Computation of the (n-1)-st Koszul Homology of Monomial Ideals and Related Algorithms

Anna M. Bigatti Eduardo Sáenz-de-Cabezón

Dipartimento di Matematica
Universita degli Studi Genova
bigatti@dima.unige.it
Departamento de Matemáticas y Computación
Universidad de La Rioja
eduardo.saenz-de-cabezon@unirioja.es

July - 29th - 2008

Outline

Outline

Outline

Outline

Outline

Koszul homology

- Given a monomial ideal I its Koszul homology is the homology of the complex

$$\mathbb{K}(I) := I \otimes \mathbb{K}$$

where $\mathbb{K}$ is the usual Koszul complex.

- This homology inherits the grading and multigrading of I.
- Fundamental equalities:

$$\beta_{i,\mathbf{a}}(I) = dim(Tor^R_{i,\mathbf{a}}(\mathbf{k}, I)) = dim(Tor^R_{i,\mathbf{a}}(I, \mathbf{k})) = dim_{\mathbf{k}}(H_{i,\mathbf{a}}(\mathbb{K}(I)))$$

where $i \in \mathbb{N}$, $\mathbf{a} \in \mathbb{N}^n$.

Koszul homology

- Given a monomial ideal I its Koszul homology is the homology of the complex

$$\mathbb{K}(I) := I \otimes \mathbb{K}$$

  where $\mathbb{K}$ is the usual Koszul complex.
- This homology inherits the grading and multigrading of I.
- Fundamental equalities:

$$\beta_{i,\mathbf{a}}(I) = dim(Tor^R_{i,\mathbf{a}}(\mathbf{k}, I)) = dim(Tor^R_{i,\mathbf{a}}(I, \mathbf{k})) = dim_{\mathbf{k}}(H_{i,\mathbf{a}}(\mathbb{K}(I)))$$

  where $i \in \mathbb{N}$, $\mathbf{a} \in \mathbb{N}^n$.

Koszul homology

- Given a monomial ideal I its Koszul homology is the homology of the complex

$$\mathbb{K}(I) := I \otimes \mathbb{K}$$

where $\mathbb{K}$ is the usual Koszul complex.

- This homology inherits the grading and multigrading of I.
- Fundamental equalities:

$$\beta_{i,\mathbf{a}}(I) = dim(Tor^R_{i,\mathbf{a}}(\mathbf{k}, I)) = dim(Tor^R_{i,\mathbf{a}}(I, \mathbf{k})) = dim_{\mathbf{k}}(H_{i,\mathbf{a}}(\mathbb{K}(I)))$$

where $i \in \mathbb{N}$, $\mathbf{a} \in \mathbb{N}^n$.

Koszul homology

- Given a monomial ideal I its Koszul homology is the homology of the complex

$$\mathbb{K}(I) := I \otimes \mathbb{K}$$

where $\mathbb{K}$ is the usual Koszul complex.

- This homology inherits the grading and multigrading of I.
- Fundamental equalities:

$$\beta_{i,\mathbf{a}}(I) = dim(Tor^R_{i,\mathbf{a}}(\mathbf{k}, I)) = dim(Tor^R_{i,\mathbf{a}}(I, \mathbf{k})) = dim_{\mathbf{k}}(H_{i,\mathbf{a}}(\mathbb{K}(I)))$$

where $i \in \mathbb{N}$, $\mathbf{a} \in \mathbb{N}^n$.

$(n-1)$st Koszul homology module

The set of multidegrees with nonzero $(n-1)$-st homology $\mathcal{B}_{n-1}(I)$ provides:

-
-
-
-
-

$(n-1)$st Koszul homology module

The set of multidegrees with nonzero $(n-1)$-st homology $\mathcal{B}_{n-1}(I)$ provides:

- Irreducible components
-
-
-
-

$(n-1)$st Koszul homology module

The set of multidegrees with nonzero $(n-1)$-st homology $\mathcal{B}_{n-1}(I)$ provides:

- Irreducible components
- Maximal standard monomials
-
-
-

$(n-1)$st Koszul homology module

The set of multidegrees with nonzero $(n-1)$-st homology $\mathcal{B}_{n-1}(I)$ provides:

- Irreducible components
- Maximal standard monomials
- Alexander dual
-
-

$(n-1)$st Koszul homology module

The set of multidegrees with nonzero $(n-1)$-st homology $\mathcal{B}_{n-1}(I)$ provides:

- Irreducible components
- Maximal standard monomials
- Alexander dual
- Scarf complex
-

$(n-1)$st Koszul homology module

The set of multidegrees with nonzero $(n-1)$-st homology $\mathcal{B}_{n-1}(I)$ provides:

- Irreducible components
- Maximal standard monomials
- Alexander dual
- Scarf complex
- ...

Standard result (proof: Hochster's formula, for instance):

$$H_{n-1,\mu}(\mathbb{K}(I)) = \begin{cases} \mathbf{k} & \text{if } \frac{x_i \cdot x^\mu}{x_1 \cdots x_n} \in I \; \forall i \text{ and } \frac{x^\mu}{x_1 \cdots x_n} \notin I \\ 0 & \text{in any other case} \end{cases}$$

Together with the fact that $\mu \notin L_I \Rightarrow \beta_{i,\mu}(I) = 0 \; \forall i$ we have a finite algorithm to compute $\mathcal{B}_{n-1}(I)$.

Standard result (proof: Hochster's formula, for instance):

$$H_{n-1,\mu}(\mathbb{K}(I)) = \begin{cases} \mathbf{k} & \text{if } \frac{x_i \cdot x^\mu}{x_1 \cdots x_n} \in I \; \forall i \text{ and } \frac{x^\mu}{x_1 \cdots x_n} \notin I \\ 0 & \text{in any other case} \end{cases}$$

Together with the fact that $\mu \notin L_I \Rightarrow \beta_{i,\mu}(I) = 0 \; \forall i$ we have a finite algorithm to compute $\mathcal{B}_{n-1}(I)$.

Irreducible decomposition I: Preparatory results

Lemma

Let x^μ a monomial and let C be the complement of $supp(x^\mu)$ in the set of all variables. Then, for any monomial x^ν

$$x^\nu \notin \mathfrak{m}^\mu \iff x^\nu \in \bigcup_{x^\rho | x^{\mu'}} x^\rho \cdot \mathbf{k}[C]$$

Corollary

Let x^μ and x^ν two different monomials such that x^μ has full support. Then

$$x^\nu | x^{\mu'} \iff x^\nu \notin \mathfrak{m}^\mu$$

Irreducible decomposition I: Preparatory results

Lemma

Let x^μ a monomial and let C be the complement of $supp(x^\mu)$ in the set of all variables. Then, for any monomial x^ν

$$x^\nu \notin \mathfrak{m}^\mu \iff x^\nu \in \bigcup_{x^\rho | x^{\mu'}} x^\rho \cdot \mathbf{k}[C]$$

Corollary

Let x^μ and x^ν two different monomials such that x^μ has full support. Then

$$x^\nu | x^{\mu'} \iff x^\nu \notin \mathfrak{m}^\mu$$

Irreducible decomposition II: Artinian case

Proposition

Let I be an artinian monomial ideal and let $\mathcal{B}_{n-1}(I)$ the set of multidegrees in which I has nonzero $(n-1)$-st Koszul homology (i.e. the maximal corners of I). The irredundant irreducible decomposition of I is

$$I = \bigcap_{\mu \in \mathcal{B}_{n-1}(I)} \mathfrak{m}^{\mu}$$

Irreducible decomposition III: Non-artinian case

Proposition

Let I be a monomial ideal and $\hat{I}$ its artinian closure. The irredundant irreducible decomposition of I is given by

$$I = \bigcap_{\mu \in \mathcal{B}_{n-1}(\hat{I})} \mathfrak{m}^{\tilde{\mu}}$$

where $\tilde{\mu}_i = 0$ if $\mu_i \gneq \lambda_{i+1}$ and $\tilde{\mu}_i = \mu_i$ in other case.

where $\lambda_i = lcm(mingens(I))_i$.

