathrm{ht}(P_1+P_3)-\mathrm{ht}(P_1)-n,\end{aligned}$$

and it is enough to replace them into the definition of A, B and C. □

B:3.3 The Stanley Conjecture for factors of monomial ideals

The results of this section are published in [APDP1].

Let $I \supsetneq J$ be two squarefree monomial ideals of S. We assume that I is generated by squarefree monomials $f_1, \ldots, f_r$ of degrees d for some $d \in \mathbb{N}$ and a set of squarefree monomials E of degree $\geq d+1$. We may suppose that either $J = 0$, or is generated by some squarefree monomials of degrees $\geq d+1$. B (resp. C) denotes the set of the squarefree monomials of degrees $d+1$ (resp. $d+2$) of $I \setminus J$.

Lemma B:3.3.1.

Let $J \subset I$ be square free monomial ideals and $j \in [n]$ be such that $(J : x_j) \neq (I : x_j)$. Then $\operatorname{depth}_S \dfrac{(I : x_j)}{(J : x_j)} \geq \operatorname{depth}_S \dfrac{I}{J}$.

Proof.

We have

$$\operatorname{pd}_S \frac{I}{J} \geq \operatorname{pd}_{S_{x_j}} \left(\frac{I}{J}\right) \otimes S_{x_j} = \operatorname{pd}_{S_{x_j}} \left(\frac{(I : x_j)}{(J : x_j)}\right) \otimes S_{x_j} = \operatorname{pd}_S \left(\frac{(I : x_j)}{(J : x_j)}\right)$$

the last equality holds since x_j does not appear among the generators of $(I : x_j)$ and $(J : x_j)$. Now it is enough to apply the Auslander-Buchsbaum Theorem. □

Lemma B:3.3.2.

Let $t \in [n]$. Suppose that $I \neq J + I \cap (x_t)$ and $\operatorname{depth}_S \dfrac{I}{J + I \cap (x_t)} = d$. If $\operatorname{depth}_S I/J \geq d + 1$ then $\operatorname{depth}_S I/J = d + 1$.

Proof.

In the following exact sequence

$$0 \longrightarrow \frac{I : x_t}{J : x_t} \xrightarrow{x_t} \frac{I}{J} \longrightarrow \frac{I}{J + I \cap (x_t)} \longrightarrow 0$$

the first term has depth $d{+}1$ by the Depth Lemma. We are done by applying the previous lemma. □

Let w_{ij} be the least common multiple of f_i and f_j and set W to be the set of all $w_{ij} \in B$.

Lemma B:3.3.3.

If $d = 1$ and $B \subset E \cup W$ then $\operatorname{depth}_S I/J = 1$.

Proof.

First suppose that $E = \emptyset$ and assume that $I = (x_1, \ldots, x_r)$. Set $S' = K[x_1, \ldots, x_r]$, $I' = I \cap S'$, $J' = J \cap S'$. By hypothesis $B \subset S'$ and it follows that $(x_{r+1}, \ldots, x_n)I \subset J$, leading to the equality $\operatorname{depth}_S I = \operatorname{depth}_{S'} I' = 1$. But $\operatorname{depth}_S J \geq 2$, if $J \neq 0$, and so $\operatorname{depth}_S I/J = 1$ by the Depth Lemma.

Now, suppose that $E \neq \emptyset$. In the following exact sequence

$$0 \longrightarrow \frac{(x_1, \ldots, x_r)}{J \cap (x_1, \ldots, x_r)} \longrightarrow \frac{I}{J} \longrightarrow \frac{I}{(J, x_1, \ldots, x_r)} \longrightarrow 0$$

the first term has depth 1 as above and the last term has depth $\geq d + 1$ since it is generated by squarefree monomials of degrees ≥ 2 from E. Again the Depth Lemma gives $\operatorname{depth}_S I/J = 1$. □

Lemma B:3.3.4.

Suppose that $I \subset S$ is generated by some squarefree monomials $f_1, ...f_r$ of degree d. Assume that for all $b \in B$ all divisors of b of degree d are among $\{f_1, \ldots, f_r\}$. Then $\operatorname{depth}_S I/J = d$.

Proof.

We prove by induction on $d \geq 1$. If $d = 1$ then apply the previous lemma. Assume that $d > 1$. We may suppose that $n \in \operatorname{supp} f_1$. $(I : x_n)$ is an extension of a squarefree monomial ideal I' of $S' = K[x_1, \dots, x_{n-1}]$ which is generated in degree $\geq d-1$. Similarly $(J : x_n)$ is generated by a squarefree monomial ideal J' of S'. Note that the generators of I' of degree $d-1$ have the form $f'_i = f_i/x_n$ for $f_i \in (x_n)$, and the squarefree monomials B' of degrees d from $I' \setminus J'$ have the form $b' = b/x_n$ for some $b \in (B \cap (x_n))$. Certainly we must consider also the case when $f_j \notin (x_n)$. If $x_n f_j \in J$ then $f_j \in (J : x_n)$ is not in B'. Otherwise, $f_j = (x_n f_j)/x_n \in B'$. Note that all divisors of degree $d-1$ of each $b' \in B'$ are among f'_i. By induction hypothesis we have $\operatorname{depth}_{S'} I'/J' = d-1$ and so $\operatorname{depth}_S (I : x_n)/(J : x_n) = d$. Now it is enough to apply Lemma B:3.3.1. □

An obstruction to improve Lemma B:3.3.3 and the above lemma is given by the following example.

Example B:3.3.5. Let $n = 5$, $d = 2$, $r = 5$, $I = (x_1x_2, x_1x_3, x_2x_3, x_1x_4, x_3x_5)$, $J = (x_1x_2x_5, x_1x_4x_5, x_2x_3x_4, x_3x_4x_5)$ and $B = \{x_1x_2x_3, x_1x_2x_4, x_1x_3x_4, x_1x_3x_5, x_2x_3x_5\}$. We have $\operatorname{depth}_S I/J = 3$ because $\operatorname{depth}_S S/J = 3$, $\operatorname{depth}_S S/I = 2$ and with the help of Depth Lemma. Note that each $b \in B$ is the least common multiple of two generators of I, but for example $b = x_1x_2x_4$ has $x_2x_4 \notin I$ as a divisor of degree 2.

Let C_3 be the set of all $c \in C$ having all divisors from $B \setminus E$ in W. In particular each monomial of C_3 is the least common multiple of three of f_i. The converse is not true as shows the following example.

Example B:3.3.6. Let $n = 4$, $d = 2$, $r = 3$, $f_1 = x_1x_2$, $f_2 = x_2x_3$, $f_3 = x_3x_4$, $I = (f_1, f_2, f_3)$ and $J = 0$. Then $c = x_1x_2x_3x_4$ is the least common multiple of f_1, f_2, f_3 but has a divisor $b = x_1x_2x_4 \in B$ which is not the least common multiple of two f_i.

The next theorem is the key result of this section; the proof of it will be presented later in this chapter. The main reason that this proof works for $r \leq 3$ but not for $r = 4$ is that in the first case $|C_3| \leq 1$ but in the second one we may have $|C_3| = 4$, which makes the things harder. However, for $r \geq 5$ will appear a new problem since we may have $B \subset W$ and $s \geq 2r$ (for example when $r = 5$, $d = 2$ we may have $s = 10 = 2r$). We remind that by Theorem B:3.1.4 we had to check Stanley's Conjecture only when $s \geq 2r$.

Theorem B:3.3.7.

Conjecture B:3.1.6 holds for $r \leq 3$, the case $r = 1$ being given in [PZ].

Example B:3.3.8. Let $n = 5$, $f_1 = x_1x_2$, $f_2 = x_1x_3$, $f_3 = x_1x_4$, $a = x_2x_3x_5$, $E = \{a\}$, $I = (f_1, f_2, f_3, a)$, $J = (x_4a)$. We have $w_{12} = f_1x_3$, $w_{13} = f_1x_4$, $w_{23} = f_2x_4$. Set $c = w_{12}x_4$, $c_1 = w_{12}x_5$, $c_2 = w_{23}x_5$, $c_3 = w_{13}x_5$. Then $C = \{c, c_1, c_2, c_3\}$ and $B \setminus E = B \cap (\cup_i [f_i, c_i])$.

Thus $s = 7$, $q = 4$, $r = 3$. It is easy to see that $\operatorname{sdepth}_S I/J = 3$. Indeed, note that c_1 is the only $c' \in C$ which is multiple of a. Suppose that there exists a partition P on $P_{I/J}$ with sdepth 4. Then we have necessarily in P the interval $[a, c_1]$. If P contains the interval $[f_1, c]$ then it must contain also the intervals $[f_2, c_2]$ and so $[f_3, c_3]$, but then $w_{13} \in [f_1, c] \cap [f_3, c_3]$, that is the union is not disjoint. If P contains the interval $[f_1, c_3]$ then P contains either $[f_3, c]$, $[f_2, c_2]$, or $[f_2, c]$, $[f_3, c_2]$, in both cases the intersection of these two intervals contains w_{23}, which is false. By Theorem B:3.3.7 we get $\operatorname{depth}_S I/J \leq 3$, this inequality being in fact an equality.

B:3.4 A special case of $r = 4$

THEOREM B:3.4.1.
Suppose that $I \subset S$ is minimally generated by some squarefree monomials $\{f_1, \dots, f_r\}$ of degrees d such that there exists $c \in C$ such that $\operatorname{supp} c \not\subset \bigcup_{i=1}^{r} \operatorname{supp} f_i$. If Conjecture B:3.1.6 holds for $r' < r$ and $\operatorname{sdepth}_S I/J = d + 1$, then $\operatorname{depth}_S I/J \leq d + 1$.

Proof.
By [PZ, Lemma 2.1] we may assume that $C \subset (W)$. By hypothesis, choose $t \in \operatorname{supp} c$ such that $t \notin \cup_{i \in [r]} \operatorname{supp} f_i$. We may suppose that $B \cap (x_t) = \{x_t f_1, \dots x_t f_e\}$ for some $e \leq r$. Set $I_t = I \cap (x_t)$, $J_t = J \cap (x_t)$ and $U_t = I_t/J_t$. Then B_t generates I_t.

First assume that $\operatorname{sdepth}_S U_t \leq d + 1$. It follows that $\operatorname{depth}_S U_t \leq d + 1$ by [DP2, Theorem 4.3]. But $U_t \cong (I : x_t)/(J : x_t)$ and so $\operatorname{depth}_S U_t \geq \operatorname{depth}_S I/J$ by Lemma B:3.3.1, which is enough.

Now assume that U_t has sdepth $\geq d+1$. Let P_{U_t} be a partition on U_t with sdepth $d+2$ and let $[b_i, c_i]$ be the disjoint intervals starting with $b_i = x_t f_i$, $i \in [e]$. We have $c_i = x_t w_{ik_i}$ for some $1 \leq k_i \leq r$, $k_i \neq i$ because $C \subset (W)$. Note that $x_t f_{k_i} \in B$ and so $k_i \leq e$. We consider the intervals $[f_i, c_i]$. These intervals contain $x_t f_i$ and w_{ik_i}. If $w_{ik_i} = w_{jk_j}$ for $i \neq j$ then we get $c_i = c_j$ which is false. Thus these intervals are disjoint.

Let I_e be the ideal generated by f_j for $e < j \leq r$ and $B \setminus (\cup_{i=1}^{e} [f_i, c_i])$. Set $J_e = I_e \cap J$ and $U_e = I_e/J_e$. Note that $c_i \notin I_e$ for any $i \in [e]$. In the following exact sequence

$$0 \to I_e/J_e \to I/J \to I/J + I_e \to 0$$

the last term has a partition of sdepth $d + 2$ given by the intervals $[f_i, c_i]$ for $1 \leq i \leq e$. It follows that $I_e \neq J_e$ because $\operatorname{sdepth}_S I/J = d + 1$. Then $\operatorname{sdepth}_S I_e/J_e \leq d + 1$ using [Ra, Lemma 2.2] and so $\operatorname{depth}_S I_e/J_e \leq d+1$ by Theorem B:3.3.7 applied for $k < r$. But the last term of the above sequence has depth $> d$ because x_t does not annihilate f_i for $i \in [e]$. With the Depth Lemma we get $\operatorname{depth}_S I/J \leq d + 1$. □

EXAMPLE B:3.4.2. Let $n = 5$, $r = 4$, $f_1 = x_2x_3$, $f_2 = x_1x_2$, $f_3 = x_3x_4$, $f_4 = x_3x_5$ and $J = (x_1x_2x_4x_5)$. We have $w_{12} = x_1x_2x_3$, $w_{13} = x_2x_3x_4$, $w_{14} = x_2x_3x_5$, $w_{34} =$

$x_3x_4x_5$, $w_{23} = x_1x_2x_3x_4$, $w_{24} = x_1x_2x_3x_5$, $C_2 = \{w_{23}, w_{24}\}$, $C = C_2 \cup \{x_1w_{34}, x_2x_3x_4x_5\}$, $\tilde{I}_1 = \{x_1f_3, x_1f_4, f_2\} \supset J$, $\tilde{I}_4 = \{f_3, x_4f_2\} \supset J$ and $B \cap (x_1) = \{x_1f_3, x_1f_4, x_4f_2, x_5f_2, w_{12}\}$, $B \cap (x_4) = \{w_{13}, w_{14}, w_{34}, x_4f_2, x_1f_3\}$. Note that $\operatorname{sdepth}_S U_1 \leq d + 1 = 3$, $\operatorname{sdepth}_S U_4 \leq 3$ because $|B \cap (x_1)| = |B \cap (x_4)| = 5 > |C \cap (x_1)| + 1 = |C \cap (x_4)| + 1 = 4$. Thus $\operatorname{depth}_S U_1 = \operatorname{depth}_S U_4 \leq 3$ and so we get $\operatorname{depth}_S I/J \leq 3$ using two different t.

Theorem B:3.4.3.

Suppose that $I \subset S$ is minimally generated by four squarefree monomials $\{f_1, \dots, f_4\}$ of degrees d such that there exists $c \in C$ such that $\operatorname{supp} c \not\subset \bigcup_{i=1}^{4} \operatorname{supp} f_i$. If $\operatorname{sdepth}_S I/J = d + 1$ then $\operatorname{depth}_S I/J \leq d + 1$.

Proof.

Apply Theorem B:3.4.1, since Conjecture B:3.1.6 holds for $r < 4$ by Theorem B:3.3.7. □

B:3.5 Proof of Theorem B:3.3.7

Suppose that $E \neq \emptyset$ and $s \leq q + r$. For $b = f_1x_i \in B$ set $I_b = (f_2, \dots, f_r, B \setminus \{b\})$, $J_b = J \cap I_b$. If $\operatorname{sdepth}_S I_b/J_b \geq d + 2$ then let P_b be a partition on I_b/J_b with sdepth $d+2$. We may choose P_b such that each interval starting with a squarefree monomial of degree d, $d+1$ ends with a monomial of C. In P_b we have some intervals $[f_k, f_kx_{i_k}x_{j_k}]$, $1 < k \leq r$ and for all $b' \in [B \setminus \{b, f_2x_{i_2}, f_2x_{j_2}, \dots, f_rx_{i_r}, f_rx_{j_r}\}]$ an interval $[b', c_{b'}]$. We define $h : [[\{f_2, \dots, f_r\} \cup B] \setminus \{b, f_2x_{i_2}, f_2x_{j_2}, \dots, f_rx_{i_r}, f_rx_{j_r}\}] \to C$ by $f_k \to f_kx_{i_k}x_{j_k}$ and $b' \to c_{b'}$. Then h is an injection and $|\operatorname{Im} h| = s - r \leq q$ (if $s = r + q$ then h is a bijection). Let g be the map $\operatorname{Im} h \to [[B \cup \{f_2, \dots, f_r\}] \setminus \{b, f_2x_{i_2}, f_2x_{j_2}, \dots, f_rx_{i_r}, f_rx_{j_r}\}]$ given by $f_kx_{i_k}x_{j_k} \to f_k$ and $h(b') \to b'$. We may suppose that all intervals of P_b starting with a monomial v of degree $\geq d + 2$ have the form $[v, v]$.

Lemma B:3.5.1.

Suppose that the following conditions hold:

1. $r = 2$, $s \leq q + 2$.

2. $$C \subset \big((f_1) \cap (f_2)\big) \cup \big((E) \cap (f_1, f_2)\big) \cup \left(\bigcup_{\substack{a, a' \in E \\ a \neq a'}} (a) \cap (a') \right),$$

3. $\operatorname{sdepth}_S I_b/J_b \geq d + 2$ *for* $b \in (f_1) \setminus (f_2)$.

Then either $\operatorname{sdepth}_S I/J \geq d+2$, or there exists a nonzero ideal $I' \subsetneq I$ generated by a subset of $\{f_1, f_2\} \cup B$ such that $\operatorname{sdepth}_S I'/J' \leq d+1$ for $J' = J \cap I'$ and $\operatorname{depth}_S I/(J, I') \geq d+1$.

Proof.

Since $\operatorname{sdepth}_S I_b/J_b \geq d+2$ we consider h, g as above for a partition P_b with sdepth $d+2$ of I_b/J_b. We have an interval $[f_2, c_2']$ in P_b. Suppose that $B \cap [f_2, c_2'] = \{u, u'\}$. By Theorem B:3.1.4 $s \geq 4$ and so there exists $a_1 \in B \setminus \{b, u, u'\}$ such that $c_1 = h(a_1)$. By recurrence choose if possible a_{p+1} to be a divisor from B of c_p which is not in $\{b, u, u', a_1, \ldots, a_p\}$ and set $c_p = h(a_p)$, $p \geq 1$. This construction ends at step $p = e$ if all divisors from B of c_{e-1} are in $\{b, u, u', a_1, \ldots, a_{e-1}\}$. Suppose that a_v, $1 \leq v < e-1$ divides c_{e-1}. Then changing in P_b the intervals $[a_k, c_k], v \leq k \leq e-1$ with the intervals $[a_v, c_{e-1}], [a_{k+1}, c_k], v \leq k \leq e-2$ we see that the new c_{e-1} is the old c_{e-2}. If this new c_{e-1} has a divisor a_e which is not in $\{b, u, u', a_1, \ldots, a_{e-1}\}$ then we continue our procedure with $c_e = h(a_e)$ and so on. After several such steps we arrive at step $p = t$ when one of the following situations holds:

1. all divisors from B of any c_p, $1 \leq p < t$ are among $\{a_1, \ldots, a_{t-1}\}$,
2. $c_{t-1} \in (u, u')$, all its divisors from B are in $\{a_1, \ldots, a_{t-1}, u, u'\}$, all divisors from B of any c_p, $1 \leq p < t-1$ are among $\{a_1, \ldots, a_{t-2}\}$, but no c_p, $1 \leq p < t$ is in (b),
3. $c_{t-1} \in (b)$, but no c_p, $1 \leq p < t-1$ is in (b).

In the first case, set $T = \{a_1, \ldots, a_{t-1}\}$, $G = B \setminus T$ and $I_1' = (f_1, G)$, $I_2' = (f_2, G)$, $I_{12}' = (f_1, f_2, G)$, $I'' = (G)$, $J_1' = I_1' \cap J$, $J_2' = I_2' \cap J$, $J_{12}' = I_{12}' \cap J$, $J'' = I'' \cap J$. Note that $I'' \neq 0$ because $b \in I''$. Consider the following exact sequence

$$0 \to I_{12}'/J_{12}' \to I/J \to I/(J, I_{12}') \to 0.$$

If $T \cap (f_1, f_2) = \emptyset$ then the last term has depth $\geq d+1$ and sdepth $\geq d+2$ using the restriction of P_b to (T) since $h(b) \notin I_{12}'$, for all $b \in T$. When the first term has sdepth $\geq d+2$ then by [Ra, Lemma 2.2] the middle term has sdepth $\geq d+2$ which is enough.

If $T \cap (f_1) = \emptyset$, but let us say $a_v \in (f_2)$ for some $1 \leq v < t$, then in the following exact sequence

$$0 \to I_1'/J_1' \to I/J \to I/(J, I_1') \to 0$$

the last term has sdepth $\geq d+2$ since $h(b) \notin I_1'$ and we may substitute the interval $[a_v, c_v]$ from the restriction of P_b by $[f_2, c_v]$, the second monomial from $[f_2, c_v] \cap B$ being also in T. As above we get either $\operatorname{sdepth}_S I/J \geq d+2$, or $\operatorname{sdepth}_S I_1'/J_1' \leq d+1$, $\operatorname{depth}_S I/(J, I_1') \geq d+1$. Similarly, we do when $T \cap (f_2) = \emptyset$ but let us say $a_v \in (f_1)$ for some $1 \leq v < t$.

Now, suppose that $a_{v_1} \in (f_1)$ and $a_{v_2} \in (f_2)$ for some $1 \leq v_1 < t$, $1 \leq v_2 < t$. We may assume that one from c_{v_1}, c_{v_2} is not in (w_{12}), let us say $c_{v_1} \notin (w_{12})$. Indeed, if $w_{12} \notin B$ then necessary $v_1 \neq v_2$, otherwise $a_{v_1} = a_{v_2} = w_{12} \in B$. Thus $c_{v_1} \neq c_{v_2}$ and so one of $c_{v_i} \notin (w_{12})$. If $w_{12} \in B$ and $c_{v_1} = x_j w_{12}$ for some $j \notin \operatorname{supp} w_{12}$, then let us say $a_{v_1} = x_j f_1$ and we have $x_j f_2 = a_v$ for some $1 \leq v < t$ by our hypothesis 1). If $c_v \in (w_{12})$ then $c_v = c_{v_1}$, which is false. If $a_{v_1} = w_{12}$ then again we have $x_j f_2 = a_v$, for some $1 \leq v < t$ and $c_v \notin (w_{12})$ because otherwise $c_v = c_{v_1}$. Take $v_2 = v$.

In the following exact sequence

$$0 \to I''/J'' \to I/J \to I/(J, I'') \to 0$$

the last term has sdepth $\geq d+2$ the last term has sdepth $\geq d+2$ since we may replace the intervals $[a_{v_1}, c_{v_1}]$, $[a_{v_2}, c_{v_2}]$ of the restriction of P_b to (T) with the disjoint intervals $[f_1, c_{v_1}]$, $[f_2, c_{v_2}]$. Also the last term has depth $\geq d+2$ because in the exact sequence

$$0 \to (f_2)/(J, I'') \cap (f_2) \to I/(J, I'') \to I/(J, I'', f_2) \to 0$$

the ends terms have depth $\geq d+1$ since $c_{v_1} \notin (f_2)$, otherwise $c_{v_1} \in (w_{12})$, which is false. As above we get either $\operatorname{sdepth}_S I/J \geq d+2$, or $\operatorname{sdepth}_S I''/J'' \leq d+1$, $\operatorname{depth}_S I/(J, I'') \geq d+1$.

In the second case, note that $c'_2 \neq c_{t-1}$ because h is injective. Then suppose that $u|c_{t-1}$, but u' is not a divisor of any c_p, $1 \leq p < t$. If $a_{t-1} \in (f_2)$ then change in P_b the interval $[a_{t-1}, c_{t-1}]$ by $[f_2, c_{t-1}]$. If a_v, $1 \leq v < t$ is the second divisor of c_{t-1} from $B \cap (f_2)$, change in P_b the intervals $[a_p, c_p]$, $v \leq p < t$ by $[a_v, c_{t-1}]$, $[a_{p+1}, c_p]$, $v \leq p < t-1$ we see that now a_v, u divide the new c_v. Changing in P_b the intervals $[a_v, c_v]$, $[f_2, c'_2]$ with $[f_2, c_v]$, $[u', c'_2]$, we may proceed as above using I'_1 and $T' = T \cup \{u\}$.

In the third case, let $c_{t-1} = bx_i$. If $a_{t-1} = f_1 x_i$ then changing in P_b the interval $[a_{t-1}, c_{t-1}]$ by $[f_1, c_{t-1}]$ we get a partition on I/J with sdepth d+2. If $f_1 x_i \in \{a_1, \ldots, a_{t-2}\}$, let us say $f_1 x_i = a_v$, $1 \leq v < t-1$ then we may replace in P_b the intervals $[a_k, c_k]$, $v \leq k \leq t-1$ with the intervals $[a_v, c_{t-1}], [a_{k+1}, c_k], v \leq k \leq t-2$. Now we see that we have in P_b the interval $[a_v, c_v]$ and switching it with the interval $[f_1, c_v]$ we get a partition with sdepth $\geq d+2$ for I/J.

Thus we may assume that $f_1 x_i \notin \{a_1, ..., a_{t-1}\}$. Suppose that $f_1 x_i \notin \{u, u'\}$. Then set $a_t = f_1 x_i$, $c_p = h(a_p), p \geq t$ and choose $a_{p+1} \notin \{b, u, u', a_t, \ldots, a_p\}$ to be a divisor of B of $c_p, p \geq t$. If $a_p = a_v$ for $v < t$, $p > t$ then again change in P_b the intervals $[a_k, c_k]$, $v \leq k \leq p-1$ with the intervals $[a_v, c_{p-1}], [a_{k+1}, c_k], v \leq k \leq p-2$. We have in P_b an interval $[f_1 x_i, c_{t-1}]$ and switching it to $[f_1, c_{t-1}]$ we get a partition with sdepth $\geq d+2$ for I/J. Thus we may suppose that $a_{p+1} \notin \{b, u, u', a_1, \ldots, a_p\}$. If the procedure stops at step $p = t_1$ because all the divisors from B of $\{c_t, \ldots c_{t_1-1}\}$ are among $T_1 = \{a_t, \ldots, a_{t_1-1}\}$ then we do as above with T_1 instead T.

If let us say $f_1 x_i = u' = w_{12}$ and $a_{t-1} \in (f_2)$ then replace in P_b the intervals $[f_2, c'_2]$, $[a_{t-1}, c_{t-1}]$ with $[f_2, c_{t-1}]$,$[u, c'_2]$. Take for a_t the former u which is now not in $[f_2, c'_2]$ and proceed as above. If $f_1 x_i = u' = w_{12}$ but $a_{t-1} \in (E)$ then there exists another divisor $\tilde{a}$ of c_{t-1} from $B \cap (f_2)$ different of w_{12}. If $\tilde{a} \in [f_2, c'_2]$ then we get $h(a_{t-1}) = c_{t-1} = c'_2 = h(f_2)$, which is false. Thus we may set $a_t = \tilde{a}$, $c_t = h(a_t)$ and proceed as above. If the procedure stops at step $p = t_1$ because all the divisors from B of $\{c_t, \ldots c_{t_1-1}\}$ are among $T_1 = \{a_t, \ldots, a_{t_1-1}\}$ then we do as above with T_1 instead T. If $c_{t-1} \in (u, u')$, but no c_p, $t \leq p < t_1$ is in (b), then we proceed as in case 2) taking $T'_1 = T_1 \cup \{u\}$, or $T'_1 = T_1 \cup \{u'\}$.

If $c_{t_1-1} = bx_j$ we continue as above with $a_{t_1} = f_1 x_j$. After several such steps we must arrive in the case $p = t_l$ when all divisors from B of c_{t_l-1} are already among $T_l = \{a_{t_{l-1}}, \ldots, a_{t_l-1}\}$, or $T'_l = T_l \cup \{u\}$, or $T'_l = T_l \cup \{u'\}$. Finally, using T_l as T, or T'_l as T' above we are done. □

LEMMA B:3.5.2.

Suppose that the following conditions hold:

1. $r = 3$, $s \leq q + 3$,

2. $C \subset \left(\bigcup_{\substack{i,j=1 \\ i \neq j}}^{3} (f_i) \cap (f_j) \right) \cup \big((E) \cap (f_1, f_2, f_3)\big) \cup \left(\bigcup_{\substack{a,a' \in E \\ a \neq a'}} (a) \cap (a') \right)$,

3. *There exists* $b \in (f_1) \setminus (f_2, f_3)$ *such that* $\operatorname{sdepth}_S I_b/J_b \geq d + 2$.

Then either $\operatorname{sdepth}_S I/J \geq d+2$*, or there exists a nonzero ideal* $I' \subsetneq I$ *generated by a subset of* $\{f_1, f_2, f_3\} \cup B$ *such that* $\operatorname{sdepth}_S I'/J' \leq d+1$ *for* $J' = J \cap I'$ *and* $\operatorname{depth}_S I/(J, I') \geq d+1$.

Proof.

Consider h, g as above for a partition P_b with sdepth $d + 2$ of I_b/J_b which exists by (3). We have two intervals $[f_2, c_2']$, $[f_3, c_3']$in P_b. Suppose that $B \cap [f_i, c_i'] = \{u_i, u_i'\}$, $1 < i \leq 3$. By Theorem B:3.1.4 $s \geq 6$ and there exists $a_1 \in B \setminus \{b, u_2, u_2', u_3, u_3'\}$ such that $c_1 = h(a_1)$. By recurrence choose if possible a_{p+1} to be a divisor from B of c_p which is not in $\{b, u_2, u_2', u_3, u_3', a_1, \ldots, a_p\}$ and set $c_p = h(a_p)$, $p \geq 1$. This construction ends at step $p = e$ if all divisors from B of c_{e-1} are in $\{b, u_2, u_2', u_3, u_3', a_1, \ldots, a_{e-1}\}$. Suppose that a_v, $1 \leq v < e - 1$ divides c_{e-1}. Then changing in P_b the intervals $[a_p, c_p], v \leq p \leq e - 1$ with the intervals $[a_v, c_{e-1}], [a_{p+1}, c_p], v \leq p \leq e - 2$ we see that the new c_{e-1} is the old c_{e-2}. If this new c_{e-1} has a divisor a_e which is not in $\{b, u_2, u_2', u_3, u_3', a_1, \ldots, a_{e-1}\}$ then we continue our procedure with $c_e = h(a_e)$ and so on. After several such steps we arrive at step $p = t$ when one of the following situations holds:

1. all divisors from B of any c_p, $1 \leq p < t$ are among $\{a_1, \ldots, a_{t-1}\}$,

2. $c_{t-1} \in (u_2, u_2', u_3, u_3')$, all its divisors from B are in $\{a_1, \ldots, a_{t-1}, u_2, u_2', u_3, u_3'\}$, all divisors from B of any c_p, $1 \leq p < t - 1$ are among $\{a_1, \ldots, a_{t-2}\}$, but no c_p, $1 \leq p < t$ is in (b),

3. $c_{t-1} \in (b)$, but no c_p, $1 \leq p < t - 1$ is in (b).

