Gröbner Basis and Applications [1] – A Survey

1 Introduction

All the polynomial ideals are associated with varieties. To operated with ideals, we will need to illustriate the computation of the *basis* of the ideal. Hironaka in 1964, see [Hir], established the existence of the standard bases for the *local properties* for the ideals of formal power series. However it was Buchberger who, in his Ph.D. thesis, see [B1], first presented an algorithm to perform the required transformation in the context of polynomial ideals in the *global cases*.
What is a Gröbner basis ? The Gröbner basis is a finite set of multivariate polynomials solves the following problem (see [B4]) :

- given a finite set F of multivariate polynomials over a field, construct a finite set F' of multivariate polynomials such that $\equiv_F = \equiv_{F'}$ and $\rightarrow_{F'}$ is Church-Rosser.

Where, for a set F of polynomials, $\equiv_F$ is the ideal congruence modulo the ideal generated by F (i.e. $f \equiv_F g \Leftrightarrow f - g \in ideal(F)$) and $\rightarrow_F$ is a certain Notherian reduction relation on polynomials introduced by F with the property that the reflexive-symmetric-transitive closure of $\rightarrow_F$ is equal to $\equiv_F$. If F' is the Gröbner bases of F, the Church-Rosser property guarantees, that for arbitrary polynomials f, g the congruence $f \equiv_F g$ can decided by computing normal forms of f and g modulo $\rightarrow_{F'}$ and checking for syntactic equality.
A Gröbner basis is used to describe an ideal $F := (f_1, \ldots, f_m)$ of the polynomial ring $k[x_1, \ldots, x_n]$. It permits us to determinate, if a given polynom f belongs to F, and if not, to calculate his remainder modulo F. Let $k[x]$ be an univariate polynomial ring, let $F := (f_1(x), \ldots, f_m(x))$ be an ideal. With the Euclidian algorithm, it is easy to reduct a given polynom $f(x)$. This method is based on the **natural notion of ordering of terms** in polynomials. For a multivariate polynom, there doesn't exist such a natural order.

2 Problems concerning the algebra of polynomial ideals and geometry of affine varieties

Below we summarize some those problems of Gröbner basis that are used in the subsequent applications, (see [BCK] and [CLO]).

Problem 1

The Ideal Description Problem : *Does every ideal* $I \subset k[x_1, \ldots, x_n]$ *have a finite generating set ? In other words, we can write* $I = (f_1, \ldots, f_s)$ *for some* $f_i \in k[x_1, \ldots, x_n]$ *?*

Problem 2

The Ideal Membership Problem : *Given* $f \in k[x_1, \ldots, x_n]$ *and an ideal* $I = (f_1, \ldots, f_s)$*, determine if* $f \in I$*. Geometrically, this is closely related to the problem of determining whether* $V(f_1, \ldots, f_s)$ *lies on the variety* $V(f)$*.*

Problem 3

The Problem of Solving Polynomial Equations : *Find all common solutions in* K^n *of a system of polynomial equations*

$$f_1(x_1, \ldots, x_n) = \ldots = f_s(x_1, \ldots, x_n) = 0$$

(This is the same as asking for the points in the affine variety $V(f_1, \ldots, f_s)$*.)*

Problem 4

The Problem of Implicitization : *Let* V *be a subset of* K^n *given parametrically as :*

$$\begin{aligned} x_1 &= g_1(t_1, \ldots, t_m) \\ &\ldots \\ x_n &= g_n(t_1, \ldots, t_m) \end{aligned}$$

If the g_i *are polynomials (or rational functions) in the variables* t_j*, the* V *will be an affine variety or part of one. Find a system of polynomial equations (in the* x_i*) that define the variety.*

Note that problems **3** and **4** are inverse problems.

3 Orderings on the monomials in $k[x_1, \ldots, x_n]$

Let $X^\alpha := x_1^{\alpha_1} \ldots x_n^{\alpha_n}$ be a monomial , where $X := x_1 \ldots x_n$ and $\alpha := (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$. We note that we can reconstruct the monomial X^α from the n-tuple of exponents α. This establishes a one-to-one correspondence between the monomials in $k[X]$ and $\mathbb{N}^n$

Definition 1 *An ordering on the monomials of $k[X]$ is a relation " $>$ " on $\mathbb{N}^n$, so that: 1) $>$ is a total ordering on $\mathbb{N}^n$. 2) if $\alpha > \beta$, for every γ in $\mathbb{N}^n$,we have $\alpha + \gamma > \beta + \gamma$. 3) $>$ is a "well-ordering", it means every nonempty subset of $\mathbb{N}^n$ has got a smallest element under $>$.*

Let α and β be in $\mathbb{N}^n$. We will use the three following orders (a complete description of the orderings satisfying above conditions is given in [Rob]) :

• **Lexicographic Order** : we say $\alpha >_{lex} \beta$ if, in the vector difference $\alpha - \beta$, the left-most nonzero entry is positive. We will write $X^\alpha >_{lex} X^\beta$ if $\alpha >_{lex} \beta$

Remark 1 *We have $x_1 >_{lex} x_2 >_{lex} \ldots >_{lex} x_n$ If we work with polynomials in two or three variables, we will call them x, y, z rather then x_1, x_2, x_3. We will also assume that the alphabetical order $x > y > z$ on the variables is used to define the lexicographic ordering unless we explicitly say otherwise.*

Examples :
(a) $(1, 2, 0) >_{lex} (0, 3, 4)$ since $\alpha - \beta = (1, -1, -4)$,
(b) $(3, 2, 4) >_{lex}$ since $\alpha - \beta = (0, 0, 3)$,
(c) $(1, 0, \ldots, 0) >_{lex} (0, 1, 0, \ldots, 0) >_{lex} \ldots >_{lex} (0, \ldots, 1)$.

• **Graded Lex Order** : we say $\alpha >_{gradlex} \beta$ if,

$$| \alpha | := \sum_{i=1}^{n} \alpha_i > | \beta | := \sum_{i=1}^{n} \beta_i, \text{ or } | \alpha | = | \beta | \text{ and } \alpha >_{lex} \beta$$

we have also $x_1 >_{gradlex} x_2 >_{gradlex} \ldots >_{gradlex} x_n$
Examples :
(a) $(1, 2, 0) >_{gradlex} (3, 2, 0)$,
(b) $(1, 2, 4) >_{gradlex} (1, 1, 5)$.

• **Graded Reverse Lex Order** : we say $\alpha >_{revgradlex} \beta$ if,

$$\mid \alpha \mid := \sum_{i=1}^{n} \alpha_i > \mid \beta \mid := \sum_{i=1}^{n} \beta_i, \text{ or } \mid \alpha \mid = \mid \beta \mid \text{ and}$$

$$\text{the right most nonzero entry in } \alpha - \beta, \text{is negative}$$

we have also $x_1 >_{revgradlex} x_2 >_{revgradlex} \cdots >_{revgradlex} x_n$
Examples :
(a) $(4,7,1) >_{revgradlex} (4,2,3)$ since $\mid (4,7,1) \mid = 12 > \mid (4,2,3) \mid = 9$
(b) $(1,5,2,2) >_{revgradlex} (4,1,3,2)$ since $a - b = (-3,4,-1,0)$.