Stanley decomposition I:Definitions

$$R/I = \bigoplus_{\mu \in \mathcal{F}} x^{\mu} \mathbf{k}[\mathbf{x}_{\mu}] \tag{1}$$

Definition

We say that a set of variables $\{x_{j_1}, \ldots, x_{j_k}\}$ is a cone of locally free directions of the monomial x^{μ} with respect to I if $\tau = \{j_1, \ldots, j_k\} \in \Delta_I^{\mu}$. The set of cones of locally free directions of x^{μ} will be denoted $LF(x^{\mu})$, and is given by the facets of Δ_I^{μ}. We say that a set of variables $\{x_{j_1}, \ldots, x_{j_k}\}$ is a cone of true free directions of the monomial $x^{\mu} \notin I$ with respect to I if $x^{\mu} \cdot x^{\sigma} \notin I$ for all monomials $x^{\sigma} \in \mathbf{k}[x_{j_1}, \ldots, x_{j_k}]$. The set of cones of true free directions of x^{μ} will be denoted $TF(x^{\mu})$. For $x^{\mu} \in I$ we state $TF(x^{\mu}) := TF(x^{\mu'})$.

Stanley decomposition II: Artinian case

Proposition

Let I be an artinian monomial ideal. A Stanley decomposition of R/I is given by

$$R/I \simeq \bigoplus_{\substack{x^\nu | x^{\mu'} \\ x^\mu \in \mathcal{B}_{n-1}(I)}} \mathbf{k} \cdot x^\mu$$

Stanley decomposition III: Non-artinian case

Proposition

Let I a monomial ideal and let $\hat{I}$ its artinian closure. There is a procedure to obtain a Stanley decomposition of I form the $(n-1)$-st Koszul homology of $\hat{I}$.

1. First we compute the Stanley decomposition of $\hat{I}$ using the previous proposition. Since $R/\hat{I} \subseteq R/I$, this is part of the Stanley decomposition of R/I, we call this part the inner part of the decomposition.
2. Take now $x^\mu \in \mathcal{B}_{n-1}(\hat{I})$ such that $\mu_i \geq \lambda_i + 1$ for some $i \in \{1, \ldots, n\}$. We call points of the skeleton (of the decomposition of R/I) to the monomials $\frac{x^\tau \cdot x^\mu}{x_1 \cdots x_n}$ such that $\tau \in \Delta_\nu^I$ and also to the monomials that are divisors of $\frac{x^\tau \cdot x^\mu}{x_1 \cdots x_n}$ in the nonfree directions of $\{x_1, \ldots, x_n\} \setminus \tau$
3. To obtain the Stanley decomposition we add the cones of the points of the skeleton in all their free directions.

MVT algorithm

- Gives the support of the mapping cone resolution of I (monomial ideal).
- Avoids computation of the differentials
- Allows further reduction steps if the cone resolution is not minimal.
- Analysis tool that gives minimal resolution (i.e. Betti numbers) in many cases.

MVT algorithm

- Gives the support of the mapping cone resolution of I (monomial ideal).
- Avoids computation of the differentials
- Allows further reduction steps if the cone resolution is not minimal.
- Analysis tool that gives minimal resolution (i.e. Betti numbers) in many cases.

