The Computation of Gröbner Bases on a Shared Memory Multiprocessor.

Jean-Philippe Vidal

August 28, 1990

CMU-CS-90-163 Z

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The principal result described in this report is the design and implementation of a parallel version of Buchberger's algorithm. Its correctness is stated and some experimental results are given. The first parts are devoted to a partial review of Gröbner bases, of Buchberger's algorithm which computes them, and of some of their applications.

Keywords: Algebraic Manipulation, Mathematical Software, Parallel Algorithms, Roots of Nonlinear Equations, Shared Memory, Special Purpose Algebraic Systems, Systems of Equations.

Contents

The computation of Gröbner bases on a shared memory multiprocessor

As soon as the author learned about Gröbner bases, he became interested by the details of their computation. The algorithm has a relatively simple form. But the way it behaves on particular examples is a complex problem which has not yet been satisfactory solved and which requires at least some elementary knowledge of algebraic geometry.

In this paper, we first review some concepts of commutative algebra which will be used later on. We then look at rewriting systems in polynomial rings and study their termination and normal form properties. This leads us to define admissible orders and Gröbner bases and to present Buchberger's algorithm. In the third section, three applications are listed: finding the intersection of two ideals, finding the quotient of two ideals, and determining the ideal of the polynomial relations among a given list of polynomials. In the next section, we look in more detail at a fourth application: the computation of the Hilbert function of an ideal (but without covering the recent developments in the determination of the dimension). The last part presents an original work: the design and preliminary implementation of a parallel algorithm to compute Gröbner bases on shared memory multiprocessors. Performance data on a reasonable set of examples are discussed.

We would like to thank Professor Dana Scott for introducing us to the subject and for his help in reviewing this paper: his comments have led to great improvements in its structure. It is also a pleasure to acknowledge the encouragement we have received from Professor Edmund Clarke for our work on the parallel version of Buchberger's algorithm. Our best thanks to Professor Bruno Buchberger who invited us on August 1987 at RISC-Linz and even gave us the listing of an implementation of the sequential algorithm [11].

1. Review of commutative algebra.

Rings are assumed to be commutative. When we use modules, we will consider them to be right-modules. This is not very relevant, as we will mainly be interested in modules which are in fact commutative rings considered as modules with respect to themselves.

1.1. Noetherian modules and rings.

The following definitions and theorems appear, for instance, on the chapter 3 of Jacobson's book, [21].

Definition 1.1 A module M satisfies the *ascending chain condition* if there is no infinite properly ascending chains $M_1 \subset M_2 \ldots \subset M_i \subset \ldots$ of submodules of M. Such a module M is called *Noetherian.*

Definition 1.2 A module M satisfies the *maximum condition* if every non-vacuous set of submodules of M contains a maximal submodule (i.e. a submodule not contained in any other submodule of the set).

Definition 1.3 A submodule M' of a right-module M for a ring R is *finitely generated* if there exist n members of M', $m_1, m_2, \ldots, m_n$, such that $M' = m_1 R + m_2 R + \ldots + m_n R$.

Proposition 1.1 *The following three statements are equivalent:*

1. *The module M satisfies the ascending chain condition.*
2. *The module M satisfies the maximum condition.*
3. *Every submodule M' of M is finitely generated.*

Definition 1.4 A ring R is *Noetherian* if and only if it is a Noetherian module with respect to itself.

Because a submodule of R (considered as a module with respect to itself) is an ideal of R (considered as a ring), we can state the following proposition :

Proposition 1.2 *The following three statements are equivalent:*

1. *The ring R satisfies the ascending chain condition (meaning that there is no infinite properly ascending chain of ideals of R.)*
2. *The ring R satisfies the maximum condition (meaning that every non-empty set of ideals of R contains a maximal element).*
3. *Every ideal I of R is finitely generated.*

1.2. Hilbert's basis theorem.

The following result is an immediate extension of the theorem stated by Hilbert, which concerned only special rings R (namely fields and the ring of integers).

Theorem 1.1 *If R is a Noetherian ring, then $R[x_1, x_2, \ldots, x_n]$ is a Noetherian ring.*

There are several proofs of this theorem. The original one, by Hilbert, uses the third characterization of Noetherian rings (every ideal has a finite basis) and works by induction on the number of variables. It can be found in the book of Nathaniel Jacobson [21]. Another proof appears in Commutative Algebra I, by Zariski and Samuel, Chapter IV, [42]. It works also by induction on the number of variables but uses the first characterization of Noetherian rings (there is no infinite properly ascending chain of ideals). Other authors (see, for example, Giusti, [17]) use Dickson's lemma (1913), which states that, in a finitely generated commutative monoid, for any sequence $\{T_n\}$, $n \in N$, there exists $p \in N$ such that, for all $n > p$, the term T_n is a multiple of one of the terms $T_0, \ldots, T_p$. The proof of Dickson's lemma works by induction over the number of generators of the monoid.

We choose to present a slight modification of Zariski and Samuel's proof:

Definition 1.5 An ideal of $R[x]$ is called a *monomial* ideal if it can be generated by a set of monomials.

Definition 1.6 If I is an ideal of $R[x]$, the monomial ideal generated by the monomials of highest degree of the polynomials of I is called the *initial ideal* of I and is denoted by $In(I)$.

Lemma 1.1 *If I and J are two ideals of $R[x]$ such that $I \subseteq J$ and $In(I) = In(J)$, then $I = J$. Therefore, if there exists an infinite properly ascending chain of ideals in the polynomial ring $R[x]$, then there exists an infinite properly ascending chain of monomial ideals in $R[x]$.*

Proof of the lemma: We suppose that $I \subseteq J$ and that $In(I) = In(J)$ and we use induction on the degree of an eventual polynomial belonging to $J - I$ (if any). Note that $0 \notin J - I$. We suppose that there is no polynomial of degree less than or equal to p which belongs to $J - I$, that P of degree $p+1$ belongs to $J - I$, and we try to derive a contradiction. $In(P)$, the monomial of highest degree of P, belongs to $In(I)$ and, therefore, there exists a polynomial $Q \in I$ such that $P - Q$ is of degree less than or equal to d. But $P - Q$ belongs to J and, because of the induction hypothesis, to I. So $P \in I$ and we obtain a contradiction. □

Lemma 1.2 *If R is a Noetherian ring, there is a bijection between the monomial ideals of $R[x]$ and the eventually stationary ascending chains of ideals of R. Moreover, if I and J are two ideals such that $I \subseteq J$, and if $I_0 \subseteq I_1 \subseteq \ldots \subseteq I_i \subseteq \ldots$ and $J_0 \subseteq J_1 \subseteq \ldots \subseteq J_i \subseteq \ldots$ are the two eventually stationary ascending chains associated respectively with I and J, then we have the following diagram:*

$$\begin{array}{ccccccccc}
I_0 & \subseteq & I_1 & \subseteq & \ldots & \subseteq & I_i & \subseteq & \ldots \\
| \cap & & | \cap & & & & | \cap & & \\
J_0 & \subseteq & J_1 & \subseteq & \ldots & \subseteq & J_i & \subseteq & \ldots
\end{array}$$

Proof of the lemma: Let's consider a monomial ideal I of $R[x]$. We will call I_i the ideal of R whose elements are the leading coefficients of the polynomials of degree i of I. $I_0 \subseteq I_1 \subseteq \ldots \subseteq I_i \subseteq \ldots$ is an ascending chain of ideals and, as R is Noetherian, this chain as to be eventually stationary. Naturally, to two different monomial ideals correspond two different chains and, for every eventually stationary ascending chain $I_0 \subseteq I_1 \subseteq \ldots \subseteq I_i \subseteq \ldots$, we can associate the monomial ideal generated by $\{r\, x^i \mid r \in I_i,\ i \geq 0\}$. □

Proof of the theorem: Because of lemma 1.1, we only need to prove that, if R is Noetherian, there is no infinite properly ascending chain of monomial ideals in $R[x]$. Let's suppose that $I^1 \subseteq I^2 \subseteq \ldots \subseteq I^p \subseteq \ldots$ is an ascending chain of monomial ideals of $R[x]$. Then, using lemma 1.2, we can associate with each I^p the chain $I_0^p \subseteq I_1^p \subseteq \ldots \subseteq I_i^p \subseteq \ldots$ of ideals of R such that they form the following diagram:

$$\begin{array}{ccccccccc}
I_0^1 & \subseteq & I_1^1 & \subseteq & \ldots & \subseteq & I_i^1 & \subseteq & \ldots \\
| \cap & & | \cap & & & & | \cap & & \\
I_0^2 & \subseteq & I_1^2 & \subseteq & \ldots & \subseteq & I_i^2 & \subseteq & \ldots \\
| \cap & & | \cap & & & & | \cap & & \\
\ldots & & \ldots & & \ldots & & \ldots & & \\
| \cap & & | \cap & & & & | \cap & & \\
I_0^p & \subseteq & I_1^p & \subseteq & \ldots & \subseteq & I_i^p & \subseteq & \ldots \\
| \cap & & | \cap & & & & | \cap & & \\
\ldots & & \ldots & & \ldots & & \ldots & &
\end{array}$$

Let $stat\,(p)$ be the smallest integer such that the chain associated with I^p becomes stationary. It is easy to see that $I^1_{stat(1)} \subseteq I^2_{stat(2)} \subseteq \ldots \subseteq \ldots I^p_{stat(p)} \subseteq \ldots$. This chain is eventually stationary. Let l be the integer such that for all $p > l$ and for all $i > stat(l)$, we have $I^p_i = I^l_{stat(l)}$. Then we consider, for $0 \leq i < stat(l)$, the vertical chains $I^1_i \subseteq I^2_i \subseteq \ldots \subseteq I^p_i \subseteq \ldots$. Each one of these chains is stationary for $p \geq stat'(i)$.

Then, for p and p' greater than $sup(l, stat'(0), \ldots, stat'(stat(l)-1))$, the two horizontal chains associated with I^p and $I^{p'}$ are equal and, therefore, the two associated monomial ideals I^p and $I^{p'}$ are also equal. So the chain of monomial ideals is eventually stationary. □

1.3. Affine algebraic varieties and Zariski topology.

The results of this subsection can be found in Fulton, [13] and in Jacobson, [21].

Let K be any field. We will call K^n the n-dimensional *vector space* of n-tuples of members of K. It is the *affine* n-space over K and its elements are called *points*. If P is a polynomial of $K[x_1, \ldots, x_n]$, we will say that $(k_1, \ldots, k_n)$ is a *zero* of P if and only if $P(k_1, \ldots, k_n) = 0$. If P is not a constant, then the set of zeros $V(P)$ of P is called a *hypersurface* (when $n = 2$, it is also called an *affine plane curve*). If P is of degree 1, then $V(P)$ is a *hyperplane* (a *line*, if $n = 2$).

More generally, we can give the following definition:

Definition 1.7 If B is a set of polynomials of $K[x_1, \ldots, x_n]$, the *affine algebraic variety defined by* B is the set $V(B)$ of common zeros of the polynomials in B:

$$V(B) = \bigcap_{P \in B} V(P)$$

Proposition 1.3 *The following properties are immediate:*

1. $V(K[x_1, \ldots, x_n]) = \emptyset$.
2. $V(\emptyset) = K^n$.
3. *If I is the ideal generated by B, then $V(B) = V(I)$.*
4. *If $\{B_i\}_{i \in E}$ is any collection of subsets of $K[x_1, \ldots, x_n]$, then $V\left(\bigcup_{i \in E} B_i\right) = \bigcap_{i \in E} V(B_i)$.*
5. *If P and Q are two polynomials of $K[x_1, \ldots, x_n]$, then $V(P\,Q) = V(P) \cup V(Q)$.*
6. *If I and J are two ideals of $K[x_1, \ldots, x_n]$, then $V(I) \cup V(J) = V(\{P\,Q | P \in I, Q \in J\})$.*
7. $\{(k_1, \ldots, k_n)\} = V(\{x_1 - k_1, \ldots, x_n - k_n\})$.

Note that property 3 above implies that every affine algebraic variety is a variety defined for a certain ideal I. From Hilbert's basis theorem and property 4 above follows that any such variety is an intersection of hypersurfaces. Finally, we can deduce from property 7 that any finite subset of K^n is an affine algebraic variety.

Proposition 1.4 *If S is a subset of K^n, the subset $\mathcal{I}(S)$ of $K[x_1,\ldots,x_n]$ which contains all the polynomials P such that $\forall\,(k_1,\ldots,k_n) \in S,\ P(k_1,\ldots,k_n) = 0$ is an ideal.*

Proposition 1.5 *The following properties are immediate:*

1. $\mathcal{I}(\emptyset) = K[x_1,\ldots,x_n]$.
2. $S \subseteq V(\mathcal{I}(S))$.
3. $I \subseteq \mathcal{I}(V(I))$.
4. *If* $S_1 \subseteq S_2$, *then* $\mathcal{I}(S_2) \subseteq \mathcal{I}(S_1)$.
5. *If* V_1 *and* V_2 *are two algebraic varieties such that* $V_1 \subset V_2$, *then* $\mathcal{I}(V_2) \subset \mathcal{I}(V_1)$.

In general, $\mathcal{I}(K^n) \neq 0$. For example, $x^2 + x \ \in \mathcal{I}(\mathbb{Z}_2)$. But the equality holds whenever K is an infinite field. This comes from the fact that if K is an infinite field and if $P \in K[x_1,\ldots,x_n]$ is not equal to 0, then there is a n-tuple $(k_1,\ldots,k_n) \in K^n$ such that $P(k_1,\ldots,k_n) \neq 0$ (see Jacobson, [20], page 136)

From the proposition 1.3, we can deduce that the affine algebraic varieties satisfy the conditions required from the closed sets of a topological space. This topology on K^n is called *Zariski topology*. The sets $O_P = \{(k_1,\ldots,k_n) \in K^n | P(k_1,\ldots,k_n) \neq 0\}$ form a basis of the open sets. Notice that, if K is not algebraically closed, we obtain $O_P = K^n$ for certain polynomials P (for example, if $K = \mathbb{R}$ and $P = x_1^2 + 1$). This topology has the following properties:

Proposition 1.6 1. *K^n is a T_1-space (i.e., for each pair of points k and k', there exist neighbourhoods $\mathcal{V}_k$ and $\mathcal{V}_{k'}$ of k and k' such that $k \notin \mathcal{V}_{k'}$ and $k' \notin \mathcal{V}_k$).*

2. *If K is an infinite field, K^n is not a Hausdorff space and, moreover, any two proper open subsets of K^n intersect. An Hausdorff space, or T_2-space, is such that for any two distinct points k and k', there exist two disjoint neighborhoods $\mathcal{V}_k$ and $\mathcal{V}_{k'}$ of k and k'.*

3. *K^n with the Zariski topology is Noetherian, in the sense that there is no infinite descending chain of varieties.*

Proof: We saw that $\{(k_1,\ldots,k_n)\} = V(\{x_1 - k_1,\ldots,x_n - k_n\})$. That means that the closure of each point is itself. Therefore its complement is an open set which does not contain the point and which contains all the other points of the space. This proves the first statement.

Next, it is sufficient to show that any two members of the basis intersect. Suppose P and Q are two non-zero polynomials of $K[x_1,\ldots,x_n]$ where K is an infinite field. Then, there exists $k \in K^n$ such that $P(k)Q(k) \neq 0$ (see [20], page 136). Then $O_P \cap O_Q = O_{PQ} \neq \emptyset$.

The third result is easily obtained from proposition 1.5 above and the fact that $K[x_1,\ldots,x_n]$ is Noetherian. □

1.4. Prime ideals and nil radicals.

The following definition generalizes the arithmetical meaning of prime. In the ring of integers $\mathbb{Z}$, the set $p\mathbb{Z}$ is a prime ideal if and only if p is prime. More generally:

Definition 1.8 An ideal I of a commutative ring R is *prime* if and only if:

$$\forall\, r_1, r_2 \in R,\ r_1 r_2 \in I \Longrightarrow r_1 \in I \text{ or } r_2 \in I$$

Proposition 1.7 *1. If I is a prime ideal of a commutative ring R, the complement $R\backslash I$ of I in R is multiplicatively closed.*

2. If B is a non empty multiplicatively closed set of a commutative ring R, an ideal I which does not intersect B and is such that any ideal containing it intersects I is a prime ideal.

Proof: The first part is a direct consequence of the definition of a prime ideal. Now, suppose that B and I verify the conditions of the second part of the proposition and that $r_1 \notin I$ and $r_2 \notin I$. Then $(r_1) + I \cap B \neq \emptyset$ and $(r_2) + I \cap B \neq \emptyset$. Let $r_3 \in (r_1) + P \cap B$ and $r_4 \in (r_2) + P \cap B$. Then $r_3\, r_4 \in (r_1\, r_2) + I \cap B$ and, therefore, $r_1\, r_2 \notin I$. □

Here is an example. Prime ideals of $\mathbb{Z}$ are the subrings $p\mathbb{Z}$ for p prime. Suppose that we take, as multiplicatively closed set of $\mathbb{Z}$, the set of powers of 6, $B = \{1, 6, 36, 6^3, \ldots, 6^n, \ldots\}$. We can see that the complement of B in $\mathbb{Z}$ is not an ideal ($3 \notin B$ but $3 + 3 \in B$). The ideals of $\mathbb{Z}$ not intersecting B are the ideals $n\,\mathbb{Z}$ such that $n \wedge 6 \neq 1$. Therefore, the prime ideals not intersecting B are exactly the ideals $p\,\mathbb{Z}$ such that p is a prime distinct from 2 and 3.

Definition 1.9 An element r of a commutative ring R is *nilpotent* if and only if there exists a natural number n such that $r^n = 0$. The nil radical, $nilrad(R)$, of a commutative ring R is the set of its nilpotent elements.

Theorem 1.2 (Krull) *The nil radical of R is the intersection of the prime ideals of R.*

Proof: First assume that r is a nilpotent element of R ($r^n = 0$) and J a prime ideal of R. Then $r^n = 0 \in J$, and so $r \in J$.

Next, suppose r is not nilpotent. The set of ideals of R not intersecting the multiplicative monoid generated by r is not empty (the null ideal belongs to it) and is inductive (i.e. the union of any chain of members of this set belongs to the set). Therefore, we can use Zorn's lemma to prove the existence of a maximal element J in this set. By proposition 1.7, J is prime and $r \notin J$. Notice that, if R is Noetherian, we do not need Zorn's Lemma. □

Definition 1.10 The *nil radical* of an ideal I of R is the set of elements r of R for which there exists a natural number n such that $r^n \in I$.

Note that the nil radical of an ideal is the preimage of the nil radical of R/I by the canonical homomorphism of R onto R/I.

