Gröbner Basis Methods For Solving Algebraic Equations

Stephen Richard Czapor

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

Research Report CS-89-51

November, 1989

GRÖBNER BASIS METHODS FOR SOLVING ALGEBRAIC EQUATIONS

by

Stephen Richard Czapor

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Applied Mathematics

Waterloo, Ontario, 1988
© S. R. Czapor 1988

Abstract

In this thesis, we examine methods of solving systems of algebraic equations (over rational numbers or rational functions) by computing Gröbner bases for the associated polynomial ideals. We give particular attention to the development of practical improvements to the algorithm of Buchberger, when the lexicographic term ordering is used.

A classical method for solving algebraic equations (using resultants) is described, to provide insight into the nature of nonlinear elimination. Motivated by the problem of polynomial simplification modulo side relations, Buchberger's algorithm for the computation of Gröbner bases (over a field) is described as a basis completion procedure for which the polynomial ring is structured with an ordering of multivariate terms. In this context, the choice of term ordering is insignificant. Therefore, the graduated ordering (with which Gröbner bases are most easily computed, in general) is usually chosen. We see, though, that the methods used to determine the solutions of a system from its Gröbner basis do depend on the choice of ordering; furthermore, for the lexicographic ordering the method is simpler and more widely applicable. We therefore consider (and provide detailed algorithm descriptions for) ways to control the explosive degree and coefficient growth endemic to this ordering. These include careful formulation of the algorithm variant, "criterion" use, S-polynomial selection and reduction strategies, and coefficient arithmetic. Empirical results (using the Maple algebra system) indicate that vast improvement to the performance of the (lexicographic) algorithm is possible. Algorithms are presented which combine the use of multivariate factorization with Buchberger's algorithm. We see that the subsequent computation of a lexicographic Gröbner basis decomposition is generally much more efficient. This technique, along with those above, is incorporated into the construction of a solver (*i.e.*, elimination algorithm). Using the full range and flexibility of the methods at hand, we then discuss the solution of a number of previously intractable problems.

Acknowledgements

I would like to thank my supervisor, Dr. K. O. Geddes, for his guidance and generous financial support throughout my Ph.D. studies. I would also like to acknowledge many fruitful discussions with my colleagues in the Symbolic Computation Group at the University of Waterloo: in particular, Michael Monagan, who suggested the idea of using fraction-free arithmetic and explained the efficient computation of contents, and offered much advice on the Maple implementations used in this thesis; also, Greg Fee, for providing a number of useful test problems and suggestions regarding the use of the Maple system. Finally, I wish to thank the Natural Sciences and Engineering Research Council for financial support in the form of a Postgraduate Scholarship.

Table of Contents

Chapter 1: Introduction 1
1.1 Motivation 1
1.2 History and Related Work 3
1.3 Outline 5

Chapter 2: Classical Elimination 7
2.1 The Theory of Resultants 7
2.2 Resultant Algorithms and Polynomial Remainder Sequences 13
2.3 Solving Algebraic Equations via Resultants 19

Chapter 3: Gröbner Bases 26
3.1 Polynomial Ideals and Standard Bases 26
3.2 Buchberger's Algorithm 33
3.3 The Lexicographic Ordering: Criteria and Selection Strategies 43
3.4 The Reduction Sub-algorithm 55

Chapter 4: Solving Algebraic Equations 71
4.1 Gröbner Bases and Systems of Algebraic Equations 71
4.2 The Use of Multivariate Factorization 84
4.3 A Variant Algorithm 93
4.4 The Construction of a Solver; Further Examples 98

Chapter 5: Conclusions 106

Appendix: List of Test Problems 109

References 126

List of Tables

Table 3.1: standard *vs.* modified use of criteria 48
Table 3.2: comparison of algorithm variants for $<_G$ 52
Table 3.3: algorithm variants and selection strategies for $<_L$ 53
Table 3.4: $<_L$-reduction over rational numbers 62
Table 3.5: $<_G$-reduction over rational numbers 63
Table 3.6: $<_L$-reduction over rational functions 63
Table 3.7: $<_G$-reduction over rational functions 64
Table 3.8: the effect of basis sorting on $<_L$-reduction (integer coeffs.) 68
Table 3.9: the effect of basis sorting on $<_L$-reduction (polynomial coeffs.) 69
Table 4.1: decomposition times for $<_G$-Gröbner bases 76
Table 4.2: decomposition times for $<_L$-Gröbner bases 80
Table 4.3: the effect of factorization in $<_L$-Gröbner bases (integer coeffs.) 91
Table 4.4: the effect of factorization in $<_L$-Gröbner bases (polynomial coeffs.) 92
Table 4.5: a hybrid decomposition algorithm 96

List of Algorithms

Algorithm 3.1: reduction (sub-) algorithm (*reduce*) 31
Algorithm 3.2: simple variant of Buchberger's algorithm (*Gbasis*) 35
Algorithm 3.3: inter-reduction of basis (*reduceset*) 40
Algorithm 3.4: Buchberger's algorithm with criteria (*Gbasis*) 41
Algorithm 3.5: normal selection strategy (*normselect*) 45
Algorithm 3.6: selection with exhaustive *criterion1* (*select1*) 46
Algorithm 3.7: selection with exhaustive *criterion2* (*select2*) 47
Algorithm 3.8: Buchberger's algorithm with modified normal selection (*Gbasis*) 47
Algorithm 3.9: Buchberger's algorithm with full basis inter-reduction (*Gbasis*) 50
Algorithm 3.10: heuristic selection strategy (*select3*) 51
Algorithm 3.11: reduction over fraction field (*reduce*) 55
Algorithm 3.12: probablistic content computation (*content*) 58
Algorithm 3.13: computation of test divisors by division (*extend1*) 59
Algorithm 3.14: computation of test divisors by factorization (*extend2*) 59
Algorithm 3.15: fraction-free reduction (*reduce*) 60
Algorithm 3.16: fraction-free reduction (*reduce*) 61
Algorithm 3.17: fraction-free reduction with trial divisions (*reduce*) 61
Algorithm 3.18: fraction-free reduction with trial divisions (*reduce*) 62
Algorithm 4.1: construction of univariate polynomials (*finduni*) 73
Algorithm 4.2: decomposition of $<_G$-basis (*decompG*) 74
Algorithm 4.3: system solution via $<_G$-bases (*solveG*) 75
Algorithm 4.4: decomposition of $<_L$-basis (*decompL*) 77
Algorithm 4.5: system solution via $<_L$-bases (*solveL*) 78
Algorithm 4.6: Buchberger's algorithm with factorization (*partdecomp*) 86
Algorithm 4.7: improved use of factorization (*partdecomp*) 88
Algorithm 4.8: hybrid decomposition algorithm (*hdecomp*) 94
Algorithm 4.9: input pre-processing (*subdivide*) 99
Algorithm 4.10: improved system solution via $<_L$-bases (*Gsolve*) 101

Chapter 1: Introduction

1.1. Motivation

In this thesis, we will consider a number of related methods for determining the exact solutions of a system of algebraic equations over a field $\boldsymbol{K}$. We note that such a system, say

$$\begin{aligned} f_1(x_1, x_2, ..., x_n) &= 0 , \\ f_2(x_1, x_2, ..., x_n) &= 0 , \\ &\vdots \\ f_k(x_1, x_2, ..., x_n) &= 0 , \end{aligned}$$

may also be viewed as a set of polynomials $F = \{f_1, ..., f_k\}$, *i.e.* a finite subset of $\boldsymbol{K}[x_1,...,x_n]$ (the n-variate polynomial ring over $\boldsymbol{K}$). Recall that a *solution* of F is an n-tuple $(\alpha_1, \alpha_2, ..., \alpha_n)$ of elements from a suitably chosen extension field of $\boldsymbol{K}$ such that

$$f_1(\alpha_1, ..., \alpha_n) = ... = f_k(\alpha_1, ..., \alpha_n) = 0 .$$

For various reasons (*e.g.* the unsolvability of univariate equations of degree higher than four), it is not always possible to obtain the solutions (or *roots*) of F in explicit form. We can, however, obtain various representations of these solutions from which explicit roots may often be derived. We will concern ourselves primarily with the problem of obtaining these representations (some of which are more useful than others), and not with the subsequent matter of their exploitation.

The algorithms we discuss are valid for any coefficient field in which arithmetic may be performed effectively. However, we will restrict our considerations to the types of equations which are (arguably) most mathematically commonplace: namely, those with integer coefficients and those with integer coefficients which also contain symbolic (or "free") parameters. Typical examples might be systems in variables x, y such as

$$F_1 := \{x^2 + xy^3 - 9,\ 3x^2y - y^3 - 4\} \tag{1.1}$$

or

$$F_2 := \{ax^2 + bxy + c,\ by^2 - cxy + d\}\,. \tag{1.2}$$

Clearly, any equation with rational number or rational function coefficients may be re-scaled to remove the fractions. Since the solution domains must involve fractions, though, we will consider $\boldsymbol{Q}$ and $\boldsymbol{Q}(a,b,...,d)$ (*i.e.* the set of rational functions in parameters $a, ..., d$) to be the fundamental domains in these respective cases. Obviously, other types of equations which may be reduced to these forms by substitutions or other rearrangements may also be considered.

The need for exact (rather than numerical) solutions is clear in the case of the system (1.2) above, since we cannot find solutions in terms of $a, ..., d$ by numerical means. It can also happen that a system (with or without free parameters) has *infinitely* many solutions; that is, the solutions can only be represented by solving for a *subset* of the original indeterminates, while considering the others as parameters. Even when there exist only finitely many solutions, numerical methods do not guarantee that *all* solutions will be found, or offer a proof of inconsistency when none can be found. For example, it is difficult to determine all real roots of the system (1.1) by fixed-point iteration, due to the proximity of some of these roots. Numerical difficulties, such as those arising from ill-conditioned input, are another possibility. Finally, it may sometimes be crucial to know that a solution value of a certain variable is, say, $\sqrt{2}$.

1.2. History and Related Work

The problem of solving algebraic equations is, of course, the fundamental issue of classical algebra. Up to the end of the eighteenth century, progress was more substantial on the solution of linear systems and of univariate equations of low degree. However, the basic ideas of nonlinear elimination seem to have been known to Euler and Bezout during this time. The elimination theory based on polynomial resultants was developed by Sylvester during the mid-nineteenth century. Historically speaking, this was well into the "modern" era of algebra. However, to distinguish this work from the more modern approach of Hilbert, we refer to the resultant technique as the "classical" method.

In the work of Hilbert, an algebraic system F was treated as a basis (*i.e.* a generating set) of a polynomial ideal, and the set of zeros as the algebraic manifold of the ideal. It is clear that systems which generate the same ideal have identical roots. However, this observation alone did not provide a new approach to finding these roots, since (until recently) there were few *constructive* results in polynomial ideal theory other than those of [Herm26]. In [Buch65], Buchberger not only presented the notion of Gröbner bases of polynomial ideals (named after his supervisor, Prof. W. Gröbner), but also an algorithm for their computation. He thus provided algorithmic solutions to a wide variety of problems in polynomial ideal theory, including that of solving algebraic systems. Since then, many papers on various aspects of Gröbner bases (and Buchberger's algorithm) have appeared. (See [Buch85] for an extensive bibliography.) However, most of the subsequent improvements to the algorithm have been carried out by Buchberger and various co-workers. Other workers have concentrated chiefly upon extending it to new domains, and upon new applications.

We mention also a number of other recent methods, although we will not attempt to describe them here. In [Laza79] and [Laza81], Lazard describes an approach based on Gaussian elimination in certain matrices, which is in a sense related to Buchberger's algorithm. A method for determining the real roots of a system based on the *cylindrical algebraic decomposition* technique of [Coll75] was first implemented in [Arno84]; see also [Dave85] for a description of the method. More recently, a geometric method based on the computation of "affine compounds" (which depends on the earlier algorithms of Lazard) was presented in [Chis83]. Other than the preliminary investigations of [Savi87], no implementations of this algorithm are known at this time.

In the Maple system for computer algebra [Char83], a solver based on a heuristic/substitution approach [Gonn86] has succeeded in solving several large systems such as Problems 16, 17 of the Appendix. The relative efficiency of this scheme when applied to such problems (*i.e.* those whose structure it can readily exploit) serves to illustrate the potential importance of heuristic techniques in the current problem.

1.3. Outline

We have already briefly discussed the nature and origins of several exact methods for solving algebraic equations. In the succeeding chapters, we will further motivate, develop, and refine the Gröbner basis methods of Buchberger. Knowledge of basic algebraic and computational concepts is assumed throughout; the reader is referred to [Waer70], [Waer53], [Gedd83] for more information on the theory and algorithms of algebraic computation.

In Chapter 2, we discuss various aspects of the resultant method of Sylvester; this provides a useful background against which to develop the more modern methods. We first present the fundamental properties of polynomial resultants. Recent improvements to algorithms for the computation of resultants are then discussed, with particular attention to the role of division processes. We see, for example, that through the use of pseudo-division the integral domains Z and $Z[a_1,...,a_m]$ need not be embedded in their respective quotient fields. Finally, the use of resultants in solving algebraic systems by elimination is described. We see that although the computation of a single resultant may be very efficient, the corresponding elimination algorithm will exhibit serious limitations.

In Chapter 3 we develop (in an independent setting) some of the theory of Gröbner (or *standard*) bases for polynomial ideals, as well as Buchberger's algorithm. We include some known complexity results, although study of this aspect of the algorithm is extremely difficult and few such results exist. The practical behaviour of the algorithm depends strongly on the type of structure imposed on the polynomial ring, namely the ordering of multivariate terms. Typically, either the so-called *graduated* or *lexicographic* orderings are chosen. A number of studies have found (*e.g.* [Buch85], [Kapu86], [Boge86]) that Gröbner bases with respect to the former term ordering are usually much easier to compute in practice. However, it turns out that the latter term ordering has properties which make it invaluable for our present purpose of solving algebraic systems. We therefore formulate a number of improvements to the algorithm when using the lexicographic ordering. Many of these are suggested by analogies with the material of Chapter 2. For example, even though we work intrinsically over a (fraction) field, it is more practical to avoid explicit fractions (*e.g.*, work over the sub-ring Z instead of Q). The cumulative effect of our improvements is that, for many practical examples, the comparison against the graduated ordering becomes much more favorable.

In Chapter 4, we present Buchberger's methods for solving algebraic equations by computing Gröbner bases, both for the graduated and the lexicographic term orderings. Since these require that a permutation of the variables be fixed (to establish an ordering of terms), we describe also a heuristic of [Boge86] for choosing the permutation. We then explore the use of multivariate factorization in the lexicographic algorithm. This, once again, is suggested by analogy with the resultant approach. We show that such an approach readily lends itself to further improvements, and new variants of the algorithm. Lastly, we discuss the formulation of a "solver" based upon the lexicographic elimination scheme. Some simple supporting algorithms for pre-processing raw input are given.

In Chapter 5, we briefly summarize our results, and discuss some possible directions for future research.

Throughout this work, our conclusions are supported by extensive empirical evidence. Clearly, such results depend on the nature of the implementations used to gather the data. All of our implementations have been done in the Maple system ([Char83], [Char86b]), a general purpose computer algebra system (and programming environment) which has been under development at the University of Waterloo for about eight years. Since all of our algorithms have been stated in an Algol-like pseudo-code to a reasonable degree of detail, the corresponding Maple code will not be presented here. Every effort has been made to ensure that direct comparisons have been conducted on a fair basis. (For example, common code is used for common sub-algorithms whenever appropriate.) A fairly broad set of test problems is used, which is listed in the Appendix. While only a few of these were previously unsolved, many were previously intractable by Gröbner basis methods (or without considerable interaction, via other methods). We have deliberately avoided consideration of trivial problems, from which no significant insights may be obtained. We have also tried to limit our use of contrived problems, since systems which arise in practice often possess structural attributes which are difficult to simulate.

Chapter 2: Classical Elimination

2.1. The Theory of Resultants

In this section, we provide a brief introduction to the basic tool of classical elimination theory, namely the polynomial resultant. Resultants were devised during the last century in connection with the polynomial greatest common divisor (GCD) problem. Accordingly, there now exists a considerable body of theory on the resultant (only some of which will be mentioned here). Resultants have remained an important mathematical tool, with applications in algebraic elimination [Waer53], quantifier elimination [Coll83], and computing in algebraic extensions [Loos83b] to name a few. Hence, resultant algorithms, too, have undergone considerable development in recent years. The relationship between the classical elimination method for solving algebraic equations (based on resultant theory) and the more general ideal-theoretic methods of the succeeding chapters is not yet completely understood. However, we will see that many analogies are possible which allow worthwhile practical improvements to the less well-developed Gröbner basis algorithms. Although we are most concerned with the domain of polynomials over a field, many of the classical results readily generalize to other domains. We will therefore endeavor to state these results in the most general possible form.

Let $\boldsymbol{R}$ be a commutative ring with identity, and let $f, g \in \boldsymbol{R}[x]$ be non-zero polynomials such that $f = \sum_{i=0}^{m} a_i x^i$ and $g = \sum_{i=0}^{n} b_i x^i$.

Definition: The *Sylvester matrix* of f, g is the $m+n$ by $m+n$ matrix

$$M = \begin{pmatrix} a_m & a_{m-1} & \cdots & a_1 & a_0 & & \\ & a_m & a_{m-1} & \cdots & a_1 & a_0 & \\ & & \cdots & \cdots & \cdots & \cdots & \\ & & & a_m & \cdots & \cdots & a_0 \\ b_n & b_{n-1} & \cdots & b_1 & b_0 & & \\ & b_n & b_{n-1} & \cdots & b_1 & b_0 & \\ & & \cdots & \cdots & \cdots & \cdots & \\ & & & b_n & \cdots & \cdots & b_0 \end{pmatrix} \tag{2.1}$$

where the upper part of the matrix consists of n rows of f coefficients, the lower part consists of m rows of g coefficients, and the entries not shown are zero.

Definition: The *resultant* of f, g (written $res(f, g)$) is the determinant of the Sylvester matrix (2.1). We also have

$$res(a,b) := 1, \quad a,b \in R ,$$
$$res(a,g) := a^n, \quad a \in R .$$

We note the obvious but important fact that $res(f, g) \in R$; that is, it does not contain the indeterminate x.

Example: For the polynomials

$$f = 3yx^2 - y^3 - 4 , \quad g = x^2 + y^3x - 9 ,$$

considered as elements of $Z[y][x] := (Z[y])[x]$, we compute

$$res(f,g) = det \begin{pmatrix} 3y & 0 & -y^3-4 & 0 \\ 0 & 3y & 0 & -y^3-4 \\ 1 & y^3 & -9 & 0 \\ 0 & 1 & y^3 & -9 \end{pmatrix}$$

$$= -3y^{10} - 12y^7 + y^6 - 54y^4 + 8y^3 + 729y^2 - 216y + 16 .$$

Obviously, resultants will be cumbersome to compute by hand except in very simple cases. It is little wonder, then, that the practical limitations of the elimination theory were not well understood until the advent of symbolic computation. We now develop the relationship between resultants and GCD's which is fundamental to resultant theory.

Lemma 2.1: Consider polynomials $f,\ g \in \boldsymbol{R}[x]$ (where $\boldsymbol{R}$ is an integral domain) of degrees $m,\ n$ respectively:

$$f(x) \ = \ a_m x^m + \ldots + a_0 \ ,$$
$$g(x) \ = \ b_n x^n + \ldots + b_0 \ ,$$

where $a_m \neq 0 \neq b_n$. Then $\exists\ \alpha,\ \beta \in \boldsymbol{R}$ such that αf and βg have a non-constant common divisor iff $\exists$ nonzero polynomials $p(x)$, $q(x)$ such that

$$degree(p) < n \ , \quad degree(q) < m \ ,$$

and

$$p(x)f(x) \ = \ q(x)g(x) \ . \tag{2.2}$$

Proof: If such a divisor $h(x)$ exists (with $degree(h) \geq 1$) we set

$$p \ = \ \frac{\alpha\beta g}{h} \ , \ q \ = \ \frac{\alpha\beta f}{h} \ .$$

Conversely, if $\gamma = gcd(f,\ g) \in \boldsymbol{R}$, we have

$$\gamma p \ = \ gcd(pf,\ pg) \ = \ gcd(qg,\ pg) \ .$$

However, the degree of the quantity on the right hand side is at least n, which contradicts the assumption that $degree(p) < n$ □

Note that in generalizing this result from the form stated in [Waer70] (*i.e.* from a field to an integral domain), we essentially end up working over the fraction field of $\boldsymbol{R}$; given a field, the proof is valid with $\alpha = \beta = 1$.

Theorem 2.2 ([Waer70]): The polynomials $f,\ g$ defined in Lemma 2.1 have a non-constant common divisor (when working over the quotient field) iff $res(f,\ g) = 0$.

Proof: By Lemma 2.1, we require the existence of $p,\ q$ as defined in (2.2). Let us write

$$p \ = \ \sum_{i=0}^{n-1} c_i x^i \ , \quad q \ = \ \sum_{i=0}^{m-1} d_i x^i \ .$$

Then the condition (2.2) is equivalent to $m+n$ linear, homogeneous equations (in the indeterminates c_i, d_i) with coefficient matrix (2.1). Neither p nor q may vanish identically, since (for example)

$$p(x) = 0 \quad \Rightarrow \quad q(x)g(x) = 0 \quad \Rightarrow \quad q(x) = 0 \ .$$

The system above has a nontrivial solution iff its determinant (*i.e.*, $res(f, g)$) vanishes. Note also that $res(f, g) = 0$ if, contrary to assumption, either f or g vanish identically □

The connection between resultants and the (extended) Euclidean algorithm is suggested by the next identity.

Theorem 2.3 ([Waer70]): Let $f, g \in \boldsymbol{R}[x]$ be polynomials of degree m, $n > 0$, respectively. Then $\exists$ polynomials $h, k \in \boldsymbol{R}[x]$ with $degree(h) < n$, $degree(k) < m$ such that

$$h f \ + \ k g \ = \ res(f, g) \ . \tag{2.3}$$

Proof: For f, g (with coefficients a_i, b_i, respectively), we form the equations

$$\begin{aligned}
x^{n-1}f &= a_m x^{m+n-1} + a_{m-1}x^{m+n-2} + \ldots + a_0 x^{n-1} \ , \\
x^{n-2}f &= a_m x^{m+n-2} + \ldots + a_0 x^{n-2} \ , \\
&\ldots \\
f &= a_m x^m + \ldots + a_0 \ , \\
x^{m-1}g &= b_n x^{m+n-1} + b_{n-1}x^{m+n-2} + \ldots + b_0 x^{m-1} \ , \\
&\ldots \\
g &= b_n x^n + \ldots + b_0 \ .
\end{aligned}$$

If we write this system in matrix form, we note that its determinant is $res(f, g)$. We then obtain a relation of the form (2.3) by solving for the last unknown (*i.e.*, "1") by Cramer's rule □

The relationship shown above is fundamental, and suggests a slightly more general approach.

Definition ([Roth84]): Let f, g be polynomials in $\boldsymbol{R}[x]$ where $\boldsymbol{R}$ is an integral domain. Then $p \in \boldsymbol{R}$ is a *pseudo-resultant* of f, g if $\exists$ polynomials h, $k \in \boldsymbol{R}[x]$ such that

$$degree(h) < degree(g) , \tag{2.4a}$$

$$degree(k) < degree(f) , \tag{2.4b}$$

$$h f + k g = p . \tag{2.4c}$$

It turns out that if $\boldsymbol{R}$ is a unique factorization domain, the set of pseudo-resultants of a fixed f, g is a principal ideal. (See Chapter 3 for the definition.) Hence, there is a "minimal resultant" of f and g, namely the generator of the ideal. While such results are of theoretical interest, the conjecture that this minimal resultant will be easier to compute in practice than the resultant has not (except, perhaps, in the case $\boldsymbol{R} = \boldsymbol{Z}$) been supported.

Finally, we show that the resultant of f, g may be written as a symmetric function of the roots of f, g.

Theorem 2.4 ([Loos83b]): Let $f(x) = a_m \prod_{i=1}^{m}(x-\alpha_i)$ and $g(x) = b_n \prod_{i=1}^{n}(x-\beta_i)$ be polynomials over an integral domain $\boldsymbol{R}$ with indeterminates α_i, β_i. Then

$$res(f, g) = a_m^n \prod_{i=1}^{m} g(\alpha_i) , \tag{2.5a}$$

$$res(f, g) = (-1)^{mn}\, b_n^m \prod_{i=1}^{n} f(\beta_i) , \tag{2.5b}$$

$$res(f, g) = a_m^n\, b_n^m \prod_{i=1}^{m}\prod_{j=1}^{n}(\alpha_i-\beta_j) . \tag{2.5c}$$

Proof: See [Waer70] or [Loos83b] for the details, which do not require that the indeterminates α_i, β_i actually be the roots of f, g □

Of course, the above result is of greatest interest to us when the roots *are* substituted for the indeterminates.

2.2. Resultant Algorithms and Polynomial Remainder Sequences

Since the advent of symbolic computation roughly thirty years ago, there has been a resurgence of interest in the computation of polynomial resultants. This has been motivated primarily by their use in the solution of systems of algebraic equations (which we discuss in the sequel). Also, progress has been hastened by their close connection with the more fundamental polynomial GCD problem. For practical reasons, one usually re-scales equations (or polynomials) with rational coefficients to remove fractions. Accordingly, previous research has concentrated chiefly on the polynomial domain $Z[x_1,...,x_s]$, which requires a more complex approach than that of polynomials over a field. Since one may view such polynomials alternately as elements of

$$R[x_s] \;:=\; Z[x_1,...,x_{s-1},x_{s+1},...,x_r][x_s]$$

for $1 \leq s \leq r$, we will sometimes write $degree(f;\, x_s)$ or $res(f,g;\, x_s)$ to avoid ambiguity.

Algorithms for computing resultants are of two basic types: those based on evaluation of determinants, and those based on polynomial division. The former class obviously includes the very simple approach of directly evaluating Sylvester's determinant (*e.g.* by minor expansion). However, a superior approach involves the construction of a condensed quantity known as *Bezout's determinant*. Given p_1, $p_2 \in R[x]$ of degrees m, n respectively with $m \geq n$, this will be a determinant of order m formed from the coefficients of p_1, p_2 in such a way as to exploit the structure of the matrix (2.1). It is equivalent to Sylvester's determinant in the sense that by evaluating either we obtain $res(p_1, p_2)$. Note, however, that Sylvester's determinant may be as large as order $2m$. A description of the procedure for constructing and evaluating Bezout's determinant may be found in [KuAd69].

Since division processes themselves are of interest in the present setting, we will examine them (and the corresponding resultant algorithms) in somewhat greater detail. We use the terminology of [Loos83a], which differs from that of [Coll67] and others. Let K be a field and p_1, $p_2 \in K[x]$ be nonzero polynomials of degrees m, n respectively, with $m \geq n$. We recall that Euclid's algorithm allows us to compute the GCD of p_1, p_2 by constructing a sequence such that

$$p_i \;=\; q_i p_{i+1} + p_{i+2} \tag{2.6}$$

where $degree(p_{i+2}) < degree(p_{i+1})$ for $1 \leq i \leq k-2$ and $p_{k+1} = 0$. Here the quantity p_{i+2} is just the remainder upon dividing p_i by p_{i+1}. The sequence $p_1, \ldots, p_k$ is therefore known as a *polynomial remainder sequence* (PRS) for p_1, p_2, which ends with $p_k = gcd(p_1,p_2)$. Of course, the domain $\boldsymbol{Z}[y,\ldots,z]$ is not a field and requires a different treatment. One approach is to embed the above [integral] domain in its quotient field, and proceed as in (2.6). We refer to this as the *Euclidean PRS* for p_1, p_2. Unfortunately, the cost of performing arithmetic over $\boldsymbol{Q}(y,\ldots,z)$ is much greater than that over $\boldsymbol{Z}[y,\ldots,z]$. This is made worse by the fact that the coefficients grow in size with each step of the process. Some improvement is possible by making each remainder monic before the next division step. That is, we divide through by the leading coefficient $lcoeff(p_{i+2})$. This corresponds to a sequence

$$p_i = q_i p_{i+1} + \beta_i p_{i+2} , \tag{2.7a}$$

$$\beta_i = lcoeff(p_i - q_i p_{i+1}) , \tag{2.7b}$$

which we call the *monic Euclidean PRS*. One effect of the re-scaling is that the new coefficients (after removing GCD's) are typically much smaller (*i.e.* coefficient growth is minimized). Also, the division itself is simpler since the leading coefficient of the divisor is 1. (See also Section 3.4 for some details in a slightly different setting.)

Another (more practical) approach, when working over an integral domain $\boldsymbol{R}$ which is not a field, involves use of a modified *pseudo-division* process. Namely, if $p_1 = \sum_{i=0}^{m} a_i x^i$, $p_2 = \sum_{i=0}^{n} b_i x^i$ where $m \geq n$, then a division step with p_2 only requires that the dividend be divisible by b_n at least $m-n+1$ times. We can therefore construct a remainder sequence

$$\alpha_i p_i = q_i p_{i+1} + p_{i+2} , \tag{2.8a}$$

$$\alpha_i = (lcoeff(p_{i+1}))^l , \quad l = degree(p_i) - degree(p_{i+1}) + 1 \tag{2.8b}$$

(where each quotient is constructed by exact divisions over $\boldsymbol{R}$), which we call the *pseudo PRS*. Although such arithmetic is much simpler than that over the quotient field, this particular PRS is thoroughly unsuitable in practice; it can be shown (see [Knut69], Vol. 2) that the coefficients of each pseudo-remainder in (2.8) grow exponentially. This can be alleviated if we divide out common divisors of the coefficients of each pseudo-remainder. Since $\boldsymbol{Z}[y,\ldots,z]$ is a unique factorization domain, we have the following.

Definition: A polynomial $f \in R[x]$ (where R is a unique factorization domain) is called *primitive* if it is either 0 or its coefficients are relatively prime. If $f = \sum_{i=0}^{m} a_i x^i$, we further define the *content* and *primitive part* as

$$content(f) := gcd(a_0, ..., a_m) = gcd(gcd(a_0, ..., a_{m-1}), a_m), \tag{2.9}$$

$$pp(f) := f/content(f), \tag{2.10}$$

where we adopt the convention that the leading coefficient of the primitive part is unit normal (*e.g.* positive, in the case of integers).

By Gauss' Lemma (*i.e.*, the product of primitive polynomials is primitive), we can then define the *primitive PRS*

$$\alpha_i p_i = q_i p_{i+1} + \beta_i p_{i+2}, \tag{2.11a}$$

$$\alpha_i = (lcoeff(p_{i+1}))^l, \quad l = degree(p_i) - degree(p_{i+1}) + 1, \tag{2.11b}$$

$$\beta_i = content(\alpha_i p_i - q_i p_{i+1}), \tag{2.11c}$$

which (applied to primitive p_1, p_2) yields $p_k = gcd(p_1, p_2)$. Although such a sequence requires many GCD sub-computations (over the coefficient domain R), it minimizes coefficient growth through the sequence while avoiding fractional arithmetic. According to Knuth ([Knut69], Vol. 2), the primitive PRS is therefore generally superior in practice to the monic Euclidean PRS. In [Hear79], a further improvement was proposed by Hearn. He suggests that, instead of formally computing the content in (2.11c), much of it can be removed by trial divisions with the set

$$\{lcoeff(p_1), lcoeff(p_2), ..., lcoeff(p_{i+1})\}.$$

That is, we may build up the factor β_i (which is removed from the raw pseudo-remainder) from these coefficients to the extent that they divide the pseudo-remainder exactly. Following this, the left-over content can be computed by (2.9) more cheaply, since it involves the GCD of smaller coefficients.

Finally, we mention a pair of sequences discovered by Collins ([Coll67]), which have been the most widely used in computer algebra systems. The *reduced PRS* is defined by

$$\alpha_i p_i = q_i p_{i+1} + \beta_i p_{i+2} , \tag{2.12a}$$

$$\alpha_i = (lcoeff(p_{i+1}))^l , \quad l = degree(p_i) - degree(p_{i+1}) + 1 , \tag{2.12b}$$

$$\beta_1 = 1 , \quad \beta_{i+1} = \alpha_i , \quad 1 \leq i \leq k-1 . \tag{2.12c}$$

This sequence removes a portion of the content of each pseudo-remainder which can be obtained without GCD sub-computations. It is usually better in practice to avoid content computations and work with slightly bigger coefficients. So, the reduced PRS is much faster than the primitive PRS for all but exceptional cases. A related (but much more complicated) sequence, the *subresultant PRS*, computes a quantity which is at least as large as (2.12c) for very little additional cost. Derivation of these sequences is quite complicated; the details may be found in [Coll67], [Brow71], [Brow78].

We now return to the question of how resultants may be computed by polynomial division. This is made possible by the following result.

Theorem 2.5 ([Loos83b]): Let $\boldsymbol{R}$ be an integral domain and $p_1, p_2 \in \boldsymbol{R}[x]$ be polynomials such that $degree(p_1) = m$, $degree(p_2) = n$. Further suppose that $lcoeff(p_1) = a_m$ and $degree(p_1 q + p_2) = l$. Then

$$res(p_1, p_1 q + p_2) = a_m^{l-n} \, res(p_1, p_2) . \tag{2.13}$$

Proof: Let α_i be the roots of p_1. Then using equation (2.5a) we have

$$\begin{aligned} res(p_1, p_2) = a_m^n \prod_{i=1}^{m} p_2(\alpha_i) &= a_m^n \prod_{i=1}^{m} (p_1(\alpha_i) q(\alpha_i) + p_2(\alpha_i)) \\ &= a_m^{n-l} \, res(p_1, p_1 q + p_2) \; \square \end{aligned}$$

Suppose that $gcd(p_1, p_2) \in \boldsymbol{R}$. If we apply Theorem 2.5, along with the observations that

$$res(p_1, p_2) = (-1)^{mn} res(p_2, p_1)$$

and

$$res(p_{k-1}, p_k) = lcoeff(p_k)^{d_{k-1}}$$

(where $d_i := degree(p_i)$, and $d_k = 0$) to the polynomial remainder sequence (2.6), we obtain

$$res(p_1,p_2) = lcoeff(p_k)^{d_{k-1}} \prod_{i=1}^{k-2} (-1)^{d_i d_{i+1}} lcoeff(p_{i+1})^{d_i - d_{i+2}} . \qquad (2.14)$$

A similar expression for the reduced PRS may be found in [Coll67].