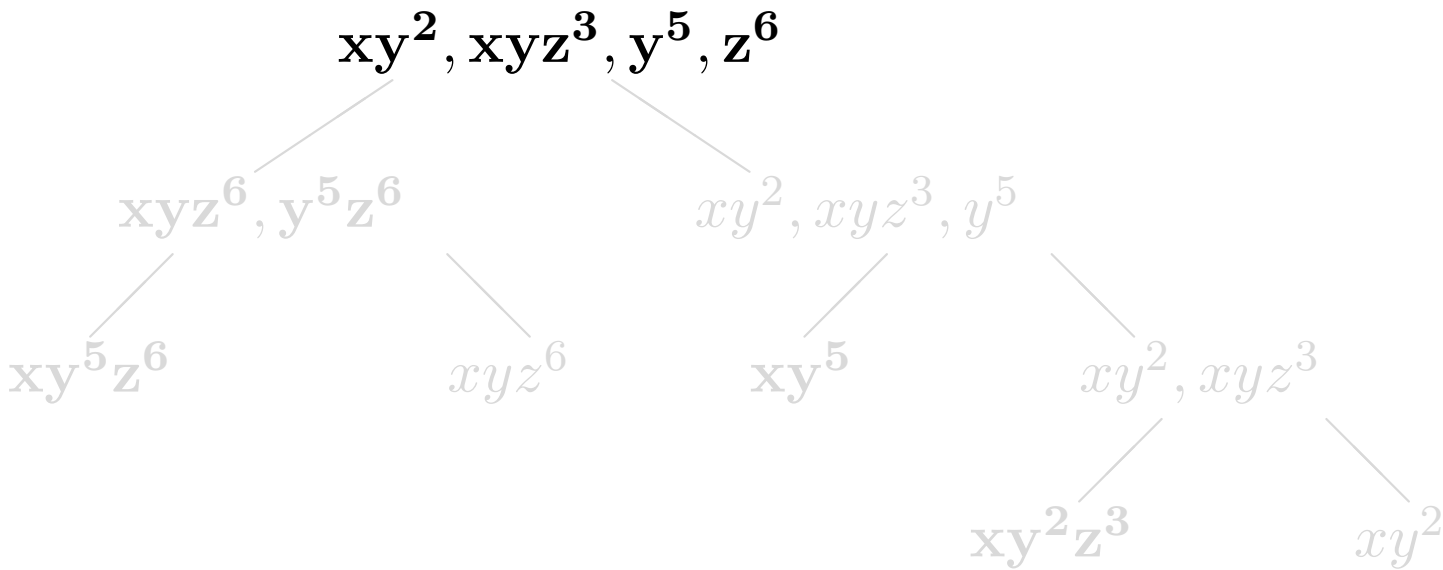
MVT algorithm

- Gives the support of the mapping cone resolution of I (monomial ideal).
- Avoids computation of the differentials
- Allows further reduction steps if the cone resolution is not minimal.
- Analysis tool that gives minimal resolution (i.e. Betti numbers) in many cases.