In the first case, set $T = \{a_1, \ldots, a_{t-1}\}$, $G = B \setminus T$ and for $k = (k_1, \ldots, k_m)$, $1 \leq k_1 < \ldots < k_m \leq 3$, $0 \leq m \leq 3$ set $I_k' = (f_{k_1}, \ldots, f_{k_m}, G)$, $J_k' = I_k' \cap J$, and $I_0' = (G)$, $J_0' = I_0' \cap J$ for $m = 0$. Note that $I_0' \neq 0$ because $b \in I_0'$. Consider the following exact sequence

$$0 \to I_k'/J_k' \to I/J \to I/(J, I_k') \to 0.$$

If $T \cap (f_1, f_2, f_3) = \emptyset$ then the last term of the above exact sequence given for $k = (1, 2, 3)$ has depth $\geq d + 1$ and sdepth $\geq d + 2$ using the restriction of P_b to (T) since $h(b) \notin I_k'$, for all $b \in T$. When the first term has sdepth $\geq d + 2$ then by [Ra, Lemma 2.2] the middle term has sdepth $\geq d + 2$ which is enough.

Suppose that $T \cap (f_{k_i}) = \emptyset$ if and only if $i \in [m]$, for some $0 \leq m < 3$ and set $k = (1, \ldots, m)$. We show that in the following exact sequence

$$0 \to I'_k/J'_k \to I/J \to I/(J, I'_k) \to 0$$

the last term has sdepth $\geq d+2$. For all $m < i \leq 3$ we find $c_{v_i} \in (f_i)$ for some $1 \leq v_i < t$. We claim that we may choose v_i such that for all $i \neq j$, $m < i, j \leq 3$ we have $v_i \neq v_j$ and one from c_{v_i}, c_{v_j} is not in (w_{ij}). Indeed, this is trivial for $m = 2$. Suppose that $m < 2$. If $w_{ij} \notin B$ then necessary $v_i \neq v_j$, otherwise $a_{v_i} = a_{v_j} = w_{ij} \in B$. Contradiction! If $w_{ij} \in B$ then suppose that $c_{v_i} = x_z w_{ij}$. Then we see that there exists $1 \leq v_j < t$ such that $a_{v_j} = x_z f_j$. If $c_{v_j} \in (w_{ij})$ then we have $c_{v_j} = x_z w_{ij} = c_{v_i}$, which is false. Thus $c_{v_j} \notin (w_{ij})$.

Therefore in the worst case, that is $m = 0$, we may choose $c_{v_1} \notin (w_{12})$, $c_{v_2} \notin (w_{23})$. If $c_{v_1} \in (w_{13})$ then we may also choose $c_{v_3} \notin (w_{13})$. In the case when $c_{v_1} \notin (w_{13})$, choose any $c_{v_3} \in (f_3)$. We conclude that the possible intervals $[f_i, c_{v_i}]$, $i \in [3]$ are disjoint. Next we change the intervals $[a_{v_i}, c_{v_i}]$, $m < i \leq 3$ from the restriction of P_b to (T) by $[f_i, c_{v_i}]$, the second monomial from $[f_i, c_{v_i}] \cap B$ being also in T. Set $k = (1, 2, 3)$. Also $I/(J, I'_k)$ has depth $\geq d+1$ because in the exact sequence

$$0 \to (f_3)/(f_3) \cap (J, I'_k) \to I/(J, I'_k) \to I/(J, I'_k, f_3) \to 0$$

the last term has has depth $\geq d+2$ since $c_{v_2} \notin (w_{23})$ and if $c_{v_1} \in (f_3)$ then $c_{v_1} \in (w_{13})$ but we may find a $b' \in (B \cap (f_1)) \setminus (f_3)$ and we may proceed as in the proof of Lemma B:3.5.1. As above we get either $\operatorname{sdepth}_S I/J \geq d+2$, or $\operatorname{sdepth}_S I'_1/J'_1 \leq d+1$, $\operatorname{depth}_S I/(J, I'_1) \geq d+1$.

In the second case, note that $c'_2, c'_3 \neq c_{t-1}$ because h is injective. Then suppose that $u_2 | c_{t-1}$, but u'_2 is not a divisor of any c_p, $1 \leq p < t$. Also suppose for a moment that $c_{t-1} \notin (u_3, u'_3)$. If $a_{t-1} \in (f_2)$ then change in P_b the interval $[a_{t-1}, c_{t-1}]$ by $[f_2, c_{t-1}]$. If a_{v_2}, $1 \leq v_2 < t$ is the second divisor of c_{t-1} from $B \cap (f_2)$, change in P_b the intervals $[a_p, c_p]$, $v_2 \leq p < t$ by $[a_{v_2}, c_{t-1}]$, $[a_{p+1}, c_p]$, $v_2 \leq p < t-1$ we see that now a_{v_2}, u_2 divide the new c_{v_2}. Change in P_b the intervals $[a_{v_2}, c_{v_2}]$, $[f_2, c'_2]$ with $[f_2, c_{v_2}]$, $[u'_2, c'_2]$, we may proceed as above using I'_k for $k = (1)$, or $k = (13)$ and $T' = T \cup \{u_2\}$.

If c_{t-1} is a multiple of u_2 and u_3 then it is a multiple of w_{23} and we may suppose $u_2 = u_3 = w_{23}$, the other u'_2, u'_3 cannot divide c_{t-1}. Then c_{t-1} have other two divisors from B one a_{v_2} from (f_2), the other a_{v_3} from (f_3). If $v_2 = t-1$ we change as above $[a_{t-1}, c_{t-1}]$ by $[f_2, c_{t-1}]$. If $v_2 < t-1$ change in P_b the intervals $[a_p, c_p]$, $v_2 \leq p < t$ by $[a_{v_2}, c_{t-1}]$, $[a_{p+1}, c_p]$, $v_2 \leq p < t-1$ and we see that now a_{v_2}, w_{23} divide the new c_{v_2}. Change in P_b the intervals $[a_{v_2}, c_{v_2}]$, $[a_{v_3}, c_{v_3}]$ by $[f_2, c_{v_2}]$, $[f_3, c_{v_3}]$. Note that the second monomial of $B \cap [f_3, c_{v_3}]$ is also in T. As above we may proceed with I_1, and $T' = T \cup \{w_{23}\}$.

In the third case, let $c_{t-1} = bx_i$. If $a_{t-1} = f_1 x_i$ then changing in P_b the interval $[a_{t-1}, c_{t-1}]$ by $[f_1, c_{t-1}]$ we get a partition on I/J with sdepth $d+2$. If $f_1 x_i \in \{a_1, \ldots, a_{t-2}\}$, let us say $f_1 x_i = a_v$, $1 \leq v < t-1$ then we may replace in P_b the intervals $[a_k, c_k], v \leq k \leq t-1$ with the intervals $[a_v, c_{t-1}], [a_{k+1}, c_k], v \leq k \leq t-2$. Now we see that we have in P_b the interval $[a_v, c_v]$ and switching it with the interval $[f_1, c_v]$ we get a partition with sdepth $\geq d+2$ for I/J.

Thus we may assume that $f_1x_i \not\in \{a_1, ..., a_{t-1}\}$. Suppose that $f_1x_i \not\in \{u_2, u_2', u_3, u_3'\}$. Then set $a_t = f_1x_i$, $c_p = h(a_p), p \geq t$ and choose $a_{p+1} \not\in \{b, u_2, u_2', u_3, u_3', a_t, \ldots, a_p\}$ to be a divisor from B of $c_p, p \geq t$. If $a_p = a_v$ for $v < t$, $p > t$ then again change in P_b the intervals $[a_k, c_k], v \leq k \leq p-1$ with the intervals $[a_v, c_{p-1}], [a_{k+1}, c_k], v \leq k \leq p-2$. We have in P_b an interval $[f_1x_i, c_{t-1}]$ and switching it to $[f_1, c_{t-1}]$ we get a partition with sdepth $\geq d+2$ for I/J. Thus we may suppose that $a_{p+1} \not\in \{b, u_2, u_2', u_3, u_3', a_1, \ldots, a_p\}$. If the procedure stops at step $p = t_1$ because all the divisors from B of $\{c_t, \ldots c_{t_1-1}\}$ are among $T_1 = \{a_t, \ldots, a_{t_1-1}\}$ then we do as above with T_1 instead T.

If let us say $f_1x_i = u_2' = w_{12}$ and $a_{t-1} \in (f_2)$ then replace in P_b the intervals $[f_2, c_2']$, $[a_{t-1}, c_{t-1}]$ with $[f_2, c_{t-1}]$,$[u_2, c_2']$. Take for a_t the former u_2 which is now not in $[f_2, c_2']$ and proceed as above. If $f_1x_i = u_2' = w_{12}$ but $a_{t-1} \in (f_3)$ then c_{t-1} is the least common multiple of f_1, f_2, f_3. If $c_{t-1} \in C_3$ then $b = w_{13}$, which is false by (3). Otherwise, c_{t-1} is also the least common multiple of two f_i (see Example B:3.3.5), but in this case it has at least four divisors from B and so we may find one $a_t \not\in \{b, u_2, u_2', u_3, u_3'\}$. If $f_1x_i = u_2' = w_{12}$ but $a_{t-1} \in (E)$ then there exists another divisor $\tilde{a}$ from $B \cap (f_2)$ different of w_{12}. If $\tilde{a} \in [f_2, c_2']$ (i.e. $\tilde{a} = u_2$) then we get $h(a_{t-1}) = c_{t-1} = c_2' = h(f_2)$, which is false. If $\tilde{a} \in (f_3)$ we get as above $b \in (f_3)$, which is again false. Thus h is defined on $\tilde{a}$ and we may set $a_t = \tilde{a}$, $c_t = h(a_t)$ and proceed as above. If the procedure stops at step $p = t_1$ because all the divisors from B of $\{c_t, \ldots c_{t_1-1}\}$ are among $T_1 = \{a_t, \ldots, a_{t_1-1}\}$ then we do as above with T_1 instead T. If $c_{t-1} \in (u_2, u_2')$, but no c_p, $t \leq p < t_1$ is in (b), then we proceed as in case 2) taking $T_1' = T_1 \cup \{u_2\}$, or $T_1' = T_1 \cup \{u_2'\}$.

If $c_{t_1-1} = bx_j$ we continue as above with $a_{t_1} = f_1x_j$. After several such steps we must arrive in the case $p = t_l$ when all divisors from B of c_{t_l-1} are already among $T_l = \{a_{t_{l-1}}, \ldots, a_{t_l-1}\}$, or $T_l' = T_l \cup \{u_2\}$, or $T_l' = T_l \cup \{u_2'\}$, or $T_l' = T_l \cup \{u_3\}$, or $T_l' = T_l \cup \{u_3'\}$. Finally, using T_l as T, or T_l' as T' above we are done. □

Proof of Theorem B:3.3.7.

By Theorem B:3.1.3 we may suppose that $s \leq q+r$. Apply induction on $|E|$. If $E = \emptyset$ we may apply [PZ]. Actually, in this case it is enough to apply as below Lemma [PZ, Lemma 2.1]. Suppose that $|E| > 0$ and $B \cap (f_1, \ldots, f_r) \neq \emptyset$, $r = 2, 3$, otherwise we get $\operatorname{depth}_S I/J \leq d+1$. Moreover, we see that $C \not\subset C_3$ because $|C_3| \leq 1$ and if $q = 1$ then we get $s < 2r$ which gives $\operatorname{depth}_S I/J \leq d+1$ by Theorem B:3.1.4. Let $c \in C \setminus C_3$. Then there exists $b \in B$ dividing c which is not the least common multiple of two f_i. Renumbering x we may suppose that $b \in (f_1) \setminus (f_2)$ if $r = 2$, or $b \in (f_1) \setminus (f_2, f_3)$ if $r = 3$.

We may assume that $C \subset (f_1, \ldots, f_r, B)$. Apply induction on $r \leq 3$. Using the main result of [PZ] and induction hypothesis on $|E|$ and r apply [PZ, Lemma 2.1]. Thus we may suppose that $C \subset \big((f_1) \cap (f_2)\big) \cup \big((E) \cap (f_1, f_2)\big) \cup \big(\cup_{a,a' \in E, a \neq a'} (a) \cap (a')\big)$, if $r = 2$, or $C \subset \big(\cup_{i,j \in [3], i \neq j} (f_i) \cap (f_j)\big) \cup \big((E) \cap (f_1, f_2, f_3)\big) \cup \big(\cup_{a,a' \in E, a \neq a'} (a) \cap (a')\big)$ if $r = 3$.

Set $I_b' = (f_2, f_r, B \setminus \{b\})$, $J_b' = I_b' \cap J$. Clearly $b \not\in I_b'$ and so in the following exact sequence

$$0 \to I_b'/J_b' \to I/J \to I/(J, I_b') \to 0$$

the last term has depth $\geq d+1$. If the first term has sdepth $\leq d+1$ then it has depth $\geq d+2$ by induction hypothesis on r, case $r=1$ being done in [PZ]. Thus we may suppose that $\operatorname{sdepth}_S I'_b/J'_b \geq d+2$ and we may apply Lemmas B:3.5.1 and B:3.5.2. Then we get either $\operatorname{sdepth}_S I/J \geq d+2$ contradicting our assumption, or there exists a nonzero ideal $I' \subsetneq I$ generated by a subset G of B, or by G and a subset of $\{f_1, f_2, f_3\}$ such that $\operatorname{sdepth}_S I'/J' \leq d+1$ for $J' = J \cap I'$ and $\operatorname{depth}_S I/(J, I') \geq d+1$. In the last case we see that $\operatorname{depth}_S I'/J' \leq d+1$ by induction hypothesis on r and so $\operatorname{depth}_S I/J \leq d+1$ by the Depth Lemma applied to the following exact sequence

$$0 \to I'/J' \to I/J \to I/(J, I') \to 0.$$

□

Chapter B:4

Stanley depth of factors of monomial ideals

The results of this chapter are published in [AP3].

Let K be a field and $S = K[x_1, \dots, x_n]$ be the polynomial ring over K in n variables. Let $I \supsetneq J$ be two squarefree monomial ideals of S.

Suppose that I is generated by squarefree monomials of degrees $\geq d$ for some positive integer d. We may assume either that $J = 0$, or J is generated in degrees $\geq d + 1$ after a multigraded isomorphism. We have $\operatorname{depth}_S I \geq d$ by [HVZ, Proposition 3.1]. Depth of I/J is a homological invariant and depends on the characteristic of the field K. The Stanley decompositions of S/J corresponds bijectively to partitions into intervals of the simplicial complex whose Stanley-Reisner ring is S/J. If Stanley's Conjecture holds then the simplicial complexes are partitionable (see [HSY]). Using this idea an equivalent definition of Stanley's depth of I/J was given in [HVZ].

Let $P_{I \setminus J}$ be the poset of all squarefree monomials of $I \setminus J$ with the order given by the divisibility. Let $\mathcal{P}$ be a partition of $P_{I \setminus J}$ in intervals $[u, v] = \{w \in P_{I \setminus J} : u|w, w|v\}$, let us say $P_{I \setminus J} = \cup_i [u_i, v_i]$, the union being disjoint. Define $\operatorname{sdepth} \mathcal{P} = \min_i \deg v_i$. Then $\operatorname{sdepth}_S I/J = \max_{\mathcal{P}} \operatorname{sdepth} \mathcal{P}$, where $\mathcal{P}$ runs in the set of all partitions of $P_{I \setminus J}$ (see [HVZ], [Sw]).

In [IKM], Ichim et. al. studied the sdepth and depth of the factor ${}^{I}/_{J}$ under polarization and reduced the Stanley's Conjecture to the case when the ideals are monomial squarefree. This is one of the best results from the last years concerning Stanley's depth. It is worth to mention that this result is not very useful for computing sdepth since it introduces many new variables that will slow down the algorithm.

Another result of [IKM] that helps in the sdepth computing is the following proposition, which extends [Ci, Lemma 1.1], [IsQu, Lemma 2.1].

Proposition B:4.0.1 $\big($[IKM, Proposition 5.1]$\big)$**.**
Let $k \in \mathbb{N}$ and I'', J'' be the monomial ideals obtained from I, J in the following way:

> *each generator whose degree in x_n is at least k is multiplied by x_n and all other generators are left unchanged.*

Then $\operatorname{sdepth}_S {}^{I}/_{J} = \operatorname{sdepth}_S {}^{I''}/_{J''}$.

Inspired by this proposition we introduced a canonical form of a factor ${}^{I}/_{J}$ of monomial ideals (see Definition B:4.1.3) and we showed easily that sdepth is invariant under taking the canonical form (see Theorem B:4.1.7). This leads us to the idea to study also the depth case (see Theorem B:4.1.11). Theorem B:4.1.12 says that Stanley's Conjecture holds for a factor of monomial ideals if and only if it holds for its canonical form. As a side result, in the depth (respectively sdepth) computation algorithm for I/J, one can first compute the canonical form and use the algorithm on this new much more simpler module.

In Example B:4.1.13 we conclude that the depth and sdepth algorithms are faster when considering the canonical form: using CoCoA[CoCoA], Singular[Singular] and Rinaldo's sdepth computation algorithm [Ri] we see a small decrease in the depth case timing, but in the sdepth case the runtime is massively reduced.

Proposition B:4.0.1 and Corollary B:4.1.10 follow from [OY, Theorem 5.2]. However, our proofs of Lemma B:4.1.9 and Corollary B:4.1.10 are completely different from those appeared in the quoted paper.

B:4.1 The canonical form of a factor of monomial ideals

Let $R = K[x_1, \ldots, x_{n-1}]$ be the polynomial K-algebra over a field K and $S := R[x_n]$. Consider $J \subsetneq I \subset R$ two monomial ideals and denote by $G(I)$, respectively $G(J)$, the minimal (monomial) system of generators of I, respectively J.

Definition B:4.1.1.

The power x_n^r ***enters in a monomial*** u if $x_n^r \mid u$ and $x_n^{r+1} \nmid u$.

We say that I is ***of type*** $(k_1, \ldots, k_s)$ ***with respect to*** x_n if $x_n^{k_i}$ are all the powers of x_n which enter in a monomial of $G(I)$ for $i \in [s]$ and $1 \leq k_1 < \ldots < k_s$.

I is ***in the canonical form with respect to*** x_n if I is of type $(1, \ldots, s)$ for some $s \in \mathbb{N}$.

We simply say that I is ***the canonical form*** if it is in the canonical form with respect to all variables $x_1, \ldots, x_n$.

Remark B:4.1.2. Suppose that I is of type $(k_1, \ldots, k_s)$ with respect to x_n. It is easy to obtain the *canonical form* I' of I *with respect to* x_n: replace $x_n^{k_i}$ by x_n^i whenever $x_n^{k_i}$ enters in a generators of $G(I)$. Applying by recurrence this procedure for other variables we get the *canonical form* of I, that is with respect to all variables. Note that a squarefree monomial ideal is of type (1) with respect to each x_i and it is in the canonical form with respect to x_i, so in this case $I' = I$.

We extend the above definition for a factor of monomial ideals.

Definition B:4.1.3.

Let $J \subsetneq I \subset S$ be two monomial ideals. We say that I/J is ***of type*** $(k_1, \ldots, k_s)$ ***with respect to*** x_n if $x_n^{k_i}$ are all the powers of x_n which enter in a monomial of $G(I) \cup G(J)$ for $i \in [s]$ and $1 \leq k_1 < \ldots < k_s$.

All the terminology presented in Definition B:4.1.1 will extend automatically to the factor case. Thus we may speak about the ***canonical form*** $\overline{I/J}$ of I/J.

REMARK B:4.1.4. In order to compute the canonical form with respect to x_n of the $(k_1, \dots, k_s)-$type factor I/J, one will replace $x_n^{k_i}$ by x_n^i whenever $x_n^{k_i}$ enters a generator of $G(I) \cup G(J)$.

EXAMPLE B:4.1.5. We present some examples where we compute the canonical form of a monomial ideal, respectively a factor of two monomial ideals.

1. Consider $S = \mathbb{Q}[x, y]$ and the monomial ideal $I = (x^4, x^3y^7)$. Then the canonical form of I is $I' = (x^2, xy)$.

2. Consider $S = \mathbb{Q}[x, y, z]$, $I = (x^{10}y^5, x^4yz^7, z^7y^3)$ and $J = (x^{10}y^{20}z^2, x^3y^4z^{13}, x^9y^2z^7)$.

   The canonical form of I/J is $\overline{I/J} = \dfrac{(x^4y^5, x^2yz^2, y^3z^2)}{(x^4y^6z, xy^4z^3, x^3y^2z^2)}$.

The canonical form of a factor of monomial ideals I/J is not usually the factor of the canonical forms of I and J as shows the following example.

EXAMPLE B:4.1.6. Let $S = \mathbb{Q}[x, y]$, $I = (x^4, y^{10}, x^2y^7)$ and $J = (x^{20}, y^{30})$ be two ideals. The canonical form of I is $I' = (x^2, y^2, xy)$ and the canonical form of J is $J' = (x, y)$. Then $J' \not\subset I'$. But the canonical form of the factor I/J is $\overline{I/J} = \dfrac{(x^2, y^2, xy)}{(x^3, y^3)}$.

Using Proposition B:4.0.1, we see that the Stanley depth of a monomial ideal does not change when considering its canonical form.

THEOREM B:4.1.7.
Let I, J be monomial ideals in S and $\overline{I/J}$ the canonical form of I/J. Then

$$\operatorname{sdepth}_S I/J = \operatorname{sdepth}_S \overline{I/J}.$$

The proof follows by inductively applying the following lemma.

LEMMA B:4.1.8.
Suppose that I/J is of type $(k_1, \dots, k_s)$ with respect to x_n and $k_j + 1 < k_{j+1}$ for some $0 \leq j < s$ (we set $k_0 = 0$). Let $G(I')$ (resp. $G(J')$) be the set of monomials obtained from $G(I)$ (resp. $G(J)$) by substituting $x_n^{k_i}$ by $x_n^{k_i - 1}$ for $i > j$ whenever $x_n^{k_i}$ enters in a monomial of $G(I)$ (resp. $G(J)$). Let I' and J' be the ideals generated by $G(I')$ and $G(J')$. Then

$$\operatorname{sdepth}_S I/J = \operatorname{sdepth}_S I'/J'.$$

The proof of Lemma B:4.1.8 follows from the proof of [IKM, Proposition 5.1] (see here Proposition B:4.0.1).

Next we focus on the $\operatorname{depth} {}^{I}/_{J}$ and $\operatorname{depth} \overline{{}^{I}/_{J}}$.

LEMMA B:4.1.9.

Let $I_0 \subset I_1 \subset \ldots \subset I_e \subset R$, $J \subset S$, $U_0 \subset U_1 \subset \ldots \subset U_e \subset R$, $V \subset S$ be some graded ideals of S, respectively R, such that $U_i \subset I_i$ for $0 \leq i \leq e$, $I_e \subset J$, $V \subset J$ and $U_e \subset V$. Consider $T_k = \sum_{i=0}^{e} x_n^i I_i S + x_n^k J$ and $W_k = \sum_{i=0}^{e} x_n^i U_i S + x_n^k V$ for $k > e$. Then $\operatorname{depth}_S \frac{T_k}{W_k}$ is constant for all $k > e$.

Proof.

Consider the following linear subspaces of S: $I := \sum_{i=0}^{e} x_n^i I_i$ and $U := \sum_{i=0}^{e} x_n^i U_i$. Note that I and U are not ideals in S.

If $I = U$, then the claim follows easily from the next chain of isomorphisms $\frac{T_k}{W_k} \cong \frac{x_n^k J}{x_n^k J \cap (I + x_n^k V)S} \cong \frac{x_n^k J}{x_n^k (I+V)S} \cong \frac{J}{(I+V)S}$ for all $k > e$, and hence $\operatorname{depth}_S \frac{T_k}{W_k}$ is constant for all $k > e$.

Assume now that $I \neq U$ and consider the following exact sequence

$$0 \to \frac{J}{V} \xrightarrow{\cdot x_n^k} \frac{T_k}{W_k} \to \frac{T_k}{W_k + x_n^k J} \to 0,$$

where the last term we denote by H_k. Note that $H_k \cong \frac{IS}{IS \cap (U + x_n^k J)S}$ and $IS \cap (U + x_n^k J)S = US + x_n^k IS$. Since $x_n^k H_k = 0$, H_k is a ${}^{S}/_{(x_n^k)}$−module. Then $\operatorname{depth}_S H_k = \operatorname{depth}_{S/(x_n^k)} H_k = \operatorname{depth}_R H_k$ because the graded maximal ideal m of R generates a zero dimensional ideal in ${}^{S}/_{(x_n^k)}$. But H_k over R is isomorphic with $\frac{\oplus_{i=0}^{k-1} I_i}{\oplus_{i=0}^{k-1} U_i} \cong \bigoplus_{i=0}^{k-1} \frac{I_i}{U_i}$, where $I_i = I_e$ and $U_i = U_e$ for $e < i < k$. It follows that $t := \operatorname{depth}_S H_k = \min_i \left\{ \operatorname{depth}_R \frac{I_i}{U_i} \right\}$.

If $\operatorname{depth}_S \frac{J}{V} = 0$, then the Depth Lemma gives us $\operatorname{depth}_S \frac{T_k}{W_k} = t = 0$ for all $k > e$ and hence we are done. Therefore we may suppose that $\operatorname{depth}_S \frac{J}{V} > 0$. Note that $t > 0$ implies $\operatorname{depth}_S \frac{T_k}{W_k} > 0$ by the Depth Lemma since otherwise $\operatorname{depth}_S \frac{T_k}{W_k} = \operatorname{depth}_S \frac{J}{V} = 0$, which is false. Next we will split the proof in two cases.

∘ Case $t = 0$.

Let $\mathcal{F} = \left\{ i \in \{0, \ldots, e\} \;\middle|\; \operatorname{depth}_R {}^{I_i}/_{U_i} = 0 \right\}$ and $L_i \subset I_i$ be the graded ideal containing U_i such that ${}^{L_i}/_{U_i} \cong H_m^0({}^{I_i}/_{U_i})$.

If $i \in \mathcal{F}$ and there exists $u \in (L \cap V) \setminus U_i$ then $(m^s, x_n^k)x_n^i u \subset W_k$ for some $s \in \mathbb{N}$, that is $\operatorname{depth}_S \dfrac{T_k}{W_k} = 0$ for all $k > e$.

Now consider the case when $L_i \cap V = U_i$ for all $i \in \mathcal{F}$. If $i \in \mathcal{F}$ then note that $L_i \subset L_j$ for $i < j \leq e$. Set $V' = V + L_e S$, $U' = U + \sum_{i \in \mathcal{F}} x_n^i L_i$ and $W_k' := U'S + x_n^k V' = U'S + x_n^k V$ because $x_n^k L_e S \subset U'S$. Consider the following exact sequence

$$0 \to \frac{W_k'}{W_k} \to \frac{T_k}{W_k} \to \frac{T_k}{W_k'} \to 0.$$

For the last term we have $H_m^0({}^{I_j}/{}_{U_j'}) = 0$, $0 \leq j \leq e$ and so the new $t > 0$, which is our next case. Thus we get $\operatorname{depth}_S \dfrac{T_k}{W_k'} > 0$ is constant for $k > e$. The first term is isomorphic to $\dfrac{U'S}{U'S \cap W_k}$. But $U'S \cap W_k = US + (U'S \cap x_n^k V)$ because $US \subset U'S$. Since $U'S \cap (x_n^k S) = x_n^k (U_e + L_e)S$ and $U_e \subset V$ it follows that

$$U'S \cap x_n^k V = x_n^k US + (x_n^k L_e S \cap x_n^k VS) = x_n^k US.$$

Consequently, the first term from the above exact sequence is isomorphic with $\dfrac{U'S}{US}$. Note that the annihilator of the element induced by some $u \in L_e \setminus V$ in ${}^{U'S}/{}_{US}$ contains a power of m and so $\operatorname{depth}_S \dfrac{U'S}{US} \leq 1$. The inequality is equality since x_n is regular on ${}^{U'S}/{}_{US}$. By the Depth Lemma we get $\operatorname{depth}_S \dfrac{T_k}{W_k} = 1$ for all $k > e$.

∘ Case $t > 0$.

If $\operatorname{depth}_R \dfrac{J}{V} \leq t = \operatorname{depth}_S H_k$ then the Depth Lemma gives us again the claim, i.e. $\operatorname{depth}_S \dfrac{T_k}{W_k} = \operatorname{depth}_S \dfrac{J}{V}$ for all $k > e$.

Assume that $\operatorname{depth}_S \dfrac{J}{V} > t$. Apply induction on t, the initial step $t = 0$ being done in the first case. Suppose that $t > 0$. Then $\operatorname{depth}_S \dfrac{J}{V} > t > 0$ implies that $\operatorname{depth}_S \dfrac{J}{V} \geq 2$ and so we may find a homogeneous polynomial $f \in m$ that is regular on $\dfrac{J}{V}$. Moreover we may find f to be regular also on all $\dfrac{I_i}{U_i}$, $i \leq e$. Then f is regular on $\dfrac{T_k}{W_k}$. Set $V'' := V + fJ$ and $U_i'' := U_i + fI_i$ for all $i \leq e$ and set $W_k'' := \sum_{i=0}^{e} x_n^i U_i'' S + x_n^k V''$. By Nakayama's Lemma we get $U'' \neq U$, and therefore $\operatorname{depth}_R \dfrac{I}{U''} = t - 1$ and by induction hypothesis it results that $\operatorname{depth}_S \dfrac{T_k}{W_k} = 1 + \operatorname{depth}_S \dfrac{T_k}{W_k''} =$ constant for all $k > e$.