The last resultant algorithm worthy of note is the modular method of [Coll71], which uses homomorphisms and the Chinese Remainder Theorem [Gedd83] to compute multivariate polynomial resultants over $\boldsymbol{Z}$. We recall that for commutative rings with identities $\boldsymbol{R}$, $\tilde{\boldsymbol{R}}$, a homomorphism ϕ of $\boldsymbol{R}$ into $\tilde{\boldsymbol{R}}$ induces a homomorphism of $\boldsymbol{R}[x]$ into $\tilde{\boldsymbol{R}}[x]$ by

$$\phi(\sum_{i=0}^{m} a_i x^i) := \sum_{i=0}^{m} \phi(a_i) x^i .$$

We then have the following result, proved in [Coll71].

Theorem 2.6: Let p_1, p_2 be as defined in Theorem 2.5, and let ϕ be a homomorphism as stated above. Then if $degree(\phi(p_1)) = m$ and $degree(\phi(p_2)) = k$, $0 \leq k \leq n$, we have

$$\phi(\ res(p_1,p_2)\ ) = \phi(a_m)^{n-k} res(\phi(p_1),\phi(p_2)) . \qquad (2.15)$$

First, Collins' algorithm makes use of *modular* homomorphisms of the type ϕ_m: $\boldsymbol{Z} \rightarrow \boldsymbol{Z}_m$ defined as the remainder upon division by m. Using the Chinese Remainder Theorem, we can compute $res(p_1,p_2)$ if $\phi_{m_i}(res(p_1,p_2))$ is known for sufficiently many prime moduli m_i. This also requires a bound on the coefficients of the resultant, which is easily obtained. For $q = \sum_{i=0}^{l} c_i(x_1,...,x_{s-1}) x_s^i \in \boldsymbol{Z}[x_1,...,x_s]$ we define

$$\|q\|_+ := \begin{cases} |\,q\,| & , q \in \boldsymbol{Z}\ (s=0) \\ \sum_{i=0}^{l} \|c_i\|_+ & , s \geq 1 . \end{cases}$$

Then if $A := max\{\|a_i\|_+ \mid 0 \leq i \leq m\}$ and $B := max\{\|b_i\|_+ \mid 0 \leq i \leq n\}$ we have

$$\|res(p_1,p_2; x_s)\|_+ \leq (m+n)!\, A^n B^m \ . \tag{2.16}$$

Next, the algorithm uses *evaluation* homomorphisms of the form

$$\psi_{(a_1,\ldots,a_s)}\colon \boldsymbol{Z}_m[x_1,\ldots,x_s] \longrightarrow \boldsymbol{Z}_m$$

and the natural isomorphism

$$\boldsymbol{R}[x_1,\ldots,x_r] \ \tilde{=}\ \boldsymbol{R}[x_1,\ldots,x_{r-1}][x_r]$$

to reduce a problem in $\boldsymbol{Z}_m[x_1,\ldots,x_r]$ to one in $\boldsymbol{Z}_m[x_r]$. In the latter (Euclidean) domain, the resultant can easily be computed via the simple PRS (2.6) using the formula (2.14). This result is then lifted back to $\boldsymbol{Z}_m[x_1,\ldots,x_r]$ by interpolation (see [Gedd83]), using also the degree bound

$$\begin{aligned} degree(res(p_1,p_2; x_r); x_s) \leq\ & degree(p_1; x_s)\, degree(p_2; x_r) \\ & + degree(p_2; x_s)\, degree(p_1; x_r) \ , \end{aligned} \tag{2.17}$$

for $1 \leq s \leq r$.

The various resultant algorithms have been analyzed and compared in [KuAd69], [Coll67], and [Coll71]. If the number of variables is considered to be fixed, then the PRS and modular algorithms require only polynomial time (with the latter having the lower worst-case complexity). In contrast, [Coll69] shows that the worst-case behavior of the Bezout method is exponential. Indeed, for sufficiently dense and large problems the modular method is clearly superior in practice as well. However, the determinant methods might still be preferable on very sparse problems. Similarly, the PRS methods would perform well on univariate problems, or those in which the polynomials involved contain any variables of low degree.

2.3. Solving Algebraic Equations via Resultants

The solution of systems of algebraic equations by nonlinear elimination has been known since the last century. Early computer implementations such as [Will62], [Mose66] marked a renewed interest in the method, and began exploration of its practical limits. Later, the implementation of [Yun73] (which comprises the "algsys" sub-system of Macsyma [Mart71]) made use of improved algorithms for the computation of resultants, polynomial GCD's and polynomial factorization. In spite of such improvements, the practical limits of the method remain modest. This is due in part to the intrinsic difficulty of the problem, and in part to the limitations imposed by the use of resultants to achieve the elimination. In this section we describe this (essentially) classical method, before presenting the Gröbner basis methods (with which it is closely related) in the succeeding chapters. We will not attempt the lengthy task of completely specifying an algorithm. Rather, we describe the basic ideas of the process with the goal of clarifying (and providing a contrast to) the latter methods.

The basis of the method is the following result, often called the *Fundamental Theorem of Resultants*:

Theorem 2.7 ([Coll71]): Let K be an algebraically closed field, and let

$$f = \sum_{i=0}^{m} a_i(x_1,...,x_{r-1})x_r^i \quad , \quad g = \sum_{i=0}^{n} b_i(x_1,...,x_{r-1})x_r^i$$

be elements of $K[x_1,...,x_r]$ of positive degrees in x_r. Then if $(\alpha_1,...,\alpha_r)$ is a common zero of f, g we have

$$res(f,g;\, x_r)(\alpha_1,...,\alpha_{r-1}) = 0 \,. \tag{2.18}$$

Conversely, if the above resultant vanishes at $(\alpha_1,...,\alpha_{r-1})$, then at least one of the following holds:

(a) $a_m(\alpha_1,...,\alpha_{r-1}) = ... = a_0(\alpha_1,...,\alpha_{r-1}) = 0$,

(b) $b_n(\alpha_1,...,\alpha_{r-1}) = ... = b_0(\alpha_1,...,\alpha_{r-1}) = 0$,

(c) $a_m(\alpha_1,...,\alpha_{r-1}) = b_n(\alpha_1,...,\alpha_{r-1}) = 0$,

(d) $\exists\ \alpha_r \in K$ such that $(\alpha_1,...,\alpha_{r-1},\alpha_r)$ is a common zero of f, g.

Proof: The first part of the result is obvious using Theorem 2.3. Now assume that (2.18) holds, and that $a_m(\alpha_1,...,\alpha_{r-1}) \neq 0$. Denote by ϕ the homomorphism corresponding to evaluation at $(\alpha_1,...,\alpha_{r-1})$. Then by Theorem 2.6 we have

$res(\phi(f), \phi(g); x_r) = 0$. If $degree(\phi(g)) = 0$, this implies (by the definition of the resultant) that

$$(\phi(g))^m = 0 \quad \Rightarrow \quad b_0(\alpha_1, ..., \alpha_{r-1}) = 0 \ ,$$

i.e., that (b) holds. If $degree(\phi(g)) > 0$, then (by Theorem 2.2) $\phi(f)$ and $\phi(g)$ have a non-constant common divisor $h \in \boldsymbol{K}[x_r]$. Since $\boldsymbol{K}$ is algebraically closed, this has a root which we denote by α_r. It follows that f, g have a common root at $(\alpha_1, ..., \alpha_{r-1}, \alpha_r)$, *i.e.* that (d) holds.

Similarly, if we assume that (2.18) holds and $b_n(\alpha_1, ..., \alpha_{r-1}) \neq 0$, we find that either (a) or (d) holds. The case (c) is the remaining possibility □

As a consequence of this result, we see that given a system of algebraic equations $f_i(x_1, ..., x_r) = 0$, $1 \leq i \leq k$, we may successively eliminate the variables x_1, x_2, ..., without losing information about the common zeros of the original system. For example, given three equations in three unknowns, say

$$F = \{f_1, f_2, f_3\} \subseteq \boldsymbol{Z}[x,y,z] \ ,$$

we may form

$$g_1(y,z) := res(f_1, f_2; x) \ , \quad g_2(y,z) := res(f_1, f_3; x)$$

and

$$h(z) := res(g_1, g_2; y) \ ;$$

then all z-values (in an algebraic extension of $\boldsymbol{Q}$) which may occur in solutions of F are roots of $h(z)$. We mention that by Theorem 2.7, if we were to eliminate *all* of the variables to obtain a non-zero constant as a resultant, then there would be *no* solutions to the given system. If we follow the computation of h by also finding univariate polynomials $p(y)$, $q(x)$, then the zeros of F are contained within the set of triples of roots of q, p, h. This obviously presents the problem of somehow deciding which triples are, in fact, solutions. One possible method (suggested in [Coll71]) is a decision method of [Tars51]. It may also happen that it is not possible to obtain a univariate polynomial in this manner for at least one of the unknowns. Namely, if there are not at most finitely many solutions (which may happen even in the "usual" case of n equations in n unknowns), then no such finite inclusion of the roots can exist. Finally, the elimination may (because of degree considerations) be much more difficult for some permutations of variables than others - and is generally a time-consuming process. For such reasons, the following (more direct) approach is usually taken.

Definition ([Yun73]): A *reduced system* of a set of multivariate polynomials $F = \{f_1, ..., f_k\} \subseteq \boldsymbol{K}[x_1,...,x_n]$ (where $k \geq n$) is a list of n sets of polynomials $G = \{E_1, ..., E_n\}$ where $E_i = \{e_{i1}, ..., e_{i(k-i+1)}\}$, $1 \leq i \leq n$ is a set of $k-i+1$ polynomials in the $n-i+1$ variables $x_i, ..., x_n$ such that the polynomials in E_i contain all roots in those variables which are possible values for common roots in E_{i-1}.

We note that G is the nonlinear analogue of a triangulation of the original system. Therefore, if a reduced system for F is known, then the problem of finding the roots of F can be reduced to a series of univariate problems. These comprise a process which Yun refers to as *back-solving*. For example, a root of E_n, say α_n, is substituted into the polynomials of E_{n-1} to obtain a collection of polynomials in $\boldsymbol{K}[x_{n-1}]$. From these polynomials (*i.e.* from their univariate GCD) we obtain a set of roots, each of which corresponds to a pair (α_{n-1}, α_n). Each pair is in turn substituted into the polynomials of E_{n-2}, and so on. We thus obtain a collection of n-tuples $(\alpha_1,...,\alpha_n)$, except for partial roots which cannot be extended (*e.g.*, when some polynomial becomes a non-zero constant after the subtitution). We note that since the number of solutions of F is, in general, an exponential function of the degrees of its polynomials, the back-solving process is inherently exponential. We also note that even when a system of equations over the integers yields a univariate polynomial of degree less than 5, explicit back-solving may become too complicated to produce useful results in practice.

Given a system $F = \{f_1, ..., f_k\} \subseteq \boldsymbol{K}[x_1,...,x_n]$, Theorem 2.7 allows us to construct a reduced system by computing sets of resultants. If $k < n$ (*i.e.* the system is under-determined, or for that matter if F contains free parameters), we only expect to be able to eliminate $x_1, ..., x_{k-1}$. However, even if the coefficient domain is actually $\boldsymbol{Q}(x_{k+1},...,x_n)[x_1,...,x_k]$, it is typical (for computational reasons discussed in the previous section) to work over $\boldsymbol{Z}[x_1,...,x_n]$ with the understanding that only the permutation of $x_1, ..., x_k$ is significant. Let us first suppose that initially each f_i contains all variables, so that $E_1 = F$. Then we might construct E_2 by choosing the polynomial of lowest degree in x_1 (say f_1 after a re-labelling), and setting

$$E_2 = \{res(f_1, f_2; x_1), res(f_1, f_3; x_1), ..., res(f_1, f_k; x_1)\} \ .$$

If there is an element of F free of x_1, we simply insert it directly into E_2 and compute one fewer resultant from the remaining polynomials. So far, we have neglected to consider the possibility that this procedure does not yield $k-i+1$ non-zero polynomials in the *i-th* step; this will be dealt with at a later point. The above process is then repeated until the last stage of resultants yields only 0, or a non-zero constant is obtained.

It should be clear that the choice of permutation of variables (*i.e.*, the order in which they are eliminated) is of practical significance. Given the bound (2.17) and the pairing scheme suggested above, it is fairly easy to devise a simple heuristic to select the next variable to be eliminated. For example, we might choose the variable for which the sum of the degrees over E_i is a minimum. (Clearly more complicated schemes could also be constructed.) However, even if the permutation is chosen so as to minimize the growth of intermediate polynomials, it is clear from (2.17) that this growth can be extremely rapid. We should therefore try to reduce it (and hence the cost of the forward elimination) whenever possible.

Since we have, by Theorem 2.4,

$$res(fg, h) = res(f, h)\, res(g, h) ,$$

an obvious way to reduce growth is by performing complete system subdivision before each stage of resultants. An additional benefit of this approach is that the removal of common factors eliminates the problem of resultants which vanish identically. Such an approach was far less practical before the rapid advances in algorithms for polynomial GCD's and factorization of the last decade. The principles of system subdivision are fairly obvious, and will not be discussed here; some details are given in [Yun73] (and also in Section 4.2).

A small example has already been given in the first section of this Chapter. However, more insight is provided by examining a slightly larger problem.

Example: Consider the polynomials of Problem 2 (a system of 3 equations in 3 variables, of degree 2; see the Appendix). Since these are dense polynomials, we may as well use the permutation of variables $x > y > z$. We then compute

$$
\begin{aligned}
g_1 &:= res(f_1, f_2; x) \\
&= -177083561152z + 2008912290149z^2 + 1225722380248y \\
&\quad - 4135529281094yz - 1574842963988y^2z - 1694793759680z^3y \\
&\quad + 233262202265z^2y^2 + 625875357534y^3z - 1036313617570y^3 \\
&\quad - 2968802649964z^3 + 4831663295762z^2y + 1313962614364z^4 \\
&\quad + 161918845417y^4 + 1480091137981y^2 - 152471471260 , \\
g_2 &:= res(f_1, f_3; x)
\end{aligned}
$$

(where the second polynomial is of the same form as the first). Both of these polynomials are primitive and irreducible. If we then compute $h_1 := res(g_1, g_2; y)$, we obtain a univariate polynomial in z of degree 16 with 100 digit coefficients. Performing these calculations in the Maple system (version 4.1, on a VAX/8650 processor) requires only 1 or 2 seconds, not including the attempts to factor g_1 and g_2. However, the fact that the above system only has 8 solutions implies that as many roots of $h_1(z)$ are extraneous. The presence of extraneous roots is easily understood if we notice that common roots of f_1, f_2 are not necessarily common roots of f_1, f_2, f_3. In this case, this troublesome aspect of using resultants can be dealt with (if not completely avoided) by the additional computation of

$$
g_3 := res(f_2, f_3; x) , \quad h_2 := res(g_1, g_3; y) ,
$$
$$
\tilde{h}_1 := gcd(h_1, h_2) .
$$

Still, we see that all intermediate computations will suffer from expression swell due to the presence of extraneous roots. The magnitude of this growth is easily established if we consider a system of n homogeneous polynomials (in n variables) of degree d. By a theorem of Bezout ([Waer53]), the number of solutions of such a system (and so, the required degree of a univariate polynomial in a reduced system) is d^n. On the other hand, we see that the use of resultants will yield a final univariate polynomial of degree $d^{2^{n-1}}$. However, it is not clear that the extraneous roots may always be removed in the above manner (*e.g.*, if the system has infinitely many solutions). We further note that as the number of variables and their respective degrees increase, we expect an exponential increase in the number of sub-systems arising from subdivision (in accordance with the greater number of extraneous roots and, probably, factors).

Example: Consider the equations of Problem 7(a) (a system of 4 equations in 4 unknowns, of degrees 1 in x, y and 2 in z, w). If we proceed as before to eliminate x, y, z (in that order), then factorization of intermediate results gives rise to 4 distinct sub-problems before the elimination of z occurs. It happens that all of these except the largest contain extraneous factors (*i.e.*, those which contain *no* common roots of the original system). If we proceed on the largest subproblem, we obtain a univariate polynomial in w of degree 36 from bivariate polynomials of degree 6 in z, w. In fact, it is known that the system has only 14 solutions.

We see from the above example that as the size of the input grows even slightly (in the number of variables or in degree), the size of the intermediate results and number of sub-problems can grow quite rapidly. For larger systems such as Problem 7(b) (a version of Problem 7(a) which contains a few more terms) or Problem 13(d), the intermediate results become too large to be represented. We note also the remarks in [Rime84] regarding the difficulty of using resultants (both within, and independent of, the algsys sub-system of Macsyma) to solve Problem 13(d). We found that, for example, Problem 4(a) (which has infinitely many solutions) could not be solved by algsys (using Unix Macsyma Beta Test Release 308, on a VAX/785 processor with 16Mb. of main memory) in over 23000 sec. of CPU time due to space limitations. In fact, as simple a system as Problem 11 is rejected by algsys as "too complicated". This is not really surprising, since it would be difficult to formulate an algorithm which offers the flexibility of interactive calculations (spoken of in [Mose66] and [Yun73]). For example, the trick of computing extra resultants and GCD's is probably not used. Suppose also that an n by n system with infinitely many solutions does not admit a reduced system. Then the partial triangulation obtained will nonetheless often allow the solutions to be represented, by considering the non-eliminated variables as parameters. However, the elimination algorithm may still compute many more than the $n(n-1)/2$ resultants implied by the definition, in an effort to produce a true reduced system.

The last matter to be discussed is that of the back-solving procedure. We recall that in our first example, each of the 8 z-roots would have to be substituted into bivariate polynomials of degree 4. But since it is known that only 8 complete roots exist, it is clear that much of the work entailed in this phase is due to the presence of extraneous information in the intermediate (bivariate) results. The numerical effects of substituting approximations to the (complex)

z-roots during back-solving are not known. And, even if the final univariate polynomial were of degree less than 5, it may not be possible to carry out the back-solving effectively (*e.g.*, when nested radicals are involved). Finally, if the solutions (and hence, univariate polynomials) contain free parameters, such difficulties become insurmountable. We conclude that a reduced system is somewhat lacking as a representation of the roots of an algebraic system. In the following chapters, we present a more flexible elimination scheme due to Buchberger which yields a more refined counterpart to the reduced system.

Chapter 3: Gröbner Bases

3.1. Polynomial Ideals and Standard Bases

In his famous paper, Hironaka [Hiro64] presented the concept of a standard basis of an ideal of formal power series. Buchberger [Buch65] later introduced the same concept for polynomial ideals, which he named Gröbner bases. In [Buch65] and [Buch70] he presented an algorithm for computing such bases, and produced related results on the solvability of algebraic equations. The notion of Gröbner bases was then further refined and analysed in [Buch76a] and [Buch76b], after which it became more widely known as an important constructive technique in polynomial ideal theory.

Herein, we concentrate on the computation and applications of Gröbner bases for ideals of polynomials over a field K. In practical terms, this will usually be either the field of rational numbers Q, or the field of rational functions $Q(a,b,...,c)$. We begin by motivating the notion of standard bases.

Consider a set of "side relations":

$$x^3yz = xz^2 \ , \ xy^2z = xyz \ , \ x^2y^2 = z^2 \ ,$$

or equivalently

$$f_1 := x^3yz - xz^2 = 0 \ ,$$
$$f_2 := xy^2z - xyz = 0 \ ,$$
$$f_3 := x^2y^2 - z^2 = 0 \ .$$

Consider also the polynomials

$$p := x^2y^2z - z^3 \ , \qquad q := xz^4 - xyz^3 \ .$$

It is obvious that $p = z\,f_3$; hence, p is equivalent to 0 modulo the side relations $\{f_1, f_2, f_3\}$. However, it is not at all clear whether q is equivalent to 0 under the same conditions, or if $\{f_1, f_2, f_3, q\}$ forms a broader set of side relations. Formally, what is desired is the following:

Definition: A *canonical simplifier* on a class of objects T with respect to an equivalence relation $\sim$ (on T) is a function σ: $T \longrightarrow T$ such that

(1) $\sigma(t) \sim t$

(2) $s \sim t \Rightarrow \sigma(s) = \sigma(t)$

for all $s, t \in T$.

Here "=" is used in the sense of equivalence of form. Hence, for each equivalence class in T, σ determines a unique representative. The value $\sigma(t)$ is called the *canonical form* of t. For our example, the following would suffice.

Definition: A *zero-equivalence* (or *normal*) simplifier is a function σ: $T \longrightarrow T$ satisfying

(1) $\sigma(t) \sim t$,

(2) $t \sim 0 \Rightarrow \sigma(t) = \sigma(0)$

for all $t \in T$.

Clearly any canonical simplifier is also a normal simplifier. In either case, one would find that

$$\sigma(q) = \sigma(0)$$

with respect to the relation "equivalence modulo $\{f_1, f_2, f_3\}$". It may seem that the computational difficulty here arises solely from the lack of structure present. For example, a univariate side-relation such as "$i^2 + 1 = 0$" may be applied using pseudo-division. However, we shall see that the addition of such structure is only a first step. In fact, it has been proven (see [Rich68]) that for some broader (but still simple) classes of expressions, the zero-equivalence problem is undecidable (and therefore that no canonical simplifers exist).

We recall that a non-empty subset I of a commutative ring with identity $\boldsymbol{R}$ is called an *ideal* if

(1) $f - g \in I$,

(2) $p f \in I$

for all $f, g \in I$, $p \in \boldsymbol{R}$. The simplest examples are the *null ideal* (consisting of the 0 element) and the *unit ideal* (consisting of the entire ring $\boldsymbol{R}$). The set

$$<p> := \{ pq \mid q \in R \}$$

of all multiples of $p \in R$ is called the *principal ideal generated by p.* (Clearly the unit ideal is $<1>$.) Similarly, for ring elements $f_1, ..., f_k \in R$, the ideal generated by these elements is

$$<f_1,...,f_k> := \{ \sum_{i=1}^{k} p_i f_i \mid p_i \in R \} .$$

Hence, a set of polynomials (side relations, or algebraic equations) $F = \{f_1, ..., f_k\}$ may be viewed as generators of $<F>$; it is therefore referred to as an *ideal basis.* The equivalence relation

$$p \equiv q \ mod \ I \quad \Leftrightarrow \quad p - q \in I$$

divides the ring into *cosets* (equivalence classes) which form the *quotient ring* R/I. Clearly, the statement "q is equivalent to 0 modulo $\{f_1, f_2, f_3\}$" is equivalent to "$q \in <f_1, f_2, f_3>$ ". It turns out that a number of other (basic) computational problems in *polynomial* ideal theory are solved by the construction of a canonical simplifier.

Define the set of n-variate terms by

$$T_n := \{ x_1^{i_1} \cdots x_n^{i_n} \mid i_1, ...,i_n \in N_0 \} .$$

Then, fix a *total ordering* $<_T$ on T_n which satisfies

(1) $1 <_T t$,

(2) $su <_T tu$ whenever $s <_T t$

for all $s, t, u \in T_n$, where $1 = x_1^0 \cdots x_n^0$. Any ordering satisfying the above conditions will suffice. (The reader is referred to [Robb85] for a more complete discussion of term orderings.) However, in practice one of the following two examples is almost always chosen. The *lexicographic* term ordering is defined by

$$s = x_1^{i_1} \cdots x_n^{i_n} <_L x_1^{j_1} \cdots x_n^{j_n} = t \quad \Leftrightarrow \quad (3.1)$$
$$\exists l \text{ such that } i_l < j_l \text{ and } i_k = j_k, \ 1 \leq k < l .$$

Note that this establishes (or alternatively, is induced by) the precedence of variables

$$x_1 >_L x_2 >_L \cdots >_L x_n$$

in the ring $\boldsymbol{K}[x_1,x_2,...,x_n]$. In the case of terms in $[x, y]$, for example,

$$1 <_L y <_L y^2 <_L \cdots <_L x <_L xy <_L \cdots <_L x^2 <_L \cdots .$$

The *graduated* (or *total degree*) term ordering is defined by

$$\begin{aligned} & s <_G t \qquad \Leftrightarrow \\ & degree(s) < degree(t) \text{ , or} \\ & \quad degree(s) = degree(t) \text{ and} \\ & \quad \exists l \text{ such that } i_l > j_l \text{ and } i_k = j_k,\ l < k \leq n \text{ .} \end{aligned} \tag{3.2}$$

That is, terms of like total degree are "graduated" using an *inverse* lexicographic ordering. For terms in $[x, y]$, we have

$$1 <_G y <_G x <_G y^2 <_G xy <_G x^2 <_G \cdots .$$

Any polynomial $p \in \boldsymbol{K}[x_1,...,x_n]$ contains a term which is maximal with respect to the chosen term ordering $<_T$. We refer to this as the *headterm* of p with respect to $<_T$, and write $hterm(p)$. Let $hcoeff(p)$ be the corresponding coefficient, so that

$$M(p) := hcoeff(p)\ hterm(p) \tag{3.3}$$

is the *leading monomial* of p (with respect to $<_T$). It is conventional to write $hterm(0) = 1$ (and of course, $hcoeff(0) = 0$). Further define

$$M(p_1,...,p_m) := lcm(M(p_1),...,M(p_m)) \text{ .} \tag{3.4}$$

Example: For the set of polynomials of $\boldsymbol{Q}[x,y,z]$ defined previously we have, with respect to the lexicographic ordering,

$$hterm(f_1) = x^3yz \text{ , } hterm(f_2) = xy^2z \text{ , } hterm(f_3) = x^2y^2 \text{ ,}$$
$$\text{and } M(f_1,f_2) = x^3y^2z \text{ .}$$

If we view them as polynomials in $\boldsymbol{Q}[z,y,x]$, then

$$hterm(f_1) = xz^2 ,$$

for example. With respect to the total degree ordering on terms of $\boldsymbol{Q}[x,y,z]$,

$$hterm(f_1) = x^3yz \quad \text{and} \quad hterm(q) = xyz^3 .$$

Definition: We say that p *reduces with respect to* q (and with respect to a fixed term ordering) if there exists a monomial in p which is divisible by $hterm(q)$.

In particular, if $p = \alpha t + r$ where $\alpha \in \boldsymbol{K}$, $t \in T_n$, $r \in \boldsymbol{K}[x_1,...,x_n]$ and $t = hterm(q)u$ then we write

$$p \underset{q}{\rightarrow} p - \alpha t \frac{q}{M(q)} = p' . \tag{3.5}$$

Otherwise, we say that p is *irreducible with respect to* q. If p reduces to p' modulo some polynomial in $F = \{f_1, ..., f_k\}$, we say that p *reduces modulo* F and write $p \underset{F}{\rightarrow} p'$; otherwise, p is in *fully reduced form modulo* F. If $M(p)$ reduces modulo F, we say that p is *M-reducible* modulo F.

We note that since the process of reduction involves subtracting an appropriate multiple of one polynomial from another, it may be viewed as one step in a generalized division.

A fundamental property of reduction is the following:

Lemma 3.1 ([Buch65]): The relation $\underset{F}{\rightarrow}$ is Noetherian; *i.e.*, there is no infinite sequence

$$p \underset{F}{\rightarrow} p_1 \underset{F}{\rightarrow} p_2 \underset{F}{\rightarrow} \ldots .$$

Let $\overset{+}{\underset{F}{\rightarrow}}$ denote the associative closure of $\underset{F}{\rightarrow}$. Then as a consequence of the above lemma, we may construct an algorithm *reduce*(p, F) to return q such that

$p \overset{+}{\succ}_F q$ and q is irreducible modulo F. That is, we form a finite sequence

$$p_0 = p \succ_F p_1 \succ_F \cdots \succ_F q = p_n \ .$$

An example of such an algorithm is as follows.

Algorithm 3.1:

procedure *reduce*(p, F)
  $q \leftarrow p$
  while $\exists\ f_k \in F$ such that $q \succ_{f_k} q'$ **do**
    choose α, t, f_k such that $q \succ_{f_k} q'$ and
      t is maximal with respect to $<_T$
    $q \leftarrow q - \alpha t \dfrac{f_k}{M(f_k)}$
  return(q)

We note (following [Buch85]) that choosing maximal t for reduction is not necessary for correctness; it is, however, necessary for efficiency. This algorithm turns out to be extremely important in what follows. It will be examined more closely in Section 3.4.

Example: Suppose we adopt the total degree ordering for polynomials of $\boldsymbol{Q}[x,y,z]$, and consider once again the polynomials

$$f_1 = x^3yz - xz^2, \quad f_2 = xy^2z - xyz, \quad f_3 = x^2y^2 - z^2,$$
$$p = zf_3 = x^2y^2z - z^3, \quad r = -xf_2 = -x^2y^2z + x^2yz \ .$$

Then

$$p \succ_{f_3} x^2y^2z - z^3 - z(x^2y^2 - z^2) = 0 \ , \text{ and}$$

$$r \succ_{f_2} 0 \ .$$

However,

$$p + r \;=\; x^2yz - z^3 \,,$$

which is irreducible modulo $\{f_1, f_2, f_3\}$.

The fact that $p+r$ is irreducible when $p, r \neq 0$ suggests that theorems on reduction will generally be difficult to prove. It also demonstrates that while

$$p \overset{+}{\succ}_F 0 \quad \Rightarrow \quad p \in \langle F \rangle \,,$$

the converse does not hold. Hence, *reduce*(, F) is not a zero-equivalence simplifier. This motivates the following definition:

Definition: An ideal basis G is called a *Gröbner basis* if

$$f \in \langle G \rangle \;\Rightarrow\; \mathit{reduce}(f, G) = 0 \,.$$

Since clearly $\mathit{reduce}(f, G) - f \in \langle G \rangle$, this states that G is a Gröbner basis precisely when its reduction algorithm is a normal simplifier on $K[x_1,\ldots,x_n]/\langle G \rangle$.

Example: For the polynomials $F = \{f_1, f_2, f_3\}$ and p, r of the previous example,

$$\begin{aligned} G := \{f_1, f_2, f_3,\; & x^2yz - z^3,\; xz^3 - xz^2, \\ & yz^3 - z^3,\; xyz^2 - xz^2,\; x^2z^2 - z^4,\; z^5 - z^4\} \end{aligned}$$

is a Gröbner basis (with respect to the total degree ordering for terms in $[x,y,z]$) such that $\langle F \rangle = \langle G \rangle$. Note that $p \overset{+}{\succ}_G 0$, $r \overset{+}{\succ}_G 0$, and $p + r \overset{+}{\succ}_G 0$, irrespective of the sequence of reductions that is followed.

3.2. Buchberger's Algorithm

The above definition for Gröbner bases presents some obvious difficulties. For example, it does not give a means of testing whether a given ideal basis is a Gröbner basis. Moreover, it does not tell us how to construct a Gröbner basis for a given ideal $<F>$. The first of Buchberger's many contributions was to show that completion of the ideal basis to standard form only requires that a certain type of "resolvent" be considered for finitely many pairs of basis polynomials. Specifically, we have the following.

Definition: The *S-polynomial of* $p, q \in \boldsymbol{K}[x_1,...,x_n]$ is

$$Spoly(p, q) := M(p, q) \left[\frac{p}{M(p)} - \frac{q}{M(q)} \right] . \tag{3.6}$$

A theorem of Buchberger then provides alternative characterizations of Gröbner bases.

Theorem 3.2 ([Buch76a]): The following are equivalent:

(1) G is a Gröbner basis;

(2) If $reduce(f, G) = g$ and $reduce(f, G) = h$, then $g = h$;

(3) $reduce(\ Spoly(f, g), G\ ) = 0$ for all $f, g \in G$.

The proof is non-trivial, and will be omitted for brevity. (Only the implications (1) $\Rightarrow$ (2) and (1) $\Rightarrow$ (3) are easily shown.) The connection between (2) and (3) becomes plausible if we view $Spoly(p, q)$ as the difference between reducing $M(p, q)$ modulo p and reducing it modulo q.

Corollary 3.3: If G is a Gröbner basis, then

$$reduce(f, G) = reduce(g, G) \quad \Leftrightarrow \quad f - g \in <G> .$$

Proof: Suppose $h = reduce(f, G) = reduce(g, G)$. Then $f-h \in <G>$ and $g-h \in <G>$. Hence

$$(f-h) - (g-h) = f-g \in <G> .$$

Now assume that $f-g \in <G>$. Then $f-g \overset{+}{\underset{G}{\rightarrow}} 0$. Hence, by a result of [Buch76a], f and g have a "common successor" with respect to the reduction process modulo G; *i.e.*, $\exists\, k$ such that $f \overset{+}{\underset{G}{\rightarrow}} k$ and $g \overset{+}{\underset{G}{\rightarrow}} k$.
Therefore

$$f \overset{+}{\underset{G}{\rightarrow}} k \overset{+}{\underset{G}{\rightarrow}} reduce(k, G)$$

and

$$g \overset{+}{\underset{G}{\rightarrow}} k \overset{+}{\underset{G}{\rightarrow}} reduce(k, G) .$$

Hence, by characterization (2), $reduce(f, G) = reduce(g, G)$ □

We see that if G is a Gröbner basis, then $reduce(\ , G)$ is not only a normal simplifier, but also a canonical simplifier. Hence we can use it to decide ideal membership, congruence modulo an ideal, and ideal inclusion (since $<E> \subseteq <F> \Leftrightarrow reduce(e, G) = 0$ for all $e \in <E>$). Furthermore, if we view T_n as a basis for $\boldsymbol{K}[x_1,...,x_n]$, then the set

$$U := \{[u] \mid u \text{ is not a multiple of a headterm of any poly. in } G\}, \quad (3.7)$$

where $[u]$ is the congruence class of u modulo G, forms a linearly independent vector space basis for $\boldsymbol{K}[x_1,...,x_n]/<F>$. This is so since for any dependence

$$c_1[u_1] + ... + c_m[u_m] = 0 ,$$

where $c_i \in \boldsymbol{K}$, $u_i \in U$ for $1 \leq i \leq m$, there corresponds a polynomial $f := c_1u_1+...+c_mu_m \in <G>$. Hence $reduce(f,G)=0$, which implies that $c_i = 0$, $1 \leq i \leq m$.

