UNIVERSITÄT KAISERSLAUTERN
Zentrum für Computeralgebra

REPORTS ON COMPUTER ALGEBRA
NO. 08

On strategies and implementations for computations of free resolutions

by

T. Siebert

September 1996

The Zentrum für Computeralgebra (Centre for Computer Algebra) at the University of Kaiserslautern was founded in June 1993 by the Ministerium für Wissenschaft und Weiterbildung in Rheinland-Pfalz (Ministry of Science and Education of the state of Rheinland-Pfalz). The centre is a scientific institution of the departments of **Mathematics, Computer Science, and Electrical Engineering** at the University of Kaiserslautern.
The goals of the centre are to advance and to support the use of Computer Algebra in industry, research, and teaching. More concrete goals of the centre include

- the development, integration, and use of software for Computer Algebra
- the development of curricula in Computer Algebra under special consideration of interdisciplinary aspects
- the realisation of seminars about Computer Algebra
- the cooperation with other centres and institutions which have similar goals

The present coordinator of the Reports on Computer Algebra is:
Olaf Bachmann (email: `obachman@mathematik.uni-kl.de`)

Zentrum für Computeralgebra
c/o Prof. Dr. G.-M. Greuel, FB Mathematik
Erwin-Schrödinger-Strasse
D-67663 Kaiserslautern; Germany
Phone: 49 - 631/205-2850   Fax: 49 - 631/205-5052
email: `greuel@mathematik.uni-kl.de`
URL: `http://www.mathematik.uni-kl.de/~zca/`

On strategies and implementations for computations of free resolutions

Thomas Siebert

September 1996

1 Introduction

In computational algebraic geometry as well as in the theory of singularities it is a fundamental tool to compute free resolutions of ideals and modules. For example, for the computation of homology and cohomology groups, for Betti numbers and other invariants. Therefore, it is necessary to provide a computer algebra system with a fast implementation of syzygies.

In recent years several methods were developed to speed up computations of syzygies. Here I would like to give an overview of the essential ideas and our practical experiences with them.

Main contributions to that development come from W. Trinks, G. Zacharias, D.A. Spear, F. O. Schreyer, M. Stillman and D. Mumford, T. Mora and Moeller. Recent improvements to compute minimal resolutions directly have been created by C. Traverso, M. Caboara, R. La Scala and by myself.

In Chapter 2.2, as a new result, it is proved that the STZS-algorithm apply also to factor rings. Further, the technique of direct reductions is used to avoid Gaussian eliminations for computations in local orderings. It is described how detailed informations on Hilbert functions of the modules of a minimal resolutions can be obtained.

At the end suggestions for the algorithms for a global homogeneous local ordering are given.

I would like to thank G. Pfister for many fruitful discussions helping me to study this subject and the DFG for giving financial support during that time.

2 The ground algorithms

2.1 The common algorithm

The common algorithm stores the reduction process of a standard basis computation and obtains in this way a basis for the module of syzygies. Assume that an R-module $M \subset R^k$ is given by the columns of a matrix:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

Then we add a new module component to each element of the given basis. The extra component then notices what happens with its corresponding element:

$$\begin{pmatrix} & & & & \\ & & A & & \\ & & & & \\ - & - & - & - & - \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \Longrightarrow \left(\begin{array}{c|c} A' & 0 \\ \hline B' & C' \end{array}\right).$$

After a standard basis computation we obtain a matrix like the right one, where A' corresponds to the standard basis of A, B' is a transformation matrix, i.e., $A' = A * B'$ and C' represents a basis of the module syzygies of A. In fact, every syzygy constitutes a certain way to obtain 0 as linear combination of the given generators. In this way, we find a basis of all syzygies. Otherwise a syzygy, which is not included, will produce elements with new leading terms.