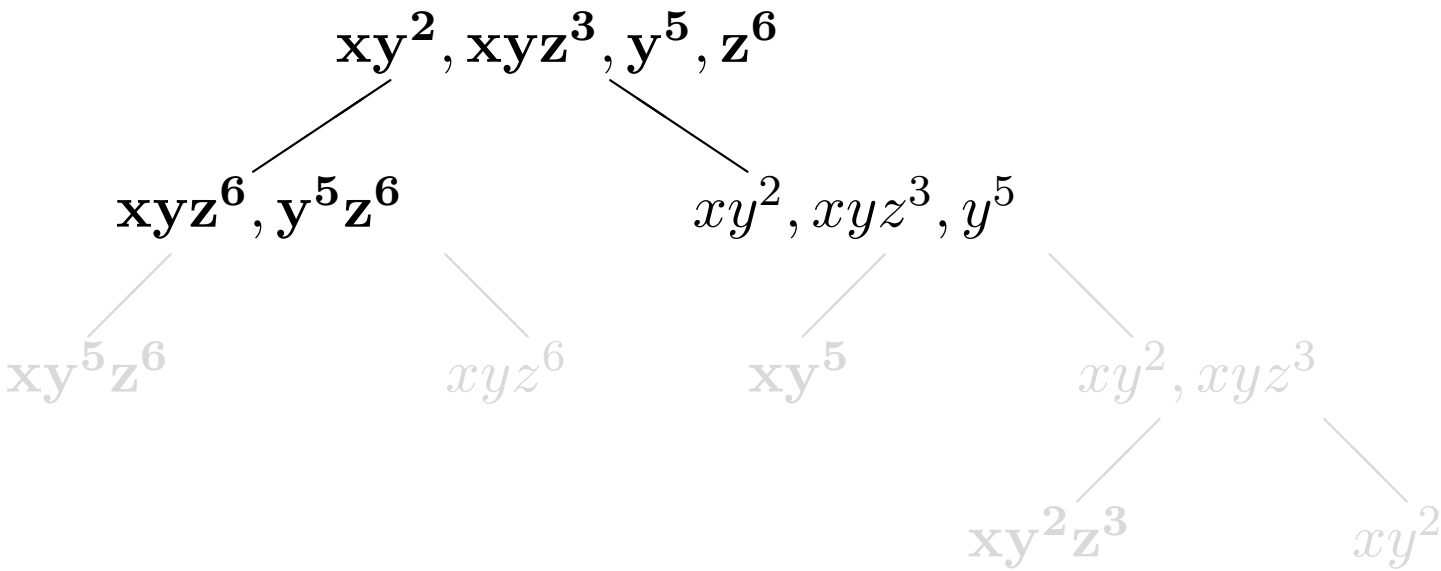
MVT algorithm

- Gives the support of the mapping cone resolution of I (monomial ideal).
- Avoids computation of the differentials
- Allows further reduction steps if the cone resolution is not minimal.
- Analysis tool that gives minimal resolution (i.e. Betti numbers) in many cases.

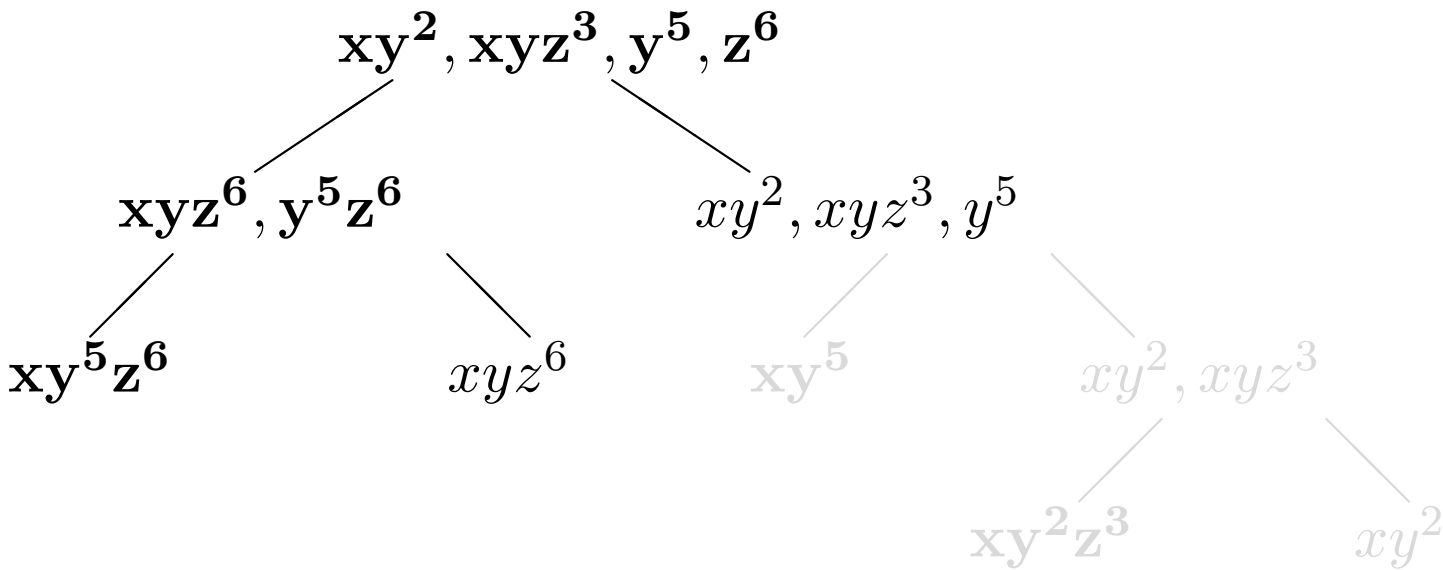
Example


$\mathbf{xy^2, xyz^3, y^5, z^6}$
$\mathbf{xyz^6, y^5z^6}$
xy^2, xyz^3, y^5
$\mathbf{xy^5z^6}$
xyz^6
$\mathbf{xy^5}$
xy^2, xyz^3
$\mathbf{xy^2z^3}$
xy^2