Finally, note that we may pass from the first case to the second one and conversely. In this way U increases at each step. By Noetherianity at last we may arrive in finite steps to the case $I = U$, which was solved at the beginning. $\square$

The next corollary is in fact [IKM, Proposition 5.1] (see Proposition B:4.0.1) for depth. It follows easily from Lemma B:4.1.9 but also from [OY, Proposition 5.2] (see also [Y, Sections 2, 3].

COROLLARY B:4.1.10.

Let $e \in \mathbb{N}$, I and J monomial ideals in $S := K[x_1, \ldots, x_n]$. Consider I' and J' be the monomial ideals obtained from I and J in the following way: each generator whose degree in $x_n \geq e$ is multiplied by x_n and all the other generators are left unchanged. Then

$$\operatorname{depth}_S {}^{I}/_{J} = \operatorname{depth}_S {}^{I'}/_{J'}.$$

This leads us to the equivalent result of Theorem B:4.1.7 for depth.

THEOREM B:4.1.11.

Let I and J be two monomial ideals in S and $\overline{{}^{I}/_{J}}$ the canonical form of ${}^{I}/_{J}$. Then

$$\operatorname{depth}_S {}^{I}/_{J} = \operatorname{depth}_S \overline{{}^{I}/_{J}}.$$

Proof.

Assume that ${}^{I}/_{J}$ is of type $(k_1, \ldots, k_s)$ with respect to x_n and obviously $\overline{{}^{I}/_{J}}$ is of type $(1, 2, \ldots, s)$ with respect to x_n. Starting with $\overline{{}^{I}/_{J}}$, we apply Corollary B:4.1.10 till we obtain an ${}^{I_1'}/_{J_1'}$ of type $(k_1, k_1 + 1, \ldots, k_1 + s - 1)$ having the same depth as $\overline{{}^{I}/_{J}}$. We repeat the process until we get ${}^{I_s'}/_{J_s'}$ of type $(k_1, k_2, \ldots, k_s)$ with respect to x_n with the unchanged depth. Now we iterate and take the next variable. At the very end the claim will follow. $\square$

Theorem B:4.1.7 and Theorem B:4.1.11 give us the following theorem:

THEOREM B:4.1.12.

The Stanley conjecture holds for a factor of monomial ideals ${}^{I}/_{J}$ if and only if it holds for its canonical form $\overline{{}^{I}/_{J}}$.

Using Theorem B:4.1.11, instead of computing the depth or the sdepth of ${}^{I}/_{J}$, $J \subsetneq I \subset S$, we can compute it for the simpler module $\overline{{}^{I}/_{J}}$.

EXAMPLE B:4.1.13. We present the different timings for the depth and sdepth computation algorithms with and without extracting the canonical form. SINGULAR[Singular] was used in the depth computations while COCOA [CoCoA] and Rinaldo's paper[Ri] were used for the Stanley depth computation.

1. Consider the ideals from Example B:4.1.5(2).

   Timing for $\operatorname{sdepth} {}^{I}\!/_{J}$ computation: 22s.

   Timing for $\operatorname{sdepth} \overline{{}^{I}\!/_{J}}$ computation: 74 ms.

2. Consider $R = \mathbb{Q}[x, y, z]$ and $I = (x^{100}yz, x^{50}yz^{50}, x^{50}y^{50}z)$. Then the canonical form is $I' = (x^2yz, xyz^2, xy^2z)$.

   Timing for $\operatorname{sdepth} I$ computation: 13m 3s.

   Timing for $\operatorname{sdepth} I'$ computation: 21 ms.

   Notice that the difference in timings is very large. Therefore using the canonical form in the sdepth computation is a very important optimization step. On the other side, the depth computation is immediate in both cases. In the last example, the timing difference can be seen.

3. Consider $R = \mathbb{Q}[x, y, z, t, v, a_1, \ldots, a_5]$,

   $$I = (v^4x^{12}z^{73}, v^{87}t^{21}y^{13}, x^{43}y^{18}z^{72}t^{28}, vxy, vyz, vzt, vtx, a_1^{7000}, a_2^{413}),$$

   $$J = (v^5x^{13}z^{74}, v^{88}t^{22}y^{14}, x^{44}y^{19}z^{73}t^{29}, v^2x^2y^2, v^2y^2z^2, v^2z^2t^2, v^2t^2x^2).$$

   Timing for $\operatorname{depth} {}^{I}\!/_{J}$ computation: 16m 11s.

   Timing for $\operatorname{depth} \overline{{}^{I}\!/_{J}}$ computation: 11m.

B:4.2 The canonical form algorithm

We sketch the simple idea of the algorithm which computes the canonical form of a monomial ideal I. This can easily be extended to compute the canonical form of ${}^{I}\!/_{J}$ by simple applying it for $G(I) \cup G(J)$ and afterwards extracting the generators corresponding to I and J. This was used in Example B:4.1.13.

The algorithm is based on Remark B:4.1.4: for each variable x_i we build the list gp in which we save the pair (g, p), were p is chosen such that x_i^p enters the $g-$generator of the monomial ideal I. This list will be sorted by the powers p as in the following example

EXAMPLE B:4.2.1. Consider the ideal $I := (x^{13}, x^4y^7, y^7z^{10}) \subset \mathbb{Q}[x, y, z]$. Then for each variable we will obtain a different gp as shown below:

- For the first variable x, gp is equal to $\begin{array}{|c|c|c|c|}\hline 2 & 4 & 1 & 13 \\ \hline\end{array}$. Therefore I is of type $(4, 13)$ with respect to x. Hence, in order to obtain the canonical form with respect to x, one has to divide the second generator by $x^{4-1} = x^3$ and the first generator by $x^{13-2} = x^{11}$. After these computation we will get $I_1 = (x^2, xy^7, y^7z^{10})$. Note that I_1 is in the canonical form w.r.t. x.

- ○ For the second variable y, gp is equal to $\begin{array}{|c|c|c|c|}\hline 3 & 7 & 2 & 7 \\ \hline\end{array}$. Similar as above, one has to divide the second and the third generator by y^6, and hence it results $I_2 = (x^2, xy, yz^{10})$. Again, I_2 is in the canonical form w.r.t. y and x.

- ○ For the last variable z, gp is equal to $\begin{array}{|c|c|}\hline 3 & 10 \\ \hline\end{array}$. We divide the third generator of I_2 by z^9 and we get our final result $I' = (x^2, xy, yz)$., which is in the canonical form with respect to all variables.

Based on the above idea, we construct two procedures: `putIn` and `canonical` – the first one constructing the list gp, and the second one computing the canonical form of a monomial ideal. The proof of correctness and termination is trivial. The procedures were written in the SINGULAR language.

```
proc putIn(intvec v, int power, int nrgen)
{
    if(size(v) == 1)
    {
        v[1] = nrgen;
        v[2] = power;
        return(v);
    }
    int i,j;
    if(power <= v[2])
    {
        for(j = size(v)+2; j >=3; j--)
        {
            v[j] = v[j-2];
        }
        v[1] = nrgen;
        v[2] = power;
        return(v);
    }
    if(power >= v[size(v)])
    {
        v[size(v)+1] = nrgen;
        v[size(v)+1] = power;
        return(v);
    }
    for(j = size(v) + 2; (j>=4) && (power < v[j-2]); j = j-2)
    {
        v[j] = v[j-2];
        v[j-1] = v[j-3];
    }
    v[j] = power;
    v[j-1] = nrgen;
    return(v);
}
```

```
proc canonical(ideal I){
    int i,j,k;
    intvec gp;
    ideal m;
    intvec v;
    v = 0:nvars(basering);
    for(i = 1; i<=nvars(basering); i++)
    {
        gp = 0;
        v[i] = 1;
        for(j = 1; j<=size(I); j++)
        {
            if(deg(I[j],v) >= 1)
            {
                gp = putIn(gp,deg(I[j],v),j);
            }
        }
        k = 0;
        if(size(gp) == 2)
        {
            I[gp[1]] = I[gp[1]]/(var(i)^(gp[2]-1));
        }
        else
        {
            for(j = 1; j<=size(gp)-2;)
            {
                k++;
                I[gp[j]] = I[gp[j]]/(var(i)^(gp[j+1]-k));
                j = j+2;
                while((j<=size(gp)-2) && (gp[j-1] == gp[j+1]) )
                {
                    I[gp[j]] = I[gp[j]]/(var(i)^(gp[j+1]-k));
                    j = j + 2;
                }
            }
            if(j == size(gp)-1)
            {
                if(gp[j-1] == gp[j+1])
                {
                    I[gp[j]] = I[gp[j]]/(var(i)^(gp[j+1]-k));
                }
                else
                {
                    k++;
                    I[gp[j]] = I[gp[j]]/(var(i)^(gp[j+1]-k));
                }
            }
        }
        v[i] = 0;
    }
    return(I);
}
```

Part C

Constructive General Neron Desingularization

Introduction

Part C starts with a small introduction on Artin Approximation and the General Neron Desingularization. We give an algorithmic proof of this desingularization in the case of Noetherian local domains of dimension 1, $u : A \to A'$ is a regular morphism, B is a finite type $A-$algebra and $v : B \to A'$ an $A-$morphism. Thus we find a smooth $A-$algebra C such that v factors through C (see Theorem C:2.0.2), that is a General Neron Desingularization of B, v. The idea is nicely presented in 3 steps in the introduction of [PfPo2] and we recall it here below:

In *step 1* we reduce the problem to the case when $H_{B/A} \cap A \neq 0$, $H_{B/A}$ being the ideal defining the nonsmooth locus of B over A. Let $0 \neq d \in H_{B/A} \cap A$. This means geometrically that $\operatorname{Spec} B_d \to \operatorname{Spec} A_d$ is smooth. In the *second step* we construct a smooth A-algebra D, $A \subset D \subset A'$ and an A-morphism $v' : B \to D/d^3D$ such that $v \equiv v'$ modulo d^3A'. If A' is the completion $\hat{A}$ of A we can use $D = A$. The *third step* resolves the singularity. If $B = A[Y]/I$, $Y = (Y_1, \ldots, Y_n)$ then we can find $f = (f_1, \ldots, f_r)$, $r \leq n$ a system of polynomials from I, and an $r \times r$-minor M of the Jacobian matrix $\left(\dfrac{\partial f_i}{\partial Y_j}\right)$ such that $d \equiv MN$ modulo I for some $N \in \big((f) : I\big)$, where (f) denotes the ideal generated by the system f. Then $v'(MN) = ds$ for some $s \in 1 + dD$. Assume that $M = \det\left(\dfrac{\partial f_i}{\partial Y_j}\right)_{1 \leq i,j \leq r}$. Let H be the matrix obtained by adding to $\left(\dfrac{\partial f}{\partial Y}\right)$ the boarder block $(0|\mathrm{Id}_{n-r})$ and let G' be the adjoined matrix of H and $G = NG'$. Consider in $D[Y,T]$, $T = (T_1, \ldots, T_n)$ the ideal $J = \big((f, s(Y - y') - dG(y')T) : d^2\big)$, where $y' \in D^n$ is lifting $v'(Y)$. Then C is a suitable localization of the $B \otimes_A D$-algebra $D[Y,T]/(I,J)$ and v extends to C by $v(T) = t = (1/d^2)H(y')\big(v(Y) - y'\big)$.

Consider the following example. Let $A = \mathbb{Q}[x]_{(x)}$, $A' = \mathbb{C}\llbracket x \rrbracket$, $B = \dfrac{A[Y_1, Y_2]}{(Y_1^2 + Y_2^2)}$ and $a \in \mathbb{C}$ a transcendental element over $\mathbb{Q}$, $\bar{u} \in \mathbb{C}\llbracket x \rrbracket \setminus \mathbb{C}[x]_{(x)}$ and $u = a + x^6\bar{u}$. Let v be given by $v(Y_1) = xu$, $v(Y_2) = xiu$, where $i = \sqrt{-1}$. In step 1 we change B by $B_1 = A[Y_1, Y_2, Y_3]/I$, $I = (Y_1^2 + Y_2^2, x - 2Y_1Y_3)$ and extend v by $v(Y_3) = 1/(2u)$. We have $4Y_1^2Y_3^2 \in H_{B/A}$ which implies $d = x^2 \in H_{B/A} \cap A$. We define $D = A[a, a^{-1}, i]$ and $v'(Y) = y' = (xa, xia, 1/(2a))$. This is step two.

To understand step 3 we simplify the example taking $B = A[Y_1, Y_2]/(Y_1Y_2 - x^2)$, $u = 1 + x^6\bar{u}$ and v given by $v(Y_1) = xu$, $v(Y_2) = x/u$. Then $d = x \in H_{B/A}$, $D = A$, $y_1' = x = y_2'$. We obtain $H = \begin{pmatrix} Y_2 & Y_1 \\ 0 & 1 \end{pmatrix}$, $G = G' = \begin{pmatrix} 1 & -Y_1 \\ 0 & Y_2 \end{pmatrix}$, $N = 1$ and $J = \big((Y_1Y_2 - x^2, Y_1 - x - xT_1 + x^2T_2, Y_2 - x - x^2T_2) : x^2\big)$. This leads to

$$J = (xT_1T_2 - x^2T_2^2 + T_1, Y_1 - x - xT_1 + x^2T_2, Y_2 - x - x^2T_2)$$

and we obtain that $C \cong \left(\frac{A[T_1,T_2]}{(xT_1T_2 - x^2T_2^2 + T_1)}\right)_{1+xT_2} \cong \big(A[T_2]\big)_{1+xT_2}$ is a smooth A-algebra.

The implementation of our algorithm is done in several examples of Chapter 3.

Now let (A,m) be a Noetherian local ring of dimension 1, $A' = \hat{A}$, B a finite type $A-$algebra and $c \in \mathbb{N}$. Suppose that A is Henselian and the completion map $A \to \hat{A}$ is regular. If A is a discrete valuation ring, then Greenberg [Gr] showed that there exists a linear map $\nu : \mathbb{N} \to \mathbb{N}$ such that for each A-morphism $v : B \to A/m^{\nu(c)}$ there exists an A-morphism $v' : B \to A$ such that $v' \equiv v$ modulo m^c, that is A has the strong Artin Approximation Property (see [PfPo1],[DP2]).

Our Corollary C:2.0.20 shows that if A is Cohen$-$Macaulay and the Jacobian locus of B is not *too small*, then there exists a linear map as in Greenberg's case of the form $c \to 2e + c$, where e depends on the polynomial system of equations defining B.

Chapter C:1

Artin approximation and General Neron Desingularization

This chapter is based on the common work of the author together with D. Popescu, [APDP2]. We first introduce a couple of definitions needed in the next sections.

Definition C:1.0.1.

A ring morphism $u : A \to A'$ of Noetherian rings has ***regular fibers*** if for all prime ideals $P \in \operatorname{Spec} A$ the ring ${}^{A'}\!/_{PA'}$ is a regular ring, i.e. its localizations are regular local rings.

It has ***geometrically regular fibers*** if for all prime ideals $P \in \operatorname{Spec} A$ and all finite field extensions K of the fraction field of ${}^{A}\!/_{P}$ the ring $K \otimes_{A/P} {}^{A'}\!/_{PA'}$ is regular.

A flat morphism of Noetherian rings u is ***regular*** if its fibers are geometrically regular.

Definition C:1.0.2.

A local ring (A, m) is ***Henselian*** if the Implicit Function Theorem (C:1.0.3) holds in A.

A Henselian Noetherian local ring A is ***excellent*** if the completion map $A \to \hat{A}$ is regular.

Theorem C:1.0.3 (Implicit Function Theorem)**.**

Let $F \in A\,[\![x_1, \ldots, x_n, y]\!]$ *such that* $F(x_1, \ldots, x_n, 0) \in \langle x_1, \ldots, x_n\rangle$ *and* $\dfrac{\partial F}{\partial y}(x_1, \ldots, x_n, 0) \notin \langle x_1, \ldots, x_n\rangle$*. Then there exists a unique*

$$y(x_1, \ldots, x_n) \in \langle x_1, \ldots, x_n\rangle\, A\,[\![x_1, \ldots, x_n]\!]$$

such that

$$F\big(x_1, \ldots, x_n, y(x_1, \ldots, x_n)\big) = 0.$$

Definition C:1.0.4.

A Noetherian local ring (A, m) has the ***Artin approximation property*** if every finite system of polynomial equations over A has a solution in A if and only if has a solution in the completion $\hat{A}$ of A.

In fact A has the Artin approximation property if and only if every finite system f of polynomial equations over A in $Y = (Y_1, \ldots, Y_n)$ has its solutions in A dense with respect to the m-adic topology in the set of its solutions in $\hat{A}$; that is, for every solution $\hat{y}$ of f in $\hat{A}$ and every positive integer c there exists a solution y in A such that $y \equiv \hat{y}$ modulo $m^c\hat{A}$.

Definition C:1.0.5.

Let K be a field. We denote by $K\langle x\rangle$, where $x = x_1, \dots, x_n$, the ***algebraic power series*** ring over K, i.e. the algebraic closure of the polynomial ring $K[x]$ in the formal power series ring $K[\![x]\!]$.

M. Artin showed in [Artin1] that the algebraic power series over a field has the property of Artin approximation. He also set in [Artin2] several conjectures, one of them is solved in the following theorem.

Theorem C:1.0.6 ([DP2, Theorem 1.3])**.**

An excellent Henselian local ring has the property of Artin approximation property.

The proof is based on the existence of the so-called *General Neron Desingularization*, that is the following theorem generalizes the Neron Desingularization [Ne], [Artin1].

Theorem C:1.0.7 (General Neron Desingularization, Popescu [DP1], [DP2], [DP3], Swan [Sw])**.**

Let $u : A \to A'$ be a regular morphism of Noetherian rings and B a finite type A-algebra. Then any A-morphism $v : B \to A'$ factors through a smooth A-algebra C, that is v is a composite A-morphism $B \to C \to A'$.

Remark C:1.0.8. Using this theorem we get an easy proof of Theorem C:1.0.6. Indeed, let f be a finite system of polynomial equations over A in $Y = (Y_1, \dots, Y_n)$ and $\hat{y}$ a solution of f in $\hat{A}$. Set $B = A[Y]/(f)$ and let the morphism

$$\begin{tikzcd} v: & B \arrow[r] & \hat{A} \\ & Y \arrow[r, mapsto] & \hat{y}. \end{tikzcd}$$

By Theorem C:1.0.7, v factors through a smooth A-algebra C, that is v is a composite A-morphism $B \to C \to A'$. Thus changing B by C we may reduce the problem to the case when B is smooth over A. Then $1 \in \big((g) : I\big)M$ for some polynomials $g = (g_1, \dots, g_r)$ from (f) and a $r \times r$-minor M of the Jacobian matrix $\left(\dfrac{\partial g}{\partial Y}\right)$. Thus $g(\hat{y}) = 0$ and $M(\hat{y})$ is invertible. By the Implicit Function Theorem there exists $y \in A$ such that $y \equiv \hat{y}$ modulo $m\hat{A}$.

The main purpose of this part is to give an algorithmic proof of Theorem C:1.0.7 when A, A' are one dimensional Noetherian local domains and $A \supset \mathbb{Q}$. This is published in [APDP2]. An extension of our algorithm can be found in [PfPo2]. We may take the same General Neron Desingularization for $v, v' : B \to A'$ if they are closed enough as Examples C:2.0.3 and C:2.0.9 show. This remark was basic in [DP8, Theorem 10]. The last section of this chapter computes the General Neron Desingularization in several examples. We should point that the General Neron Desingularization is not unique and it is better to speak above about **a** General Neron Desingularization.

Definition C:1.0.9.

A Noetherian local ring (A, m) has the ***strong Artin approximation property*** if for every finite system of polynomial equations f in $Y = (Y_1, \ldots, Y_n)$ over A there exists a map $\nu : \mathbb{N} \to \mathbb{N}$ with the following property

> If $y' \in A_n$ satisfies $f(y') \equiv 0$ modulo $m^{\nu(c)}$, $c \in \mathbb{N}$, then there exists a solution $y \in A^n$ of f with $y \equiv y'$ modulo m^c.

M. Greenberg [Gr] proved that excellent Henselian discrete valuation rings have the strong Artin approximation property and ν is linear in this case. M. Artin [Artin1] showed that the algebraic power series ring over a field has also the strong Artin approximation property. In [PfPo1] (see also [KMPPR]) it states that Noetherian complete local rings have the strong Artin approximation property and it follows that A has the strong Artin approximation property if it has the Artin approximation property. Thus Theorem C:1.0.6 gives that excellent Henselian local rings have the strong Artin approximation property. An easy proof of this fact is given in [DP2, 4.5] using Theorem C:1.0.7 and ultrapower methods.

When A' is the completion of a Cohen-Macaulay local ring A of dimension 1 we show that we may have a linear Artin function as it happens in the Greenberg's case (see C:2.0.19). More precisely, the Artin function is given by $c \mapsto 2e + c$, where e depends from the polynomial system of equations defining B. Later this result was extended in [PfPo2] in the case of non Cohen-Macaulay local rings of one dimension. If $\dim A > 1$ then ν may be not linear as shows [Ro].

CHAPTER C:2

A METHOD TO COMPUTE A GENERAL NERON DESINGULARIZATION IN THE FRAME OF ONE DIMENSIONAL LOCAL DOMAINS

Let $u : A \to A'$ be a flat morphism of Noetherian local domains of dimension 1. Suppose that $A \supset \mathbb{Q}$ and the maximal ideal m of A generates the maximal ideal of A'. Then u is regular morphism. Moreover, we suppose that there exist canonical inclusions $k = {}^{A}\!/_{m} \to A$, $k' = {}^{A'}\!/_{mA'} \to A'$ such that $u(k) \subset k'$.

Let $B = {}^{A[Y]}\!/_{I}$, $Y = (Y_1, \ldots, Y_n)$. If $f = (f_1, \ldots, f_r)$, $r \leq n$ is a system of polynomials from I then we can define the ideal Δ_f generated by all $r \times r-$minors of the Jacobian matrix $\left(\dfrac{\partial f_i}{\partial Y_j}\right)$. After Elkik [El], let $H_{B/A}$ be the radical of the ideal $\sum_f \big((f) : I\big)\Delta_f B$, where the sum is taken over all systems of polynomials f from I with $r \leq n$. Then B_P, $P \in \operatorname{Spec} B$ is essentially smooth over A if and only if $P \not\supset H_{B/A}$ by the Jacobian criterion for smoothness. Thus $H_{B/A}$ measures the non smooth locus of B over A.

DEFINITION C:2.0.1.
B is ***standard smooth*** over A if there exists f in I as above such that

$$1 \in \big((f) : I\big)\Delta_f B.$$

The aim of this chapter is to give an easy algorithmic proof of the following theorem.

THEOREM C:2.0.2.
Any A-morphism $v : B \to A'$ factors through a standard smooth A-algebra B'.

If A is essentially of finite type over $\mathbb{Q}$, then the ideal $H_{B/A}$ can be computed in SINGULAR by following its definition but it is easier to describe only the ideal $\sum_f \big((f) : I\big)\Delta_f B$ defined above. This is the case considered in our algorithmic part, let us say $A \cong {}^{k[x]}\!/_{F}$ for some variables $x = (x_1, \ldots x_t)$, and the completion of A' is ${}^{K[\![x]\!]}\!/_{F}$ for some field extension $k \subset K$. When v is defined by polynomials y from $K[x]$ then our problem is easy. Let L be the field obtained by adjoining to k all coefficients of y. Then $R = {}^{L[x]}\!/_{F}$ is a subring of A' containing $\operatorname{Im} v$ which is essentially smooth over A. Then we may take B' as a standard smooth A-algebra such that R is a localization of B'. Consequently we suppose usually that y is not polynomial defined and moreover L is not a finite type field extension of k.

We may suppose that $v(H_{B/A}) \neq 0$. Indeed, if $v(H_{B/A}) = 0$ then v induces an A-morphism $v' : B' = B/H_{B/A} \to A'$ and we may change (B, v) by (B', v'). Applying this trick several times we reduce to the case $v(H_{B/A})\neq 0$. However the fraction field of $\operatorname{Im} v$ is essentially smooth over A by separability, that is $H_{\operatorname{Im} v/A}A'\neq 0$ and in the worst case our trick will change B by $\operatorname{Im} v$ after several steps.

Choose $P'\in\Delta_f\big((f):I\big)\setminus I$ for some system of polynomials $f=(f_1,\ldots,f_r)$ from I and $d'\in\big(v(P')A'\big)\cap A$, $d'\neq 0$. Moreover we may choose P' to be from $M\big((f):I\big)$ where M is a $r\times r-$minor of $\left(\dfrac{\partial f}{\partial Y}\right)$. Then $d'=v(P')z\in\big(v(H_{B/A})\big)\cap A$ for some $z\in A'$. Set $B_1={}^{B[Z]}/_{(f_{r+1})}$, where $f_{r+1}=-d'+P'Z$ and let $v_1:B_1\to A'$ be the map of B-algebras given by $Z\to z$. It follows that $d'\in\big((f,f_{r+1}):(I,f_{r+1})\big)$ and $d'\in\Delta_f$, $d'\in\Delta_{f_{r+1}}$. Then $d=d'^2\equiv P$ modulo (I,f_{r+1}) for $P=P'^2Z^2\in H_{B_1/A}$. For the reduction change B by B_1 and the Jacobian matrix $J=\left(\dfrac{\partial f}{\partial Y}\right)$ will be now the new J given by $\begin{pmatrix} J & 0\\ * & P'\end{pmatrix}$. Note that $d\in H_{B/A}\cap A$.

Example C:2.0.3. Let $a_1,a_2\in\mathbb{C}$ be two elements algebraically independent over $\mathbb{Q}$ and ρ a root of the polynomial T^2+T+1 in $\mathbb{C}$. Then $k'=\dfrac{\mathbb{Q}(a_1,a_2)[a_3]}{(a_3^2+a_3+1)}\cong\mathbb{Q}(\rho,a_1,a_2)$. Let $A=\left(\dfrac{\mathbb{Q}[x_1,x_2]}{(x_1^3-x_2^2)}\right)_{(x_1,x_2)}$ and $B=\dfrac{A[Y_1,Y_2,Y_3]}{(Y_1^3-Y_2^3)}$, $A'=\dfrac{k'[\![x_1,x_2]\!]}{(x_1^3-x_2^2)}$ and the map v defined as

$$\begin{tikzcd}
v: & B \arrow[r] & A' \\
 & Y_1 \arrow[r, mapsto] & a_1x_2 \\
 & Y_2 \arrow[r, mapsto] & a_1a_3x_2 \\
 & Y_3 \arrow[r, mapsto] & a_1\displaystyle\sum_{i=0}^{30}\frac{x_1^i}{i!}+a_2x_2\sum_{i=31}^{50}\frac{x_1^i}{i!}
\end{tikzcd}$$

This is an easy example. Indeed, let $v'':B''=A\big[a_3,a_1x_2,v(Y_3)\big]\to A'$ be the inclusion. We have $\operatorname{Im} v\subset B''\cong\dfrac{A[T,Y_1,Y_3]}{(T^2+T+1)}$ and $B''_{2a_3+1}\cong\left(\dfrac{A[T,Y_1,Y_3]}{(T^2+T+1)}\right)_{2T+1}$ is a smooth A-algebra, which could be taken as a General Neron Desingularization of B. Applying our algorithm we will get more complicated General Neron Desingularizations but useful for an illustration of our construction.

Then $\operatorname{Im} v$, the new B will be $\dfrac{B}{\operatorname{Ker} v}$, where the kernel is generated by the following polynomial:

```
ker[1]=Y1^2+Y1*Y2+Y2^2
```

Next we choose $f=Y_1^2+Y_1Y_2+Y_2^2$ and we have $M=2Y_2+Y_1$ and $1\in\big((f):I\big)$ and hence $P'=Y_1+2Y_2$. Therefore $v(P')=(2a_1a_3+a_1)\cdot x_2$ and $d'=x_2$, $z=\dfrac{1}{2a_1a_3+a_1}$. Therefore $d=d'^2=x_2^2$.

To be able to construct $\mathbb{Q}\left[\frac{1}{2a_1a_3+a_1}\right][x]$ in Singular we add a new variable a and we factorize with the corresponding polynomial $2a_1a_3\cdot a+a_1\cdot a-1$. We consider a as a new parameter from $k'\subset A'$. Then we replace B by $B_1=\frac{B[Y_4]}{(-d'+P'Y_4)}$ and extend v to a map $v_1:B_1\to A'$ given by $Y_4\mapsto a$. Changing B by B_1 we may assume that $d\in H_{B/A}\cap A$.

Example C:2.0.4. Note that we could use B instead $\operatorname{Im} v$. In this case we choose $f=Y_1^3-Y_2^3$ and take $M=3Y_2^2$ and $1\in\big((f):I\big)$. Therefore we obtain $P'=3Y_2^2$, $d'=x_2^2$, $d=x_2^4$ and the next computations are harder as we will see in the Examples C:2.0.16 and C:3.0.3.

Remark C:2.0.5. We would like to work above with $A''=\frac{\mathbb{C}\,[\![x_1,x_2]\!]}{(x_1^3-x_2^2)}$ instead of A', v being given by $v(Y_2)=a_1\rho x_2$. But this is hard since we cannot work in Singular with an infinite set of parameters. We have two choices. If the definition of v involves only a finite set of parameters then we proceed as Example C:2.0.3 using some $A'\supset\operatorname{Im} v$. Otherwise, we will see later that in the computation of the General Neron Desingularization we may use only a finite number of the coefficients of the formal power series defining $v(Y)$ and so this computation works in Singular.