Equally important is the observation that characterization (3) provides a means of algorithmically constructing Gröbner bases, appropriately known as *Buchberger's algorithm*:

Algorithm 3.2:

procedure *Gbasis*(F)
  $G \leftarrow F$; $k \leftarrow length(G)$
  $B \leftarrow \{ [i, j] \mid 1 \leq i < j \leq k \}$
  while $B \neq \emptyset$ **do**
    $[i, j] \leftarrow selectpair(B, G)$; $B \leftarrow B - \{[i, j]\}$
    $f \leftarrow reduce(\ Spoly(G_i, G_j), G)$
    if $f \neq 0$ **then**
      $G \leftarrow G \cup \{f\}$; $k \leftarrow k + 1$
      $B \leftarrow B \cup \{ [i, k] \mid 1 \leq i < k \}$
  return(G)

Since $Spoly(G_i, G_j)$ and f are in $<F>$, G remains an ideal basis for $<F>$. Also, if

$$g \ \overset{+}{\rightarrow}_G \ h \ \neq \ 0 \, ,$$

then

$$g \ \overset{+}{\rightarrow}_{G \cup \{h\}} \ 0 \, .$$

To see that this algorithm terminates, we consider the set

$$P_k := \{ \text{ headterms of polynomials in } G \text{ after the } k\text{-}th \text{ extension } \} \, .$$

Then we see that

$$<P_1> \ \subset \ <P_2> \ \subset \ \ldots$$

is a strictly increasing sequence. Hence, by a theorem of Hilbert (see [Waer53], p. 20) on ascending chains of polynomial ideals, it must terminate.

Many connections between this and other algorithms have been noted. It specializes to Euclid's algorithm in the case of univariate polynomials, and to Gauss' algorithm in the case of linear polynomials. The connection with other division/elimination processes (*e.g.* pseudo-division and resultants) was explored briefly in [Pohs81], and more completely (for the bivariate case) in [Laza85]. Generalizations to polynomials over various Euclidean rings are discussed in [Kand84], where the connection with the Knuth-Bendix algorithm [Knut67] is also mentioned. In [Buch83b] and [LeCh83] the Knuth-Bendix and Buchberger

algorithms are both viewed as algorithms of a more general "critical pair/completion" type. When F consists only of polynomials of the form $s - t$ for $s, t \in T_n$ (*i.e.* differences of terms), Buchberger's algorithm specializes to an algorithm for the uniform word problem for finitely generated commutative semigroups. (See [Ball81].) Through this relationship, it is shown in [Mayr82] that the congruence problem for polynomial ideals is exponentially space complete: in the worst case "a degree bound growing exponentially in the number of variables is unavoidable". Hence, the problem of constructing Gröbner bases is an intrinsically hard one.

However, this poor worst case behavior does not mean that the algorithm will always behave poorly in practice. The progress of the EEZ-Hensel factorization algorithm [Wang78] should serve to demonstrate that much depends on the development of practical improvements, and upon the nature of the particular problem at hand. Indeed, with the implementations reported in [Boge85] and [Kapu86] a number of interesting problems have already been solved. (See [Boge86], for example.)

One finds, upon applying the algorithm to some examples, that it can indeed produce complex calculations for apparently simple input. We observe that most of the computational cost comes from the reduction step (*i.e.* where polynomial arithmetic is performed). Also, as the number of polynomials grows, the number of S-polynomial pairs which must be considered grows rapidly. However, as the algorithm progresses, more and more of these reductions will lead to 0. For such reasons, worthwhile improvements will include:

(1) predicting (*i.e.* avoiding) reductions which will necessarily yield 0;

(2) improving the efficiency of the reduction sub-algorithm;

(3) minimizing the overall complexity of the calculations (*i.e.* the number and degree of polynomials).

Not surprisingly, Buchberger has made important contributions, particularly with regard to item (1) above. Specifically, in [Koll78], [Buch79a], [Buch79b] he (along with various co-workers) developed a set of criteria for determining *a priori* a large proportion of those pairs whose reductions will yield 0. (Note the recommendation in [Buch76a] that "one should first concentrate more on establishing criterions ... which reduce the complexity of the algorithm, rather than trying to obtain complexity estimations for crude versions of the algorithm".) His criteria are made possible by the following results.

Lemma 3.4: If $M(p, q) = \alpha M(p)M(q)$, then

$$Spoly(p, q) \overset{+}{\gg}_{\{p, q\}} 0 .$$

Proof: Define $R(p) := p - M(p)$. Then we have

$$\begin{aligned} \alpha^{-1} Spoly(p, q) &= M(q)p - M(p)q \\ &= M(q)R(p) - M(p)R(q) . \end{aligned}$$

The result then follows from the observation that all of the terms in $M(q)R(p)$ and $M(p)R(q)$ must be distinct, and the fact that

$$M(p) \gg_p R(p) \quad \square$$

Lemma 3.5: For arbitrary p, q, r,

$$\frac{M(p,q,r)}{M(p,q)} Spoly(p,q) + \frac{M(p,q,r)}{M(q,r)} Spoly(q,r) + \frac{M(p,q,r)}{M(r,p)} Spoly(r,p) = 0 .$$

Proof: A straightforward application of the definitions yields

$$M(p,q,r) \left[\frac{p}{M(p)} - \frac{q}{M(q)} + \frac{q}{M(q)} - \frac{r}{M(r)} + \frac{r}{M(r)} - \frac{p}{M(p)}\right] = 0 \quad \square$$

Each lemma suggests that under certain conditions, certain pairs [i, j] (*i.e.* the S-polynomial $Spoly(G_i, G_j)$) may be skipped. We therefore reduce the S-polynomial only if the pair $[i, j]$ meets the following criteria:

$$\begin{aligned} &criterion1([i, j], G) \iff \\ &\quad lcm(hterm(G_i), hterm(G_j)) \neq hterm(G_i)\, hterm(G_j) ; \end{aligned}$$

$$\begin{aligned} &criterion2([i, j], B, G) \iff \\ &\quad \neg\, \exists\, k, 1 \leq k \leq length(G), \text{ such that} \\ &\qquad i \neq k \neq j , \\ &\qquad M(G_k) \mid M(G_i, G_j), \\ &\qquad [i, k] \notin B, \ [k, j] \notin B . \end{aligned}$$

We note that if $M(r) \mid M(p,q)$ in Lemma 3.5, then

$$Spoly(p,q) + \frac{M(p,q,r)}{M(q,r)} Spoly(q,r) + \frac{M(p,q,r)}{M(r,p)} Spoly(r,p) = 0,$$

since $M(p,q,r) = M(p,q)$. So, if $Spoly(q,r)$ and $Spoly(r,p)$ have already been reduced to f, g, respectively, it seems plausible that

$$Spoly(p,q) \overset{+}{\rightarrow}_{G \cup \{f,g\}} 0 .$$

However, a rigorous proof that this use of *criterion2* is correct is rather involved; the details may be found in [Buch79b]. Roughly speaking, the practical effect is to reduce the number of S-polynomial reductions from $O(N^2)$ to $O(N)$, where N is the length of the basis. Recently, a slightly different formulation of these criteria was presented in [Geba86] which appears to further reduce the number of 0-reductions.

To this point, we have not discussed the procedure *selectpair*, which chooses the next S-polynomial pair for reduction from the set B. In fact, *any* pair may be selected without affecting the correctness of the algorithm. However, the manner in which it is chosen can certainly affect the complexity of the algorithm, in that some pairs will lead to more complicated polynomials than others. Moreover, it can affect the effectiveness of the criteria themselves. Buchberger has recommended that at each iteration we choose $[i, j]$ such that

$$M(G_i,G_j) = min_{<_T} \{M(G_u,G_v) \mid [u, v] \in B\}, \tag{3.8}$$

henceforth referred to as the *normal selection strategy*. Deviation from this strategy can affect the power of *criterion2* since it becomes less likely that, at a given point, the appropriate pairs $[i, k]$ and $[k, j]$ have already been considered. In addition, it can be shown [Buch79b] that if this strategy is followed, the reduction sub-algorithm must produce a unique result. Therefore, if a criterion rejects a pair, then *all* possible reductions of the corresponding S-polynomial must lead to 0. (If this were not guaranteed, it is argued that it might sometimes be preferable to ignore the criterion and generate the new polynomial.) Buchberger therefore refers to *criterion1* and *criterion2*, when applied in conjunction with the normal selection strategy, as "good" criteria.

In order to bound the computational complexity of the algorithm, it is necessary to determine a bound on the maximum degree of any polynomial which the algorithm may produce. (This yields a bound on the maximum number of polynomials, and also the maximum number of reduction steps for

each.) Since this is extremely difficult, research has concentrated primarily on the bivariate case. In [Buch79a], Buchberger examines the algorithm using another criterion (which is equivalent to *criterion2* when the normal selection strategy is used), with the total degree term ordering. He finds that the degree of any polynomial produced is bounded (roughly) by $4D_F$, where

$$D_F := \max \{degree(F_i) \mid 1 \leq i \leq length(F)\} .$$

(Actually, he also produces a tighter but more complicated bound.) It can then be shown that the number of computational steps required is bounded by

$$C_F := 2(length(F) + 16D_F^2)^4 .$$

In [Buch83a], these results are refined and expanded to include the following.

Theorem 3.6: For every natural number D, $\exists\ F \subseteq K[x,y]$ with $D = D_F$ such that

(a) for all Gröbner bases for F with respect to $<_G$, $D_G \geq 2D - 1$;

(b) for all Gröbner bases for F with respect to $<_L$, $D_G \geq D^2 - D + 1$.

Hence, in the bivariate case, quadratic growth is intrinsic to the lexicographical ordering. In [Wink84], Winkler showed that for $F \subseteq K[x,y,z]$,

$$D_G \leq (8D_F + 1)\, 2^{d_F} ,$$

where $d_F := min\{degree(F_i) \mid 1 \leq i \leq length(F)\}$, when the total degree ordering is used. Other useful complexity results include those of [Laza83], [Moll84], [Gius85].

We conclude this section with the issue of uniqueness. It should be fairly clear that in general, Gröbner bases are not unique. For instance, in our previous example, $G - \{f_1\}$ is also a Gröbner basis for $<F>$. We now show that this is easily remedied.

Definition: A Gröbner basis G is *reduced* if for all $f \in G$, $f = reduce(f, G-\{f\})$ and $hcoeff(f) = 1$.

Theorem 3.7 ([Buch76b]): If $<G_1>=<G_2>$, and G_1, G_2 are reduced Gröbner bases, then $G_1 = G_2$.

Clearly, the specification that the polynomials be monic is arbitrary, and could be replaced by some other normalization. Let us assume that this is done in some consistent manner by the reduction sub-algorithm. Then a set F (not necessarily a Gröbner basis) can be transformed to reduced form with the following:

Algorithm 3.3:

procedure *reduceset*(F)
  return(*normalize*(*minimize*(F))) ,

where we have

procedure *minimize*(F)
  R $\leftarrow$ F ; P $\leftarrow \emptyset$
  while $R \neq \emptyset$ **do**
    $h \leftarrow$ *selectpoly*(R) ; $R \leftarrow R - \{h\}$
    $h \leftarrow$ *reduce*(h, P)
    if $h \neq 0$ **then**
      $Q \leftarrow \{ q \in P$ such that $M(h) \mid M(q) \}$
      $R \leftarrow R \cup Q$
      $P \leftarrow P - Q \cup \{h\}$
  return(P)

and

procedure *normalize*(F)
  $S \leftarrow F$; $T \leftarrow \emptyset$
  while $S \neq \emptyset$ **do**
    $h \leftarrow$ *selectpoly*(S) ; $S \leftarrow S - \{h\}$
    $h \leftarrow$ *reduce*(h, $F-\{h\}$)
    $T \leftarrow T \cup \{h\}$
  return(T)

The procedure *minimize* ensures that all dependencies are removed from the basis. If it is applied at the end of Algorithm 3.2, we can limit the number of dependencies (and therefore the number of reductions) in a simple manner. We note that if $M(q) \mid M(p)$, then $Spoly(p,q)$ is equal (up to a rescaling) to the reduction of p modulo q. Suppose then that in Algorithm 3.2, we choose a pair $[i, j]$ such that $M(G_j) \mid M(G_i)$, and compute

$$Spoly(G_i, G_j) \twoheadrightarrow_G f \ .$$

Then G_i has been reduced to either 0 or a new basis element, and is therefore *redundant*. Such polynomials may be removed from the basis before invoking *reduceset*. From these remarks it is also clear that the procedure *selectpoly* parallels the normal selection strategy if it chooses the polynomial h which is $<_T$-minimal. But, it is important to realize that (unlike S-polynomial reduction under normal selection) the result of *reduceset* will not be unique if F is not already a Gröbner basis.

An arbitrary input set F may also contain unnecessary polynomials (due to trivial dependencies) or extremely simple polynomials (*e.g.* of degree 1). It is therefore reasonable to apply the above algorithm not only at the end of Algorithm 3.2, but at the beginning as well. Buchberger's algorithm (including the use of *criterion1* and *criterion2*) then takes the following form:

Algorithm 3.4:

procedure *Gbasis*(F)
  $G \leftarrow reduceset(F)$; $k \leftarrow length(G)$; $R \leftarrow \emptyset$
  $B \leftarrow \{ [i, j] \mid 1 \leq i < j \leq k \}$
  while $B \neq \emptyset$ **do**
    $[i, j] \leftarrow normselect(B, G)$; $B \leftarrow B - \{[i, j]\}$
    if $hterm(G_j) \mid hterm(G_i)$ **then** $R \leftarrow R \cup \{G_i\}$
    else if $hterm(G_i) \mid hterm(G_j)$ **then** $R \leftarrow R \cup \{G_j\}$
    if *criterion1*$([i, j], G)$ **and** *criterion2*$([i, j], B, G)$ **then**
      $f \leftarrow reduce(\ Spoly(G_i, G_j), G)$
      if $f \neq 0$ **then**
        $G \leftarrow G \cup \{f\}$; $k \leftarrow k + 1$
        $B \leftarrow B \cup \{ [i, k] \mid 1 \leq i < k \}$
  return($reduceset(G - R)$)

The inter-reduction of the basis polynomials may be carried even further. In [Buch85] it is recommended that it be performed after *each* new non-zero polynomial is formed in *Gbasis*. This slightly complicates the algorithm, in that any M-reductions will require transformation of the set B. (See [Buch85], p. 6.13 for the details.) It is hoped that by keeping the polynomials as reduced, and as few, as possible the complexity of the algorithm may be reduced. Still, it should be noted that the precise effect of such a modification might be difficult to predict, in general. We will examine this additional variant of the algorithm more closely in the next section.

3.3. The Lexicographic Ordering: Criteria and Selection Strategies

It should be clear from the previous sections that a Gröbner basis with respect to one term ordering $<_T$ is not necessarily a Gröbner basis with respect to another term ordering. In fact, Buchberger's algorithm using $<_L$ will tend to display much different behavior than when using $<_G$. For example, the available complexity results and empirical evidence suggest that the overall complexity of the algorithm is greater when using $<_L$. When using $<_G$, the algorithm benefits from the fact that a bound on the total degree of a polynomial yields a bound on the number of terms in the polynomial. Specifically, we note the following.

Lemma 3.8 ([Knut69], Vol. 1): The number of distinct n-variate terms of total degree k is

$$\binom{k+n-1}{n-1} ;$$

hence the number of distinct terms of degree less than or equal to d is

$$\sum_{k=0}^{d} \binom{k+n-1}{n-1} = \binom{d+n}{n} .$$

Therefore, if $hterm(p) <_G hterm(q)$ we can say that p is simpler than q in the above sense. Moreover, this fact simplifies many details which arise in the implementation of Algorithm 3.4. Whenever some degree of freedom exists (*e.g.* in the selection of S-polynomial pairs), a reasonable approach is to "minimize with respect to $<_G$". For $<_L$, there is no counterpart to Lemma 3.8; it is possible that $hterm(p) <_L hterm(q)$, but that p is much more complex in many senses. For this reason, the above strategy may prove to be a poor one when $<_L$ is used. In order to develop more practical strategies, it is becomes necessary to better understand the nature of the algorithm when using $<_L$. The remainder of this section will deal primarily with this term ordering.

We have already noted that an S-polynomial (or reduction) may be viewed as one step of a generalized division. This is made clearer if we consider the specialization to univariate polynomials (*cf.* one step of a polynomial division). Noting also the specialization to linear polynomials (*i.e.* Gauss' algorithm), we see that there is a connection between S-polynomials and other nonlinear elimination processes. This is illustrated by the following example.

Example: Consider the polynomials

$$p_1 := 3x^2y - y^3 - 4\ , \quad p_2 := x^2 + xy^3 - 9\ ,$$

where $x >_L y$. Then

$$r := res(p_1,p_2;\, x) = -3y^{10} - 12y^7 + y^6 - 54y^4 + 8y^3 + 729y^2 - 216y + 16;$$

$$p_3 := prem(p_1,p_2;\, x) = -3xy^4 - y^3 + 27y - 4\ , \quad prem(p_2,p_3;\, x) = r\ .$$

However, we may also compute:

$$\begin{aligned}
Spoly(p_1,\, p_2) &= (3x^2y - y^3 - 4) - 3y(x^2 + xy^3 - 9) = p_3\ , \\
Spoly(p_2,\, p_3) &= -3y^4(x^2 + xy^3 - 9) - x(-3xy^4 - y^3 + 27y - 4) \\
&= -3xy^7 + xy^3 - 27xy + 4x + 27y^4 \\
&\gtrdot_{p_3} xy^3 - 27xy + 4x + y^6 + 4y^3 \ = \ p_4\ , \\
Spoly(p_3,\, p_4) &= 81xy^2 - 12xy - 3y^7 - 12y^4 + y^3 - 27y + 4 \ = \ p_5\ , \\
Spoly(p_4,\, p_5) &= 12xy^2 - 2187xy + 324x + 3y^8 + 81y^6 + 12y^5 - y^4 + 324y^3 \\
&\qquad + 27y^2 - 4y \\
&\gtrdot_{p_5} -\frac{19667}{9}xy + 324x + \frac{4}{9}y^7 + 81y^6 + 12y^5 + \frac{7}{9}y^4 + \frac{8744}{27}y^3 \\
&\qquad + 27y^2 - \frac{16}{27} \\
&\sim 59001xy - 8748x - 81y^8 - 12y^7 - 2187y^6 - 324y^5 - 21y^4 \\
&\qquad - 8744y^3 - 729y^2 + 16 \ = \ p_6\ , \\
Spoly(p_5,\, p_6) &= 192xy + \ldots \gtrdot_{p_6} \frac{559872}{19667}x + \ldots \\
&\sim 768x + 59001y^9 + 8748y^8 + 1296y^7 + 236196y^6 + 15325y^5 \\
&\qquad + 2268y^4 + 1062354y^3 + 64y^2 - 14337243y + 2122308 = p_7, \\
Spoly(p_6,\, p_7) &= -2239488x - \ldots \gtrdot_{p_7} -1160372667y^{10} + \ldots \ \sim \ r\ .
\end{aligned}$$

Here, $\sim$ denotes similarity in $\boldsymbol{Q}[x,y]$ (*i.e.*, equivalence up to a re-scaling), and *prem* is the pseudo-remainder given by (2.8).

We observe that just as a resultant may be computed via a series of pseudo-remainders, a pseudo-remainder may be computed via a series of reduced S-polynomials. Note however that the S-polynomial sequence contains many more steps than the single resultant computation. We also mention that in the course of Algorithm 3.4, the S-polynomials will be chosen and reduced in different sequences than that above. Hence, the exact relationship with resultants is elusive. (See [Laza85].) Roughly stated, though, we see that the effect of the algorithm is to eliminate and separate the variables x_1, x_2, ..., x_{n-1} (according to the precedence which induces the lexicographical ordering). Therefore, if there exists a permutation of variables $x_{\pi(1)}$, ..., $x_{\pi(n)}$ such that with respect to the associated lexicographical ordering the input basis F is "nearly triangular" (*i.e.* contains polynomials of successively fewer variables), the algorithm may be very fast. Accordingly, the complexity of the calculations produced by the algorithm depends very strongly on the permutation of variables. For the purposes of the present discussion, we will simply regard this ordering (and the induced lexicographic order) as fixed.

We now note that the polynomials of the previous example have the following structure.

Definition: An *M-chain* is a finite subset $\{p_1, ..., p_k\} \subseteq K[x_1,...,x_n]$ such that $M(p_{i+1}) \mid M(p_i)$ for $1 \leq i \leq k-1$.

Since the elimination and separation of variables tends to proceed by small steps, there will (at a given point in Algorithm 3.4) often exist many subsequences of polynomials in the partial basis G which form such chains. Hence there will exist many pairs for which the quantity $M(G_i, G_j)$ is identical.

Let us now apply these observations to modify our use of Buchberger's criteria. Recall that we begin the main loop of Algorithm 3.4 by choosing a pair according to the "normal" selection strategy:

Algorithm 3.5:

procedure *normselect*(B, G)

return$([i, j] \mid M(G_i, G_j) = min_{<_T}\{M(G_u,G_v) \mid [u, v] \in B\})$

The pair must then pass both criteria before reduction of the S-polynomial proceeds. Due to the ongoing separation of variables, pairs which fail to satisfy *criterion1* should be fairly common. While the main cost of the algorithm is that of the reduction step, there will nonetheless be an "overhead" associated with adding, selecting and removing pairs from the set B. Since no new polynomial will ever result from one of the above pairs, this overhead is essentially wasted in such cases. As *criterion1* does not depend on the set B, these pairs could in fact be detected before they are added to B. We must be careful, though, that such a modification does not adversely affect the complexity of the calculations which follow.

Let us suppose that we partition the set B by

$$B^{(1)} := \{ [i, j] \in B \mid \neg \mathit{criterion}1([i, j], G) \},$$
$$\tilde{B} := B - B^{(1)} .$$

Then omitting pairs which do not satisfy the criterion is equivalent to a new selection strategy, namely:

Algorithm 3.6:

procedure *select1*(B, G)
  if $B^{(1)} \neq \emptyset$ **then return**($[i, j] \in B^{(1)}$)
  else return(*normselect*$(\tilde{B}, G)$)

If at some point $B^{(1)}$ is not empty, we might choose a pair which is not minimal (in the sense of (3.8)); hence the S-polynomial is not guaranteed to reduce *uniquely* to 0. However, the existence of such a reduction suggests that the pair should be rejected. Moreover, we are only avoiding a computation which would also be avoided under the normal selection strategy.

When all $B^{(1)}$ pairs have been rejected, we have

$$\mathit{select}1(B, G) = \mathit{normselect}(B, G) .$$

At this point, the pair must pass *criterion2* (which is, under *normselect*, a "good" criterion). (In fact, since it depends on B, the absence of $B^{(1)}$ pairs may even increase the power of *criterion2*.) Still, there may exist many pairs which are $<_T$-minimal, especially if G admits many M-chains. It is then possible that some (but not all) may fail to satisfy *criterion2*. Since the criterion is relatively inexpensive to apply, and since it becomes more powerful as pairs are removed from B, it would be feasible to apply it exhaustively to the set of $<_T$-minimal

pairs. We write the corresponding selection strategy (and re-phrasing of Algorithm 3.4) as follows.

Algorithm 3.7:

procedure *select2*(B, G)
  $C^{(2)} \leftarrow B$; $R \leftarrow \emptyset$
  while $C^{(2)} \neq \emptyset \neq B$ **do**
    $C \leftarrow \{ [i, j] \in B \mid [i, j] = normselect(B, G) \}$
    $C^{(2)} \leftarrow \{ [i, j] \in C \mid \neg criterion2([i, j], B, G) \}$
    $B \leftarrow B - C^{(2)}$
    for $[i, j] \in C^{(2)}$ **do**
      if $hterm(G_j) \mid hterm(G_i)$ **then** $R \leftarrow R \cup \{G_i\}$
      else if $hterm(G_i) \mid hterm(G_j)$ **then** $R \leftarrow R \cup \{G_j\}$
  if $B \neq \emptyset$ **then return**([$normselect(C, G), B, R$])
  else return([, B, R])

Algorithm 3.8:

procedure *Gbasis*(F)
  $G \leftarrow reduceset(F)$; $k \leftarrow length(G)$; $R \leftarrow \emptyset$
  $B \leftarrow \{ [i, j] \mid 1 \leq i < j \leq k$ and $criterion1([i, j], G) \}$
  while $B \neq \emptyset$ **do**
    [$[i, j], B, Q$] $\leftarrow select2(B, G)$; $R \leftarrow R \cup Q$
    if $B \neq \emptyset$ **then**
      $B \leftarrow B - \{[i, j]\}$
      if $hterm(G_j) \mid hterm(G_i)$ **then** $R \leftarrow R \cup \{G_i\}$
      else if $hterm(G_i) \mid hterm(G_j)$ **then** $R \leftarrow R \cup \{G_j\}$
      $f \leftarrow reduce(\ Spoly(G_i, G_j), G)$
      if $f \neq 0$ **then**
        $G \leftarrow G \cup \{f\}$; $k \leftarrow k + 1$
        $B \leftarrow B \cup \{ [i, k] \mid 1 \leq i < k$ and $criterion1([i, k], G) \}$
  return($reduceset(G - R)$)

It remains to be seen if such changes will actually yield any substantial practical improvement. Let us compare the standard formulation (Alg. 3.4) with Algorithm 3.8 (where all other sub-algorithms are identical) on a variety of problems with integer coefficients (which are described in the Appendix).

Problem		Term Ordering			
		$<_G$		$<_L$	
		Alg. 3.4	Alg. 3.8	Alg. 3.4	Alg. 3.8
1	0-reductions	5	3	3	1
	time (sec.)	3	3	3	2
	space (Kb.)	336	303	320	262
2	0-reductions	5	5	17	10
	time	11	10	169	84
	space	680	696	1188	1073
3	0-reductions	110	101	74	35
	time	723	636	216	123
	space	1442	1360	1122	1073
4(a)	0-reductions	20	20	77	53
	time	40	40	930	498
	space	992	975	1565	1318
4(b)	0-reductions	21	21	83	55
	time	40	40	1038	530
	space	983	975	1507	1327
5(a)	0-reductions	16	16	6	2
	time	45	41	24	13
	space	1057	1032	1016	876
5(b)	0-reductions	16	16	31	15
	time	58	57	1216	474
	space	1090	1064	1810	1671
5(c)	0-reductions	16	16	67	40
	time	60	58	8616	3653
	space	1065	1065	2680	2638
6(a)	0-reductions	16	16	54	44
	time	43	42	379	250
	space	992	991	1212	1180
6(b)	0-reductions	18	18	68	63
	time	49	48	749	534
	space	1024	1016	1704	1335
6(c)	0-reductions	18	18	152	110
	time	50	49	18121	10262
	space	1057	1032	2990	2703
7(a)	0-reductions	18	18	48	16
	time	77	76	11914	3070
	space	1148	1147	3064	2810

Table 3.1: standard *vs.* modified use of criteria
(Maple 4.1 on a VAX/8650 processor.)

We first note the obvious difference between the times for the two term orderings: in general, Gröbner bases are much easier to compute with respect to the graduated term ordering. Note also that in this case the algorithm is largely insensitive to permutations of the variable ordering. We remark, however, that if we choose a permutation of variables which is highly favorable for the lexicographic ordering (due to sparse input), the lexicographic basis may actually be easier to compute. For this reason, a dense problem of low degree in few variables may be much more difficult than a structured problem in many more variables, when $<_L$ is used. (Compare Problems 3 and 7(a), for example.) Finally, we note that our modified application of Buchberger's criteria can result in much lower computing time for a wide range of problems when using the lexicographic ordering.

To this point, we have not discussed the manner in which we should choose from among the equivalent, admissible pairs left at the end of Algorithm 3.7. Once again, an approach is suggested by our previous example. After the polynomial "p_6" is formed, the set contains a number of M-chains. Accordingly there are several pairs for which $M(p_i, p_j) = xy^3$, for example. We have only to compare $Spoly(p_4, p_5)$ and $Spoly(p_4, p_7)$ to see that some pairs will lead to polynomials of higher degree than others. This is unavoidable, since the elimination/separation of headterms causes an increase in the degree of the variables which are subordinate in $<_L$. The procedure *normselect* should therefore take into account the degrees of the polynomials in each pair. A reasonable heuristic is then to always choose the pair which was created first (*i.e.* from "earlier" polynomials), when a choice exists. Throughout this chapter, we use this approach implicitly for Algorithm 3.5. By the same token, we see that the manner in which the input is pre-reduced (by *reduceset*) can affect the degree of the input to the main loop of Algorithm 3.8 (and thereby the overall complexity). But since a general strategy is not clear, we will continue to use our previous approach in what follows.

It is also important to note that the above degree of freedom (in selecting S-polynomials) is lost if the basis polynomial are inter-reduced during the main loop. This suggests that the algorithm might produce *more* complex calculations than if the polynomials were not inter-reduced. We consider the following variant of the algorithm. (See also [Buch85], p. 6.13.)

Algorithm 3.9:

procedure *Gbasis*(F)
  $G \leftarrow$ *reduceset*(F) ; $k \leftarrow$ *length*(G)
  $B \leftarrow \{ [i, j] \mid 1 \leq i < j \leq k$ and *criterion1*($[i, j]$, G) $\}$
  while $B \neq \emptyset$ **do**
    $[i, j] \leftarrow$ *normselect*(B, G) ; $B \leftarrow B - \{[i, j]\}$
    if *criterion2*($[i, j]$, B, G) **then**
      $f \leftarrow$ *reduce*(*Spoly*(G_i, G_j), G)
      if $f \neq 0$ **then**
        $P \leftarrow$ *reduceset*($G \cup \{f\}$)
        $Q \leftarrow \{ h \in P \mid \exists$ g $\in G$ with *hterm*(g)=*hterm*(h) $\}$
        $k \leftarrow$ *length*(Q)
        $S \leftarrow P - Q$
        $C \leftarrow \emptyset$
        for $[i, j] \in B$ **do**
          if $\exists$ $[l,m]$ with *hterm*(G_i)=*hterm*(Q_l), *hterm*(G_j)=*hterm*(Q_m)
          then
            $C \leftarrow C \cup \{ [l, m] \}$
        for $s \in S$ **do**
          $Q \leftarrow Q \cup \{s\}$; $k \leftarrow k + 1$
          $C \leftarrow C \cup \{ [i, k] \mid 1 \leq i < k$ and *criterion1*($[i, k]$, G) $\}$
        $G \leftarrow Q$; $B \leftarrow C$
  return(G)

Finally, the previous argument also suggests that the normal selection strategy itself may not always be optimal, in the sense of the overall complexity of calculations produced. It is possible that by forming S-polynomials of lower degree we can minimize the degree growth as the elimination proceeds. Instead of *select2*, Algorithm 3.8 could use the following heuristic selection strategy.

Algorithm 3.10:

procedure *select3*(B, G)
  $D^{(2)} \leftarrow B$; $R \leftarrow \emptyset$
  while $D^{(2)} \neq \emptyset \neq B$ **do**
    $d \leftarrow min \{ degree(Spoly(G_u,G_v)) \mid [u, v] \in B \}$
    $D \leftarrow \{ [i, j] \in B \mid degree(Spoly(G_i,G_j)) = d \}$
    $D^{(2)} \leftarrow \{ [i, j] \in D \mid \neg criterion2([i, j], B, G) \}$
    $B \leftarrow B - D^{(2)}$
    for $[i, j] \in D^{(2)}$ **do**
      if $hterm(G_j) \mid hterm(G_i)$ **then** $R \leftarrow R \cup \{G_i\}$
      else if $hterm(G_i) \mid hterm(G_j)$ **then** $R \leftarrow R \cup \{G_j\}$
  if $B \neq \emptyset$ **then return**([*normselect*(D, G), B, R])
  else return([, B, R])

In Tables 3.2 and 3.3, we compare selection strategies and algorithm variants for each term ordering. Note that, once again, identical code is used for common sub-algorithms such as reduction. It is apparent that when the graduated term ordering is used, the behaviour of Algorithm 3.9 is not markedly different from that of the other formulations. The overhead of many extra calls to the *reduce* procedure has made our implementation slower than Algorithm 3.8; but it is reasonable to expect that for some problems (or different implementations), Algorithm 3.9 might yield an improvement.

When the lexicographic ordering is used, we find a surprising difference between algorithm variants and selection strategies. Algorithm 3.9 does indeed suffer from a large intermediate degree growth (*i.e.* expression swell) on many problems. We find that it only performs well on problems of special structure. It is therefore clear that this approach should not be used in the stated form. It may still be possible to combine it with a heuristic to decide *when* to interreduce the basis. However, Algorithm 3.8 is a more practical variant. (In fact, any variant in which the "redundant" polynomials are removed from the basis before termination will suffer from this expression swell, to some extent.)

Problem		Algorithm		
		3.4	3.8	3.9
2	max. degree	4	4	4
	time (sec.)	11	10	12
	space (Kb.)	680	696	614
3	max. degree	4	4	4
	time	723	636	1952
	space	1442	1360	1262
4(a)	max. degree	4	4	4
	time	40	40	61
	space	992	975	926
4(b)	max. degree	4	4	4
	time	40	40	80
	space	983	975	877
5(a)	max. degree	3	3	3
	time	45	41	68
	space	1057	1032	1032
5(b)	max. degree	3	3	3
	time	58	57	83
	space	1090	1064	1024
5(c)	max. degree	3	3	3
	time	60	58	88
	space	1065	1065	1065
6(a)	max. degree	5	5	5
	time	43	42	78
	space	992	991	1008
6(b)	max. degree	5	5	5
	time	49	48	96
	space	1024	1016	1049
6(c)	max. degree	5	5	5
	time	50	49	96
	space	1057	1032	1049
7(a)	max. degree	4	4	4
	time	77	76	123
	space	1148	1147	1155
9	max. degree	11	11	11
	time	226	218	1015
	space	1188	1196	1212

Table 3.2: comparison of algorithm variants for $<_G$ (Maple 4.1 on a VAX/8650 processor.)