Proposition 1.8 *The nil radical of the ideal I is equal to the intersection of all the prime ideals of R containing I.*

Proof: it is well-known that the canonical homomorphism ν from R to R/I maps ideals of R containing I to ideals of R/I in a one-to-one way. Moreover, it can be proven easily that ν maps prime ideals of R containing I to prime ideals of R/I in a one-to-one way. From this, the proposition follows immediately.□

1.5. Prime spectrum of a commutative ring.

Let $X(R)$ denote the set of prime ideals of R. For every subset B of R, let $Pr(B)$ be the set of all prime ideals of R containing B. $Id(B)$ or more simply (B) denote the ideal generated by the set B.

Proposition 1.9 *The following are immediate:*

1. $\emptyset = Pr(\{1\})$ *and* $X(R) = Pr(\{0\})$.
2. *If* $\{B_i\}_{i\in E}$ *is a collection of subsets of* R, $\bigcap_{i\in E} Pr(B_i) = Pr(\bigcup_{i\in E} B_i)$.
3. *If* B_1 *and* B_2 *are two subsets of* R, $Pr(B_1) \cup Pr(B_2) = Pr(Id(B_1)Id(B_2))$.

We see from this result that the sets $Pr(B)$ can be considered as the closed sets of a topology that they determine on $X(R)$.

Definition 1.11 $X(R)$ with the topology just defined is called the *spectrum of* R, and the subset $X_{max}(R)$ consisting of maximal ideals with the induced topology is called the *maximum spectrum.*

The open sets are the complements $X(R) - Pr(B) = \bigcup_{r\in B}(X(R) - Pr(\{r\}))$. The set $X(R) - Pr(\{r\})$ is just the set of prime ideals not containing $r \in R$. We call it X_r and we note that $\{X_r \mid r \in R\}$ is a base of the open sets of $X(R)$. Such topologies were introduced by M.H. Stone for Boolean algebras (in which case $X_{max}(R)$ is a compact Hausdorff space with a basis of clopen subsets), and by N. Jacobson for arbitrary rings.

The space $X(R)$ is not always a Hausdorff space. In particular, if R is a domain, then 0 is a prime ideal of R whose closure is the whole space $X(R)$. Therefore, 0 can't be separated from any other prime ideal and $X(R)$ is not even a T_1-space.

Even $X_{max}(R)$ is not always a Hausdorff space. For example, the maximal ideals of $\mathbb{Z}$ are the ideals (p) where p is prime. Therefore, $X_{max}(\mathbb{Z})$ is a T_1-space. The closure of any infinite set of maximal ideals of $\mathbb{Z}$ is the whole space (because 0 is the only integer which belongs to an infinite number of prime ideals (p) and because $Pr(0) = X_{max}(\mathbb{Z})$). So the closed sets are $\emptyset$, the finite subsets and the whole space $X_{max}(\mathbb{Z})$. So the proper open sets are complementary of finite subsets and, therefore, any two of them intersect. So, $X_{max}(\mathbb{Z})$ is not an Hausdorff space.

Proposition 1.10 $X(R)$ *and* $X_{max}(R)$ *are compact spaces.*

Proof: We need only to prove that for every open covering of $X(R)$ with elements of the base $\{X_r \mid r \in R\}$, there is a finite open covering contained in it. Suppose $X(R) = \bigcup_{r \in B} X_r$. Then $X(R) = X(R) - Pr(B)$. So $Pr(B) = Pr(I) = \emptyset$, where I is the ideal generated by B. So $I = R$ and there exist $r_1, \ldots, r_n \in B$ and $s_1, \ldots, s_n \in R$ such that $1 = \sum_{i=1}^{n} r_i \; s_i$. Finally $Pr(I) = Pr(\{r_1, \ldots, r_n\})$ and $X(R) = \bigcup_{i=1}^{n} X_{r_i}$.

The proof for $X_{max}(R)$ is similar. □

1.6. Hilbert's Nullstellensatz.

We return to affine algebraic varieties to see how the concept of nil radical of an ideal can be used in this context. It's easy to see that, for any ideal I of $K[x_1, \ldots, x_n]$ (K is any field), $V(I) = V(nilrad(I))$. We introduced previously the ideal $\mathcal{I}(V)$ as the ideal of all polynomials vanishing on V. Hilbert's Nullstellensatz states that, when K is algebraically closed, $\mathcal{I}(V(I)) = nilrad(I)$.

Theorem 1.3 (Nullstellensatz) *Let K be an algebraically closed field and I be an ideal in the polynomial ring $K[x_1, \ldots, x_n]$. Suppose $Q(k_1, \ldots, k_n) = 0$ for all $(k_1, \ldots, k_n) \in V(I)$. Then $Q \in nilrad(I)$.*

There is a very similar presentation of this theorem, called sometimes *Weak Nullstellensatz* (at least, in [13]):

Theorem 1.4 *If I is a proper ideal in $K[x_1, \ldots, x_n]$, then $V(I) \neq \emptyset$.*

Following Fulton, we prove the Weak Nullstellensatz theorem first and use it to prove the main theorem. We need a lemma, due to Zariski, on fields which are ring-finite over one of their subfields. A ring R is *ring-finite* over one of its subrings R', if there exist n elements of R, $r_1, \ldots, r_n$ such that the ring homomorphism ϕ from $R'[x_1, \ldots, x_n]$ to R given by $\phi(P) = P(r_1, \ldots, r_n)$ is onto (we will write that $R = R'[r_1, \ldots, r_n]$). We also say that R is module-finite over R' if R is a finitely generated R'-module.

Lemma 1.3 *If a field K is ring-finite over a subfield K', then K is module-finite over K'.*

Proof of the lemma: Notice first that, if R is a domain which is ring-finite over one of its subrings R', then R is module-finite over R' if and only if R is integral (in the case of fields, we say algebraic) over R' ([13], chapter 1, page 30, exercise 1-48). The implication from right to left is easy. The other one is proved by considering a particular generating set of R (as a R'-module), $\{s_1, \ldots, s_p\}$ and, for any $r \in R$ the system of linear equations $\{s_i \; r - \sum_{j=1}^{p} r_{ij} s_j = 0\}_{1 \leq i \leq p}$. As $(s_1, \ldots, s_p)$ is a non-trivial solution of this system, considered in the quotient field of R (R is a domain), the determinant (which is in $R'[r]$) is equal to zero.

Now, suppose that $K = K'[k_1, \ldots, k_n]$. The proof works by induction on n. For $n = 1$, we can suppose than $k_1 \neq 0$ and we can express $1/k_1$ as a member of $K'[k_1]$, so there is a polynomial $P \in K'[x]$ such that $1/k_1 = P(k_1)$. From this, we see that k_1 is algebraic over K' and that K is a K'-module of dimension at most $degree(P) + 1$.

Next, if $n > 1$, $K = K'[k_1, \ldots, k_n] = K'(k_1)[k_2, \ldots, k_n]$. By induction hypothesis, $K'(k_1)[k_2, \ldots, k_n]$ is a finitely generated $K'(k_1)$-module. If k_1 is algebraic over K', the result is immediate.

Otherwise, we know that there exist monic polynomials in $K'(k_1)[x]$, $P_2, \ldots, P_n$ s.t. $P_i(k_i) = 0$. Let $Q \in K'[k_1]$ be the common denominator of all the fractions appearing in $P_2, \ldots, P_n$. $Q \neq 0$ because k_1 is not algebraic over K'. From this, we can find monic polynomials in $K'[k_1][x]$, $P_2', \ldots, P_n'$ such that $P_i'(Q\ k_i) = 0$. As the set of integral elements over $K'[k_1]$ is a subring of K containing $K'[k_1]$, we see that for any $k \in K = K'[k_1, \ldots, k_n]$, there is an integer p such that $Q^p k$ is integral over $K'[k_1]$. For $k = 1/P(k_1)$ where P is a polynomial in $K'[x]$ such that Q and P have no common factor, this gives a contradiction with the fact that k_1 is not algebraic over K'. □

Proof of the Weak Nullstellensatz: I is contained in a maximal ideal J and $V(J) \subseteq V(I)$. Now, $K[x_1, \ldots, x_n]/J$ is a field which is ring-finite over K. From the lemma, it is algebraic over K and, as K is algebraically closed, $K[x_1, \ldots, x_n]/J = K$. The residue of x_i modulo J is $k_i \in K$. Then $x_i - k_i \in J$, for all $1 \leq i \leq n$. But $(x_1 - k_1, \ldots, x_n - k_n)$ is a maximal ideal, so $J = (x_1 - k_1, \ldots, x_n - k_n)$. Then $V(J) = \{(k_1, \ldots, k_n)\} \neq \emptyset$. □

Proof of the Nullstellensatz: This proof is adapted from Rabinowitsch. Suppose the ideal I of $K[x_1, \ldots, x_n]$ is generated by $P_1, \ldots, P_p$. Consider the ideal J generated by $P_1, \ldots, P_p, x_{n+1}\ Q - 1$ in $K[x_1, \ldots, x_n, x_{n+1}]$. As Q vanishes on $V(I) \subseteq K^n$, $V(J) = \emptyset$ (this is a subset of K^{n+1}). From the weak Nullstellensatz, $1 \in J$, so there exist polynomials $Q_0, \ldots, Q_p \in K[x_1, \ldots, x_{n+1}]$ such that $1 = Q_0\ (x_{n+1}\ Q - 1) + \sum_{i=1}^{p} Q_i P_i$. Now, we take the quotient of $K[x_1, \ldots, x_{n+1}]$ by the ideal $(x_{n+1}\ Q - 1)$. By multiplying each side of the above equation by the suitable power of $\overline{Q}$, we obtain that there exist an integer i and polynomials $R_1, \ldots, R_p$ in $K[x_1, \ldots, x_n, x]$ such that the equation $\overline{Q}(\overline{x}_1, \ldots, \overline{x}_n)^i = \sum_{i=1}^{p} \overline{R}_i(\overline{x}_1, \ldots, \overline{x}_n, \overline{Q}(\overline{x}_1, \ldots, \overline{x}_n))\overline{P}_i(\overline{x}_1, \ldots, \overline{x}_n)$ holds in the quotient ring. As $K[x_1, \ldots . x_n]$ is isomorphic to the subring of $K[x_1, \ldots, x_{n+1}]/(x_{n+1}Q - 1)$ generated by $\{\overline{x}_1, \ldots, \overline{x}_n\}$, this relation holds in $K[x_1, \ldots, x_n]$. This proves that $Q \in nilrad(I)$. □

Proposition 1.11 *If K is an algebraically closed field, then the maximal ideals of $K[x_1, \ldots, x_n]$ are exactly the ideals generated by $\{x_1 - k_1, \ldots, x_n - k_n\}$ where $(k_1, \ldots, k_n) \in K^n$.*

Proof: the fact that the ideal $Id(\{x_1 - k_1, \ldots, x_n - k_n\})$ is maximal is immediate. Conversely, suppose that I is a maximal ideal. Then, by the weak Nullstellensatz, there exists $k \in K^n$ such that $k \in V(I)$. Then the Nullstellensatz implies that I is included in the nil radical of the ideal $Id(\{x_1 - k_1, \ldots, x_n - k_n\})$, which happens to be $Id(\{x_1 - k_1, \ldots, x_n - k_n\})$ itself. As I is maximal, then $I = Id(\{x_1 - k_1, \ldots, x_n - k_n\})$. □

Notice that this result becomes false when K is not algebraically closed. For example, in $\mathbb{R}[x_1, x_2]$, the ideal $Id(\{x_1 - 1,\ x_2^2 + x_2 + 1\})$ is a maximal ideal. Note also that the ideal generated by $P_1(x_1), \ldots, P_n(x_n)$, where $P_1, \ldots, P_n$ are irreducible univariate polynomials over K, is not always a maximal ideal. For example, in $\mathbb{R}[x_1, x_2]$, we have $Id(\{x_1^2 + 3, 3x_2^2 + 4\}) \subset Id(\{2x_1 - 3x_2, 3x_2^2 + 4\})$, and this is a proper inclusion.

Proposition 1.12 *When K is algebraically closed, the maximum spectrum of $A = K[x_1, \ldots, x_n]$ and K^n with its Zariski topology are homeomorphic spaces.*

Proof: The canonical homeomorphism Ψ from K^n to $X_{max}(A)$ is given by $\Psi(k_1, \ldots, k_n) = (\{x_1 - k_1, \ldots, x_n - k_n\})$. By proposition 1.11, this is a bijection. A closed set of K^n for the Zariski topology

is a variety V defined by a certain ideal I. A polynomial $P \in I$ vanishes for any $(k_1, \ldots, k_n) \in V$, therefore $P \in nilrad(Id(\{x_1 - k_1, \ldots, x_n - k_n\}))$ which is $Id(\{x_1 - k_1, \ldots, x_n - k_n\})$ itself. So, $\Psi(V)$ is included in the set of maximal ideals containing I. Now, if $Id(\{x_1 - k_1, \ldots, x_n - k_n\})$ is a maximal ideal containing I, then any polynomial of I vanishes on $(k_1, \ldots, k_n)$. So, $\Psi(V)$ is the set of maximal ideals containing I and is therefore a closed set of $X_{max}(A)$.

Conversely, consider a closed set $Max(I)$ of $X_{max}(A)$, namely the set of maximal ideals containing a given ideal I. From above, we can conclude that $\Psi^{-1}(Max(I)) = V(I)$, and therefore it is a closed set for the Zariski topology. □

2. Rewriting systems in polynomial rings.

We saw in the previous section that an ideal can be represented by a finite basis of polynomials. Now we want to know whether this representation is *computationally effective*, i.e. whether there exists an algorithm which, given a polynomial P and a finite set of polynomials defining an ideal I, decides if P belongs to I.

We call a *power product* an element of the free commutative monoid generated by $x_1, \ldots, x_n$. The power product corresponding to a monomial is just the monomial divided by its coefficient.

Definition 2.1 A rule r in A is a pair for which the first component is a monomial m of A and for which the second component is a polynomial P of A such that none of its monomials has the same power product as m. We will write $r : m \rightarrow P$

Definition 2.2 $Q_1 \in A$ is *reduced in one step to* $Q_2 \in A$ *by the rule* $r : m \rightarrow P$ (we will write this relation as $Q_1 \longrightarrow_r^1 Q_2$) iff a monomial of Q_1 is divided by the left-hand side of r (i.e. there exists a monomial $m' \in A$ and a polynomial $Q \in A$ such that $P = m\, m' + Q$ and no power product of S is equal to the power product of $m\, m'$) and $Q_2 = m'\, P + Q$.

Q_1 is *reduced in one step to* Q_2 *w.r.t. the set of rules* $\mathcal{R}$ (we write $Q_1 \longrightarrow_{\mathcal{R}}^1 Q_2$) if it is reduced in one step by some rule r in $\mathcal{R}$.

We will call $\longrightarrow_{\mathcal{R}}^+$ the *transitive closure* of $\longrightarrow_{\mathcal{R}}^1$ and $\longrightarrow_{\mathcal{R}}^*$ its *reflexive, transitive closure.*

We remark that if Q_1 is reduced to Q_2 by the rule $r : m \rightarrow P$, then Q_1 is reduced to Q_2 by any rule of the form $k\, m \rightarrow k\, P$ for any $k \in K$. Therefore, we can choose rules with power products as left-hand sides. Notice that this is not true for a polynomial ring with coefficients in a ring instead of a field, for example, in $\mathbb{Z}[x_1, x_2]$, the polynomial x_1^2 can be reduced by the rule $x_1 \rightarrow 2$ but not by the rule $2\, x_1 \rightarrow 4$.

Proposition 2.1 *If* $\mathcal{R}$ *is a finite set of rules* $r_i : m_i \rightarrow P_i$ *and if* $Q_1 \longrightarrow_{\mathcal{R}}^* Q_2$, *then* $Q_1 - Q_2$ *belongs to the ideal generated by the polynomials* $m_i - P_i$. *We call* $Id(\mathcal{R})$ *this ideal and* $\mathcal{B}(\mathcal{R})$ *the set of polynomials* $m_i - P_i$.

So, to prove that a given polynomial P belongs to an ideal I, we can try to form a set of rules $\mathcal{R}$ from a finite basis of I and try to reduce P to 0 w.r.t. this set of rules. But there are several problems with this scheme. First, we want to restrict the space of sets of rules to the ones for which the reduction always terminates. This is the subject of the next section.

2.1. Termination of the reduction and admissible orders.

In order to use rules, it is often very important to see that their applications terminate in finitely many steps.

Definition 2.3 A relation $\succ$ on a set is *Noetherian* if there is no infinite descending chain $T_1 \succ T_2 \ldots \succ T_i \succ \ldots$.

Obviously, there are some systems of rules $\mathcal{R}$ for which the relation $\longrightarrow^1_{\mathcal{R}}$ is not Noetherian. For example, in $\mathbf{Q}[x_1, x_2]$, the relation $\longrightarrow^1_R$ for $\mathcal{R} = \{x_1 \rightarrow x_2 + 1, x_2 \rightarrow x_1^2 + 1\}$ is not Noetherian. The chain $x_1 \longrightarrow^1_{\mathcal{R}} x_2 + 1 \longrightarrow^1_{\mathcal{R}} x_1^2 + 2 \longrightarrow^1_{\mathcal{R}} x_1\, x_2 + x_1 + 2 \ldots$ is infinite.

Definition 2.4 Given a set of rules $\mathcal{R}$, we define the relation $\gg_{\mathcal{R}}$ on the set of power products of the variables $x_1, \ldots, x_n$ by the following conditions:

1. If T_1 is a power product appearing on the left-hand side of a rule r and if T_2 is a power product appearing on the right-hand side of r, then $T_1 \gg_{\mathcal{R}} T_2$.
2. (Transitive closure). If T_1, T_2, T_3 are power products such that $T_1 \gg_{\mathcal{R}} T_2$ and $T_2 \gg_{\mathcal{R}} T_3$, then $T_1 \gg_{\mathcal{R}} T_3$.
3. (Multiplicative closure). If T_1, T_2 are power products such that $T_1 \gg_{\mathcal{R}} T_2$ and if T_3 is any power product, then $T_1\, T_3 \gg_{\mathcal{R}} T_2\, T_3$.

We call *strict order* an antireflexive, antisymmetric, transitive relation.

Proposition 2.2 *If $\longrightarrow^1_{\mathcal{R}}$ is Noetherian, then $\gg_{\mathcal{R}}$ is a partial strict order .*

Lemma 2.1 *If $T_1 \gg_{\mathcal{R}} T_2$, then there exists an element $k \in K$ and a polynomial $P \in A$ such that $T_1 \longrightarrow^+_{\mathcal{R}} k\, T_2 + P$ and T_2 is not a power product of P.*

Proof of the lemma: We follow the inductive definition of $\gg_{\mathcal{R}}$. (1) If there exists a rule $k\, T_1 \rightarrow k'\, T_2 + P$, the result is immediate.