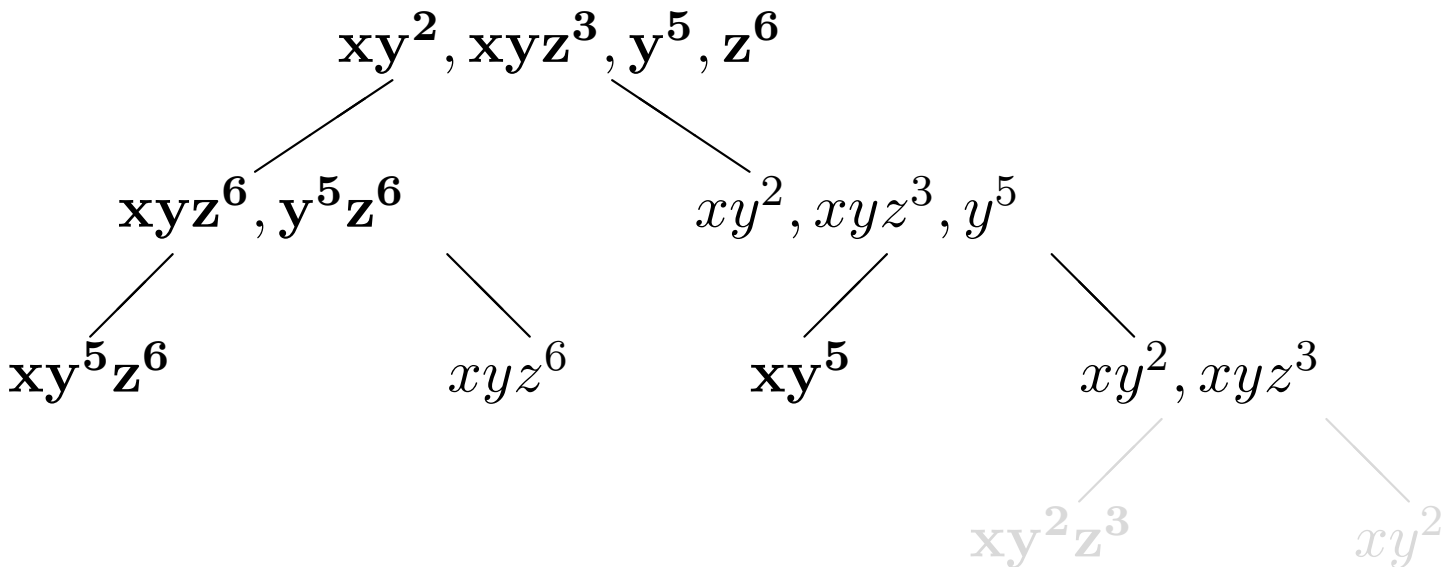
Example


$\mathbf{xy^2, xyz^3, y^5, z^6}$
$\mathbf{xyz^6, y^5z^6}$
xy^2, xyz^3, y^5
$\mathbf{xy^5z^6}$
xyz^6
$\mathbf{xy^5}$
xy^2, xyz^3
$\mathbf{xy^2z^3}$
xy^2