Following this idea we obtain an easy implementation of this algoritm:

$W :=$ **Syz(S)**

INPUT: S, a basis of an ideal

OUTPUT: S' a minimal set of generators; W' the module of syzygies of S'

> $W := \emptyset$
> $T :=$ `initSyz` (S)
> $T' :=$ `std` (S)
> `setRegularity(T')`
> `WHILE` $T' \neq \emptyset$ `DO`
>
> > `IF t`$\in T'$ `is syzygy` W `:=` $W \cup \{$`t`$\}$
> > T' `:=` $T' \setminus \{$`t`$\}$
>
> `END`
> (S', W') `:= minimize(`S, W`).`

$T :=$ **initSyz(S)**

`INPUT:` S`, a standard basis` $v_1, \ldots, v_s$

`OUTPUT:` T`, the set` $[v_1, 1, 0, \ldots, 0], \ldots, [v_s, 0, \ldots, 0, 1]$

> $i := r$; $T := \emptyset$
> `S := interReduce(S)`
> `WHILE` $S \neq \emptyset$ `DO`
>
> > $s :=$ `the first element from` S
> > $i := i + 1$
> > $S := S \backslash \{s\}$
> > $s := s + e_i$
> > $T := T \cup \{s\}$
>
> `END`
> `setSyzygyOrder(r)`

$(\mathbf{W}', \mathbf{S}') :=$ **minimize(W, S)**

`INPUT:` S`, a basis,` W`, its syzygies`

`OUTPUT:` S'`, a minimal system of generators,` W'`, its syzygies`

> W' `:=` W
> S' `:=` S
> `WHILE searchUnit(`W'`,i,j) DO`
>
> > W' `:= Gauss(`W'`,i,j)`
> > S' `:=` $S' \setminus \{s_i\}$
>
> `END`

In the interreduction we subtract multiples of elements from others as long as any leading term divides another one. The interreduction of the input must always precede the adding of new module components. Otherwise this would result in a much more complicated structure of the module of syzygies.

In the part of minimization one searches for units of the ground ring in the entres of the matrix C' corresponding to the syzygies. Then one performs a Gaussian elimination with the associated row and cancels the row and the column where the unit was found at the end:

$$\begin{pmatrix} s_{11} & \ldots & s_{1j} & \ldots \\ \vdots & & \vdots & \\ s_{i1} & \ldots & u_1 & \ldots \\ \vdots & & \vdots & \end{pmatrix} \Longrightarrow \begin{pmatrix} s'_{11} & \ldots & s_{1j} & \ldots \\ \vdots & & \vdots & \\ 0 & \ldots & u_1 & 0 \\ \vdots & & \vdots & \end{pmatrix}$$

$$\Longrightarrow \begin{pmatrix} s'_{11} & \dots & s'_{1j-1} & s'_{1j+1} & \dots \\ \vdots & & \vdots & & \\ s'_{i-11} & & s'_{i-1j-1} & s'_{i-1j+1} & \\ s'_{i+11} & & s'_{i+1j-1} & s'_{i+1j+1} & \\ \vdots & & \vdots & \vdots & \end{pmatrix}.$$

Notice that for each Gaussian elimination one has to multiply all remaining columns with the unit u_1 which implies that the length of all polynomials grows at least by the number $l_1 = length(u_1)$ of monomials of u_1. In the case of global computaions all units are just constants. However, for local or mixed orderings these units are polynomials in the non global variables and the length of the polynomials grows during n Gaussian elemenations by:

$$l_1^n l_2^{n-1} \dots l_{n-1}^2 l_n,$$

where $l_i = length(u_i)$ is the number of monomials in the i-th unit at the begin of the minimization. To decrease this growth one might use Bareiss' algorithm, but this would imply additional polynomial divisions.

Our experience agrees with this observation: in a non-global case it is often the minimization which is the really hard part to compute a minimal resolution, whereas in the global case it is always the standard basis.

2.2 The STZS-algorithm

Let $S = \{g_1, \dots, g_q\}$ be a standard basis of $I \subseteq K[x]^r$.

For $K[x]^q = \sum_{i=1}^q K[x]e_i$ we choose the following **monomial-weighted ordering** $<_1$ (depending on S): $x^\alpha e_i <_1 x^\beta e_j$ if and only if either $L(x^\alpha g_i) < L(x^\beta g_j)$ or $L(x^\alpha g_i) = L(x^\beta g_j)$ and $i > j$.

For g_i, g_j, having the leading term in the same component, that is $L(g_i) = x^{\alpha_i} e_k, L(g_j) = x^{\alpha_j} e_k$ we consider $\text{spoly}(g_i, g_j) := m_{ji} g_i - m_{ij} g_j$ with $m_{ji} = c(g_j) \frac{lcm(L(g_i), L(g_j))}{x^{\alpha_i}}$.