(2) If there exists a power product T_3 such that $T_1 \gg_{\mathcal{R}} T_3$ and $T_3 \gg_{\mathcal{R}} T_2$, our induction hypothesis is that there exist $k, l \in K$ and $P, Q \in A$ such that $T_1 \longrightarrow^+_{\mathcal{R}} k\, T_3 + P$ and $T_3 \longrightarrow^+_{\mathcal{R}} l\, T_2 + Q$ and T_3 is not a power product of P, nor T_2 a power product of Q. Moreover, we suppose that T_2 is not a power product of P (otherwise, $T_1 \longrightarrow^+_{\mathcal{R}} k\, T_3 + P$ is the reduction we are looking for). Then, $T_1 \longrightarrow^+_{\mathcal{R}} k\, l\, T_2 + k\, Q + P$ and T_2 is not a power product of $k\, Q + P$.

(3) If there exist power products T_3, T_4, T_5 such that $T_1 = T_3\, T_4$, $T_2 = T_3\, T_5$ and $T_4 \gg_{\mathcal{R}} T_5$, our induction hypothesis is that there exist $k \in K$ and $P \in A$ such that $T_4 \longrightarrow^+_{\mathcal{R}} k\, T_5 + P$ and T_5 is not a power product of P. Then, $T_1 \longrightarrow^+_{\mathcal{R}} k\, T_2 + T_3\, P$ and T_2 is not a power product of $T_3\, P$. □

Proof of 2.2: If $\gg_{\mathcal{R}}$ is a transitive relation, in order not to be a strict order relation, it has to be either not antireflexive or not antisymmetric. By transitivity, if it is not antisymmetric, it is not antireflexive. So, let's suppose that there exists a power product T such that $T \gg_{\mathcal{R}} T$. By

the lemma above, there exist $k \in K$ and $P \in R$ such that $T \longrightarrow_{\mathcal{R}}^{+} k\,T + P$ and T is not a power product of P. Then, the chain:

$$T \longrightarrow_{\mathcal{R}}^{+} k\,T + P \longrightarrow_{\mathcal{R}}^{+} k^2\,T + k\,P + P \longrightarrow_{\mathcal{R}}^{+} \ldots \longrightarrow_{\mathcal{R}}^{+} k^n\,T + (k^{n-1} + \ldots + 1)\,P \longrightarrow_{\mathcal{R}}^{+} \cdots$$

is infinite. □

Definition 2.5 A *total admissible order* is a total order $\succ$ on the set of power products such that:

1. Every power product T satisfies $T \succ 1$, and
2. For all power products T_1, T_2, T_3, the condition $T_1 \succ T_2$ implies $T_1\,T_3 \succ T_2\,T_3$.

Condition 2 means that $\succ$ is multiplicatively closed. In the following definitions, among all the admissible orders which are equivalent by permutation of the variables, we choose the ones which verify $x_1 \succ x_2 \succ \ldots \succ x_{n-1} \succ x_n$ Here are some very important admissible orders:

Definition 2.6 $x_1^{i_1} \ldots x_n^{i_n} \succ_L x_1^{j_1} \ldots x_n^{j_n}$ iff there exists $p\ (1 \leq p \leq n)$ such that $i_1 = j_1$, $\ldots$, $i_{p-1} = j_{p-1}$ and $i_p > j_p$.

Definition 2.7 $x_1^{i_1} \ldots x_n^{i_n} \succ_{TL} x_1^{j_1} \ldots x_n^{j_n}$ iff $i_1 + \ldots + i_n > j_1 + \ldots + j_n$ or $i_1 + \ldots + i_n = j_1 + \ldots + j_n$ and $x_1^{i_1} \ldots x_n^{i_n} \succ_L x_1^{j_1} \ldots x_n^{j_n}$ for the lexicographic order.

Definition 2.8 $x_1^{i_1} \ldots x_n^{i_n} \succ_{TR} x_1^{j_1} \ldots x_n^{j_n}$ iff $i_1 + \ldots + i_n > j_1 + \ldots + j_n$ or $i_1 + \ldots + i_n = j_1 + \ldots + j_n$ and there exists $p\ (1 \leq p \leq n)$ such that $i_n = j_n$, $\ldots$, $i_{p+1} = j_{p+1}$ and $i_p < j_p$.

Note that $\succ_L$ is the lexicographic order, $\succ_{TL}$ is the total degree ordering refined by the lexicographic order, and $\succ_{TR}$ is the total degree ordering refined by the reverse lexicographic order.

If $\succ$ is an admissible order and P a polynomial, we will call $In_{\succ}(P)$ the *leading monomial* of P, $InPp_{\succ}(P)$ the *leading power product* of P, and $InC_{\succ}(P)$ the *leading coefficient* of P. When we consider only one admissible ordering, the index $\succ$ will be omitted.

The proof that we present here of the following result may be found, for example, in Winkler's Ph.D. thesis, [39].

Proposition 2.3 *An admissible order is a well-ordering (i.e., there is no infinite descending chain of power products).*

Proof: this is an immediate consequence of Dickson's Lemma. The result comes from the fact that the existence of two power products T_1 and T_2 such that $T_1 \succ T_1T_2$ with $T_2 \neq 1$ is in contradiction with the definition of an admissible order.□

Given a set B of polynomials and a total admissible order $\succ$, we can find a corresponding set of rules by taking, for each polynomial P of B, the monomial m of P with the leading power product w.r.t. $\succ$ as left-hand side, and $P - m$ as right-hand side. We will designate by $\mathcal{R}(B, \sigma)$ the set of rules obtained from the set of polynomials B by using the total admissible order $\succ$.

Proposition 2.4 $\longrightarrow^1_{\mathcal{R}(S,\succ)}$ *is a Noetherian relation for any set of polynomials* S *and any admissible total order* $\succ$.

Proof: See Winkler, [39], 3.1. □

Proposition 2.5 *Let* $\mathcal{R}$ *be a set of rules. If* $\longrightarrow^1_{\mathcal{R}}$ *is Noetherian, then there exists an admissible order* $\succ$ *such that* $\mathcal{R} = \mathcal{R}(\mathcal{P}(\mathcal{R}), \succ)$.

Proof: We define the relation $\succ$ on power products as follows: $T_1 \succ T_2$ iff either $T_1 \gg_R T_2$ (see definition 2..4) or T_1 and T_2 are not comparable by $\gg_R$ and $T_1 \succ_L T_2$. Here $\succ_L$ is the lexicographic ordering.

If $\gg_R$ is a partial strict order, then $\succ$ is a total admissible order and $\mathcal{R} = \mathcal{R}(\mathcal{P}(\mathcal{R}), \succ)$. □

So, by combining propositions 2.2 and 2.4, we see that the sets of rules for which the reduction relation is Noetherian are exactly those generated from sets of polynomials by using a total admissible ordering. Therefore, in the next sections, we will only consider systems of rules generated by a system of polynomials and an admissible order.

2.2. Gröbner bases and Buchberger's algorithm.

The systems of rules we consider differ from usual term rewriting systems for essentially two reasons: the first one is that all the terms are ground terms, i.e., there is no rewriting rule containing variables. Note that $x_0, \ldots, x_n$ are indeterminates, sometimes called variables as algebraic objects, but they are *constants* of the rewriting systems. The second reason is that the reduction does not only consist in a replacement of one term by another but also in a simplification of the resulting polynomial according to the algebraic laws of the ring A. Nevertheless, the two problems are very similar, and the same terminology applies to both. For an introduction to rewriting systems in general, we suggest to the review by Huet and Oppen, [19], or the article by Knuth and Bendix, [22] which describes their completion algorithm. To understand more deeply the relations between reduction systems in ideal theory and rewriting systems in general, see Winkler's Ph.D thesis, [39]. For a historical review, see Buchberger, [9].

In the previous section, we restricted the universe of the sets of rules to those for which the reduction is Noetherian. We ended up with sets of rules inferred from a set of polynomials and from an admissible order. We can now introduce the concept of normal form:

Definition 2.9 A polynomial P is *in normal form w.r.t. a system of rules* $\mathcal{R}$ iff there exists no polynomial Q such that $P \longrightarrow^1_{\mathcal{R}} Q$.

A polynomial P has at least one normal form w.r.t. a Noetherian reduction. We are interested in obtaining a unique normal form. Why ?

We can call $\longleftrightarrow^*$ the *reflexive, symmetric, transitive closure* of $\longrightarrow^1$ (we omit $\mathcal{R}$ when we consider only one set of rules). $\longleftrightarrow^*$ is an equivalence relation in A. For a rewriting system in general, this is an equivalence relation on the set of terms.

Definition 2.10 The *unique normal form property* holds for a relation $\longrightarrow^1$ iff $\longrightarrow^1$ is such that every polynomial has a unique normal form w.r.t. it.

The following proposition is immediate:

Proposition 2.6 *If $\longrightarrow^1$ verifies the unique normal form property, then, for all $P, Q \in A$, we have $P \longleftrightarrow^* Q$ iff P and Q have the same normal form.*

In this case, $\longleftrightarrow^*$ is computable. This becomes essential if we add the following remark:

Proposition 2.7 *If $\longrightarrow^1$ is the reduction relation associated with a basis of an ideal I and an admissible order $\succ$, then, for all P, Q in A, we have $P \longleftrightarrow^* Q$ iff $P - Q \in I$.*

For a reduction relation with the unique normal form property, let's call $nf(P)$ the *normal form corresponding to* P. Then we see that $nf(A)$ is isomorphic to A/I. Therefore, this gives a way of computing the natural homomorphism from A onto A/I. In particular, this gives a solution to the membership problem: a polynomial P belongs to I iff $nf(P) = 0$. Therefore, systems of rules for which the unique normal form property holds are extremely useful.

Now, given an admissible order $\succ$ and a basis B of an ideal I, how can we obtain another basis B' of I such that the unique normal form property holds for the reduction associated with B' and $\succ$? This is the purpose of Buchberger's algorithm, which appeared in Buchberger's thesis, [6], in 1965 and has been extensively studied since. The basis B' is called a Gröbner basis or standard basis of I. (The second denomination refers to the work of Hironaka, [18].)

Buchberger's algorithm was designed by using techniques which are very similar to the work of Knuth and Bendix, [22]. The details of this approach can be found in Winkler's thesis and in two early articles by Buchberger, [5] and [4]. More recently, a more algebraic presentation of the algorithm appeared. We will follow this approach here by quoting and explaining the Fundamental Theorem that Robbiano presented in his Tutorial at Computers and Mathematics 1989, [32].

Let's recall first a basic definition:

Definition 2.11 A ring R is *graded* by a monoid E if there exists a family of additive subgroups $\{R_i\}_{i \in E}$ such that $R = \bigoplus_{i \in E} R_i$ and $R_i R_j \subseteq R_{i+j}$ for all $i, j \in E$. If R is a graded ring, an R-module M is *graded* if there exists a family of additive subgroups $\{M_i\}_{i \in E}$ such that $M = \bigoplus_{i \in E} M_i$ and $R_i M_j \subseteq M_{i+j}$ for all $i, j \in E$. The elements of $\bigcup_{i \in E} R_i$ (resp. $\bigcup_{i \in E} M_i$) are called the *homogeneous elements* of R (resp. M).

We choose $\mathbb{N}^n$ as the monoid of the indices and $A_{(i_1,\ldots,i_n)} = \{k\, x_1^{i_1} \ldots x_n^{i_n} \mid k \in K\}$ as additive subgroups of A. For the power product $T = x_1^{i_1} \ldots x_n^{i_n}$, we call $log(T)$ the vector $(i_1, \ldots, i_n)$.

Given any p-tuple of elements of N^n, $((i_{1,1}, \ldots, i_{1,n}), \ldots, (i_{p,1}, \ldots i_{p,n}))$, one can define a graded structure on A^p by choosing, as additive subgroups,

$$(A^p)_{(j_1,\ldots,j_n)} = \{(k_1\, T_1, \ldots, k_p\, T_p) \mid log(T_l) + (i_{l,1}, \ldots, i_{l,n}) = (j_1, \ldots, j_n)\ ,\ \forall l, 1 \leq l \leq p\},$$

where $k_1, \ldots, k_p$ are elements of K and $T_1, \ldots, T_p$ are power products.

Suppose that the ideal I of A has the finite basis $B = (P_1, \ldots, P_p)$. Let's consider a particular admissible order $\succ$. We can grade A^p with the p-tuple of vectors $(log(InPp(P_1)), \ldots, log(InPp(P_p)))$. Now, any member of A^p is decomposed uniquely in a sum of elements of the sets $(A^p)_{(i_1,\ldots,i_n)}$. We will call the *leading vector* of $(Q_1, \ldots, Q_p)$ the homogeneous part of $(Q_1, \ldots, Q_p)$ belonging to the set $(A^p)_{(i_1,\ldots,i_n)}$ for which $x_1^{i_1} \ldots x_n^{i_n}$ is the highest w.r.t. $\succ$. We will write it $Lv((Q_1, \ldots, Q_p))$.

Also we define the two homomorphisms λ and Λ from A^p to A by:

$$\lambda((Q_1, \ldots, Q_p)) = \sum_l Q_l \, P_l, \text{ and}$$

$$\Lambda((Q_1, \ldots, Q_p)) = \sum_l Q_l \, In(P_l).$$

Following Robbiano, we say that an homogeneous element H of A^p *extends* to $U \in A^p$ iff $Lv(U) = H$.

Now, we are ready to state the Fundamental Theorem:

Theorem 2.1 *Let $\succ$ be an admissible order, $B = (P_1, \ldots, P_p)$ a p-tuple of non-zero elements of A, and I the ideal generated by the elements in B. Then the following conditions are equivalent:*

(A) For every $P \in I \backslash \{0\}$, there exist p polynomials $Q_1, \ldots, Q_p \in A \backslash \{0\}$ such that $P = \sum_l Q_l \, P_l$ with $InPp_\succ(P) \succ InPp_\succ(Q_l) \, InPp_\succ(P_l)$ for every l, $1 \leq l \leq p$.

(B) $\{In_\succ(P_1), \ldots, In_\succ(P_p)\}$ generates $In_\succ(I)$.

(C-I) $P \in I \iff P \longrightarrow^*_{\mathcal{R}(B,\succ)} 0$.

(C-II) Every $P \in I$ has a unique normal form w.r.t. the reduction $\longrightarrow^*_{\mathcal{R}(B,\succ)}$ and this normal form is 0.

(C-III) Every $P \in A$ has a unique normal form w.r.t. $\longrightarrow^*_{\mathcal{R}(B,\succ)}$.

(D-I) Every homogeneous element of $Ker(\Lambda)$ extends to an element of $Ker(\lambda)$.

(D-II) There exists an homogeneous basis of $Ker(\Lambda)$ which extends to elements of $Ker(\lambda)$.

The proof of this important result can be found, for example, in Robbiano, [32].

Definition 2.12 If B satisfies any of these conditions, then B is a *Gröbner basis* of I.

If every polynomial P of B is in normal form for the reduction w.r.t. the set $\mathcal{R}(B \backslash \{P\}, \succ)$, then B is called a *reduced* Gröbner basis for I.

Note that condition (D-II) of theorem 2.1 implies that, if we can find a homogeneous basis of the module of syzygies of $(In(P_1), \ldots, In(P_p))$ such that, for every vector $(Q_1, \ldots, Q_p)$ of this basis, $\sum_l Q_l \, P_l \longrightarrow^*_{\mathcal{R}(B,\succ)} 0$, then B is a Gröbner basis of I w.r.t $\succ$.

We now introduce a very convenient homogeneous basis of the syzygies of $(P_1, \ldots, P_p)$. To make the definitions readable, we denote $T(i) = InPp_\succ(P_i)$ and $T(i,j) = lcm(InPp_\succ(P_i), InPp_\succ(P_j))$. We will also use the notation $T(i,j,l)$ for $lcm(InPp_\succ(P_i), InPp_\succ(P_j), InPp_\succ(P_l))$.

```
Input:  a list of polynomials (P1,...,Pp), basis of an ideal I.
Output:  a Gröbner basis B of I for the admissible order ≻.

  B:= (P1,...,Pp);
  label:= p;
  Pairs:= {(i,j) | 1 ≤ i < j ≤ p};
  while (Pairs ≠ ∅) do
    begin
      choose (i,j) from Pairs; Pairs:= Pairs\{(i,j)};
      P:= nf_{R(B,≻)}(S(i,j));
      if (P ≠ 0) then
        begin
          label:= label + 1;
          P_label:= P;
          B:= (B,P_label);
          Pairs:= Pairs ∪ {(i,label) | 1 ≤ i < label};
        end;
    end;
```

Figure 1: Buchberger's algorithm

Proposition 2.8 *Let $(e_1, \ldots, e_p)$ be the canonical basis of A^p. Then the vectors*

$$s(i,j) \;=\; InC_{\succ}(P_j)\,\frac{T(i,j)}{T(i)}\,e_i \;-\; InC_{\succ}(P_i)\,\frac{T(i,j)}{T(j)}\,e_j,$$

where $1 \leq i < j \leq p$, form an homogeneous basis of the module of syzygies of $(In_{\succ}(P_1), \ldots, In_{\succ}(P_p))$.

Proof: See Robbiano, [32]. □

Definition 2.13 The polynomial

$$S(i,j) \;=\; \lambda(s(i,j)) \;=\; InC_{\succ}(P_j)\,\frac{T(i,j)}{T(i)}\,P_i \;-\; InC_{\succ}(P_i)\,\frac{T(i,j)}{T(j)}\,P_j$$

is called the *S-polynomial* of P_i and P_j.

Thus, to verify that $(P_1, \ldots, P_p)$ is a Gröbner basis, we only need to check that all the S-polynomials $S(i,j)$ can be reduced to 0 (according to condition (D-II) of the Fundamental Theorem above). We can introduce now in Figure 1 the elementary version of Buchberger's algorithm. In this program $nf_{\mathcal{R}(B,\succ)}(P)$ will designate any normal form of P w.r.t. the reduction $\longrightarrow^*_{\mathcal{R}(B,\succ)}$.

Theorem 2.2 *The algorithm of Figure 1 computes a Gröbner basis of I w.r.t. $\succ$.*

Proof: if the algorithm terminates, its output B is a Gröbner basis because it satisfies (D-II) above. As every polynomial added to B belongs to I, B is a Gröbner basis of I. Finally, let's call I_{label} the ideal generated by $In(B)$ after P_{label} has been added to B. These ideals form a strictly increasing sequence of ideals. As A is a Noetherian ring, this sequence has to be finite and the algorithm terminates. □

2.3. Improvements in the algorithm.

The two improvements that we are going to describe are of different natures. The first one, called Criterion 2 by Buchberger, [8], avoids certain reductions by forecasting that certain very particular S-polynomials can be reduced to 0. The second one is a recent refinement by Gebauer and Möller, [16], of Criterion 1 by Buchberger. They show that $Ker(\Lambda)$ is in fact generated by a subset of the $s(i,j)$ and, therefore, that only a subset of the S-polynomials $S(i,j)$ need to be reduced.

The first improvement concerns pairs of polynomials with co-prime leading power products and is based on the next proposition.

Proposition 2.9 *If* $T(i,j) = T(i)T(j)$, *then* $s(i,j)$ *extends to an element of* $Ker(\lambda)$.