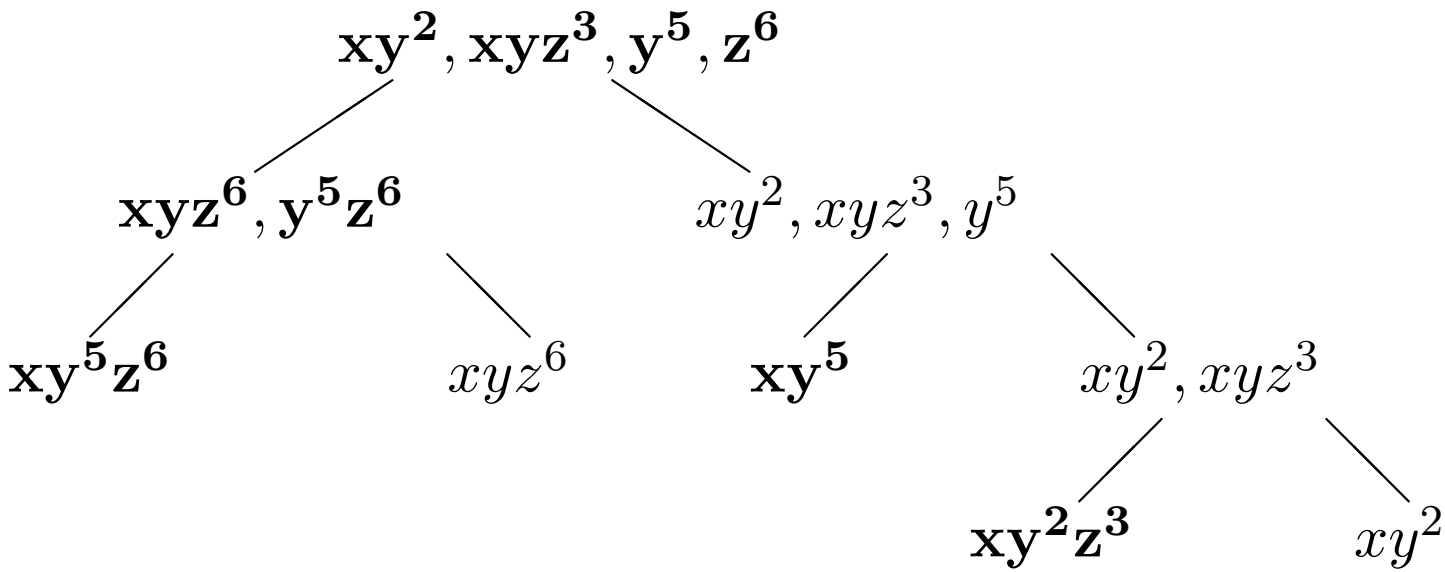
Example


$\mathbf{xy^2, xyz^3, y^5, z^6}$
$\mathbf{xyz^6, y^5z^6}$
xy^2, xyz^3, y^5
$\mathbf{xy^5z^6}$
xyz^6
$\mathbf{xy^5}$
xy^2, xyz^3
$\mathbf{xy^2z^3}$
xy^2

Example


$\mathbf{xy^2, xyz^3, y^5, z^6}$
$\mathbf{xyz^6, y^5z^6}$
xy^2, xyz^3, y^5
$\mathbf{xy^5z^6}$
xyz^6
$\mathbf{xy^5}$
xy^2, xyz^3
$\mathbf{xy^2z^3}$
xy^2

Example


$\mathbf{xy^2, xyz^3, y^5, z^6}$
$\mathbf{xyz^6, y^5z^6}$
xy^2, xyz^3, y^5
$\mathbf{xy^5z^6}$
xyz^6
$\mathbf{xy^5}$
xy^2, xyz^3
$\mathbf{xy^2z^3}$
xy^2

Adaptation to the computation of $\mathcal{B}_{n-1}$

- Pruning by number of generators.
- Pruning by number of variables.
- Result is a small set of candidates to be elements of $\mathcal{B}_{n-1}$.
- Apply test to have $(n-1)$-st homology. Always decides.

Adaptation to the computation of $\mathcal{B}_{n-1}$

- Pruning by number of generators.
- Pruning by number of variables.
- Result is a small set of candidates to be elements of $\mathcal{B}_{n-1}$.
- Apply test to have $(n-1)$-st homology. Always decides.

Adaptation to the computation of $\mathcal{B}_{n-1}$

- Pruning by number of generators.
- Pruning by number of variables.
- Result is a small set of candidates to be elements of $\mathcal{B}_{n-1}$.
- Apply test to have $(n-1)$-st homology. Always decides.

Adaptation to the computation of $\mathcal{B}_{n-1}$

- Pruning by number of generators.
- Pruning by number of variables.
- Result is a small set of candidates to be elements of $\mathcal{B}_{n-1}$.
- Apply test to have $(n-1)$-st homology. Always decides.