Because S is a standard basis we obtain ([SI], Corollary 1.11)

$$(1 + h_{ij})(m_{ji} g_i - m_{ij} g_j) = \sum \xi_\nu^{ij} g_\nu$$

with $L(h_{ij}) < 1$ if $h_{ij} \neq 0$ and $L(\xi_\nu^{ij} g_\nu) < L(m_{ji} g_i)$.

For $j > i$ such that g_i, g_j have leading term in the same component, let

$$\tau_{ij} := (1 + h_{ij})(m_{ji} e_i - m_{ij} e_j) - \sum \xi_\nu^{ij} e_\nu.$$

Let $\ker(K[x]^q \to K[x]^r, \sum w_i e_i \mapsto \sum w_i g_i)$ denote the **module of syzygies, syz**(I), of $\{g_1, \dots, g_q\}$. The following proposition was proved independently by F. O. Schreyer, W. Trinks, G. Zacharias and D. A. Spear.

Proposition 2.1 *With respect to the ordering $<_1$ the following holds:*

1) $L(\tau_{ij}) = m_{ji} e_i$.

2) $\{\tau_{ij} \mid i < j$ s.t. $L(g_i), L(g_j)$ are in the same component $\}$ is a standard basis for syz(I).

For a proof see, for example, [S] or [SI].

The main advantage of this algorithm is that the spolynomials always reduce to zero and thus, the sets of elements and of pairs do not change during the computation. It also allows us to consider only pairs built from one fixed element at the same time which results in much smaller pair sets. The knowledge of the leading terms of the syzygies is used to avoid useless computations which means cancelling those pairs the syzygy of which has a leading term divisible by another.

Considering the n-th iterated module of this resolution with the (n-1)-th iterated monomial-weighted ordering, the leading term of an arbitrary STZS-syzygy multiplied with the corresponding weight-monomial is just the lcm of an n-tupel of initial terms of the input. Therefore, an n-th syzygy of the STZS-resolution corresponds to a critical n-tupel in the terminology of [MM], Part II. The reduction of the associated spolynomial to zero corresponds to the computation of the lift of the morphism of leading terms to the complete generators described in Lemma 7.6 there. However, the step-by-step inspection of prospective leading terms gives the opportunity to compute always relatively to a minimal G-resolution of the leading terms. Hence, one of the problems of [MM] and [MM1] disappears here as the STZS-resolution is a minimal T-resolution.

The STZS-algorithm works not only for polynomial rings or their localizations, but also for factor rings. Assume $A = k[\underline{x}]/I$ be the factor of a polynomial ring after a certain ideal and let $I = (f_1, \ldots, f_m)$ be given by a standard (Groebner) basis. Define the normal form with respect to I of an element of an arbitrary module to be the component-wise normal form.

Definition 2.2 *A subset G of a module $M \subset A^n$ is called a standard basis for M if*

- *the elements of G are given by representatives in normal form with respect to I and*
- *the leading term $lt(m)$ of any element $m \in M$ in normal form with respect to I is divisible by some leading term $lt(g)$ for $g \in G$.*

Assume $g_i \in G$ and $lt(g_i) = m * e_i$. Denote by kgv_{ij} the lowest common multiple of m and $lt(f_j)$. Then $(kgv_{ij}/m) * g_i - (kgv_{ij}/lt(f_j)) * f_j e_i \in M$ and, therefore, (G is standard basis!):

$$(1 + h_{ij})((kgv_{ij}/m) * g_i - (kgv_{ij}/lt(f_j)) * f_j) = \sum h_k * g_k \mod I,$$

with $h_k \in k[\underline{x}]$ and $lt(h_k * g_k) < kgv_{ij} * e_i$. Now, set

$$\nu_{ij} = (1 + h_{ij})(kgv_{ij}/m) * e_i - \sum h_k * e_k.$$

Proposition 2.3 *With respect to the ordering $<_1$ the following holds:*

1) $lt(\nu ij) = (kgv_{ij}/m)e_i$.

2) $lt(\tau ij) = m_{ji}e_i$.

3) $\{\tau_{ij}\} \cup \{\nu_{ij}\}$ is a standard basis for syz(M) as A-module.