Remark 2 *The latter of these ordering may appear as rather strange ; but it should be emphasized that it is generally the one for which the computation of a Gröbner basis has the best theoretical and practical complexity.*

Proposition 1 *These three orderings are monomial orderings.*•

Some Examples :
Let $p := 4xy^2z + 4z^2 - 5x^3 + 7x^2z^2 \in k[x,y,z]$. Then :
(a) with respect to the *lex*-order, we could have :

$$p := -5x^3 + 7x^2z^2 + 4xy^2z + 4z^2$$

(b) with respect to the *gradlex*-order, we could have :

$$p := 7x^2z^2 + 4xy^2z - 5x^3 + 4z^2$$

(c) with respect to the *revgradlex*-order, we could have :

$$p := 4xy^2z + 7x^2z^2 - 5x^3 + 4z^2$$

Definitions 1 *Let $f := \sum_\alpha c_\alpha x^\alpha$ be a nonzero polynomial in $k[x_1, \ldots, x_n]$ and let $>$ be a monomial order. The* **multidegree** *of f is*

$$md(f) := max(\alpha \in \mathbb{N}^n / c_\alpha \neq 0)$$

The **leading monomial** *of f is*

$$lm(f) := x^{md(f)}, \text{ (with coefficient 1)}$$

The **leading coefficient** *of f is*

$$lc(f) := a_{md(f)} \in k$$

The **leading term** *of f is*

$$in(f) := lc(f) \cdot lm(f)$$

4 A division algorithm in $k[x_1, \ldots, x_n]$

Reducing a monomial m by a polynomial p consists in replacing m by $m - \frac{m}{\mathrm{lmon}(p)} \frac{p}{\mathrm{lcoef}(p)}$ if m is a multiple of the leading monomial $\mathrm{lmon}(p)$ of p, or in doing nothing if it is not a multiple. *Reducing* a polynomial p by a set of polynomials S consists in reducing the monomials in p by the polynomials of S, while it is possible. For easy reasons of efficiency, the biggest (for the monomial ordering) monomials of p have to be reduced first. As an example, it is worth to remark that reducing an univariate polynomial by another consists in computing the rest of euclidean division. It should also be noted that, in the general case, the reduction depends on some choices, and, thus, that the reduction has not an uniquely defined result.
We can use any one of our orderings on monomials, see section **3**, to formulated a division algorithm for general polynomials extending the usual algorithm in $k[x]$. Let $>$ be a fixed monomial order on $\mathbb{N}^n$.

Theorem 1 (Division Algorithm in $k[x_1, \ldots, x_n]$) : *Let $F := (f_1, \ldots, f_m)$ be a ordered, (by $>$), m-tuple of polynomials in $k[x_1, \ldots, x_n]$. Then every $f \in k[x_1, \ldots, x_n]$ can be written as*

$$f = a_1 f_1 + \ldots + a_m f_m + r \ ,$$

where a_i and r are in $k[x_1, \ldots, x_n]$, and there doesn't exist j, $1 \leq j \leq m$, such that $in(f_j)$ divides r. We call r a **remainder** *of F by division by F.*•

Algorithm 1 Division Algorithm (see [CLO]) :

input : $f_1, \ldots, f_m, f$

output : $a_1, \ldots, a_m, r$

Main Idea : *"Pick off" the initial form of the f each time.*

begin

$a_1 := 0, \ldots, a_m := 0, r := 0$

$q := f$

while $q \neq 0$ do

$j := 1$

pickterme:=false

while ($j \leq m$ and pickterme=false) do

if $in(f_j)$ divides $in(q)$ then

$a_j := a_j + in(q)/in(f_j)$

$q := q - (in(q)/in(f_j)) \cdot f_j$

pickterme:=true

else $j := j + 1$

if pickterme=false then

$r := r + in(q)$

$q := q - in(q)$

end

A important property of the division algorithm in k[x] is that the remainder is uniquely determinated. But this can fail when there is more than one variable, and the decomposition modulo F depends on the ordering of $f_1, \ldots, f_m$ (see below).

Example ([CLO]) :
Let $f_1 := x^3 - xy,\ f_2 := x - y^4,\ f := x^5y^2 \in k[x, y]$ with the lexicographic order. Then :

$$a_1 := 0,\ a_2 := 0,\ r := 0,\ q = f := x^5y^2$$

– *(1 pass) :* we see that $in(f_1)$ divides $in(q)$, so

$$a_1 := a_1 + in(q)/in(f_1) = 0 + x^2y^2 = x^2y^2$$

$$q := q - [in(q)/in(f_1)]f_1 = x^5y^2 - x^2y^2(x^3 - xy) = x^3y^3$$

The remainder r stay 0.
– *(2 pass) :* $in(f_1)$ divides $in(q)$, so

$$a_1 := a_1 + in(q)/in(f_1) = x^2y^2 + y^3$$

$$q := q - [in(q)/in(f_1)]f_1 = x^3y^3 - y^3(x^3 - xy) := xy^4$$

– *(3 pass) :* $in(f_1)$ does not divides $in(q)$ but $in(f_2)$, so

$$a_2 := a_2 + in(q)/in(f_2) = 0 + y^4 = y^4$$

$$q := q + [in(q)/in(f_2)]f_2 = xy^4 - y^4(x - y^4) = y^8$$

– *(4 pass) :* neither $in(f_1)$ nor $in(f_2)$ divides $in(q)$, so

$$r := r + in(q) = 0 + y^4 = y^4$$

$$q := q - in(q) = 0$$

and the main loop terminated. The result of the division algorithm is :

$$f = a_1f_1 + a_2f_2 + r, \text{ or}$$
$$x^5y^2 = (x^2y^2 + y^3)(x^3 + xy) + y^4(x - y^4) + y^8$$

Now let us perform the same division, changing only the order of the pair of polynomials of divisors. We start with $f_1 := x - y^4,\ f_2 := x^3 - xy,\ f := x^5y^2 \in k[x, y]$. Then :

$$a_1 := 0,\ a_2 := 0,\ r := 0,\ q = f := x^5y^2$$

After 6 pass the algorithm terminates. The final result is :

$$f = a_1f_1 + a_2f_2 + r, \text{ or}$$

$$x^5y^2 = (x^4y^2 + x^3y^6 + x^2y^{10} + xy^{14} + y^{18})(x - y^4) + 0(x^3 - xy) + y^{22}$$

Remark 3 *In the previous example the a_i and the r can change if we simply rearrange the f_i. The a_i and the r may also change if we change the monomial ordering.*

Remark 4 *The nicest features of the division algorithm in $k[x]$, is the way it completely solves the ideal membership problem (see* **Problem 2** *and section 6.1). Do we get something similar here ?*
Corollary of Theorem 1 : *If after division of f by $F := (f_1, \ldots, f_s)$ we obtain a zero remainder, then :*

$$f = a_1 f_1 + \ldots + a_s f_s$$

so that $f \in (f_1, \ldots, f_s)$.

The previous corollary gives a sufficient condition for the ideal membership.
Counter-example : *Let $f_1 := xy + 1,\ f_2 := y^2 - 1 \in k[x, y]$ with the lexicographic order. Diving $f := xy^2 - x$ by $F := (f_1, f_2)$ we have :*

$$xy^2 - x = y(xy + 1) + 0(y^2 - 1) + (-x - y),$$

by $F := (f_2, f_1)$:

$$xy^2 - x = x(y^2 - 1) + 0(xy + 1) + 0$$

From the second calculation, we known that $f \in (f_2, f_1)$. However, we see by the first calculation that even if $f \in (f_1, \ldots, f_s)$ it is still possible to obtain a non-zero remainder on division by $F := (f_1, \ldots, f_s)$ if the f_i is ordered incorrectly, or the f_i themselves are "wrong" in some way. It is too much to hope that f has a well-defined remainder of f on "division by the ideal I", since the remainder can change if we simply list of generators of I in a different order.

5 Hilbert Basis Theorem and Gröbner bases

5.1 The Hilbert basis theorem

Our aim here is to study a certain familly of ideals, which will be usefull for constructing the Gröbner bases . Our considerations will be also lead to ideal with "good" properties relative to the division algorithm. The key idea we will use is that once we have chosen a monomial ordering, each $f \in k[x_1, \ldots, x_n]$ has a unique initial form $in(f)$.