Remark C:2.0.6. As one may observe, the algorithm could compute also in the case when A' is not a domain, but there exist $P\in M\big((f):I\big)$ as above and a regular element $d\in m$ with $d\equiv P$ modulo I. If A is Cohen-Macaulay we may reduce to the case when there exists a regular element $d\in H_{B/A}\cap A$. However, it is hard usually to reduce to the case when $d\equiv P$ modulo I for some $P\in M\big((f):I\big)$. Sometimes this is possible as shows the following example.

Example C:2.0.7. Let $a_1,a_2\in\mathbb{C}$ be two elements algebraically independent over $\mathbb{Q}$. Consider $A=\left(\frac{\mathbb{Q}[x_1,x_2,x_3]}{(x_2^3-x_3^2,x_1^3-x_3^2)}\right)_{(x_1,x_2,x_3)}$ and $B=\frac{A[Y_1,Y_2,Y_3]}{(Y_1^3-Y_2^3)}$, $K'=\frac{\mathbb{Q}(a_1,a_2)[a_3]}{(a_3^2-a_1a_2)}$, $A'=\frac{K'\,[\![x_1,x_2,x_3]\!]}{(x_2^3-x_3^2,x_1^3-x_3^2)}$ and the map v defined as

$$\begin{tikzcd}
v: & B \arrow[r] & A' \\
 & Y_1 \arrow[r, mapsto] & a_3x_1 \\
 & Y_2 \arrow[r, mapsto] & a_3x_2 \\
 & Y_3 \arrow[r, mapsto] & a_1\displaystyle\sum_{i=0}^{30}\frac{x_3^i}{i!}+a_2\sum_{i=31}^{50}\frac{x_3^i}{i!}
\end{tikzcd}$$

Then $\operatorname{Im}\ v$, the new B, will be $\frac{B}{\operatorname{Ker}\ v}$, where the kernel is generated by six polynomials:

```
ker[1]=x2*Y1-x1*Y2
ker[2]=Y1^3-Y2^3
ker[3]=x1*Y1^2-x2*Y2^2
ker[4]=x1^2*Y1-x2^2*Y2
ker[5]=x1*x2^2*Y2-x3^2*Y1
ker[6]=x1^2*x2*Y2^2-x3^2*Y1^2
```

Next we choose $f = x_2Y_1 - x_1Y_2$ and we have $M = -x_1$. We may take $N = -x_3^2 \in \big((f) : I\big)$ and $P' = x_1x_3^2$. Note that $x_1 - x_2$ is a zero divisor in A but $d' = P'$ is regular in A. In this example we may take $d = d' = P' = P$.

Remark C:2.0.8. Replacing B by $\operatorname{Im} v$ can be a hard goal if A' is a factor of the power series ring over $\mathbb{C}$ in some variables x and $v(Y)$ is defined by formal power series whose coefficients form an infinite field extension F of $\mathbb{Q}$. If $v(Y)$ are polynomials in x as in Examples C:2.0.3 and C:2.0.7 then it is trivial to find a General Neron Desingularization of B as we explained already in the last ideas presented in the previous chapter. For instance in Example C:2.0.7, B' could be a localization of $K' \otimes_{\mathbb{Q}} A$. Thus Examples C:2.0.3 and C:2.0.7 have no real importance, they being useful only for an illustration of our algorithm. This is the reason that in the next examples the field L obtained by adjoining to k all coefficients of y, will be an infinite type field extension of k and $v(Y)$ are not all polynomials in x.

However, this will complicate the algorithm because we are not able to tell to the computer who $v(Y)$ is and so how to obtain d'. We may choose an element $a \in m$ and find a minimal $c \in \mathbb{N}$ such that $a^c \in \big(v(M)\big) + \big(a^{2c}\big)$ (this is possible because $\dim A = 1$). Set $d' = a^c$. It follows that $d' \in \big(v(M)\big) + (d'^2) \subset \big(v(M)\big) + (d'^4) \subset \ldots$ and so $d' \in \big(v(M)\big)$, that is $d' = v(M)z$ for some $z \in A'$. Certainly we cannot find precisely z but later it is enough to know just a kind of truncation of it modulo d'^6.

Example C:2.0.9. Let $a_i \in \mathbb{C}$, $i \in \mathbb{N}$ be elements algebraically independent over $\mathbb{Q}$ and ρ a root of the polynomial $T^2 + T + 1$ in $\mathbb{C}$. Let $A = \left(\dfrac{\mathbb{Q}[x_1, x_2]}{(x_1^3 - x_2^2)}\right)_{(x_1,x_2)}$, $B = \dfrac{A[Y_1, Y_2, Y_3]}{(Y_1^2 + Y_1Y_2 + Y_2^2)}$, $A' = \dfrac{\mathbb{C}\llbracket x_1, x_2\rrbracket}{(x_1^3 - x_2^2)}$ and the map v defined as

$$\begin{tikzcd}
v: & B \arrow[r] & A' \\
& Y_1 \arrow[r, mapsto] & a_1\left(x_2 + \sum_{i\geq 7} a_i x_2^i\right) \\
& Y_2 \arrow[r, mapsto] & a_1a_3\left(x_2 + \sum_{i\geq 7} a_i x_2^i\right) \\
& Y_3 \arrow[r, mapsto] & a_1\sum_{i=0}^{9}\frac{x_1^i}{i!} + x_2\sum_{i=10}^{\infty} a_{i-8}\frac{x_1^i}{i!}
\end{tikzcd}$$

As in Example C:2.0.3 we may take $d' = x_2$, $d = d'^2$ and a. Our algorithm follows the same path as in Examples C:2.0.3, C:2.0.15 and C:3.0.2 providing the same General Neron Desingularization. This time we cannot find an easy General Neron Desingularization as in the first part of Example C:2.0.3.

Example C:2.0.10. Let $A = \dfrac{\mathbb{Q}[x_1,x_2]_{(x_1,x_2)}}{(x_1^2 - x_2^3)}$ and $A' = \dfrac{\mathbb{C}\,[\![x_1,x_2]\!]}{(x_1^2 - x_2^3)}$. Then the inclusion $A \subset A'$ is regular. Let $\theta_i = \displaystyle\sum_{j=0}^{\infty} \alpha_{ij} x_2^j + x_1 \sum_{j=0}^{\infty} \beta_{ij} x_2^j \in \mathbb{C}\,[\![x_1,x_2]\!]$ for $i = 3,4$ with $\alpha_{i0} = 1$ and $y_1 = \dfrac{\theta_3^3}{\theta_4^2}$, $y_2 = \dfrac{\theta_4^2}{\theta_3}$, $y_3 = x_2\theta_3$, $y_4 = x_2\theta_4$. Let $f_1 = Y_3^2 - x_2^2 Y_1 Y_2$, $f_2 = Y_4^2 - x_2 Y_2 Y_3$ be polynomials in $A[Y]$, $Y = (Y_1,\dots,Y_4)$ and set $B = {}^{A[Y]}/_{f}$, $f = (f_1, f_2)$.

If R is a domain and $u \in R$ is such that $Y^2 - u \in R[Y]$ has no solutions in $Q(R)$ then it is easy to see that $R[Y]/(Y^2 - u)$ is a domain too. In our case we obtain that $R = A[Y_1,Y_2,Y_3]/(f_1)$ and $B = R[Y_4]/(f_2)$ are domains too. Then the map $v : B \to A'$ given by $Y \mapsto y = (y_1,\dots,y_4)$ is injective if we suppose that θ_3, θ_4 are algebraically independent over A. This follows since B is a domain and $\dim B = \operatorname{tr}\deg_{Q(A)} Q(B) = \operatorname{tr}\deg_{Q(A)} Q(\operatorname{Im} v) = 2 = \dim \operatorname{Im} v$. Moreover we assume that the fields $L_i = \mathbb{Q}\big((\alpha_{ij},\beta_{ij})_j\big)$, $i = 3,4$ have infinite transcendental degree over $\mathbb{Q}$. The Jacobian matrix $\left(\dfrac{\partial f}{\partial Y}\right)$ have a $2\times 2-$minor $M = \det\left(\dfrac{\partial f_i}{\partial Y_j}\right)_{\substack{1\le i\le 2\\ 3\le j\le 4}} = 4Y_3Y_4 \notin (f)$. Note that $v(M) = x_2^2 y_5$, where $y_5 = \dfrac{1}{4\theta_3\theta_4}$. Then we may take $B_1 = B[Y_5]/(f_3)$, $f_3 = -x_2^2 + MY_5$ and v_1 given by $Y_5 \mapsto y_5$. Clearly, $P = M^2Y_5^2 \in H_{B_1/A}$ and $0 \neq d = x_2^4 = v_1(P) \in A$.

Thus we may suppose that there exists $f = (f_1,\dots,f_r)$, $r \le n$ a system of polynomials from I, a $r \times r-$minor M of the Jacobian matrix $\left(\dfrac{\partial f_i}{\partial Y_j}\right)$ and $N \in \big((f) : I\big)$ such that $0 \neq d \equiv MN$ modulo I. Set $\bar{A} = A/(d^3)$, $\bar{A}' = A'/d^3A'$, $\bar{u} = \bar{A} \otimes_A u$, $\bar{B} = B/d^3B$, $\bar{v} = \bar{A} \otimes_A v$. Clearly, $\bar{u}$ is a regular morphism of Artinian local rings.

Remark C:2.0.11. The whole proof could work with $\bar{A} = A/d^2u$ for any $u \in m$. We prefer to take $u = d$ as is done in [DP2] and [DP4] but we could choose $u \neq d$, $u \in m\backslash m^2$ for easy computations.

By [GD, 19, 7.1.5] for every field extension L/k there exists a flat complete Noetherian local $\bar{A}$-algebra $\tilde{A}$, unique up to an isomorphism, such that $m\tilde{A}$ is the maximal ideal of $\tilde{A}$ and $\tilde{A}/m\tilde{A} \cong L$. It follows that $\tilde{A}$ is Artinian. On the other hand, we may consider the localization A_L of $L \otimes_k \bar{A}$ in $m(L \otimes_k \bar{A})$ which is Artinian and so complete. By uniqueness we see that $A_L \cong \tilde{A}$. Set $k' = A'/mA'$. It follows that $\bar{A}' \cong A_{k'}$. Note that A_L is essentially smooth over A by base change and $\bar{A}'$ is a filtered union of sub-$\bar{A}$-algebras A_L with L/k finite type field sub extensions of k'/k.

Let v be given by $Y \to y \in A'^n$. Choose L/k a finite type field extension such that A_L contains the residue class $\bar{y} \in \bar{A}'^n$ induced by y. In fact $\bar{y}$ is a vector of polynomials in the generators of m with the coefficients c_ν in k' and we may take $L = k\big((c_\nu)_\nu\big)$. Then $\bar{v}$ factors through A_L. Assume that $k\big[(c_\nu)_\nu\big] \cong k\big[(U_\nu)_\nu\big]/\bar{J}$ for some new variables U and a prime ideal $\bar{J} \subset k[U]$. We have $H_{L/k} \neq 0$ because L/k is separable. Then we may assume that there exist $\omega = (\omega_1, \ldots, \omega_p)$ in $\bar{J}$ such that $\rho = \det\left(\dfrac{\partial \omega_i}{\partial U_\nu}\right)_{i,\nu \in [p]} \neq 0$ and a nonzero polynomial $\tau \in \big((\omega) : \bar{J}\big) \setminus \bar{J}$. Thus L is a fraction ring of the smooth k-algebra $\big(k[U]/(\omega)\big)_{\rho\tau}$. Note that ω, ρ, τ can be considered in A because $k \subset A$ and $c_\nu \in A'$ because $k' \subset A'$.

Then $\bar{v}$ factors through a smooth $\bar{A}$-algebra $C \cong \big(\bar{A}[U]/(\omega)\big)_{\rho\tau\gamma}$ for some polynomial γ which is not in $m\big(\bar{A}[U]/(\omega)\big)_{\rho\tau}$.

Lemma C:2.0.12.
There exists a smooth A-algebra D such that $\bar{v}$ factors through $\bar{D} = \bar{A} \otimes_A D$.

Proof.
By our assumptions $u(k) \subset k'$. Set $D = \big(A[U]/(\omega)\big)_{\rho\tau\gamma}$ and $w : D \to A'$ be the map given by $U_\nu \to c_\nu$. We have $C \cong \bar{A} \otimes_A D$. Certainly, $\bar{v}$ factors through $\bar{w} = \bar{A} \otimes_A w$ but in general v does not factor through w. □

Remark C:2.0.13. If $A' = \hat{A}$ then $\bar{A} \cong \bar{A}'$ and we may take $D = A$.

Remark C:2.0.14. Suppose that $k \subset A$ but $L \not\subset A'$ and so $k' \not\subset A'$. Then $D = \left(\dfrac{A[U, Z]}{(\omega - d^3 Z}\right)_{\rho\tau\gamma}$, $Z = (Z_\nu)$ is a smooth A-algebra and $\bar{D} \cong C[Z]$. Since $\bar{v}$ factors through a map $C \to \bar{A}'$ given by $U \mapsto \lambda + d^3 A'$ for some λ in A' we see that $\omega(\lambda) \equiv 0$ modulo d^3, that is $\omega(\lambda) = d^3 z$ for some z in A'. Let $w : D \to A'$ be the A-morphism given by $(U, Z) \mapsto (\lambda, z)$. Certainly, $\bar{v}$ factors through $\bar{w} = \bar{A} \otimes_A w$ but in general v does not factor through w. If also $k \not\subset A$ then the construction of D goes as above but using a lifting of ω, τ, γ from $k[U]$ to $A[U]$. In both cases we may use D as it follows.

Example C:2.0.15. We reconsider Example C:2.0.3. We already know that $d = x_2^2$. The algorithm gives us the following output:

```
This is C:
//   characteristic : 0
//   number of vars : 5
//        block   1 : ordering dp
//                  : names    a1 a3 a x1 x2
//        block   2 : ordering C
// quotient ring from ideal
_[1]=3*a1*a+2*a3+1
```

```
_[2]=a3^2+a3+1
_[3]=x1^3-x2^2
_[4]=x2^6
This is D:
//   characteristic : 0
//   number of vars : 5
//        block   1 : ordering dp
//                  : names    a1 a3 a x1 x2
//        block   2 : ordering C
// quotient ring from ideal
_[1]=3*a1*a+2*a3+1
_[2]=a3^2+a3+1
_[3]=x1^3-x2^2
```

Indeed,

$$C = \frac{\bar{A}[a_1, a_3, a]}{(3a_1a + 2a_3 + 1, a_3^2 + a_3 + 1, x_2^6)}$$

and

$$D = \frac{A[a_1, a_3, a]}{(3a_1a + 2a_3 + 1, a_3^2 + a_3 + 1)}.$$

Note that the $3a_1a + 2a_3 + 1$ comes from the standard basis computation of the ideal $(2a_1a_3a + a_1a - 1, a_3^2 + a_3 + 1)$ and in D we have a_3, a_1, $2a_3 + 1$ invertible.

Example C:2.0.16. Now we reconsider Example C:2.0.4. We know that $d = x_2^4$. The algorithm gives us the following output:

```
This is C:
//   characteristic : 0
//   number of vars : 5
//        block   1 : ordering dp
//                  : names    a1 a3 a x1 x2
//        block   2 : ordering C
// quotient ring from ideal
_[1]=a3^2+a3+1
_[2]=x1^3-x2^2
_[3]=a1^2*a-a3
_[4]=x2^12

This is D:
//   characteristic : 0
//   number of vars : 5
//        block   1 : ordering dp
//                  : names    a1 a3 a x1 x2
//        block   2 : ordering C
// quotient ring from ideal
_[1]=a3^2+a3+1
_[2]=x1^3-x2^2
_[3]=a1^2*a-a3
```

Indeed,

$$C = \frac{\bar{A}[a_1, a_3, a]}{(a_3^2 + a_3 + 1, a_1^2 a - a_3, x_2^{12})}$$

and

$$D = \frac{A[a_1, a_3, a]}{(a_3^2 + a_3 + 1, a_1^2 a - a_3)}.$$

Note that a_3, a_1 are invertible in D.

Example C:2.0.17. In the case of Example C:2.0.7 we obtain the following output:

```
This is C:
//   characteristic : 0
//   number of vars : 5
//        block   1 : ordering dp
//                  : names    a1 a3 x1 x2 x3
//        block   2 : ordering C
// quotient ring from ideal
_[1]=x2^3-x3^2
_[2]=x1^3-x3^2
_[3]=x3^8
This is D:
//   characteristic : 0
//   number of vars : 5
//        block   1 : ordering dp
//                  : names    a1 a3 x1 x2 x3
//        block   2 : ordering C
// quotient ring from ideal
_[1]=x2^3-x3^2
_[2]=x1^3-x3^2
```

Indeed this is the case since we have $d = x_1 x_3^2$ and hence

$$C = \frac{\bar{A}[a_1, a_3]}{(x_3^8)}$$

and

$$D = A[a_1, a_3].$$

Example C:2.0.18. In Example C:2.0.10 we consider a_1, a_2 algebraically independent over $\mathbb{Q}$ and set $\theta'_3 = 1 + a_1x_2$ and $\theta'_4 = 1 + a_2x_2^2$. Suppose that $\theta'_i \equiv \theta_i$ modulo x_2^{12}. We have $y_3 = x_2\theta_3$, $y_4 = x_2\theta_4$, $y_1 = \dfrac{\theta_3^3}{\theta_4^2}$, $y_2 = \dfrac{\theta_4^2}{\theta_3}$, $y_5 = \dfrac{1}{4\theta_3\theta_4}$. Choose y'_i, $i \in [5]$ polynomials with degrees ≤ 11 in x_2 and linear in x_1 such that $y'_i \equiv y_i \operatorname{mod}(x_1^2, x_2^{12})$. We get $y'_1 \equiv y_1 = \theta_3^3/\theta_4^2 \equiv \theta'^3_3/\theta'^2_4 \operatorname{mod}(x_1^2, x_2^{12})$ and similarly for y'_i, $i > 1$. Here we use the fact that $\theta_4^{-2} = \sum_{j=1}^{e}(1-\theta_4^2)^j$ for some $e \gg 0$ because $1 - \theta_4^2$ is nilpotent in the ring $\bar{A}[a_1, a_2, a_3, a_4]$. Thus the coefficients of y'_i, $i \in [5]$ belong to the field L obtained by adjoining to $\mathbb{Q}$ the coefficients of θ'_3, θ'_4. Note that in this case $L = Q\big(\mathbb{Q}[a_1, \ldots, a_4]\big)$. Then we obtain the following output:

```
This is C:
//   characteristic : 0
//   number of vars : 4
//        block   1 : ordering dp
//                  : names    a1 a2 x1 x2
//        block   2 : ordering C
// quotient ring from ideal
_[1]=x2^3-x1^2
_[2]=x1^8
This is D:
//   characteristic : 0
//   number of vars : 4
//        block   1 : ordering dp
//                  : names    a1 a2 x1 x2
//        block   2 : ordering C
// quotient ring from ideal
_[1]=x2^3-x1^2
```

Thus $C = \dfrac{\bar{A}[a_1, \ldots, a_4]}{(x_2^{12})} \cong \bar{A}[a_1, \ldots, a_4]$ which is smooth over $\bar{A}$. Then D is a localization of $A[a_1, \ldots, a_4]$, where θ'_3, θ'_4 must be invertible.

Back to our proof note that the composite map $\bar{B} \to C \to \bar{D}$ is given by $Y \to y' + d^3D$ for some $y' \in D^n$. Thus $I(y') \equiv 0$ modulo d^3D. Since $\bar{v}$ factors through $\bar{w}$ we see that $\bar{w}(y' + d^3D) = \bar{y}$. Set $\tilde{y} = w(y')$. We get $y - \tilde{y} \in d^3A'^n$, let us say $y - \tilde{y} = d^2\varepsilon$ for $\varepsilon \in dA'^n$.

We have $d \equiv P$ modulo I and so $P(y') \equiv d$ modulo d^3 in D because $I(y') \equiv 0$ modulo d^3D. Thus $P(y') = ds$ for a certain $s \in D$ with $s \equiv 1$ modulo d. Assume that $P = NM$ for some $N \in \big((f) : I\big)$. Recall from beginning of Chapter C:2 that the new M is now the old one multiplied with P' and the new N is the old one multiplied with Z^2. Let H be the $n \times n-$matrix obtained adding down to $\left(\dfrac{\partial f}{\partial Y}\right)$ as a border the block $\big(0 \mid \mathrm{Id}_{n-r}\big)$.

Let G' be the adjoint matrix of H and $G = NG'$. We have

$$GH = HG = NM\mathrm{Id}_n = P\mathrm{Id}_n$$

and so

$$ds\mathrm{Id}_n = P(y')\mathrm{Id}_n = G(y')H(y').$$

Then $t := H(y')\varepsilon \in dA'^n$ satisfies

$$G(y')t = P(y')\varepsilon = ds\varepsilon$$

and so

$$s(y - \tilde{y}) = dw\big(G(y')\big)t.$$

Let

$$h = s(Y - y') - dG(y')T, \tag{C:2.1}$$

where $T = (T_1, \dots, T_n)$ are new variables. The kernel of the map $\varphi : D[Y,T] \to A'$ given by $Y \to y$, $T \to t$ contains h. Since

$$s(Y - y') \equiv dG(y')T \text{ modulo } h$$

and

$$f(Y) - f(y') \equiv \sum_j \frac{\partial f}{\partial Y_j}(y')(Y_j - y'_j)$$

modulo higher order terms in $Y_j - y'_j$ by Taylor's formula we see that for $p = \max_i \deg f_i$ we have

$$s^p f(Y) - s^p f(y') \equiv \sum_j s^{p-1} d\frac{\partial f}{\partial Y_j}(y')G_j(y')T_j + d^2Q = s^{p-1}dP(y')T + d^2Q \tag{C:2.2}$$

modulo h where $Q \in T^2D[T]^r$. This is because $(\partial f/\partial Y)G = (P\mathrm{Id}_r|0)$. We have $f(y') = d^2b$ for some $b \in dD^r$. Then

$$g_i = s^p b_i + s^p T_i + Q_i, \qquad i \in [r] \tag{C:2.3}$$

is in the kernel of φ because $d^2\varphi(g) = d^2g(t) \in \big(h(y,t), f(y)\big) = (0)$.

Set $E = \dfrac{D[Y,T]}{(I,g,h)}$ and let $\psi : E \to A'$ be the map induced by φ. Clearly, v factors through ψ because v is the composed map $B \to B \otimes_A D \cong D[Y]/I \to E \xrightarrow{\psi} A'$.

Note that the $r \times r-$minor s' of $(\partial g/\partial T)$ given by the first $r-$variables T is from $s^{rp} + (T) \subset 1 + (d,T)D[Y,T]$ because $Q \in (T)^2$. Then $U = \big(D[Y,T]/(h,g)\big)_{ss'}$ is smooth over D. We claim that $I \subset (h,g)D[Y,T]_{ss's''}$ for some other $s'' \in 1 + (d,T)D[Y,T]$. Indeed, we have $PI \subset (h,g)D[Y,T]_s$ and so $P\big(y' + s^{-1}dG(y')T\big)I \subset (h,g)D[Y,T]_s$. Since $P\big(y' + s^{-1}dG(y')T\big) \in P(y') + d(T)$ we get $P\big(y' + s^{-1}dG(y')T\big) = ds''$ for some $s'' \in 1 + (d,T)D[Y,T]$. It follows that $s''I \subset (h,g)D[Y,T]_{ss'}$ because d is regular in U, the map $D \to U$ being flat, and so $I \subset (h,g)D[Y,T]_{ss's''}$. Thus $E_{ss's''} \cong U_{s''}$ is a B-algebra which is also standard smooth over D and A.

As $w(s) \equiv 1$ modulo d and $w(s'), w(s'') \equiv 1$ modulo (d,t), $d,t \in mA'$ we see that $w(s), w(s'), w(s'')$ are invertible because A' is local and ψ (thus v) factors through the standard smooth A-algebra $E_{ss's''}$.

Let (A,m) be a Cohen-Macaulay local ring (for example a reduced ring) of dimension 1, $A' = \hat{A}$ the completion of A, $B = A[Y]/I$, $Y = (Y_1, \dots, Y_n)$ a finite type $A-$algebra and $c, e \in \mathbb{N}$. Suppose that there exist $f = (f_1, \dots, f_r)$ in I, a $r \times r-$minor M of the Jacobian matrix $\left(\frac{\partial f}{\partial Y}\right)$, $N \in \big((f):I\big)$ and an A-morphism $v : B \to A/m^{2e+c}$ such that $\big(v(MN)\big) \supset m^e/m^{2e+c}$.

THEOREM C:2.0.19.
Then there exists an A-morphism $v' : B \to \hat{A}$ such that $v' \equiv v$ modulo m^c, that is $v'(Y+I) \equiv v(Y+I)$ modulo m^c.

Proof.
We note that the proof of Theorem C:2.0.2 can work somehow in this case. Let $y' \in A^n$ be an element inducing $v(Y+I)$. Then

$$m^e \subset \big((MN)(y')\big) + m^{2e+c} \subset \big((MN)(y')\big) + m^{3e+2c} \subset \dots$$

by hypothesis. It follows that $m^e \subset \big((MN)(y')\big)$. Since A is Cohen-Macaulay we see that m^e contains a regular element of A and so $(MN)(y')$ must be regular too.

Set $d = (MN)(y')$. Next we follow the proof of Theorem C:2.0.2 with $D = A$, $s = 1$, $P = MN$ and H, G such that .

$$d\mathrm{Id}_n = P(y')\mathrm{Id}_n = G(y')H(y').$$

Let

$$h = Y - y' - dG(y')T,$$

where $T = (T_1, \dots, T_n)$ are new variables. We have

$$f(Y) - f(y') \equiv dP(y')T + d^2Q$$

modulo h where $Q \in T^2A[T]^r$. But $f(y') \in m^{2e+c}A^r \subset d^2m^cA^r$ and we get $f(y') = d^2b$ for some $b \in m^cA^r$. Set $g_i = b_i + T_i + Q_i$, $i \in [r]$ and $E = A[Y,T]/(I,h,g)$. We have an A-morphism $\beta : E \to A/m^c$ given by $(Y,T) \to (y',0)$ because $I(y') \equiv 0$ modulo m^{2e+c}, $h(y',0) = 0$ and $g(0) = b \in m^cA^r$.

As in the proof of Theorem C:2.0.2 we have $E_{s's''} \cong U_{s''}$, where $U = \left(\frac{A[Y,T]}{(g,h)}\right)_{s'}$. This isomorphism follows because d is regular in A and so in U. Consequently, $E_{s's''}$ is smooth over A. Note that β extends to a map $\beta' : E_{s's''} \to A/m^c$. By the Implicit

Function Theorem β' can be lifted to a map $w : E_{s's''} \to \hat{A}$ which coincides with β' modulo m^c. It follows that the composite map v', $B \to E_{s's''} \xrightarrow{w} \hat{A}$ works.

□

Corollary C:2.0.20.

In the assumptions of the above theorem, suppose that (A, m) is excellent Henselian. Then there exists an A-morphism $v'' : B \to A$ such that $v'' \equiv v$ modulo m^c, that is $v''(Y + I) \equiv v(Y + I)$ modulo m^c.

Proof.

An excellent Henselian local ring (A, m) has the property of Artin approximation by [DP2], that is the solutions in A of a system of polynomial equations f over A are dense in the set of the solutions of f in $\hat{A}$. By Theorem C:2.0.19 we get an A-morphism $v' : B \to \hat{A}$ such that $v' \equiv v$ modulo m^c. Then there exists an A-morphism $v'' : B \to A$ such that $v'' \equiv v' \equiv v$ modulo m^c by the property of Artin approximation. □

Theorem C:2.0.21.

Let (A, m) be a Cohen-Macaulay local ring of dimension one, $B = A[Y]/I$, $Y = (Y_1, \dots, Y_n)$ a finite type A-algebra, $e \in \mathbb{N}$ and $f = (f_1, \dots, f_r)$ a system of polynomials from I. Suppose that A is excellent Henselian and there exist a $r \times r$-minor M of the Jacobian matrix $\left(\frac{\partial f}{\partial Y}\right)$, $N \in \big((f) : I\big)$ and $y' \in A^n$ such that $I(y') \equiv 0$ modulo m^e and $\big((NM)(y')\big) \supset m^e$. Then the following statements are equivalent:

1. *there exists $y'' \in A^n$ such that $I(y'') \equiv 0$ modulo m^{3e} and $y'' \equiv y'$ modulo m^e,*
2. *there exists $y \in A^n$ such that $I(y) = 0$ and $y \equiv y'$ modulo m^e.*

For the proof apply the above corollary and Theorem C:2.0.19.

CHAPTER C:3

THE IMPLEMENTATION IN SINGULAR AND SOME EXAMPLES

EXAMPLE C:3.0.1. We would like to compute Example C:2.0.3 in SINGULAR using GND.lib. We quickly recall the example.