Proof: Again, the main thing to show is that the leading term of an arbitrary syzygy in normal form is divisible by that of a certain STZS-syzygy. Assume $s = \sum_{l=1}^{r} h_l * e_l$ to be the choosen syzygy of M. Then $\sum_{l=1}^{r} h_l * g_l \in I * A^n$. Let $lt(s) = m * e_i$ be the leading term of (the normal form of) s with respect to the ordering $<_1$ on $k[\underline{x}]^r$. We distinguish two cases:

1. $m * lt(g_i) \in In(I)$. Let f_j be an element of I dividing $m * lt(g_i)$. Then also kgv_{ij} divides $m * lt(g_i)$ and, hence, $lt(\nu_{ij})$ divides $lt(s)$.
2. $m * lt(g_i) \notin In(I)$. There exists at least one other monomial $n * e_{i'}$ with $m * lt(g_i) = n * lt(g_{i'})$ and, following the arguments of the proof of the general theorem, one sees that $lt(\tau_{ii'})$ divides $lt(s)$.

For a fast implementation of this algorithm one has to handle the monomial-weighted orderings. These orderings give weights to the elements of the modules of syzygies by multiplying the corresponding components with certain monomials. Consider for example, [MM], 3.4 (ii).

If the ordering on the module checks first the ordering of the involved monomials and then their component, one can go a step further: Let $\tilde{F}$ denote the transformation of a given basis $F = \{\underline{f}\}$ of a module by multiplying all monomials of the i-th component by the monomial $lm(g_i)$ for all i. Then the next statement is more or less self-evident.

Lemma 2.4 *The set F is a standard basis of a module with regard to the monomial-weighted ordering $<_1$ if and only if $\tilde{F}$ is a standard basis in the given ordering $<$.*

Proof: One point to observe is that the map assigning i to the module component of $lm(g_i)$ must be monotone, i.e., it must not disturb the ordering of the monomials. An other is that in monomial-weighted orderings the comparison of monomials precedes that of module components. Therefore, the trick applies only to these orderings.

To implement monomial-weighted orderings this little lemma is very useful since one needs only to replace the "ones" of the unit matrix by the corresponding weight monomials in the initial procedure.

We have also experimented with a complete disregard to special orderings and a decision to reduce always by checking divisibility - which was an idea of F. O. Schreyer. This works as one is sure that all reduces to zero, but, in practice a divisibility check applied to all occurring monomials takes more time than a polynomial arithmetic dealing with polynomials as ordered lists.

We observed a great sensitivity against the ordering of the input and have found, for the homogeneous global case, the degree reverse lexicographical ordering beginning with the greater element to be definitively the best choice.

These considerations lead us to the following implementation:

Let S be a standard basis of $I \subseteq K[x]^r = \sum_{i=1}^{r} K[x]e_i$. We may assume that $L(s) \not| L(s')$ for different $s, s' \in S$. Let $q := \#S$.

$W :=$ **Syz(S)**

INPUT: S, a standard basis

OUTPUT: W, the module of syzygies of S

```
W := ∅
T := initSyz (S)
L := initPairs (T)
WHILE L ≠ ∅ DO

    p := (a,b,s) the last element from L
    L := L\{(a,b,s)}
    s := spoly(a,b)
    h := NF(s|T)
    W := W ∪ {h}

END
W := cancelWeightMonomials(W).
```

$T :=$ **initSyz(S)**

`INPUT:` S, a standard basis $v_1, \ldots, v_s$

`OUTPUT:` T, the set $[v_1, lm(v_1), 0, \ldots, 0], \ldots, [v_s, 0, \ldots, 0, lm(v_s)]$

```
i := r; T := ∅
S := reOrderDegRevLex(S)
WHILE S ≠ ∅ DO
```

```
    s := the first element from S
    i := i + 1
    S := S\{s}
    s := s + lm(s)e_i
    T := T ∪ {s}

END
setSyzygyOrder(r)
```

Notice that the regularity (see 3.1) does not apply to STZS-algorithm as it requires that the resolution, as a minimal one, does not have constant entres. Then the bound for degrees can be decreased by one in the next module beginning with the highest.

But here, this contradicts the computation of a standard basis which allows only to cancel the whole resolution in a fixed degree. However, we have not found any example where this cancellation gives any advantage.