Proof: If $T(i,j) = T(i)T(j)$, then $R = InC_{\succ}(P_j)\,P_j\,e_i - InC_{\succ}(P_i)\,P_i\,e_j$ extends $s(i,j)$. Obviously, $R \in Ker(\lambda)$. □

So, before an S-polynomial is reduced, the algorithm should check if the two polynomials in the pair verify the condition of Proposition 2.9. This is the second criterion of Buchberger.

The second improvement is based on the consideration of redundant elements of the basis of syzygies. Gebauer and Möller, [16], use the resolution of Taylor, [36], to describe their enhancement of the algorithm. Taylor studied the resolution of monomial ideals and found homogeneous bases for the different modules in this resolution. These are called Taylor bases. We have already seen that the module of syzygies of $(In(P_1),\ldots,In(P_p))$ is generated by $(s(i,j))_{1\le i<j\le p}$. To avoid tiresome multiplications by coefficients, we suppose that the polynomials $P_1,\ldots,P_p$ are normalized, i.e. that their leading coefficients are equal to 1. This does not imply any restriction because $(P_1,\ldots,P_p)$ and $(P_1/InC(P_1),\ldots,P_p/InC(P_p))$ generate the same ideal. Then we have

$$s(i,j) = T(i,j)/T(i)e_i - T(i,j)/T(j)e_j.$$

Now we can look at the submodule of $A^{p(p-1)/2}$ of the syzygies of $(s(i,j))_{1\le i<j\le p}$. If we label the canonical unit vectors $(e_{ij})_{1\le i<j\le p}$, this submodule can be expressed as:

$$S^{(2)} = \{\sum_{1\le i<j\le p} P_{ij}\,e_{ij} \in A^{p(p-1)/2} \mid \sum_{1\le i<j\le p} P_{ij}\,s(i,j) = 0\}.$$

This has the Taylor basis $(s(i,j,l))_{1\le i<j<l\le p}$ with

$$s(i,j,l) = \frac{T(i,j,l)}{T(i,j)}\,e_{ij} - \frac{T(i,j,l)}{T(i,l)}\,e_{il} + \frac{T(i,j,l)}{T(j,l)}\,e_{jl}.$$

If we grade $A^{p(p-1)/2}$ by using the vectors $(log(T(i,j)))_{1\le i<j\le p}$, then $s(i,j,l)$ is homogeneous and belongs to $A^{p(p-1)/2}_{log(T(i,j,l))}$

Now, what happens if $T(i,j,l)$ is equal to $T(i,j)$, $T(i,l)$ or $T(j,l)$? Suppose, for example, that $T(i,j,l) = T(i,j)$. Then

$$s(i,j,l) = e_{ij} - \frac{T(i,j,l)}{T(i,l)} e_{il} + \frac{T(i,j,l)}{T(j,l)} e_{jl}$$

is a syzygy of $(s(i,j))_{1 \leq i < j \leq p}$ and, therefore,

$$s(i,j) - \frac{T(i,j,l)}{T(i,l)} s(i,l) + \frac{T(i,j,l)}{T(j,l)} s(j,l) = 0.$$

So, $s(i,j)$ can be expressed in terms of other members of the basis, and it can be removed from the homogeneous basis. This means that it is unnecessary to consider the pair of polynomials (P_i, P_j) during the execution of the algorithm.

Naturally, after eliminating $s(i,j)$ from the basis of first syzygies, we can take another look at $S^{(2)}$ to see if there exists another syzygy $s(i',j')$ which can be expressed as a combination of syzygies of lower or equal degree and, if there exists any, we can remove it from the basis and go on iteratively. But, we must proceed in order: if $T(i,j,l) = T(i.j) = T(i,l)$, for example, then

$$s(i,j) - s(i,l) + \frac{T(i,j,l)}{T(j,l)} s(j,l) = 0,$$

but we do not want to remove both $s(i,j)$ and $s(i,l)$ from the basis of syzygies because of this relation.

The details on the way these considerations are actually used to update the set of pairs when a polynomial is added to the basis may be found in Gebauer and Möller, [16].

These two improvements of the basic algorithm are essential to the efficiency of the computation: with them, only a small fraction of all the pairs in the Cartesian product are considered. We refer the reader to the last section of this work. We present there some examples of computations. For each example, we give the number #P of polynomials in the final basis (including redundant polynomials) and the number #R of pairs for which the S-polynomials has been reduced. The comparison between #R and $\#P \times (\#P - 1)/2$ gives a good idea of the gain that the use of these two criteria allows us to make.

3. Applications of Gröbner bases in commutative ring theory.

The constructions described in this section are borrowed from Gianni, Trager and Zacharias, [29]. The constructions studying subalgebras of the polynomial ring are due to Shannon and Sweedler, [34]. Many of these constructions are consequences of the well-known following theorem:

Theorem 3.1 *If B is the reduced Gröbner basis of an ideal I in $K[u_1,\ldots,u_n,x_1,\ldots,x_n]$ with respect to an admissible order $\succ$ in which the relations $x_p \succ u_1^{i_1}\ldots u_n^{i_n}$ hold for all p in $\{1,\ldots,n\}$ and all n-tuples $(i_1,\ldots,i_n)$ in $\mathbb{N}^n$, then $B \cap K[u_1,\ldots,u_n]$ is the reduced Gröbner basis of $I \cap K[u_1,\ldots,u_n]$ for the induced order.*

Proof: Let's notice first that, if $\succ$ has the property defined above polynomials of $K[u_1,\ldots,u_n]$ can only be reduced by using polynomials of $K[u_1,\ldots,u_n]$. So, polynomials of $I \cap K[u_1,\ldots,u_n]$

which are reduced to 0 by members of G are reduced to 0 by members of $B \cap K[u_1,\ldots,u_n]$. This defines $B \cap K[u_1,\ldots,u_n]$ as a Gröbner basis of $I \cap K[u_1,\ldots,u_n]$. The fact that it is a reduced Gröbner basis follows directly from the fact that B is reduced.□

In their formal statement of this property, Gianni, Trager and Zacharias add that B is a Gröbner basis (possibly not reduced) of $(K[u_1,\ldots,u_n])[x_1,\ldots,x_m]$ for the order induced by $\succ$ on this structure.

3.1. Intersection of two ideals.

Let I and J be two ideals of $K[x_1,\ldots,x_n]$ and let u be a new indeterminate (i.e. a member of an extension of K s.t. $x_1,\ldots,x_n,u$ are algebraically independent over K). $(u\ I)$ is the ideal of $K[x_1,\ldots,x_n,u]$ generated by $\{u\ P \mid P \in I\}$.

Proposition 3.1 $(u\ I,(1-u)\ J) \cap K[x_1,\ldots,x_n]$ *is equal to* $I \cap J$.

So, using the contraction mechanism explained in the introduction of this section, we can obtain a Gröbner basis of $I \cap J$.

Proof: $I \cap J \subseteq (u\ I,\ (1-u)\ J) \cap K[x_1,\ldots,x_n]$. Suppose $P \in I \cap J$. Then $P = u\ P + (1-u)\ P$.

$(u\ I,\ (1-u)\ J) \cap K[x_1,\ldots,x_n] \subseteq I \cap J$. Suppose $P = u\ P_I + (1-u)\ P_J$ and that u is not a variable in P. Then, $P = P_I = P_J$ and, therefore, $P \in I \cap J$ □

For Example, $I = (x^2+y^2-z^2)$ and $J = (x-z, y-2z)$ are two ideals of $\mathbb{C}[x,y,z]$. We can compute a Gröbner basis for an ideal whose corresponding variety is the union in the complex projective plane of the conic defined by I and of the point $(1,2,1)$ defined by J.

The reduced Gröbner basis of $(\ u(x^2+y^2-z^2), (1-u)(x-z), (1-u)(y-2z))$ w.r.t. the lexicographic order (in which $u \succ_L x \succ_L y \succ_L z$) is:

$$\{u\,x - u\,z - x + z,\ -u\,y + 2\,u\,z + y - 2\,z, 4\,u\,z^2 + x^2 + y^2 - 5\,z^2,$$
$$-x^2\,y + 2\,x^2\,z - y^3 + 2\,y^2\,z + y\,z^2 - 2\,z^3,\ x^3 - x^2\,z + x\,y^2 - x\,z^2 - y^2\,z + z^3\}$$

So the last two polynomials give a basis of $I \cap J$. It may not be surprising that these two last polynomials are $-(y-2\,z)\,(x^2+y^2-z^2)$ and $(x-z)\,(x^2+y^2-z^2)$.

If we choose now $J = (x-z,\ y)$, which determines the point $(\lambda,0,\lambda)$ lying on the conic, the Gröbner basis we find for $I \cap J$ is $\{x^2+y^2-z^2\}$, as expected.

3.2. The ideal quotient of two ideals $I : J$.

We consider here the case where I and J are two ideals of $K[x_1,\ldots,x_n]$. In fact, the construction is still valid when I and J are ideals in the polynomial ring over certain rings (namely Noetherian commutative rings with identity in which linear equations are solvable) if the generators of J are not zero divisors. See [29] for details.

Suppose J is generated by $\{P_1,\ldots,P_p\}$. Then:

$$I : J = \{Q \mid Q\ J \subseteq I\} = \bigcap_{i=1}^{p} \{Q \mid Q\ P_i \in I\} = \bigcap_{i=1}^{p} I : (P_i)$$

In the last subsection, we describe how to find bases of intersections of ideals. We discuss now how to compute a basis of $I : (P)$. Note that

$$I : (P) = \{Q \mid Q\ P \in I\} = \{Q \mid Q\ P \in I \cap (P)\} = (I \cap (P)) : (P).$$

We know how to compute a Gröbner basis $\{Q_1, \ldots, Q_l\}$ of $I \cap (P)$ w.r.t. $\succ$. Then $\{Q_1/P, \ldots, Q_l/P\}$ generates $I : (P)$, because the Q_i/P are indeed polynomials, as $Q_i \in (P)$.

It is not immediate that $\{Q_1/P, \ldots, Q_l/P\}$ is a Gröbner basis w.r.t $\succ$. This stems from the fact that the leading monomial of Q_i w.r.t. $\succ$ is equal to the leading monomial of Q_i/P multiplied by the leading monomial of P. Then we can reduce the S-polynomials $Spol(Q_i/P, Q_j/P)$ w.r.t. $\{Q_1/P, \ldots, Q_l/P\}$ by following exactly the reduction of the S-polynomials $Spol(Q_i, Q_j)$ w.r.t. the basis $\{Q_1, \ldots, Q_l\}$. Therefore, $Spol(Q_i/P, Q_j/P)$ is reduced to 0 for all pairs (i,j), and this is one of the characterizations of a Gröbner basis.

Take for example $I = ((x-z)(x^2+y^2-z^2), (y-2z)(x^2+y^2-z^2))$ and $J = (x-z, y-2z)$. Then $I \cap (x-z) = ((x-z)(x^2+y^2-z^2))$ and $I \cap (y-2z) = ((x-2z)(x^2+y^2-z^2))$. Therefore, $I : J = I : (x-z) = I : (y-2z) = (x^2+y^2-z^2)$.

3.3. The ideal of polynomial relations among $P_1, \ldots, P_p$.

Gianni et al., [29], give a succinct description of this construction and of a more general one which allows us to compute a basis for the kernel of a given ring-homomorphism from $R[y_1, \ldots, y_p]$ to $R[x_1, \ldots, x_n]/I$ which is invariant on R . The paper by Shannon and Sweedler, [34], is more detailed and describes a subalgebra membership test as well. We will consider only the case when R is in fact a field K.

The idea is to consider the ideal $(y_1 - P_1, \ldots, y_p - P_p)$ in $K[x_1, \ldots, x_n, y_1, \ldots, y_p]$ and to find the intersection of this ideal with $K[y_1, \ldots, y_p]$. This intersection describes the ideal of the relations between $P_1, \ldots, P_p$. The correctness of this construction depends on the following result:

Proposition 3.2 *Let R be a ring containing a ring R' and let $\{y_1, \ldots, y_p\}$ be a set of algebraically independent variables over R'. Let ϕ be a ring-homomorphism from $R'[y_1, \ldots, y_p]$ to R invariant on R' and such that $\phi(y_i) = r_i$. Then the kernel of ϕ is equal to $Id(\{y_1 - r_1, \ldots, y_p - r_p\}) \cap R'[y_1, \ldots, y_p]$ in which $Id(\{y_1 - r_1, \ldots, y_p - r_p\})$ is the ideal of $R[y_1, \ldots, y_p]$ generated by $y_1 - r_1, \ldots, y_p - r_p$.*

Proof: suppose that $P \in Id(\{y_1 - r_1, \ldots, y_p - r_p\}) \cap R'[y_1, \ldots, y_p]$. Then $\phi(P(y_1, \ldots, y_p)) = P(r_1, \ldots, r_p) = 0$. Conversely, if $P \in Ker\ \phi$, then in $R[y_1, \ldots, y_p]/Id(\{y_1 - r_1, \ldots, y_p - r_p\})$, we have $P(\overline{y_1}, \ldots, \overline{y_p}) = P(r_1, \ldots, r_p) = 0$ and, therefore, P belongs to the intersection of the ideal with $R'[y_1, \ldots, y_p]$. □

Thus we can obtain the ideal of the relations among the polynomials $P_1, \ldots, P_p$ by using once more the "contraction" algorithm to compute $Id(\{y_1 - P_1, \ldots, y_p - P_p\}) \cap K[y_1, \ldots, y_p]$.

If the ring R is equal to $K[x_1,\ldots,x_n]/I$ for a certain ideal I instead of $K[x_1,\ldots,x_n]$ itself, we first use the fact that

$$(K[x_1,\ldots,x_n]/I)[y_1,\ldots,y_p] = K[x_1,\ldots,x_n,y_1,\ldots,y_p]/I^e,$$

where I^e is the extension of I to an ideal of $R[x_1,\ldots,x_n,y_1,\ldots,y_p]$. We then remember that, for any ring R, there is a one-to-one correspondence between the ideals of R containing the ideal I and the ideals of R/I. Therefore we find the ideal of polynomial relations between $P_1,\ldots,P_p$ by computing a Gröbner basis of $Id(I^e \cup \{y_1 - P_1,\ldots,y_p - P_p\}) \cap K[y_1,\ldots,y_p]$.

4. Dimension of ideals and computation of the Hilbert polynomial.

The dimension of a homogeneous ideal I is an important characterization because it gives the first description of the projective variety associated with I. For example, in $\mathbb{C}[x,y,z]$, the only ideal of dimension -1 is $\mathbb{C}[x,y,z]$ itself and the projective variety associated with it is empty. Ideals of dimension 0 and 1 are respectively associated with finite set of points and curves of the projective plane. Once a Gröbner basis for I is found, there exists a simple algorithm which computes the dimension of I.

4.1. Basic definitions and Macaulay's result.

Recall that a homogeneous module M over a homogeneous ring R, where $R = \bigoplus_{i\in E} R_i$, E being a monoid, is such that there exists additive subgroups M_i, $i \in E$ such that $\bigoplus_{i\in E} M_i = M$ and, for all $i,j \in E$, $R_i M_j \subseteq M_{i+j}$.

Definition 4.1 An ideal I of R (resp. a submodule M' of M) is *homogeneous* iff there exists a family of aditive subgroups $\{I_i\}_{i\in E}$ of R (resp $\{M'_i\}_{i\in E}$ of M) such that $I = \bigoplus_{i\in E} I_i$ (resp. $M' = \bigoplus_{i\in E} M'_i$) and, for every $i \in E$, $I_i \subseteq R_i$ (resp. $M'_i \subseteq M_i$).

Proposition 4.1 *$R/I = \bigoplus_{i\in E} R_i/I_i$ (resp. $M/M' = \bigoplus_{i\in E} M_i/M'_i$) is a graded ring (resp a R-graded module)*

Definition 4.2 A homomorphism $h : M \longrightarrow M'$ between two homogeneous R-modules is *homogeneous* (of degree 0) iff, for all $i \in E$, $h(M_i) \subseteq M'_i$

Before we take the particular example of the polynomial ring, we need another set of definitions, from homological algebra. They can be found in N. Jacobson, [21], chapter 6.

Definition 4.3 A *complex* C for a ring R is a set $\{M_i\}$ of R-modules indexed by $\mathbb{Z}$ together with an indexed set $\{h_i\}$ of R-homomorphisms $h_i : M_i \longrightarrow M_{i-1}$ such that $Im(h_i) \subseteq Ker\,(h_{i-1})$ (i.e. $h_{i-1}\, h_i = 0$) for all i. A *positive* complex is such that $M_i = 0$ for $i < 0$.

Definition 4.4 Let M be an R-module. A *complex over* M is a positive complex C together with a homomorphism $\epsilon : M_0 \to M$, called an augmentation, such that $\epsilon\, h_1 = 0$. The complex C, ϵ over M is called a *resolution* if the sequence:

$$\ldots \to M_n \xrightarrow{h_n} M_{n-1} \to \ldots \to M_1 \xrightarrow{h_1} M_0 \xrightarrow{\epsilon} M \to 0$$

is exact (i.e. if $Im\, h_i = Ker\, h_{i-1}$ for all $i > 1$, $Im\, h_1 = Ker\, \epsilon$ and $Im\, \epsilon = M$).

Recall that an R-module M is free if it is isomorphic to a direct sum of R (i.e. $M = \bigoplus_{E'} R$ for a certain index set E').

Definition 4.5 A resolution is a *free resolution* if every module M_i is free .

Definition 4.6 Suppose that all the modules M_i and the module M in the resolution C, ϵ are graded . Then, C, ϵ is a *homogeneous* free resolution if ϵ is homogeneous and if h_i is homogeneous for every i.

Now that we have all the definitions that we need, we turn to the particular case of the polynomial ring. We put on the polynomial ring $R = K[x_1, \ldots, x_n]$ (over the field K) the natural graduation induced by the total degree function: $R = \bigoplus R_i$ where R_i is the set of homogeneous polynomials of degree i (completed by the zero polynomial). In this case, each R_i, considered as a vector space over K, has the finite dimension

$$dim\, R_i = \binom{i+n-1}{n-1}$$

A homogeneous ideal I is given by a finite basis of homogeneous polynomials. Each submodule R_i/I_i has a finite dimension over K.

Definition 4.7 The *Hilbert function* $H : \mathbb{N} \longrightarrow \mathbb{N}$ is defined by:

$$H(i) = dim(R_i/I_i)$$

It is a well known result of Hilbert that this function is equal to a polynomial HP with rational coefficients for i sufficiently large. The details of the proof can be found in English, for example, in Mora and Möller, [26]. They list three different forms of this polynomial. The one originally described by Hilbert is:

$$HP(i) = h_0 \binom{i}{d} + \cdots + h_l \binom{i}{d-l} + \cdots + h_{d-1} \binom{i}{1} + h_d$$

Definition 4.8 The degree d of HP is the *Krull dimension* of the ideal.