Problem		Alg. 3.8		Alg 3.9
		select2	*select3*	
2	max. degree	12	8	10
	time (sec.)	84	28	68
	space (Kb.)	1073	934	1548
3	max. degree	9	8	9
	time	123	392	115
	space	1073	1417	950
4(a)	max. degree	29	9	$\geq$40
	time	498	55	$>$14812
	space	1318	1040	$>$12000
4(b)	max. degree	31	11	57
	time	530	41	12188
	space	1327	1008	6234
5(a)	max. degree	12	10	11
	time	13	17	15
	space	876	983	1155
5(b)	max. degree	21	10	$\geq$31
	time	474	57	$>$60000
	space	1671	1286	$\geq$6176
5(c)	max. degree	24	10	$\geq$29
	time	3653	119	$>$36000
	space	2638	1655	$>$12000
6(a)	max. degree	20	13	$\geq$27
	time	250	83	$>$3404
	space	1180	1335	$>$12000
6(b)	max. degree	23	13	27
	time	534	129	2081
	space	1335	1245	3097
6(c)	max. degree	42	13	$\geq$42
	time	10262	226	$>$25026
	space	2703	1319	$>$12000
7(a)	max. degree	31	14	$\geq$47
	time	3070	219	$>$60000
	space	2810	2253	$\geq$11305
9	max. degree	70	$\geq$19	69
	time	46019	$>$60000	16180
	space	9280	$\geq$8528	8217

Table 3.3: algorithm variants and selection strategies for $<_L$ (Maple 4.1 on a VAX/8650 processor.)

What is perhaps more surprising is the degree of success of the (somewhat crude) heuristic selection strategy. By minimizing degree growth, it often results in computations of lower overall complexity (and therefore lower computing time) than the normal strategy. We expect that the relative density of 0-reductions will be larger (*e.g.* Problem 3), since *criterion2* is no longer as powerful. It may also happen that S-polynomials chosen in this manner will require reduction paths which contain many more (single reduction) steps than would a $<_L$-minimal S-polynomial. So, this strategy could still be slower than the normal strategy despite the degree reduction (*e.g.* Problem 9). Still, it shows that to some extent the sensitivity of the algorithm to permutations of the variable ordering is a consequence of the degree growth which (unavoidably) results from a normal selection strategy. The development of a more effective general strategy for the lexicographic ordering (possibly as a compromise between the normal strategy and a heuristic) is an important problem for future study.

3.4. The Reduction Sub-algorithm

It should be clear that for a given variant (and selection strategy) the efficiency of Buchberger's algorithm depends to a large extent upon the efficiency of the reduction sub-algorithm. We note that Algorithm 3.2 appears simple only because the awkward details of this process (which are not mentioned in Algorithm 3.1) are hidden in the procedure *reduce*. After presenting a clearer statement of the reduction process, we will examine ways in which its general efficiency may be improved. Once again, it happens that when $<_L$ is used, much greater attention must be devoted to the details of the sub-algorithm.

Following Algorithm 3.1, we see that a polynomial p is fully reduced iff:

(1) $M(p)$ is irreducible, and

(2) $p - M(p)$ is fully reduced.

We recall that $M(p)$ is reducible (modulo F) iff the subset

$$R_p := \{ f \in F \text{ s.t. } hterm(f) \mid hterm(p) \} \tag{3.9}$$

is non-empty. A convenient criterion for full reduction is then that

$$R_p = R_{p_1} = \ldots = \emptyset ,$$

where $p_1 := p - M(p)$, $p_{i+1} := p_i - M(p_i)$. Algorithm 3.1 may then be re-phrased as follows:

Algorithm 3.11:

procedure *reduce*(q, F)
  $p \leftarrow q$; $h \leftarrow 0$
  while $p \neq 0$ **do**
    $R_p \leftarrow \{ f \in F \text{ s.t. } hterm(f) \mid hterm(p) \}$
    while $R_p \neq \emptyset$ **do**
      $f \leftarrow selectpoly(R_p)$
      $p \leftarrow p - M(p) f / M(f)$
      $R_p \leftarrow \{ f \in F \text{ s.t. } hterm(f) \mid hterm(p) \}$
    if $h = 0$ **and** $p \neq 0$ **then** $p \leftarrow p / hcoeff(p)$
    $h \leftarrow h + M(p)$; $p \leftarrow p - M(p)$
  return(h)

The details of the procedure *selectpoly* (which need not be identical with that of Algorithm 3.3) will be discussed later in this section. For now (and in the previous section as well) we assume that it simply returns the first available $f \in R_p$.

The first detail to be considered is that of the type of arithmetic used in the reduction step (3.5). It is common in the literature to perform the arithmetic precisely as stated above. We must realize, however, that if the coefficient field $\boldsymbol{K}$ is one of rational numbers ($\boldsymbol{Q}$) or rational functions ($\boldsymbol{Q}(a,b,...,c)$) - which are arguably most common - a single $\boldsymbol{K}$-multiplication requires several integer (or rational polynomial) steps. For example, to compute the product

$$\frac{e}{f} := \frac{a}{b} \times \frac{c}{d}$$

(where a/b and c/d are already in lowest terms), we must compute either

$$e = ac\ ,\quad f = bd\ ,\quad u = gcd(e,\ f)\ ,$$
$$e = \frac{e}{u}\ ,\quad f = \frac{f}{u}\ ,$$

or

$$u = gcd(a,\ d)\ ,\quad v = gcd(c,\ b)\ ,$$
$$\tilde{a} = \frac{a}{u}\ ,\ \tilde{b} = \frac{b}{v}\ ,\ \tilde{c} = \frac{c}{v}\ ,\ \tilde{d} = \frac{d}{u}\ ,$$
$$e = \tilde{a}\tilde{c}\ ,\quad f = \tilde{b}\tilde{d}\ .$$

(The latter may be more efficient if the integers involved are large.) Therefore, for these two important fields, the cost of coefficient operations may be relatively large. This is, in part, the reason for the re-scaling done at the end of Algorithm 3.11 (as suggested in [Buch85]). Since

$$M(p)\ \frac{f}{M(f)} = \frac{hcoeff(p)}{hcoeff(f)}\ \frac{hterm(p)}{hterm(f)}\ f\ ,$$

the requirement that $hcoeff(f) = 1$ (where f is a generic element of a reducing basis F) immediately saves a multiplication and a division in (3.5). Hence, if f is used in many reductions or S-polynomials, the cost of the re-scaling becomes worthwhile. Moreover, in a manner analogous to the monic Euclidean PRS, the cancellation of greatest common divisors in the remaining coefficients of f helps to control coefficient growth.

The analogy with polynomial remainder sequences (which is particularly appropriate when $<_L$ is used) suggests another approach: in the manner of the primitive PRS we might avoid explicit fractions when $\boldsymbol{K}$ is the quotient field of a unique factorization domain (*e.g.* $\boldsymbol{Z}$ when the field is $\boldsymbol{Q}$). Let us suppose, for example, that $p,\ f \in \boldsymbol{Q}[x_1,\ldots,x_n]$ contain π, ϕ terms, respectively. Suppose further that f is monic and that we use the simpler formulation of rational arithmetic. Then the reduction step (3.5) requires $(\phi-1)$ rational multiplications, and up to $(\pi+\phi-2)$ rational additions. In terms of integer operations, this translates to $(\pi+2\phi-3)$ GCD computations, $(3\pi+5\phi-8)$ multiplications, $(2\pi+4\phi-6)$ divisions and $(\pi+\phi-2)$ additions. If, on the other hand, we take $p,\ f \in \boldsymbol{Z}[x_1,\ldots,x_n]$ where f is primitive (*i.e.* the GCD of its coefficients with respect to the basis T_n is 1) and compute

$$p \gg hcoeff(f)\, p \;-\; hcoeff(p)\, \frac{hterm(p)}{hterm(f)}\, f \quad , \tag{3.10}$$

then only $(\pi+\phi)$ multiplications and $(\pi+\phi-2)$ additions are required. Of course, this says nothing of the relative size of the integers involved in these operations (which we expect to be smaller in the rational arithmetic). However, noting the superiority of the primitive PRS in the context of Chapter 2, we hope to make similar gains in the present one. In fact, there may be an even greater advantage to the fraction-free approach to reduction.

In practice, the primitive PRS is rarely used because of the availability of such schemes as the reduced and subresultant PRS, which remove a smaller (but more easily obtained) quantity than the full content from each pseudo-remainder. (No counterpart to these schemes currently exists for reduction.) The success of these schemes depends on the fact that they will usually remove a large proportion of the content at each stage; otherwise, the leftover content is carried into the next pseudo-remainder and produces rapid expression swell. In contrast, the linear coefficient growth in (3.10) may be well controlled if the reducing basis F consists of primitive polynomials. Therefore the overhead of content sub-computations need not be prohibitive if the required frequency of these computations is low.

Secondly, it is usually possible to remove the content from a polynomial much more cheaply than the definition (2.9) suggests. Specifically, we will examine the computation of content by a probabilistic algorithm [Mona86], and the removal of content by trial divisions. Consider a polynomial $p = \sum_{i=1}^{\pi} p_i t_i$,

where $p_i \in K$, $t_i \in T_n$. We obviously have

$$g_1 := gcd(p_1, p_2) \geq gcd(p_1,...,p_\pi) = content(p) \ .$$

However, we also note that if $g_1 \mid p_i$ for $3 \leq i \leq \pi$, then $g_1 = content(p)$. So, we might obtain the content at a cost as small as that of $(\pi-1)$ divisions; on the other hand, it will not exceed $(\pi-1)$ GCD's and

$$\sum_{k=2}^{\pi-1} (\pi-k) = (\pi - 1)\,(\frac{\pi}{2} - 1)$$

divisions. The likelihood of early success is clearly greater if p_1, p_2 are "small" compared to the rest of the coefficients. Rather than incur additional overhead by sorting, we can in practice use the fact that the head coefficient itself is often the smallest. An example of such an algorithm is as follows.

Algorithm 3.12:

procedure *content*(p)
  $g_1 \leftarrow hcoeff(p)$
  $C \leftarrow \{p_1, ..., p_\pi\} - \{g_1\}$
  for $C_i \in C$ **while** $g_1 \neq 1$ **do**
    if $g_1 \mid C_i$ **then next**
    else $g_1 \leftarrow gcd(g_1, C_i)$
  return(g_1)

Note that over $K = Q(a,b,...,c)$, for "divisibility" we require exact divisibility in $Z[a,b,...,c]$.

We now observe that by working in $Z[a,b,...,c]$ in the above case, we may use a trial division method similar to that of [Hear79] for the primitive PRS. While Hearn's scheme uses only the leading coefficients of sequence polynomials as trial divisors, it is necessary in the present case to use *ratios* of head coefficients of the reducing basis. The set of trial divisors could be obtained by multivariate factorization of the head coefficients. However, a much cheaper method is to perform the decomposition by trial divisions as well. If $H = \{hcoeff(G_1), ..., hcoeff(G_k)\}$ are the head coefficients of a basis G, we obtain a set of trial divisors as follows:

Algorithm 3.13:

$D \leftarrow \emptyset$; **for** $h \in H$ **do** $D \leftarrow extend1(D, h)$

where we define

procedure *extend1*(D_0, h)
  $D \leftarrow D_0$; $R \leftarrow \{h\} - Z$
  while $R \neq \emptyset$ **do**
    $k \leftarrow selectpoly(R)$; $R \leftarrow R - \{k\}$
    for $d \in D$ **do**
      while $d \mid k$ **and** $k \notin Z$ **do** $k \leftarrow \frac{k}{d}$
    if $k \notin Z$ **then**
      $E \leftarrow \emptyset$
      for $d \in D$ **do**
        if $k \mid d$ **then** $E \leftarrow E \cup \{d\}$; $R \leftarrow R \cup \{\frac{d}{k}\}$
      $D \leftarrow D - E \cup \{k\}$; $R \leftarrow R - Z$
  return(D)

In our experience, it seems that that the set D obtained as above (and updated during the course of Algorithm 3.8) often accounts for *all* of the content which occurs during the reduction of S-polynomials, when $<_L$ is used. So, further decomposition would only slow the trial division process. However, when $<_G$ is used we do not expect the head coefficients of (consecutive) basis polynomials to be related in the same manner. Suppose $D \subseteq \boldsymbol{Z}[a,b,...,c]$ contains only primitive polynomials which do not factor. Then it may be extended with respect to a new primitive polynomial as follows.

Algorithm 3.14:

procedure *extend2*(D, c)
  $k \leftarrow c$
  for $d \in D$ **do**
    while $d \mid k$ **and** $k \notin Z$ **do** $k \leftarrow \frac{k}{d}$
  $E \leftarrow \{$ distinct factors of $k \}$
  return$(D \cup E)$

Once again, the set of possible divisors may be updated as new polynomials are added in the course of Algorithm 3.8. Then during reduction, we may attempt to remove content by trial divisions with such a set as described in [Hear79].

At this point, a wide range of choices exists for the frequency and type of content removal in a fraction-free reduction scheme. Since each variation complicates the sub-algorithm in a slightly different way, we present only one version here. (This should make the construction of the other possible schemes clear.)

Algorithm 3.15:

procedure *reduce*(q, F)
  if $q = 0$ **then return**(0)
  $[p, \sigma, \kappa] \leftarrow Mreduce(\ q/content(q), F, 1)$
  $h \leftarrow M(p)\ ;\ p \leftarrow p - h$
  $contin \leftarrow 1$
  while $p \neq 0$ **do**
    $[r, \sigma, \kappa] \leftarrow Mreduce(p, F, contin)$
    if $\kappa \neq contin$ **then**
      $hcont \leftarrow content(h)\ ;\ \tilde{\sigma} \leftarrow hcont\ \sigma$
      $\gamma \leftarrow gcd(\tilde{\sigma}, \kappa)$
      if $\gamma \neq 1$ **then**
        $\kappa \leftarrow \dfrac{\kappa}{\gamma}\ ;\ \sigma \leftarrow \dfrac{\tilde{\sigma}}{\gamma}$
        $h \leftarrow \dfrac{h}{hcont}$
      $contin \leftarrow \kappa$
    $h \leftarrow \sigma h + \kappa M(r)\ ;\ p \leftarrow r - M(r)$
  return(h)

where we define

```
procedure Mreduce(q, F, contin)
    p ← q ; σ ← 1 ; κ ← contin ; cflag ← true
    big ← 3 max{ length(hcoeff(F_i)) | F_i ∈ F }
    R_p ← { f ∈ F s.t. hterm(f) | hterm(p) }
    while R_p ≠ ø do
        cflag ← false
        f ← selectpoly(R_p)
        u ← hterm(p)/hterm(f)
        γ ← gcd(hcoeff(p), hcoeff(f))
        m_1 ← hcoeff(f)/γ ; m_2 ← hcoeff(p)/γ
        p ← m_1 p − m_2 u f
        σ ← m_1 σ
        if p ≠ 0 then
            if length(hcoeff(p)) > big then
                pcont ← content(p) ; cflag ← true
                p ← p/pcont ; κ ← κ pcont
        else
            return( [0, σ, 0] )
        R_p ← { f ∈ F s.t. hterm(f) | hterm(p) }
    if cflag then return( [p, σ, κ] )
    else
        pcont ← content(p)
        return( [p/pcont, σ, κ pcont] )
```

With the above algorithm, we hope to remove content only when the coefficients become large compared to those of the polynomials in F. For our implementation, we have used the Maple function *length*, which returns the number of digits of an integer and the "internal path length" (of the Maple representation) of a polynomial. Other formulations worth considering are:

Algorithm 3.16: compute and remove the content (via Alg. 3.12) after each single reduction step; after secondary calls to *Mreduce*, remove any accumulated overall content as above.

Algorithm 3.17: after each reduction step, attempt to remove content by trial divisions (using divisors computed by Alg. 3.13 for $<_L$, or Alg. 3.14 for $<_G$), followed by Algorithm 3.12; at the end of each secondary *Mreduce*

call, remove any overall content as above (preceding content calculation by trial divisions).

Algorithm 3.18: attempt to remove content by trial divisions only when the coefficients become large; if the coefficients remain large, remove the left-over content by Algorithm 3.12; before the next *Mreduce* call, remove the overall content as above.

We will now compare reduction schemes on problems with both integer and polynomial coefficients, using the framework of Algorithm 3.8 with the normal selection strategy.

Problem		Reduction Algorithm		
		3.11 (monic)	3.15 (primitive)	3.16 (primitive)
2	time (sec.)	1269	282	317
	space (Kb.)	950	1106	1081
3	time	501	500	491
	space	1049	1065	1073
4(b)	time	2618	2157	2152
	space	1360	1335	1327
5(a)	time	134	52	54
	space	713	934	901
5(b)	time	7346	1725	1815
	space	1786	1630	1671
5(c)	time	63164	13351	13884
	space	3465	2580	2515
6(a)	time	1629	954	1035
	space	1081	1212	1180
6(b)	time	3401	2077	2224
	space	1114	1343	1335
6(c)	time	90759	35809	37295
	space	2802	2703	2736
7(a)	time	60002	11290	11391
	space	4762	3056	2826

Table 3.4: $<_L$ - reduction over rational numbers
(Maple 4.0 on a VAX/785 processor.)

Problem		Reduction Algorithm		
		3.11 (monic)	3.15 (primitive)	3.16 (primitive)
2	time (sec.)	57	40	40
	space (Kb.)	590	704	688
3	time	2705	2642	2595
	space	1327	1360	1368
4(b)	time	176	174	167
	space	951	983	983
5(c)	time	315	236	238
	space	893	1097	1065
6(c)	time	221	210	202
	space	958	1057	1049
7(a)	time	393	304	309
	space	926	1171	1139

Table 3.5: $<_G$ - reduction over rational numbers
(Maple 4.0 on a VAX/785 processor.)

Problem		Reduction Algorithm				
		3.11	3.15	3.16	3.17	3.18
10	time (sec.)	50	16	35	42	13
	space (Kb.)	1000	803	1032	1000	827
11	time	159	54	75	80	53
	space	1057	1098	1106	1081	1098
12	time	200	548	647	327	124
	space	1155	1589	1573	1253	1237
4(c)	time	4576	1348	1441	1781	1212
	space	1992	3270	2187	2154	2146
13(a)	time	10396	2517	2793	3467	2607
	space	2482	2428	2424	2548	2449
13(b)	time	2620	1651	1780	1953	1172
	space	2024	2646	2613	2514	2408
13(c)	time	>79655	>87438	>150000	55493	15084
	space	>16000	>16000	≥ 6209	4710	3637
14	time	1029	852	934	698	534
	space	2531	3604	3612	3285	3039
15	time	4976	4931	5104	2793	2375
	space	6914	7528	7750	7627	7512

Table 3.6: $<_L$ - reduction over rational functions
(Maple 4.1 on a VAX/8650 processor.)

Problem		Reduction Algorithm				
		3.11	3.15	3.16	3.17	3.18
10	time (sec.)	25	177	164	157	198
	space (Kb.)	844	1294	1270	1368	1335
11	time	15	16	19	23	18
	space	788	868	893	868	860
12	time	11	10	14	18	12
	space	648	688	778	860	712
4(c)	time	3536	2640	2287	2740	3089
	space	2425	2580	2449	2368	2531
13(a)	time	43	31	34	44	37
	space	910	958	950	909	926
13(b)	time	19	17	18	23	19
	space	804	885	885	868	868
13(c)	time	30	32	33	37	33
	space	868	933	934	917	909
13(d)	time	446	269	315	451	301
	space	1622	1737	1745	1728	1745
14	time	25	23	27	37	27
	space	803	892	925	868	900
15	time	27	24	30	45	30
	space	828	909	958	909	942

Table 3.7: $<_G$ - reduction over rational functions
(Maple 4.1 on a VAX/8650 processor.)

It is apparent that for $<_G$ there is little consistent difference between reduction schemes. This is largely because in this case the coefficient growth is not extreme. For example, the coefficients of the primitive polynomials produced by the algorithm during Problem 7(a) do not exceed 45 digits. By comparison, the lexicographic algorithm will produce (primitive) polynomials with 470 digit coefficients, when applied to the same problem. Hence, for $<_L$ there is a noticeable difference between the monic and primitive schemes. This may *appear* to be more pronounced in the integer case because space limitations prevent us from displaying the asymptotic behavior of the polynomial class. We mention that in the implementations of these schemes, the integer operations such as *gcd* and *content* are part of the compiled kernel of the Maple system; therefore they are relatively fast. (The integer content function is in fact similar to Algorithm 3.12.) For this reason, there is little (or no) advantage to Algorithm 3.15 over Algorithm 3.16 in this case. (The latter was used throughout the previous section.) The different details of rational polynomial arithmetic

make reduction speed somewhat more dependent on individual problem structure. Since rational function coefficients may be kept in *factored normal form* [Gedd83], their simplification may be very fast *if* they consist only of simple factors (*e.g.* Problem 12). We mention that the monic approach is much slower if the re-scaling is done more often (see [Czap86a]), or if the rationals are kept in *expanded normal form.* We also note that coefficient growth in problems with free parameters is quite often extremely rapid. So, the monic scheme may again be very slow asymptotically. Moreover, if we choose to avoid rationals, the removal of content by trial divisions can result in substantial savings by avoiding expensive multivariate GCD computations. (Univariate GCD computation is relatively efficient in Maple due to the *gcdheu* algorithm [Char84].) It seems that Algorithm 3.15 can sometimes be slower than Algorithm 3.16, since the content sub-computations (as well as the reduction arithmetic) become more complex if content is allowed to accumulate. The cost of trial divisions does not seem to be similarly affected; however, for large problems (*i.e.* with several parameters) the more conservative Algorithm 3.17 might be appropriate. Recently, some progress has been made towards the development of p-adic and modular methods for Gröbner basis computation (see [Trin84], [Wink87], resp.); for obvious reasons, this is of particular importance for $<_L$.

We will now consider the details of the procedure *selectpoly* of Algorithm 3.11 (which differ from those of its counterpart in Algorithm 3.3). Namely, the degree of freedom present when R_p has many elements leaves the problem of choosing a particular $f \in R_p$. There are two obvious possibilities: we can either choose f such that $hterm(f)$ is minimal among the headterms in R_p (as in Alg. 3.3), or choose f such that it is maximal. If we use the graduated term ordering, the former strategy is reasonable by virtue of Lemma 3.8 (and perhaps also the observation that extreme coefficient growth is less of a problem than for $<_L$). However, if we use the lexicographic ordering there are once again many reasons why this may not work well in practice.

Example: Suppose that $x >_L y$, and consider the polynomials

$$p = -3xy^4 - y^3 + 27y - 4 ,$$

and

$$f_1 = xy^3 - 27xy + 4x + y^6 + 4y^3 ;$$
$$f_2 = 81xy^2 - 12xy - 3y^7 - 12y^4 + y^3 - 27y + 4 ;$$
$$f_3 = 59001xy - 8748x - 81y^8 - 12y^7 - 2187y^6 - 324y^5 - 21y^4 - 8744y^3 - 729y^2 + 16 ;$$

compare the reduction of p modulo (in sequence) f_1, f_2, f_3 to the reduction modulo f_3 alone.

We first note that using a reducer of lower order headterm can result in a reduced object of relatively high degree (and accordingly, more terms). So, if a non-normal selection strategy is followed, the overall complexity of the algorithm may increase unless we select each reducer carefully. An approach consistent with Algorithm 3.10 would be to select f such that the total degree of the reduced object is minimal (with ties broken lexicographically). Since this essentially removes the degree of freedom in this case, we turn our attention to the normal selection strategy. Although the final result must be unique with this strategy, it is possible that the number of reduction steps may increase. We further note that since the lower order polynomials necessarily appear at a later point in the algorithm, and since there is (linear) coefficient growth associated with reduction, we expect that their coefficients will usually be relatively large. Therefore, the cost of coefficient arithmetic may also be greater if we use the lower order polynomials. (This is another reason, then, why Algorithm 3.9 may be slow.) In practice, the optimal choice will depend on p as well as R_p. However, given the apparent importance of coefficient operations in the $<_L$ reduction process, we will consider a heuristic which seeks to minimize the cost of this arithmetic.

Let us consider a reduction step (3.10) in the particular case of polynomials over the rationals (*i.e.* all coefficient arithmetic is done over Z). Suppose that p, f have π, ν terms, respectively (where $f \in R_p$) and that

$$length(hcoeff(f)) = N' , \quad length(||f||_\infty) = N ,$$
$$length(hcoeff(p)) = P' , \quad length(||p||_\infty) = P .$$

Further suppose that the times required to multiply and add integers of lengths M, N are (c_1MN) and $c_2(M+N)$, respectively, and that the time required to multiply two terms in n variables is bounded by (c_3n). (Note that, practically speaking, the degrees of all monomials should be reasonably small.) Then the

total cost of the reduction (3.10) is bounded by

$$C = c_1\pi N'P + c_1\nu P'N + c_2\ min(\nu,\ \pi)(N'P+P'N) + c_3\nu n\ . \qquad (3.11)$$

Some simple experiments reveal that for the Maple system we have $c_1 \simeq .1\ c_2$ and $c_3 \ll c_2$. In fact, since the Maple *length* function will compute a value similar to πP for a polynomial $p \in Z[x_1,...,x_n]$, we can (roughly) model polynomial arithmetic in much the same way; in this case, we find that $c_1 \simeq c_2$ and $c_3 \ll c_2$. Although it is an abuse of notation, we will therefore write

$$\|f\|_\infty := max\{\ length(f_i)\ \mid\ 1 \leq i \leq \nu\ \}$$

for $f = \sum_{i=1}^{\nu} f_i t_i$ where $f_i \in Z[a,b,...,c]$. Now, if $N' \simeq N$, then (3.11) is approximately

$$N[c_1\pi P + c_1\nu P' + c_2\ min(\nu,\ \pi)(P + P')\ ]\ . \qquad (3.12)$$

So, as ν becomes large, only the contribution of the second term in (3.12) increases. Apparently, the cost is more strongly dependent on N than ν. We might therefore use a quantity such as

$$N\nu^{1/2} \simeq (NN'\nu)^{1/2}$$

as a heuristic measure of the cost of using f for the reduction. If $N' \ll N$, then (3.11) is approximately

$$c_1P'\nu N + c_2\ P'min(\nu,\ \pi)N\ . \qquad (3.13)$$

Noting that NN' is not much larger than N, and that $P' \leq P$ (*i.e.* (3.13) is smaller than (3.12)), the quantity $NN'\nu$ again seems appropriate. We therefore define

$$complexity(f)\ :=\ length(\|f\|_\infty)\ length(hcoeff(f))\ \nu\ . \qquad (3.14)$$

Then if the reduction basis F is sorted in order of ascending complexity, $selectpoly(R_p)$ will obtain the heuristically optimal reducer for p by taking the first available element of R_p. If $\nu \gg \pi$ or $\nu \ll \pi$, the quantities νN and N', respectively, are better measures of the reduction cost. However, there is no way to take this into account without considering also the object p. Moreover, if p is an S-polynomial formed from the basis containing f, we expect that $\pi \sim O(\nu)$.

Let us now compare reduction strategies for Algorithm 3.8 (with the normal selection strategy), using a reducer based on Algorithm 3.16. The reduction basis can be sorted:

(a) such that the headterms are $<_L$-ascending,

(b) such that the headterms are $<_L$-descending,

(c) such that the complexities (3.14) are ascending.

In order to ensure that the same input reaches the main loop in each case, the initial pre-reductions are carried out using the descending order only.

Problem		Basis Order		
		$<_L$ - ascending	$<_L$ - descending	(3.14) - ascending
2	time (sec.)	570	350	280
	space (Kb.)	1164	1172	1057
3	time	646	632	636
	space	1221	1222	1122
4(b)	time	3164	2858	1979
	space	1417	1417	1319
4(d)	time	85271	128724	47175
	space	4072	4956	3396
5(c)	time	10911	13753	6061
	space	2687	2507	2376
6(c)	time	19490	43448	17195
	space	2556	2744	2742
7(a)	time	7675	12000	5826
	space	2458	3202	3080
9	time	119770	207602	90238
	space	7600	9636	8296

Table 3.8: the effect of basis sorting on $<_L$-reduction, integer coefficients (Maple 4.0 on a VAX/785 processor.)

Problem		Basis Order		
		$<_L$ - ascending	$<_L$ - descending	(3.14) - ascending
10	time (sec.)	1378	37	36
	space (Kb.)	2859	1057	1032
11	time	287	77	82
	space	1196	1106	1171
12	time	49616	636	678
	space	3457	1548	1597
4(c)	time	1378	1639	1182
	space	1975	2252	2097
13(a)	time	1903	2583	1181
	space	1991	2482	1802
13(b)	time	1226	1821	1019
	space	1500	2539	1728
13(c)	time	22579	>150000	18822
	space	3216	≥ 5844	2990
14	time	876	1038	1016
	space	2499	3604	2949
15	time	4523	5697	5377
	space	6300	7766	7600

Table 3.9: the effect of basis sorting on $<_L$-reduction, polynomial coefficients (Maple 4.1 on a VAX/8650 processor.)

In constructing the above heuristic, we have neglected the fact that Algorithm 3.16 actually removes a GCD before cross-multiplication. Nonetheless, it seems to work well when one $<_L$-based strategy or the other performs poorly. Since the choice of a reducer with small coefficients limits coefficient growth, such an approach also helps to minimize the cost of content removal after reduction. Note that for problems with polynomial coefficients, the result of using even slightly bigger coefficients can be quite dramatic. This presents yet another reason why variants of the algorithm which do not (in general) remove the "redundant" polynomials are preferable when $<_L$ is used.

In the sequel, then, we will use [reduction] Algorithms 3.16 and 3.18 for problems with integer and polynomial coefficients, respectively; the reduction bases will be sorted according to the heuristic. We re-iterate, however, that a completely different type of reduction heuristic is suggested for a non-normal

selection strategy. Also, our experience suggests that for $<_G$, relatively little improvement over a $<_G$-ascending basis can be obtained using the heuristic, when fraction-free arithmetic is used. It then turns out to be somewhat more convenient to use Algorithm 3.11 in this case.

Chapter 4: Solving Algebraic Equations

4.1. Gröbner Bases and Systems of Algebraic Equations

We have already discussed the applications of Gröbner bases in algebraic simplification. It has also been shown how they may be used in the computation of polynomial GCD's and factorization [Gian85]. However, our primary point of interest (and historically, among the original applications) is their usefulness in solving systems of algebraic equations. In [Buch70], it was shown that by transforming an arbitrary basis of a [zero-dimensional] polynomial ideal into a Gröbner basis, certain questions about the solvability of the associated algebraic system could be answered. Buchberger's method was later applied by Trinks [Trin78] to determine certain solutions of a system of six equations in six unknowns. With the continued study of Gröbner bases, it became clear that many of Buchberger's results were applicable also to ideals which were not necessarily zero-dimensional. Nonetheless, it is noted in [Pohs81] that the methods were not as suitable in practice as was originally hoped. These authors therefore examined the combined use of Buchberger's results with the classical elimination theory. However, their ideas do not seem to have been pursued further. Instead, the improvements made possible by Buchberger's criteria (see Chapter 3) and his own consolidation of the various Gröbner basis techniques [Buch85] have allowed a number of fairly difficult systems to be solved in [Boge86]. In what follows, we speak interchangeably of F as either a basis for an ideal in $K[x_1,...,x_n]$, or the associated system of algebraic equations over K.

The main principle underlying the results of this section is that if $<F>=<G>$, then the common zeros of the sets F and G are identical. This alone is of little use, since the ideals are, in general, infinite sets. Hence, the construction of a Gröbner basis becomes quite useful.

Theorem 4.1 ([Buch85]): Let G be the reduced Gröbner basis of F. Then F is solvable (*i.e.* $\exists$ $(\alpha_1,...,\alpha_n)$ in an algebraic extension of K such that $f(\alpha_1,...,\alpha_n) = 0$ for all $f \in F$) iff $1 \notin G$.

Proof: Clearly the system is inconsistent iff there exists a combination of the original polynomials which equals 1. This is equivalent to the requirement that $1 \in <F>$. However, $<F>=<G>$, which by the definition of Gröbner bases implies that $reduce(1, G) = 0$. This is only possible if $1 \in G$. In fact, since G is, by assumption, reduced we must also have $G = \{1\}$ □

Roughly stated, the *dimension* of $<F>$ is the number of independent parameters upon which the solutions of F must depend. (See [Waer53] for a more precise definition.)

Theorem 4.2 ([Buch85]): Let G be the reduced Gröbner basis of F, and let H be the set

$$H := \{ hterm(g) \mid g \in G \}.$$

Then F has finitely many solutions iff for all $1 \leq i \leq n$, $\exists\ k$ such that $x_i^k \in H$.

Proof: Consider the set U defined in (3.7), which we recall forms a linearly independent vector space basis for $\boldsymbol{K}[x_1,...,x_n]/<F>$. Clearly this basis is finite iff H has the above property. But the quotient structure is finite-dimensional (as a vector space) iff $<F>$ is zero-dimensional, *i.e.* F has finitely many solutions □

Example: For the polynomials

$$F := \{x^3yz - xz^2,\ xy^2z - xyz,\ x^2y^2 - z^2\}$$

we find that using $<_L$ (where $x >_L y >_L z$) the reduced Gröbner basis is

$$G := \{x^2y^2 - z^2,\ x^2yz - z^3,\ x^2z^2 - z^4,\ xy^2z - xyz,$$
$$xyz^2 - xz^2,\ xz^3 - xz^2,\ yz^3 - z^3,\ z^5 - z^4\}.$$

Since $1 \notin G$, the system does possess solutions. However, since the requirements of Theorem 4.2 are not met, there are infinitely many.