Definition 2 *An ideal I in $k[x_1, \ldots, x_n]$ is a* **monomial ideal** *if this is generated by collection (not necessarily finite) of monomials.*

Definition 3 *Let I be an ideal in $k[x_1, \ldots, x_n]$, other than $\{0\}$. Let note by $in(I)$ the set $\{in(f)/f \in I\}$ and by $(in(I))$ the ideal generated by the elements of $in(I)$.*

Remark 5 *$(in(f_1), \ldots, in(f_t))$ and $(in(I))$ can be different ideals. In general $(in(f_1), \ldots, in(f_t)) \subset (in(I))$.*

Lemma 1 (Membership problem for monomial ideals) : *Let A be a subset in $\mathbb{N}^n$ (possibly infinite). Let $I := (x^\alpha/\alpha \in A)$ be a monomial ideal. The $f \in I$ iff every term in f is divisible by one of x^α.*

Proof :

($\Rightarrow$) Let $f := \sum_{i=1}^{s} g_i x^{\alpha_i}$, where $g_i \in k[x_1, \ldots, x_n]$. Every monomial x^β of f has $\beta = \alpha_i + \gamma$ for some a_i and $\gamma \in \mathbb{Z}^n_{\geq 0}$.

($\Leftarrow$) Easy.•

Theorem 2 *Every monomial ideal I generated by a finite collection of monomials.*•

Theorem 3 (Hilbert basis theorem) : *Every ideal I in $k[x_1, \ldots, x_n]$ has a finite generating set. That is, $I = (g_1, \ldots, g_r)$ for some $g_1, \ldots, g_r \in I$.*

Proof :

By the previous theorem the monomial ideal $((in(I))$ generated by a finite collection of monomials let $(x^{\alpha_1}, \ldots, x^{\alpha_t})$. Let $g_i \in I$. We can see that each x^{α_i} is actually the leading monomial of g_j.

$$x_\alpha := \sum_{j=1}^{t} f_j in(g_j)$$

$f_j \in k[x_1, \ldots, x_n]$; but x^α is a monomial then $x^\alpha = c \cdot m \cdot in(g_j)$, where $c \in k$, some monomial m and $g_j \in I$. **Hence each of the generating monomials of $(in(I))$ is the leading monomial of some element g_j of I.** Then

$$(in(I)) = (in(g_1), \ldots, in(g_t))$$

We can see now our claim, that $I = (g_1, \ldots, g_t)$ for some $g_1, \ldots, g_t \in k[x_1, \ldots, x_n]$. The fact that $(g_1, \ldots, g_t) \subset I$ is easy.

Now $I \subset (g_1, \ldots, g_t)$. Let $f \in I$. Using Divisor algorithm f/G, where $G = (g_1, \ldots, g_t)$, we can remove $in(f)$ with some multiple of one of the g_i, because $in(f) \in (in(G)) = (in(g_1), \ldots, in(g_t))$. But $f - m \cdot g_i \in I$. At the termination of the division our result will be

$$f = \sum_{i=1}^{t} a_i g_i + 0$$

for some $a_i \in k[x_1, \ldots, x_n]$. But the remainder is zero, then, $f \in (g_1, \ldots, g_t)$.•

5.2 The Gröbner bases – Some Properties

Definition 4 *Fix a monomial order. A finite subset $G := \{g_1, \ldots, g_r\}$ of an ideal I is said to be a Gröbner basis , abbr.* **GB**, *(or* **standard basis***) if*

$$(in(g_1), \ldots, in(g_r)) = (in(I)).$$

Corollary 1 *Fix a monomial order. Then every ideal $I \subset k[x_1, \ldots, x_n]$ other than $\{0\}$ has a Gröbner basis . Furthermore, any Gröbner basis for I is a basis of I.*

Proof :

Consequence of Hilbert Basis Theorem.•

For ideals generated by linear polynomials, a Gröbner basis for *lex*-order is determined by the row echelon form of the matrix made form the coefficients of the generators.

Proposition 2 *Let $J \subset \mathbb{R}[x_1, \ldots, x_n]$ be a ideal generated by the rows of $A(x_1, \ldots, x_n)^t$, where A is an $m \times n$ matrix is a row echelon form. The given generators form a Gröbner basis for J with respect to a suitable lexicographic order.*•

Example :
Consder the ideal $J := (g_1, g_2) = (x + z, y - z)$. The g_1 and g_2 form a Gröbner basis using *lex*-order in $\mathbb{R}[x, y, z]$ since $(in(J)) = (x, z)$ and $(in(g_1), in(g_2)) = (x, z)$ (proof). Notice that the generators for the ideal J come form a row echelon matrix of coefficient :

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$

Proposition 3 *Let $G := \{g_1, \ldots, g_r\}$ be a Gröbner basis for an ideal $I \subset k[x_1, \ldots, x_n]$ and let $f \in I$. The remainder $\bar{f}^G$ of the reduction of f by G is unique (Division algorithm) and it is independent of the ordering of G used in the Division algorithm.*

Proof :

Let $f = q_1 + r_1 = q_2 + r_2$ be two expressions, both obtained by division, but with the element of G ordered by different ways. Then $r_1 - r_2 = q_1 - q_2 \in I$. If $r_1 - r_2 \neq 0$, since G is a Gröbner basis of I, the leading monomial of every element (and the $r_1 - r_2$) of I is a multiple of the leading monomial of some element of G. This contradicts in the fact that none of the monomials of r_1 or r_2 or $r_1 - r_2$ is a multiple of the leading monomials of any of the elements of G.•

Proposition 4 *Let G be a Gröbner basis for I, and $f \in I$:*

$$f \in I \Longleftrightarrow \bar{f}^G = 0 \bullet$$

Remark 6 *This corollary solves the* **Ideal membership problem,** *(***Problem 2** *section 2 and section 6.1). This was one of Buchberger's original intentions.*

Proposition 5 *If G is an ideal basis for I with the property $\bar{f}^G = 0$ for all the $f \in I$, then G is a Gröbner basis for I.* •

5.3 S–polynomials

The main algorithm for computing Gröbner bases is Buchberger's one, which he introduced in 1965, in his thesis [B1], [B2], [B3], [B4]. This algorithm is developed from two basic tools, the reduction or division process and the critical pairs or S–polynomials.

Definition 5 *Let f, g in $k[x_1, \ldots, x_n]$ be non-zero polynomials.*
(1) If $md(f) = \alpha$ and $md(g) = \beta$ then let γ be $(\gamma_1, \ldots, \gamma_n)$, where $\gamma_i := max(\alpha_i, \beta_i)$, for each i. We call x^γ the **least common multiple** *of $lm(f)$ and $lm(g)$, and we note $x^\gamma := lcm(lm(f), lm(g))$.*
*(2) The S–***polynomial** *of f and g is*

$$S(f,g) := \frac{x^\gamma}{in(f)} f - \frac{x^\gamma}{in(f)} g$$

Example 1 : Let $f := x^3y^2 - x^2y^3 + x$ and $g := 3x^4y + y^2$ in $\mathbb{R}[x, y]$ with *gradlex* order. Then $\gamma := (4, 2)$ and

$$\begin{aligned} S(f,g) &:= \frac{x^4y^2}{x^3y^2}f - \frac{x^4y^2}{3x^4y}g \\ &:= -x^3y^3 + x^2 - \frac{1}{3}y^3 \end{aligned}$$

Properties (see [CLO]) :

(1) An S–polynomial $S(f, g)$ is "designed" to produce cancellation of leading terms.

(2) We have $S(f, g) \in (f, g)$ and $S(f, g) = S(af, bg)$ for every f, g in $k[x_1, \ldots, x_n]$ and every $a, b \in k$.