For example, consider the complex projective plane represented by $(\mathbb{C}^3 - \{(0,0,0)\})/\sigma$, where σ is the equivalence relation on $\mathbb{C}^3$ defined by $u\ \sigma\ v$ iff the two vectors u and v are linearly dependent. We want to study the intersection of the algebraic curves of equations $y = 0$ and $y\,z - x^2 + z^2 = 0$. The points on the intersection form an algebraic variety $V(I)$ where I is generated by $(y,\ y\,z - x^2 + z^2)$ or $(y,\ -x^2 + z^2)$. It is easy to see that this set of zeros consists of the two points $(k,0,k)$ and $(k,0,-k)$ in the projective plane. We determine the Hilbert polynomial of the ideal I by the following steps:

- $I_0 = \{0\}$, so $H(0) = 1$.
- $\{y\}$ is a basis of I_1, so $H(1) = 2$.
- $\{y\,x,\ y\,z,\ y\,x,\ x^2 - z^2\}$ is a basis of I_2, so $H(2) = 6 - 4 = 2$
- For $R = \mathbb{C}[x,y,z]$, $dim\ R_i = (i+2)\,(i+1)/2$. Then, $I_i = y\,R_{i-1} + (x^2 - z^2)\,R_{i-2}$ and $y\,R_{i-1} \cap (x^2 - z^2)\,R_{i-2} = y\,(x^2 - z^2)\,R_{i-3}$. So

$$H(i) = \frac{(i+2)\,(i+1)}{2} - \frac{(i+1)\,i}{2} - \frac{i\,(i-1)}{2} + \frac{(i-1)\,(i-2)}{2} = 2$$

we see then that $HP(i) = 2$ for $i > 0$ and the dimension of the ideal is 0. The ideals for which the corresponding projective variety is a finite set of points have always dimension 0.

Definition 4.9 The *regularity* of the ideal I is the smallest integer p such that the Hilbert function and the Hilbert polynomial have the same value for all $i \geq p$.

In our example, the regularity of the ideal I is 1. Now, to help us compute the Hilbert polynomial, we need a very important result which is an extension of a result by Macaulay in 1926, [24]. Originally, Macaulay considered only lexicographic orders.

Theorem 4.1 *Let I be a homogeneous ideal of $K[x_1,\ldots,x_n]$ and $\succ$ an admissible order. Then, I and $In_\succ(I)$ have the same Hilbert function.*

Proof: If $dim(I_i) = p$, then there exists a system of p independent polynomials in I_i. By Gaussian elimination of this system of polynomials written in the basis of the power products of degree i ordered in decreasing order w.r.t. $\succ$, we obtain a system of p polynomials with distinct leading power products. So $dim((In(I))_i) \geq dim(I_i)$.

Because $In(I)$ is a monomial ideal, $(In(I))_i$ as a vector space admits a basis of power products. Each power product T of this basis is equal to the leading power product of a polynomial of I_i or is a multiple of the leading power product of a polynomial P of I_j for some $j \leq i$. But, in the second case, we can multiply P by the suitable power product of degree $i - j$, namely $T/InPp(P)$, and, as $\succ$ is an admissible order, we obtain a polynomial of I_i whose leading power product is equal to T. So $(In(I))_i$ is generated by the leading monomials of the polynomials of I_i. □

4.2. Computation of the Hilbert polynomial.

In their EUROCAL83 paper, [26], Mora and Möller describe an algorithm to compute the Hilbert polynomial of a homogeneous ideal. This algorithm, due, originally, to Buchberger, [6], takes as input the set of leading monomials of a Gröbner basis of the homogeneous ideal I for any admissible order.

The idea behind the method can be described as follows. The dimension of R_i is $\binom{i+n-1}{n-1}$. Now, we want to subtract from this number the number of power products of degree i which are multiples of a power product in the basis of $In(I)$. If $i \geq j$, each power product of degree j has $\binom{i-j+n-1}{n-1}$ multiples of degree i. If we subtract from $dim\, R_i$ the number of multiples of degree i of each power product in the basis, we may subtract several times the same power product. So we must add to this new number the number of common multiples of each pair $\{T_1, T_2\}$ of power products in the basis of $In(I)$ (i.e. the number of multiples of the least common multiple of T_1 and T_2). But once again, we may add several times the same power product, so we need to consider triples of members of the basis and their common multiples and so on, until we consider the common multiples of all the power products of the basis of $In(I)$.

This is explained more formally in [26] by the introduction of a homogeneous free resolution of $R/In(I)$. Let's suppose the basis of $In(I)$ has p elements $m_1, \ldots, m_p$. The least common multiple of $m_{i_1}, \ldots, m_{i_l}$ will be denoted $lcm(m_{i_1}, \ldots, m_{i_l})$. The ring R is graded by using the total degree. For each l $(1 \leq l \leq p)$, the free module $M_l = R^{\binom{p}{l}}$ over R is graded in the following intricate way:

- The canonical basis of elements $e_j = (0, \ldots, 0, 1, 0, \ldots, 0)$ is reindexed by the set of l-tuples $\{(i_1, \ldots, i_l) | 1 \leq i_1 < \ldots < i_l \leq p\}$.
- We define the module-endomorphism α_l of M_l by $\alpha_l(e_{(i_1,\ldots,i_l)}) = lcm(m_{i_1}, \ldots, m_{i_l})\, e_{(i_1,\ldots,i_l)}$.
- Now, the component of M_l of degree q is defined by: $(M_l)_q = \alpha_l^{-1}(R_q \times \ldots \times R_q)$.

We must also define a family of homomorphisms. For $l > 1$, $h_l : M_l \to M_{l-1}$ is defined by:

$$h_l(e_{(i_1,\ldots,i_l)}) = \sum_{1 \leq \mu \leq l} \frac{(-1)^\mu\, lcm(m_{i_1}, \ldots, m_{i_l})}{lcm(m_{i_1}, \ldots, m_{i_{\mu-1}}, m_{i_{\mu+1}}, \ldots, m_{i_l})}\, e_{(i_1,\ldots,i_{\mu-1},i_{\mu+1},\ldots,i_l)}.$$

The homomorphism $h_1 : M_1 \to R$ is defined by $h_1(e_i) = m_i$, and $h_0 : R \to R/In(I)$ is just the canonical projection.

Naturally, we recognize the Taylor resolution of a monomial ideal that we used in section 2 to prove the correctness of Buchberger's algorithm.

Proposition 4.2 *The resolution*

$$0 \to M_p \xrightarrow{h_p} M_{p-1} \xrightarrow{h_{p-1}} \ldots M_1 \xrightarrow{h_1} R \xrightarrow{h_0} R/In(I) \to 0$$

is a homogeneous free resolution of $R/In(I)$.

Proof: See [26]. □

By restricting each endomorphism to the component of degree i, we obtain another exact sequence:

$$0 \to (M_p)_i \stackrel{h_p^i}{\to} (M_{p-1})_i \stackrel{h_{p-1}^i}{\to} \ldots (M_1)_i \stackrel{h_1^i}{\to} R_i \stackrel{h_0^i}{\to} (R/In(I))_i \to 0.$$

We can then consider all these modules as vector spaces over K. By applicating $p+1$ times the dimension formula, we obtain:

Proposition 4.3 *The Hilbert function of the ideal I is given by the formula*

$$H(i) = dim\, R_i + \sum_{l=1}^{p} (-1)^l\, dim\, (M_l)_i.$$

Here

$$dim(M_l)_i = \sum_{(j_1,\ldots,j_l)} dim^*(i - degree(\, lcm(\, m_{j_1}, \ldots, m_{j_l}))),$$

and $dim^*(i) = dim\ R^i$ if $i \geq 0$ and 0 otherwise.

This proves that the Hilbert function is equal to a polynomial for i large enough and gives a way to compute it.

The initial algorithm by Buchberger and two improvements are published in the paper by Mora and Möller, [26]. They are based on the formula above. Some other algorithms to compute the dimension of the ideal I once a Gröbner basis has been computed have been designed, for example by Kredel and Weispfenning, [23]. The problem with all these is that the computation of a Gröbner basis takes a long time, both in terms of worst complexity and of "observed" complexity. To avoid it, new algorithms are designed which do not need the computation of a Gröbner basis. Let us mention here the contribution of Giusti, [17], and the joint work of Galligo and Traverso, [15].

5. A parallel algorithm to compute Gröbner bases.

We first review the different contributions which have been brought by various people to the parallelization of Buchberger's algorithm. Then, we present our parallel algorithm to compute Gröbner bases and we prove its correctness. Finally, we describe the particular features of our implementation on an Encore Multimax machine and give a series of experimental results using examples commonly found in the literature.

5.1. History of the parallelization of Buchberger's algorithm.

The first two attempts to parallelize Buchberger's algorithm are due to Watt, [38] and to Buchberger himself, [10].

In his thesis, Watt proposes general tools to build computer algebra systems on networks. They allow only a limited interaction between processes. In fact, a process can only ask other processes to execute a program and to send back the results to it. For the computation of Gröbner bases, this leads to a hierarchical algorithm in which the main process asks other processes to simultaneously reduce the S-polynomials, waits for all of them to send back their result which are added to the basis. Then, the main process asks the other processes to simultaneously reduce each member of the basis w.r.t. the other members to obtain a basis in normal form. As Ponder states it in [31], this parallel inter-reduction is incorrect, because, if there are two polynomials with the same leading power product in the input basis of the inter-reduction, there will not be any polynomial with this leading power product in the result of the inter-reduction. Indeed, each one of them would have been reduced by the other. Beside this, there are two other main problems with this algorithm. First, pairs of polynomials whose S-polynomial has already been reduced are not remembered. Therefore, the same reductions are performed many times. Secondly, as the length of the reduction of an S-polynomial varies widely, many processors will be idle waiting for the others to finish their reduction. Watt does not give any indication on the performance of his implementation.

In [10], Buchberger describes two parallelizations of his algorithm which avoid the pitfalls described above. They are given as examples of use of the L-machine, a parallel machine designed at RISC-Linz for the specific use of parallel symbolic computation. The two schemes have a lot in common. They are based on the model of a set of processes which do not share memory but can exchange messages. In both, there is a main process called the *S Pair Set Manager* which distributes the work directly to the *Critical Pair Reducers.* When a Critical Pair Reducer has finished its reduction, it sends back the resulting polynomial to the S Pair Set Manager. If the resulting polynomial is not equal to 0, then the Critical Pair Reducer must also initiate the propagation of this new polynomial among all the Critical Pair Reducers. In this propagation lies the difference between the two schemes. In the first one, the Critical Pair Reducers are the leaves of a tree, whose internal nodes and root are processors of two other types, named respectively *G Set Distributor* and *Root Process.* New polynomials are propagated in the tree from a leaf. In the second scheme, Critical Pair Reducers form a ring and new polynomials are distributed in the ring. Therefore, this second scheme uses only two types of processes: the S Pair Set Manager and the Critical Pair Reducer. The main difference with Watt's approach is that the processes are less tightly coupled. The S Pair Set Manager does not wait for all the Critical Pair Reducers to complete their reductions before it sends to idle processes new pairs to reduce. Instead, as soon as it receives the result of a reduction, it stops the Critical Pair Reducers whose current work becomes useless because of the new polynomial entering the basis (if any), it recomputes the set of critical pairs to be reduced and it sends pairs to reduce to all the idle Critical Pair Reducers. As Buchberger did not implement these schemes, it is difficult to evaluate their efficiency. We can expect, though, that, when a large number of Critical Pair Reducers are used, the S Pair Set Manager becomes a bottleneck. Nevertheless, Buchberger's parallel versions seem much more efficient than Watt's scheme and lead the way to practical parallel implementations of the computation of Gröbner bases.

Ponder in [31] proposes four different parallelizations of Buchberger's algorithm. The two first are inspired by Watt's approach. They are in fact the two natural ways to transform Watt's algorithm into a correct one. With the method 1, S-polynomials are computed in parallel and, then, the basis is sequentially reduced. With the method 2, the S-polynomials are computed one by one, but the reduction of the basis w.r.t. each polynomial entering the basis is done in parallel. Ponder gives some performance results for these two algorithms. It seems that the average parallelism is very low, almost always under 1.5. Moreover, the time of the computation is

always much larger than the one taken by the careful sequential implementation made by Zacharias on top of Macsyma, [41]. Among the reasons of this meager speed-up, there are the criticisms that we formulate against Watt's approach, namely that the system does not keep track of the pairs already computed and that the reducers are tightly coupled. Moreover, Ponder does not seem to use the criteria described by Buchberger in [7] which avoid the computation of many useless S-polynomials. Ponder describes two other ways to parallelize Buchberger's algorithm. The first is a divide-and-conquer method which consists of recursively finding a Gröbner basis for disjoint subsets of the input set of polynomials and combining the obtained Gröbner bases. The problem is that this final combination seems generally as expensive as would have been the computation of the Gröbner basis for the input set from the start. The second uses Watt's idea of collusion, [38]. The Gröbner bases of an ideal w.r.t. several admissible orderings are computed simultaneously until one computation ends. Some statistics on how the order can change the time of the computation are given in [31] and the estimated efficiency of this collusion scheme is discussed. All four proposed algorithms are designed for a network of processors.

Melenk and Neun propose in [25] a fine-grain parallelization of the algorithm. In fact, it is the reduction that is parallelized, by using a pipeline of processes. As soon as the leading monomial of the result of one step of the reduction has been found, another process can begin a new reduction step before the first one is finished. What is needed is a synchronization tool to control production and access of successive monomials. This fine-grain parallelism should work reasonably well on models with low latency, such as shared memory multiprocessors. They give some simulation results in the paper. For two and three processors, the speed-up is almost maximal (equal to 2 and 3 respectively). With more processors, it seems that the speed-up cannot go over 4.

Siegl in [35] presents a different scheme. The underlying model of computation is a network of processors exchanging messages. Each process has access to only one polynomial of the basis. Therefore, reductions are done by pipelining the polynomial which is reduced through all the processors, each processor reducing it w.r.t. the polynomial that is attached to it. If the resulting polynomial is not equal to zero, then a process is added to the pipeline and the new member of the basis is associated with it. Then, another pipeline computation takes place to update the set of pairs that need to be reduced: the new polynomial is propagated from processor to processor, and each processor considers the pair formed by its associated polynomial and by the new member of the basis. It applies the criteria to avoid unnecessary reductions. It seems that there is a main process which initializes the computation, proposes to the pipeline S-polynomials that need to be reduced and also collects the S-polynomials that other processors decide to reduce (in the second pipelined operation) so that they can be reduced later. Some performance results are given. They concern relatively small examples. For these, the speed-up is always between 2.5 and 6. An interesting issue is the behavior of this system when the number of polynomials in the basis increases drastically. The size of the polynomial going from processor to processor to be reduced is also a critical issue. It may be the case that, when these polynomials get big, a lot of time is spent in moving them around.

Finally, Senechaud describes in [33] a parallelized version of Buchberger's algorithm on Boolean polynomials. The scheme is similar to Siegl's. The main difference is that each process "contains" more than one polynomial of the basis. Also, it seems that the algorithm does not use the criteria proposed by Buchberger to avoid unnecessary reductions. The only performance result concerns the computation of a 32 polynomials input basis with 5 variables. The speed-up factor is 3.3 with 2 processors, 5.7 with 4 processors and stays under 8 for additional processors.

Before we introduce our own parallel version of Buchberger's algorithm, let us present in the next section the tools that we will need to synchronize the actions of the different processes.

5.2. Description of the needed synchronizing tools.

The following synchronizing concepts appear in many places in the literature. Our source is the book "Operating Systems Principles" by Per Brinch Hansen, [3], pages 77-116.

Critical regions. A variable v of type T which is shared among several processes is defined by the notation:

```
shared T v
```

If the value of v changes during the computation, a mechanism is needed to prevent one process from reading the value of v when another process is updating it. Otherwise, if v has several components, the process which reads may end up with an illegal value of v, with certain components of the old value and the other components of the new value. The tools which allow such mutual exclusion of processes sharing a variable are called *critical regions*. As stated by Brinch Hansen, they have the following properties:

- A process waiting before a critical region will eventually enter it.
- If several critical regions refer to the same variable, only at most one process can be in one of these critical regions.
- A process entering a critical region will leave it eventually.

To define a critical region *protecting* the variable v, we will use this notation:

```
region v do statement
```

Waiting for a condition to hold. Sometimes, it is needed that a process, in the middle of a critical region for a shared variable v, tests for some Boolean expression involving components of v and goes on only when the Boolean expression is true. Now, if the Boolean expression is false and if the process just stops in its critical region, it prevents any other process to enter a critical region for the same variable v and, therefore, the variable v will never be updated and the process will stay in its critical region for ever. This is in contradiction with the third condition defining critical regions. So, we need to modify our mechanism.

In a critical region for v, a process tests a Boolean expression involving components of v by using the primitive `await`. If the expression is true, then the process goes on. If the expression is false, then the process leaves its critical region temporarily and joins an event queue associated with v. When a process leaves definitely a critical region for v (i.e. not by executing an `await` statement) , it may have modified the value of v. Therefore, the processes on the event queue are allowed to enter again their critical regions just before the `await` statement, and to test again the Boolean expression. Of course, they do so one at a time to ensure mutual exclusion.

Readers and writers exclusion. Brinch Hansen use the readers and writers exclusion as an example of the use of conditional critical regions ([3], pages 106-114). We would like to define several primitive constructs, allowing us to express this higher level of synchronization in a clear and easy way.

We want to define in particular a synchronization tool associated with a shared variable v with the following properties:

- Any number of processes which desire to read v may do it concurrently.
- Only one process may change the value of v at any time.
- No process is reading v when another process changes its value.
- Any process requesting a read or update of v will eventually be allowed to do so.
- Writers have priority over readers in the following sense: as soon as a process wants to change the value of v, no other process is allowed to read it until the writing process has updated v.

We understand that the second constraint, restricting the number of writers to one, diminishes the generality of the construct and suppresses the symmetry of the problem. Nevertheless, we keep it because this makes the expression of our algorithm very simple.

When each location of the shared memory is protected by such a mechanism, we obtain the Concurrent Read Exclusive Write (CREW) version of the Parallel Random Access Machine (PRAM). Therefore, to precise that a shared variable v of type T is protected by such mechanism, we will declare it as follows:

```
CREW T v
```

We also define two primitives to grant read and write accesses to v. First:

```
with read_access v do P
```

When a process executes this statement, it looks if a process has already requested to change the value of v. If so, it waits until the writer has finished and it tries again to gain read access to v. When it is successful, it executes the statement P and leaves the protected region. Next:

```
with write_access v do P
```

The process executing this statement needs first to compete among all the processes which requested to write on v. When it becomes the only process allowed to write, it prevents other processes to obtain read access to v, waits until all the processes which read have left the region, executes the statement P and leaves the region.

To see how to implement these constructs with conditional critical region, see Brinch Hansen's book. The exclusion among writing processes is not enforced in Brinch Hansen's presentation but it is straightforward to add, since we need only to enclose writing regions for v in critical regions for a new variable.