Example: For the polynomials

$$F := \{3x^2y - y^3 - 4,\ x^2 + xy^3 - 9\}$$

we find that with $<_G$ (where $x >_G y$) the reduced Gröbner basis is

$$G := \{3x^2y - y^3 - 4,\ 9x^4 - y^4 + 12xy^2 - 81x^2 + 27y^2 - 4y,$$
$$y^5 + 3x^3 + 4y^2 - 27x,\ xy^3 + x^2 - 9\}.$$

Therefore there exist finitely many solutions.

We note two important features of the above examples. First, they are independent of the choice of term ordering. Second, neither result requires that the solutions themselves be constructed. This latter point may be quite important in practice, for the methods for constructing the solutions *do* depend on the term ordering chosen (and may not be practical even if the Gröbner basis construction is; see Example 8 of Section 4.4).

Let us first examine the method for finding the solutions when the total degree ordering is used. We assume that our system F has only *finitely many* solutions, so that the reduced Gröbner basis G satisfies the conditions of Theorem 4.2. Then for each $x_l \in \{x_1, ..., x_n\}$ there must exist a univariate polynomial $r_l \in \langle F \rangle \cap K[x_l]$ which includes all possible values of x_l which may occur in a solution of F. We can, in fact, construct the smallest such polynomial using a method presented in [Buch85].

Algorithm 4.1:

procedure *finduni*(G, x)
  $k \leftarrow 0$
  do
    $p_k \leftarrow reduce2(x^k, G)$
    if $\exists\ (d_0,...,d_k) \neq (0,...,0)$ s.t. $\sum_{j=0}^{k} d_j p_j = 0$ **then**
      return$\left(\frac{1}{d_k} \sum_{j=0}^{k} d_j x^j \right)$
    else
      $k \leftarrow k + 1$

where we define

procedure *reduce2*(q, G)
  $p \leftarrow q$; $h \leftarrow 0$
  while $p \neq 0$ **do**
    $R_p \leftarrow \{ f \in G \text{ s.t. } hterm(f) \mid hterm(p) \}$
    while $R_p \neq \emptyset$ **do**
      $f \leftarrow selectpoly(R_p)$
      $p \leftarrow p - M(p)f/M(f)$
      $R_p \leftarrow \{ f \in G \text{ s.t. } hterm(f) \mid hterm(p) \}$
    $h \leftarrow h + M(p)$; $p \leftarrow p - M(p)$
  return(h)

(The procedure *reduce2* is clearly just Algorithm 3.11 with no re-scaling of the reduced object.) If we require that

$$r = \sum_{j=0}^{k} d_j x^j \in \langle F \rangle \cap K[x] ,$$

then we must have

$$reduce2(r, G) = \sum_{j=0}^{k} d_j \; reduce2(x^j, G) = 0 .$$

This condition yields a system of linear homogeneous equations (*i.e.* the coefficients of the associated polynomial in $K[x_1,...,x_n]/\langle F \rangle$), which will have a non-trivial solution iff such a polynomial r exists. We mention that if the reduced forms of the powers x^k were re-scaled by the reduction algorithm, we would not obtain the true reduced form of r. Since "monic" reduction works fairly well for $<_G$, the simplest approach is to use *reduce2*.

By constructing a univariate r_l for each l in $1 \leq l \leq n$, we obtain a finite inclusion of the roots of F. Since not all n-tuples thus defined will be solutions of the original system, it is noted in [Boge86] that the inclusion may be refined by decomposition of the ideal. This may be accomplished using [univariate] factorization, as follows.

Algorithm 4.2:

```
procedure decompG(F)
  S ← { [Gbasis(F), ∅] }
  for k ∈ {n, n−1, ..., 1} do
    T ← ∅
    for [G, U] ∈ S do  (refine G)
      r ← finduni(G, x_k)
      R ← { distinct factors of r }
      if R = {r} then
        T ← T ∪ { [G, U ∪ {r}] }
      else
        for R_i ∈ R do
          T ← T ∪ { [Gbasis(G ∪ {R_i}), U ∪ {R_i}] }
    S ← T
  return(S)
```

Whether or not the ideal is fully decomposed, we may explicitly obtain the roots from a Gröbner basis (or one of its components) by a method of [Buch85]. (Note the similarity to Algorithm 4.2.)

Algorithm 4.3:

procedure *solveG*(F)
  $S \leftarrow \{\ [F, \emptyset]\ \}$
  for $k \in \{n, n-1, ..., 1\}$ **do**
    $T \leftarrow \emptyset$
    for $[G, (\alpha_{k+1}, ..., \alpha_n)] \in S$ **do**
      $H \leftarrow \{\ g(x_1, ..., x_k, \alpha_{k+1}, ..., \alpha_n)\ \mid\ g \in G\ \}$
      $H \leftarrow Gbasis(H)$
      $r \leftarrow finduni(H, x_k)$
      if $r \neq 1$ **then**
        $T \leftarrow T \cup \{\ [H, (\alpha, \alpha_{k+1}, ..., \alpha_n)]\ \mid r(\alpha) = 0\ \}$
    $S \leftarrow T$
  $U \leftarrow \emptyset$
  for $[H, (\alpha_1, ..., \alpha_n)] \in S$ **do**
    $U \leftarrow U \cup \{\ (\alpha_1, ..., \alpha_n)\ \}$
  return(U)

Of course, it will not always be possible to solve the univariate polynomials exactly. Also, the refinement of each Gröbner basis may not be practical since it might involve very complicated extensions of $\boldsymbol{K}$. (Note, however, the method suggested by Example 6.10, p. 6.22 of [Buch85].) For such reasons, we will not consider Algorithm 4.3 further here. Still, these methods are obviously of great practical and theoretical interest. In Table 4.1 we give some indication of the cumulative cost of the ideal decomposition computed in Algorithm 4.2.

We note that Algorithm 4.1 is not always applicable, since it requires that the ideal be zero-dimensional. It is also possible that a Gröbner basis for a "nearly triangular" system in many variables may be impossible to compute using $<_G$. And, even if the basis can be found using $<_G$, it may happen that the construction of the univariate polynomials is difficult because of the intrinsic structure of the ideal (*i.e.* its solutions) itself. (This occurs with Problem 9, for example.)

Problem		compute Gröbner basis	first refinement	second refinement
2	time (sec.)	14	54	144
	space (Kb.)	582	967	1040
5(a)	time	48	238	291
	space	893	1155	1270
6(c)	time	54	198	407
	space	950	1163	1253
7(a)	time	99	670	5239
	space	926	1573	2351
9	time	231	4584	>150000
	space	1163	2351	≥6259

Table 4.1: decomposition times for $<_G$-Gröbner bases (Maple 4.1 on a VAX/8650 processor.)

The above limitations require that we also consider the use of $<_L$-Gröbner bases for determining the solutions - in spite of the difficulty of computing such bases. Given the relationship with classical elimination techniques, the nature of these methods should already be fairly clear. This is made more precise by the following result (which is similar to Lemma 6.8 in [Buch85]).

Theorem 4.3: Let $<_T$ be a total ordering on T_n which is such that $s <_T t$ for all $s \in T_n \cap K[x_k,...,x_n]$, $t \in T_n - K[x_k,...,x_n]$, and let G be a Gröbner basis with respect to $<_T$. Then

$$<G> \cap K[x_k,...,x_n] \;=\; < G \cap K[x_k,...,x_n] > .$$

Proof: Take $f \in <G> \cap K[x_k,...,x_n]$. Since G is a Gröbner basis, $f \overset{*}{\to}_G 0$. But since f contains only the variables x_k, ..., x_n this means that there exist $p_i \in K[x_k,...,x_n]$, $g_i \in G^{(k)} := G \cap K[x_k,...,x_n]$ such that

$$f \;=\; \sum_{i=1}^{m} p_i g_i \;,$$

which implies that $f \in <G^{(k)}>$.

Conversely, if G is a Gröbner basis with respect to $<_T$, then $G^{(k)}$ must also be a Gröbner basis, since the premise clearly requires that

$$Spoly(p, q) \underset{G^{(k)}}{\Rightarrow} 0$$

for all $p, q \in G^{(k)} \subseteq G$ (*i.e.* by Theorem 3.2, part (3)) □

The meaning of the above result is that for any such term ordering (*e.g.* one which is lexicographic in at least the first $k-1$ variables), the *k-th elimination ideal* of G is generated precisely by those polynomials in G which only depend on the last $(n-k+1)$ variables. For an arbitrary set F, this means that by transforming to a $<_L$-Gröbner basis G (which has the same roots, since $<F> = <G>$), we may obtain the roots from an equivalent but triangular system. Consider, for example, the conditions of Theorem 4.2. These dictate that if $<F>$ is zero-dimensional, G will contain a single univariate polynomial (*i.e.* the polynomial in $<F> \cap \boldsymbol{K}[x_n]$ of minimal degree), and at least one polynomial in each elimination ideal for which the appropriate variable is separated. In fact, each subset $G \cap \boldsymbol{K}[x_k,...,x_n]$ may contain other polynomials which are not fully separated; however, because these are $<_L$-Gröbner bases it follows that no simpler (*i.e.* more fully separated) basis for $<F>$ exists. The extent of separation will generally increase if we decompose the basis as in Algorithm 4.2.

Algorithm 4.4:

procedure *decompL*(F)
  $S \leftarrow Gbasis(F)$; $T \leftarrow \emptyset$
  while $S \neq \emptyset$ **do**
    $p \leftarrow selectpoly(S)$; $S \leftarrow S - \{p\}$
    $P \leftarrow \{$ distinct factors of p $\}$
    if $P = \{p\}$ **then**
      $T \leftarrow T \cup \{p\}$
    else
      return($\bigcup_{P_i \in P}$ *decompL*($S \cup \{P_i\} \cup T$))
  return($\{[T]\}$)

We may then determine the roots through a "back-solving" process analogous to that used with resultants.

Algorithm 4.5:

procedure *solveL*(F)
  $S \leftarrow \{ [F, \emptyset] \}$
  for $k \in \{n, n-1, ..., 1\}$ **do**
    $T \leftarrow \emptyset$
    for $[G, (\alpha_{k+1}, ..., \alpha_n)] \in S$ **do**
      $H \leftarrow \{ g(x_k, \alpha_{k+1}, ..., \alpha_n) \mid g \in G^{(k)} - \boldsymbol{K}[x_{k+1}, ..., x_n] \}$
      $H \leftarrow Gbasis(H)$
      $r \leftarrow selectpoly(H \cap \boldsymbol{K}[x_k])$
      if $r \neq 1$ **then**
        $T \leftarrow T \cup \{ [G, (\alpha, \alpha_{k+1}, ..., \alpha_n)] \mid r(\alpha) = 0 \}$
    $S \leftarrow T$
  $U \leftarrow \emptyset$
  for $[G, (\alpha_1, ..., \alpha_n)] \in S$ **do**
    $U \leftarrow U \cup \{ (\alpha_1, ..., \alpha_n) \}$
  return(U)

We mention that, in principle, this is equivalent to Algorithm 4.3. However, after substituting each partial root, determination of the next univariate polynomial only requires refinement of the next elimination ideal. Since this is a univariate sub-problem, it is actually equivalent to computing the GCD of the polynomials in H. In fact it has recently been shown ([Gian87], [Kalk87]) that even this refinement is superfluous (since the GCD must appear directly in H).

It is apparent that the solution procedure used for $<_L$ is simpler than that required for $<_G$. This may be unimportant if computation of the $<_L$ basis for a given zero-dimensional ideal is not possible. However, the elimination property of $<_L$-Gröbner bases means that the back-solving procedure will be applicable even when there are infinitely many solutions. If, during back-solving, there are not finitely many solutions in a particular variable x_k, we simply continue back-solving for x_{k-1} in the extended domain $\boldsymbol{K}(x_k)[x_1, ..., x_{k-1}]$. That is, by transforming to a $<_L$-Gröbner basis, we make the reduction to a zero-dimensional ideal transparent.

Example: For the polynomials of Problem 4(b), the (reduced) Gröbner basis with respect to the lexicographic ordering based on $q >_L c >_L p >_L d$ is

$$\begin{aligned}
\{&157032q - 41791pd^6 + 79963pd^4 + 1831614pd^2 + 1750356p \\
&\quad + 41791d^6 - 79963d^4 - 1831614d^2 - 1907388 , \\
&3965058cp - 3965058c + 1609747pd^7 + 12400589pd^5 + 26405460pd^3 \\
&\quad + 15017706pd - 1609747d^7 - 12400589d^5 - 26405460d^3 - 15017706d , \\
&785160p^2 + 3760661pd^6 + 28249264pd^4 + 53588400pd^2 + 30005208p \\
&\quad - 3760661d^6 - 28249264d^4 - 53588400d^2 - 30790368 , \\
&529pd^8 + 4472pd^6 + 11244pd^4 + 11232pd^2 + 3888p \\
&\quad - 529d^8 - 4472d^6 - 11244d^4 - 11232d^2 - 3888\} .
\end{aligned}$$

We observe that the system has infinitely many solutions; however, the Gröbner basis property guarantees that this is the most refined result possible for this permutation of variables. (Recall that when using resultants, it is not always clear if a "reduced system" is possible or not.) Then, except for finitely many values of d, we may solve the last polynomial for p; this yields the solutions

$$\{q - 1, p - 1\} .$$

Otherwise, d must be a root of a univariate polynomial of degree 8. Decomposing the basis by Algorithm 4.4 yields the corresponding solutions in the form of the fully separated component

$$\begin{aligned}
\{&785160q - 3342751d^6 - 71904540d^2 - 29048894d^4 - 50649408 , \\
&3965058c - 1609747d^7 - 12400589d^5 - 26405460d^3 - 15017706d , \\
&785160p + 3760661d^6 + 28249264d^4 + 53588400d^2 + 30790368 , \\
&529d^8 + 4472d^6 + 11244d^4 + 11232d^2 + 3888\} .
\end{aligned}$$

In contrast to the resultant method, the degree of the univariate polynomial which must be solved at each stage of Algorithm 4.5 can be no larger than the number of solutions. It is also important to note (following [Pohs81]) that the extra freedom allowed by computing the elimination in smaller steps (*i.e.* by S-polynomials) means that the intermediate results, too, may be of much lower degree that those of the resultant method. (Compare the degree bounds in Table 3.3 with the corresponding examples in Chapter 2, for example.) So, while triangulation using resultants (when possible) may be much faster than computing a $<_L$-Gröbner basis, the latter may be possible when the former is intractable. And, the difficulty of back-solving will certainly be lower for Gröbner bases than for the reduced systems obtained via resultants. (In many cases, it amounts to no more than a substitution.)

In Table 4.2, we give some examples of the cumulative cost (using the normal selection strategy) of decomposition of $<_L$-Gröbner bases.

Problem		compute Gröbner basis	first refinement	full refinement
2	time (sec.)	73	84	85
	space (Kb.)	1057	1057	1057
5(a)	time	14	35	36
	space	940	940	940
6(c)	time	4521	4542	4551
	space	2768	2768	2768
7(a)	time	1564	1584	1585
	space	3096	3096	3096
9	time	23840	24763	24765
	space	8828	8828	8828

Table 4.2: decomposition times for $<_L$-Gröbner bases (Maple 4.1 on a VAX/8650 processor.)

When the ultimate goal of computing a $<_L$-Gröbner basis is to solve a system of algebraic equations, we must consider the problem of choosing a "suitable" permutation of variables x_1, ..., x_n. Unlike the resultant method, we must fix the *entire* ordering before starting the elimination. But since it is impossible to predict the structure of the results *after* reductions begin, it is extremely difficult to choose a good permutation *a priori*. We will therefore briefly discuss a heuristic for this purpose which is due to [Boge86].

Definition: The *incidence polynomial* in a variable x_k corresponding to a polynomial $f = \sum_{i=1}^{\phi} a_i t_i$ (where $a_i \in \boldsymbol{K}$, $t_i \in T_n$) is

$$p_k(x_k) := g(1,...,1,x_k,1,...,1)$$

where $g := \sum_{i=1}^{\phi} t_i$. For a set F, the incidence polynomial in x_k is

$$P_k := \sum_{f \in F} p_k(x_k) .$$

Note that these are polynomials over the natural numbers. A total ordering of terms therefore induces an ordering of these univariate polynomials by:

$$\begin{aligned} & p <_U q \qquad \Leftrightarrow \\ & degree(p) < degree(q) , \text{ or} \\ & \quad degree(p) = degree(q) \text{ and } hcoeff(p) < hcoeff(q), \text{ or} \\ & \quad M(p) = M(q) \text{ and } p - M(p) <_U q - M(q) . \end{aligned}$$

Definition: An ordering of variables $x_1 >_L ... >_L x_n$ is *heuristically optimal* for the polynomials in $F \subseteq \boldsymbol{K}[x_1,...,x_n]$ iff

$$P_1(y) <_U P_2(y) <_U \cdots <_U P_n(y) .$$

Example: For the polynomials

$$F := \{3x^2y - y^3 - 4,\ x^2 + xy^3 - 9\}$$

we find that

$$\begin{aligned} P_x(x) &= 2x^2 + 1x + 3 , \\ P_y(y) &= 2y^3 + 0y^2 + 1y + 3 . \end{aligned}$$

For example, y occurs in the system to the power 3 a total of 2 times. Therefore, the heuristically optimal permutation is $x >_L y$.

The idea behind the heuristic is simple: it makes sense to try to eliminate first those variables which occur in the lowest degree, and in the fewest places. In [Boge86], a number of examples are given in which the $<_L$-basis can be computed with respect to the heuristically optimal order. We add that this optimization involves only very simple computations. Clearly this type of tool is essential for the application of the $<_L$-Gröbner basis method in an automated solver, or when the system F contains many variables. However, it seems that this heuristic is most useful in avoiding very bad permutations, as opposed to selecting very favourable ones: it is not difficult to find examples where other permutations are better. For example, the permutation used in Problem 6(c) of Table 3.1 is heuristically optimal, but the permutations of 6(a), 6(b) are much better. Also, the non-optimal permutation used in Problem 4(d) of Table 3.8 is the only one in which the $<_L$-basis may be computed in practice.

A serious limitation of the heuristic is that it cannot determine the true lexicographic structure of its input. As an example, consider the system

$$F := \{x^2 + yz + 1,\ y^2 + xz^3 + 1,\ z^2 + xy + 1\}.$$

The incidence polynomials

$$\begin{aligned} P_x &= x^2 + 2x + 6\,, \\ P_y &= y^2 + 2y + 6\,, \\ P_z &= z^3 + z^2 + z + 6\,, \end{aligned}$$

suggest either the permutation $x >_L y >_L z$ or $y >_L x >_L z$. In fact, computing the basis in the first requires 28 S-polynomial reductions; in the second, only 21 reductions (and roughly half the time) are needed. It is not difficult (in hindsight) to see why the system

$$\{y^2 + xz^3 + 1,\ yx + z^2 + 1,\ yz + x^2 + 1\}$$

is simpler in the ordering $y >_L x >_L z$ than is

$$\{x^2 + yz + 1,\ xy + z^2 + 1,\ xz^3 + y^2 + 1\}$$

in $x >_L y >_L z$.

While the case of equal incidence polynomials is not uncommon, such considerations of structure can completely outweigh those of the heuristic. For example, consider the system

$$F := \{x^3 + xyz + 1,\ y^2 + xz + y + 1,\ z^2 + y + 1\}.$$

The heuristically optimal permutation is $z >_L y >_L x$. However, it is actually much easier to compute the basis using $x >_L y >_L z$! In this permutation, the input contains a very simple polynomial (*i.e.* one of low order) which has a bounding effect on all reductions which occur during the algorithm.

Another limitation (which is not, strictly speaking, intrinsic to this heuristic) is that we cannot tell *a priori* what structure the basis will take once reductions begin. It can happen that after a few reductions (*e.g.* after the prereduction step in Algorithm 3.8), another application of the heuristic reveals that some other permutation may be more suitable. (For example, the permutation used in Problem 6(b) is obtained in this way from Problem 6(c).) We conclude that, in practice, several factors will influence which permutations are truly optimal. If several permutations are tried (*e.g.* heuristically optimal ones), then inspection of the degrees of intermediate results will suggest which is most favourable.

4.2. The Use of Multivariate Factorization

We have seen that the lexicographic Gröbner basis method for solving algebraic equations cannot (and should not) be abandoned. Since the efficiency of the algorithm can be significantly improved (by improving the sub-algorithms on which it depends), it may not be quite as impractical compared to the total degree method as was previously thought. We now examine the possibility of improving the process further, by adapting Buchberger's algorithm to the specific goal of computing the decomposed elimination ideals. In this section, we consider $<_L$ exclusively.

Suppose that during the execution of the main loop of Algorithm 3.8, we obtain a non-zero polynomial

$$f = f_1^{e_1} f_2^{e_2} \cdots f_n^{e_n} , \tag{4.1}$$

where the f_i are distinct and do not factor. We immediately note the following results.

Lemma 4.4: Let f be as in (4.1). Then

$$M(f) = \prod_{i=1}^{m} M(f_i)^{e_i} .$$

Proof: It is sufficient to show that $M(f) = M(p)M(q)$ when $f = pq$. Define $R(s) := s - M(s)$, so that

$$p = M(p) + R(p) , \quad q = M(q) + R(q) ,$$

and

$$f = M(p)M(q) + M(p)R(q) + M(q)R(p) + R(p)R(q) .$$

Then

$$\begin{aligned} hterm(f) &= hterm(M(p)M(q)) \\ &>_L max_{<_L}\{ hterm(M(p)R(q)), hterm(M(q)R(p)) \} \\ &>_L hterm(R(p)R(q)) . \end{aligned}$$

Clearly, the head coefficient will also be $hcoeff(M(p)M(q))$ □

Theorem 4.5: If $g \overset{*}{\gtrdot}_G f = \prod_{i=1}^{m} f_i^{e_i}$, then for $1 \leq i \leq m$,

$$g \overset{*}{\gtrdot}_{G \cup \{f_i\}} 0 .$$

Proof: Again, it suffices to show that if $f = pq$, then $f \gtrdot_p 0$. By Lemma 4.4, we have $M(f) = M(p)M(q)$. Therefore, f is M-reducible modulo p:

$$\begin{aligned} f &\gtrdot_p f - M(f)\frac{p}{M(p)} \\ &= M(p)M(q) + M(p)R(q) + M(q)R(p) + R(p)R(q) \\ &\qquad - \frac{M(p)M(q)}{M(p)}(M(p) + R(p)) \\ &= p\,R(q) \ . \end{aligned}$$

We see that this reduction cancels exactly the monomials of $p\,M(q)$. Therefore the result (if non-zero) is still M-reducible modulo p. But since reduction is Noetherian, we must therefore obtain 0 (in a finite number of steps) □

The significance of this result is that, having obtained a factorization (4.1) during the course of the algorithm, we may consider each factor in turn as a subproblem - in which none of the information computed to that point need be re-computed. Given the exponential character of the algorithm (and its dependence on the degree of the input), we expect that the time required to compute the Gröbner bases corresponding to all of the factors f_i will be smaller than the time needed to compute the single basis using f. Since the ultimate goal of computing the Gröbner basis is to obtain the *decomposed* elimination ideals, we can in effect use a "divide-and-conquer" approach without the need to reconstruct the results.

Let us assume, for the moment, that the polynomials in F do not factor before or during the pre-reduction phase. (The *reduceset* procedure can obviously be modified to make use of factorization at a later time.) Then we can compute a partial decomposition into component Gröbner bases using the following variant of Algorithm 3.8.

Algorithm 4.6:

procedure *partdecomp*(F)
  solbases $\leftarrow \emptyset$
  splitGbasis(F)
  return(*solbases*)

where we define

procedure *splitGbasis*(F, *newpoly*, A, Q)
  if #arguments $= 1$ **then** (top level call)
    $G \leftarrow$ *reduceset*(F) ; $k \leftarrow$ *length*(G) ; $R \leftarrow \emptyset$
    $B \leftarrow \{ [i, j] \mid 1 \leq i < j \leq k$ and *criterion1*$([i, j], G) \}$
  else (lower level call with one new polynomial)
    $G \leftarrow F \cup \{$*newpoly*$\}$; $k \leftarrow$ *length*(F)$+1$; $R \leftarrow Q$
    $B \leftarrow A \cup \{ [i, k] \mid 1 \leq i < k$ and *criterion1*$([i, k], G) \}$
  while $B \neq \emptyset$ **do**
    $[[i, j], B, Q] \leftarrow$ *select2*(B, G) ; $R \leftarrow R \cup Q$
    if $B \neq \emptyset$ **then**
      $B \leftarrow B - \{[i, j]\}$
      if *hterm*$(G_j) \mid$ *hterm*(G_i) **then** $R \leftarrow R \cup \{G_i\}$
      else if *hterm*$(G_i) \mid$ *hterm*(G_j) **then** $R \leftarrow R \cup \{G_j\}$
      $f \leftarrow$ *reduce*(*Spoly*(G_i, G_j), G)
      if $f = \prod_{i=1}^{m} h_i^{e_i} \neq 0$ **then**
        $H \leftarrow \{h_1, ..., h_m\}$
        if $f \neq h_1$ **then** (create sub-problems for distinct factors)
          for $h_i \in H$ **do**
            splitGbasis(G, h_i, B, R)
        else (no factorizations; proceed as usual)
          $G \leftarrow G \cup \{f\}$; $k \leftarrow k+1$
          $B \leftarrow B \cup \{ [i, k] \mid 1 \leq i < k$ and *criterion1*$([i, k], G) \}$
  solbases $\leftarrow$ *solbases* $\cup \{$*reduceset*$(G - R)\}$
  return

Of course, we know neither the expected frequency of successful factorizations, nor the cost of unsuccessful attempts. However, it turns out that in practice the polynomials produced split into factors (often non-monomial) reasonably often. Furthermore, since fast heuristic irreducibility testing is possible [Mona87], the overhead of unsuccessful factorizations need not be great. We

also find, though, that some factors (particularly those which correspond to extraneous roots) will tend to re-appear at several points in the algorithm. This could (potentially) cause an exponential increase in the number of sub-problems created, and thereby significantly slow the algorithm. We will therefore attempt to limit the number of sub-problems as follows. Suppose that f yields the distinct factors $\{f_1, ..., f_m\}$. Then we create m sub-problems, namely

$$\{f_1 = 0\} \ ,$$

$$\{f_1 \neq 0 \ , \ f_2 = 0\} \ ,$$

$$\vdots$$

$$\{f_1 \neq 0 \ , \ ... \ , \ f_{m-1} \neq 0 \ , \ f_m = 0\} \ .$$

Each sub-problem then contains a list of *non-zero* polynomials which can be rejected if they appear at a later point. Since the simplest (*e.g.* linear) factors seem most likely to re-appear, the list of factors of f can be sorted so that the smallest factors are treated first. We mention that this approach is also followed in the solver of [Gonn86]; there it is further suggested that the list of non-zero quantities may be initialized at the start of the algorithm (*e.g.* if only non-trivial solutions are desired).

We now recall that in Section 3.3, we found that full inter-reduction of the basis polynomials is not usually desirable. And, a suitable heuristic to determine *when* to inter-reduce the basis may be difficult to construct. However, an obvious special case, for which limited inter-reduction is reasonable, would be the appearance of a univariate polynomial. We note the following result.

Theorem 4.6: Suppose $p \in F \cap \boldsymbol{K}[x_j]$ where $F \subseteq \boldsymbol{K}[x_1,...,x_n]$, and that p does not factor. Then if $<F> \neq <1>$, p is the polynomial in $<F> \cap \boldsymbol{K}[x_j]$ of lowest degree.

Proof: Suppose the minimal polynomial is $q \in <F> \cap \boldsymbol{K}[x_j]$ such that $0<degree(q)<degree(p)$. Then since p, q are univariate, p is M-reducible modulo q. Now, it cannot happen that

$$p \overset{+}{\rightarrow}_q r \in <F> \cap \boldsymbol{K}[x_j]$$

where $0<degree(r)<degree(q)$, since by assumption q is minimal. Suppose then that

$$p \stackrel{+}{\rightarrow}_q 0 \ .$$

Then there must exist $a_i \in K$ and numbers $i \in N_0$ such that

$$p = \sum_{i=0}^{l} a_i \ (x_j)^i \ q = q \ (\sum_{i=0}^{l} a_i \ (x_j)^i \ ) \ ,$$

which contradicts the assumption that p does not factor □

Corollary 4.7: If $<F>$ contains univariate polynomials (in the same variable) p, q such that $degree(p) > degree(q)$ and p does not factor, then $<F> = <1>$.

When a factorization succeeds in Algorithm 4.6, it is not uncommon for some of the factors to be univariate. This (in contrast to Algorithm 3.8) is true even if $<F>$ is not zero-dimensional, since the factor may be extraneous or correspond to a sub-problem (ideal component) with finitely many solutions. Moreover, if and when univariates appear during Algorithm 3.8, they are usually of degree comparable to the rest of the basis. The above result implies that any univariates obtained during Algorithm 4.6 are *either* minimal or extraneous. If another such univariate appears, the sub-problem is extraneous and may be abandoned without further computation. In any case, the partial basis should definitely be *post*-reduced modulo the univariate, since it will be minimal for that sub-problem. An improved version of the algorithm is as follows.

Algorithm 4.7:

procedure *partdecomp*(F)
  solbases $\leftarrow \emptyset$; *nonzero* $\leftarrow \emptyset$; $\tilde{F} \leftarrow$ *reduceset*(F)
  for j **from** *1* **to** n **do**
    if $\exists \ p \in \tilde{F} \cap K[x_j]$ **then** $U_j \leftarrow$ true **else** $U_j \leftarrow$ false
  splitGbasis($\tilde{F}$, *nonzero*, U)
  return(*solbases*)

where we define

procedure *splitGbasis*(F, *nonzero*, T, *newpoly*, A, Q)
  $U \leftarrow T$
  if #arguments $= 3$ **then**
    $G \leftarrow F$; $k \leftarrow length(G)$; $R \leftarrow \emptyset$
    $B \leftarrow \{\, [i, j] \mid 1 \leq i < j \leq k$ and $criterion1([i, j], G) \,\}$
  else
    $G \leftarrow F \cup \{newpoly\}$; $k \leftarrow length(F)+1$; $R \leftarrow Q$
    $B \leftarrow A \cup \{\, [i, k] \mid 1 \leq i < k$ and $criterion1([i, k], G) \,\}$
  while $B \neq \emptyset$ **do**
    $[\, [i, j], B, Q \,] \leftarrow select2(B, G)$; $R \leftarrow R \cup Q$
    if $B \neq \emptyset$ **then**
      $B \leftarrow B - \{[i, j]\}$
      if $hterm(G_j) \mid hterm(G_i)$ **then** $R \leftarrow R \cup \{G_i\}$
      else if $hterm(G_i) \mid hterm(G_j)$ **then** $R \leftarrow R \cup \{G_j\}$
      $f \leftarrow reduce(\, Spoly(G_i, G_j), G)$
      if $f = \prod_{i=1}^{m} h_i^{e_i} \neq 0$ **then**
        if $\exists\ j$ such that ($f \in K[x_j]$ **and** U_j) **then return**
        $H \leftarrow \{h_1, ..., h_m\}$
        if $f \neq h_1$ **then**
          for $h_m \in H - nonzero$ **do**
            if $\exists\ j$ such that $h_m \in K[x_j]$ **then**
              if U_j **then next** h_m
              else $V \leftarrow U$; $V_j \leftarrow$ true
              $E \leftarrow \emptyset$
              for $g \in G - R$ **do**
                $E \leftarrow E \cup \{\, reduce(g, \{h_m\}) \,\}$
              if $1 \notin E$ **and** $E \cap nonzero = \emptyset$ **then**
                $splitGbasis(E \cup \{h_m\}, nonzero \cup \{h_1, ..., h_{m-1}\}, V)$
            else
              $splitGbasis(G, nonzero \cup \{h_1, ..., h_{m-1}\}, U, h_m, B, R)$
        else
          $G \leftarrow G \cup \{f\}$; $k \leftarrow k+1$
          $B \leftarrow B \cup \{\, [i, k] \mid 1 \leq i < k$ and $criterion1([i, k], G) \,\}$
          if $\exists\ j$ such that $f \in K[x_j]$ **then** $U_j \leftarrow$ true
  $solbases \leftarrow solbases \cup \{reduceset(G - R)\}$
  return

We mention that the post-reduction could be approached differently. For example, the set of redundant polynomials "R" need not be removed before post-reduction. The algorithm will then be more space-efficient (for reasons discussed earlier). However, the number of subsequent 0-reductions (and the cost of the post-reductions) will also tend to be greater.

We will now compare the algorithms 4.6 and 4.7 with the standard algorithm 3.8, using the normal selection strategy and [optimized] reduction algorithms 3.16 and 3.18 (for integer and polynomial coefficient problems, respectively). All of the timings were made using Maple version 4.1 on a VAX/8650 processor.

We see that when the normal selection strategy is used, the use of factorization during the course of the algorithm can significantly improve its efficiency. Since new factorizations may be possible after post-reduction, the decomposition is not necessarily complete. However, in practice very little additional decomposition is required. We observe that the changes made in Algorithm 4.7 do indeed yield a practical improvement to this approach. The number of subproblems considered is controlled not only by storing non-zero polynomials, but through the special handling of univariate factors. (See also [Czap86b] and [Czap87] for some different comparisons.) We also note that Algorithm 4.7 is slightly less sensitive than Algorithm 3.8 to permutations of the variable ordering. Even for systems with finitely many solutions, this approach is now competitive with that based on the total degree term ordering (*cf.* Table 4.1).