(3) For every f, g in $k[x_1, \ldots, x_n]$, for every α, β in $\mathbb{N}^n$, there exists γ in $\mathbb{N}^n$ such that $S(x^\alpha f, x^\beta g) = x^\gamma S(f, g)$ where

$$x^\gamma := \frac{lcm(x^\alpha lm(f), x^\beta lm(g))}{lcm(lm(f), lm(g))}.$$

(4) Let $p_1, \ldots, p_m \in k[x_1, \ldots, x_n]$ with $multideg(p_i) = \alpha$, $\alpha \in \mathbb{Z}^n_{\geq 0}$ for all i. If for some set of $c_i \in k$,

$$multideg(\sum_{i:=1,m} c_i p_i) < \alpha, \text{ then } \sum_{i:=1,n} c_i p_i$$

is some linear combination (with coefficient in k), of S–polynomials $S(p_i, p_j)$, $1 \leq i, j \leq m$.

Using the S–polynomials we have got an other characterization of Gröbner bases , which is, from an **algorithmic point of view**, **better** than the definition.

Theorem 4 *Let I be a polynomial ideal. Then a basis $G := \{g_1, \ldots, g_s\}$ for I is a Gröbner basis for I if and only if for all pairs $j \neq j$, the remainder of the division of $S(g_i, g_j)$ by G (listed in some order) is zero.*

Proof (see also [CLO]) :

($\Rightarrow$) Let G is a Gröbner basis . Since $S(g_i, g_j) \in I$ by proposition 4 the remainder is zero.

($\Leftarrow$) Let $f \in I$, $f \neq 0$. We must show that if the S–polynomials all give remainders of zero on division, then $in(f) \in (in(g_1), \ldots, in(g_s))$. Since $I := (g_1, \ldots, g_s)$, there are constants $c_{i\alpha} \in k$ such that

$$(\dagger) \qquad f = \sum_{i=1}^{s} [\sum_{\alpha \in \mathbb{Z}^n_{\geq 0}} c_{i\alpha} x^{\alpha}] g_i$$

Let $\delta := max\{\alpha + multideg(g_i) : c_\alpha \neq 0\}$, and let q be the sum of all terms in the expression ($\dagger$) for f that contribute terms of multi-degree δ :

$$q := \sum_i c_{i\alpha(i)} x^{\alpha(i)} g_i,$$

where for each i, there is at most one $\alpha(i)$ such that $x^{\alpha(i)} g_i$ has multi-degree δ, and we include only those terms in the sums. We will also assume that among all possible expressions ($\dagger$) for f, we have selected one in which δ is minimal.

There are two cases :

(a) If *multideg(f)=*δ (so that not all the terms of multi-degree δ in ($\dagger$) cancel out), then $in(f) = in(q) \in (in(g_1), \ldots, in(g_s))$, which is what we wanted to prove. We are done in this case.

(b) If, on the other hand, $multideg(f) < \delta$, then the terms of multi-degree δ in f, and hence q cancel. By properties (4), q can be rewritten as :

$$q := \sum_{i \neq j} d_{ij} S(x^{\alpha(i)} g_i, x^{\alpha(j)} g_j)$$

for some constants $d_{ij} \in k$. However, by the definition of the S–polynomial

$$S(x^{\alpha(i)} g_i, x^{\alpha(j)} g_j) = x^{\beta_{ij}} S(g_i, g_j),$$

where $x^{\beta_{ij}}$ is a monomial. The $S(g_i, g_j)$ give a remainder of zero on division by G (by hypothesis). Applying the division algorithm, then multiplying by $x^{\beta_{ij}}$, we can replace each of the $x^{\beta_{ij}} S(g_i, g_j)$ by a combination $\sum \alpha_i g_i$. We claim that every term that occurs in this expression has multidegree less than δ. The reason is that in the expression $\sum \beta_i g_i$

for the S–polynomial given by the division algorithm, the initial form of the S–polynomial appears in exactly one term in the sum $\sum \beta_i g_i$, and the other terms are produced by subsequent steps in the algorithm. These subsequent terms are strictly less than the initial form in our monomial order. This means that all the terms in $\sum \alpha_i g_i = x^{\beta_{ij}} \sum b_i g_i$ have multi-degree δ. Subtituting back in equation (†), we get an expression for f of the same form as (†), but that involves only terms of multi-degree less than δ. This contradicts the way δ was chosen, so this case actually does not occur. •

Example 2 : Let $I = (f_1, f_2) = (x^3 - 2xy, x^2y - 2y^2 + x)$ using the *grlex*. Then (f_1, f_2) is not a Gröbner basis for I, since $x^2 \in (in(I))$, but $x^2 \notin (in(f_1), in(f_2))$. To produce a Gröbner basis , one natural idea is to try first to extend the original generating set to a Gröbner basis by adding more polynomials in I. In one sense, this adds nothing new, and even introduces an element of redundancy. However, the extra information we get from a Gröbner basis more than makes up to this. $S(f_1, f_2) = -x^2 \in I$, but its remainder on division by $F = (f_1, f_2)$ is $-x^2$, which is nonzero. Hence we should include that remainder in our generating set, as a new generator called f_3. Thus we let $F = (f_1, f_2, f_3)$. We test to see if this new set is a Gröbner basis for I, using the theorem 4, which gives a completely algorithmic way to decide if a set is a Gröbner basis . We compute :

$$\begin{array}{lcl} S(f_1, f_2) & = & f_3, \text{ so} \\ \overline{S(f_1, f_2)}^F & = & 0 \\ S(f_1, f_3) & = & (x^3 - 2xy) - (-x)(-x^2) \\ & = & -2xy \notin (in(F)). \end{array}$$

Hence we must add $f_4 = -2xy$ to our generating set, and we rename F as $F = (f_1, f_2, f_3, f_4)$. Then

$$\begin{array}{lcl} \overline{S(f_1, f_2)}^F & = & \overline{S(f_1, f_3)}^F = 0 \\ S(f_1, f_4) & = & y(x^3 - 2xy) - (-1/2)x^2(-2xy) = -2xy^2 = yf_4, \text{ so} \\ \overline{S(f_1, f_4)}^F & = & 0 \\ S(f_2, f_3) & = & (x^2y - 2y^2 + x) - (-y)(-x^2) = -2y^2 + x \end{array}$$

We have $in(-2y^2 + x) = -2y^2$. This is not contained in the ideal $(in(F)) = (x^3, x^2y, x^2, xy)$ generated by the leading monomials of the elements of F. Thus we must also add $f_5 = -2y^2 + x$ to our generating set. We check

$$S(f_1, f_5) = y^2(x^3 - 2xy) - (-1/2x^3)(-2y^2 + x) = -2xy^3 + (1/2)x^4$$

so $\overline{S(f_1, f_5)}^F = 0$. Similarly, all the other $\overline{S(f_i, f_j)}^F = 0$, so that $G = \{f_1, f_2, f_3, f_4, f_5\} = \{x^3 - 2xy, x^2y - 2y^2 + x, -x^2, -2xy, -2y^2 + x\}$ is a Gröbner basis for I.

5.4 Buchberger's Algorithm

The procedure we used in the previous example of adjoining $\overline{S(f_i, f_j)}^F$ to F if it is nonzero can be elaborated into an algorithm for producing Gröbner bases . The **Buchberger's Algorithm** gives us a method how to construct such a basis in a finite number of operations. The version we will present is a very rudimentary one. The general idea is to try to extend the original generating set to a Gröbner basis by adding more polynomials in I.