Let $a_1, a_2 \in \mathbb{C}$ be two elements algebraically independent over $\mathbb{Q}$ and ρ a root of the polynomial $T^2 + T + 1$ in $\mathbb{C}$. Then $k' = \dfrac{\mathbb{Q}(a_1,a_2)[a_3]}{(a_3^2+a_3+1)} \cong \mathbb{Q}(\rho, a_1, a_2)$. Let $A = \left(\dfrac{\mathbb{Q}[x_1,x_2]}{(x_1^3-x_2^2)}\right)_{(x_1,x_2)}$, $B = \dfrac{A[Y_1,Y_2,Y_3]}{(Y_1^3-Y_2^3)}$, $A' = \dfrac{k'\llbracket x_1,x_2\rrbracket}{(x_1^3-x_2^2)}$ and the map v defined as

$$\begin{tikzcd}
v: & B \arrow[r] & A' \\
 & Y_1 \arrow[r, mapsto] & a_1x_2 \\
 & Y_2 \arrow[r, mapsto] & a_1a_3x_2 \\
 & Y_3 \arrow[r, mapsto] & a_1\sum_{i=0}^{30}\frac{x_1^i}{i!} + a_2x_2\sum_{i=31}^{50}\frac{x_1^i}{i!}
\end{tikzcd}$$

In order to compute this we type the following lines in the SINGULAR terminal.

```
LIB "GND.lib";                               //load the library
ring All = 0,(a1,a2,a3,x1,x2,Y1,Y2,Y3),dp;   //define the ring
int nra = 3;                                 //number of a's
int nrx = 2;                                 //number of x's
int nry = 3;                                 //number of Y's
ideal xid = x1^3-x2^2;                       //define the ideal from A
ideal yid = Y1^3-Y2^3;                       //define the ideal from B
ideal aid = a3^2+a3+1;                       //define the ideal from k'
poly y;
int i;
for(i=0;i<=30;i++)
{
  y = y + a1*x1^i/factorial(i);
}
for(i=31;i<=50;i++)
{
  y = y + a2*x2*x1^i/factorial(i);
}
ideal f = a1*x2,a1*a3*x2,y;                 //define the map v
desingularization(All, nra,nrx,nry,xid,yid,aid,f,"debug");
```

Example C:3.0.2. We continue on the idea of Examples C:2.0.3, C:2.0.15. The bordered matrix H defined above is equal to

$$H = \begin{pmatrix} 2Y_1+Y_2 & Y_1+2Y_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ Y_4 & 2Y_4 & 0 & Y_1+2Y_2 \end{pmatrix}$$

and hence $G = N \cdot G'$ is equal to

$$G = Y_4^2 \cdot \begin{pmatrix} 0 & 0 & Y_1^2+4Y_1Y_2+4Y_2^2 & 0 \\ Y_1+2Y_2 & 0 & -2Y_1^2-5Y_1Y_2-2Y_2^2 & 0 \\ 0 & Y_1^2+4Y_1Y_2+4Y_2^2 & 0 & 0 \\ -2Y_4 & 0 & 3Y_1Y_4 & Y_1+2Y_2 \end{pmatrix}$$

and $s = 1$. This is obvious in our case since $y' = y$ and always $d = P(y)$ because $I(y) = 0$ and $d \equiv P$ modulo I. Using the definition of h in Equation C:2.1, we get that

$$\begin{aligned} h_1 &= Y_1 - (x_2^4) \cdot T_3 - (a_1x_2), \\ h_2 &= Y_2 - \frac{x_2^3}{2a_1a_3+a_1} \cdot T_1 + \frac{a_3x_2^4+2x_2^4}{2a_3+1} \cdot T_3 - (a_1a_3x_2), \\ h_3 &= Y_3 - (x_2^4) \cdot T_2 - \left(\frac{1}{6!}a_1x_1^6 + \frac{1}{5!}a_1x_1^5 + \frac{1}{4!}a_1x_1^4 + \frac{1}{3!}a_1x_1^3 + \frac{1}{2}a_1x_1^2 + \right. \\ & \left. a_1x_1 + a_1 \right), \\ h_4 &= Y_4 + \frac{2x_2^2}{(2a_1a_3+a_1)^3} \cdot T_1 - \frac{3x_2^3}{a_1^2(2a_3+1)^3} \cdot T_3 - \frac{x_2^3}{2a_1a_3+a_1} \cdot T_4 + \frac{1}{2a_1a_3+a_1}. \end{aligned}$$

From Equation C:2.2 we get that

$$\begin{aligned} Q_1 &= \frac{x_2^2}{(2a_1a_3+a_1)^2} \cdot T_1^2 - \frac{3x_2^3}{a_1(2a_3+1)^2} \cdot T_1T_3 + \frac{3a_3^2x_2^4+3a_3x_2^4+3x_2^4}{(2a_3+1)^2} \cdot T_3^2, \\ Q_2 &= -\frac{4x_2}{(2a_1a_3+a_1)^4} \cdot T_1^2 + \frac{12x_2^2}{a_1^3(2a_3+1)^4} \cdot T_1T_3 - \frac{9x_2^3}{a_1^2(2a_3+1)^4} \cdot T_3^2 + \\ & \frac{2x_2^2}{(2a_1a_3+a_1)^2} \cdot T_1T_4 - \frac{3x_2^3}{a_1(2a_3+1)^2} \cdot T_3T_4 \end{aligned}$$

and therefore following the definition of g in Equation C:2.3 we have

$$\begin{aligned} g_1 &= Q_1 + T_1 + (a_1^2a_3^2 + a_1^2a_3 + a_1^2), \\ g_2 &= Q_2 + T_2. \end{aligned}$$

We print now the algorithm's debug output using the line codes from Example C:3.0.1.

```
This is the bordered matrix H:
2*Y1+Y2,Y1+2*Y2,0,0,
0,      0,      1,0,
1,      0,      0,0,
Z,      2*Z,    0,Y1+2*Y2
This is G:
0,                 0,     G[1,3],   0,
Y1*Y4^2+2*Y2*Y4^2,0,     G[2,3],   0,
0,                 G[3,2],0,        0,
-2*Y4^3,           0,     3*Y1*Y4^3,Y1*Y4^2+2*Y2*Y4^2

G[1,3]=Y1^2*Y4^2+4*Y1*Y2*Y4^2+4*Y2^2*Y4^2
G[2,3]=-2*Y1^2*Y4^2-5*Y1*Y2*Y4^2-2*Y2^2*Y4^2
G[3,2]=Y1^2*Y4^2+4*Y1*Y2*Y4^2+4*Y2^2*Y4^2

s = 1
h =
_[1]=Y1+(-x2^4)*T3+(-a1*x2)
_[2]=Y2+(-x2^3)/(2*a1*a3+a1)*T1+(a3*x2^4+2*x2^4)/(2*a3+1)*T3+
(-a1*a3*x2)
_[3]=Y3+(-x2^4)*T2+(-a1*x1^6-6*a1*x1^5-30*a1*x1^4-120*a1*x1^3
-360*a1*x1^2-720*a1*x1-720*a1)/720
_[4]=Y4+(2*x2^2)/(8*a1^3*a3^3+12*a1^3*a3^2+6*a1^3*a3+a1^3)*T1+
(-3*x2^3)/(8*a1^2*a3^3+12*a1^2*a3^2+6*a1^2*a3+a1^2)*T3+
(-x2^3)/(2*a1*a3+a1)*T4-1/(2*a1*a3+a1)

m = 2
QT =
_[1]=(x2^2)/(4*a1^2*a3^2+4*a1^2*a3+a1^2)*T1^2+
(-3*x2^3)/(4*a1*a3^2+4*a1*a3+a1)*T1*T3+
(3*a3^2*x2^4+3*a3*x2^4+3*x2^4)/(4*a3^2+4*a3+1)*T3^2

_[2]=(-4*x2)/(16*a1^4*a3^4+32*a1^4*a3^3+24*a1^4*a3^2+8*a1^4*a3+a1^4)
*T1^2+(12*x2^2)/(16*a1^3*a3^4+32*a1^3*a3^3+24*a1^3*a3^2+8*a1^3*a3
+a1^3)*T1*T3+(-9*x2^3)/(16*a1^2*a3^4+32*a1^2*a3^3+24*a1^2*a3^2
+8*a1^2*a3+a1^2)*T3^2+(2*x2^2)/(4*a1^2*a3^2+4*a1^2*a3+a1^2)*T1*T4
+(-3*x2^3)/(4*a1*a3^2+4*a1*a3+a1)*T3*T4

f =
f[1]=Y1^2+Y1*Y2+Y2^2
f[2]=Y1*Y4+2*Y2*Y4+(-x2^2)

g =
_[1]=(x2^2)/(4*a1^2*a3^2+4*a1^2*a3+a1^2)*T1^2+(-3*x2^3)/(4*a1*a3^2+
4*a1*a3+a1)*T1*T3+(3*a3^2*x2^4+3*a3*x2^4+3*x2^4)/(4*a3^2+4*a3+1)*T3^2
```

```
+T1+(a1^2*a3^2+a1^2*a3+a1^2)
_[2]=(-4*x2)/(16*a1^4*a3^4+32*a1^4*a3^3+24*a1^4*a3^2+8*a1^4*a3+a1^4)
*T1^2+(12*x2^2)/(16*a1^3*a3^4+32*a1^3*a3^3+24*a1^3*a3^2+8*a1^3*
a3+a1^3)*T1*T3+(-9*x2^3)/(16*a1^2*a3^4+32*a1^2*a3^3+24*a1^2*a3^2
+8*a1^2*a3+a1^2)*T3^2+(2*x2^2)/(4*a1^2*a3^2+4*a1^2*a3+a1^2)*T1*T4
+(-3*x2^3)/(4*a1*a3^2+4*a1*a3+a1)*T3*T4+T2
```

Thus the General Neron Desingularization is a localization of $D[Y,T]/(h,g) \cong D[T]/(g)$.

Example C:3.0.3. In the case of Example C:2.0.4 and C:2.0.16 we obtain that the bordered matrix

$$H = \begin{pmatrix} 3Y_1^2 & -3Y_2^2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -6Y_2Y_4 & 0 & -3Y_2^2 \end{pmatrix}$$

and hence $G = N \cdot G'$ is equal to

$$G = Y_4^2 \cdot \begin{pmatrix} 0 & 0 & 9Y_2^4 & 0 \\ -3Y_2^2 & 0 & -3Y_1^2Y_2^2 & 0 \\ 0 & 9Y_2^4 & 0 & 0 \\ 6Y_2Y_4 & 0 & -18Y_1^2Y_2Y_4 & -3Y_2^2 \end{pmatrix}$$

and $s = 1$. Using the definition of h in Equation C:2.1, we get that

$$\begin{aligned}
h_1 &= Y_1 - (x_2^8) \cdot T_3 + (-a_1x_2)\,, \\
h_2 &= Y_2 + \frac{x_2^6}{3a_1^2a_3^2} \cdot T_1 - \frac{x_2^8}{a_3^2} \cdot T_3 - (a_1a_3x_2)\,, \\
h_3 &= Y_3 - (x_2^8) \cdot T_2 - \left(\frac{1}{12!}a_1x_1^{12} + \frac{1}{11!}a_1x_1^{11} + \frac{1}{10!}a_1x_1^{10} + \frac{1}{9!}a_1x_1^9 + \frac{1}{8!}a_1x_1^8 \right. \\
&\quad \left. + \frac{1}{7!}a_1x_1^7 + \frac{1}{6!}a_1x_1^6 + \frac{1}{5!}a_1x_1^5 + \frac{1}{4!}a_1x_1^4 + \frac{1}{3!}a_1x_1^3 + \frac{1}{2}a_1x_1^2 + a_1x_1 + a_1 \right), \\
h_4 &= Y_4 + \frac{2x_2^5}{9a_1^5a_3^5} \cdot T_1 - \frac{2x_2^7}{3a_1^3a_3^5} \cdot T_3 + \frac{x_2^6}{3a_1^2a_3^2} \cdot T_4 + \frac{1}{3a_1^2a_3^2}.
\end{aligned}$$

From Equation C:2.2 we get that

$$\begin{aligned}
Q_1 &= \frac{x_2^{10}}{27a_1^6a_3^6} \cdot T_1^3 - \frac{x_2^{12}}{3a_1^4a_3^6} \cdot T_1^2T_3 + \frac{x_2^{14}}{a_1^2a_3^6} \cdot T_1T_3^2 + \frac{a_3^6x_2^{16} - x_2^{16}}{a_3^6} \cdot T_3^3 - \\
&\quad \frac{x_2^5}{3a_1^3a_3^3} \cdot T_1^2 + \frac{2x_2^7}{a_1a_3^3} \cdot T_1T_3 + \frac{3a_1a_3^3x_2^9 - 3a_1x_2^9}{a_3^3} \cdot T_3^2, \\
Q_2 &= \frac{2ax_2^9}{27a_1^9a_3^9} \cdot T_1^3 - \frac{2x_2^{11}}{3a_1 75a_3^9} \cdot T_1^2T_3 + \frac{2x_2^{13}}{a_1^5a_3^9} \cdot T_1T_3^2 - \frac{2x_2^{15}}{a_1^3a_3^9} \cdot T_3^3 + \\
&\quad \frac{x_2^{10}}{9a_1^6a_3^6} \cdot T_1^2T_4 - \frac{2x_2^{12}}{3a_1^4a_3^6} \cdot T_1T_3T_4 + \frac{x_2^{14}}{a_1^2a_3^6} \cdot T_3^2T_4 - \frac{x_2^4}{3a_1^6a_3^6} \cdot T_1^2 + \\
&\quad \frac{2x_2^6}{a_1^4a_3^6} \cdot T_1T_3 - \frac{3x_2^8}{a_1^2a_3^6} \cdot T_3^2 - \frac{2x_2^5}{3a_1^3a_3^3} \cdot T_1T_4 + \frac{2x_2^7}{a_1a_3^3} \cdot T_3T_4
\end{aligned}$$

and therefore following the definition of g in Equation C:2.3 we have

$$\begin{aligned} g_1 &= Q_1 + T_1, \\ g_2 &= Q_2 + T_2 \end{aligned}$$

To obtain this with SINGULAR , we use the same code lines as in Example C:3.0.1, but we change the last one with

```
desingularization(All, nra,nrx,nry,xid,yid,aid,f,"injective","debug");
```

Doing this, the algorithm will not compute the kernel because of the "`injective`" argument.

EXAMPLE C:3.0.4. We do now the same computations for Examples C:2.0.7 and C:2.0.17. The bordered matrix H is equal to

$$H = \begin{pmatrix} x_2 & -x_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -x_1x_3^2 \end{pmatrix}$$

and hence $G = N \cdot G'$ is equal to

$$G = Y_4^2 \cdot \begin{pmatrix} 0 & 0 & x_1^2x_3^4 & 0 \\ -x_1x_3^4 & 0 & x_1x_2x_3^4 & 0 \\ 0 & x_1^2x_3^4 & 0 & 0 \\ 0 & 0 & 0 & -x_1x_3^2 \end{pmatrix}$$

and $s = 1$. Using the definition of h in Equation C:2.1, we get that

$$\begin{aligned} h_1 &= Y_1 + (x_1^3x_3^6) \cdot T_3 - (a_3x_1), \\ h_2 &= Y_2 - (x_1^2x_3^6) \cdot T_1 + (x_1^2x_2x_3^6) \cdot T_3 - (a_3x_2), \\ h_3 &= Y_3 + (x_1^3x_3^6) \cdot T_2 - \left(\frac{1}{7!}a_1x_3^7 + \frac{1}{6!}a_1x_3^6 + \frac{1}{5!}a_1x_3^5 \right. \\ & \left. \quad + \frac{1}{4!}a_1x_3^4 + \frac{1}{3!}a_1x_3^3 + \frac{1}{2!}a_1x_3^2 + a_1x_3 + a_1 \right), \\ h_4 &= Y_4 + (x_1^2x_3^4) \cdot T_4 + 1. \end{aligned}$$

From Equation C:2.2 we obtain

$$\begin{aligned} Q_1 &= 0, \\ Q_2 &= 0 \end{aligned}$$

and therefore following the definition of g in Equation C:2.3 we have

$$\begin{aligned} g_1 &= T_1 \\ g_2 &= T_2. \end{aligned}$$

To compute this with the SINGULAR library we type the following in the terminal:

```
ring All = 0,(a1,a2,a3,x1,x2,x3,Y1,Y2,Y3),dp;
int nra = 3;
int nrx = 3;
int nry = 3;
ideal xid = x2^3-x3^2,x1^3-x3^2;
ideal yid = Y1^3-Y2^3;
ideal aid = a3^2-a1*a2;
poly y;
int i;
for(i=0;i<=30;i++)
{
  y = y + a1*x3^i/factorial(i);
}
for(i=31;i<=50;i++)
{
  y = y + a2*x3^i/factorial(i);
}
ideal f = a3*x1,a3*x2,y;
desingularization(All, nra,nrx,nry,xid,yid,aid,f,"debug");
```

The algorithm's output is as expected:

```
This is the nice bordered matrix H:
(x2),(-x1),0,0,
0,   0,    1,0,
1,   0,    0,0,
0,   0,    0,(-x1*x3^2)
This is G:
0,              0,               (x1^2*x3^4)*Y4^2, 0,
(-x1*x3^4)*Y4^2,0,               (x1*x2*x3^4)*Y4^2,0,
0,              (x1^2*x3^4)*Y4^2,0,                0,
0,              0,               0,                (-x1*x3^2)*Y4^2

s = 1

h =
h[1]=Y1+(x1^3*x3^6)*T3+(-a3*x1)
h[2]=Y2+(-x1^2*x3^6)*T1+(x1^2*x2*x3^6)*T3+(-a3*x2)
h[3]=Y3+(x1^3*x3^6)*T2+(-a1*x3^7-7*a1*x3^6-42*a1*x3^5-210*a1*x3^4
     -840*a1*x3^3-2520*a1*x3^2-5040*a1*x3-5040*a1)/5040
h[4]=Y4+(-x1^2*x3^4)*T4+1

m = 1

QT =
QT[1]=0
QT[2]=0
```

```
f =
f[1]=(x2)*Y1+(-x1)*Y2
f[2]=(x1*x3^2)*Y4+(-x1*x3^2)


g
_[1]=T1
_[2]=T2
```

Thus the General Neron Desingularization is a localization of $D[Y, T_3, T_4]/(h) \cong D[T_3, T_4]$.

EXAMPLE C:3.0.5. We do now the same computations for Example C:2.0.18. In this example, the computations are much more complicated. The output is unfortunately too big but we will try our best to describe the result.

The bordered matrix H is equal to

$$H = \begin{pmatrix} 0 & x_2 \cdot Y_3 & x_2 \cdot Y_2 & -2 \cdot Y_4 & 0 \\ x_1^2 \cdot Y_2 & x_1^2 \cdot Y_1 & -2 \cdot Y_3 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4Y_4Y_5 & 4Y_3Y_5 & 4Y_3Y_4 \end{pmatrix}$$

and hence $G = N \cdot G'$ is equal to

$$G = Y_5^2 \cdot \begin{pmatrix} 0 & 0 & 0 & 16Y_3^2Y_4^2 & 0 \\ 0 & 0 & 16Y_3^2Y_4^2 & 0 & 0 \\ 0 & -8Y_3Y_4^2 & 8x_1^2 \cdot Y_1Y_3Y_4^2 & 8x_1^2 \cdot Y_2Y_3Y_4^2 & 0 \\ -8Y_3^2Y_4 & -4x_2 \cdot Y_2Y_3Y_4 & G[4,3] & 4x_1^2x_2 \cdot Y_2^2Y_3Y_4 & 0 \\ 8Y_3^2Y_5 & x_2 \cdot Y_2Y_3Y_5 + 2Y_4^2Y_5 & G[5,3] & G[5,4] & 4Y_3Y_4 \end{pmatrix},$$

where

$$\begin{aligned} G[4,3] &= \quad 4x_1^2x_2 \cdot Y_1Y_2Y_3Y_4 + 2x_2 \cdot Y_3^3Y_4, \\ G[5,3] &= -4x_1^2x_2 \cdot Y_1Y_2Y_3Y_5 - 2x_2 \cdot Y_3^3Y_5 - 2x_1^2 \cdot Y_1Y_4^2Y_5 \text{ and} \\ G[5,4] &= -4x_1^2x_2 \cdot Y_2^2Y_3Y_5 - 2x_1^2 \cdot Y_2Y_4^2Y_5 \end{aligned}$$

and

$$\begin{aligned} s = \; & a_1^8a_2^2x_2^{12} - 2a_1^5a_2^4x_2^{13} + a_1^2a_2^6x_2^{14} - 2a_1^6a_2^3x_2^{12} + 2a_1^3a_2^5x_2^{13} - a_1^4a_2^4x_2^{12} + 2a_1a_2^6x_2^{13} + \\ & 2a_1^8a_2x_2^{10} - 4a_1^5a_2^3x_2^{11} + 4a_1^2a_2^5x_2^{12} - 4a_1^6a_2^2x_2^{10} + 4a_1^3a_2^4x_2^{11} + a_2^6x_2^{12} + 2a_1a_2^5x_2^{11} + \\ & a_1^8x_2^8 - 2a_1^5a_2^2x_2^9 + 3a_1^2a_2^4x_2^{10} - 2a_1^6a_2x_2^8 + 2a_1^3a_2^3x_2^9 + a_1^4a_2^2x_2^8 - 2a_1^4a_2x_2^6 + \\ & 2a_1a_2^3x_2^7 + 2a_1^2a_2^2x_2^6 + 2a_2^3x_2^6 - 2a_1^4x_2^4 + 2a_1a_2^2x_2^5 + 2a_1^2a_2x_2^4 + 1 \end{aligned}$$

Using the definition of h in Equation C:2.1, we get that

$$\begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{pmatrix} = s \cdot \begin{pmatrix} Y_1 - y_1' \\ Y_2 - y_2' \\ Y_3 - y_3' \\ Y_4 - y_4' \\ Y_5 - y_5' \end{pmatrix} - x_2^4 G(y') \cdot \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{pmatrix},$$

where

$$\begin{array}{rcl}
y_1' & = & -18a_1^2a_2^5x_2^{12} + 5a_1^3a_2^4x_2^{11} + 7a_2^6x_2^{12} - 18a_1a_2^5x_2^{11} + 15a_1^2a_2^4x_2^{10} - 4a_1^3a_2^3x_2^9 - \\
 & & 6a_2^5x_2^{10} + 15a_1a_2^4x_2^9 - 12a_1^2a_2^3x_2^8 + 3a_1^3a_2^2x_2^7 + 5a_2^4x_2^8 - 12a_1a_2^3x_2^7 + 9a_1^2a_2^2x_2^6 - \\
 & & 2a_1^3a_2x_2^5 - 4a_2^3x_2^6 + 9a_1a_2^2x_2^5 - 6a_1^2a_2x_2^4 + a_1^3x_2^3 + 3a_2^2x_2^4 - 6a_1a_2x_2^3 + 3a_1^2x_2^2 - \\
 & & 2a_2x_2^2 + 3a_1x_2 + 1 \\
y_2' & = & a_1^{12}x_2^{12} + 2a_1^{10}a_2x_2^{12} - a_1^{11}x_2^{11} + a_1^8a_2^2x_2^{12} - 2a_1^9a_2x_2^{11} + a_1^{10}x_2^{10} - a_1^7a_2^2x_2^{11} + \\
 & & 2a_1^8a_2x_2^{10} - a_1^9x_2^9 + a_1^6a_2^2x_2^{10} - 2a_1^7a_2x_2^9 + a_1^8x_2^8 - a_1^5a_2^2x_2^9 + 2a_1^6a_2x_2^8 - a_1^7x_2^7 + \\
 & & a_1^4a_2^2x_2^8 - 2a_1^5a_2x_2^7 + a_1^6x_2^6 - a_1^3a_2^2x_2^7 + 2a_1^4a_2x_2^6 - a_1^5x_2^5 + a_1^2a_2^2x_2^6 - 2a_1^3a_2x_2^5 + \\
 & & a_1^4x_2^4 - a_1a_2^2x_2^5 + 2a_1^2a_2x_2^4 - a_1^3x_2^3 + a_2^2x_2^4 - 2a_1a_2x_2^3 + a_1^2x_2^2 + 2a_2x_2^2 - a_1x_2 + \\
 & & 1 \\
y_3' & = & a_1x_2^2 + x_2 \\
y_4' & = & a_2x_2^3 + x_2 \\
y_5' & = & \dfrac{a_2^2}{4}x_2^4 - \dfrac{a_1^3 + a_1a_2}{4}x_2^3 + \dfrac{a_1^2 - a_2}{4}x_2^2 - \dfrac{a_1}{4}x_2 + \dfrac{1}{4}.
\end{array}$$

However, the output is too big to be printed. Following the idea in the above examples, we compute Q and g. This is even bigger than h so we print the numerators and denominators of the coefficients just till degree 10 in the x_i's. However in some cases even the terms till degree 10 will be too many to write down and hence we will print just the first terms and "…" .

As a small remark, Q_3 contains also terms in degree 3 in the T_i but the numerator of the coefficients have power greater than 10 and therefore they do not appear in our short-cutting.

$$\begin{aligned}
Q_1 = {} & \frac{3a_1^2x_1^2x_2^8 - 2a_1x_1^2x_2^7 + 4a_2x_1^2x_2^8 + x_1^2x_2^6}{4a_1a_2^2x_2^5 + 8a_1a_2x_2^3 + 4a_1x_2 + 4a_2^2x_2^4 + 8a_2x_2^2 + 4} \cdot T_1T_4 - \frac{x_2^6}{4a_2^2x_2^4 + 8a_2x_2^2 + 4} \cdot T_1^2 \\
& + \frac{-a_1^4x_2^{10} + a_1^3x_2^9 - 2a_1^2a_2x_2^{10} - a_1^2x_2^8 + 2a_1a_2x_2^9 + a_1x_2^7 - a_2^2x_2^{10} - 2a_2x_2^8 - x_2^6}{4a_1a_2^2x_2^5 + 8a_1a_2x_2^3 + 4a_1x_2 + 4a_2^2x_2^4 + 8a_2x_2^2 + 4} \cdot T_1T_2 \\
& + \frac{-5a_1^4x_2^{10} + 4a_1^3x_2^9 - 12a_1^2a_2x_2^{10} - 3a_1^2x_2^8 + 8a_1a_2x_2^9 + 2a_1x_2^7 - 6a_2^2x_2^{10} - 4a_2x_2^8 - x_2^6}{16a_1^2a_2^2x_2^6 + 32a_1^2a_2x_2^4 + 16a_1^2x_2^2 + 32a_1a_2^2x_2^5 + 64a_1a_2x_2^3 + 32a_1x_2 + 16a_2^2x_2^4 + 32a_2x_2^2 + 16} \cdot T_2^2 \\
& + \frac{a_1^2x_1^2x_2^8 + 2a_1^2x_2^{10} + 2a_1x_1^2x_2^7 + 4a_1x_2^9 + x_1^2x_2^6 + 2x_2^8}{4a_1a_2^2x_2^5 + 8a_1a_2x_2^3 + 4a_1x_2 + 4a_2^2x_2^4 + 8a_2x_2^2 + 4} \cdot T_1T_3 \\
& + \frac{a_1x_1^2x_2^7 - 2a_1x_2^9 + 2a_2x_1^2x_2^8 - 4a_2x_2^{10} + x_1^2x_2^6 - 2x_2^8}{8a_1^2a_2^2x_2^6 + 16a_1^2a_2x_2^4 + 8a_1^2x_2^2 + 16a_1a_2^2x_2^5 + 32a_1a_2x_2^3 + 16a_1x_2 + 8a_2^2x_2^4 + 16a_2x_2^2 + 8} \cdot T_2T_3 \\
& + \frac{-x_1^4x_2^6 + 4x_1^2x_2^8 - 4x_2^{10}}{16a_1^2a_2^2x_2^6 + 32a_1^2a_2x_2^4 + 16a_1^2x_2^2 + 32a_1a_2^2x_2^5 + 64a_1a_2x_2^3 + 32a_1x_2 + 16a_2^2x_2^4 + 32a_2x_2^2 + 16} \cdot T_3^2 \\
& + \frac{6a_1^2x_1^2x_2^8 - 3a_1x_1^2x_2^7 + 6a_2x_1^2x_2^8 + x_1^2x_2^6}{8a_1^2a_2^2x_2^6 + 16a_1^2a_2x_2^4 + 8a_1^2x_2^2 + 16a_1a_2^2x_2^5 + 32a_1a_2x_2^3 + 16a_1x_2 + 8a_2^2x_2^4 + 16a_2x_2^2 + 8} \cdot T_2T_4 \\
& + \frac{-x_1^4x_2^6 + 2x_1^2x_2^8}{8a_1^2a_2^2x_2^6 + 16a_1^2a_2x_2^4 + 8a_1^2x_2^2 + 16a_1a_2^2x_2^5 + 32a_1a_2x_2^3 + 16a_1x_2 + 8a_2^2x_2^4 + 16a_2x_2^2 + 8} \cdot T_3T_4
\end{aligned}$$

$$+\frac{-x_1^4x_2^6}{16a_1^2a_2^2x_2^6+32a_1^2a_2x_2^4+16a_1^2x_2^2+32a_1a_2^2x_2^5+64a_1a_2x_2^3+32a_1x_2+16a_2^2x_2^4+32a_2x_2^2+16}\cdot\ T_4^2$$

$$\begin{aligned}Q_2 = \quad & \frac{-x_2^6}{4a_1^2x_2^2+8a_1x_2+4}\cdot T_2^2+\frac{3a_1^2x_1^2x_2^8+3a_1x_1^2x_2^7-2a_2x_1^2x_2^8+x_1^2x_2^6}{2a_1^2x_2^2+4a_1x_2+2}\cdot T_2T_3+\frac{-x_1^4x_2^6}{4a_1^2x_2^2+8a_1x_2+4}\cdot\ T_3^2\\ & +\frac{a_1^2x_1^2x_2^8-a_1x_1^2x_2^7+2a_2x_1^2x_2^8+x_1^2x_2^6}{2a_1^2x_2^2+4a_1x_2+2}\cdot T_2T_4+\frac{-x_1^4x_2^6+2x_1^2x_2^8}{2a_1^2x_2^2+4a_1x_2+2}\cdot T_3T_4+\frac{-x_1^4x_2^6}{4a_1^2x_2^2+8a_1x_2+4}\cdot\ T_4^2\end{aligned}$$