Problem		Algorithm		
		3.8	4.6	4.7
2	sub-problems	1	3	3
	time (sec.)	73	64	66
	space (Kb.)	1057	1245	1351
3	sub-problems	1	13	7
	time	128	93	55
	space	1073	1232	1106
4(b)	sub-problems	1	14	7
	time	468	265	152
	space	1311	1704	1564
4(d)	sub-problems	1	56	15
	time	11506	3642	2841
	space	3359	2974	2736
5(a)	sub-problems	1	3	3
	time	14	41	36
	space	940	1140	1073
5(b)	sub-problems	1	5	5
	time	352	214	146
	space	1737	1548	1548
5(c)	sub-problems	1	9	7
	time	1504	692	397
	space	2417	2023	1974
6(a)	sub-problems	1	9	6
	time	218	182	145
	space	1188	1434	1540
6(c)	sub-problems	1	31	12
	time	4521	1518	818
	space	2768	2761	2548
7(a)	sub-problems	1	5	5
	time	1564	717	579
	space	3096	2458	2613
9	sub-problems	1	16	3
	time	23840	3783	2152
	space	8828	3940	4036

Table 4.3: the effect of factorization in $<_L$-Gröbner bases (integer coefficients)

Problem		Algorithm		
		3.8	4.6	4.7
10	sub-problems	1	1	1
	time (sec.)	13	14	14
	space (Kb.)	827	999	999
11	sub-problems	1	2	2
	time	60	73	73
	space	1171	1253	1204
12	sub-problems	1	3	3
	time	127	140	69
	space	1245	1442	1180
4(c)	sub-problems	1	5	3
	time	832	730	960
	space	2007	1925	2012
13(a)	sub-problems	1	12	8
	time	1035	222	160
	space	1900	1400	1368
13(b)	sub-problems	1	6	4
	time	730	52	48
	space	1819	1065	1139
13(c)	sub-problems	1	12	8
	time	2638	492	198
	space	2335	1507	1384
14	sub-problems	1	3	2
	time	698	70	63
	space	3252	1344	1302
15	sub-problems	1	3	2
	time	3529	149	141
	space	8364	1785	1785

Table 4.4: the effect of factorization in $<_L$-Gröbner bases (polynomial coefficients)

4.3. A Variant Algorithm

We have seen that the use of factorization during Buchberger's Algorithm yields a significant improvement when the lexicographic term ordering is used. However, to this point we have not exploited the freedom to apply different methods to distinct sub-problems, when a system subdivision occurs. It is in this context that hybrid methods such as that of [Pohs81] seem most attractive. For example, when a polynomial of low degree appears (*e.g.* as a factor of a larger polynomial), it would be reasonable to continue elimination of the appropriate variable using pseudo-remaindering. In this section, we discuss a different type of hybrid approach, and suggest reasons why it may be useful for a particular class of problems.

In particular, consider systems in which the polynomials are *dense*. For such input, the degree and coefficient growth during elimination is extreme - particularly when resultants are used. Therefore, even an apparently modest system of four equations in four variables (Problem 7(b)) cannot be triangulated in this way. Using Algorithm 4.7 (with the normal selection strategy), progress is somewhat better; however, the limits of the algorithm remain modest. In problems such as the example above, one finds that the difficulty lies primarily in sub-problems which correspond to extraneous factors. We recall that in Algorithm 3.8, if f is a reduced S-polynomial formed from elements of G, then $f \in <G>$. However, if $f = \prod_{i=1}^{m} h_i$, we have three possibilities for each factor:

(a) $h_i \in <G>$;

(b) $<G> \subset <G, h_i> \neq <1>$;

(c) $<G, h_i> = <1>$.

That is, the factor may be consistent, part of a larger ideal, or extraneous. It seems that when the underlying ideal is $<1>$, the subsequent rate of coefficient growth is often the worst possible. Otherwise, the constraint of ideal membership usually requires that the intermediate polynomials have a certain structure (*e.g.* the coefficients in an M-chain of polynomials will occasionally get smaller). It would therefore seem preferable to continue the lexicographic elimination only on sub-problems corresponding to cases (a) and (b). If only consistent subsystems are considered, then Theorem 4.6 permits a slightly different approach.

Normally, the algorithm ensures maximal separation of variables by carrying out an exhaustive set of S-polynomial reductions (only some of which will be avoided by the standard criteria). Many of these reductions will occur *after* full

separation has been obtained, since the possibility still exists that $<G>=<1>$. Excluding this possibility, we can often detect full separation of variables directly.

Definition: The variable x_k is (*fully*) *separated* in the set $F \subseteq \boldsymbol{K}[x_1,...,x_n]$ if:

(a) $\exists\ p \in F \cap \boldsymbol{K}[x_k]$ which does not factor, or

(b) $\exists\ p \in F$ such that $hterm(p)=x_k$, and all x_i in $p - M(p)$ are separated.

The above requirement is a strong one, but seems to occur commonly in practice when factorization is used (and for sub-problems with finitely many solutions). By Theorem 4.6, any polynomial which contains only separated variables must reduce to zero. We may therefore ignore pairs which fail to satisfy

criterion3$([i, j], G)$ $\Leftrightarrow$

$Spoly(G_i,G_j)$ contains at least one variable which is not separated in G, and *criterion1*$([i, j], G)$.

There remains the question of how to detect the extraneous factors in practice. We recall that even for dense problems, it is often relatively easy to compute the Gröbner basis with respect to the total degree term ordering. (See Table 3.1.) If (for each subsystem) the $<_G$-Gröbner basis is known, then the factors of any new polynomial can be tested for consistency by computing the refinement of the basis with respect to each in turn. Let $<_G$-*Gbasis* and $<_G$-*reduce* denote the total degree Gröbner basis and reduction algorithms, respectively. Then a hybrid variant of Algorithm 4.7 may be constructed as follows.

Algorithm 4.8:

procedure *hdecomp*(F)
  solbases $\leftarrow \emptyset$; *nonzero* $\leftarrow \emptyset$; $T \leftarrow <_G$-*Gbasis*(F)
  if $T = \{1\}$ **then return**($\{1\}$)
  else
    hsplit(*reduceset*(F), *nonzero*, T)
    return(*solbases*)

where we have

procedure *hsplit(F, nonzero, T, newpoly, A, Q)*
  if #arguments = 3 **then**
    $G \leftarrow F$; $k \leftarrow length(G)$; $R \leftarrow \emptyset$
    $B \leftarrow \{ [i, j] \mid 1 \leq i < j \leq k$ and $criterion3([i, j], G) \}$
  else
    $G \leftarrow F \cup \{newpoly\}$; $k \leftarrow length(F)+1$; $R \leftarrow Q$
    $B \leftarrow A \cup \{ [i, k] \mid 1 \leq i < k$ and $criterion3([i, k], G) \}$
  while $B \neq \emptyset$ **do**
    $[[i, j], B, Q] \leftarrow select2(B, G)$; $R \leftarrow R \cup Q$
    if $B \neq \emptyset$ **then**
      $B \leftarrow B - \{[i, j]\}$
      if $hterm(G_j) \mid hterm(G_i)$ **then** $R \leftarrow R \cup \{G_i\}$
      else if $hterm(G_i) \mid hterm(G_j)$ **then** $R \leftarrow R \cup \{G_j\}$
      if $criterion3([i, j], G)$ **then** $f \leftarrow reduce(Spoly(G_i, G_j), G)$
      else $f \leftarrow 0$
      if $f = \prod_{i=1}^{m} h_i^{e_i} \neq 0$ **then**
        $H \leftarrow \{h_1, ..., h_m\}$
        if $f \neq h_1$ **then**
          for $h_m \in H - nonzero$ **do**
            $q \leftarrow <_G\text{-}reduce(h_m, T)$
            if $q \neq 0$ **then**
              $\tilde{T} \leftarrow <_G\text{-}Gbasis(T \cup \{q\})$
              if $\tilde{T} = \{1\}$ **then next** h_m
            else $\tilde{T} \leftarrow T$
            if $\exists\ j$ such that $h_m \in K[x_j]$ **then**
              $E \leftarrow \emptyset$
              for $g \in G - R$ **do**
                $E \leftarrow E \cup \{ reduce(g, \{h_m\}) \}$
              if $E \cap nonzero = \emptyset$ **then**
                $hsplit(E \cup \{h_m\}, nonzero \cup \{h_1, ..., h_{m-1}\}, \tilde{T})$
            else
              $hsplit(G, nonzero \cup \{h_1, ..., h_{m-1}\}, \tilde{T}, h_m, B, R)$
        else
          $G \leftarrow G \cup \{f\}$; $k \leftarrow k+1$
          $B \leftarrow B \cup \{ [i, k] \mid 1 \leq i < k$ and $criterion3([i, k], G) \}$
  $solbases \leftarrow solbases \cup \{reduceset(G - R)\}$
  return

Clearly, if new factorizations are possible after the pre-reduction phase, or after post-reduction modulo a univariate, we may miss opportunities to apply *criterion3*. Rather than complicate the statement of the algorithm, we will assume that this does not occur often enough in practice to seriously affect the efficiency of the scheme. We also mention that the separation criterion is easily implemented by using a table which contains a "flag" for each variable (and which is updated with each new polynomial). Some timings for Algorithm 4.8 are given in Table 4.5. We have used Algorithm 3.8 for the procedure $<_G$-*Gbasis*, Algorithm 3.16 for $<_G$-*reduce* (with a $<_G$-ascending basis), and Algorithm 3.16 (heuristically optimized) for lexicographic reduction.

Problem		Algorithm	
		4.7	4.8
2	time (sec.)	66	56
	space (Kb.)	1351	1212
3	time	55	912
	space	1106	1450
4(b)	time	152	249
	space	1564	1516
4(d)	time	2841	>50000
	space	2736	≥4512
5(c)	time	397	407
	space	1974	1760
6(c)	time	818	642
	space	2548	2351
7(a)	time	579	543
	space	2613	2367
7(b)	time	>30296	16135
	space	>16000	10124
8(a)	time	20475	13502
	space	10379	6796
9	time	2152	3075
	space	4036	4210

Table 4.5: a hybrid decomposition algorithm
(Maple version 4.1 on a VAX/8650 processor.)

We note that only Algorithm 4.8 is able to cope with the dense Problem 7(b) (when normal selection is used). This is of interest because most problems tend to become denser as the first few variables are eliminated. Surprisingly,

this scheme is also competitive on most of the other test problems - in spite of the fact that it must also perform a separate total degree computation. Obviously, a more efficient deterministic algorithm for checking consistency would make this approach even more competitive. And, there is probably a marginal improvement possible by making the application of factorization and the use of *criterion3* exhaustive. Finally, other variants may be useful in particular applications. For example, during a problem containing infinitely many solutions, the appearance of a sub-problem with only finitely many (as revealed through the parallel total degree computation) would allow the application of Algorithm 4.2 to that sub-problem.

4.4. The Construction of a Solver; Further Examples

Our previous results illustrate a number of ways in which the speed and power of the [lexicographic Gröbner basis] elimination method may be dramatically improved. Still, it is not difficult to find examples for which the *direct* application of Algorithm 4.7 is inadvisable. In some cases, it may be crucial to pose the problem in the correct (or optimal) manner. In others, the raw input must be pre-processed carefully before being passed to Algorithm 4.7. In this section, we describe an approach to this pre-processing which seems appropriate for the Gröbner basis method. We then describe a number of examples which help to illustrate the effective use of the range of Gröbner basis methods in practice.

Given the inherent difficulty of the elimination process it seems clear that, in general, we should attempt to simplify the raw input as much as possible. Since this essentially simple task may become cumbersome for very large systems, we must consider how it might be automated (*i.e.* performed algorithmically). For example, an initial sub-division of the input into component subsystems is recommended (except, perhaps, for the simplest of input). During this process, it is useful to consider "nonzero" information as part of each subproblem (as in Algorithm 4.7). However, it is often possible to accomplish a great deal more during the sub-division.

Consider, for example, an input set which contains the polynomial

$$x^2y + 4xy + 4y = y(x+2)^2 = 0 \ .$$

The subsystems corresponding to $\{y=0\}$ and $\{y\neq 0, x=-2\}$, respectively, should obviously be simpler than the original system. This is especially so in view of the fact that (modulo the corresponding substitutions for y, x respectively) the subsystems will contain one fewer variable. The simplification "$y=0$" can be made *before* choosing a permutation of variables (which will induce a lexicographic term ordering) for the subsystem. And, after such a simplification the chance of making a good choice via a heuristic is probably much better. Moreover, after one such substitution a number of further factorizations and substitutions may be possible. It is important to note, though, that these (trivial) substitutions are of a type that would be made during Algorithm 4.7 itself, irrespective of the permutation of variables. For example, if a system is of high degree in x and low degree in y, then a substitution such as "$x = 2y^3+1$" could have a disastrous effect. It therefore seems best to restrict

the simplifications made in pre-processing to the class of trivial linear substitutions, *i.e.* those based on polynomials of the form

$$ax_i + bx_j - c = 0 ,$$

where $a, b, c \in K$ and $x_i, x_j \in \{x_1, ..., x_n\}$. Then the simpler of the substitutions

$$x_i = \frac{c - bx_j}{a} , \quad x_j = \frac{c - ax_i}{b}$$

is a reasonable choice. Accordingly, the scheme presented below can be used to perform the initial simplification and subdivision of a system F, where the input quantities in *nonzero* cannot vanish.

Algorithm 4.9:

```
procedure subdivide(F, nonzero)
    subsystems ← ø ; irreducible ← ø
    F̃ ← ø
    for f ∈ F−{0} do
        F̃ ← F̃ ∪ { f/content(f) }
    if 1 ∈ F̃ then
        return( {[ {1}, nonzero ]} )
    else
        split(irreducible, F̃, nonzero)
    return(subsystems)
```

where we define

procedure *split*(*irreducible, reducible, nonzero*)
  $G \leftarrow$ *irreducible* ; $F \leftarrow$ *reducible*
  if $F \cap$ *nonzero* $\neq \emptyset$ **then return**
  if $1 \in F$ **then** $G \leftarrow \{1\}$; $F \leftarrow \emptyset$
  for $f = \prod_{i=1}^{m} h_i^{e_i} \in F$ **do**
    $F \leftarrow F - \{f\}$; $H \leftarrow \{h_1, ..., h_m\}$
    if $H \neq \{f\}$ **or** f trivial, linear **then**
      if $H \cap G \neq \emptyset$ **then next** f
      for $h_m \in H - nonzero$ **do**
        if h_m trivial, linear **then**
          $\tilde{G} \leftarrow \emptyset$
          for $g \in G$ **do**
            $\tilde{G} \leftarrow \tilde{G} \cup \{ reduce(g, \{h_m\}) \}$
          $D \leftarrow \tilde{G} \cap G$; $E \leftarrow \tilde{G} - G$
          for $f \in F$ **do**
            $E \leftarrow E \cup \{ reduce(f, \{h_m\}) \}$
          $E \leftarrow E - \{0\}$
          split$(D \cup \{h_m\}, E,$ *nonzero* $\cup \{h_1, ..., h_{m-1}\})$
        else
          split$(G \cup \{h_m\}, F,$ *nonzero* $\cup \{h_1, ..., h_{m-1}\})$
      return
    else
      $G \leftarrow G \cup \{f\}$
  subsystems $\leftarrow$ *subsystems* $\cup \{ [G,$ *nonzero*$] \}$
  return

The result of the above algorithm is a collection of simplified subsystems, each with a set of non-zero polynomials. For each sub-problem, we may then apply the heuristic of [Boge86] (see Section 4.1) to select a permutation of variables. Once a lexicographic ordering is defined, the subsystem may be treated in the manner of Algorithm 4.7. We point out that the pre-reduction phase can also be modified to utilize factorization in much the same way as Algorithm 4.7. The remaining details of the resulting "solver" are presented below.

Algorithm 4.10:

procedure *Gsolve*(*zero, nonzero*)
  solbases $\leftarrow \emptyset$
  subsystems $\leftarrow$ *subdivide*(*zero, nonzero*)
  for $[F, N] \in$ *subsystems* **do**
    if $F = \{1\}$ **then next**
    choose a permutation $x_1 >_L \ldots >_L x_n$
    reducedbases $\leftarrow$ *splitredset*(F, N)
    for $[\tilde{F}, \tilde{N}] \in$ *reducedbases* **do**
      for j **from** 1 **to** n **do**
        if $\exists\ p \in \tilde{F} \cap K[x_j]$ **then** $U_j \leftarrow$ true **else** $U_j \leftarrow$ false
      splitGbasis($\tilde{F}, \tilde{N}, U$)
  return(*solbases*)

where we define

procedure *splitredset*(*F, nonzero*)
  reducedbases $\leftarrow \emptyset$
  splitmin(*F, nonzero*)
  return(*reducedbases*)

procedure *splitmin*(*F, nonzero*)
  $R \leftarrow F$; $P \leftarrow \emptyset$
  while $R \neq \emptyset$ **do**
    $h \leftarrow$ *selectpoly*(R) ; $R \leftarrow R - \{h\}$
    $h \leftarrow$ *reduce*(h, P)
    if $h \in$ *nonzero* **then return**
    if $h = \prod_{i=1}^{m} h_i^{e_i} \neq 0$ **then**
      $H \leftarrow \{h_1, \ldots, h_m\}$
      if $H \neq \{h\}$ **then**
        for $h_m \in H -$ *nonzero* **do**
          splitmin($R \cup P \cup \{h_m\}$, *nonzero* $\cup \{h_1, \ldots, h_{m-1}\}$)
        return
      else
        $Q \leftarrow \{\, q \in P$ such that $M(h) \mid M(q)\,\}$
        $R \leftarrow R \cup Q$; $P \leftarrow P - Q \cup \{h\}$
  splitnorm(*P, nonzero*)
  return

procedure *splitnorm*(F, *nonzero*)
  $S \leftarrow F$; $T \leftarrow \emptyset$
  while $S \neq \emptyset$ **do**
    $h \leftarrow$ *selectpoly*(S) ; $S \leftarrow S - \{h\}$
    $h \leftarrow$ *reduce*(h, $F - \{h\}$) $= \prod_{i=1}^{m} h_i^{e_i}$
    if $h \in$ *nonzero* **then return**
    $H \leftarrow \{h_1, ..., h_m\}$
    if $H \neq \{h\}$ **then**
      for $h_m \in H -$ *nonzero* **do**
        splitmin($S \cup T \cup \{h_m\}$, *nonzero* $\cup \{h_1, ..., h_{m-1}\}$)
      return
    else
      $T \leftarrow T \cup \{ h \}$
  reducedbases $\leftarrow$ *reducedbases* $\cup$ { [T, *nonzero*] }
  return

We conclude this section with some final examples. (All of the timings were made using Maple version 4.1 on a VAX/8650 processor.)

Example 1: A system of Shadwick (Problem 16 in the Appendix). This is a very large system (65 equations in 22 variables) whose simple structure permits many initial simplifications. We find that, using Algorithm 4.10, the resulting (fairly simple) subsystems may be triangulated using the normal selection strategy and the heuristically optimal variable permutations.

Time required: 530 sec.

Space required: 1138 Kb.

Example 2: A system of Allaway (Problem 17 in the Appendix), containing 21 equations in 18 variables and 2 free parameters. This system helps to emphasize the importance of posing the problem properly. In raw form, it could not be solved by any Gröbner basis method. But, noting the simple equation

$$t_{22}t_{33} - 2a^2 t_{33} + a\, t_{22}^2 \;=\; 0\,,$$

(and the fact that $t_{22} \neq 0 \neq t_{33}$), we find that $t_{22} \neq 2a^2$. Hence, the variable t_{33} may be determined in terms of t_{22}. After removing t_{22} from the variable list, we find that Algorithm 4.9 succeeds (without losing any particular solutions).

Time required: 2257 sec.

Space required: 1442 Kb.

Example 3: A system of Filliman (Problem 18), containing 12 equations in 9 variables. A non-heuristically optimal permutation of variables (see the Appendix) is chosen to exploit the partial triangulation of the input. Still, this system could not be triangulated by a lexicographic method when using the normal selection strategy. However, with the heuristic selection strategy (and corresponding reduction strategy) we find that Algorithm 4.7 succeeds. In fact, we may find the solutions *directly* from the Gröbner basis with respect to the total degree ordering (because of special structure in this case). (The latter method is several times faster.)

Time required: 1142 sec. (for $<_G$), 6923 sec. (for $<_L$)

Space required: 1941 Kb., 5108 Kb., respectively.

Example 4: A system of Jayakrishna (Problem 19) containing 3 equations in 3 variables with 3 free parameters. Despite the small size of this system, it could not be triangluated in the normal selection strategy because of the extreme growth of the polynomial coefficients encountered. Using heuristic selection/reduction (and, of course, content removal based on trial divisions), it is easily solved by Algorithm 3.8.

Time required: 230 sec.

Space required: 1319 Kb.

Example 5: Randomly generated systems of 4 dense quadratics in 4 variables (Problem 7(b)), and 3 dense cubics in 3 variables (Problem 8(b)). As discussed earlier, such problems would require a special approach to be possible in the normal selection strategy with the lexicographic order. The univariate polynomials could possibly be computed using Algorithm 4.1; however, this process would be time-consuming (*cf.* Problem 7(a) in Table 4.1), and even if possible would not yield as elegant a representation of the solutions. Instead, we apply

the heuristic selection/reduction approach in Algorithm 3.8 to obtain the fully triangulated and separated lexicographic Gröbner bases.

Time required: 1483 sec., 9958 sec., respectively.

Space required: 2228 Kb., 3457 Kb., respectively.

Example 6: A system of Katsura (Problem 20) containing 6 equations in 6 variables. Given the success of the method used in Example 5, the same approach is applied to this problem. The structural similarity with Problem 6 allows a good (non-heuristically optimal) permutation of variables to be chosen. For example, we may precede the choice of permutation by pre-reduction modulo a linear polynomial; we then notice that some polynomials do not contain all of the remaining variables. Note that in this case the final Gröbner basis does decompose to a small extent; therefore, Algorithm 4.7 may have been slightly faster.

Time required: 7284 sec.

Space required: 3965 Kb.

Example 7: A system of Rimey (Problem 13(d)) containing 3 equations in 3 variables, with 4 free parameters. We find that the growth of the polynomial coefficients makes this problem extremely difficult, if not impossible. (See [Rime84] for comments on an unusual method of solution.) It was not possible to triangulate this system using Algorithm 4.7 in either the normal or heuristic selection strategies. Neither was it possible to compute a univariate polynomial from the total degree Gröbner basis. (Both schemes would require in excess of 16Mb. of storage.) We then note that, in the heuristic selection strategy, progress is actually quite good *until* (within the largest sub-problem) the algorithm produces a polynomial with extremely large coefficients. If we revert to the normal strategy just before this point, the rest of the triangulation is completed within our 16Mb. limit.

Time required: 27044 sec.

Space required: 15712 Kb.

Example 8: A system of Carminatti and McLenaghan (Problem 21) containing 3 equations in 3 variables. Because of the high degree and density of the input, the solutions to this system could not be computed by any means. However, in [Carm87] it is only required that the system possess at most finitely many solutions. We may then use the total degree term ordering in Algorithm 3.8, and halt the computation as soon as finiteness is established. (That is, as soon as the partial basis contains three separated headterms, we know that the final basis is either 1 or still contains three such headterms.)

Time required: 678 sec.

Space required: 2564 Kb.

Chapter 5: Conclusions

In this thesis, we have considered a pair of methods for solving systems of algebraic equations by computing Gröbner bases. Compared to the older resultant method, these schemes appear to be much more flexible, yield more refined results, and have somewhat broader practical limits. This is in spite of the fact that resultant algorithms have been well developed over the last 100 years or so. We have also seen that the more widely applicable method based on the lexicographic term ordering is extremely sensitive to a variety of factors which do not affect the correctness of the algorithm (or its practical behaviour when the graduated ordering is used). By considering it as, primarily, an *elimination* algorithm (and through careful observation), we have formulated a number of improvements which have broadened the limits and applicability of the lexicographic method. These include:

(1) A demonstration that inter-reduction of the basis polynomials during the course of the algorithm can cause a large intermediate degree growth. It follows that any variant of the algorithm which removes redundant (in the sense of Section 3.2) polynomials during the main loop will suffer to some extent from this phenomenon. It also follows that in the simpler variant of the algorithm (which is preferred), there will tend to exist many M-chains of polynomials which may lead to extra unnecessary S-polynomial reductions.

(2) A sub-algorithm for applying Buchberger's criteria (for avoiding 0-reductions) in an exhaustive manner; when used in conjunction with the "normal" S-polynomial selection strategy, this can drastically reduce the number of 0-reductions. Hence, the possible problem described above is minimized.

(3) A number of fraction-free reduction schemes which exploit not only the similarities with the primitive PRS, but also the difference in the required frequency of content removal. In particular, a trial division approach of Hearn, suitably modified, proves to be quite efficient. (Indeed, it is crucial for large problems such as Problem 13(d).)

(4) A heuristic for reducing the cost of coeffient arithmetic in the reduction sub-algorithm, when the normal S-polynomial selection strategy is used. This exploits both the large degree of freedom present in choosing a reducing polynomial (due to the presence of M-chains in the simple variant), and the relative simplicity of fraction-free (as compared to rational) arithmetic. This results in lower overall reduction times than strategies based on lexicographic headterms.

(5) A heuristic strategy for S-polynomial selection, which can result in lower degree bounds on intermediate results (and hence much lower computing time) than the normal strategy. Although not described explicitly, an analogous strategy for reduction should be used in conjunction with this scheme. While not clearly superior on *all* problems, this approach has allowed a number of previously intractable problems to be solved.

(6) The use of multivariate factorization in Buchberger's algorithm, when the ultimate goal is the solution of algebraic equations. This readily allows further improvements (which, for example, help to minimize the number of sub-problems), and suggests new variants of the algorithm. We see that in this context, some inter-reduction of the (incomplete) basis and removal of redundant elements is reasonable. Note also that this technique can be applied to the inter-reduction sub-algorithms as well.

(7) A possible algorithm for pre-processing raw input systems. By subdividing into component subsystems and exploiting (only) trivial linear substitutions, we strengthen the heuristic of [Boge86] for choosing a permutation of variables. Note that other types of preliminary substitutions may violate the heuristic; however, those which do not will likely occur *after* the ordering is imposed anyway.

It is not our intention to suggest that the use of the lexicographic term ordering is to be preferred (in general) to the graduated ordering. We have, however, broadened the class of ideals for which the computation of lexicographic Gröbner bases may be considered feasible.

It is clear that much effort has been expended in order to reduce the cost of coefficient arithmetic. Therefore, the development of complete modular or Hensel-type Gröbner basis algorithms is of obvious importance (for either term

ordering). Noting also the importance of selection strategies, further research in this area is indicated. Parallel formulations of the algorithm, as discussed briefly in [Watt85], may also offer a direct means of exploiting a number of different S-polynomial choices. Hybrid algorithms such as that of [Pohs81] (which combines S-polynomial reduction with resultants) may also prove more efficient than pure Gröbner basis methods. We mention that the above hybrid, by assuming a form similar to Buchberger's algorithm, seems to offer a more robust elimination scheme (than the one dicussed in Section 2.3) which will use S-polynomials only sparingly, in practice. On one hand, it is clear that our improvements to Buchberger's algorithm can also be applied to such a hybrid scheme. However, it seems likely that the extra flexibility (and lower degree bounds) offered by pure S-polynomial reduction will lead to superior hybrids in which this technique plays a dominant role.

Appendix: List of Test Problems

Although many of our test systems appear elsewhere, we list the entire set here (in original form) for completeness. We specify the polynomials f_i which are the left hand sides of polynomial equations of the form "$f_i = 0$"; the algebraic system is then defined as a set $F = \{f_1, ..., f_k\}$, together with a list of variables (*i.e.* those indeterminates for which we wish to solve). We then define a Gröbner basis problem by ordering the list of variables. We will write $>$ instead of the more cumbersome $>_L$ to indicate lexicographic precedence. Note that, in some cases, we have used a number of different variable permutations or different simplifications to alter the difficulty of the problem. Note also that no permutation is specified for Problem 16, since Algorithm 4.10 gives rise to a number of sub-problems with distinct variable lists.

Problem 1 [Buch85]:

$$\begin{aligned} f_1 &= 4x^2 + xy^2 - z + 1/4 \,, \\ f_2 &= 2x + y^2 z + 1/2 \,, \\ f_3 &= x^2 z - 1/2x - y^2 \,, \end{aligned}$$

using $x > y > z$.

Problem 2:

$$\begin{aligned} f_1 &= 343 + 804z - 919z^2 - 996y + 705yz - 312y^2 - 252x + 526xz \\ &\quad + 202xy - 761x^2 \,, \\ f_2 &= -869 + 443z - 175z^2 - 335y - 58yz + 893y^2 - 566x + 544xz \\ &\quad - 174xy + 915x^2 \,, \\ f_3 &= -494 + 642z + 98z^2 - 213y + 527yz - 880y^2 - 743x - 380xz \\ &\quad - 768xy - 465x^2 \,, \end{aligned}$$

using $x > y > z$.

Problem 3 [Boge86]:

$$
\begin{aligned}
f_1 &= b_1 + b_2 + b_3 - (a + b)\,, \\
f_2 &= b_2c_2 + b_3c_3 - (1/2 + 1/2b + b^2 - ab)\,, \\
f_3 &= b_2c_2^2 + b_3c_3^2 - (1/3a + ab^2 - 4/3b - b^2 - b^3)\,, \\
f_4 &= b_3a_{32}c_2 - (a(1/6+1/2b+b^2) - 2/3b - b^2 - b^3)\,, \\
f_5 &= b_2c_2^3 + b_3c_3^3 - (1/4 + 1/4b + 5/2b^2 + 3/2b^3 + b^4 - a(b+b^3))\,, \\
f_6 &= b_3c_3a_{32}c_2 - (1/8 + 3/8b + 7/4b^2 + 3/2b^3 + b^4 - a(1/2b+1/2b^2+b^3))\,, \\
f_7 &= b_3a_{32}c_2^2 - (1/12 + 1/12b + 7/6b^2 + 3/2b^3 + b^4 - a(2/3b+b^2+b^3))\,, \\
f_8 &= 1/24 + 7/24b + 13/12b^2 + 3/2b^3 + b^4 - a(1/3b+b^2+b^3)\,,
\end{aligned}
$$

using $b_1 > a_{32} > b_2 > b_3 > a > c_3 > c_2 > b$.

Problem 4 [Czap86a]:

$$
\begin{aligned}
f_1 &= 2 - 4b + 2q + c^2 + 2b^2 - 2qb + 4pb - 2pq - 2p^2 - 2bdc - 2qc^2 \\
&\quad + 2qdc - 2pdc - 4pb^2 + 2pqb + b^2d^2 + 2qbdc + q^2c^2 - 2q^2dc \\
&\quad + q^2d^2 + 2pbdc + 2pbd^2 + 2pqdc - 2pqd^2 + p^2d^2 + 2p^2b^2 \\
&\quad - 2qbd^2 - 2pb^2d^2 - 2pqbdc + 2pqbd^2 - 2p^2bd^2 + p^2b^2d^2\,, \\
f_2 &= 2c - bc - 2bd - 2qc + qd + pc - pd + b^2d + qbc + pbc + pbd \\
&\quad - pqc + 2pqd - 2p^2d - 2pb^2d - pqbc + p^2bd + p^2b^2d\,, \\
f_3 &= 2 - 2q - b^2 - 2pq + 2p^2 + 2pb^2 - p^2b^2\,, \\
f_4 &= 4p + 3c^2 + b^2 - 4q^2 - 6bdc - 6qc^2 - 2pb^2 + 3b^2d^2 + 6qbdc + 3q^2c^2 \\
&\quad - 3q^2d^2 + 6pbdc + 6pqd^2 - 3p^2d^2 + p^2b^2 - 6pb^2d^2 - 6pqbdc + 3p^2b^2d^2\,;
\end{aligned}
$$

(a) substitute $b=2$, and use the ordering $d > q > p > c$.

(b) substitute $b=2$, and use the ordering $q > c > p > d$.

(c) use the ordering $q > c > d > p$, so that b becomes a parameter.

(d) use the ordering $c > d > q > b > p$.

Problem 5 [Trin78]:

$$f_1 = 45p + 35s - 165b - 36\,,$$

$$
\begin{aligned}
f_2 &= 35p + 40z + 25t - 27s \ , \\
f_3 &= 15w + 25ps + 30z - 18t - 165b^2 \ , \\
f_4 &= -9w + 15pt + 20zs \ , \\
f_5 &= wp + 2zt - 11b^3 \ , \\
f_6 &= 99w - 11sb + 3b^2 \ ;
\end{aligned}
$$

(a) use the ordering $w > p > z > t > s > b$.

(b) use the ordering $w > b > p > z > s > t$.

(c) use the ordering $b > t > s > w > p > z$.

Problem 6 [Boge86]:

$$
\begin{aligned}
f_1 &= u_0^2 - u_0 + 2u_1^2 + 2u_2^2 + 2u_3^2 + 2u_4^2 \ , \\
f_2 &= 2u_0u_1 + 2u_1u_2 + 2u_2u_3 + 2u_3u_4 - u_1 \ , \\
f_3 &= 2u_0u_2 + u_1^2 + 2u_1u_3 + 2u_2u_4 - u_2 \ , \\
f_4 &= 2u_0u_3 + 2u_1u_2 + 2u_1u_4 - u_3 \ , \\
f_5 &= u_0 + 2u_1 + 2u_2 + 2u_3 + 2u_4 - 1 \ ;
\end{aligned}
$$

(a) use the ordering $u_4 > u_2 > u_0 > u_3 > u_1$.

(b) use the ordering $u_4 > u_0 > u_2 > u_3 > u_1$.

(c) use the ordering $u_4 > u_0 > u_3 > u_2 > u_1$.

Problem 7:

$$
\begin{aligned}
f_1 &= 4 + 8w - 10w^2 - 10z + 7zw - 3z^2 - 3y + 6yw + 2yz + c_1y^2 \\
&\quad - 9x + 5xw - 2xz - 4xy + d_1x^2 \ , \\
f_2 &= 9 - 6w + 6w^2 - 2z + 10zw - 5z^2 + 7y + yw - 2yz + c_2y^2 \\
&\quad - 9x - 8xw - 4xz - 8xy + d_2x^2 \ , \\
f_3 &= -9 - 2w + 10w^2 - 9z + 8zw + 10z^2 - 3y - 2yw - 2yz + c_3y^2 \\
&\quad - 2x + 3xw - 9xz + 7xy + d_3x^2 \ , \\
f_4 &= -7 + 9w + 2w^2 + 3z + 10zw - 2z^2 + 8y + 4yw - 3yz + c_4y^2 \\
&\quad + 3x - 6xw + 9xy + d_4x^2 \ ;
\end{aligned}
$$

(a) substitute $c_1=c_2=c_3=c_4=d_1=d_2=d_3=d_4=0$, and use the ordering $x>y>z>w$.