Theorem 5 ***Algorithm 2*** Buchberger's Algorithm :

input : $F := (f_1, \ldots, f_n)$

output : $G := (g_1, \ldots, g_s)$ *Gröbner basis for I, with* $F \subset G$

$G := F$

repeat

$G' := G$

for each pair $\{p, q\}, p \neq q$ in G' do

$S := \overline{S(p, q)}^{G'}$

if $S \neq 0$ then $G := G \cup \{S\}$

until $G = G'$

Proof :

If $G = G'$ the algorithm terminates and all $\overline{S(p, q)}^G = 0$, so G is a Gröbner basis by proposition 4.

After each pass through the main loop,

$$(\ddagger) \qquad (in(G')) \subset (in(G)),$$

since $G' \subset G$. Futhermore, if, after a pass through the loop, $G \neq G'$, then $in(G') \subset in(G)$, but they not equal. The reason is that the initial forms of the S that were adjoined to G in the pass cannot lie in $(in(G'))$ (otherwise, those initial forms would have been removed in the computation of the remainder on division by G'). By contraposition, if we ever have $(in(G')) = (in(G))$, then $G' = G$.

By the (‡) the ideals $(in(G'))$ formed in successive iterations of the loop are part of an ascending chain of ideals in $k[x_1, \ldots, x_n]$. But $k[x_1, \ldots, x_n]$ is a noetherian ring, so, after a finite number of iterations the chain will stabilize. As result, the algorithm will always terminate. •

Remark 7 *A constructive proof is given of the terminantion of the algorithm for computing Gröbner bases in polynomial rings over a* constructively noetherian *ring, see [JL]. They combine the constructive approach of Richman and Seidenberg with the Gröbner basis methode and they introduce a new induction principle proposed by Per Martin-Löf in the definition of noetherian rings.*

The adds of polynomials in I may introduce an element of redundancy. Note (as a first improvement) that once a remainder $\overline{S(p,q)}^{G'} = 0$, the remainder will stay zero even if we adjoin further elements to the end of the ordered generating set G'. Thus, there is no reason to recompute those remainders on subsequent passes throught the main loop. It is possible to ameliorate this algorithm, by using the following results, and limitate the efforts of calculation, in the reduction algorithm, by ordering the polynomials $f_1, \ldots, f_n$ such that the leading terms grow (it will diminuate the number of necessary tests for finding a f_j such that $in(f_j)$ divides $in(q)$) and in the Buchberger's algorithm, by considering at first the pairs (p, q) such that $lcm(in(p), in(q))$ is as small as possible. (The probability that $\overline{S(p,q)}^{G'} \neq 0$ will be greater. Thus, we will faster add elements in the basis which we want to construct, and the followings S–polynomial reductions will have more chance to be nil).

Lemma 2 *(see also [CLO]) : Let G be a Gröbner basis for a polynomial ideal I. Let p in G be a polynomial, such that $in(p) \in (in(G \backslash \{p\}))$. Then $G \backslash \{p\}$ is also a Gröbner basis for I.*•

Definition 6 *A* **reduced Gröbner basis** *for a polynomial ideal I is a Gröbner basis G for I such that : (1) $lc(p) = 1$ for all p in G (2) for every p in G, no monomial of p lies in $(in(G\backslash\{p\}))$.*

Proposition 6 *(see also [CLO]) : Let I be a polynomial ideal other than $\{0\}$. Then, for a given monomial ordering, I has an unique reduced Gröbner bases.*•

Example : Let the Gröbner basis to the ideal I studied in example 2 in the section 5.3. The Gröbner basis $\{f_1, f_2, f_3, f_4, f_5\}$ is not reduced since :

a) Some of the leading coefficients are different from 1. However, this is not a serious objection-we can simply multiply all the generators by suitable constants to make this true.

b) For example, $in(f_1) = x^3 in(f_3)$. We can dispense with f_3 the reduced Gröbner basis . Similarly, since $in(f_2) = x^2 y = -(1/2)x in(f_4)$, we can also eliminate f_2. There are no futher cases where the initial form of a generator is a multiple of the initial form of another generator. Hence

$$f_3 = x^2,\ f_4 = xy,\ f_5 = y^2 - (1/2)x$$

do form a reduced Gröbner basis for I.

5.5 Improvements for the Gröbner bases

For an efficient implementation, this algorithm needs several improvements which concern :

Criteria for avoiding to compute S–polynomials for which one may predict that they reduce to 0. *These criteria may avoid to compute up to 90% of the S–polynomials, but improvements are yet needed because 90% of the remaining one may reduce to* 0 *(see also Appendix I).*

Strategies for choosing the S–polynomials to compute first and for selecting the polynomial by which to reduce when there is a choice. The strategy which is used has a big influence on the number of S–polynomials to compute and on the size of the coefficients to handle ; both parameters are critical for efficients computations [Sugar].

The way of normalizing the polynomials which appear : they are defined up to the multiplication by a constant. When dealing with polynomials over the rationals, it is clearly better to remove denominators for

working with integer coefficients, but one has to remove the gcd of the coefficients at some steps.

The data structures which lead to efficient computations.

The state of the art concerning these optimizations appears mainly in [Sugar]. However the research in this field is active and many improvement are implemented in some systems without being published.
In spite of these improvements, Gröbner bases computations need often very long computations and a lot of memory space. Thus further improvement are yet needed. We describe now two of them.

5.5.1 Modular computations (see also [La2])

As the size of the coefficients is frequently the limiting factor in Gröbner bases computations, it is natural to use modular technics.
However, there is no criterium for detecting primes of bad reduction, even if it is easy to prove that there are finite in number. Also, there is no sharp bound on the size of the coefficients of the result, which makes interpolation questionable.
Both difficulties are solved by tracing, i.e. by remembering the leading monomials of the polynomials which appear, the S–polynomials which are computed and the polynomial by which they are reduced [Trav].
Such a trace may be used in two ways :
The **first** one consists in doing several modular computations, throwing away those for which the trace differ and incrementally interpolate the modular results until the interpolated one does not change anymore.
Practically, this method works well, because almost all primes are of good reduction (if primes of the order of the machine word size are used, bad primes are very rare). However the result is only obtained with a very high probability, not proved.
The **second** method consists in doing only a few modular computations and in eliminating from the trace the S–polynomials which are not useful for computing the Gröbner basis , because they reduce to 0 or to a polynomial which is not used for the computation of the ones which appear in the final reduced base. Then this shorter trace is used for conducting non modular computations without wasting time for computing polynomials equal to 0.
With this method also, the result is not proved, but the polynomials which are obtained are certainly in the ideal generated by given polynomials.

For proving that the obtained result is correct, further computations are needed :

Verification that input polynomials reduce to 0 by the obtained base. This is rather easy.

Verification that the polynomials of the base are in the ideal generated by the input. This is only needed for first method.

Verification that the result is a Gröbner basis .

The last two verifications may be almost as difficult as computing the Gröbner basis base by not modular method. Several criteria for avoiding them may be found in [Trav].
A further criterium for the last verification, which does not appear in this paper, may be useful in the case of a finite number of solutions which are simple : Use the guessed base for computing the solutions and verify that these solutions are effectively solutions of the given system. This a posteriori verification works because, if a set of polynomials is not a
Gröbner basis , there are too many irreducible monomials and thus too many guessed solutions.