$$\begin{aligned}Q_3 = \quad & \frac{2a_1^3x_2^9-4a_1^2a_2x_2^{10}-2a_1^2x_2^8+2a_1a_2x_2^9+2a_1x_2^7-2a_2x_2^8-2x_2^6}{4a_1a_2^3x_2^7+12a_1a_2^2x_2^5+12a_1a_2x_2^3+4a_1x_2+4a_2^3x_2^6+12a_2^2x_2^4+12a_2x_2^2+4}\cdot T_1T_2\\ & +\frac{7a_1^3x_2^9-28a_1^2a_2x_2^{10}-7a_1^2x_2^8+21a_1a_2x_2^9+7a_1x_2^7-21a_2^2x_2^{10}-21a_2x_2^8-7x_2^6}{...+48a_1^2x_2^2+48a_1a_2^3x_2^7+144a_1a_2^2x_2^5+144a_1a_2x_2^3+48a_1x_2+16a_2^3x_2^6+48a_2^2x_2^4+48a_2x_2^2+16}\cdot\ T_2^2\\ & +\frac{2a_1^2x_1^2x_2^8+2a_1^2x_2^{10}+4a_1x_1^2x_2^7+4a_1x_2^9-2a_2x_1^2x_2^8-2a_2x_2^{10}+2x_1^2x_2^6+2x_2^8}{4a_1a_2^3x_2^7+12a_1a_2^2x_2^5+12a_1a_2x_2^3+4a_1x_2+4a_2^3x_2^6+12a_2^2x_2^4+12a_2x_2^2+4}\cdot T_1T_3\\ & +\frac{7a_1^2x_1^2x_2^8+4a_1^2x_2^{10}+14a_1x_1^2x_2^7+8a_1x_2^9+7a_2x_1^2x_2^8+4a_2x_2^{10}+7x_1^2x_2^6+4x_2^8}{...+24a_1^2x_2^2+24a_1a_2^3x_2^7+72a_1a_2^2x_2^5+72a_1a_2x_2^3+24a_1x_2+8a_2^3x_2^6+24a_2^2x_2^4+24a_2x_2^2+8}\cdot T_2T_3\\ & +\frac{-7x_1^4x_2^6-8x_1^2x_2^8-4x_2^{10}}{...+48a_1^2x_2^2+48a_1a_2^3x_2^7+144a_1a_2^2x_2^5+144a_1a_2x_2^3+48a_1x_2+16a_2^3x_2^6+48a_2^2x_2^4+48a_2x_2^2+16}\cdot\ T_3^2\\ & +\frac{6a_1^2x_1^2x_2^8-4a_1x_1^2x_2^7+6a_2x_1^2x_2^8+2x_1^2x_2^6}{4a_1a_2^3x_2^7+12a_1a_2^2x_2^5+12a_1a_2x_2^3+4a_1x_2+4a_2^3x_2^6+12a_2^2x_2^4+12a_2x_2^2+4}\cdot T_1T_4\\ & +\frac{21a_1^2x_1^2x_2^8-14a_1x_1^2x_2^7+35a_2x_1^2x_2^8+7x_1^2x_2^6}{...+24a_1^2x_2^2+24a_1a_2^3x_2^7+72a_1a_2^2x_2^5+72a_1a_2x_2^3+24a_1x_2+8a_2^3x_2^6+24a_2^2x_2^4+24a_2x_2^2+8}\cdot T_2T_4\\ & +\frac{-7x_1^4x_2^6-4x_1^2x_2^8}{...+24a_1^2x_2^2+24a_1a_2^3x_2^7+72a_1a_2^2x_2^5+72a_1a_2x_2^3+24a_1x_2+8a_2^3x_2^6+24a_2^2x_2^4+24a_2x_2^2+8}\cdot T_3T_4\\ & +\frac{-7x_1^4x_2^6}{...+48a_1^2x_2^2+48a_1a_2^3x_2^7+144a_1a_2^2x_2^5+144a_1a_2x_2^3+48a_1x_2+16a_2^3x_2^6+48a_2^2x_2^4+48a_2x_2^2+16}\cdot\ T_4^2\\ & +\frac{a_1^4x_2^{10}-a_1^2a_2x_2^{10}-a_2^2x_2^{10}+a_2x_2^8-x_2^6}{4a_2^3x_2^6+12a_2^2x_2^4+12a_2x_2^2+4}\cdot T_1^2+\frac{-x_2^6}{2a_2^2x_2^4+4a_2x_2^2+2}\cdot T_1T_5\\ & +\frac{-3a_2^2x_2^{10}-6a_2x_2^8-3x_2^6}{4a_1^2a_2^2x_2^6+8a_1^2a_2x_2^4+4a_1^2x_2^2+8a_1a_2^2x_2^5+16a_1a_2x_2^3+8a_1x_2+4a_2^2x_2^4+8a_2x_2^2+4}\cdot T_2T_5\\ & +\frac{9a_1^2x_1^2x_2^8+6a_1^2x_2^{10}+9a_1x_1^2x_2^7+6a_1x_2^9+3x_1^2x_2^6+2x_2^8}{4a_1^2a_2^2x_2^6+8a_1^2a_2x_2^4+4a_1^2x_2^2+8a_1a_2^2x_2^5+16a_1a_2x_2^3+8a_1x_2+4a_2^2x_2^4+8a_2x_2^2+4}\cdot T_3T_5\\ & +\frac{3a_1^2x_1^2x_2^8-3a_1x_1^2x_2^7+12a_2x_1^2x_2^8+3x_1^2x_2^6}{4a_1^2a_2^2x_2^6+8a_1^2a_2x_2^4+4a_1^2x_2^2+8a_1a_2^2x_2^5+16a_1a_2x_2^3+8a_1x_2+4a_2^2x_2^4+8a_2x_2^2+4}\cdot T_4T_5\end{aligned}$$

The denominators of the coefficients from Q are invertible in D because they follow from θ_3' and θ_4' which are invertible in D. Thus $Q \in D[T_1, \dots, T_5]$. Having Q_i we obtain g_i:

$$\begin{array}{rll}
g_1 = Q_1 + & \left(\ldots-6a_1^4x_2^4+30a_1^3a_2^3x_2^9+45a_1^2a_2^4x_2^{10}+6a_1^2a_2^2x_2^6+6a_1^2a_2x_2^4+6a_1a_2^3x_2^7+6a_1a_2^2x_2^5+6a_2^3x_2^6+1\right) & \cdot T_1 \\
g_2 = Q_2 + & \left(\ldots-6a_1^4x_2^4+30a_1^3a_2^3x_2^9+45a_1^2a_2^4x_2^{10}+6a_1^2a_2^2x_2^6+6a_1^2a_2x_2^4+6a_1a_2^3x_2^7+6a_1a_2^2x_2^5+6a_2^3x_2^6+1\right) & \cdot T_2 \\
+ & \left(\ldots+6a_1^6x_2^4-30a_1^5a_2^3x_2^9-45a_1^4a_2^4x_2^{10}-6a_1^4a_2^2x_2^6-6a_1^4a_2x_2^4-6a_1^3a_2^3x_2^7-6a_1^3a_2^2x_2^5-6a_1^2a_2^3x_2^6-a_1^2\right) & \\
g_3 = Q_3 + & \left(\ldots-6a_1^4x_2^4+30a_1^3a_2^3x_2^9+45a_1^2a_2^4x_2^{10}+6a_1^2a_2^2x_2^6+6a_1^2a_2x_2^4+6a_1a_2^3x_2^7+6a_1a_2^2x_2^5+6a_2^3x_2^6+1\right) & \cdot T_3 \\
+ & \left(\ldots+24a_1^2a_2^5x_2^{10}+18a_1^2a_2^4x_2^8+a_1^2a_2^2x_2^4+a_1^2a_2x_2^2+12a_1a_2^5x_2^9+a_1a_2^3x_2^5+a_1a_2^2x_2^3+6a_2^6x_2^{10}+a_2^3x_2^4\right). &
\end{array}$$

The General Neron Desingularization is a localization of $D[Y, T]/(h, g)$. To compute this example with our SINGULAR library, we will need a function

```
invp(poly p, int bound,string param,string variab)
```

which computes computes the inverse of p till order bound in $\mathbb{Q}(\texttt{param})[\texttt{variab}]$.

The input for this example is the following:

```
ring All = 0,(a1,a2,x1,x2,Y1,Y2,Y3,Y4),dp;
int nra = 2;
int nrx = 2;
int nry = 4;
ideal xid = x1^2-x2^3;
ideal yid = Y3^2-x1^2*Y1*Y2,Y4^2-x2*Y2*Y3;
ideal aid = 0;
poly y1,y2,y3,y4;
y3 = 1+a1*x2;
y4 = 1+a2*x2^2;
string as,xs;
if(nra != 0)
{
  as = string(var(1));
  for( int i=2;i<=nra;i++)
  {
    as = as+","+string(var(i));
  }
}
if(nrx!=0)
{
  xs = string(var(nra+1));
  for(int i=nra+2;i<=nra+nrx;i++)
  {
    xs = xs+","+string(var(i));
  }
}
y1 = y3^3*invp(y4^2,12,as,xs);
y2 = y4^2*invp(y3,12,as,xs);
y3 = x2*y3;
y4 = x2*y4;
ideal f = y1,y2,y3,y4;
desingularization(All, nra,nrx,nry,xid,yid,aid,f,"injective","debug");
```

REMARK C:3.0.6. If we restrict our present algorithm to the case when A' is the completion of A then we might get a faster algorithm using the idea of the proof of Theorem C:2.0.19. This algorithm could be useful in the arc frame.

APPENDIX

Examples used in Part A

Examples for ALL vs JUST strategies

In the following tables we print the ideals corresponding to the examples presented when comparing the ALL vs JUST timings in Section A:1.3.

	Generators for ideal
Example A:1	$17 \cdot x^2y,$ $22 \cdot y^3z + 3 \cdot x^2z^2 + 28 \cdot y^2z^2 + 9 \cdot yz^2 + 83 \cdot x^2 + 13 \cdot yz,$ $66 \cdot y^3z + 63 \cdot xyz^2 + 85 \cdot z^3$
Example A:2	$27 \cdot xyz + 13 \cdot x^2 + 89 \cdot y^2 + 42 \cdot xz,$ $35 \cdot x^3 + 68 \cdot xy,$ $44 \cdot x^3 + 13 \cdot y^3 + 81 \cdot y^2z + 4 \cdot yz^2$
Example A:3	$98 \cdot x^3y + 45 \cdot yz^3,$ $16 \cdot xy^3 + 50 \cdot x^2y + 45 \cdot y^2 + 82 \cdot z^2,$ $9 \cdot x^3y + 49 \cdot x^2yz + 61 \cdot y^2z + 52 \cdot z^3$
Example A:4	$25 \cdot y^3,$ $60 \cdot y^3 + 12 \cdot xyz + 54 \cdot y^2z + 98 \cdot yz^2 + 35 \cdot x^2 + 88 \cdot xy + 19 \cdot z^2,$ $87 \cdot x^2y + 96 \cdot x^2$
Example A:5	$76 \cdot y^2z + 61 \cdot y^2 + 51 \cdot yz + 19 \cdot z^2,$ $31 \cdot x^3 + 3 \cdot xy^2 + 70 \cdot y^2z,$ $84 \cdot x^3 + 30 \cdot x^2z + 44 \cdot xz$
Example A:6	$19 \cdot x^3 + 2 \cdot xy + 29 \cdot y^2,$ $9 \cdot xy^2 + 42 \cdot y^3 + 2 \cdot yz^2,$ $54 \cdot xy^2 + 83 \cdot x^2z + 98 \cdot xy + 78 \cdot yz$
Example A:7	$72 \cdot x^3y + 50 \cdot x^2yz,$ $64 \cdot x^3z + 30 \cdot x^2yz + 74 \cdot x^3 + 38 \cdot xy^2 + 74 \cdot z^3,$ $76 \cdot x^2y^2 + 13 \cdot y^2 + 40 \cdot z^2$
Example A:8	$27 \cdot xz^3 + 87 \cdot y^2z + 3 \cdot z^2,$ $67 \cdot x^3z + 42 \cdot y^3z + 67 \cdot x^2y + 90 \cdot yz^2 + 73 \cdot xy,$ $38 \cdot x^3y + 69 \cdot x^2yz$
Example A:9	$21 \cdot x^3y + 18 \cdot x^3z + 45 \cdot x^2yz + 100 \cdot xy^2z + 43 \cdot yz,$ $85 \cdot x^2y + 93 \cdot xy^2,$ $14 \cdot y^2z^2 + 6 \cdot x^2y + 91 \cdot z^2$
Example A:10	$69 \cdot y^2,$ $47 \cdot x^3y + 82 \cdot xyz^2 + 74 \cdot yz^3 + 55 \cdot xyz + 96 \cdot xz^2 + 46 \cdot x^2,$ $16 \cdot xy^2z + 17 \cdot z^4 + 36 \cdot yz^2$

Table 3.1: Examples for ALL Strategy

	Generators for ideal
Example B:1	$6 \cdot x^3z + 29 \cdot x^2z + 42 \cdot xy,$ $x^3z + 47 \cdot x^2yz + 28 \cdot xz^2 + 46 \cdot x^2,$ $96 \cdot z^3$
Example B:2	$9 \cdot x^2y^2 + 51 \cdot x^3z + 10 \cdot z^3 + 28 \cdot x^2 + 7 \cdot y^2,$ $43 \cdot x^3y + 3 \cdot x^2z^2 + 86 \cdot xyz^2 + 24 \cdot z^4 + 67 \cdot x^2z + 68 \cdot yz^2 + 27 \cdot xy,$ $23 \cdot xz^2$
Example B:3	$50 \cdot xz^3 + 49 \cdot yz^2 + 15 \cdot z^2,$ $2 \cdot x^3y + 16 \cdot y^3z + 74 \cdot y^3 + 53 \cdot x^2,$ $4 \cdot xy^2$
Example B:4	$57 \cdot xyz^2 + 32 \cdot y^3 + 26 \cdot yz^2 + 24 \cdot z^2,$ $27 \cdot y^4 + 33 \cdot y^3z + 94 \cdot z^2,$ $52 \cdot x^2z$
Example B:5	$91 \cdot y^4 + 20 \cdot x^3z + 34 \cdot x^2,$ $38 \cdot xyz^2 + 18 \cdot x^3 + 95 \cdot x^2z + 82 \cdot yz,$ $98 \cdot y^3$

Table 3.2: Examples for JUST Strategy

Example 18

Here we enter the details of *Example 18* presented in Section A:1.4. This example used to run out of memory causing the crash of Singular . With the newly implemented strategies over $\mathbb{Z}[x]$, the example finishes instantly.

We consider over $\mathbb{Z}[x, y, z]$ the `dp` ordering and the ideal `I` generated by the following 70 polynomials $f_1, \ldots, f_{70}$.

$$
\begin{aligned}
f_1 &= 42 \cdot x^3z + y^2z - yz + 11 \cdot y \\
f_2 &= y^3z^2 - y^2z^2 + 11 \cdot y^2z + 484 \\
f_3 &= y^4z - y^3z - 10648 \cdot x^3 + 11 \cdot y^3 - 44 \cdot y^2 + 44 \cdot y \\
f_4 &= x^3yz^2 - 2 \\
f_5 &= 11 \cdot x^3y^2z + 484 \cdot x^3 + 2 \cdot y^2 - 2 \cdot y \\
f_6 &= 117128 \cdot x^6 - 121 \cdot x^3y^3 + 968 \cdot x^3y^2 - 968 \cdot x^3y + 2 \cdot y^4 - 4 \cdot y^3 + 2 \cdot y^2 \\
f_7 &= 121 \cdot x^6z^3 + yz - z + 11 \\
f_8 &= 2178 \cdot x^2y^2z^3 - 1452 \cdot x^2yz^3 + 15972 \cdot x^2yz^2 \\
f_9 &= 1452 \cdot x^2y^3z^2 - 1452 \cdot x^2y^2z^2 + 7986 \cdot x^2y^2z \\
f_{10} &= -726 \cdot x^3y^2z^2 + 484 \cdot x^3yz^2 + y^4z^2 - 5324 \cdot x^3yz - y^3z^2 + 22 \cdot y^3z - 11 \cdot y^2z \\
&\quad +121 \cdot y^2 \\
f_{11} &= 2904 \cdot x^2y^3z^2 - 2178 \cdot x^2y^2z^2 + 23958 \cdot x^2y^2z + 351384 \cdot x^2 \\
f_{12} &= 726 \cdot x^2y^4z - 726 \cdot x^2y^3z + 7730448 \cdot x^5 + 31944 \cdot x^2y^2 - 31944 \cdot x^2y \\
f_{13} &= -968 \cdot x^3y^3z + 726 \cdot x^3y^2z - 2 \cdot y^5z - 7986 \cdot x^3y^2 + 4 \cdot y^4z + 21296 \cdot x^3y \\
&\quad -22 \cdot y^4 - 2 \cdot y^3z - 10648 \cdot x^3 + 110 \cdot y^3 - 132 \cdot y^2 + 44 \cdot y \\
f_{14} &= 95832 \cdot x^2y^2z^2 - 63888 \cdot x^2yz^2 + 702768 \cdot x^2yz \\
f_{15} &= 63888 \cdot x^2y^3z - 63888 \cdot x^2y^2z + 351384 \cdot x^2y^2 \\
f_{16} &= -5 \cdot y^6z^2 + 9 \cdot y^5z^2 - 88 \cdot y^5z - 4 \cdot y^4z^2 + 253 \cdot y^4z - 363 \cdot y^4 - 264 \cdot y^3z \\
&\quad +968 \cdot y^3 + 88 \cdot y^2z - 484 \cdot y^2 \\
f_{17} &= 726 \cdot x^5z^3 - 6 \cdot x^2y^2z^3 + 3 \cdot x^2yz^3 - 33 \cdot x^2yz^2 \\
f_{18} &= 726 \cdot x^5yz^2 - 3 \cdot x^2y^3z^2 + 3 \cdot x^2y^2z^2 \\
f_{19} &= -242 \cdot x^6z^2 + 3 \cdot x^3y^2z^2 - x^3yz^2 + 22 \cdot x^3yz \\
f_{20} &= -9 \cdot x^2y^3z^4 + 6 \cdot x^2y^2z^4 - 66 \cdot x^2y^2z^3 \\
f_{21} &= -6 \cdot x^2y^4z^3 + 6 \cdot x^2y^3z^3 - 33 \cdot x^2y^3z^2 \\
f_{22} &= 4 \cdot x^3y^3z^3 - 2 \cdot x^3y^2z^3 + 33 \cdot x^3y^2z^2 \\
f_{23} &= -12 \cdot x^2y^4z^3 + 9 \cdot x^2y^3z^3 - 31944 \cdot x^5z^2 - 99 \cdot x^2y^3z^2 + 264 \cdot x^2y^2z^2 - 132 \cdot x^2yz^2 \\
f_{24} &= -3 \cdot x^2y^5z^2 + 3 \cdot x^2y^4z^2 - 63888 \cdot x^5yz \\
f_{25} &= 7 \cdot x^3y^4z^2 - 5 \cdot x^3y^3z^2 + 66 \cdot x^3y^3z - 176 \cdot x^3y^2z + 88 \cdot x^3yz \\
f_{26} &= 15972 \cdot x^5yz^2 - 66 \cdot x^2y^3z^2 + 33 \cdot x^2y^2z^2 - 363 \cdot x^2y^2z - 15972 \cdot x^2 \\
f_{27} &= -33 \cdot x^2y^4z + 33 \cdot x^2y^3z - 351384 \cdot x^5 - 1452 \cdot x^2y^2 + 1452 \cdot x^2y
\end{aligned}
$$

$$
\begin{aligned}
f_{28} &= -5324 \cdot x^6yz + 11 \cdot x^3y^2z + 121 \cdot x^3y^2 - 968 \cdot x^3y + 484 \cdot x^3 - 4 \cdot y^3 + 6 \cdot y^2 - 2 \cdot y \\
f_{29} &= -99 \cdot x^2y^4z^3 + 66 \cdot x^2y^3z^3 - 726 \cdot x^2y^3z^2 - 4356 \cdot x^2y^2z^2 + 2904 \cdot x^2yz^2 \\
&\quad -31944 \cdot x^2yz \\
f_{30} &= -66 \cdot x^2y^5z^2 + 66 \cdot x^2y^4z^2 - 363 \cdot x^2y^4z - 2904 \cdot x^2y^3z + 2904 \cdot x^2y^2z \\
&\quad -15972 \cdot x^2y^2 \\
f_{31} &= -11 \cdot x^3y^4z^2 + 22 \cdot x^3y^3z^2 - 8 \cdot y^4z + 12 \cdot y^3z - 44 \cdot y^3 - 4 \cdot y^2z + 22 \cdot y^2 \\
f_{32} &= -132 \cdot x^2y^5z^2 + 99 \cdot x^2y^4z^2 - 702768 \cdot x^5yz - 1089 \cdot x^2y^4z - 2904 \cdot x^2y^3z \\
&\quad +2904 \cdot x^2y^2z - 47916 \cdot x^2y^2 \\
f_{33} &= -33 \cdot x^2y^6z + 33 \cdot x^2y^5z - 351384 \cdot x^5y^2 - 1452 \cdot x^2y^4 + 1452 \cdot x^2y^3 \\
f_{34} &= 22 \cdot x^3y^5z - 11 \cdot x^3y^4z + 363 \cdot x^3y^4 - 968 \cdot x^3y^3 + 484 \cdot x^3y^2 - 4 \cdot y^5 + 6 \cdot y^4 \\
&\quad -2 \cdot y^3 \\
f_{35} &= 33 \cdot x^5y^2z^3 - 1452 \cdot x^5z^2 + 12 \cdot x^2y^2z^2 - 6 \cdot x^2yz^2 \\
f_{36} &= -33 \cdot x^5y^3z^2 - 2904 \cdot x^5yz \\
f_{37} &= -33 \cdot x^6y^2z^2 - 8 \cdot x^3y^2z + 4 \cdot x^3yz \\
f_{38} &= -263538 \cdot x^5y^2z + 726 \cdot x^2y^4z - 363 \cdot x^2y^3z - 7730448 \cdot x^5 + 3993 \cdot x^2y^3 \\
&\quad -31944 \cdot x^2y^2 + 31944 \cdot x^2y \\
f_{39} &= -170069856 \cdot x^8 + 87846 \cdot x^5y^3 - 1405536 \cdot x^5y^2 + 363 \cdot x^2y^5 + 1405536 \cdot x^5y \\
&\quad -3267 \cdot x^2y^4 + 5808 \cdot x^2y^3 - 2904 \cdot x^2y^2 \\
f_{40} &= 87846 \cdot x^6y^2 - 468512 \cdot x^6y + 363 \cdot x^3y^4 + 234256 \cdot x^6 - 4235 \cdot x^3y^3 + 5808 \cdot x^3y^2 \\
&\quad -8 \cdot y^5 - 1936 \cdot x^3y + 20 \cdot y^4 - 16 \cdot y^3 + 4 \cdot y^2 \\
f_{41} &= -2108304 \cdot x^5y^2z^2 + 1089 \cdot x^2y^5z^2 + 1405536 \cdot x^5yz^2 - 9438 \cdot x^2y^4z^2 \\
&\quad -15460896 \cdot x^5yz + 7986 \cdot x^2y^4z + 14520 \cdot x^2y^3z^2 \\
&\quad -63888 \cdot x^2y^3z - 5808 \cdot x^2y^2z^2 + 63888 \cdot x^2y^2z \\
f_{42} &= -1405536 \cdot x^5y^3z + 726 \cdot x^2y^6z + 1405536 \cdot x^5y^2z - 6534 \cdot x^2y^5z - 7730448 \cdot x^5y^2 \\
&\quad +3993 \cdot x^2y^5 + 11616 \cdot x^2y^4z - 31944 \cdot x^2y^4 - 5808 \cdot x^2y^3z + 31944 \cdot x^2y^3 \\
f_{43} &= 726 \cdot x^3y^5z - 4598 \cdot x^3y^4z + 3993 \cdot x^3y^4 + 5808 \cdot x^3y^3z - 16 \cdot y^6z - 21296 \cdot x^3y^3 \\
&\quad -1936 \cdot x^3y^2z + 40 \cdot y^5z + 10648 \cdot x^3y^2 - 88 \cdot y^5 - 32 \cdot y^4z + 132 \cdot y^4 + 8 \cdot y^3z \\
&\quad -44 \cdot y^3 \\
f_{44} &= -2811072 \cdot x^5y^3z + 1452 \cdot x^2y^6z + 2108304 \cdot x^5y^2z - 12705 \cdot x^2y^5z - 11595672 \cdot x^5y^2 \\
&\quad +11979 \cdot x^2y^5 + 20328 \cdot x^2y^4z - 127776 \cdot x^2y^4 - 8712 \cdot x^2y^3z + 111804 \cdot x^2y^3 \\
f_{45} &= -702768 \cdot x^5y^4 + 363 \cdot x^2y^7 + 702768 \cdot x^5y^3 - 3267 \cdot x^2y^6 + 5808 \cdot x^2y^5 - 2904 \cdot x^2y^4 \\
f_{46} &= 363 \cdot x^3y^6 - 2299 \cdot x^3y^5 + 2904 \cdot x^3y^4 - 8 \cdot y^7 - 968 \cdot x^3y^3 + 20 \cdot y^6 - 16 \cdot y^5 + 4 \cdot y^4 \\
f_{47} &= -702768 \cdot x^8z^2 - 726 \cdot x^5y^3z^2 + 2904 \cdot x^5y^2z^2 + 24 \cdot x^2y^4z^2 - 36 \cdot x^2y^3z^2 \\
&\quad +12 \cdot x^2y^2z^2 \\
f_{48} &= -1405536 \cdot x^8yz + 726 \cdot x^5y^4z - 5808 \cdot x^5y^3z + 5808 \cdot x^5y^2z \\
f_{49} &= 726 \cdot x^6y^3z - 3872 \cdot x^6y^2z + 1936 \cdot x^6yz - 16 \cdot x^3y^4z + 24 \cdot x^3y^3z - 8 \cdot x^3y^2z \\
f_{50} &= -15460896 \cdot x^8yz - 3993 \cdot x^5y^4z + 31944 \cdot x^5y^2z + 264 \cdot x^2y^5z - 527076 \cdot x^5y^2 \\
&\quad -396 \cdot x^2y^4z + 1452 \cdot x^2y^4 + 132 \cdot x^2y^3z - 726 \cdot x^2y^3 \\
f_{51} &= -7730448 \cdot x^8y^2 + 3993 \cdot x^5y^5 - 31944 \cdot x^5y^4 + 31944 \cdot x^5y^3 \\
f_{52} &= 3993 \cdot x^6y^4 - 21296 \cdot x^6y^3 + 10648 \cdot x^6y^2 - 88 \cdot x^3y^5 + 132 \cdot x^3y^4 - 44 \cdot x^3y^3 \\
f_{53} &= -1452 \cdot x^5yz^4 + 726 \cdot x^5z^4 - 7986 \cdot x^5z^3 + 726 \cdot x^2z^2 \\
f_{54} &= 87846 \cdot x^8z^3 - 726 \cdot x^5y^2z^3 + 726 \cdot x^5yz^3 + 726 \cdot x^2yz - 726 \cdot x^2z \\
f_{55} &= 726 \cdot x^6yz^3 - 363 \cdot x^6z^3 + 3993 \cdot x^6z^2 - 242 \cdot x^3z + y^2z - 2 \cdot yz + 11 \cdot y + z - 11 \\
f_{56} &= -2178 \cdot x^5y^2z^5 + 1452 \cdot x^5yz^5 - 15972 \cdot x^5yz^4
\end{aligned}
$$

$$
\begin{aligned}
f_{57} &= -1452 \cdot x^5y^3z^4 + 1452 \cdot x^5y^2z^4 - 7986 \cdot x^5y^2z^3 \\
f_{58} &= 1089 \cdot x^6y^2z^4 - 726 \cdot x^6yz^4 + 7986 \cdot x^6yz^3 + y^3z^2 - 3 \cdot y^2z^2 + 11 \cdot y^2z + 2 \cdot yz^2 \\
&\quad -22 \cdot yz \\
f_{59} &= -2904 \cdot x^5y^3z^4 + 2178 \cdot x^5y^2z^4 - 23958 \cdot x^5y^2z^3 + 63888 \cdot x^5yz^3 - 31944 \cdot x^5z^3 \\
&\quad -31944 \cdot x^2z \\
f_{60} &= -726 \cdot x^5y^4z^3 + 726 \cdot x^5y^3z^3 - 11595672 \cdot x^8z^2 - 31944 \cdot x^2y + 31944 \cdot x^2 \\
f_{61} &= 1452 \cdot x^6y^3z^3 - 1089 \cdot x^6y^2z^3 + 11979 \cdot x^6y^2z^2 - 31944 \cdot x^6yz^2 + 15972 \cdot x^6z^2 \\
&\quad +3 \cdot y^4z - 6 \cdot y^3z + 33 \cdot y^3 + 3 \cdot y^2z - 121 \cdot y^2 + 132 \cdot y - 44 \\
f_{62} &= -726 \cdot x^8z^5 + 3 \cdot x^2yz^3 \\
f_{63} &= -363 \cdot x^8yz^4 + 3 \cdot x^2y^2z^2 - 3 \cdot x^2yz^2 \\
f_{64} &= 363 \cdot x^9z^4 - x^3yz^2 - x^3z^2 \\
f_{65} &= -15972 \cdot x^8yz^4 - 2904 \cdot x^5yz^3 + 1452 \cdot x^5z^3 + 33 \cdot x^2y^2z^2 + 1452 \cdot x^2z \\
f_{66} &= 3993 \cdot x^8y^2z^3 + 527076 \cdot x^8z^2 + 33 \cdot x^2y^3z - 33 \cdot x^2y^2z + 1452 \cdot x^2y - 1452 \cdot x^2 \\
f_{67} &= 7986 \cdot x^9yz^3 + 1452 \cdot x^6yz^2 - 726 \cdot x^6z^2 + 11 \cdot x^3y^2z - 22 \cdot x^3yz + 4 \cdot y^2 - 6 \cdot y + 2 \\
f_{68} &= 263538 \cdot x^8y^2z^3 - 1405536 \cdot x^8yz^3 + 702768 \cdot x^8z^3 - 5808 \cdot x^5y^3z^3 + 8712 \cdot x^5y^2z^3 \\
&\quad -2904 \cdot x^5yz^3 + 702768 \cdot x^5z - 363 \cdot x^2y^3z + 2904 \cdot x^2y^2z - 2904 \cdot x^2yz \\
f_{69} &= 255104784 \cdot x^11z^2 - 131769 \cdot x^8y^3z^2 + 1054152 \cdot x^8y^2z^2 - 1054152 \cdot x^8yz^2 \\
&\quad +702768 \cdot x^5y - 363 \cdot x^2y^4 - 702768 \cdot x^5 + 3267 \cdot x^2y^3 - 5808 \cdot x^2y^2 + 2904 \cdot x^2y \\
f_{70} &= -131769 \cdot x^9y^2z^2 + 702768 \cdot x^9yz^2 - 351384 \cdot x^9z^2 + 2904 \cdot x^6y^3z^2 - 4356 \cdot x^6y^2z^2 \\
&\quad +1452 \cdot x^6yz^2 - 363 \cdot x^3y^3 + 2299 \cdot x^3y^2 - 2904 \cdot x^3y + 8 \cdot y^4 + 968 \cdot x^3 - 20 \cdot y^3 \\
&\quad +16 \cdot y^2 - 4 \cdot y
\end{aligned}
$$

Despite of how *complicated* the ideal I seems to be, its standard basis consists of few small polynomials. We print below the result using SINGULAR .

```
ring r = integer,(x,y,z),dp;
ideal I = ...;
std(I);
//>_[1]=18
//>_[2]=6z-12
//>_[3]=2y-4
//>_[4]=2z2+4z+8
//>_[5]=yz+z+3
//>_[6]=3x2z-15x2
//>_[7]=x2y+3x2z+x2
//>_[8]=x3+10z
//>_[9]=x2z2-4x2z-11x2
```

Using the new strategy described in Algorithm A:1.4.1, we compute the standard basis over $\mathbb{Q}[x, y, z]$ and see that it is equal to $\langle 1 \rangle$. Hence the ideal contains a constant. Using SINGULAR 's syzygy procedure syz, we are not lucky enough to already get access to 18, but a multiple of it: 6 133 248. This seems to be a big number, but if we add this constant to the generators of I the computations will drastically speed up the computations.