3 Improvements

3.1 The regularity

In the case of global homogeneous computations a very useful invariant exists for minimal resolutions due to D. Mumford. This invariant is called regularity and denoted by $r(I)$ where $I \subset k[\underline{x}]^r$ is an arbitrary submodule of a free module.

Definition 3.1 *Assume that the i-th module of syzygies of I is generated by elements f_j^i with $j = 1, \ldots, n_j$ then $r(I)$ is the minimum of all integers such that for each i:*

$$max_j\{deg(f_j^i)\} \leq r(I) + i.$$

From [BM] we know:

Proposition 3.2 *The regularity is a upper semi-continuous function on flat families of modules. Hence,*

$$r(I) \leq r(in(I)),$$

where $in(I)$ denotes the ideal of leading terms of I.

Our tests have shown that in almost all cases (exept those which are very close to monomial ideals) the additional computation of the resolution of the module or ideal of leading terms takes less time than one obtains by using this bound for the degrees. Therefore, it is advisable to use the regularity by default for quasihomogeneous, global computations.

3.2 Direct minimizations

One possibility to speed up the common algorithm is obtained by using information from the standard basis computation directly to minimize the input. One knows about elements not belonging to a minimal basis as they are used multiplied only with a constant, but not with a proper monomial somewhere in the reduction. If this happens one has, moreover, two elements of the module or ideal having the same leading term. By exchanging both we take out the superfluous generator and if the reduction of the spolynomial yields zero, we skip the corresponding syzygy. Otherwise we think of the remaining polynomial as a new generator.

To implement this we have to rewrite the normal form procedure in the following manner:

$g = \mathbf{NF}(f, F)$
INPUT: $F = \{g_1, \dots, g_n\}$ generators of the given ideal, f a spolynomial
OUTPUT: the normal form of f with respect to F

exchanged:=FALSE
while $S := \{f_i \in F | lt(f_i) | lt(g)\} \neq \emptyset$

if $(\exists f_i \in input : lt(f_i) = c * lt(g))$ and not exchanged
$h := f_i$
$f_i := g$
$input := input \setminus \{f_i\}$
exchanged:=TRUE
else choose $h \in S$ w.r.t. the strategy
g:=spoly(g,h)

if exchanged

if $g \in syz(input)$

$g := 0$

else

renew(g)
$F := F \setminus \{f_i | f_i$ was reduced with $h\}$

In the case of global homogeneous computations it easy to organize the algorithm in such a way that no generator which occur in a direct reduction was used before. The simple solution is to compute degree by degree.

For local computations a more sophisticated strategy is necassary. One computes again degree by degree with respect to the leading terms. But, whenever the degree changes, one has to use a lazy strategy, which means to continue with the other pairs of same degree first. Usually direct reductions are to prefer for any other kind of reductions, even with bad eca1are preferred for any other kind of reductions, even with bad ecart. Our experiences show that it is advisable to keep back multiples of elements with good ecart if they are destroyed by direct reductions.

The optimal effect of the direct reductions may be obtained applying a technique of R. LaScala [LS], which uses the fact that a syzygy produced as reduction of an spolynomial and the reductum (if it is not 0) not survives in the minimized complex. Hence, there is no need to equip this syzygy with a component in the next computation.

3.3 The Hilbert series of resolutions

The Hilbert function has become an important tool for computations of Gröbner bases of quasihomogeneous ideals or modules during recent years. For a detailed discussion of the usage of Hilbert functions in that context I would like to refer to [GMRT].

Let the $k[\underline{x}]$–module M be given by generators $m_1, \dots, m_n$ constituting a standard base of M with respect to the degree compatible ordering $<$.

Lemma 3.3 *Resolving the initial module $In(M) = (lt(m_1), \dots, lt(m_n))$ by the STZS-method, one obtains the Hilbert series as well as the numbers of generators and their degrees of the STZS-resolution of M.*

Proof: Of course, this resolution of leading terms can be continued to a resolution of $M = (m_1, \ldots, m_n)$ (see [MM]). The Hilbert functions are invariants of the initial modules (see [GMRT] Theorem 1) and from the properties of the STZS-resolution follows $Syz_i(In(M)) = In(Syz_i(M))$.