5.5.2 Change base algorithms

1) Change base algorithm by linear algebra (see also [La2] and [FGLM])

The second improvement to Gröbner basis algorithms is the algorithm for changing orderings for Gröbner bases [FGLM]. It works only in the zero–dimensional case (finite number of solutions).
Its main idea is that Buchberger's algorithm is much more efficient with degree orderings (*gradlex*, *revgradlex*) than with lexicographical ordering *lex*; on the other hand, lexicographical ordering is more suitable for computing the solutions of a system [Tr], the degree orderings (more precisely the *revgradlex*-order), gives only a general description of the ideal (dimension, number of solutions, Hilbert function, ...) but not the solutions. Thus, it is much more faster to compute the base for a degree ordering by Buchberger's algorithm, and to deduce from it the lexicographical base. This made first by means of linear algebra and recently by using Hilbert functions, [GMRT]. For this purpose, one could use Buchberger's algorithm again, but linear algebra is more efficient : let us recall that the quotient of a polynomial

ring by a zero–dimensional ideal is a finite vector space ; it has the irreducible monomials (for a Gröbner basis) as a linear base ; the expression of a polynomial on this base is simply obtained by reducing this polynomial ; finally, the elements of a reduced Gröbner basis are obtained by expressing the minimal (for divisibility) reducible monomials on this linear base.
Thus the change base algorithm [FGLM] is the following :

Generate the monomials by increasing new ordering and reduce them by old base.

If an obtained polynomial is linearly independent from the previous generated ones, then the corresponding monomial is irreducible for the new base.

If this obtained polynomial is dependent from the previous ones, then the dependence relation (between monomials) gives a member of the new Gröbner basis . Discard from the generating process all multiples of the corresponding monomial (which is the leading monomial of the new Gröbner basis member).

Stop when all monomials have been generated or discarded.

This algorithm stops eventually because the number of irreducible monomials and the number of leading monomials in a minimal Gröbner basis are both finite.
The computations are mainly linear algebra on the base of irreducibles monomials (for old Gröbner basis) and reduction operations. The data structures may be managed in order that the number of operations of this algorithm is polynomial in the number of irreducible monomials (number of solutions counted with multiplicities). If the size of the coefficients is taken into account, the complexity of this algorithm may be exponential in the number of variables, but only for very irregular sets of irreducible monomials ; for most entries, the complexity is polynomial in the number of irreducible monomials.
These complexity results are of practical interest : for most entries, the time needed by change base algorithm is much less than the time needed for computing the base for the degree ordering.

2) Change base algorithm by the algebra structure

A new algorithm for computing Gröbner basis by change of ordering that seems faster than the two previous ones, [FGLM] and [GMRT]. The main

idea of this algorithm is to use the algebra structure of the quotient space instead of linear algebra. This algorithm is proposed by Jean-Charles Faugère in 1994. It has made a very first implementation of the algorithm in AXIOM.

5.6 Some computational languages for the Gröbner bases

(1) MAPLE V.2 : Very slow see [CGG+]. To access to the Gröbner basis package, type

```
> with(grobner);
```

(2) MATHEMATICA 2.2.1: Slower than Maple. To access to the Gröbner basis package, type

```
In[1]:= GroebnerBasis[polylist,varlist]
```

(3) REDUCE 3.5 : [Fit] and [Hea]. We notice that REDUCE is faster with small homogeneous systems but cannot compute the cyclic 6 because the Buchberger algorithm is not implemended with sugar strategy. To access to the Gröbner basis package, type

```
1:  load groebner;
```

REDUCE is more performant than (1) and (2), because it has the fastest implementation and offers most possibilities, see also Appendix II.
(4) AXIOM 4.1 : [JSW], [JS]. Seems to be the best general computer algebra system for solving polynomials systems essentially because of a goal implementation of the sugar strategy. However, computations with $\mathbb{Z}/p\mathbb{Z}$ are not very fast.
(5) MACSYMA
(6) COCOA : Spezialized program, for Mac.
(7) ALPI 1.95 : [Alp], [TD], results found in the [GMRT].
(8) MAS 0.7 : A lot of function related to Gröbner basis , very slow.
(9) MACAULAY : [SB]. Very efficient for modular computations. No support for the integer case.
(10) GB : [Fau]. From Jean-Charles Faugère is the most efficiently actually.

As a conclusion we give the class of each systems as it appear in PoSSo '1995 in Paris, by Jean-Charles Faugère.

MATHEMATICA	4
MAPLE	4.5
MAS	4.5
MACAULAY	$\times$
REDUCE	5
ALPI	6(?)
AXIOM	6
GB	7

6 Applications of Gröbner bases

Let us return to the basic problems, see section 2, and see to what extent being able to compute Gröbner bases furnisches solutions.

6.1 The Ideal Membership Problem

As we mentioned in Remark 6, knowing a Gröbner bases $G = \{g_1, \ldots, g_s\}$ for an ideal I gives an algorithmic solution to the ideal membership problem. We need only compute a remainder to answer this question, since :

$$f \in I \Longleftrightarrow \bar{f}^G = 0$$

Example : Let $I = (f_1, f_2) = (xz - y^2, x^3 - z^2) \in \mathbb{R}[x, y, z]$, and $f = -4x^2y^2z^2 + y^6 + 3z^5$, we use lex-order. Our question is $f \in I$?
First we check if the generate set $\{f_1, f_2\}$ of ideal is a Gröbner basis of I.

$$S(f_1, f_2) = x^2y^2 - z^3 \text{ and } in(S(f_1, f_2)) = x^2y^2$$

If (f_1, f_2) is a Gröbner basis for I then $in(I)$ also contains $in(S(f_1, f_2)) = x^2y^2$, see proof of the theorem 4 section 5.3. But $x^2y^2 \notin (in(f_1), in(f_2)) = (xz, x^3)$, so the set $\{f_1, f_2\}$ is not a Gröbner basis of I. Hence we compute the Gröbner basis of I.

$$G = (f_2, f_3, f_4, f_1, f_5) = (x^3 - z^2, x^2y^2 - z^3, xy^4 - z^4, xz - y^2, y^6 - z^5)$$

We may now test polynomials for membership in I. Diving f by G we find :

$$f = 0f_1 + 0f_2 - 4z^2 f_3 + 0f_4 + 1f_5 + 0.$$

By the proposition 4, we have that $f \in I$. (Note that the G is a reduced Gröbner basis for I since both conditions of the definition 6 are satisfied.) Consider now the polynomial $f = xy - 5z^2 + x$. An easily manner to answer in our question if $f \in I$ is the following : if $f \in I$ then $in(f) \in (in(G))$, see proof of theorem 3 section 5.1, but $in(f) = xy \notin (in(G)) = (x^3, x^2y^2, xy^4, xz, y^6)$, so $f \notin I$.

6.2 The Problem of Solving Polynomial Equations

We will investigate how th Gröbner basis technique might be apllied to solve systems of polynomial equations of several variables. The real solutions of the system are coding by the interval method or by the Thom's coding method, see [LRR].

7 APPENDIX I : Gröbner bases Computation Using Syzygies

The most time consuming part of the Algorithm 2, see section 5.4, is the reduction of the S–polynomials. If an S–polynomial reduces modulo F, where F is a (finite) ideal basis of a polynomial ideal $k[x_1, \ldots, x_n]$, to the zero polynomial, its computation and reduction give no contribution to the final Gröbner basis . Such zero reduction can be avoided, if a criterion is available, which predits that the S–polynomial reduces to 0. Different kinds of criteria for avoiding many of these zero reductions are developped by Buchberger, [B4]. The crucial idea for involving syzygies in the detection of superfluous zero reductions can be easily understood by using an observation by Lazard [La1]. Since an ideal can be considered as an infinite dimensional vector spase, an ideal possesses simultaneously ideal bases and Gröbner bases. Lazard pointed out the connection between Gröbner bases and linear bases of the same ideal. Buchberger's algorithm can be considered as a triangularisation of a linear basis by Gaussian elimination. The reduction of a polynomial to zero means a linear dependence of this polynomial on the polynomials employed in the reduction procedure. Since all polynomials are here of type $t_i f_i$, $f_i \in F$, t_i a power product, the linear dependence relation can be written as a syzygial relation

$$\sum_{f_i \in F} g_i f_i = 0$$

where the g_i are linear combinations of some power products t_{ij}, i.e. they are polynomials. The vector $(g_1, \ldots, g_s)$ is called **syzygy** if $F = (f_1, \ldots, f_s)$. Interpreting every S–polynomial reduction to 0 as syzygy, we can predict every zero reduction provided we known a suitable basis of the module of syzygies.