We show how 6 133 248 can be generated using $f_1, \ldots, f_{70}$.

$$\begin{aligned}
6\,133\,248 = \quad & \left(-231913440x^6z + 9838752x^3y - 4599936x^3z + 69696y\right) \cdot f_1 + \\
& \left(-1584y^3 + 13464y^2 + 12672y\right) \cdot f_2 + \\
& \left(-115956720x^6z + 1756920x^3y - 1724976x^3z - 40409160x^3 + 52272y + 200376\right) \cdot f_3 + \\
& \left(-38652240x^3y^3 + 38652240x^3y^2 + 154608960x^3 + 1149984y^2 + 4983264y + 3066624\right) \cdot f_4 + \\
& \left(-2551047840x^6z + 38652240x^3y - 37949472x^3z - 889001520x^3 + 574992y + 2779128\right) \cdot f_5 + \\
& \left(-159720x^3y^2 - 159720x^3y - 319440x^3 - 2376y^2 - 10296y - 12672\right) \cdot f_{10} + \\
& \left(-79860x^4 + 110xy^3 + 110xy^2 + 220xy - 2508x\right) \cdot f_{11} + \\
& \left(26136\right) \cdot f_{13} + \\
& \left(3630x^4y^2 + 3630x^4y + 7260x^4 + 54xy^2 + 154xy + 96x\right) \cdot f_{14} + \\
& \left(1540xy + 517x\right) \cdot f_{15} + \\
& \left(-792\right) \cdot f_{16} + \\
& \left(425920x\right) \cdot f_{18} + \\
& \left(-115956720x^3y^2 - 115956720x^3y - 231913440x^3 - 1724976y^2 - 7474896y - 4599936\right) \cdot f_{19} + \\
& \left(-1756920x^4 + 2420xy^3 + 2420xy^2 + 4840xy - 132616x\right) \cdot f_{26} + \\
& \left(851840x\right) \cdot f_{27} + \\
& \left(287496\right) \cdot f_{28} + \\
& \left(-38720x\right) \cdot f_{38}
\end{aligned}$$

Examples for Signature Algorithm

These are the ideals presented in Section A:2.4: Timings when comparing the updated version of the implemented Gröbner basis algorithm over $\mathbb{Z}[x]$ in Singular (`std`) with the extended variant of the F5 algorithm over the integers (`sba`).

These ideals are considered over $\mathbb{Z}[x, y]$ with the `dp` ordering.

	Generators for ideal
Example C:1	$95 \cdot y^3$, $54 \cdot x^2y + 33 \cdot xy^2 + 26 \cdot xz + 91 \cdot z^2$, $55 \cdot x^3 + 69 \cdot y^2$
Example C:2	$55 \cdot y^2z + 7 \cdot x^2 + 26 \cdot xy$, $50 \cdot x^3 + 35 \cdot x^2z + 52 \cdot y^2$, $81 \cdot z^3$
Example C:3	$14 \cdot x^3 + 57 \cdot xy$, $72 \cdot y^3 + 12 \cdot xy$, $27 \cdot x^2z + 43 \cdot y^2z + 69 \cdot z^2$
Example C:4	$2 \cdot x^2z + 67 \cdot y^2z + 24 \cdot z^2$, $3 \cdot x^3$, $55 \cdot y^3 + 67 \cdot x^2 + 85 \cdot xy$
Example C:5	$352 \cdot x^2y + 670 \cdot y^3 + 273 \cdot yz$, $718 \cdot x^3$, $961 \cdot y^2z + 283 \cdot xz^2 + 723 \cdot y^2 + 401 \cdot yz$
Example C:6	$19 \cdot xy^2 + x^2z + 51 \cdot y^2z + 87 \cdot xz^2 + 5 \cdot x^2 + 36 \cdot y^2$, $12 \cdot x^2z$, $98 \cdot yz^2 + 30 \cdot xy + 36 \cdot z^2$
Example C:7	$507 \cdot xy^2 + 308 \cdot xz$, $108 \cdot x^3 + 801 \cdot y^3$, $197 \cdot x^2z + 382 \cdot y^2z + 689 \cdot xz + 464 \cdot z^2$

Table 3.3: Examples for the Signature Based Algorithm

	Generators for ideal
Example C:8	$334 \cdot yz^2$, $900 \cdot x^3 + 396 \cdot xy^2 + 599 \cdot y^2$, $157 \cdot x^2z + 562 \cdot xyz + 72 \cdot xy + 799 \cdot z^2$
Example C:9	$556 \cdot xy^2 + 362 \cdot y^2z + 141 \cdot y^2$, $295 \cdot x^2y + 303 \cdot x^2$, $335 \cdot x^3 + 590 \cdot xy^2 + 541 \cdot z^2$
Example C:10	$366 \cdot xy^2 + 920 \cdot y^3$, $921 \cdot y^2z + 131 \cdot xz^2 + 888 \cdot z^2$, $224 \cdot x^2z + 281 \cdot xy + 23 \cdot yz$
Example C:11	$97 \cdot xy^2 + 647 \cdot y^3 + 715 \cdot z^2$, $694 \cdot x^2y + 65 \cdot xy + 536 \cdot y^2$, $85 \cdot y^3 + 267 \cdot x^2z$
Example C:12	$112 \cdot y^2z + 104 \cdot x^2 + 241 \cdot xz$, $662 \cdot x^2y + 773 \cdot xy^2$, $225 \cdot x^2z + 683 \cdot y^2z + 790 \cdot z^2$
Example C:13	$959 \cdot y^3 + 397 \cdot yz$, $915 \cdot x^2z + 953 \cdot y^2z + 346 \cdot xz^2 + 431 \cdot xy + 909 \cdot z^2$, $504 \cdot x^3$
Example C:14	$887 \cdot x^3$, $793 \cdot xy^2 + 589 \cdot x^2z + 474 \cdot xz + 366 \cdot yz$, $265 \cdot xy^2 + 20 \cdot xz^2 + 449 \cdot y^2$
Example C:15	$83 \cdot x^3 + 809 \cdot x^2y + 58 \cdot xz$, $68 \cdot xy^2 + 783 \cdot z^2$, $367 \cdot x^2z + 539 \cdot xz^2 + 336 \cdot yz$
Example C:16	$306 \cdot x^2z + 330 \cdot xz^2 + 611 \cdot z^2$, $572 \cdot xy^2 + 494 \cdot y^2$, $361 \cdot x^2y + 95 \cdot xyz + 578 \cdot xz$
Example C:17	$976 \cdot x^2y + 713 \cdot x^2z + 512 \cdot z^2$, $369 \cdot y^3 + 326 \cdot x^2z$, $145 \cdot xy^2 + 585 \cdot xz + 884 \cdot yz$

Table 3.4: Examples for the Signature Based Algorithm

PROGRAMMING IN SINGULAR 'S KERNEL

Introduction

Singular is a computer algebra system for polynomial computations that has been developed to help the users in various fields of mathematics like algebraic geometry, singularity theory and commutative algebra. It is a free software (GNU Public license) and it was supported by Deutsche Forschungsgemeinschaft (DFG), Stiftung Rheinland-Pfalz for Innovation and Volkswagen Stiftung.

The idea of Singular began in 1984 when computing invariants of ideals and modules in local rings (like Milnor, Tjurina numbers and dimensions) was needed to answer the following question: *Do there exist non-quasi homogeneous and complete intersection singularities with exact Poincaré complex?* The first step was to create a polynomial ring environment and implement a Gröbner basis algorithm to make use of its properties. This first version was written in basic on a 8-bit ZX Spectrum machine. Shortly after, these where implemented for Atari using Modula-2 language and Singular was born. In 1993 the code was rewritten in C-language.

In the first version of Singular (1997), one could compute multivariate polynomial factorization, gcd, syzygies and resolutions. In the following year, version 1.2 was released with faster algorithms, procedures for primary decompositions and normalization. Version 1.3 was again optimized, numerical data types and algorithms were added and one could compute moduli of space curves. Singular 2.0 was released in the early 2000's and the textbook *A Singular Introduction to Commutative Algebra* was published as both an introduction to the software and to commutative algebra. The book has many examples on how to use computer algebra techniques in algebraic geometry, commutative algebra and more. In 2004 at the International Symposium on Symbolic and Algebraic Computation (ISSAC), the Singular team was awarded the Richard D. Jenks Memorial Prize for Excellence in Software Engineering for Computer Algebra. A couple of years later, a second enhanced edition of the Textbook was published containing more examples and accompanied by the release of Singular 3.0, where non-commutative computations were possible together with absolute factorization and dynamic modules.

In the past 30+ years many developers worked on improving the algorithms, adding features and building new libraries. At the moment the source code contains 9 663 files (in total more than 1 million code lines) and 145 libraries and it can get really hard to start working in the sources.

This part will provide the reader with some basic guidelines into programming in Singular 's kernel, especially the part of the kernel specialized in the standard bases computation (the so-called GBEngine).

The Structure

The first chapter consists of the requirements needed to program in Singular 's kernel and explains some very basic concepts like `git`, compiling and building Singular .

The second chapter introduces the main structures in the sources like polynomials and ideals and lists the main related procedures pertaining to these concepts, e.g adding two polynomials, searching for constants in ideals, etc.

The third chapter explores the connection between the Singular language is and the internal structures.

The last chapter describes the adding of new structures and features to Singular while providing several examples.

Git, Compilation and Building

This section introduces the reader to the building of SINGULAR from the source files. A more detailed tutorial can be found on the Singular's GitHub Wiki page. This is by no means a ***Git*** tutorial, the purpose is to obtain some insight into the methods and tools used to develop SINGULAR .

In order to maintain a software as vast as SINGULAR , a large number of developers is required and it is vital that everybody works on the latest version. A problem may occur when two persons (say DevA and DevB) work on the same file in parallel. This leads to two versions of the file (say ***fileA*** and ***fileB***). If the two developers wish to make their changes public, the the following scenario may happen: DevA will replace the old file with ***fileA***. DevB will be unable to do the same since in this way DevA's changes will be lost. Hence DevB will have to change ***fileA*** manually.

Git is a free (version control) software that helps in this process. If the same lines of code were modified, a ***conflict*** appears and the code needs to be reviewed by one of the developers. Otherwise the ***merge*** is done automatically. In addition to this git also provides the entire history of commits in order to help identifying the commit to blame when tests fail. A web version of Git with graphic interface, community and other useful tools can be found on ***GitHub***.

In order to install SINGULAR on a Linux machine, certain prerequisites are needed. The complete list of software that one needs to install can be found at github.com/Singular/Sources/wiki/Step-by-Step-Installation-Instructions-for-Singular , but the main ones are:

- `git` – version control,
- `gcc` – C++ compiler,
- `automake` – tool for generating compliant makefiles,
- `autoconf` – automatic configure script builder,
- `make` – utility for directing compilation,
- `readline`,
- `GMP` – multiprecission integer library (https://gmplib.org/)
- `NTL` – high-performance C++ library providing data structures and algorithms for manipulating arbitrary length integers, vectors, matrices, and polynomials over the integers and over finite fields.

Singular can be found as a Git repository at https://github.com/Singular/Sources. In order to implement some changes, first one needs to clone the repository locally, for example in the *Tutorial* folder. This can be done by using the following command line in the Linux terminal:

```
git clone https://github.com/Singular/Sources Tutorial
```

and generate the configure script using

```
cd Tutorial
./autogen.sh
```

The most common way to configure and build Singular is to run

```
./configure
make
```

Now the program binary can be found in `Tutorial/Singular/Singular`.

Sometimes a debug version is required – it is slightly slower, but sets debug flags and therefore in case of a segmentation fault one can identify the file and line number where it crashed using the debugger `gdb`'s backtrace command `bt`.

One can build a debug version by typing

```
./configure --enable-debug --disable-optimizationflags
make
```

Since building Singular is time consuming, it is recommended to use different folders for the various versions. Otherwise, the rebuilt of the entire software instead of only parts of it will be required. In the following example we configure and build Singular in another folder, say `Tutorial/Folder`.

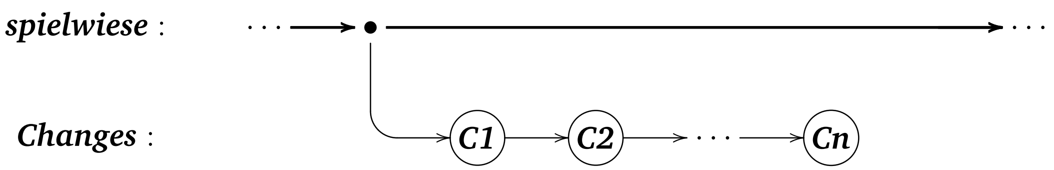
```
cd Tutorial
#creates a new folder
mkdir Folder
#changes the current working directory
cd Folder
#configures using the sources in Tutorial
../configure
#builds Singular in parallel using 3 processors
make -j3
```

This newly build version of Singular can be found as the path `Tutorial/Folder/Singular/Singular`.

The repository can be viewed as a tree, the main branch is called ***spielwiese***. In order to modify it, a copy needs to be downloaded into the local machine and merged again to the main branch sometimes in the future. Note that this branch could have evolved in the meantime. To avoid name confusion a new branch say (***Changes***) starting from the old one needs to be created. One can do this by typing

```
git checkout -b Changes
```

After the changes are *commited* (can be several commits, call them (***C1***), (***C2***), . . . , (***Cn***)), the tree would look like this:



Assume that currently you are working with two repositories (or remotes):

- `singular` – this is the official SINGULAR repository
- `user` – this is the cloned repository containing the two branches: `spielwiese` and `Changes`

Then, the file `Tutorial/.git/config` will look like the following:

```
[remote "singular"]
fetch = +refs/heads/*:refs/remotes/origin/*
url = git@github.com:Singular/Sources.git
[remote "user"]
fetch = +refs/heads/*:refs/remotes/origin/*
url = git@github.com:git_user_name/Sources.git
```

One can make the changes public by using the ***push*** command – this will merge the changes and upload them to your git account:

```
git push user Changes
```

In order to merge the local changes and make them public one needs to do the following:

```
#get the updates of all repositories
git remote update
#changes the current working branch
git checkout Changes
#updates the spielwiese branch
git pull singular spielwiese
#uploads the current branch
git push user Changes
```

Another useful command worth mentioning is `git grep`. This searches in the entire repository for instances where the exact same text appears. In the example below, we use it to search for the definition of `mora` – the function that computes standard bases in the case of local and mixed orderings.

```
cd Tutorial
git grep mora
```

and see that it is defined in `kernel/GBEngine/kstd1.cc`. However, most text editors have better ways to find location of a procedure or definition: `vim` has `ctags` and `emacs` has `etags`.

Assume that an error occurred when we ran the debug version of Singular located at `Tutorial/debug/Singular/Singular` for the input file `Desktop/test`. In order to fix this segmentation fault we need to use a debugger like `gdb` as in the following example:

```
# starts debugging the binary
gdb Tutorial/debug/Singular/Singular
#run the program using as input the test file
r ~/Desktop/test
```

When it breaks at the segmentation fault that caused SINGULAR to crash, one can backtrace to find the cause of this error by typing `bt`. Adding breakpoints is useful when debugging:

```
# creates a breakpoint in file GBEngine/kutil.cc. line 413
b kutil.cc:413
# creates a breakpoint at each call to the function redRing
b redRing
```

After a breakpoint, one can type the following:

`step` (or `s`)	will go to the next line of code, will step in functions as well
`next` (or `n`)	will execute the next line of code, will not enter functions
`finish` (or `fin`)	will continue till the end of the current function
`continue` (or `c`)	will continue till the next breakpoint

Internal structures

Polynomials

The most used structure in SINGULAR is the polynomial. The source code contains several representations for polynomials, depending on the purpose.

A main assumption for the occurring polynomials is that they are very sparse, i.e. most monomials have the coefficient set to 0. For instance, for the univariate polynomial $x^{100}+1$, instead of representing it as an array

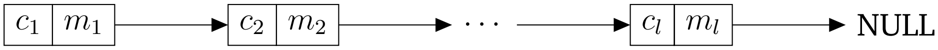
$$(1, 0\ldots, 0, 1),$$

where the $i-$th position holds the coefficient of x^{i+1}, one can save it as a list that contains both the coefficient and the exponents.

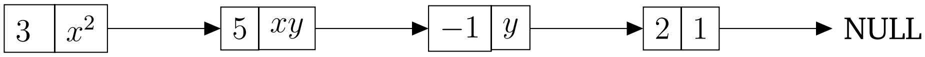
The poly structure

As mentioned above, there are several different structures for storing polynomials. The `poly` structure is the most used and it is optimized for the standard basis computations. For these computations we need to have fast access to the leading term, and the most used polynomial operations are the leading term comparison and divisibility. The structure consists of linked monomials that are ordered with respect to the chosen monomial ordering. Therefore, if the monomial ordering is unknown, then we cannot construct polynomials. This is the reason why in SINGULAR we always define a ring with a monomial ordering. The `poly` structure is *sparse distributive*

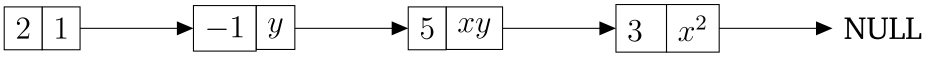
Let R be a ring together with a monomial ordering $<$ on $\mathrm{Mon}(x_1, \ldots, x_n)$ and $f = c_1 \cdot m_1 + c_2 \cdot m_2 + \ldots + c_l \cdot m_l \in R[x_1, \ldots, x_n]$ a polynomial, with the coefficients $c_i \in R$ and the ordered monomials $m_i < m_{i+1}$. Then `poly` will save f as



EXAMPLE. Consider $f = 3x2 + 5xy - y + 2 \in R[x, y]$ with the `dp` ordering. Then f will be internally represented as



Note that by changing the monomial ordering to `ls`, f will be stored as

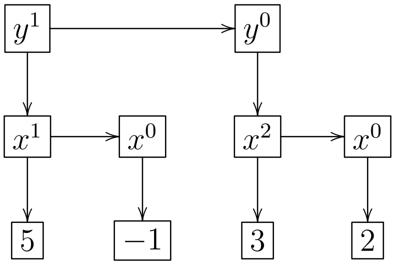


REMARK.
Note that the `poly` structure is ring dependent.

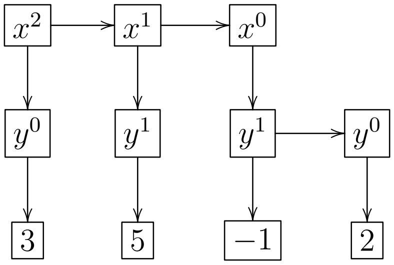
The CanonicalForm structure

In *factory*, a standalone library for factorizing multivariate polynomials, the main polynomial operation is multiplication. Therefore, another structure used to store polynomials arises: ***CanonicalForm***. This is a *sparse recursive* structure where the polynomials in $R[x_1, \ldots, x_n]$ are seen recursively as polynomials in $R[x_1, \ldots, x_{n-1}][x_n]$.

EXAMPLE. Take the same example as before: $f = 3x^2 + 5xy - y + 2 \in R[x, y]$ with the `dp` ordering. Then $f = (5x - 1) \cdot y^1 + (3x^2 + 2) \cdot y^0 \in R[x][y]$, and hence f will be stored as a CanonicalForm



Note that this representation does not depend on the monomial ordering, but on the order of the variables. For instance, if we consider $f \in R[y, x]$, then the Canonical Form will look like



We will go more into detail in the `poly` structure. For a more in depth overview and differences between the two structures see Hans Schönemann's work [BS] and [S1].

Back to polys

In the following we will take a closer look at the `poly` structure since it was the one used in the implementations mentioned throughout this thesis.

The definition of the `poly` structure can be found in the sources files in `libpolys/polys/monomials/monomials.h`. The main procedures for `polys` can be found in the `p_polys.h` and `p_polys.cc` located in the same folder as the previous one. Note that a `poly` does not know in which ring it exists. The reason for this is memory: if in each monomial we would save the ring, then a lot of memory would have been used. This also means that for each polynomial operation we need to name the ring. For Example the function header that adds two polynomials p, q in the ring r looks like

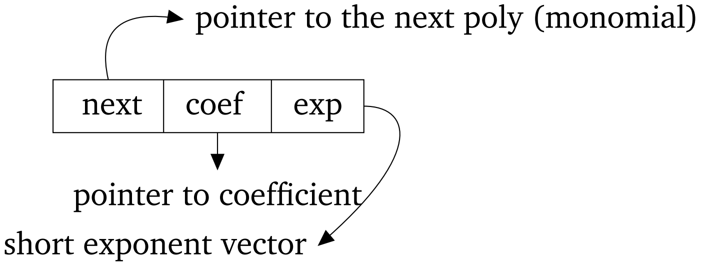
```
poly p_Add_q(poly p, poly q, const ring r);
```

For the current ring, named `currRing`, one can use the macro `pAdd(p,q)` instead of `p_Add_q(p,q,currRing)`. Most of these function usually delete the input polynomials and the reason can be understood by reviewing the following scenario: Let p, q, s, t be four polynomials and call `pAdd(pAdd(p,q),pAdd(s,t))` in order to compute the sum $p+q+s+t$. The second and third `pAdd`'s will return two polynomials, and since these polynomials will not be deleted, a memory leak will occur.

In order to keep the original polynomiaos, one needs to add two copies of them by using `pAdd(pCopy(p),pCopy(q))` and these copies will be deleted instead of the originals.

There exist however several polynomial operations that do not delete the arguments.

Returning to the `poly` structure, the definition of the structure looks like this:



In the short exponent vector we will *squeeze* the exponents vector. For more details see O. Bachmann and H. Schönemann's paper [BS] or the sources files `/libpolys/polys/monomials/p_polys.cc` at the documentation for the function `p_GetShortExpVector` that stores the short exponent vector.

Remark.

The `poly` structure defined for polynomials $f \in R[x]$ can be extended for module elements $F = (f_1, \ldots, f_m) \in R[x]^m$. Let $\varepsilon_1, \ldots, \varepsilon_m$ be the canonical basis of the free module. We write $F = f_1 \cdot \varepsilon_1 + \ldots + f_m \cdot \varepsilon_m$. In the short exponents vector we will also save the component of the monomial. If the element is a polynomial, we will set it's component by default to 0.

Here are the basic functions when working with polynomials.

pNew()	○ allocates space for a new polynomial
pInit()	○ initializes a new polynomial, and sets everything to 0
pOne()	○ returns a polynomial equal to 1 in currRing
pISet(long i)	○ returns a polynomial p=i in currRing
pNSet(number n)	○ returns a polynomial p=n in currRing
pWrite(poly p)	○ prints the polynomial p in the current ring
pSetCoeff(poly p,number n)	○ sets the leading coefficient of p to n
pGetCoeff(poly p)	○ returns the leading coefficient of p
pGetComp(poly p)	○ returns the component of p
pSetComp(poly p,int c)	○ sets the component of p to c
pHead(poly p)	○ returns a copy of the leading term of p
pCopy(poly p)	○ returns a copy of p
pNext(poly p)	○ returns the next monomial in p
pLmDelete(poly p)	○ deletes the leading term of p
pLmDeleteAndNext(poly p)	○ deletes the leading term and returns the next monomial
pLmCmp(poly p,poly q)	○ compares LM(p) with LM(q), returns
	-1 if $LM(p) < LM(q)$
	0 if $LM(p) = LM(q)$
	1 if $LM(p) > LM(q)$
pLtCmp(poly p,poly q)	○ compares LT(p) with LT(q) with the absolute value of the coefficients
pLmDivisibleBy(poly p,poly q)	○ return True if $LM(q) \mid LM(p)$
pLmShortDivisibleBy(p,q)	○ fast pre-test for divisibility, returns p does not divide q or q may be divisible by p
pNeg(poly p)	○ returns -p
ppMult_nn(poly p,number n)	○ returns n·p, does not destroy p
pMult_nn(poly p,number n)	○ returns n·p, deletes p
ppMult_mm(poly p,poly m)	○ returns for a monomial m, m·p, does not destroy p
pMult_mm(poly p,poly m)	○ returns for a monomial m, m·p, deletes p
pMult(poly p, poly q)	○ returns p·q, deletes p and q
ppMult_qq(poly p,poly q)	○ returns p·q, does not deletes p,q
pMinus_mm_Mult_qq(p, m, q)	○ returns p-m·q, destroys p
pPlus_mm_Mult_qq(p, m, q)	○ returns p+m·q, destroys p
pAdd(poly p, poly q)	○ returns the sum p+q, deletes p and q
pSub(poly p,poly q)	○ returns p-q, deletes p,q
pPower(poly p, int i)	○ returns the i-th power of p
pIsConstant(poly p)	○ returns True if p is constant

pIsUnit(poly p)	◦ returns True if p is constant and it is a unit in currRing
pLcm(poly a,poly b,poly m)	◦ saves lcm(LM(p),LM(q)) in m
pDiff(poly p, int k)	◦ returns the partial differentiate of a by the k-th variable, does not destroy p

Table 3.5: Polynomial related procedures

Ideals

Another important structure in SINGULAR is the ideal. It is stored as an array of polynomials (polyset) and it can store modules because the polynomials may have non-zero components as we seen above. The ideals structure share the same internal structure with matrices and maps. It is defined in `libpolys/polys/simpleideals.h` and looks like

```
struct sip_sideal
{
  poly*   m;
  long rank;
  int nrows;
  int ncols;
}
```

The pointer m will point to a chunk of memory of size $\texttt{nrows} \cdot \texttt{ncols} \cdot \mathrm{size}(\texttt{poly})$. Note that some of the polynomials $m[0], \dots, m[\texttt{ncols} - 1]$ may be null.

Note that in SINGULAR the ideal structure is a set of polynomials that can be interpreted as a generating set or standard basis depending on the context. Ideals, modules, matrices and maps are build the same at the C−level despite that they have different types in the interpreter.

For an ideal, $\texttt{nrow} = 1$, $\texttt{ncol} \geq 0$ is the number of generators of the ideal, and `rank = 1`. For the module everything remains the same as for ideals, except the $\texttt{rank} \geq 1$.

It is easy to see that this would be also an effective structure to store a matrix when $\texttt{nrow} \geq 1$. However, in the following we will restrict ourselves to ideals and modules.