5.3. Description of the algorithm.

First we restate Buchberger's algorithm in Figure 2, in a way which reveals the opportunities to design a parallel version of it.

```
Input:  a set B of polynomials, basis of an ideal I.

Basis:= B;
Pairs:= init_pairs(Basis);
while (size(Pairs) <> 0) do
  begin
    pair:= choose(Pairs);
    Q:= reduce(spol(pair),Basis);
    if (Q ≠ 0) then
      begin
        Basis:= Basis ∪ {Q};
        Pairs:= update_pairs(Pairs,Basis,Q)
      end;
  end;
```

Figure 2: High-level description of Buchberger's algorithm

If we dispose of several processors, it is possible to reduce several pairs simultaneously. At first, we do not want to go at a finer granularity than the one described by the algorithm above. The only two structures that we need to provide with a mechanism of exclusion are the list of polynomials and the list of pairs to be reduced. The ways the processes share these two variables are distinct:

- Each time the set of pairs is accessed, it is modified. Therefore, a critical region is sufficient to protect the set of pairs. But, before a pair is chosen, it is necessary to know if there is any pair in the set. For this, we need to access the structure, so this must be done inside the critical region. Therefore, we need to use an `await` statement.
- During the reduction of the S-polynomial of the chosen pair, a process uses many times the list of polynomials in the basis but never updates it. Processes reducing a polynomial should be allowed to use this list simultaneously. This is important because the time spent in the reduction process is much longer than the one spent in updating the basis. So we need to install a mechanism such that several concurrent readers or one exclusive writer (but not both) use the list of polynomials at a given time. Therefore, a CREW mechanism is needed to protect the set of polynomials.

We can now present in Figure 3 the algorithm that each process executes. The different processes are started after the initialization of the set of pairs to compute has been done by a single process. There is, however, an obvious problem with this program: the computation ends by a deadlock. When `size(Pairs) = 0` and when all the processes are waiting for new pairs in the `await` statement, then they will wait for ever. To transform the above program in a deadlock-free one is straightforward: we need a way to count the processes waiting for the event `size(Pairs)>0`.

```
shared list_of_pairs Pairs;
CREW list_of_pols Basis;


process worker(i)

  begin
    while (true) do
      begin
        region Pairs do
          begin
            await (size(Pairs) > 0);
            pair[i]:= choose(Pairs);
          end;
        with read_access Basis do
          Q[i]:= reduce(spol(pair[i]),Basis);
        if (Q <> 0) then
          with write_access Basis do
            region Pairs do
              begin
                Basis:= Basis ∪ {Q[i]};
                Pairs:= update_pairs(Pairs,Basis,Q[i]);
              end;
      end;
  end;
```

Figure 3: First attempt

```
shared list_of_pairs Pairs;
shared integer nwaiting;
CREW list_of_pols Basis;

process worker(i)

  begin
    while (true) do
      begin
        region Pairs do
          begin
            nwaiting = nwaiting + 1;
            if ((nwaiting == ntotalprocesses)
                      and (size(Pairs) == 0) then
                conclude();
            await (size(Pairs) > 0);
            pair[i]:= choose(Pairs);
            nwaiting = nwaiting - 1;
          end;
        with read_access Basis do
          Q[i]:= reduce(spol(pair[i]),Basis);
        if (Q[i] <> 0) then
          with write_access Basis do
            region Pairs do
              begin
                Basis:= Basis ∪ {Q[i]};
                Pairs:= update_pairs(Pairs,Basis,Q[i]);
              end;
      end;
  end;
```

Figure 4: A terminating version

The shared variable `nwaiting` will be associated with the same regions than the variable `Pairs`. Then, before executing the `await` statement in the first critical region, a process tests if the two following conditions are simultaneously satisfied: all the processes (except the tester itself, of course) are waiting and the set of pairs to compute is empty. In this case, it calls the function `conclude` which terminates the computation and make all the processes exit: we present this program in Figure 4.

5.4. Correctness of the program.

Susan Owicki introduced proofs of the correctness of parallel algorithms in her thesis, [27] and in an article she co-authored in 1976 with David Gries, [28]. We will not prove formerly our algorithm correct in this section but only annotate it with assertions which give strong evidence of its correctness.

Owicki introduces *auxiliary variables*. These are variables which appear only in left-hand sides of some assignments and whose only function is to be a witness of the meaningful components of the state of the program at a given point in time. We need a way to tell if a process is currently reducing a pair or if it has already finished and is waiting to begin to compute a new one. Instead of introducing an auxiliary variable, we use the variables `pair[i]`. We present our modified program in Figure 5.

To transform `pair[i]` in a witness of the state of the processor, we had to change slightly the program. Even when the pair is reduced to 0, the process enters a critical region for `Pairs`. This is unnecessary and moderately inefficient but allows us to define a nice invariant for the resource `Pairs`.

We would like to prove that the program above computes the Gröbner basis of an ideal. We will use the following characterization of Gröbner bases which is very similar to the condition (D-II) of the Fundamental Theorem presented in Section 2. A set B of polynomials is a Gröbner basis w.r.t. the admissible order $\succ$ iff, for every pair (P_1, P_2) of members of B, there is at least one reduction to 0 of the S-polynomial of P_1 and P_2 with respect to the set of rules $\mathcal{R}(B, \succ)$. Naturally, as the reduction w.r.t. a Gröbner basis has the property that any polynomial has a unique normal form, in fact, all the possible reductions of the S-polynomial of P_1 and P_2 will end at 0. But we only need to prove that our program satisfies the weaker requirement and we will obtain the stronger for free.

The first observation is that at most one process at a time is updating any of the two shared objects `Pairs` or `Basis`. This is just because any change to `Pairs` or `Basis` is made inside a critical region for `Pairs`. It is also true that all n variables `pair[i]` are updated in a critical region associated with `Pairs`.

Therefore, we introduce an invariant involving `Pairs`, `Basis`, and the variables `pair[i]`. This invariant $\mathcal{I} = Inv(Pairs, Basis, pair[1], \ldots, pair[n])$ must be true whenever there is no process in a critical region for `Pairs`. It says that the only pairs for which the S-polynomial may not be reduced to 0 w.r.t. `Basis` are the ones in `Pairs`, the ones stored in the local variables `pair[i]`, if any, and their symmetric ((P,Q) and (Q,P) are two symmetric pairs).

```
shared list_of_pairs Pairs;
shared integer nwaiting;
CREW list_of_pols Basis;

process worker(i)

  begin
    while (true) do
      begin
        region Pairs do
          begin
            nwaiting = nwaiting + 1;
            if ((nwaiting == ntotalprocesses)
                  and (size(Pairs) == 0)) then
                conclude();
            await (size(Pairs) > 0);
            pair[i]:= choose(Pairs);
            nwaiting = nwaiting - 1;
          end;
        with read_access Basis do
          Q[i]:= reduce(spol(pair[i]),Basis);
        region Pairs do
          begin
            if (Q[i] <> 0) then
                with write_access Basis do
                  begin
                    Basis:= Basis ∪ {Q[i]};
                    Pairs:= update_pairs(Pairs,Basis,Q[i]);
                  end;
            pair[i]=0;
          end;
      end;
   end;
```

Figure 5: Use of `pair[i]` as auxiliary variables

Formally stated, the invariant reads as follows:

$$\begin{array}{l}\forall\,(P,Q) \in Basis \times Basis \\ \quad \{P \neq Q \ \wedge\ (P,Q) \notin Pairs \ \wedge\ (Q,P) \notin Pairs \ \wedge \\ \qquad [\forall i \in \{1,\ldots,n\},\ pair[i] = 0 \vee (pair[i] \neq (P,Q) \wedge pair[i] \neq (Q,P))]\} \implies \\ \quad spol(P,Q) \longrightarrow^{*}_{Basis} 0\end{array}$$

We supposed $\mathcal{I}$ to be true before the n workers begin to work. This means that the set of Pairs has been correctly initialized, first by generating all the pairs {P, Q} of distinct polynomials in the initial basis and then by eliminating pairs that do not need to be computed because of the criteria presented in subsection 2.3 above. Initially, all the variables `pair[i]` are equal to 0.

As `Pairs`, `Basis`, and `pair[i]` are only changed in the critical regions for `Pairs`, we want to state, for each of these two critical regions, that, if $\mathcal{I}$ is true after the process enters the region, it will be true just before the process leaves the region. Our annotated program is in Figure 6.

We leave to the reader the proof of the assertions. If we assume that they are true, then, when a process reaches the statement `conclude()`, the formula

$$\forall\,(P,Q) \in Basis \times Basis, P \neq Q \implies spol(P,Q) \longrightarrow_{Basis} 0$$

holds. Therefore, at this point `Basis` is a Gröbner basis for the given admissible order.

5.5. Improvement of the algorithm.

Now that we stated the correctness of a simple version of the algorithm, we would like to make some progress towards more efficient algorithms.

More concurrency between writers and readers. We remark first that a process which computes a lengthy reduction can prevent another one (or several) from updating the basis of polynomials and the set of pairs. Therefore, we need to work at a finer granularity. For this, we need to make the distinction between the polynomials in the basis and the basis itself, which is just a list of pointers to the polynomials. We need to enforce a CREW mechanism on the list, but not on the polynomials themselves, because, once created, they are just read and never modified.

Thus we can rewrite the statements:

```
with read_access Basis do
  Q[i]:= reduce(spol(pair[i]),Basis);
```

in the following way. After it has computed the S-polynomial of the pair and stored the result in `Q[i]`, the process asks for a grant to read the list of polynomials. Then it traverses this list, looking for a polynomial with which it can reduce `Q[i]`. If it finds such polynomial, it returns a pointer to it; otherwise, it returns the null pointer. Then, it releases its grant to read the list of polynomials and, if the search on the list was successful, it reduces `Q[i]` by the polynomial it found, reclaims the grant to read the list of polynomials, tries to find a new polynomial to reduce `Q[i]`, and continue on until it cannot reduce `Q[i]` any more.

Here is the modified code:

```
Q[i]:= spol(pair[i]);
with read_access Basis do
  red[i]:= reduce_choice(Q[i],Basis);
while (red[i]≠0) do
  begin
    Q[i]:= reduce_one_step(Q[i],red[i]);
    with read_access Basis do
      red[i]:= reduce_choice(Q[i],Basis);
  end;
```

With this version, between two steps of the reduction a writer can update the basis. The drawback is that more time is spent in locking. But this is worth the trouble, for the time spent in the function `reduce_choice` is considerably less than the time spent in the function `reduce_one_step`. Therefore, the constraints imposed on the writers by the readers are much smaller in this version.

Avoiding polynomials with the same leading power product in the basis. In the version of the algorithm that we described in the Section 6.3, the synchronization is such that the program oscillates between two modes. In the first one, all the processes reduce pairs concurrently (if there are enough pairs to reduce). In the second one, all the processes update the basis successively (if the pair they considered did not reduce to 0) and then wait for the other processes to do so. Thus, it is possible that two polynomials resulting of concurrent reductions have the same leading power product. After the improvement described in the previous subsection, the program does not oscillate between two modes any more. Nevertheless, it is still possible to introduce elements with the same leading power.

Adding to the basis a polynomial P with the same leading power product as another polynomial Q in the basis is inefficient for the following reason. The insertion of Q increases the size of the set of pairs by 1, the new pair being $\{P,Q\}$ (it is also possible that some pairs $\{P,R\}$ are replaced by the corresponding pairs $\{Q,R\}$). When the S-polynomial of $\{P,Q\}$ is computed, the operation is exactly similar to the reduction of Q by P. So, if Q had been reduced by P before being inserted in the basis, we would have spared one call to `update_pairs` which can be quite expensive.

To forbid insertion of polynomials with the same leading power, we enclose the part of the code in which the basis is updated in another critical region, to enforce mutual exclusion between potential writers. When a process enters this critical region, it tries to reduce `Q[i]` another time. With this additional mechanism, polynomials added to the basis are always reduced w.r.t. the ones which are already in the basis. Here is the new code:

```
if (Q[i] <> 0) then
  begin
    region update do
      begin
        Q[i]:= reduce(Q[i],Basis);
        if (Q[i] <> 0) then
          region Pairs do
```

```
            with write_access Basis do
              begin
                Basis:= Basis ∪ {Q[i]};
                Pairs:= update_pairs(Pairs,Basis,Q[i]);
              end;
    end;
end;
```

This feature allows the number of polynomials introduced in the basis to be smaller. But the large size of the critical region `update` limits the concurrency. Some more extensive experiments are needed to decide if it is a useful feature or not. In the meantime, our program will allow the user to choose one policy or the other by setting a flag.

Removing the CREW mechanism. The algorithms we presented so far used a CREW mechanism to protect the list of polynomials in the basis. This allows a compact description of the algorithm, but we can remove it. To do this, we need to implement the list of polynomials as a linked list. The different processes update the list of polynomials inside a critical region which ensure mutual exclusion among writers. But readers have free access to this list. We consider here only the case in which polynomials in the basis are never deleted from it. Therefore, we only need to be able to add an element in a list in one machine instruction. Here is how we do this: to insert element a between b and c, the process makes a point to c (at this point the list ... b c ... is still valid and other processes can use it). Then, in one instruction, it makes a point to c and the list is updated.

5.6. Features of the implementation.

Our program is written in the C programming language. A package handling multivariate polynomials was developed first. It can be easily adapted to polynomials with coefficients in any computable ring but we used it only with integer coefficients. A package to handle integers of arbitrary length was available: **cmump** was developed at Carnegie-Mellon University by Rex Dwyer, Lyle McGeoch, Guy Jacobson and Bennet Yee on top of the **mp** package developed at Bell Labs. We didn't need to build a package handling rational numbers because, by multiplying by suitable coefficients, the computation of the Gröbner basis of an ideal of $\mathbf{Q}[x_1, \ldots, x_n]$ can be done with polynomials in $\mathbb{Z}[x_1, \ldots, x_n]$.

The polynomials are implemented as lists of monomials written in decreasing order w.r.t. the chosen admissible order. Each power product is implemented as an array of short integers. Each integer in the array represents the power of a certain variable in the power product. So, we choose the obvious implementation of power products and we didn't try the one used in the program Macaulay, where power products are represented by their ordinal number w.r.t. the chosen admissible order. Macaulay is the fastest implementation of Buchberger's algorithm. It is written by Bayer and Stillman. Hopefully, this will be tried in the future.

Useless pairs are filtered by the function `updatePairs`. The algorithm it uses is described in a recent paper by Gebauer and Möller, [16], where a proof of correctness can also be found.

The synchronization between the processors was implemented by using the **cthread** package developed at Carnegie-Mellon University by Eric C. Cooper and Richard D. Draves, see [12].

```
shared list_of_pairs Pairs;
shared integer nwaiting;
CREW list_of_pols Basis;

process worker(i)

  begin
    {pair[i] = 0}
    while (true) do
      begin
        region Pairs do
          begin
            {I ∧ pair[i] = 0}
            nwaiting:= nwaiting + 1;
            if ((nwaiting = ntotalprocesses)
                  and (size(Pairs) == 0) then
                {I ∧ (size(Pairs) = 0) ∧ ∀ i ∈ {1,...,n} , pair[i] = 0}
                conclude();
            await (size(Pairs) > 0);      {I ∧ (size(Pairs) ≥ 1) ∧ (pair[i] = 0)}
            pair[i]:= choose(Pairs);      {I ∧ (size(Pairs) ≥ 0) ∧ (pair[i] ≠ 0)}
            nwaiting:= nwaiting - 1;
          end;
          with read_access Basis do
              begin
                AuxBasis[i]:= Basis;
                Q[i]:= Reduce(Spol(pair[i]),Basis);
                {Spol(pair[i]) ⟶_Basis Q[i]}
              end;
          {Spol(pair[i]) ⟶_AuxBasis[i] Q[i]}
          region Pairs do
              begin
                {I ∧
                  [(Q[i] = 0 ∧ Spol(pair[i]) ⟶_AuxBasis[i] 0) ∨
                    (Q[i] ≠ 0 ∧ Spol(pair[i]) ⟶_AuxBasis[i] ∪ {Q[i]} 0)] ∧
                  pair[i] ≠ 0 ∧ AuxBasis[i] ⊆ Basis}
                if (Q[i] ≠ 0) then
                  with write_access Basis do
                    begin                  {I ∧ (Q[i] ≠ 0) ∧ (Spol(pair[i]) ⟶_Basis ∪{Q[i]} 0)}
                      Basis:= Basis ∪ Q[i];
                      {Inv(Pairs, Basis − {Q[i]}, pair[0],...,pair[n]) ∧ (Spol(pair[i]) ⟶_Basis 0)}
                      Pairs:= update_pairs(Pairs,Basis,Q[i])
                      {I ∧ (Spol(pair[i]) ⟶_Basis 0)}
                    end;
                  {I ∧ Spol(pair[i]) ⟶_Basis 0 ∧ pair[i] ≠ 0}
                  pair[i]:= 0;             {I ∧ pair[i] = 0}
              end;
        end;
      end;
```

Figure 6: Annotated program

This package allows parallel programming in C under the MACH operating system, developed at Carnegie-Mellon University by Rich Rashid and al., [1]. This package gave us the level of abstraction that we needed; only the "concurrent readers - exclusive writer" mechanism was not provided by it.

Our goal was to design a tool with all the "traditional" options (i.e., choice of the order on the power products, possible deletion of redundant polynomials in the basis in the middle of the computation, different levels of traces, different ways to choose a polynomial to reduce another one when several polynomials in the basis may be selected) plus several options directly related to the parallel version of the algorithm (i.e., number of processors involved in the computation and choice among two algorithms for the dynamic allocation of memory). Several systems have been already implemented to test the options not related to parallelism. We mention the AIPI system built by Carlo Traverso and Leombattista Donati in Pisa, [37].

Choice of the order of the power products. Only certain admissible orders can be used: lexicographic order, total degree order refined by lexicographic order, total degree refined by reverse lexicographic order and the very useful type of order defined as follows. The variables are split in two groups $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$. To compare two power produts, the two subproducts containing only the variables of group x are compared first according to a certain order (in our case, the total degree order refined by reverse lexicographic order). Ties are broken by comparing the two subproducts containing only the variables of group y. This order is a compromise between the pure lexicographic order, which is useful to eliminate variables, and the total degree order refined by the reverse lexicographic order, for which the computations are generally faster.

Deletion of redundant polynomials in the basis. When the option is not selected, all the polynomials which are added to the basis are kept in it until the final reduction step. When the option is selected and after a pair (P, Q) has been reduced such that the leading power product of P divides the leading power of Q, the polynomial Q is redundant and it is eliminated from the basis after all the remaining pairs of which it is a member have been reduced.

Different ways to pick a pair in the set of pairs. Each time a processor executes the main loop, it has to choose a pair in the set of pairs. Numerous articles present the heuristic which consists in choosing a pair for which the least common multiple of the two leading power products is of lowest degree. When several such pairs exist, ties are broken by taking the one for which this least common multiple is the lowest w.r.t. the admissible order. To test the importance of this decision, we offer several other possibilities: smallest lcm w.r.t. the admissible order first, highest lcm w.r.t. the admissible order first, lcm of highest degree first (ties are broken by choosing the pair with highest lcm w.r.t. the admissible order).