Notation : Let $f, g \in F$.

$$f \longrightarrow_F^+ g$$

if $f = g$ or there exists a sequence $g_1, \ldots, g_t = g$ such that

$$f \longrightarrow_F g_1 \longrightarrow_F \ldots \longrightarrow_F g_t = g.$$

Criterion 1 : Let $f, g \in k[x_1, \ldots, x_n]$ such that $lcm(lm(f), lm(g)) = lm(f)lm(g)$, then $\overline{S(f,g)}^{\{f,g\}} = 0$.

Criterion 2 : Let $f, g, h \in k[x_1, \ldots, x_n]$ be such that

$$in(h) \text{ divides } lcm(in(f), in(g))$$

then $S(f, g)$ is linear combination with polynomial coefficients of $S(f, h)$ and $S(h, g)$.

If in the Buchberger's algorithm, we arrive at a collection of polynomials P containing three polynomials f, g, h satisfying $in(h)$ divides $lcm(in(f), in(g))$, and it has already been determined that

$$S(f, h) \longrightarrow_P^+ 0 \text{ and } S(h, g) \longrightarrow_P^+ 0$$

then it is not necessary to reduce $S(f, g)$ modulo P as well. By the criterion 2, $S(f, g)$ yields a syzygy that is already in the collection of syzygies generated by $S(f, h)$ and $S(h, g)$.

8 APPENDIX II : The computer algebra system REDUCE **3.5** (by F. Rouillier Univ. Rennes 1)

WARNING : This chapter gives any explanations, how to use REDUCE and his Gröbner basis package. It may not be enough, if you want to discover this computational system. For more informations see in an user‘s manual.

8.1 Loading and quitting

If you want to work with REDUCE, you have to call it with

$$REDUCE$$

In REDUCE, all command ends with a semicolon. The promt has the following form :

$$1 : < instruction > ;$$

If you want to load the Gröbner basis package, call it with

$$2 : \ load\ groebner;$$

If you want to quit REDUCE, write

$$3 : \ bye\ ;$$

8.2 Declarations

An number from the type integer may be in principle as long as wanted.
There exist some reserved variables. For example e (base of natural logarithm), i ($i^2 := -1$), infinity, nil (synonym of zero), pi, ...
If you want to do an assignment for a variable, you can do it as following :

1: a := x*y+3*z-2;
A **list** can be created with :
2: ListName := {a,b,c,d} ;
Lists are very usefull, in particulary in the Gröbner basis package. There exist some operations with the list:
3: part(ListName,n); which gives the n^{th} element of the list, or an error.
4: length(ListName); which gives the length of the list
5: rest(ListName); returns its argument with the first element removed, or an error.

6: a . {b,c}; or 7: a cons {b,c}; returns {a,b,c}
8: append({a,b},{c,d}); gives {a,b,c,d}
9: reverse({a,b,c}); restitues {c,b,a}
Array declarations in REDUCE are similar to Fortran dimension statements. For example 10: array a(d1,d2,d3);
Array indices each range from 0 to the value declared. All array elements are initialized at 0 at declaration time. If an array is referenced by an index outside its range, an error occurs.

The command **define** allows a user to supply a new name for any identifier or replace it by any well-formed expression. Its argument is a list of expressions of the form :
$< identifier >:=< number >|< identifier >|< operator >;$

8.3 Statements

The **group statement** is a construct used where REDUCE expects a single statement, but a series of actions needs to be performed. It is formed by enclosing one or more statements between << and >>. The statements are executed one after another.
The construction begin ... end is possible too. It is named a **compound statement**

Conditional statement :
11: if <boolean expression> then <statement> [else <statement>];
for statement :
12: for $\{< var >:=< number > \{step < number > until\} < number > \} < action >< expr >;$ or
13: for $\{each < var > in < list >\} < action >< expr >;$
where $< action >:= do \mid product \mid sum \mid collect \mid join.$
Instead for j:=a step 1 until b do ..., it is possible to write for j:=a:b do ...
while ... do, and **repeat ... until** :
14: while <boolean expression> do <statement>;
15: repeat <statement> until <boolean expression>;
return$(< value >)$ permits us to restitue a value calculated in a compound or if or group statement. With **end;** it is possible to stop an execution, without quitting REDUCE. And CTRL Z stops a programm without quitting REDUCE, with CTRL C the programm is stopped, but we quit REDUCE.
ws calls the last result, and **ws N** calls the result which was calculated in the prompt nummer N, if it exist.

8.4 Files

If you want to save some results in a file, write :
16: out "FileName.red" ;
17: f; g; h;
18: shut "FileName.red" ;
where FileName.red is the name of the file, and f, g and h are the saved results.
If you want to load a file,call it with :
19: in "FileName.red" ;

8.5 Options (switchs, on and off declarations)

If you work with integers, the default calculating mode is integer. If you want to have a rational result, you must ask it with
20: on rational ;
By default,results are given in the following form : $x^2 + 3 * x * y - 1$
There can be written like $x**2 + 3 * x * y - 1$. This is necessary, if you save the result in a file, and need to use it again latter, else REDUCE can't read it. For this, write
21: off nat ;
With
22: on time ; the computer indicates the time needed for executing an instruction, after each command. The time depends on the computer system.
23: on demo ; stop after each instruction, if a command contains many instructions.
24: on fact ; factorises the expressions, when it is possible.

8.6 The Groebner basis package

The Gröbner basis package is called in REDUCE with
25: load groebner;
In REDUCE, a monomial ordering is called a **term order**. REDUCE knows most of the possible monomial orderings, in particular **lex**, **gradlex** and **revgradlex**, which are so named. REDUCE can also work with products orders and weight orders. The default mode is lex. In REDUCE a term order is specified by means of the **torder** command.
For example
26: torder gradlex;

Since a monomial order depends also on how the variables are ordered, REDUCE needs to know both the term order and a list of variables. To indicate how the variables are ordered in a REDUCE command, include a list of variables in the input.
For computing a Gröbner basis , use the command **groebner**.
27: torder <order wanted>;
28: groebner(polylist,varlist);
or with
29: torder <gradlex or revgradlex>;
30: g:=groebner(polylist,varlist);
31: gb:=glexconvert g;
where $polylist := \{f_1, \ldots, f_m\}$ and $varlist := \{x_1, \ldots, x_n\}$.
glexconvert converts the Gröbner basis calculated with torder gradlex or revgradlex into the corresponding Gröbner basis for torder lex. It is in generally faster to calculate a Gröbner basis for torder lex with glexconvert.
If **gltbasis** is set on (off by default) the leading terms of the result basis are extracted. They are collected in a basis of monomials, which is available as value of the global variable with the name **gltb**.

Some other commands are:
preduce(f,polylist,varlist); gives the remainder of f on division by the polynomials in polylist using the monomial ordering specified by torder and varlist.
gsplit(f,varlist) computes lt(f) and f-lt(f)
gsort(f,varlist) prints out the terms of a polynomial according to the term order.
gspoly(f,g,varlist) calculates a S_polynom S(f,g)
greduce(f,polylist,varlist) gives the remainder of f on division by the Gröbner basis of the ideal generated by the input polynomials.
preducet(f,Gb-basis) can be used to find the quotients in the division algorithm.
gzerodim!?(Gb-basis,varlist) tests a Gröbner basis to see if the equations have finitely many solutions over an algebraically closed field.
groesolve(polylist,varlist) attemps to find all solutions of a system of polynomial equations.
idealquotient(polylist,expr,varlist) calculates the quotient of an ideal and an expression.
hilbertpolynomial(polylist,varlist) gives the Hilbert polynomial of an ideal

Remark 8 *preduce and groebner are the most used commands in the* REDUCE *Gröbner basis package. If the polynomial used in preduce or groebner have integer or rational coefficients,* REDUCE *will assume that we are working over the field* $\mathbf{Q}$*. There are no limitation in the size of the coefficients. Another possible coefficient field is the Gaussian rational numbers* $\mathbf{Q}(i)$*. Use the command*

32: on complex; before computing the Gröbner basis . Similary, to compute over a finite field with p elements, use
33: on modular; setmod p; REDUCE can also work with coefficients that lie in function rational fields. To tell REDUCE that a certain variable is in the base field (a "parameter"), omit it simply from the variable list varlist in the input.