Now, we assume to minimize the STZS-resolution $Syz_i(M)$, $i = 1, \ldots, p$ of M. This means, to find constant entres in the syzygies of level i and to cancel the corresponding element $m_{i_l} \in Syz_{i-1}$. The component i_l in $Syz_i(M)$ is set to zero by Gaussian eliminations and the syzygy is taken out together with its component in Syz_{i+1}. (compare R. La Scala [LS] Chapter 4)

Denote by sf_{i-1} the set of superfluous generators in $Syz_{i-1}(M)$. By the descritpion above we get:

Lemma 3.4 *The Hilbert function H_i of the i-th module in the minimal resolution is given by:*

$$H_i = H_{Syz_i} - \sum_{m \in sf_{i-1}}^{r} H_{k[\underline{x}]} T^{deg(m)}.$$

Remark: For computing the exact Hilbert function of the i-th module in the minimal resolution it is not necessary to know the concrete generators. It suffices to know the number and degree of a minimal set of generators. However, one has to be careful while counting explicit generators. They may be transformed by the Gaussian elimination or by cancellation of components of syzygies which are taken out into zero!

Theorem 1 *By resolving the initial module (or ideal) of the input and detection of direct reductions in $Syz_{i-1}(M)$ one obtains the Hilbert function H_i of the module $Syz_i(M)$. The Hilbert functions $H_i Syz_j$ for $j > i$ are upper bounds for the H_j.*

Proof: By searching for direct reductions one obtains a minimal set of generators for the actual module in the minimal resolution. Comparing them with the generators in the STZS-resolution one finds all superfluous elements in that resolution and can compute the Hilbert functions of the minimal resolution according to the lemmas above.

The interpretation of Hilbert functions in the context of syzygies differs from that in the Gröbner base algorithm. Once having detected all necessary pairs (and checked this by the Hilbert function) of a certain degree in the module $Syz_{i-1}(M)$, all remaining pairs of the same degree will yield syzygies. It seems to be a good idea to wait with their reduction until the next computation. Here one has a complete knowledge of the Hilbert function and can decide how many pairs must be reduced to obtain the correct Hilbert function in $Syz_i(M)$ up to the current degree. The rest of the pairs yields superfluous generators and their computation can be skipped. Therefore, the usage of Hilbert functions suggests a computation degree by degree over the whole resolvent.

3.4 The algorithm of Traverso/Caboara

This algorithm is a clever way to put all the different computations of standard bases of ideals and modules and of their syzygies in a common hull. The main point is to think of an algebra $k[\underline{x}][\underline{y}]$ as infinite module over $k[\underline{x}]$ and to handle all computations in that context. Of course, module components are not variables and so one has to rewrite the normal form procedure to distinguish the variables. One obtains the advantage of having an “automatic” treatment of additional information, such as module components, module indices in resolutions or membership in minimal sets of generators. For a detailed description I would like to refer to [TC].

4 Suggestions for the algorithms

Here I would like to propose a variant of the algorithm for global (quasi-)homomogeneous ideals or modules realizing the ideas of the Sections 3.2 and 3.3. The main question is, whether the usage of Hilbert functions yields better restrictions to the pair sets than the regularity does. This will (hopefully) be answered in the near future by programming it in SINGULAR.