The structure `ideal` is a pointer to `sip_sideal`. The following table contains some basic ideal operations that can be used in the sources of singular. The definitions can be found in `simpleideals.cc`. The more advanced procedures, like intersection, syzygy computation can be found in `ideals.cc`. As in the polynomial case, the name of the functions for the ideals start with `id_...` and it also needs the ring as input. As before, some of these functions can be called with `id...` and will use the current ring as input.

idInit(int s,int k)	◦ returns an initialized ideal of size s and rank k
id_Delete(ideal* h,ring r)	◦ deletes the ideal h from r
idSkipZeroes(ideal h)	◦ gives h the minimal possible size (deletes the Null entries)
idElem(ideal h)	◦ returns the number of non zero elements
IDELEMS(ideal h)	◦ returns the total number of elements
id_Copy(ideal h,ring r)	◦ returns a copy of h in r
id_CopyFirstK (ideal h,int k,ring r)	◦ returns a copy of the first k elements
id_SimpleAdd(ideal h1, h2,ring r)	◦ adds two ideals without simplifying the result
id_Add(ideal h1, h2,ring r)	◦ adds two ideals skipping double elements and searching also for units
id_RankFreeModule(ideal h,ring r)	◦ returns the rank of h
id_FreeModule(int i, ring r)	◦ returns the free module of rank i in r
id_PosConstant(ideal h, ring r)	◦ returns position of generator with leading term a constant or -1 if no such generator was found
id_Head(ideal h,ring r)	◦ returns the ideal of the leading terms of h
id_MaxIdeal(int i,ring r)	◦ returns the i-th power of the maximum ideal
id_Transp(ideal M,ring r)	◦ returns the transpose of the module M
id_Mult (ideal h1,h2,ring r)	◦ returns h1$\cdot$ h2, at least one of the ideals has to have rank 1
id_Homogen(ideal h, int i,ring r)	◦ returns the homogenized h w.r.t. the i-th variable
id_Jet(ideal h,int i,ring r)	◦ returns the i-th jet of the elements of h
id_Subst(ideal h,int i,poly p,ring r)	◦ returns h, where the i-th variable was substituted by p
idPrint(ideal h)	◦ prints h in currRing (works only in the debug version and should be avoided)
ipPrint_MA	◦ prints the ideal
idInsertPoly (ideal h, poly p)	◦ adds p to h if $p \neq 0$
idIs0 (ideal h)	◦ returns True if h is the 0 ideal
idMinBase (ideal h)	◦ returns a minimal set of generators of h using standard basis
idSect(ideal h1,ideal h2)	◦ returns the intersection of h1 with h2
idSyzygies	◦ returns the syzygies of the input
idQuot	◦ returns the quotient of the input

Table 3.6: Ideal related procedures

Strategy

The next structure plays an important role in the standard bases algorithm. It stores the status of a Gröbner basis computation: partial standard basis, the pair list and several special routines. It is present in most of the Gröbner basis related functions as an argument called `strat`.

We start by briefly describing the `sTObject` class. The class stores the polynomials used in the reductions and it is a foundation for the class used for the L−pairs. The main members are `poly p` – the polynomial, `poly sig` – the signature in case of a signature based algorithm, `unsigned long sevSig` – the short exponent vector for the signature, `int ecart` – the ecart of the polynomial and `long FDeg` – the degree. `TObject` is a pointer to this class.

The class `sLObject` starts from `sTObject` and adds the things needed by the pair list. We shortly present the main members: `unsigned long sev` – the short exponent vector for the polynomial `p`, `poly p1,p2` – the polynomials that generated this pair, `poly lcm` – the least common multiple of the leading terms of `p1` and `p2` used in many criteria. `LObject` is a pointer to this class.

Since SINGULAR has many reduction procedures, many orderings for the pair list among others, we save all the used procedures also in the strategy class `skStrategy`. The following table contains the main members of the class.

LObject P	pointer to the current pair that is considered
ideal Shdl	stores the partial standard basis as an ideal
polyset S	stores the partial standard basis as polyset (part of Shdl)
int sl	last used index in S
int smax	size of S
LSet L	the pair list
int Ll	last used index in L (the current index since we consider pairs from the end of L)
int Lmax	size of L
LSet B	an intermediate pair list for the Buchberger algorithm in each loop we add the new pair in B, apply several criteria and finally add them into L
int Bl, Bmax	similar as Ll and Lmax
TSet T	the polynomials used in the reduction procedures
int tl, tmax	similar as Ll and Lmax

unsigned long* sevS	stores the short exponents vectors for elements from S
unsigned long* sevT	stores the short exponents vectors for elements from T
int (*red)(LObject, kStrategy)	pointer to the used reduction procedure
void (*initEcart)(TObject)	pointer to the function that sets the ecart, length and degree
int(*posInT)(TSet T, LObject)	positioning of a new LObject in T, used for ordering T
int(*posInL)(LSet L, LObject,...)	positioning of a new LObject in L used for ordering L
void(*enterS)(LObject,kstrategy)	pointer to the procedure that adds an element to S
TObject* R[]	array of pointers to T
int S_2_R[]	S_2_R[i] yields the Tobject which corresponds to S[i]

When working with the signature based algorithms, we add these members to the strategy class:

polyset syz	the syzygies (used in the Syzygy Criterion from F5)
polyset sig	the signatures (used in F5 algorithms)
unsigned long* sevSyz	stores the short exponents vectors for elements from syz
unsigned long* sevSig	stores the short exponents vectors for elements from sig
int *posInLSba	position function used in the F5C strategy for signatures
bool sigdrop	True if we have found a sigdrop, False else

The interpreter structure leftv

This structure links the Singular language with the C++ language from it's source files. `leftv` is used in the next two chapters and it is a pointer to `sleftv`. This contains the members that one would expect:

<table>
<tr><td>int rtyp</td><td>stores what type of data it is. A full correspondence can be found in <code>table.h</code>
<table>
<tr><th>Singular Type</th><th>C++ Type</th></tr>
<tr><td>int</td><td>INT_CMD</td></tr>
<tr><td>poly</td><td>POLY_CMD</td></tr>
<tr><td>ideal</td><td>IDEAL_CMD</td></tr>
<tr><td>module</td><td>MODULE_CMD</td></tr>
<tr><td>ring</td><td>RING_CMD</td></tr>
<tr><td>...</td><td>...</td></tr>
</table>
</td></tr>
<tr><td>void* data</td><td>points to the data in the memory</td></tr>
<tr><td>leftv next</td><td>points to the next entry (for example in a list of polys)</td></tr>
</table>

We only included the members needed in the following section. A complete list can be found in the file `subexpr.h`.

SEARCHING FUNCTIONS FROM THE INTERPRETER

This section explores the path the data takes from the SINGULAR terminal to the C/C++ files that store the internal structures. The connection between SINGULAR 's interpreter and the kernel is made by two files: table.h and iparith.cc. table.h unveils the connection between the procedures called in the SINGULAR language and the kernel. For instance for the std command that computes the standard basis looks like this:

```
{D(jjSTD), STD_CMD, IDEAL_CMD, IDEAL_CMD, ALLOW_PLURAL | ALLOW_RING},
{D(jjSTD), STD_CMD, MODUL_CMD, MODUL_CMD, ALLOW_PLURAL | ALLOW_RING},
```

The first column displays the name of the procedure, the second the command, the third column the type of the result, the forth the type of the input, and on the last column the context: whether it is allowed in the non commutative case (Plural) or in rings that are not fields.

The std command corresponds to the first row when the input consists of ideals and on the second row when it consists of modules. Note that in this case, both call the same function: jjSTD. The definition of this function can be found in iparith.cc. It extracts the input data (the input ideal or module) and call kStd – the main procedure for standard bases computation.

The next example illustrates what happens when we add two polynomials in the interpreter.

```
D(jjPLUS_P), '+',POLY_CMD,POLY_CMD,POLY_CMD,ALLOW_PLURAL|ALLOW_RING
```

This part from table.h is responsible for adding two polynomials and the procedure jjPLUS_P is called. Here is the definition of jjPLUS_P as in iparith.cc:

```
static BOOLEAN jjPLUS_P(leftv res, leftv u, leftv v)
{
  res->data = (char *)(pAdd((poly)u->CopyD(POLY_CMD) , (poly)v->CopyD(
   POLY_CMD)));
  return jjPLUSMINUS_Gen(res,u,v);
}
```

The result is stored in res, while u and v are the first, respectively second arguments of the "+" command. Note that we used the pAdd function and as seen earlier, the function deletes its arguments. This is why we first copy the input using CopyD. In this case jjPLUSMINUS_Gen will do nothing since it is used for lists of polynomials as seen in the following example:

```
ring r = 0,(x,y),dp;
list p = x,y;
list q = x2,y;
p[1..2]+q[1..2];
//>x+x2 2y
```

An example worth analyzing in depth is the standard basis algorithm. The part responsible for the standard bases algorithms is located in the folder `kernel/GBEngine`. The main files are:

`kstd1.cc`	handles the local and mixed case of the standard basis and contains also several reduction procedures
`kstd2.cc`	handles the global case of standard basis and contains several reduction procedures
`kutil.h`	defines the class for TObjects and LObjects
`kutil.cc`	contains many functions including different positioning, adding new pairs and initializing parts of the used strategy
`kspoly.cc`	defines functions needed for the computation of a new s−polynomial or reduction steps
`kInline.h`	defines TObjects and LObjects related functions

Earlier we saw that when the command `std` is called in SINGULAR , we linked it to `jjSTD` defined in the file `iparith.cc`. Here, after extracting the input from the `leftv` structure, we run the function `kStd`. This is the main standard basis function and it is defined in `kstd1.cc`. `kStd` takes as input the ideals F and Q, where F is the input ideal and Q is the quotient ideal in the case of a quotient ring (`qring`), the weights in case of a weighted ordering and several flags. In `kStd` a new strategy is defined, where we set some flags like the homogeneity of the input and check for example whether the ordering is a lexicographic ordering or not. If the monomial ordering is a global one, `bba` (Buchberger algorithm) defined in `kstd2.cc` is called. Otherwise `mora` (Mora algorithm − the generalization of Buchberger for local cases considering the ecart) defined in `kstd1.cc`.

The next step in both functions is setting the strategy. This is denoted by the object `strat` of `kStrategy` type as seen in the earlier section.

`initBuchMoraPos`	sets the positioning functions for the set T and L
`initBuchMoraCrit`	sets the used criteria
`initBba / initMora`	sets the reduction procedures, enterS and initEcart
`initBuchMora`	initializes the sets S, L and T and adds the input as initial pairs to L

Then, as in Buchbergers algorithm, we loop till the set L is empty and take the last element from the pair list (since it is the smallest element with respect to the chosen pair ordering). P will point to this pair. Usually we compute for the pairs the so-called short s−polynomial, namely the leading term of the normal s−polynomial, used since it consumes less memory. If P is a short s−polynomial, we need to build the normal one and we start by reducing and tail reducing it with the set T.

If it can be reduced to 0, then simply delete this pair and continue with the next loop. However, if we obtained a non zero polynomial, we have to add the s−polynomials to the pair list L when working over fields (`enterpairs`) or add the corresponding strong polynomials, s−pairs and the extended polynomial (`superenterpairs`) to L when computing over rings. Next, add the element to the reduction set T and to the partial standard basis S. In the case of fields, we first add the new pairs to the set B and after applying several criteria on the set B, we add it to L.

At the end we do a complete reduction and delete the sets L,T and at the end of `kStd` we delete the strategy.

ADDING NEW FEATURES

The easiest way to add something in SINGULAR is by adding it in the `extra.cc` file. This can be found in the `Singular` folder of the source files and uses SINGULAR 's command `system`:

```
system("name_of_function", input_of_function);
```

When this function is called, it first tests if the first argument is equal to the string "`name_of_function`" by means of an *if* statement and if so, runs the code through there.

This is useful since one can use the internal structures in SINGULAR interpreter. We provide a couple of very easy examples just to show how this works. First of all in `extra.cc` there are two functions: `jjSYSTEM` and `jjEXTENDED_SYSTEM`. The first one only accepts documented functions, while in the second one, experimental functions can also be added. So the *new function* blueprint would be this block added in the `jjEXTENDED_SYSTEM` function.

```
/*========new functions feature========*/
if(strcmp(sys_cmd,"test")==0)
{
  // your code in here
  return FALSE;
}
else
```

EXAMPLE. This example illustrates the adding of a function that takes a string as input and prints it. Name this instance with *test1* and the input is stored in `leftv h`. Then we have to add these lines of code:

```
//checks if the argument is non-empty and if it is of string type
if ((h!=NULL) && (h->Typ()==STRING_CMD))
{
  //define a new string as the content of the input
  char *str = (char *)h->Data();
  PrintS("This is the string:");  //prints the string
  Print(str);                     //prints str
  PrintLn();                      //prints a new line
}
```

Now we have to rebuild the sources via `make`. We will display SINGULAR 's output with a >> symbol. If we type now in SINGULAR the following is displayed

```
system("test1","Example");
>> This is the string: Example
```

We can extend the function for polynomials, ideals and rings by adding the following at the end of `extra.cc`:

```
/*========new functions feature========*/
if(strcmp(sys_cmd,"test1")==0)
{
  if ((h!=NULL) && (h->Typ()==STRING_CMD))
  {
    char *str = (char *)h->Data();
    PrintS("This is the string:"); Print(str);PrintLn();
  }
  if ((h!=NULL) && (h->Typ()==POLY_CMD))
  {
    poly p = (poly)h->Data();
    PrintS("This is the poly: "); pWrite(p);
  }
  if ((h!=NULL) && (h->Typ()==IDEAL_CMD))
  {
    ideal I = (ideal)h->Data();
    PrintS("This is the ideal:");PrintLn();
    for(int j=0;j<IDELEMS(I);j++)
    {
      PrintS("I[%i]",j+1);pWrite(I->m[j]);
    }
  }
  if ((h!=NULL) && (h->Typ()==RING_CMD))
  {
    ring R = (ring)h->Data();
    PrintS("This is the ring:");PrintLn();
    rWrite(R);PrintLn();
  }
  return FALSE;
}
else
```

After rebuilding, we can do the following in SINGULAR :

```
system("test1","Example");
>> This is the string: Example
ring r = integer,(x,y),dp;
system("test1",r);
>>This is the ring:
>>//   coeff. ring is : Integers
>>//   number of vars : 2
>>//        block   1 : ordering dp
>>//                  : names    x y
poly f = x2+y;
system("test1",f);
>>This is the poly: x2+y
ideal i = 4,13x,7x2;
system("test1",i);
>>This is the ideal:
>>I[1]4
>>I[2]13x
>>I[3]7x2
```

The next function takes to polynomials p, q as input and returns the sum, difference and product between them.

Example. This feature will be added in the `jjEXTENDED_SYSTEM` in `extra.cc`. note that we use `pCopy` since the functions `pAdd`, `pSub` and `pMult` delete the input polynomials.

```
/*========new functions feature========*/
if(strcmp(sys_cmd,"test2")==0)
{
  if ((h!=NULL) && (h->Typ()==POLY_CMD))
  {
    poly p = (poly)h->Data();
    h = h->next;
    if ((h!=NULL) && (h->Typ()==POLY_CMD))
    {
      poly q = (poly)h->Data();
      PrintS("Sum  = ");pWrite(pAdd(pCopy(p),pCopy(q)));
      PrintS("Sub  = ");pWrite(pSub(pCopy(p),pCopy(q)));
      PrintS("Mult = ");pWrite(pMult(pCopy(p),pCopy(q)));
    }
  }
  return FALSE;
}
else
```

After rebuilding we can do the following in Singular :

```
ring r = 0,(x,y),dp;
poly f = x2+y;
poly g = 3x-y;
system("test2",f,g);
>> Sum  = x2+3x
>> Sub  = x2-3x+2y
>> Mult = 3x3-x2y+3xy-y2
```

Example. Recall that in Part A we explained Algorithm A:1.4.1 (preIntegerCheck), that takes the ideal I as input and returns a constant (or monomial) that can be generated with I, if such constant (or monomial) exists. We implemented this function in Singular and, in order to use it from Singular 's terminal, we need to add the following block:

```
if(strcmp(sys_cmd,"test3")==0)
{
  if ((h!=NULL) && (h->Typ()==IDEAL_CMD))
  {
    ideal I = (ideal)h->Data();
    res->rtyp=POLY_CMD ;
    res->data=(void*) preIntegerCheck(I,NULL);
  }
  return FALSE;
}
else
```

The output is saved in the variable `leftv res`. In this case it is a polynomial containing the returned value of the algorithm `preIntegerCheck`. After rebuilding Singular , we can type:

```
ring r = integer,(x,y),dp;
ideal i = 4x+16,2x+y,y+13;
def p = system("test3",i);
p;
>> 42
i = i[1],i[2];
def p = system("test3",i);
p;
>> 0
```

In the first case, $42 \in i$ since $42 = i[1] - 2 \cdot i[2] + 2 \cdot i[3]$. In the second case no constant / monomial was found in the ideal i.

Another method to add new features to Singular is by building a dynamic module. The functions written in these modules will work like a function from a Singular library (type `proc`). This is done by using the command `iiAddCproc` which builds the Singular procedure from these C functions. Writing a function in C/C++ directly to sources will speed up the computation time since converting the data from the interpreter to the kernel takes time.

Adding a new datatype to the Singular language may prove to be challanging since the interpreter will have to be changed in multipl places. However, there exists a tool, Blackbox, that does these changes automatically. It consists by registering the data name and related procedures like initialization, deletion, copying, print and operations. For more details see [S2].

References

[CoCoA] J. Abbott, A. M. Bigatti, *CoCoALib: a C++ library for doing Computations in Commutative Algebra*, available at http://cocoa.dima.unige.it/cocoalib.

[Artin1] M. Artin, *Algebraic approximation of structures over complete local rings*, Publ. Math. IHES, **36**, (1969), 23-58.

[Artin2] M. Artin, *Constructions technques for algebraic spaces*, Actes Congres. Intern. Math., (1970), 419-423.

[BS] O. Bachmann and H. Schönemann, *Monomial representations for Gröbner bases computations*, ISSAC '98, (1998), 309-316.

[BGS] A.M. Bigatti, P. Gimenez, E. Sáenz-de-Cabezón, *Monomial Ideals, Computations and Applications*, Springer, (2013).

[BiHo] C. Biro, D.M. Howard, M.T. Keller, W.T. Trotter, S.J. Young, *Interval partitions and Stanley depth*, J. Combin. Theory, Ser. A, **117**, (2010), 475-482.

[BH] W. Bruns, J. Herzog, *Cohen-Macaulay rings*, Revised edition. Cambridge University Press, (1998).

[BIS] W. Bruns, B. Ichim, C. Söger, *The power of pyramid decomposition in Normaliz,* J. Symb. Comp., **54**, (2016), 513-536.

[BKU] W. Bruns, C. Krattenthaler, J. Uliczka, *Stanley decompositions and Hilbert depth in the Koszul complex*, J. Commutative Alg., **2**, (2010), 327-357.

[BMU] W. Bruns, J. Moyano-Fernández, J. Uliczka, *Hilbert regularity of ZZ-graded modules over polynomial rings*, (2013), arXiv:AC/1308.2917.

[Bu] B. Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, PhD thesis, University of Innsbruck, (1965).

[Ci] M. Cimpoeas, *Stanley depth of complete intersection monomial ideals*, Bull. Math. Soc. Sc. Math. Roumanie **51**(99), (2008), 205-211.

[Singular] W. Decker, G.-M. Greuel, G. Pfister, H. Schönemann, Singular 4-1-3 — A computer algebra system for polynomial computations, available at www.singular.uni-kl.de, (2016).

[DuGo] A. Duval, B. Goeckner, C. Klivans, J. Martin, *A non-partitionable Cohen-Macaulay simplicial complex*, Adv. Math., **299**, (2016), 381-395.

[E] C. Eder, *Signature-based algorithms to compute standard bases*, PhD thesis, University of Kaiserslautern, (2012).

[EF] C. Eder, J.-C. Faugère, *A survey on signature-based Gröbner basis computations*, arXiv:AC/1404.1774, (2014).

[EP] C. Eder, J. Perry, *F5C: A variant of Faugères F5 algorithm with reduced Gröbner bases*, J. of Symb. Comp., **45**, 12, (2010), 1442-1458.

[EPP] C. Eder, G. Pfister, A. Popescu *New Strategies for Standard Bases over Z*, (2016), arXiv:AC/1609.04257.

[El] R. Elkik, *Solutions d'equations a coefficients dans un anneaux henselien*, Ann. Sci. Ecole Normale Sup., **6**, (1973), 553-604.

[F] J.-C. Faugère, *A new efficient algorithm for computing Gröbner bases without reduction to zero F5*, ISSAC 2002, (2002), 75-82.

[G] H. Grauert, *Über die Deformation isolierter Singularitäten analytischer Mengen*, Inventiones Mathematicae, **15**, (1972), 171-198.

[Gr] M. Greenberg, *Rational points in henselian discrete valuation rings*, Publ. Math. IHES, **31**, (1966), 59-64.

[GP] G.-M. Greuel, G. Pfister, *A* SINGULAR *Introduction to Commutative Algebra*, Springer-Verlag, (2008).

[GD] A. Grothedieck, J. Dieudonne, *Elements de geometrie algebrique, IV, Part 1*, Publ. Math. IHES, (1966).

[H] J. Herzog, *A Survey on Stanley Depth*, Lecture Notes in Mathematics, Springer-Verlag, (2013).

[HPV] J. Herzog, D. Popescu, M. Vlădoiu, *Stanley depth and size of a monomial ideal,* Proc. Amer. Math. Soc., **140**, (2012), 493-504.

[HSY] J. Herzog, A. Soleyman Jahann, S. Yassemi, *Stanley decompositions and partitionable simplicial complexes,* J. Alg. Comb., 27, (2008), 113-125.

[HVZ] J. Herzog, M. Vlădoiu, X. Zheng, *How to compute the Stanley depth of a monomial ideal,* J. Algebra, **322**, (2009), 3151-3169.

[Hi] H. Hironaka, *Resolution of Singularities of an Algebraic Variety over a Field of Characteristic Zero,* Annals of Mathematics, **79**, (1964).

[IKM] B. Ichim, L. Katthän, J. J. Moyano-Fernández, *The behaviour of Stanley depth under polarization*, J. Comb. Theory, Series A, **135**, (2015), 332-347.

[IM] B. Ichim, J. J. Moyano-Fernández, *How to compute the multigraded Hilbert depth of a module*, Math. Nachr. **287**, No. 11-12, (2014), 1274-1287, arXiv:AC/1209.0084v3.

[IZ] B. Ichim, A. Zarojanu, *An algorithm for computing the multigraded Hilbert depth of a module,* Experimental Mathematics, **23**:3, (2014), 322-331, arXiv:AC/1304.7215v3.

[Is] M. Ishaq, *Upper bounds for the Stanley depth*, Comm. in Alg., **40**, (2012), 87-97.

[IsQu] M. Ishaq, M. I. Qureshi, *Upper and lower bounds for the Stanley depth of certain classes of monomial ideals and their residue class rings*, Comm. in Alg., Volume **41**, (2013), 1107-1116.

[Ka] L. Katthän, *Stanley depth and simplicial spanning trees*, J. of Algebraic Combinatorics, **42**, (2015), 507-536.

[KMPPR] H. Kurke, T. Mostowski, G. Pfister, D. Popescu, M. Roczen, *Die Approximationseigenschaft lokaler Ringe*, Springer Lect. Notes in Math., **634**, Springer-Verlag, Berlin-New York, (1978).

[Li] D. Lichtblau, *Effective computation of strong Gröbner bases over Euclidean domains*, Illinois J. of Math. **56(1)**, (2012), 177-194.

[Ly] G. Lyubeznik, *On the Arithmetical Rank of Monomial ideals*, J. of Algebra **112**, (1988), 86-89.

[MS] J. Murdock, J. Sanders, *A new transvectant algorithm for nilpotent normal forms,* J. of Diff. Eq., **238**(1), (2007), 234-256.

[Ne] A. Neron, *Modeles minimaux des varietes abeliennes sur les corps locaux et globaux*, Publ. Math. IHES, **21**, (1964).

[OY] R. Okazaki, K. Yanagawa, *Alexander duality and Stanley depth of multigraded modules*, J. of Algebra **340**, (2011), 35-52.

[PfPo1] G. Pfister, D. Popescu, *Die strenge Approximationseigenschaft lokaler Ringe*, Inventiones Math., **30**, (1975),145-174.

[PfPo2] G. Pfister, D. Popescu, *Constructive General Neron Desingularization for one dimensional local rings*, to appear in J. of Symb. Comp., arXiv:AC/1512.08435.

[AP1] A. Popescu, *Special Stanley Decompositions*, Bull. Math. Soc. Sc. Math. Roumanie, **53(101)**, (2010), 363-372, arXiv:AC/1008.2924v2.

[AP2] A. Popescu, *An algorithm to compute the Hilbert depth* , J. of Symb. Comput., **66**, (2015), 1-7, arXiv:AC/1307.6084v3.

[AP3] A. Popescu, *Depth and Stanley depth of the canonical form of a factor of monomial ideals*, Bull. Math. Soc. Sc. Math. Roumanie, **57(105)**, (2014), 207-216, arXiv:AC/1402.5826v3

[APDP1] A. Popescu, D. Popescu, *Four generated, squarefree, monomial ideals* , in "Bridging Algebra, Geometry, and Topology", Editors Denis Ibadula, Willem Veys, Springer Proceed. in Math., and Statistics, **96**, (2014), 231-248, arXiv:AC/1309.4986v5.

[APDP2] A. Popescu, D. Popescu, *A method to compute the General Neron Desingularization in the frame of one dimensional local domains*, to appear in *Singularities and Computer Algebra - Festschrift for Gert-Martin Greuel, On the occasion of his 70th birthday*, Editors W. Decker, G. Pfister, M. Schulze, Springer Monograph.

[DP1] D. Popescu, *General Neron Desingularization*, Nagoya Math. J., **100**, (1985), 97-126.

[DP2] D. Popescu, *General Neron Desingularization and approximation*, Nagoya Math. J., **104**, (1986), 85-115.

[DP3] D. Popescu, *Letter to the Editor. General Neron Desingularization and approximation*, Nagoya Math. J., **118**, (1990), 45-53.

[DP4] D. Popescu, *Artin Approximation*, in "Handbook of Algebra", vol. 2, Ed. M. Hazewinkel, Elsevier, (2000), 321-355.

[DP5] D. Popescu, *Depth of factors of square free monomial ideals*, Proceedings of AMS **142**, (2014), 1965-1972, arXiv:AC/1110.1963v5.

[DP6] D. Popescu, *Upper bounds of depth of monomial ideals*, J. Commutative Algebra, **5**, (2013), 323-327, arXiv:AC/1206.3977.

[DP7] D. Popescu, *Stanley conjecture on intersections of four monomial prime ideals*, Comm. in Alg. **41**, (2010), 4351-4362.

[DP8] D. Popescu, *Around General Neron Desingularization*, J. of Alg. and its Appl., **16**,(2017).

[PQ] D. Popescu, I. Qureshi, *Computing the Stanley depth*, J. Algebra, **323**, (2010), 2943-2959.

[PZ] D. Popescu, A. Zarojanu, *Three generated, squarefree, monomial ideals,* Bull. Math. Soc. Sci. Math. Roumanie, **58(106)**, (2015), no 3, 359-368, arXiv:AC/1307.8292v6.

[Ra] A. Rauf, *Depth and Stanley depth of multigraded modules*, Comm. Algebra, **38**, (2010), 773-784.

[RTU] G. Restuccia, Z. Tang, R. Utano, *Stanley Conjecture on monomial ideals of mixed products*, J. Comm. Alg., **7**, (2015), 77-85.

[Ri] G. Rinaldo, *An algorithm to compute the Stanley depth of monomial ideals*, Le Matematiche, Vol. LXIII, (2008), 243-256.

[Ro] G. Rond, *Sur la linearite de la fonction de Artin*, Ann. Sci. Ecole Norm. Sup., (4), **38**, (2005), 979-988.

[S1] H. Schönemann, *Polynomials in Factory and Singular*, Nikolaus-Konferenz Aachen, (1999).

[S2] H. Schönemann, *Extending Singular with new types and algorithms*, International Congress on Mathematical Software (ICMS), (2016).

[Sh] Y.H. Shen, *Lexsegment ideals of Hilbert depth 1*, (2012), available at arXiv:AC/1208.1822v1.

[St] R. P. Stanley, *Linear Diophantine equations and local cohomology*, Invent. Math., **68**, (1982), 175-193.

[SW] B. Sturmfels, N. White, *Stanley decompositions of the bracket ring*, Math. Scand., **67**, (1990), 183-189.

[Sw] R. Swan, *Neron-Popescu desingularization*, in "Algebra and Geometry", Ed. M. Kang, International Press, Cambridge, (1998), 135-192.

[Ta] Z. Tang, *Stanley depth of certain Stanley-Reisner rings*, J. Algebra, **409**, (2014), 430-443.

[Uli] J. Uliczka, *Remarks on Hilbert series of graded modules over polynomial rings*, Manuscripta Math., **132** (2010), 159-168.

[Vi] R. H. Villarreal, *Monomial Algebras*, Marcel Dekker Inc., New York, (2001).

[W] O. Wienand, *Algorithms for Symbolic Computation and their Applications - Standard Bases over Rings and Rank Tests in Statistics*, PhD thesis, University of Kaiserslautern, (2011).

[Y] K. Yanagawa, *Sliding functor and polarization functor for multigraded modules*, Cummunications in Algebra, **40**, (2012), 1151-1166.

Wissenschaftlicher Werdegang

2008	Abitur, *Colegiul Naţional Sfântul Sava*, Bukarest, Rumänien
2008 - 2011	Bachelor im Mathematik, *Universität Bukarest*, Rumänien
2008 - 2011	Level 1: Zertifikat für Lehramt, *Universität Bukarest*, Rumänien
2011 - 2013	Master im Mathematik, *Technische Universität Kaiserslautern*
2013 - 2016	Promotion im Mathematik , *Technische Universität Kaiserslautern*

Curriculum Vitae

2008	Abitur, *Colegiul Naţional Sfântul Sava*, Bucharest, Romania
2008 - 2011	Bachelor in Mathematics, *University of Bucharest*, Romania
2008 - 2011	Certification for teaching (I), *University of Bucharest*, Romania
2011 - 2013	Master in Mathematics, *University of Kaiserslautern*, Germany
2013 - 2016	PhD in Mathematics , *University of Kaiserslautern*, Germany