(b) substitute $c_1=-8$, $c_2=6$, $c_3=-3$, $c_4=-6$, $d_1=-1$, $d_2=-5$, $d_3=6$, $d_4=-3$, and use $z>x>w>y$.

Problem 8:

$$\begin{aligned} f_1 &= 4 + 8z - 10z^2 - 10z^3 + 7y - 3yz - 3yz^2 + 6y^2 + 2y^2z - 8y^3 - 9x \\ &\quad + 5xz - 2xz^2 - 4xy - xyz + 9xy^2 - 6x^2 + 6x^2z - 2x^2y + c_1x^3 , \\ f_2 &= -5 + 7z + z^2 - 2z^3 + 6y - 9yz - 8yz^2 - 4y^2 - 8y^2z - 5y^3 - 9x \\ &\quad - 2xz + 10xz^2 - 9xy + 8xyz + 10xy^2 - 3x^2 - 2x^2z - 2x^2y + c_2x^3 , \\ f_3 &= -2 + 3z - 9z^2 + 7z^3 + 6y - 7yz + 9yz^2 + 2y^2 + 3y^2z + 10y^3 - 2x \\ &\quad + 8xz + 4xz^2 - 3xy - 6xyz + 3xy^2 - 6x^2 + 9x^2y + c_2x^3 ; \end{aligned}$$

(a) substitute $c_1=c_2=0$, and use the ordering $x>y>z$.

(b) substitute $c_1=10$, $c_2=-3$, and use $x>y>z$.

Problem 9 [Boge86]:

$$\begin{aligned} f_1 &= u_4^4 - 20/7\,a_{46}^2 , \\ f_2 &= a_{46}^2u_3^4 + 7/10\,a_{46}u_3^4 + 7/48\,u_3^4 - 50/27\,a_{46}^2 - 35/27\,a_{46} - 49/216 , \\ f_3 &= a_{46}^5u_4^3 + 7/5\,a_{46}^4u_4^3 + 609/1000\,a_{46}^3u_4^3 + 49/1250\,a_{46}^2u_4^3 \\ &\quad - 27391/800000\,a_{46}u_4^3 + 3/5\,a_{46}^6u_3u_4^2 + 3/7\,a_{46}^5u_3u_4^2 - 343/128000\,u_3^3 \\ &\quad - 1029/160000\,u_4^3 + 63/200\,a_{46}^3u_3u_4^2 + 147/2000\,a_{46}^2u_3u_4^2 \\ &\quad + 4137/800000\,a_{46}u_3u_4^2 - 7/20\,a_{46}^4u_3^2u_4 - 77/125\,a_{46}^3u_3^2u_4 \\ &\quad - 23863/60000\,a_{46}^2u_3^2u_4 - 1078/9375\,a_{46}u_3^2u_4 - 24353/1920000\,u_3^2u_4 \\ &\quad - 3/20\,a_{46}^4u_3^3 - 21/100\,a_{46}^3u_3^3 - 91/800\,a_{46}^2u_3^3 - 5887/200000\,a_{46}u_3^3 , \end{aligned}$$

using $u_4>u_3>a_{46}$.

Problem 10 [Czap86a]:

$$\begin{aligned} f_1 &= a + cy + fy^2 + x(b + ye) + dx^2 , \\ f_2 &= g + cx + 1/2ex^2 + y(h + 2fx) + jy^2 , \end{aligned}$$

using $x>y$.

Problem 11 [Czap86a]:

$$\begin{aligned} f_1 &= 9k^2\gamma + 3k\alpha\beta + 3\beta\gamma - k^2\alpha \,, \\ f_2 &= 9k\gamma + 2k\beta - 6\alpha\gamma \,, \\ f_3 &= k^3 + 9k^2\alpha^2 + 9k\beta^2 + 9\gamma^2 - mk^3\gamma \,, \end{aligned}$$

using $\alpha > \beta > \gamma$.

Problem 12:

$$\begin{aligned} f_1 &= x^2 + ayz + gx \,, \\ f_2 &= z^2 + bxy + hz \,, \\ f_3 &= y^2 + czx + ky \,, \end{aligned}$$

using $x > y > z$.

Problem 13 [Rime84]:

$$\begin{aligned} f_1 &= \lambda x + \epsilon yz - x(x^2 + \alpha y^2 + \beta z^2) \,, \\ f_2 &= \lambda y + \epsilon zx - y(y^2 + \alpha z^2 + \beta x^2) \,, \\ f_3 &= \lambda z + \epsilon xy - z(z^2 + \alpha x^2 + \beta y^2) \,; \end{aligned}$$

(a) substitute $\alpha=1$ in f_1 only, $\beta=\epsilon=\lambda=1$ throughout, and use $x > y > z$.

(b) substitute $\alpha=1$ in f_1, f_3, $\beta=1$ throughout, and use $x > y > z$.

(c) substitute $\alpha=3$, $\beta=1$, and use $x > y > z$.

(d) use the ordering $x > y > z$.

Problem 14:

$$\begin{aligned} f_1 &= x_4x_3 + 19x_2x_1 + ex_1^2 \,, \\ f_2 &= x_4 + 127x_3 + x_2 + x_1 + b \,, \\ f_3 &= 33x_3x_4 + cx_2^2 + 981x_2x_3 \,, \\ f_4 &= x_1x_3x_4 + dx_3 \,, \end{aligned}$$

using $x_4 > x_3 > x_2 > x_1$.

Problem 15:

$$
\begin{aligned}
f_1 &= x_4x_3 + ax_2x_1 + ex_1^2 \ , \\
f_2 &= x_4 + 19x_3 + x_2 + x_1 + c^3 \ , \\
f_3 &= 7x_3x_4 + cx_2^2 + 22x_2x_3 \ , \\
f_4 &= x_1x_3x_4 + dx_3 \ ,
\end{aligned}
$$

using $x_4 > x_3 > x_2 > x_1$.

Problem 16 [Gonn87]:

$$
\begin{aligned}
f_1 &= \epsilon^2 - 1 \ , \\
f_2 &= -2/5\epsilon a_{16} + a_{16}\kappa \ , \\
f_3 &= 6/5\epsilon - 2/5\epsilon a_7 + a_7\kappa \ , \\
f_4 &= 1 - 2/5a_2\epsilon + a_2\kappa \ , \\
f_5 &= a_{14}\kappa - 2/5\epsilon a_{14} \ , \\
f_6 &= -2/5\epsilon a_{15} + a_{15}\kappa \ , \\
f_7 &= a_6\kappa - 2/5a_6\epsilon - 1/5\epsilon + 2\kappa \ , \\
f_8 &= -6a_4\kappa + 2a_4\epsilon - 4/5\epsilon a_5 - 4/5\epsilon a_3 + 2a_5\kappa \ , \\
f_9 &= -4/5\epsilon a_{10} + 4\epsilon a_9 + 2a_{13}\kappa - 12a_9\kappa - 4/5\epsilon a_{13} \ , \\
f_{10} &= -18a_{19}\kappa + 6\epsilon a_{19} - 4/5\epsilon a_{18} \ , \\
f_{11} &= -4/5\epsilon a_{12} - 8/5\epsilon a_{11} + 2\epsilon a_{10} + 2a_{12}\kappa - 6a_{10}\kappa \ , \\
f_{12} &= 4\epsilon a_{18} - 8/5\epsilon a_{17} - 12a_{18}\kappa \ , \\
f_{13} &= -6a_{17}\kappa - 12/5\epsilon a_{20} + 2\epsilon a_{17} \ , \\
f_{14} &= -4/5a_6\epsilon - 14/5\epsilon + 3a_8\kappa - 6a_7\kappa - 6/5\epsilon a_8 + 2\epsilon a_7 \ , \\
f_{15} &= -4/5\epsilon a_{15} + 4\epsilon a_{16} - 12a_{16}\kappa \ , \\
f_{16} &= -8/5\epsilon a_{14} - 6a_{15}\kappa + 2\epsilon a_{15} \ , \\
f_{17} &= 2\epsilon a_{13} - 6a_{13}\kappa - 4/5\epsilon a_{12} \ , \\
f_{18} &= a_4 + 2a_3 - 5/2\epsilon a_4\kappa - 15/2a_1\kappa + 75/2\epsilon a_1\kappa^2 + 5\epsilon a_{11} - 15\epsilon a_3\kappa \ , \\
f_{19} &= -2a_3\kappa + 3/2a_4\kappa + 5\epsilon a_{10}\kappa + 5\epsilon a_{17} - 2a_{10} + 2a_9 + 4a_{11} \\
&\quad - 5\epsilon a_9\kappa - 5/2\epsilon a_4\kappa^2 \ ,
\end{aligned}
$$

$$
\begin{aligned}
f_{20} &= -6a_{18} - 43/2a_9\kappa + 16/5\epsilon a_9 + 4a_{17} + 3a_{19} + 6a_{10}\kappa - 8/5\epsilon a_{10} \\
&\quad - 15/2\epsilon a_{19}\kappa + 25\epsilon a_{18}\kappa + 75/2\epsilon a_9\kappa^2 + 4/5\epsilon a_{11} , \\
f_{21} &= -16/5\epsilon a_{18} + 48/5\epsilon a_{19} + 14a_{18}\kappa + 4/5\epsilon a_{17} + 315/2\epsilon a_{19}\kappa^2 - 153/2a_{19}\kappa , \\
f_{22} &= 3a_{10} + 2a_{11} - 3/2a_3\kappa + 5\epsilon a_4\kappa^2 + 15\epsilon a_{20} - 2a_4\kappa - 10\epsilon a_{11}\kappa \\
&\quad - 15/2\epsilon a_{10}\kappa + 15/2\epsilon a_3\kappa^2 , \\
f_{23} &= -4/5\epsilon a_{11} - 13/2a_{10}\kappa + 4a_{11}\kappa + 20a_9\kappa - 16/5\epsilon a_9 - 15\epsilon a_{18}\kappa + 12a_{20} \\
&\quad + 15/2\epsilon a_{10}\kappa^2 + 8/5\epsilon a_{10} + 6a_{18} - 6a_{17} - 30\epsilon a_9\kappa^2 + 30\epsilon a_{17}\kappa , \\
f_{24} &= -48/5\epsilon a_{19} + 20a_{17}\kappa + 32/5\epsilon a_{18} + 66a_{19}\kappa - 105\epsilon a_{19}\kappa^2 - 87/2a_{18}\kappa \\
&\quad - 4\epsilon a_{17} + 175/2\epsilon a_{18}\kappa^2 + 12/5\epsilon a_{20} , \\
f_{25} &= 3a_{10}\kappa + 5a_{17} + 1/2a_{11}\kappa + 4/5\epsilon a_9 + 1/5\epsilon a_{11} + 5\epsilon a_9\kappa^2 - 5/2\epsilon a_{11}\kappa^2 \\
&\quad + 15\epsilon a_{20}\kappa - 5\epsilon a_{10}\kappa^2 - 25/2\epsilon a_{17}\kappa - 4a_9\kappa - 2/5\epsilon a_{10} , \\
f_{26} &= -37/2a_{17}\kappa + 30a_{18}\kappa + 12/5\epsilon a_{19} + 15\epsilon a_{19}\kappa^2 - 50\epsilon a_{18}\kappa^2 - 4\epsilon a_{18} \\
&\quad + 17/5\epsilon a_{17} - 12a_{19}\kappa + 18a_{20}\kappa + 75/2\epsilon a_{17}\kappa^2 - 12/5\epsilon a_{20} , \\
f_{27} &= -3/2a_{20}\kappa - 4a_{18}\kappa - 4/5\epsilon a_{17} - 15\epsilon a_{17}\kappa^2 + 15/2\epsilon a_{20}\kappa^2 + 3/5\epsilon a_{20} \\
&\quad + 4/5\epsilon a_{18} + 5\epsilon a_{18}\kappa^2 + 8a_{17}\kappa , \\
f_{28} &= -2a_6 + a_7 - 2/5a_2\epsilon + 4a_2\kappa - 3 - 5/2\epsilon a_7\kappa + 5\epsilon a_{14} , \\
f_{29} &= -15a_7\kappa - 46/5\epsilon + 4a_{14} + 32\kappa - 8/5a_6\epsilon - 5\epsilon a_{16}\kappa + 14/5\epsilon a_7 - 6a_{15} \\
&\quad + 2a_{16} + 4a_6\kappa + 20\epsilon a_7\kappa^2 + 20\epsilon a_{15}\kappa , \\
f_{30} &= 46/5\epsilon a_{16} - 16/5\epsilon a_{15} + 120\epsilon a_{16}\kappa^2 + 12a_{15}\kappa + 4/5\epsilon a_{14} - 66a_{16}\kappa , \\
f_{31} &= -8/5\epsilon a_7 - 14\kappa + 5\epsilon + 8a_7\kappa - 6a_{14} + 2/5a_6\epsilon - 15/2\epsilon a_{15}\kappa + 3a_{15} \\
&\quad - a_6\kappa + 20\epsilon a_{14}\kappa - 10\epsilon a_7\kappa^2 , \\
f_{32} &= -34a_{15}\kappa + 26/5\epsilon a_{15} - 4\epsilon a_{14} + 60\epsilon a_{15}\kappa^2 + 16a_{14}\kappa - 32/5\epsilon a_{16} \\
&\quad + 40a_{16}\kappa - 60\epsilon a_{16}\kappa^2 , \\
f_{33} &= 4/5\epsilon a_{16} - 4a_{16}\kappa - 20\epsilon a_{15}\kappa^2 + 20\epsilon a_{14}\kappa^2 - 2\epsilon a_{15} + 13a_{15}\kappa \\
&\quad + 7/5\epsilon a_{14} + 5\epsilon a_{16}\kappa^2 - 10a_{14}\kappa , \\
f_{34} &= a_{13} - 6a_{12} + 8a_{10} - 4a_{11} - 6a_3\kappa - 17/2a_5\kappa + 12/5\epsilon a_3 - 6a_4\epsilon + 30a_4\kappa \\
&\quad + 12/5\epsilon a_5 + 15/2\epsilon a_5\kappa^2 - 30\epsilon a_{10}\kappa - 30\epsilon a_4\kappa^2 + 15\epsilon a_{12}\kappa - 5/2\epsilon a_{13}\kappa ,
\end{aligned}
$$

$$f_{35} = -8/5\epsilon a_{11} + 36/5\epsilon a_{10} - 26a_{10}\kappa + 175/2\epsilon a_{13}\kappa^2 - 111/2a_{13}\kappa + 10a_{12}\kappa + 44/5\epsilon a_{13} - 60\epsilon a_{18}\kappa - 4a_{17} - 16/5\epsilon a_{12} - 124/5\epsilon a_9 + 172a_9\kappa + 16a_{18} - 300\epsilon a_9\kappa^2 ,$$

$$f_{36} = 426a_{19}\kappa - 8/5\epsilon a_{17} - 282/5\epsilon a_{19} - 810\epsilon a_{19}\kappa^2 + 12\epsilon a_{18} - 46a_{18}\kappa ,$$

$$f_{37} = -12a_{20} + 16a_{17} - 20a_{11}\kappa + 18a_{13}\kappa + 54a_{10}\kappa - 90\epsilon a_{10}\kappa^2 + 75/2\epsilon a_{12}\kappa^2 - 49/2a_{12}\kappa - 40a_9\kappa + 4\epsilon a_{12} - 42/5\epsilon a_{10} + 32/5\epsilon a_9 - 16/5\epsilon a_{13} + 60\epsilon a_9\kappa^2 + 28/5\epsilon a_{11} - 60\epsilon a_{17}\kappa - 25\epsilon a_{13}\kappa^2 ,$$

$$f_{38} = -60a_{17}\kappa - 148/5\epsilon a_{18} - 24/5\epsilon a_{20} + 96/5\epsilon a_{19} - 120a_{19}\kappa + 76/5\epsilon a_{17} + 220a_{18}\kappa + 180\epsilon a_{19}\kappa^2 - 420\epsilon a_{18}\kappa^2 ,$$

$$f_{39} = -54/5\epsilon a_{17} + 48/5\epsilon a_{20} + 78a_{17}\kappa + 32/5\epsilon a_{18} - 40a_{18}\kappa - 150\epsilon a_{17}\kappa^2 - 42a_{20}\kappa + 60\epsilon a_{18}\kappa^2 ,$$

$$f_{40} = 14\epsilon - 48\kappa - 4a_{14} + 8a_{15} + 42/5\epsilon a_8 + 60\epsilon a_8\kappa^2 - 12a_6\kappa - 45a_8\kappa + 4a_6\epsilon + 78a_7\kappa - 30\epsilon a_{15}\kappa - 62/5\epsilon a_7 - 120\epsilon a_7\kappa^2 ,$$

$$f_{41} = 44/5\epsilon a_{15} + 268a_{16}\kappa - 188/5\epsilon a_{16} - 480\epsilon a_{16}\kappa^2 - 32a_{15}\kappa - 8/5\epsilon a_{14} ,$$

$$f_{42} = 102a_{15}\kappa + 44/5\epsilon a_{14} - 74/5\epsilon a_{15} - 32a_{14}\kappa - 40a_{16}\kappa + 32/5\epsilon a_{16} - 180\epsilon a_{15}\kappa^2 + 60\epsilon a_{16}\kappa^2 ,$$

$$f_{43} = -96a_9\kappa - 94/5\epsilon a_{13} - 210\epsilon a_{13}\kappa^2 + 126a_{13}\kappa + 64/5\epsilon a_9 - 16/5\epsilon a_{10} + 180\epsilon a_9\kappa^2 + 12a_{10}\kappa - 18a_{12}\kappa + 28/5\epsilon a_{12} + 4/5\epsilon a_{11} ,$$

$$f_{44} = 4/5\epsilon a_{17} - 288a_{19}\kappa + 540\epsilon a_{19}\kappa^2 - 32/5\epsilon a_{18} + 192/5\epsilon a_{19} + 24a_{18}\kappa ,$$

$$f_{45} = -96a_{18}\kappa + 12/5\epsilon a_{20} + 64/5\epsilon a_{18} - 32/5\epsilon a_{17} + 180\epsilon a_{18}\kappa^2 + 24a_{17}\kappa ,$$

$$f_{46} = -16/5\epsilon a_{15} + 4/5\epsilon a_{14} + 64/5\epsilon a_{16} + 180\epsilon a_{16}\kappa^2 + 12a_{15}\kappa - 96a_{16}\kappa ,$$

$$f_{47} = 2 + a_6 - 5\epsilon\kappa - 3/5a_2\epsilon - 5/2\epsilon a_6\kappa + 3/2a_2\kappa ,$$

$$f_{48} = -5/2\epsilon a_{15}\kappa - a_6\kappa - 14\kappa - 7/5\epsilon a_7 - 10\epsilon a_7\kappa^2 + 22/5\epsilon + 15/2a_7\kappa + 2/5a_6\epsilon + a_{15} ,$$

$$f_{49} = -a_{15}\kappa + 2/5\epsilon a_{15} - 20\epsilon a_{16}\kappa^2 - 11/5\epsilon a_{16} + 27/2a_{16}\kappa ,$$

$$f_{50} = -8/5\epsilon + 5/2\epsilon a_7\kappa^2 + 2/5\epsilon a_7 - 5\epsilon a_6\kappa^2 + 4a_6\kappa + 2a_{14} - 2a_7\kappa + 8\kappa - 10\epsilon\kappa^2 - 4/5a_6\epsilon - 5\epsilon a_{14}\kappa ,$$

$$f_{51} = -8/5\epsilon a_{15} + 4/5\epsilon a_{16} - 2a_{14}\kappa + 4/5\epsilon a_{14} + 5\epsilon a_{16}\kappa^2 - 15\epsilon a_{15}\kappa^2 - 4a_{16}\kappa + 10a_{15}\kappa ,$$

$$f_{52} = -\epsilon a_{14} - 2a_{15}\kappa + 2/5\epsilon a_{15} + 5/2\epsilon a_{15}\kappa^2 - 10\epsilon a_{14}\kappa^2 + 13/2a_{14}\kappa ,$$

$$f_{53} = 2a_{12} - 5a_{10} + 4a_{11} + 6a_3\kappa + 13a_5\kappa - 14/5\epsilon a_3 + 7a_4\epsilon - 36a_4\kappa - 14/5\epsilon a_5 - 15\epsilon a_5\kappa^2 + 15\epsilon a_{10}\kappa + 45\epsilon a_4\kappa^2 - 5\epsilon a_{12}\kappa ,$$

$$f_{54} = 8/5\epsilon a_{11} - 32/5\epsilon a_{10} + 20a_{10}\kappa - 35\epsilon a_{13}\kappa^2 + 25a_{13}\kappa - 2a_{12}\kappa - 22/5\epsilon a_{13} + 4a_{17} + 4/5\epsilon a_{12} + 22\epsilon a_9 - 136a_9\kappa - 10a_{18} + 30\epsilon a_{18}\kappa + 210\epsilon a_9\kappa^2 ,$$

$$f_{55} = 45\epsilon a_{19} - 300a_{19}\kappa + 8/5\epsilon a_{17} + 34a_{18}\kappa + 495\epsilon a_{19}\kappa^2 - 10\epsilon a_{18} ,$$

$$f_{56} = 12a_{20} - 10a_{17} + 20a_{11}\kappa - 4a_{13}\kappa - 51a_{10}\kappa + 75\epsilon a_{10}\kappa^2 - 25\epsilon a_{12}\kappa^2 + 18a_{12}\kappa + 22a_9\kappa - 16/5\epsilon a_{12} + 44/5\epsilon a_{10} - 4\epsilon a_9 + 4/5\epsilon a_{13} - 30\epsilon a_9\kappa^2 - 32/5\epsilon a_{11} + 30\epsilon a_{17}\kappa + 5\epsilon a_{13}\kappa^2 ,$$

$$f_{57} = 48a_{17}\kappa + 128/5\epsilon a_{18} + 24/5\epsilon a_{20} - 12\epsilon a_{19} + 66a_{19}\kappa - 68/5\epsilon a_{17} - 166a_{18}\kappa - 90\epsilon a_{19}\kappa^2 + 270\epsilon a_{18}\kappa^2 ,$$

$$f_{58} = -54/5\epsilon a_{20} + 105\epsilon a_{17}\kappa^2 + 42a_{20}\kappa - 4\epsilon a_{18} - 66a_{17}\kappa + 22a_{18}\kappa + 53/5\epsilon a_{17} - 30\epsilon a_{18}\kappa^2 ,$$

$$f_{59} = -102/5\epsilon + 62\kappa + 4a_{14} - 5a_{15} - 33/5\epsilon a_8 - 45\epsilon a_8\kappa^2 + 13a_6\kappa + 69/2a_8\kappa - 24/5a_6\epsilon - 73a_7\kappa + 15\epsilon a_{15}\kappa + 63/5\epsilon a_7 + 105\epsilon a_7\kappa^2 ,$$

$$f_{60} = -42/5\epsilon a_{15} - 210a_{16}\kappa + 166/5\epsilon a_{16} + 330\epsilon a_{16}\kappa^2 + 27a_{15}\kappa + 8/5\epsilon a_{14} ,$$

$$f_{61} = -88a_{15}\kappa - 52/5\epsilon a_{14} + 72/5\epsilon a_{15} + 34a_{14}\kappa + 22a_{16}\kappa - 4\epsilon a_{16} + 135\epsilon a_{15}\kappa^2 - 30\epsilon a_{16}\kappa^2 ,$$

$$f_{62} = 108a_9\kappa + 26/5\epsilon a_{10} - 18a_{10}\kappa - 8/5\epsilon a_{11} + 165\epsilon a_{13}\kappa^2 + 20a_{12}\kappa + 91/5\epsilon a_{13} - 180\epsilon a_9\kappa^2 - 110a_{13}\kappa - 34/5\epsilon a_{12} - 16\epsilon a_9 ,$$

$$f_{63} = -36a_{18}\kappa - 540\epsilon a_{19}\kappa^2 + 324a_{19}\kappa - 48\epsilon a_{19} - 8/5\epsilon a_{17} + 52/5\epsilon a_{18} ,$$

$$f_{64} = -24/5\epsilon a_{20} - 180\epsilon a_{18}\kappa^2 + 108a_{18}\kappa + 52/5\epsilon a_{17} - 36a_{17}\kappa - 16\epsilon a_{18} ,$$

$$f_{65} = -18a_{15}\kappa - 180\epsilon a_{16}\kappa^2 - 16\epsilon a_{16} + 108a_{16}\kappa - 8/5\epsilon a_{14} + 26/5\epsilon a_{15} .$$

Problem 17 [Char86a]:

$$
\begin{aligned}
f_1 &= (1-b)t_{20}t_{22}t_{33} + (1-b)t_{32}t_{21}t_{22} + (b-1)t_{21}^2t_{33} + (b-1)t_{31}t_{22}^2 - 8a^2t_{33} \\
&\quad + 6t_{22}^2a + (2-6b+4b^2)t_{22}t_{33} - 2t_{32}at_{22} - t_{30}t_{22}^2a + (6a-4ba)t_{33}t_{21} \\
&\quad + 4t_{20}a^2t_{33} + t_{32}t_{20}at_{22} - t_{21}t_{20}at_{33} ,
\end{aligned}
$$

$$
\begin{aligned}
f_2 &= t_{20}t_{22}t_{33} + t_{32}t_{21}t_{22} - t_{21}^2t_{33} - t_{31}t_{22}^2 - 16a^2t_{33} + 11t_{22}^2a \\
&\quad + (-9b+2b^2+5)t_{22}t_{33} - 3t_{32}at_{22} + (9a-2ba)t_{33}t_{21} + 2t_{20}a^2t_{33} ,
\end{aligned}
$$

$$
f_3 = -10a^2t_{33} + 6t_{22}^2a + (4-3b)t_{22}t_{33} - t_{32}at_{22} + 3t_{21}at_{33} ,
$$

$$
f_4 = t_{22}t_{33} - 2a^2t_{33} + t_{22}^2a ,
$$

$$
\begin{aligned}
f_5 &= -36a^2t_{33}t_{44} + 18t_{20}a^2t_{33}t_{44} - 9t_{30}t_{22}^2at_{44} - 9t_{21}t_{20}at_{33}t_{44} + 54t_{22}^2at_{44} \\
&\quad - 24t_{32}at_{22}t_{44} + 9t_{32}t_{20}at_{22}t_{44} + 2t_{32}^2at_{44} - t_{32}^2t_{20}at_{44} \\
&\quad - 2t_{31}at_{33}t_{44} + t_{31}t_{20}at_{33}t_{44} + t_{32}t_{30}at_{22}t_{44} - t_{42}t_{20}t_{33}^2a \\
&\quad + 6t_{43}at_{22}t_{33} - t_{43}t_{30}at_{22}t_{33} - 24t_{33}^2at_{22} + t_{40}t_{33}^2at_{22} + 2t_{42}t_{33}^2a \\
&\quad - 2t_{43}t_{32}at_{33} + t_{43}t_{32}t_{20}at_{33} + (b-1)t_{32}^2t_{21}t_{44} + (1-b)t_{31}t_{21}t_{33}t_{44} \\
&\quad + (1-b)t_{41}t_{33}^2t_{22} + (b-1)t_{42}t_{21}t_{33}^2 + (-24b+18b^2+6)t_{22}t_{33}t_{44} \\
&\quad + (9-9b)t_{20}t_{22}t_{33}t_{44} + (9b-9)t_{21}^2t_{33}t_{44} + (9b-9)t_{31}t_{22}^2t_{44} \\
&\quad + (b-1)t_{30}t_{22}t_{33}t_{44} + (9-9b)t_{32}t_{21}t_{22}t_{44} + (1-b)t_{32}t_{31}t_{22}t_{44} \\
&\quad + (b-1)t_{43}t_{31}t_{22}t_{33} + (1-b)t_{43}t_{32}t_{21}t_{33} + (-18ba+36a)t_{44}t_{33}t_{21} ,
\end{aligned}
$$

$$
\begin{aligned}
f_6 &= -t_{32}^2t_{21}t_{44} + t_{41}t_{33}^2t_{22} - t_{42}t_{21}t_{33}^2 - 84a^2t_{33}t_{44} + 15t_{20}a^2t_{33}t_{44} \\
&\quad - 3t_{30}t_{22}^2at_{44} - 3t_{21}t_{20}at_{33}t_{44} + 117t_{22}^2at_{44} - t_{30}t_{22}t_{33}t_{44} \\
&\quad - 44t_{32}at_{22}t_{44} + 3t_{32}t_{20}at_{22}t_{44} + t_{32}t_{31}t_{22}t_{44} + 3t_{32}^2at_{44} \\
&\quad - 3t_{31}at_{33}t_{44} + t_{31}t_{21}t_{33}t_{44} + 11t_{43}at_{22}t_{33} - t_{43}t_{31}t_{22}t_{33} - 50t_{33}^2at_{22} \\
&\quad + 3t_{42}t_{33}^2a - 3t_{43}t_{32}at_{33} + t_{43}t_{32}t_{21}t_{33} + (17-44b+15b^2)t_{22}t_{33}t_{44} \\
&\quad + (12-3b)t_{20}t_{22}t_{33}t_{44} + (3b-12)t_{21}^2t_{33}t_{44} + (3b-12)t_{31}t_{22}^2t_{44} \\
&\quad + (12-3b)t_{32}t_{21}t_{22}t_{44} + (66a-15ba)t_{44}t_{33}t_{21} ,
\end{aligned}
$$

$$
\begin{aligned}
f_7 &= (17-24b+3b^2)t_{22}t_{33}t_{44} + 3t_{20}t_{22}t_{33}t_{44} - 3t_{21}^2t_{33}t_{44} - 3t_{31}t_{22}^2t_{44} \\
&\quad + 3t_{32}t_{21}t_{22}t_{44} - 69a^2t_{33}t_{44} + 3t_{20}a^2t_{33}t_{44} + (36a-3ba)t_{44}t_{33}t_{21} \\
&\quad + 87t_{22}^2at_{44} - 24t_{32}at_{22}t_{44} + t_{32}^2at_{44} - t_{31}at_{33}t_{44} + 6t_{43}at_{22}t_{33} \\
&\quad - 35t_{33}^2at_{22} + t_{42}t_{33}^2a - t_{43}t_{32}at_{33} ,
\end{aligned}
$$

$$f_8 = (7-4b)t_{22}t_{33}t_{44} - 24a^2t_{33}t_{44} + 6t_{21}at_{33}t_{44} + 27t_{22}^2at_{44}$$
$$- 4t_{32}at_{22}t_{44} + t_{43}at_{22}t_{33} - 10t_{33}^2at_{22} ,$$
$$f_9 = t_{22}t_{33}t_{44} - 3a^2t_{33}t_{44} + 3t_{22}^2at_{44} - t_{33}^2at_{22} ,$$
$$f_{10} = - 108a^2t_{22}t_{44}t_{55} + 36t_{32}a^2t_{44}t_{55} + 18t_{30}a^2t_{22}t_{44}t_{55}$$
$$- 18t_{32}t_{20}a^2t_{44}t_{55} - 9t_{21}t_{30}at_{22}t_{44}t_{55} - 9t_{20}^2at_{33}t_{44}t_{55}$$
$$+ 54t_{21}at_{22}t_{44}t_{55} - 54t_{43}t_{22}^2at_{55} - 18t_{42}at_{22}t_{33}t_{55}$$
$$+ 9t_{21}t_{32}t_{20}at_{44}t_{55} + 216t_{22}^2at_{33}t_{55} - 9t_{40}t_{22}^2at_{33}t_{55}$$
$$- 9t_{43}t_{32}t_{20}at_{22}t_{55} - 2t_{30}at_{33}t_{44}t_{55} + 9t_{42}t_{20}at_{22}t_{33}t_{55}$$
$$+ 9t_{43}t_{30}t_{22}^2at_{55} + 24t_{43}t_{32}at_{22}t_{55} - t_{32}t_{43}t_{30}at_{22}t_{55} - 2t_{32}^2t_{43}at_{55}$$
$$+ t_{32}^2t_{43}t_{20}at_{55} + t_{30}t_{20}at_{33}t_{44}t_{55} + t_{31}t_{30}at_{22}t_{44}t_{55} + 2t_{31}t_{32}at_{44}t_{55}$$
$$- t_{31}t_{32}t_{20}at_{44}t_{55} - 24t_{32}at_{22}t_{33}t_{55} + t_{32}t_{40}at_{22}t_{33}t_{55} + 2t_{32}t_{42}at_{33}t_{55}$$
$$- t_{32}t_{42}t_{20}at_{33}t_{55} - t_{53}t_{30}at_{22}t_{33}t_{44} - 2t_{53}t_{32}at_{33}t_{44} + t_{53}t_{32}t_{20}at_{33}t_{44}$$
$$+ 24t_{54}t_{33}^2at_{22} - t_{54}t_{40}t_{33}^2at_{22} - 2t_{54}t_{42}t_{33}^2a - 120t_{33}^2at_{22}t_{44}$$
$$+ t_{50}t_{33}^2at_{22}t_{44} + 2t_{52}t_{33}^2at_{44} - t_{52}t_{20}t_{33}^2at_{44} + 6t_{53}at_{22}t_{33}t_{44}$$
$$+ t_{54}t_{42}t_{20}t_{33}^2a - 6t_{54}t_{43}at_{22}t_{33} + t_{54}t_{43}t_{30}at_{22}t_{33} + 2t_{54}t_{43}t_{32}at_{33}$$
$$- t_{54}t_{43}t_{32}t_{20}at_{33} + (24b-6-18b^2)t_{21}t_{33}t_{44}t_{55} + (9-9b)t_{21}^2t_{32}t_{44}t_{55}$$
$$+ (9b-9)t_{31}t_{21}t_{22}t_{44}t_{55} + (9b-9)t_{21}t_{20}t_{33}t_{44}t_{55} + (9b-9)t_{41}t_{22}^2t_{33}t_{55}$$
$$+ (9b-9)t_{43}t_{32}t_{21}t_{22}t_{55} + (9-9b)t_{42}t_{21}t_{22}t_{33}t_{55} + (9-9b)t_{43}t_{31}t_{22}^2t_{55}$$
$$+ (1-b)t_{21}t_{30}t_{33}t_{44}t_{55} + (b-1)t_{32}t_{43}t_{31}t_{22}t_{55} + (1-b)t_{32}^2t_{43}t_{21}t_{55}$$
$$+ (1-b)t_{31}^2t_{22}t_{44}t_{55} + (b-1)t_{31}t_{32}t_{21}t_{44}t_{55} + (1-b)t_{32}t_{41}t_{22}t_{33}t_{55}$$
$$+ (b-1)t_{32}t_{42}t_{21}t_{33}t_{55} + (1-b)t_{53}t_{32}t_{21}t_{33}t_{44} + (b-1)t_{54}t_{41}t_{33}^2t_{22}$$
$$+ (1-b)t_{51}t_{33}^2t_{22}t_{44} + (b-1)t_{52}t_{21}t_{33}^2t_{44} + (b-1)t_{53}t_{31}t_{22}t_{33}t_{44}$$
$$+ (1-b)t_{54}t_{42}t_{21}t_{33}^2 + (1-b)t_{54}t_{43}t_{31}t_{22}t_{33} + (b-1)t_{54}t_{43}t_{32}t_{21}t_{33}$$
$$+ (12a-36ba)t_{33}t_{44}t_{55} + (18ba+12a)t_{55}t_{44}t_{33}t_{20}$$
$$+ (-36a+18ba)t_{55}t_{44}t_{21}t_{32} + (12a-18ba)t_{55}t_{44}t_{22}t_{31} ,$$
$$f_{11} = - 288a^2t_{22}t_{44}t_{55} + 84t_{32}a^2t_{44}t_{55} + 15t_{30}a^2t_{22}t_{44}t_{55}$$
$$- 15t_{32}t_{20}a^2t_{44}t_{55} - 3t_{21}t_{30}at_{22}t_{44}t_{55} - 3t_{20}^2at_{33}t_{44}t_{55}$$
$$+ 117t_{21}at_{22}t_{44}t_{55} - 117t_{43}t_{22}^2at_{55} - 33t_{42}at_{22}t_{33}t_{55} + 3t_{21}t_{32}t_{20}at_{44}t_{55}$$