The CoCoA “family” (http://cocoa.dima.unige.it)

- CoCoA-4: current system 4.7
- CoCoALib: C++ library.
- CoCoAServer “server program” coupled with CoCoA-4, gives access to some features of CoCoALib. Easily extensible.
- CoCoA-5: future system whose core will be CoCoALib, extended language and capabilities.

CoCoALib. Current state.

- CoCoALib offers, among other things, good data types for monomial ideals.
- CoCoALib is relatively easy to use
- This implementation is distributed with CoCoALib
- CoCoAServer is easily extensible, so this implementation (and also Frobby) is accessible from CoCoA-4

Accessible via prototype CoCoAServer from CoCoA-4.
Code developed on GNU/Linux machines and MacOS X.
GMP used for big integer arithmetic and high precision floats.

Implementation of the adaptation of MVT algorithm to the computation of $(n-1)$st Koszul homology shows good performance in practice.

Table: Irreducible decomposition of random generic ideals in ten variables. Source: [Roune 08]

$\lvert min(I)\rvert$	$\lvert irr(I)\rvert$	Macaulay2	Monos(Alex.)	Monos(Scarf)	Frobby	MVT
40	52131	226s	521s	10 s	1s	2.52s
80	163162	OOM	OOT	54s	4s	8.88s
120	411997	OOM	OOT	198s	9s	24.27s
160	789687	OOM	-	563s	19s	50.78s
200	1245139	OOM	-	OOM	29s	86.50s

Afterwords

There seems to be a lack of balance between the importance of monomial ideals and the theoretical methods to deal with them on one side and the presence of specific algorithms targeted to monomial ideals in today's Computer Algebra Systems on the other.

We showed an example in which the combinatorial nature of monomial ideals can be used to relate different kind of computations (homological-combinatorial-algebraic).

The algorithms we presented have been implemented using the `C++` library `CoCoALib` which brings together the capabilities of the `C++` programming language with structures and procedures from commutative algebra.

Even if our focus on these implementations is on simplicity rather than on efficiency, the direct use of the `C++` language using `CoCoALib` allows to obtain good performance with relatively small programming efforts.

Afterwords

There seems to be a lack of balance between the importance of monomial ideals and the theoretical methods to deal with them on one side and the presence of specific algorithms targeted to monomial ideals in today's Computer Algebra Systems on the other.

We showed an example in which the combinatorial nature of monomial ideals can be used to relate different kind of computations (homological-combinatorial-algebraic).

The algorithms we presented have been implemented using the `C++` library `CoCoALib` which brings together the capabilities of the `C++` programming language with structures and procedures from commutative algebra.

Even if our focus on these implementations is on simplicity rather than on efficiency, the direct use of the `C++` language using `CoCoALib` allows to obtain good performance with relatively small programming efforts.

Afterwords

There seems to be a lack of balance between the importance of monomial ideals and the theoretical methods to deal with them on one side and the presence of specific algorithms targeted to monomial ideals in today's Computer Algebra Systems on the other.
We showed an example in which the combinatorial nature of monomial ideals can be used to relate different kind of computations (homological-combinatorial-algebraic).
The algorithms we presented have been implemented using the `C++` library `CoCoALib` which brings together the capabilities of the `C++` programming language with structures and procedures from commutative algebra.
Even if our focus on these implementations is on simplicity rather than on efficiency, the direct use of the `C++` language using `CoCoALib` allows to obtain good performance with relatively small programming efforts.

Afterwords

There seems to be a lack of balance between the importance of monomial ideals and the theoretical methods to deal with them on one side and the presence of specific algorithms targeted to monomial ideals in today's Computer Algebra Systems on the other.
We showed an example in which the combinatorial nature of monomial ideals can be used to relate different kind of computations (homological-combinatorial-algebraic).
The algorithms we presented have been implemented using the `C++` library `CoCoALib` which brings together the capabilities of the `C++` programming language with structures and procedures from commutative algebra.
Even if our focus on these implementations is on simplicity rather than on efficiency, the direct use of the `C++` language using `CoCoALib` allows to obtain good performance with relatively small programming efforts.