There is another issue concerning the order in which the pairs are reduced. Some pairs are particular, because the leading power product of one of the polynomial divides the leading power product of the other polynomial. When the S-polynomial of these pairs are reduced, this corresponds to an inter-reduction of the basis, i.e. to a simplification of the basis (at least, in theory). Therefore, it is very tempting to compute these pairs first. Following the work of Traverso and Donati in the AIPI system, we let the user choose if he or she wants to give priority to these particular pairs.

Different options to select polynomials in the basis for reduction. When we want to reduce a polynomial P w.r.t. the basis B, it is possible that several polynomials of the basis have leading monomials dividing the leading monomial of P and, therefore, we have to choose among

them. The Gröbner basis that we obtain is independent from this choice but it can affect the length of the computation. The user has several choices: the chosen polynomial can either be the one with the smallest leading power product w.r.t. the admissible order $\succ$, the one with the highest leading power product w.r.t. $\succ$, the one which arrived first in the basis or the one with the smallest number of monomials.

Allocation of dynamic memory. The two functions `malloc` and `free` which are used in the dynamic allocation of memory in C have been rewritten in the cthread package to ensure the correctness of their behaviors when they are used simultaneously by several processes. But the solution which was adopted consists in enclosing the uniprocessor version of `malloc` and `free` in a critical region. Our program make a very large use of `malloc` and `free`: for each polynomial which appears during the computation, we allocate separately memory for its global structure and for each of its coefficients. In an attempt to allow more concurrency between the processes, we implemented another mechanism to allocate memory dynamically and we let the user choose the one which fits his/her needs best.

Our mechanism is very simple, because each process has its own list of free blocks. When a process frees some memory, it returns it to its own list of free blocks, even if another process allocated it. When a process allocates some memory, it takes it from its list of free blocks, using a simple algorithm. If there is no free block left, the process asks the operating system for more working space.

5.7. Experimental results.

The particular examples that we use come from two articles, one by Boege, Gebauer and Kredel, [2], and the other one by Traverso and Donati, [37]. These examples are listed in Appendix B.

Our main goal is to show that our concurrent version of Buchberger's algorithm leads to some speed-up. Therefore, we are mostly interested in the comparison of several computations executed with the same setting of the flags but with a different number of processors in each case. For this reason, all the examples listed here were done with this setting of the flags:

- There is no deletion of redundant elements of the basis during the computation.
- When several polynomials can be used to reduce another one, the one which entered the basis first is selected (chronological order).
- The useless pair elimination is done by using the Gebauer-Möller criterion which is defined in [16].
- The pair which is selected in the set of pairs is the one with least common multiple of smallest degree. If there exist several such pairs, we choose the one with smallest lcm w.r.t. the admissible order. We do not give any priority to pairs for which the leading power product of one of the polynomial is divisible by the leading power product of the other polynomial.
- We limit ourselves to three admissible orders: pure lexicographic order, total order refined by lexicographical order and pure lexicographic order refined by reverse lexicographical order. We list in Appendix B the order chosen among the variables. We tried to keep the same order than in [37] when it was possible.

Name	O	1 proc			2 proc			5 proc			12 proc		
		#P	#R	time	#P	#R	Time	#P	#R	time	#P	#R	Time
Arnborg4	$\succ_L$	12	14	0.3 (0.3)	12	14	0.2 (0.2)	12	14	0.2 (0.2)	12	14	0.3 (0.3)
Arnborg5	$\succ_{TL}$	59	147	57 (67)	59	148	30 (33)	64	159	13 (13)	61	158	13 (10)
Arnborg5	$\succ_{TR}$	47	106	38 (41)	47	106	20 (21)	50	112	9 (9)	58	129	12 (8)
Butcher	$\succ_{TR}$	66	159	326 (368)	70	171	196 (209)	74	180	74 (77)	79	199	69 (72)
Katsura4	$\succ_{TL}$	21	38	16 (17)	22	39	10 (10)	24	41	4 (4)	29	53	5 (4)
Katsura5	$\succ_{TR}$	26	68	1089 (1103)	26	68	549 (551)	28	70	200 (146)	29	71	151 (79)
Lazard	$\succ_{TR}$	35	54	51 (56)	33	54	17 (22)	30	48	2.5 (2.4)	31	58	3.8 (2.1)
Morgenstern	$\succ_{TR}$	29	94	44 (45)	29	94	19 (19)	29	94	11 (8)	29	94	11 (8)
Pavelle4	$\succ_{TR}$	16	32	4 (4)	16	32	2.3 (2.3)	16	32	1.3 (1.2)	16	35	1.2 (1.1)
Robbiano	$\succ_L$	30	91	10 (11)	42	97	8 (7)	36	86	2.8 (3.5)	42	107	3.1 (3.1)
Rose	$\succ_{TR}$	22	39	37 (40)	23	40	16 (16.7)	25	49	5 (5)	26	54	3.5 (2.7)
Trinks1	$\succ_{TL}$	18	27	10 (10)	19	29	4 (5)	19	28	2.2 (1.7)	19	30	1.8 (1.9)
Trinks2	$\succ_L$	22	28	4 (4)	23	31	2 (2)	22	29	0.9 (1)	22	29	0.9 (1)
Valla	$\succ_{TR}$	80	524	905 (932)	80	524	474 (482)	80	524	237 (216)	90	535	178 (129)

Figure 7: Experimental results.

- We use two different algorithms for dynamic allocation of memory: one with a global list of free blocks for all processors and one with a separate list for each processor.

For each computation, we give in Figure 7 the number of different polynomials which enter the basis "#P"(including the initial ones), the number of pairs which are reduced "#R" and the time the computation takes in seconds on a Multivax Encore (16 processors). The times we give do not include the final interreduction which occurs at the end of the computation to obtain the unique reduced basis. This is our experience that this time is much smaller and, therefore, is far less important than the time needed to compute a Gröbner basis which is not reduced. The computations with normal allocation policy and with our allocation policy differ essentially in the time of the execution. Therefore, we give only one value for #P and #R for each example, write the two times in the same column and put parentheses around the time obtained with our own allocation policy.

When the number of processors is greater than one, there is naturally no guarantee that another computation with the same flags will give the same numbers : this depends on the way the different processors interleave. In fact, results can vary widely. We give the best results we obtained. We hope to explain and to correct these wide variations in a later work.

The results are encouraging, especially for a small number of processors. In particular, it seems that using two processors instead of one cuts the the time almost by two, at least for computations which are long enough. Naturally, when the number of processors is large, it is very possible that some processors are idle, waiting for other processors to finish their reduction and to put some pairs in the set of pairs to compute. It would be interesting to have a measure of the maximum number of processors that will be used during a computation in terms of algebraic structure of the ideal.

5.8. Further possible improvements and conclusion.

We do not talk here of the the improvements that could come from a better understanding of the behavior of the algorithm. This is a hard problem that requires theoretic developments first. We

hope that our program, by providing examples, will encourage these developments. So, we will concentrate here on two technical improvements that could more easily be made in the future.

Package for integers of arbitrary length. We did not take the time to rewrite it. It seems that we could gain some speed-up by rewriting the function which performs multiplications. In the **cmump** package, the naive algorithm is used ($O(m\ p)$ for two integers of respective sizes m and p). We could try to implement a faster algorithm, although this would certainly slow down the computation of Gröbner bases for which the coefficients of the polynomials stay in a reasonable size. So, here also, theoretical results are needed. We could also choose radically different implementations of integers of arbitrary length. For example, factorizing the integers into their prime decompositions can give good results in some cases. In this representation, products, least common multiples, and greatest common divisors are very easy to compute, but addition is very expensive to perform. Therefore, this representation of integers will give good results only when the number of additions performed is small compared to the number of multiplications.

Galligo and al suggest in [14] a partly factorized form for coefficients of polynomials. Their basic idea is to replace computations of the GCD by computations of a divisor which is large enough but faster to compute than the GCD: the Greater Easy Common Divisor. We would like to implement their algorithms to compare it with the one we already have.

Another way to implement integers of arbitrary length is by using modular arithmetic and especially the Chinese Remainder Theorem. An integer l is represented by an array of numbers l modulo $p_1,\ldots,l$ modulo p_k where $p_1,\ldots,p_k$ are distinct primes. With this implementation, multiplication and addition are easy to perform. Several implementations take this approach but they limit themselves to a number of primes fixed in advance. Macaulay, for example, computes the Gröbner bases in $Z_{31991}[x_1,\ldots,x_n]$. The problem with this approach is that we do not know how big the coefficients will get and, therefore, it is impossible to know beforehand how many primes are needed.

In another very similar approach, the Gröbner basis in $Q[x_1,\ldots,x_n]$ is computed by computing first the Gröbner bases in $Z_{p_i}[x_1,\ldots,x_n]$ for several prime numbers p_i and then lifting them to $Q[x_1,\ldots,x_n]$. This method has a same drawback: we do not know how many primes are needed. We refer the interested reader to a recent paper by Winkler, [40].

Package for multivariate polynomials over $\mathbb{Z}$. We can list several different implementations of the multivariate polynomials that are worth trying in Buchberger's algorithm. None of them is based on a recursive representation ($\mathbf{Q}[x_1,\ldots,x_n] = ((\ldots(\mathbf{Q}[x_1])[x_2])\ldots)[x_n]$) but on expanded representation of polynomials. So a polynomial is always represented as a list of monomials. A monomial is composed of a coefficient, which is an integer of arbitrary length, and an exponent, which is represented as an array of integers. Based on this, two representations can be designed: in the first one, the coefficient itself is stored in the structure; in the other one, only a pointer to the coefficient is stored. The advantage of the first solution, beside the small gain of space, is that it needs fewer calls to the dynamic allocation functions. But the second solution is much simpler.

The list of monomials can be represented either as an array or as a linked list. The advantage of the representation by array is, once again, that it generates fewer calls to the dynamic allocation functions. But linked lists would be very efficient to implement the addition of two polynomials P and Q in the special case where the result is put in P or in Q. As this kind of function on polynomials is used during most of the time of the reduction, this could lead to some reasonable speed-up.

In the current version, the representation of monomials contains a pointer to the coefficient, instead of the coefficient itself, and the list of monomials is implemented as an array. We hope to test other implementations in the future.

Conclusion. As computer algebra is a subject which grows extremely fast, this presentation of Gröbner bases is not as comprehensive as the author would have liked. In particular, the list of applications that are mentioned is certainly not exhaustive. Some very important ones are not considered (e.g., the determination of the solutions of an algebraic equation and the recent article by Gianni, Trager and Zacharias, [29], on primary decomposition). There is much, therefore, that remains to be done.

We have put forward as the essential contribution of this work the design and implementation of a parallel version of Buchberger's algorithm. It gave us some experience on the size of the overload due to the collaboration between processors. The fact that the number of polynomials introduced in the basis does not grow too fast when the number of processors increases is very encouraging. This makes a usable distributed version of the algorithm a feasible goal.

A Table of notations.

In each table, we list the notations which have been used for the first time in the corresponding section.

Notations for section 1.

E, E'	index sets
i, j	elements of an index set (possibly $\mathbb{N}$)
l, n, p	integers
R, R'	Noetherian commutative ring and subring
ϕ	homomorphism from $R'[x_1, \ldots, x_n]$ to R
r, r_i, r_{ij}, s_i	elements of R
K, K'	field and subfield
k_i	elements of K
k	element of K or K^n
M, M', M'', M_i	modules or submodules over R or K
m, m_i	elements of the module M
x, x_i	elements of an extension of R or K algebraically independent over R or K (variables)
A	polynomial ring $R[x_1, \ldots, x_n]$ or $K[x_1, \ldots, x_n]$
$P, P', P'', P_i,$ Q, Q', Q'', Q_i $I, I', I'', I_i,$ J, J', J'', J_i	polynomials of A ideals of R or A
$\overline{P}$	the class of P in R/I (or in A/I)
μ	canonical homomorphism from R (or A) to R/I (or A/I)
T, T_i	terms of the commutative monoid generated by $\{x_1, \ldots, x_n\}$ (power products)
B	(finite) subset of elements of R or A
$(B), Id(B)$	the ideal of A generated by the elements of B
$\succ$	order relations on the set of power products
$In_\succ(P),\ In(P)$	highest monomial of P w.r.t. $\succ$
$In_\succ(I),\ In(I)$	ideal of the highest monomials w.r.t. $\succ$ of all the elements of I
S, S_i	subset of K^n
$\mathcal{I}(S)$	ideal of $K[x_1, \ldots, x_n]$ containing all the polynomials which vanish on S
V, V_i	affine algebraic variety of K^n
$V(P), V(B), V(I)$	affine algebraic variety of K^n associated with $P \in A$ or B, $I \subseteq A$
O_P	$\{(k_1, \ldots, k_n) \in K^n \mid P(k_1, \ldots, k_n) \neq 0\}$
$\mathcal{V}_k$	neighborhood of $k \in K^n$ for the Zariski topology
$nilrad(R),\ nilrad(I)$	nil radical of R or of an ideal I of R or A
$X(R), X_{max}(R)$	set of prime ideals of R, set of maximal ideals of R
$Pr(B)$	set of prime ideals of R containing B
$Max(B)$	set of maximal ideals of R containing B
X_r	$X(R) - Pr(\{r\})$
Ψ	homeomorphism between $X_{max}(A)$ and K^n

Notations for section 2.

m, m', m_i	monomial of A
r, r_i	rule of the form $m \rightarrow P$
$\mathcal{R}$	set of rules
$Q_1 \longrightarrow_r^1 Q_2$	Q_1 is reduced in one step to Q_2 w.r.t the rule r
$Q_1 \longrightarrow_{\mathcal{R}}^1 Q_2$	Q_1 is reduced in one step to Q_2 w.r.t. one of the rules of $\mathcal{R}$
$\longrightarrow_{\mathcal{R}}^+$	transitive closure of $\longrightarrow_{\mathcal{R}}^1$
$\longrightarrow_{\mathcal{R}}^*$	reflexive, transitive closure of $\longrightarrow_{\mathcal{R}}^1$
$Id(\mathcal{R})$	ideal generated by the polynomials associated with the rules of $\mathcal{R}$
$\mathcal{B}(\mathcal{R})$	set of polynomials associated with the rules of $\mathcal{R}$
$\gg_{\mathcal{R}}$	partial order on the set of power products associated with the set of rules $\mathcal{R}$
$\succ_L$	lexicographic order on the set of power products
$\succ_{TL}$	total degree order refined by the lexicographic order
$\succ_{TR}$	total degree order refined by the reverse lexicographic order
$InPp_{\succ}(P), InPp(P)$	highest power product of P w.r.t. $\succ$
$InC_{\succ}(P), InC(P)$	coefficient of $In_{\succ}(P)$
$\mathcal{R}(B, \succ)$	set of rules associated with the set of polynomials B and the total admissible order $\succ$
$\longleftrightarrow_{\mathcal{R}}^*$	reflexive, symmetric, transitive closure of $\longrightarrow_{\mathcal{R}}^1$
$nf(P), nf_{\mathcal{R}}(P)$	normal form of P w.r.t. $\longrightarrow_{\mathcal{R}}^1$
$nf(B)$	set of the normal forms of the elements of B
$A_{(i_1,\ldots,i_n)}$	$\{k\, x_1^{i_1} \ldots x_n^{i_n} \mid k \in K\}$
$log(T)$	if $T = x_1^{i_1} \ldots x_n^{i_n}$, designs the vector $(i_1, \ldots, i_n)$
$(A^p)_{(i_1,\ldots,i_n)}$	additive subgroups of A^p defining its graduation
$Lv((Q_1, \ldots, Q_p))$	leading vector of the vector $(Q_1, \ldots, Q_p)$ of the R-module A^p.
λ, Λ	homomorphisms from A^p to A
U, H	vectors of A^p
$T(i)$	$InPp_{\succ}(P_i)$
$T(i,j)$	$lcm(InPp_{\succ}(P_i), InPp_{\succ}(P_j))$
$T(i,j,l)$	$lcm(InPp_{\succ}(P_i), InPp_{\succ}(P_j), InPp_{\succ}(P_l))$
$(e_1, \ldots, e_p)$	canonical basis of the A-module A^p
$(s(i,j))_{1 \leq i < j \leq p}$	vectors of A^p, members of the first Taylor basis of the module of syzygies of $(In(P_1), \ldots, In(P_p))$
$S(i,j)$	S-polynomial of P_i and P_j
$(e_{ij})_{1 \leq i < j \leq p}$	canonical basis of the module $A^{p(p-1)/2}$
$S^{(2)}$	second module of syzygies of $(In(P_1), \ldots, In(P_p))$
$(s(i,j,l))_{1 \leq i < j < l \leq p}$	vectors of $A^{p(p-1)/2}$, members of the Taylor basis of $S^{(2)}$

Notations for section 3.

u, u_i, y, z	other variables to define polynomial rings over K
$I : J$	quotient of the ideal I by the ideal J
$Spol(P, Q)$	S-polynomial of the polynomials P and Q
I^e	extension of the ideal I of a ring R' to an ideal of a larger ring R

Notations for section 4.

h, h_i, ϵ	homomorphisms between R-modules
C	complex for the ring R
H	Hilbert function
HP	Hilbert polynomial
d	integer denoting the dimension of an ideal
σ	equivalence relation on C^3
α_i	endomorphism of M_l use to define a graduation

B Listing of the examples.

Arnborg4

Order on the variables : $x \succ y \succ z \succ t$

$B = \{x+y+z+t, xy+yz+zt+tx, xyz+yzt+ztx+txy, xyzt+1\}$

Example listed by Traverso and Donati, [37]. Comes initially from Arnborg and Davenport.

Arnborg5

Order on the variables : $x \succ y \succ z \succ t \succ u$

$B = \{x+y+z+t+u, xy+yz+zt+tu+ux, xyz+yzt+ztu+tux+uxy, xyzt+yztu+ztux+tuxy+uxyz, xyztu+1\}$

Example listed by Traverso and Donati, [37]. comes initially from Arnborg and Davenport.

Butcher

Order on the variables : $a \succ c2 \succ c3 \succ b3 \succ b2 \succ a32 \succ b1 \succ b$

$B = \{b1+b2+b3-a-b, 2b2c2+2b3c3-1-b-2b^2+2ab, 3b2c2^2+3b3c3^2-a-3ab^2+4b+3b^2+3b^3, 6b3a32c2-a-3ab-6ab^2+4b+6b^2+6b^3, 4b2c2^3+4b3c3^3-1-b-10b^2-6b^3-4b^4+4ab+4ab^3, 8b3c3a32c2-1-3b-14b^2-12b^3-8b^4+4ab+4ab^2+8ab^3, 12b3a32c2^2-1-b-14b^2-18b^3-12b^4+8ab+12ab^2+12ab^3, 1+7b+26b^2+36b^3+24b^4-8ab-24ab^2-24ab^3\}$

Example listed by Boege et al, [2]. Comes initially by Butcher and Runge-Kutta.