$S = \mathbf{Res}(F)$

INPUT: $F = \{g_1, \ldots, g_n\}$ generators of the given ideal,

OUTPUT: $S = \{S_1, \ldots, S_n\}$ the resolution of F,

$(F', Syz_F) = computeFirstSyzygies(F)$

$S' = resolve(In(F'))$

$hilb_{.} = computeHilbertFunctions(S')$

$L = buildPairs(Syz_F)$

$i = 1$

WHILE $(L \neq \emptyset)$

$h = 0$ //the Hilbert function of the actual module S_i

$L' = \emptyset$

$deg = deg_{min}$

WHILE $(isNotComplete(h, hilb_i))$

$G = reduce(L', deg, hilb_i, h)$

$L' = buildPairs(G)$

$h = computeHilbertFunction(G)$

WHILE $(isNotComplete(h, hilb_i, deg))$

$G = G \cup reduce(L, deg, hilb_i, h)$

$h = computeHilbertFunction(G)$

$checkForDirectReductions(L, S_{i-1}, deg, hilb_i)$

$deletePairs(L, deg)$

$deg = deg + 1$

$S_i = G$

$L = L'$

$i = i + 1$

The main content of the above procedures is

computeFirstSyzygies	is the usual algorithm with direct reductions.
resolve	resolves the initial ideal of the input.
computeHilbertFunctions	computes the Hilbert functions of the resolution of the leading terms.
buildPairs	constructs the pair set from the given generators.
isNotComplete	checks whether the Hilbert function given by the computed generators is just complete (in general or up to the degree deg - depending on the circumstances).
reduce	reduces as many pairs as it is prescribed by the Hilbert functions.
checkForDirectReductions	tries to find the remaining possibilities to minimize the set of generators.
deletePairs	cancels superfluous pairs.

References

[BM] D. Bayer and D. Mumford, What can be computed in algebraic geometry?, Proceedings of Computational algebraic geometry and commutative algebra Cortona 1991, Cambridge, (1993), 1-48.

[GMRT] P. Gianni, T. Mora, L. Robbiano and C. Traverso, Hilbert functions and Buchberger algorithm, Preprint, (1994).

[SI] H. Grassmann, G.M. Greuel, B. Martin, W. Neumann, G. Pfister, Th. Siebert, W. Pohl and H. Schoenemann: Standard bases, syzygies and their implementation in SINGULAR, Preprint 251 (1994), University Kaiserslautern.

[LS] R. LaScala, Strategies for minimal free resolutions, to appear.

[MM] H. M. Möller and T. Mora, New constructive methods in classical ideal theory, J. of Algebra 100, 138-178 (1986).

[MM1] H. M. Möller and T. Mora, Computational aspects of reduction strategies to construct resolution of monomial ideal, ???.

[S] Schreyer, F.-O.: Die Berechnung von Syzygien mit dem verallgemeinerten Weierstrasschen Divisionssatz. Diplomarbeit, Hamburg (1980).

[TC] C. Traverso and M. Caboara, The powerful Buchberger algorithm for modules, Preprint Pisa, (1996).

List of papers published in the Reports on Computer Algebra series

[1] H. Grassmann, G.-M. Greuel, B. Martin, W. Neumann, G. Pfister, W. Pohl, H. Schönemann, and T. Siebert. Standard bases, syzygies and their implementation in singular. 1996.

[2] H. Schönemann. Algorithms in singular. 1996.

[3] R. Stobbe. FACTORY: a C++ class library for multivariate polynomial arithmetic. 1996.

[4] O. Bachmann and H. Schönemann. A Manual for the MPP Dictionary and MPP Library. 1996.

[5] O. Bachmann, S. Gray, and H. Schönemann. MPP: A Framework for Distributed Polynomial Computations. 1996.

[6] G.M. Greuel and G. Pfister. Advances and improvements in the theory of standard bases and syzygies. 1996.

[7] G.M. Greuel. Description of SINGULAR: A computer algebra system for singularity theory, algebraic geometry, and commutative algebra. 1996.

[8] T. Siebert. On strategies and implementations for computations of free resolutions. September 1996.

[9] B. Reinert. Introducing reduction to polycyclic group rings – a comparison of methods. October 1996.

[10] O. Bachmann, S. Gray, and H. Schönemann. A proposal for syntactic data integration for math protocols. January 1997.

[11] O. Bachmann. Effective simplification of cr expressions. January 1997.

[12] O. Bachmann, S. Gray, and H. Schönemann. MP Prototype Specification. January 1997.

[13] O. Bachmann. MPT – a library for parsing and Manipulating MP Trees. January 1997.

[14] K. Madlener and B. Reinert. Relating rewriting techniques on monoids and rings: Congruences on monoids and ideals in monoid rings. September 1997.

[15] B. Martin and T. Siebert. Splitting Algorithm for vector bundles. September 1997.

[16] K. Madlener and B. Reinert. String Rewriting and Gröbner Bases – A General Approach to Monoid and Group Rings. October 1997.

[17] Thomas Siebert. An algorithm for constructing isomorphisms of modules. January 1998.

[18] O. Bachmann and H. Schönemann. Monomial Representations for Gröbner Bases Computations. January 1998.

[19] B. Reinert, K. Madlener, and T. Mora. A note on nielsen reduction and coset enumeration. February 1998.