9 References.

[Alp] Alpi : a system for experimenting with Buchbergeralgorithm. alpha version in Common Lisp available by anonymous ftp in gauss.dm.unipi.it (131.114.6.55)

[B1] Buchberger, B. (1965). Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. *Ph. D. Thesis,* Innsbruck.

[B2] Buchberger, B. (1970). Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleischunssystem, *Aeq. Math.* **4**, 374–383.

[B3] Buchberger, B. (1979). A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Basis. *Proc. EUROSAM 79, Lect. Notes in Comp. Sci.* **72**, Springer Verlag, 3–21.

[B4] Buchberger, B. (1985). Gröbner Bases : an Algorithmic Method in Polynomial Ideal Theory. In *Recent trends in multidimensional system theory,* Bose Ed., Reidel.

[BCK] B. Buchberger, G. E. Collins and B. Kutzler. (1988). Algebraic Methods for Geometric Reasoning. *Ann. Rev. Comput. Sci. 1988, 3:85 119.*

[CGG+] B. Char, K. Geddes, G. Gonnet, B. Leong, M. Monagan and S. Watt. *Maple V Library Reference Manual.* Springer-Verlang, 1991. 3^{th} printing, 1993.

[CGH1] Caniglia, L., Galligo A. and Heintz J. (1989a). Some new effectivity bounds in computational geometry. *Proceedings of AAECC-6, Roma 1988, Lect. Notes in Comp. Sci.***357**, Springer Verlag, 131–152.

[CGH2] Caniglia, L., Galligo A. and Heintz J. (1989b). How to compute the projective closure of an affine algebraic variety in subexponential time. To appear in *Proceedings of AAECC-7, Toulouse, 1989*

[CLO] Cox, Little, O'Shea. Ideals, Varieties, and Algorithms. *Undergrduate Texts in Mathematics. Springer Verlag, 1992.*

[D1] Davenport, J.H. (1987). Looking at a set of equations. Technical report 87–06, University of Bath.

[DST] Davenport, J.H., Siret, Y. and Tournier, E. (1986). Calcul Formel, Systèmes et Algorithmes de Manipulation Algébriques. Masson. English translation : Computer Algebra. Academic Press, 1988.

[Fau] Faugère, J.C. *On line documentation of GB*. Available at http://posso.ibp.fr/Gb.html.

[FGLM] Faugère, J.C., Gianni, P., Lazard, D. and Mora, T. (1988). Efficient computation of zero–dimensional Gröbner bases by change of ordering. Technical Report LITP 89–52 *submitted to J. Symb. Comp.*

[Fit] Fitch J. Solving algebraic promblems with Reduce. *J. Symb. Comp.*, 1:211-277, 1985.

[Gian] Gianni, P. (1987). Properties of Gröbner bases under specializations. *European Conference on Computer Algebra, Leipzig, GDR, 1987.* (J.H. Davenport, ed.) Lect. Notes in Comp. Sc. **378**, 293–297.

[GM] Gianni, P. and Mora T. (1987). Algebraic solution of systems of polynomial equations using Gröbner bases. *Applied Algebra, Algebraic Algorithms and Error–Correcting Codes (AAECC 5), Menorca, Spain, 1987.* Lect. Notes in Comp. Sc. **356**, 247–257.

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

[GTZ] Gianni, P., Trager, B. and Zacharias, G. (1988). Gröbner bases and primary decomposition of polynomial ideals. *J. Symb. Comp.* **6**, 149–167.

[Hir] Hironaka H. (1964). Resolution of Singularities of an Algebraic Variety over a Field of Characteristic Zero I, II. *Ann. Math.*, **79**, pp. 109-326.

[Hea] Hearn A. Reduce user's manual, Version 3.3. *The RAND Corp.*, report cp 78 edition, Jully 1987.

[JL] Jacobsson, C., Löfwall, C. (1991). Standard Bases for general coefficient rings and a new constructive proof of Holbert's basis theorem. *J. Symb. Comp.* **12**, 337-371.

[JS] Jenks R., Sutor R. Axiom, the Scientific Computation System. *Springer-Verlang*, 1992.

[JSW] Jenks R., Sutor R, Watt M. Scratchpad II: An abstract databype system for mathematical computation. *In R. Janβen, editor, Trends in Computer Algebra, Proc. Internat. Symp.*, pages 12-37, Bad Neuenahr, May 1987. Springer L.N.C.S. 296.

[KFF] Kobayashi, H., Fujise, T. and Furukawa, A. (1988). Solving systems of algebraic equations by general elimination method. *J. Symb. Comp.* **5**, 303–320.

[KMH] Kobayashi, H., Moritsugu, S. and Hogan, R.W. (1988). On Solving Systems of Algebraic Equations. *Proc. ISSAC 88.*

[La1] Lazard, D. (1983). Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations. Proc. EUROCAL 83. Lect. Notes in Comp. Sci. **162**, Springer, 146–157.

[La2] Lazard, D. (1989). Solving zero–dimensional algebraic systems. *To appear.*

[La3] Lazard, D. (1990). A new method for solving algebraic systems of positive dimension. Technical Report LITP 89–77 *To appear in Discr. App. Math.*

[LRR] Z. Ligatsikas, R. Rioboo & M.-F. Roy (1993). Generic Computation of the Real Closure of an Ordered Field, in *Symbolic Computation - New Trends and Developments*, International IMACS Symposium SC 93, Lille France June 14-17, 1993.

[Rob] Robbiano, L. (1985). Term orderings on the polynomial ring. *Proceedings of Eurocal'85 (Linz).* Lect. Not. in Comp. Sc. **204**, 513-517.

[SB] Stillman M. and Bayer D. Macaulay User Manual, 1989. Available by anonymous ftp on : zarinski.harvard.edu

[Sugar] Giovini, A., Mora, T., Niesi, G., Robbiano, L. and Traverso, C. (1991). "A sugar cube please" or Selection strategies in the Buchberger algorithm. *Proc. ISSAC'91*, to appear.

[TD] Traverso C. and Donati L. Experimenting the Gröbner basis algorithm with Alpi system. *In ACM Press, editor, Proceedings of the*

1986 International Symposium on Symbolic and Algebraic Computation. ISSAC '89, 1989.

[Trav] Traverso, C. (1988). Gröbner trace algorithms. *Proceedings of ISSAC 88, Roma.* Lect. Notes in Comp. Sc. **358**, 125–138.

[Tr] Trinks, W. (1978). Über B. Buchbergers Verfahren, Systeme Algebraischer Gleichungen zu Lösen. *J. Number Th.* **10**, 475–488.

[Mö] Möller, H.M. (1989). To appear.

[W] Wilkinson, J.H. (1959). The Evaluation of the Zeros of III–conditioned polynomials. *Num. Math.* **1**, 150–166, 167–180.

[WT] Wu Wen–Tsün (1987). A Zero Structure Theorem for Polynomial Equation Solving. *Math.–Mechanization Research Preprints* **1**, 2–12, Academica Sinica, Beijing.