Katsura4

Order on variables : $u4 \succ u0 \succ u3 \succ u2 \succ u1$

$B = \{u0^2-u0+2u1^2+2u2^2+2u3^2+2u4^2, 2u0u1+2u1u2+2u2u3+2u3u4-u1, 2u0u2+u1^2+2u1u3+2u2u4-u2, 2u0u3+2u1u2+2u1u4-u3, u0+2u1+2u2+2u3+2u4-1\}$

Example listed by Boege et al, [2]. Comes initially from Katsura. Laurent series.

Katsura5

Order on variables : $u5 \succ u4 \succ u0 \succ u3 \succ u2 \succ u1$

$B = \{u0^2-u0+2u1^2+2u2^2+2u3^2+2u4^2+2u5^2, 2u0u1+2u1u2+2u2u3+2u3u4+2u4u5-u1, 2u0u2+u1^2+2u1u3+2u2u4+2u3u5-u2, 2u0u3+2u1u2+2u1u4+2u2u5-u3, 2u0u4+2u1u3+2u1u5+u2^2-u4, u0+2u1+2u2+2u3+2u4+2u5-1\}$

Example listed by Boege et al, [2]. Comes initially from Katsura. Laurent series.

Lazard

Order on variables : $x \succ y \succ z$

$B = \{x^2yz + xy^2z + xyz^2 + xyz + xy + xz + yz, x^2y^2z + xy^2z^2 + x^2yz + xyz + yz + x + z, x^2y^2z^2 + x^2y^2z + xy^2z + xyz + xz + z + 1\}$

Example listed by Traverso and Donati, [37]. Comes initially from Lazard.

Morgenstern

Order on variables : $x \succ y \succ a \succ b \succ c \succ d \succ r \succ s \succ t$

$B = \{xb - ya, xd - d - yc + y, b^2 + a^2 - r^2, c^2 - 2c + 1 + d^2 - s^2, a^2 - 2ac + c^2 + b^2 - 2bd + d^2 - t^2\}$

Example listed by Traverso and Donati, [37]. Comes initially from Morgenstern.

Pavelle4

Order on variables : $x \succ y \succ z \succ t \succ a \succ b \succ c \succ d$

$B = \{xy + xz + xt - a, yx + yz + yt - b, zy + zy + zt - c, tx + ty + tz - d\}$

Example found in Pavelle, [30], page 35. This comes from a problem in chemistry. The variables are renamed to correspond to the paper by Traverso and Donati.

Robbiano

Order on variables : $x \succ y \succ z \succ t$

$B = \{x^31 - x^6 - x - y, x^8 - z, x^10 - t\}$

Example listed by Traverso and Donati, [37]. Comes initially from by Robbiano.

Rose

Order on variables : $u3 \succ u4 \succ a46$

$B = \{7u4^4 - 20a46^2, 2160a46^2u3^4 + 1512a46u3^4 + 315u3^4 - 4000a46^2 - 2800a46 - 490, 67200000a46^5u4^3 + 94080000a46^4u4^3 + 40924800a46^3u4^3 + 2634240a46^2u4^3 - 2300844a46u4^3 - 432180u4^3 + 28800000a46^5u3u4^2 + 40320000a46^6u3u4^2 + 21168000a46^3u3u4^2 + 4939200a46^2u3u4^2 + 347508a46u3u4^2 - 23520000a46^4u3^2u4 - 41395200a46^3u3^2u4 - 26726560a46^2u3^2u4 - 7727104a46u3^2u4 - 852355u3^2u4 - 10080000a46^4u3^3 - 14112000a46^3u3^3 - 7644000a46^2u3^3 - 1978032a46u3^3 - 180075u3^3\}$

Example listed by Boege et al,[2]. Comes initially from Rose. General equilibrium Model, 1984.

Trinks1

Order on variables : $s \succ t \succ z \succ p \succ w \succ b$

$B = \{45p + 35s - 165b - 36, 35p + 40z + 25t - 27s, 15w + 25ps + 30z - 18t - 165b^2, -9w + 15pt + 20zs, wp + 2zt - 11b^3, 99w - 11sb + 3b^2\}$

Example listed by Boege et al,[2]. Comes initially from Trinks. Ideal A, 9/12/1983.

Trinks2

Order on variables : $s \succ t \succ z \succ p \succ w \succ b$

$B = \{45p + 35s - 165b - 36, 35p + 40z + 25t - 27s, 15w + 25ps + 30z - 18t - 165b^2, -9w + 15pt + 20zs, wp + 2zt - 11b^3, 99w - 11sb + 3b^2, 10000b^2 + 6600b + 2673\}$

Example listed by Boege et al,[2]. Comes initially from Trinks 2. Ideal P = A + F7 R, 10/12/1983.

Valla

Order on variables : $x \succ y \succ z \succ t \succ u \succ v \succ w \succ a \succ b \succ c \succ A \succ B \succ C \succ D \succ E \succ F \succ G \succ H \succ I \succ J$

$B = \{-z^2u+2yzv-xv^2-y^2a+xua+A^3, -ztu+ytv+yzw-xvw-y^2b+xub+B^3, -t^2u+2ytw-xw^2-y^2c+xuc+C^3, -ztv+z^2w+yta-xwa-yzb+xvb+D^3, -t^2v+ztw+ytb-xwb-yzc+xvc+E^3, -tv^2+zvw+tua-ywa-zub+yvb+F^3, -tvw+zw^2+tub-ywb-zuc+yvc+G^3, -t^2a+2ztb-xb^2-z^2c+xac+H^3, -twa+tvb+zwb-yb^2-zvc+yac+I^3, -w^2a+2vwb-ub^2-v^2c+uac+J^3\}$

Example listed by Traverso and Donati, [37]. Comes initially from Valla.

C Sample of trace of the program.

The program was run with 5 processors on the very simple example Hairer1 in [2]. This is a very elementary example. It is used here to show how the processors interleave and what kind of trace can be obtained. We choose an intermediary level of trace. For higher levels, the result of each reduction step is printed.

Here is the trace that we obtain :

```
Admissible order : lexicographic.
Number of processors : 5
Dynamic allocation of memory :  standard.


Initially :
Set of polynomials :
Polynomial 1 : c2  - a21
Polynomial 2 : - a31  + c3  - a32
Polynomial 3 : b1  + b2  + b3  -1
Polynomial 4 : 2 c2 b2  + 2 c3 b3  -1
Polynomial 5 : 3 c2^2 b2  + 3 c3^2 b3  -1
Polynomial 6 : 6 c2 a32 b3  -1

Set of pairs :
(1,4)
(1,6)
(4,5)

Reduction of the pair (1,4) by worker 1

Polynomial 7 created by worker 1 : -2 a21 b2  -2 c3 b3  + 1

Set of pairs :
(1,6)
(4,5)

Reduction of the pair (1,6) by worker 1

Reduction of the pair (4,5) by worker 0

Polynomial 8 created by worker 1 : -6 a21 a32 b3  + 1

Set of pairs :
(7,8)

Reduction of the pair (7,8) by worker 1
```

```
Polynomial 9 created by worker 0 : 6 a21 c3 b3  -3 a21  -6 c3^2 b3  + 2

Set of pairs :
(7,9)
(8,9)

Reduction of the pair (7,9) by worker 0

Reduction of the pair (8,9) by worker 2

Polynomial 10 created by worker 1 : -6 c3 a32 b3^2  + 3 a32 b3  - b2

Set of pairs :
(9,10)

Reduction of the pair (9,10) by worker 1

Polynomial 11 created by worker 2 : -3 a21 a32  -6 c3^2 a32 b3  + c3  + 2 a32

Set of pairs :
(8,11)
(7,11)

Reduction of the pair (8,11) by worker 2

Reduction of the pair (7,11) by worker 4

Polynomial 12 created by worker 1 : -6 c3 a32 b3  + 2 c3 b2  + 2 c3 b3  +
4 a32 b3  -1

Set of pairs :
(10,12)
(11,12)

Reduction of the pair (10,12) by worker 1

Reduction of the pair (11,12) by worker 2

Polynomial 13 created by worker 4 : 4 c3^2 b2^2  + 4 c3^2 b2 b3  +
8 c3 a32 b2 b3  -6 c3 a32 b3  -4 c3 b2  -4 a32 b2  + 3 a32

Set of pairs :
(7,13)
(12,13)

Reduction of the pair (7,13) by worker 4
```

```
Reduction of the pair (12,13) by worker 3

Polynomial 14 created by worker 1 : -2 c3 b2 b3  -2 c3 b3^2  -4 a32 b3^2
+ 3 a32 b3  - b2  + b3

Set of pairs :
(12,14)
(9,14)
(13,14)

Reduction of the pair (12,14) by worker 1

Reduction of the pair (9,14) by worker 2

Reduction of the pair (13,14) by worker 3

Polynomial 15 created by worker 1 : 2 c3 b2^2  + 2 c3 b2 b3  + 12 a32^2 b3^2
- 9 a32^2 b3  + 4 a32 b2 b3  + 3 a32 b2  -2 b2

Set of pairs :
(14,15)
(13,15)
(7,15)

Reduction of the pair (14,15) by worker 1

Reduction of the pair (13,15) by worker 2

Reduction of the pair (7,15) by worker 3

Polynomial 16 created by worker 1 : 12 a32^2 b3^3  -9 a32^2 b3^2  +
6 a32 b2 b3  - b2^2  - b2 b3

Set of pairs :
(12,16)
(11,16)

Reduction of the pair (12,16) by worker 1

Reduction of the pair (11,16) by worker 2

Total number of polynomials which entered the basis
      (including the initial ones) : 17
Number of S-polynomials which have been reduced : 21
Approximate number of arithmetical operations on short integers
     during phase 1 : 1291
Number of collisions : 30
```

```
Elapsed time for phase 1 : 407 milliseconds.

Final interreduction
Deletion of polynomial 4
Deletion of polynomial 5
Deletion of polynomial 6
Deletion of polynomial 8
Deletion of polynomial 10
Deletion of polynomial 13


Polynomial 1 : - c2  + a21

Polynomial 2 : a31  - c3  + a32

Polynomial 3 : - b1  - b2  - b3  + 1

Polynomial 7 : -2 a21 b2  -2 c3 b3  + 1

Polynomial 9 : -6 a21 c3 b3  + 3 a21  + 6 c3^2 b3  -2

Polynomial 11 : 9 a21 a32  + 6 c3^2 b2  + 6 c3^2 b3  + 4 c3 b2  + 4 c3 b3
- 6 c3  + 8 a32 b3  -6 a32  -2

Polynomial 12 : 6 c3 a32 b3  -2 c3 b2  -2 c3 b3  -4 a32 b3  + 1

Polynomial 14 : -2 c3 b2 b3  -2 c3 b3^2  -4 a32 b3^2  + 3 a32 b3  - b2  + b3

Polynomial 15 : 2 c3 b2^2  -2 c3 b3^2  + 12 a32^2 b3^2  -9 a32^2 b3  +
4 a32 b2 b3  + 3 a32 b2  -4 a32 b3^2  + 3 a32 b3  -3 b2  + b3

Polynomial 16 : -12 a32^2 b3^3  + 9 a32^2 b3^2  -6 a32 b2 b3  + b2^2
+ b2 b3



Approximate number of arithmetical operations on short integers
during final reduction : 52
Elapsed time during final interreduction : 105 milliseconds.
Waiting for a lock : 0
```

References

[1] Accetta, M., Baron, R., Golub, D., Rashid, R., Tevanian, A., and Young, M. *Mach : A New Kernel Foundation For UNIX Development.* in: **Proceedings Summer 1986 USENIX Technical Conference and Exhibition.** 1986, pp. 93–112.

[2] Boege, W., Gebauer, R., and Kredel, H. *Some Examples for Solving Systems of Algebraic Equations by Calculating Groebner Bases.* **Journal of Symbolic Computation**, vol. 1 (1986), pp. 83–98.

[3] Brinch Hansen, P. **Operating System Principles.** Prentice-Hall, 1973.

[4] Buchberger, B. *Some properties of Grőbner-bases for polynomial ideals.* **ACM SIGSAM Bulletin**, vol. 10 (1976), pp. 19–24.

[5] Buchberger, B. *A theoretical basis for the reduction of polynomials to canonical forms.* **ACM SIGSAM Bulletin**, vol. 10 (1976), pp. 19–29.

[6] Buchberger, B. *An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Polynomial Ideal (German).* Ph.D. Thesis, Univ. Innsbruck, 1965.

[7] Buchberger, B. *A criterion for detecting unnecessary reductions in the construction of Grőbner bases.* in: **Proceedings of EUROSAM 79, Lectures Notes in Computer Science 72.** 1979, pp. 3–21.

[8] Buchberger, B. *Grőbner Bases : An Algorithmic Method In Polynomial Ideal Theory.* in: **Recent Trends In Multidimensional Systems Theory**, edited by N.K.Bose. D.Reidel Publishing Company, 1985, pp. 184–232.

[9] Buchberger, B. *History and Basic Features of the Critical-Pair/Completion Procedure.* **Journal of Symbolic Computation**, vol. 3 (1987), pp. 3–38.

[10] Buchberger, B. *The Parallelization of Critical Pair/Completion Procedures on the L-Machine.* in: **Proceedings of the Japanese Symposium on Functional Programming.** 1987, pp. 54–61.

[11] Bucherger, B. *User's Manual for GROBNER, a Grőbner Bases Package in muMATH.* no. RISC-Linz 87-38, University of Linz (Austria), 1987.

[12] Cooper, E. C. and Draves, R. P. *C Threads.* no. CMU-CS-88-154, Computer Science Department, Carnegie Mellon University, June 1988.

[13] Fulton, W. **Algebraic curves : an introduction to algebraic geometry.** Benjamin, 1969.

[14] Galligo, A., Pottier, L., and Traverso, C. *Greater Easy Common Divisor and standard basis completion algorithms.* in: **ISSAC 88, Lectures Notes in Computer Science.** Springer Verlag, Berlin-Heidelberg-New York, 1988.

[15] Galligo, A. and Traverso, C. *Practical determination of the dimension of an algebraic variety.* in: **Computers and Mathematics 1989.** Springer Verlag, 1989, pp. 46–52.

[16] Gebauer, R. and Möller, M. *On an installation of Buchberger's algorithm.* **Journal of Symbolic Computation**, vol. 6 (1988), pp. 275–286.

[17] Giusti, M. *Combinatorial dimension theory of algebraic varieties.* **Journal of Symbolic Computation**, vol. 6 (1988), pp. 249–265.

[18] Hironaka, H. *Resolution of singularities of an algebraic variety over a field of characteristic 0.* **Ann. Math**, vol. 79 (1964), pp. 109–326.

[19] Huet, G. and Oppen, D. *Equations and rewrite rules : a survey.* in: **Formal Language Theory, Perspectives and Open Problems**. London : Academic Press, 1980, pp. 349–405.

[20] Jacobson, N. **Basic Algebra I, second edition**. Freeman, 1985.

[21] Jacobson, N. **Basic Algebra II**. Freeman, 1980.

[22] Knuth, D. E. and Bendix, P. B. *Simple Word Problems in Universal Algebras.* in: **Proceedings of the Conference on Computational Problems in Abstract Algebra, Oxford 1967**, edited by J. Leech. Pergamon Press, 1970, pp. 263–298.

[23] Kredel, H. and Weispfenning, V. *Computing Dimension and Independent Sets for Polynomial Ideals.* **Journal of Symbolic Computation**, vol. 6 (1988), pp. 231–247.

[24] Macaulay, F. S. *Some properties of enumeration in the theory of modular systems.* **Proc. London Math. Soc.**, vol. 26 (1927), pp. 531–555.

[25] Melenk, H. and Neun, W. *Parallel Polynomial Operations in the Large Buchberger Algorithm.* in: **Computer Algebra and Parallelism, Workshop at the TIM3 Laboratory, University of Grenoble, France**. Academic Press, London, 1988.

[26] Mora, F. and Möller, M. *The computation of the Hilbert function.* in: **EUROCAL 83, Lecture notes on Computer Science**. 1983.

[27] Owicki, S. *Axiomatic Proof Techniques fo Parallel Programs.* Ph.D. Thesis, Cornell University, 1975.

[28] Owicki, S. and Gries, D. *Verifying properties of parallel programs : an axiomatic approach.* **Communications of the ACM**, vol. 19 (1976), pp. 279–285.

[29] Patrizia Gianni, B. T. and Zacharias, G. *Gröbner Bases and Primary Decomposition of Polynomial Ideals.* **Journal of Symbolic Computation**, vol. 6 (1988), pp. 149–167.

[30] Pavelle, R. **Applications of Computer Algebra**. Kluwer Academic Publishers, 1986.

[31] Ponder, C. G. *Evaluation of "Performance Enhancements" in Algebraic Manipulation Systems.* Ph.D. Thesis, University of California, Berkeley, August 1988.

[32] Robbiano, L. *Gröbner Bases : a foundation for Commutative Algebra.* in: **Computers and Mathematics 1989**. 1989.

[33] Senechaud, P. *Implementation of a parallel algorithm to compute a Gröbner basis on boolean polynomials.* in: **Computer Algebra and Parallelism, Workshop at the TIM3 Laboratory, University of Grenoble, France**. Academic Press, London, 1988, pp. 159–166.

[34] Shannon, D. and Sweedler, M. *Using Gröbner bases to determine algebra membership, split surjective algebra homomorphism and determine birational equivalence.* **Journal of Symbolic Computation**, vol. 6 (1988), pp. 267–273.

[35] Siegl, K. *A Parallel Version of Buchberger's Algorithm in STRAND88.* no. RISC-Linz 90-17.0, Johannes Kepler University, Linz (Austria), 1990.

[36] Taylor, D. *Ideals Generated by Monomials in an R-sequence.* Ph.D. Thesis, University of Chicago, 1966.

[37] Traverso, C. and Donati, L. *Experimenting the Gröbner basis algorithm with the AIPI system.* in: **Proceedings ISSAC 1989**. 1989.

[38] Watt, S. M. *Bounded Parallelism in Computer Algebra.* Ph.D. Thesis, University of Waterloo, May 1986.

[39] Winkler, F. *The Church-Rosser Property in Computer Algebra and Special Theorem Proving: An Investigation of Critical Pair/Completion Algorithms.* Ph.D. Thesis, Johannes Kepler Universität Linz, 1984.

[40] Winkler, F. *A p-adic approach to the Computation of Gröbner Bases.* **Journal of Symbolic Computation**, vol. 6 (1988), pp. 287–304.

[41] Zacharias, G. *Generalized Gröbner Bases in Commutative Polynomial Rings.* Bachelor Thesis, Lab. for Computer Science, MIT (Cambridge), 1978.

[42] Zariski, O. and Samuel, P. **Commutative algebra, Vol 2**. D.Van Nostrand Company, Inc., 1960.