$$
\begin{aligned}
&+ 522t_{22}^2at_{33}t_{55} - 3t_{40}t_{22}^2at_{33}t_{55} - 3t_{43}t_{32}t_{20}at_{22}t_{55} - 3t_{30}at_{33}t_{44}t_{55} \\
&+ 3t_{42}t_{20}at_{22}t_{33}t_{55} + 3t_{43}t_{30}t_{22}^2at_{55} + t_{21}t_{30}t_{33}t_{44}t_{55} \\
&+ 44t_{43}t_{32}at_{22}t_{55} - t_{32}t_{43}t_{31}t_{22}t_{55} - 3t_{32}^2t_{43}at_{55} + t_{32}^2t_{43}t_{21}t_{55} \\
&+ t_{31}^2t_{22}t_{44}t_{55} - t_{31}t_{32}t_{21}t_{44}t_{55} + 3t_{31}t_{32}at_{44}t_{55} + t_{32}t_{41}t_{22}t_{33}t_{55} \\
&- 50t_{32}at_{22}t_{33}t_{55} - t_{32}t_{42}t_{21}t_{33}t_{55} + 3t_{32}t_{42}at_{33}t_{55} + t_{53}t_{32}t_{21}t_{33}t_{44} \\
&- 3t_{53}t_{32}at_{33}t_{44} + 50t_{54}t_{33}^2at_{22} - t_{54}t_{41}t_{33}^2t_{22} - 3t_{54}t_{42}t_{33}^2a \\
&- 274t_{33}^2at_{22}t_{44} + t_{51}t_{33}^2t_{22}t_{44} + 3t_{52}t_{33}^2at_{44} - t_{52}t_{21}t_{33}^2t_{44} \\
&- t_{53}t_{31}t_{22}t_{33}t_{44} + 11t_{53}at_{22}t_{33}t_{44} + t_{54}t_{42}t_{21}t_{33}^2 + t_{54}t_{43}t_{31}t_{22}t_{33} \\
&- 11t_{54}t_{43}at_{22}t_{33} - t_{54}t_{43}t_{32}t_{21}t_{33} + 3t_{54}t_{43}t_{32}at_{33} \\
&+ (44b-17-15b^2)t_{21}t_{33}t_{44}t_{55} + (12-3b)t_{21}^2t_{32}t_{44}t_{55} + (3b-12)t_{31}t_{21}t_{22}t_{44}t_{55} \\
&+ (3b-12)t_{21}t_{20}t_{33}t_{44}t_{55} + (3b-12)t_{41}t_{22}^2t_{33}t_{55} + (3b-12)t_{43}t_{32}t_{21}t_{22}t_{55} \\
&+ (12-3b)t_{42}t_{21}t_{22}t_{33}t_{55} + (12-3b)t_{43}t_{31}t_{22}^2t_{55} + (40a-84ba)t_{33}t_{44}t_{55} \\
&+ (15ba+22a)t_{55}t_{44}t_{33}t_{20} + (15ba-66a)t_{55}t_{44}t_{21}t_{32} \\
&+ (22a-15ba)t_{55}t_{44}t_{22}t_{31} ,
\end{aligned}
$$

$$
\begin{aligned}
f_{12} = &- 291a^2t_{22}t_{44}t_{55} + 3t_{21}^2t_{32}t_{44}t_{55} + 69t_{32}a^2t_{44}t_{55} \\
&+ 3t_{30}a^2t_{22}t_{44}t_{55} - 3t_{32}t_{20}a^2t_{44}t_{55} - 3t_{31}t_{21}t_{22}t_{44}t_{55} \\
&- 3t_{21}t_{20}t_{33}t_{44}t_{55} + 87t_{21}at_{22}t_{44}t_{55} - 87t_{43}t_{22}^2at_{55} - 18t_{42}at_{22}t_{33}t_{55} \\
&+ 465t_{22}^2at_{33}t_{55} - 3t_{41}t_{22}^2t_{33}t_{55} - 3t_{43}t_{32}t_{21}t_{22}t_{55} - t_{30}at_{33}t_{44}t_{55} \\
&+ 3t_{42}t_{21}t_{22}t_{33}t_{55} + 3t_{43}t_{31}t_{22}^2t_{55} + 24t_{43}t_{32}at_{22}t_{55} - t_{32}^2t_{43}at_{55} \\
&+ t_{31}t_{32}at_{44}t_{55} - 35t_{32}at_{22}t_{33}t_{55} + t_{32}t_{42}at_{33}t_{55} - t_{53}t_{32}at_{33}t_{44} \\
&+ 35t_{54}t_{33}^2at_{22} - t_{54}t_{42}t_{33}^2a - 225t_{33}^2at_{22}t_{44} + t_{52}t_{33}^2at_{44} \\
&+ 6t_{53}at_{22}t_{33}t_{44} - 6t_{54}t_{43}at_{22}t_{33} + t_{54}t_{43}t_{32}at_{33} + (24b-17-3b^2)t_{21}t_{33}t_{44}t_{55} \\
&+ (-69ba+51a)t_{33}t_{44}t_{55} + (3ba+12a)t_{55}t_{44}t_{33}t_{20} + (3ba-36a)t_{55}t_{44}t_{21}t_{32} \\
&+ (-3ba+12a)t_{55}t_{44}t_{22}t_{31} ,
\end{aligned}
$$

$$
\begin{aligned}
f_{13} = &(-7+4b)t_{21}t_{33}t_{44}t_{55} + (31a-24ba)t_{33}t_{44}t_{55} - 85t_{33}^2at_{22}t_{44} \\
&+ 195t_{22}^2at_{33}t_{55} - 141a^2t_{22}t_{44}t_{55} + 2t_{20}at_{33}t_{44}t_{55} + 24t_{32}a^2t_{44}t_{55} \\
&- 6t_{32}t_{21}at_{44}t_{55} + 2t_{31}at_{22}t_{44}t_{55} + 27t_{21}at_{22}t_{44}t_{55} - 27t_{43}t_{22}^2at_{55} \\
&- 3t_{42}at_{22}t_{33}t_{55} + 4t_{43}t_{32}at_{22}t_{55} + 10t_{54}t_{33}^2at_{22} - 10t_{32}at_{22}t_{33}t_{55}
\end{aligned}
$$

$$
\begin{aligned}
&+ t_{53}at_{22}t_{33}t_{44} - t_{54}t_{43}at_{22}t_{33}\ ,\\
f_{14} = &- t_{21}t_{33}t_{44}t_{55} + (9a-3ba)t_{33}t_{44}t_{55} - 15t_{33}^2at_{22}t_{44}\\
&+ 39t_{22}^2at_{33}t_{55} - 33a^2t_{22}t_{44}t_{55} + 3t_{32}a^2t_{44}t_{55}\\
&+ 3t_{21}at_{22}t_{44}t_{55} - 3t_{43}t_{22}^2at_{55} + t_{54}t_{33}^2at_{22} - t_{32}at_{22}t_{33}t_{55}\ ,\\
f_{15} = &\ t_{33}t_{44}t_{55} - 3at_{22}t_{44}t_{55} + 3t_{22}^2t_{33}t_{55} - t_{33}^2t_{22}t_{44}\ ,\\
f_{16} = &- 192a^2t_{33}t_{44}t_{55} + 432t_{22}^2at_{44}t_{55} + 96t_{20}a^2t_{33}t_{44}t_{55}\\
&- 72t_{30}t_{22}^2at_{44}t_{55} - 72t_{21}t_{20}at_{33}t_{44}t_{55} + 32t_{32}^2at_{44}t_{55}\\
&- 240t_{32}at_{22}t_{44}t_{55} + 72t_{32}t_{20}at_{22}t_{44}t_{55} - 32t_{31}at_{33}t_{44}t_{55}\\
&+ 16t_{31}t_{20}at_{33}t_{44}t_{55} + 16t_{32}t_{30}at_{22}t_{44}t_{55} - 16t_{42}t_{20}t_{33}^2at_{55}\\
&- 16t_{32}^2t_{20}at_{44}t_{55} - 384t_{33}^2at_{22}t_{55} + 16t_{40}t_{33}^2at_{22}t_{55} + 32t_{42}t_{33}^2at_{55}\\
&+ 6t_{42}at_{22}t_{44}t_{55} - t_{42}t_{30}at_{22}t_{44}t_{55} - 2t_{32}t_{42}at_{44}t_{55} + 120t_{43}at_{22}t_{33}t_{55}\\
&- 16t_{43}t_{30}at_{22}t_{33}t_{55} - 32t_{43}t_{32}at_{33}t_{55} + 16t_{43}t_{32}t_{20}at_{33}t_{55}\\
&+ 2t_{41}at_{33}t_{44}t_{55} - t_{41}t_{20}at_{33}t_{44}t_{55} + 120t_{44}^2at_{22}t_{33} - t_{50}t_{44}^2at_{22}t_{33}\\
&- 2t_{52}t_{44}^2at_{33} + t_{52}t_{20}t_{44}^2at_{33} - 6t_{53}t_{44}^2at_{22} + t_{53}t_{30}t_{44}^2at_{22}\\
&+ t_{32}t_{42}t_{20}at_{44}t_{55} - t_{43}t_{40}at_{22}t_{33}t_{55} - 2t_{43}t_{42}at_{33}t_{55} + t_{43}t_{42}t_{20}at_{33}t_{55}\\
&- 6t_{43}^2at_{22}t_{55} + t_{43}^2t_{30}at_{22}t_{55} + 2t_{43}^2t_{32}at_{55} - t_{43}^2t_{32}t_{20}at_{55}\\
&+ 2t_{53}t_{32}t_{44}^2a - t_{53}t_{32}t_{20}t_{44}^2a - 24t_{54}at_{22}t_{33}t_{44} + t_{54}t_{40}at_{22}t_{33}t_{44}\\
&+ 2t_{54}t_{42}at_{33}t_{44} - t_{54}t_{42}t_{20}at_{33}t_{44} + 6t_{54}t_{43}at_{22}t_{44} - t_{54}t_{43}t_{30}at_{22}t_{44}\\
&- 2t_{54}t_{43}t_{32}at_{44} + t_{54}t_{43}t_{32}t_{20}at_{44} + (-96ba+240a)t_{21}t_{33}t_{44}t_{55}\\
&+ (-120b+96b^2+24)t_{22}t_{33}t_{44}t_{55} + (72b-72)t_{31}t_{22}^2t_{44}t_{55}\\
&+ (72b-72)t_{21}^2t_{33}t_{44}t_{55} + (72-72b)t_{20}t_{22}t_{33}t_{44}t_{55} + (16b-16)t_{30}t_{22}t_{33}t_{44}t_{55}\\
&+ (72-72b)t_{32}t_{21}t_{22}t_{44}t_{55} + (16-16b)t_{32}t_{31}t_{22}t_{44}t_{55} + (16b-16)t_{32}^2t_{21}t_{44}t_{55}\\
&+ (16-16b)t_{31}t_{21}t_{33}t_{44}t_{55} + (16b-16)t_{42}t_{21}t_{33}^2t_{55} + (1-b)t_{40}t_{22}t_{33}t_{44}t_{55}\\
&+ (b-1)t_{41}t_{21}t_{33}t_{44}t_{55} + (16-16b)t_{41}t_{33}^2t_{22}t_{55} + (b-1)t_{42}t_{31}t_{22}t_{44}t_{55}\\
&+ (1-b)t_{32}t_{42}t_{21}t_{44}t_{55} + (16b-16)t_{43}t_{31}t_{22}t_{33}t_{55} + (16-16b)t_{43}t_{32}t_{21}t_{33}t_{55}\\
&+ (b-1)t_{43}^2t_{32}t_{21}t_{55} + (b-1)t_{51}t_{44}^2t_{22}t_{33} + (1-b)t_{52}t_{21}t_{44}^2t_{33}\\
&+ (1-b)t_{53}t_{31}t_{44}^2t_{22} + (b-1)t_{43}t_{41}t_{22}t_{33}t_{55} + (1-b)t_{43}t_{42}t_{21}t_{33}t_{55}\\
&+ (1-b)t_{43}^2t_{31}t_{22}t_{55} + (b-1)t_{53}t_{32}t_{21}t_{44}^2 + (1-b)t_{54}t_{41}t_{22}t_{33}t_{44}
\end{aligned}
$$

$$+ (b-1)t_{54}t_{42}t_{21}t_{33}t_{44} + (b-1)t_{54}t_{43}t_{31}t_{22}t_{44} + (1-b)t_{54}t_{43}t_{32}t_{21}t_{44} ,$$

$$\begin{aligned}
f_{17} = &- 496a^2t_{33}t_{44}t_{55} + 1044t_{22}^2at_{44}t_{55} + 104t_{20}a^2t_{33}t_{44}t_{55} \\
&- 42t_{30}t_{22}^2at_{44}t_{55} - 42t_{21}t_{20}at_{33}t_{44}t_{55} + 56t_{32}^2at_{44}t_{55} \\
&- 500t_{32}at_{22}t_{44}t_{55} + 42t_{32}t_{20}at_{22}t_{44}t_{55} - 56t_{31}at_{33}t_{44}t_{55} \\
&+ 4t_{31}t_{20}at_{33}t_{44}t_{55} + 4t_{32}t_{30}at_{22}t_{44}t_{55} - 4t_{42}t_{20}t_{33}^2at_{55} + t_{40}t_{22}t_{33}t_{44}t_{55} \\
&- t_{41}t_{21}t_{33}t_{44}t_{55} - 4t_{32}^2t_{20}at_{44}t_{55} - 896t_{33}^2at_{22}t_{55} + 4t_{40}t_{33}^2at_{22}t_{55} \\
&+ 56t_{42}t_{33}^2at_{55} - t_{42}t_{31}t_{22}t_{44}t_{55} + 11t_{42}at_{22}t_{44}t_{55} + t_{32}t_{42}t_{21}t_{44}t_{55} \\
&- 3t_{32}t_{42}at_{44}t_{55} + 250t_{43}at_{22}t_{33}t_{55} - 4t_{43}t_{30}at_{22}t_{33}t_{55} - 56t_{43}t_{32}at_{33}t_{55} \\
&+ 4t_{43}t_{32}t_{20}at_{33}t_{55} + 3t_{41}at_{33}t_{44}t_{55} - t_{43}^2t_{32}t_{21}t_{55} + 274t_{44}^2at_{22}t_{33} \\
&- t_{51}t_{44}^2t_{22}t_{33} - 3t_{52}t_{44}^2at_{33} + t_{52}t_{21}t_{44}^2t_{33} - 11t_{53}t_{44}^2at_{22} \\
&+ t_{53}t_{31}t_{44}^2t_{22} - t_{43}t_{41}t_{22}t_{33}t_{55} + t_{43}t_{42}t_{21}t_{33}t_{55} - 3t_{43}t_{42}at_{33}t_{55} \\
&- 11t_{43}^2at_{22}t_{55} + t_{43}^2t_{31}t_{22}t_{55} + 3t_{43}^2t_{32}at_{55} + 3t_{53}t_{32}t_{44}^2a \\
&- t_{53}t_{32}t_{21}t_{44}^2 + t_{54}t_{41}t_{22}t_{33}t_{44} - 50t_{54}at_{22}t_{33}t_{44} - t_{54}t_{42}t_{21}t_{33}t_{44} \\
&+ 3t_{54}t_{42}at_{33}t_{44} - t_{54}t_{43}t_{31}t_{22}t_{44} + 11t_{54}t_{43}at_{22}t_{44} + t_{54}t_{43}t_{32}t_{21}t_{44} \\
&+ (-104ba+500a)t_{21}t_{33}t_{44}t_{55} + (-250b+104b^2+74)t_{22}t_{33}t_{44}t_{55} \\
&+ (42b-114)t_{31}t_{22}^2t_{44}t_{55} + (42b-114)t_{21}^2t_{33}t_{44}t_{55} - 3t_{54}t_{43}t_{32}at_{44} \\
&+ (-42b+114)t_{20}t_{22}t_{33}t_{44}t_{55} + (4b-20)t_{30}t_{22}t_{33}t_{44}t_{55} \\
&+ (-42b+114)t_{32}t_{21}t_{22}t_{44}t_{55} + (20-4b)t_{32}t_{31}t_{22}t_{44}t_{55} + (4b-20)t_{32}^2t_{21}t_{44}t_{55} \\
&+ (20-4b)t_{31}t_{21}t_{33}t_{44}t_{55} + (4b-20)t_{42}t_{21}t_{33}^2t_{55} + (20-4b)t_{41}t_{33}^2t_{22}t_{55} \\
&+ (4b-20)t_{43}t_{31}t_{22}t_{33}t_{55} + (20-4b)t_{43}t_{32}t_{21}t_{33}t_{55} ,
\end{aligned}$$

$$\begin{aligned}
f_{18} = &- 480a^2t_{33}t_{44}t_{55} + 930t_{22}^2at_{44}t_{55} + 36t_{20}a^2t_{33}t_{44}t_{55} \\
&- 6t_{30}t_{22}^2at_{44}t_{55} - 6t_{21}t_{20}at_{33}t_{44}t_{55} - 4t_{30}t_{22}t_{33}t_{44}t_{55} + 28t_{32}^2at_{44}t_{55} \\
&- 350t_{32}at_{22}t_{44}t_{55} + 6t_{32}t_{20}at_{22}t_{44}t_{55} + 4t_{32}t_{31}t_{22}t_{44}t_{55} - 4t_{32}^2t_{21}t_{44}t_{55} \\
&- 28t_{31}at_{33}t_{44}t_{55} + 4t_{31}t_{21}t_{33}t_{44}t_{55} - 4t_{42}t_{21}t_{33}^2t_{55} - 760t_{33}^2at_{22}t_{55} \\
&+ 4t_{41}t_{33}^2t_{22}t_{55} + 28t_{42}t_{33}^2at_{55} + 6t_{42}at_{22}t_{44}t_{55} - t_{32}t_{42}at_{44}t_{55} \\
&+ 175t_{43}at_{22}t_{33}t_{55} - 4t_{43}t_{31}t_{22}t_{33}t_{55} - 28t_{43}t_{32}at_{33}t_{55} + 4t_{43}t_{32}t_{21}t_{33}t_{55} \\
&+ t_{41}at_{33}t_{44}t_{55} + 225t_{44}^2at_{22}t_{33} - t_{52}t_{44}^2at_{33} - 6t_{53}t_{44}^2at_{22} \\
&- t_{43}t_{42}at_{33}t_{55} - 6t_{43}^2at_{22}t_{55} + t_{43}^2t_{32}at_{55} + t_{53}t_{32}t_{44}^2a
\end{aligned}$$

$$
\begin{aligned}
&- 35t_{54}at_{22}t_{33}t_{44} + t_{54}t_{42}at_{33}t_{44} + 6t_{54}t_{43}at_{22}t_{44} - t_{54}t_{43}t_{32}at_{44} \\
&+ (-36ba+350a)t_{21}t_{33}t_{44}t_{55} + (-175b+36b^2+85)t_{22}t_{33}t_{44}t_{55} \\
&+ (6b-48)t_{31}t_{22}^2t_{44}t_{55} + (6b-48)t_{21}^2t_{33}t_{44}t_{55} \\
&+ (-6b+48)t_{20}t_{22}t_{33}t_{44}t_{55} + (-6b+48)t_{32}t_{21}t_{22}t_{44}t_{55} ,
\end{aligned}
$$

$$
\begin{aligned}
f_{19} = {} & (-4ba+100a)t_{21}t_{33}t_{44}t_{55} - 220a^2t_{33}t_{44}t_{55} + (45-50b+4b^2)t_{22}t_{33}t_{44}t_{55} \\
&- 6t_{31}t_{22}^2t_{44}t_{55} - 6t_{21}^2t_{33}t_{44}t_{55} + 6t_{20}t_{22}t_{33}t_{44}t_{55} + 6t_{32}t_{21}t_{22}t_{44}t_{55} \\
&+ 4t_{20}a^2t_{33}t_{44}t_{55} + 390t_{22}^2at_{44}t_{55} + 4t_{32}^2at_{44}t_{55} - 100t_{32}at_{22}t_{44}t_{55} \\
&- 4t_{31}at_{33}t_{44}t_{55} - 300t_{33}^2at_{22}t_{55} + 4t_{42}t_{33}^2at_{55} + t_{42}at_{22}t_{44}t_{55} \\
&+ 50t_{43}at_{22}t_{33}t_{55} - 4t_{43}t_{32}at_{33}t_{55} + 85t_{44}^2at_{22}t_{33} - t_{53}t_{44}^2at_{22} \\
&- t_{43}^2at_{22}t_{55} - 10t_{54}at_{22}t_{33}t_{44} + t_{54}t_{43}at_{22}t_{44} ,
\end{aligned}
$$

$$
\begin{aligned}
f_{20} = {} & 10t_{21}at_{33}t_{44}t_{55} - 48a^2t_{33}t_{44}t_{55} + (11-5b)t_{22}t_{33}t_{44}t_{55} + 78t_{22}^2at_{44}t_{55} \\
&- 10t_{32}at_{22}t_{44}t_{55} - 56t_{33}^2at_{22}t_{55} + 5t_{43}at_{22}t_{33}t_{55} \\
&+ 15t_{44}^2at_{22}t_{33} - t_{54}at_{22}t_{33}t_{44} ,
\end{aligned}
$$

$$
f_{21} = t_{22}t_{33}t_{44}t_{55} - 4a^2t_{33}t_{44}t_{55} + 6t_{22}^2at_{44}t_{55} - 4t_{33}^2at_{22}t_{55} + t_{44}^2at_{22}t_{33} ,
$$

(where none of t_{22}, t_{33}, t_{44}, t_{55} may vanish), using $t_{50}>t_{51}>t_{40}>t_{52}>t_{41}>t_{53}>t_{30}>t_{54}>t_{42}>t_{55}>t_{20}>t_{31}>t_{21}>t_{43}>t_{32}>t_{44}>t_{33}>t_{22}$.

Problem 18 [Fee87]:

$$
\begin{aligned}
f_1 &= a^2 + 2b^2 + e^2 - 1 , \\
f_2 &= 2c^2 + 2g^2 - 1 , \\
f_3 &= 2d^2 + f^2 + 2h^2 - 1 , \\
f_4 &= 2(cf - cd + 2dg + 2ch + 2dh) - a_la , \\
f_5 &= cf - a_lb , \\
f_6 &= 2cd - a_le , \\
f_7 &= af - ad + 2ah + de + bf - a_lc , \\
f_8 &= 2ad + 2ah - a_lg , \\
f_9 &= ce + bc - 2bg , \\
f_{10} &= ce - a_ld - ac + 2ag ,
\end{aligned}
$$

$$f_{11} = 2ac + 2bc - a_l f \ ,$$
$$f_{12} = 2ac + 2ag - a_l h \ ,$$

using $a_l > e > h > d > f > a > g > b > c$.

Problem 19 [Gonn87]:

$$f_1 = x_1 + x_2 + x_3 - a x_1 x_2 x_3 \ ,$$
$$f_2 = x_1^2 + x_2^2 + x_3^2 - b x_1^2 x_2^2 x_3^2 \ ,$$
$$f_3 = x_1^3 + x_2^3 + x_3^3 - c x_1^3 x_2^3 x_3^3 \ ,$$

using $x_1 > x_2 > x_3$.

Problem 20 [Boge86]:

$$f_1 = u_0^2 - u_0 + 2u_1^2 + 2u_2^2 + 2u_3^2 + 2u_4^2 + 2u_5^2 \ ,$$
$$f_2 = 2u_0u_1 + 2u_1u_2 + 2u_2u_3 + 2u_3u_4 + 2u_4u_5 - u_1 \ ,$$
$$f_3 = 2u_0u_2 + u_1^2 + 2u_1u_3 + 2u_2u_4 + 2u_3u_5 - u_2 \ ,$$
$$f_4 = 2u_0u_3 + 2u_1u_2 + 2u_1u_4 + 2u_2u_5 - u_3 \ ,$$
$$f_5 = 2u_0u_4 + 2u_1u_3 + 2u_1u_5 + u_2^2 - u_4 \ ,$$
$$f_6 = u_0 + 2u_1 + 2u_2 + 2u_3 + 2u_4 + 2u_5 - 1 \ ,$$

using $u_5 > u_3 > u_4 > u_2 > u_1 > u_0$.

Problem 21 [Carm87]:

$$\begin{aligned} f_1 = &- 13968xv^3 - 48384x^4 - 300504 - 232488x^2 - 1224v^4 - 2448u^2v^2 - 1224u^4 \\ &- 163176x^2u^2 - 20936x^2v^2 - 258000u^2 - 59360v^2 + 103328x^3v - 38352uv^2 \\ &+ 615400u - 38352u^3 + 56400xuv + 389008xv + 25800x^2u - 13968xu^2v \ , \end{aligned}$$

$$\begin{aligned} f_2 = &- 37854976x - 22279728v - 29636864x^3 - 5786368x^5 - 29376v^5 - 29376u^4v \\ &- 378720v^4x - 406944u^4x - 1066560x^2v^3 - 58752u^2v^3 + 5031360x^4v \\ &+ 1877920v^2x^3 - 20137056u^2x^3 - 785664u^2v^2x - 2184000x^2u^2v \\ &+ 1948640v^3 - 1723584v^2x - 551424uv^3 - 437760u^2v + 13865936x^3u \\ &+ 82308304xu - 7837248xu^3 - 551424u^3v + 18411664vx^2 + 17756592uv \end{aligned}$$

$$- 23676960u^2x + 22946672uvx^2 - 6568288xuv^2 ,$$

$$f_3 = 610786944xv^3 - 5256000u^5 + 1604140416 - 1615166080x^4 - 1394907904x^2$$
$$+ 67633216v^4 - 3297059104u + 1402343392u^2 - 36670848u^4 - 1335202752v^2$$
$$+ 152446560u^3 - 6143872u^2v^2 - 383146720x^2u^2 - 1283825280x^2v^2$$
$$- 596035648x^3v + 24176640v^3x^3 - 12908160xv^5 - 1938816u^2v^4$$
$$- 79511680x^5v + 307022080x^4v^2 - 50359680v^4x^2 - 1233792u^4v^2$$
$$- 118892160u^4x^2 - 2505599232x^4u^2 + 5451998848x^2u + 719349312uv^2$$
$$- 166993920u^2v^2x^2 - 30855168xu^2v^3 - 234542592u^2vx^3 - 281756160x^6$$
$$- 881280v^6 - 17947008xu^4v - 4756990912xv + 87472064xu^2v - 176256u^6$$
$$- 1441644576u^3x^2 - 15526848uv^4 - 20782848u^3v^2 + 805335456x^4u$$
$$+ 4259029664xuv - 196488640uv^3x - 641656256uv^2x^2 + 5376498336x^3uv$$
$$- 235852800u^3vx ,$$

using $u > x > v$.

References

[Arno84]
D. S. Arnon, G. E. Collins, S. McCallum: "Cylindrical Algebraic Decomposition I, II", SIAM J. Comp. **13**, 1984, pp. 865-877, 878-889.

[Ball81]
A. M. Ballantyne, D. S. Lankford: "New Decision Algorithms for Finitely Presented Commutative Semigroups", Comp. Math. Appl. **7**, 1981, pp. 159-165.

[Boge85]
W. Böge, R. Gebauer, H. Kredel: "Gröbner Bases Using SAC-2", Proc. EUROCAL '85, Vol. 2, Linz, April, 1985, (B. F. Caviness, ed.), Springer *Lecture Notes in Computer Science* **204**, pp. 272-274.

[Boge86]
W. Böge, R. Gebauer, H. Kredel: "Some Examples for Solving Systems of Algebraic Equations by Calculating Gröbner Bases", J. Symbolic Comp. **2**, No. 1, 1986, pp. 83-98.

[Brow71]
W. S. Brown, J. F. Traub: "On Euclid's Algorithm and the Theory of Subresultants", J. ACM **18**, No. 4, 1971, pp. 505-514.

[Brow78]
W. S. Brown: "The Subresultant PRS Algorithm", ACM TOMS **4**, 1978, pp. 237-249.

[Buch65]
B. Buchberger: "An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal (German)", Ph.D. Thesis, Univ. of Innsbruck, Math. Inst., 1965.

[Buch70]
B. Buchberger: "An algorithmical criterion for the solvability of algebraic systems of equations (German)", Aequationes mathematicae **4**, No. 3, 1970, pp. 374-383.

[Buch76a]
B. Buchberger: "A Theoretical Basis for the Reduction of Polynomials to Canonical Forms", ACM SIGSAM Bull. **10**, No. 3, 1976, pp. 19-29.

[Buch76b]
B. Buchberger: "Some Properties of Gröbner-Bases for Polynomial Ideals", ACM SIGSAM Bull. **10**, No. 4, 1976, pp. 19-24.

[Buch79a]
B. Buchberger: "A criterion for detecting unnecessary reductions in the construction of Gröbner bases", Proc. EUROSAM '79, Marseille, June, 1979, (W. Ng, ed.), Springer *Lecture Notes in Computer Science* **72**, pp. 3-21.

[Buch79b]
B. Buchberger, F. Winkler: "Miscellaneous results on the construction of Gröbner bases for polynomial ideals", Tech. Rep. 137, Univ. of Linz, Math. Inst., 1979.

[Buch83a]
B. Buchberger: "A note on the complexity of constructing Gröbner bases", Proc. EUROCAL '83, London, March, 1983, (H. van Hulzen, ed.), Springer *Lecture Notes in Computer Science* **162**, pp. 137-145.

[Buch83b]
B. Buchberger, R. Loos: "Algebraic Simplification", in *Computer Algebra - Symbolic and Algebraic Computation*, (B. Buchberger, G. Collins, R. Loos eds.), 2nd edition, Springer Wein - New York, 1983, pp. 11-43.

[Buch85]
B. Buchberger: "Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory", in *Progress, directions and open problems in multidimensional systems theory*, (N.K. Bose, ed.), D. Reidel Publishing Co., 1985, pp. 184-232.

[Carm87]
J. Carminatti, R. G. McLenaghan: "An explicit determination of the space-times on which the conformally invariant scalar wave equation satisfies Huygens' principle II: Petrov Type D space-times", to appear in Ann. Inst. Henri Poicaré, Phys. Théor.

[Char83]
B.W. Char, K.O. Geddes, W.M. Gentleman, G.H. Gonnet: "The design of Maple: A compact, portable, and powerful computer algebra system", Proc. EUROCAL '83, London, March, 1983, (H. van Hulzen, ed.), Springer *Lecture Notes in Computer Science* **162**, pp. 101-115.

[Char84]
B.W. Char, K.O. Geddes, G.H. Gonnet: "GCDHEU: Heuristic Polynomial GCD Algorithm Based on Integer GCD Computation", Proc. EUROSAM 84, Cambridge, July, 1984, (J.P. Fitch, ed.), Springer *Lecture Notes in Computer Science* **174**, pp. 285-296.

[Char86a]
B.W. Char: Private communication, 1986.

[Char86b]
B.W. Char, G.J. Fee, K.O. Geddes, G.H. Gonnet, M.B. Monagan: "A Tutorial Introduction To Maple", J. Symbolic Comp. **2**, No. 2, 1986.

[Chis83]
A. L. Chistov, D. Yu. Grigoryev: "Subexponential-time solving systems of algebraic equations I, II", E-10-83, LOMI Preprints, Leningrad, 1983.

[Coll67]
G. E. Collins: "Subresultants and reduced polynomial remainder sequences", J. ACM **14**, No. 1, 1967, pp. 128-142.

[Coll69]
G. E. Collins: "Comment on a Paper by Ku and Adler", (letter to the editor), Comm. ACM **12**, No. 6, 1969, p. 302.

[Coll71]
G. E. Collins: "The Calculation of Multivariate Polynomial Resultants", J. ACM **18**, No. 4, 1971, pp. 515-532.

[Coll75]
G. E. Collins: "Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition", Proc. Second GI Conference on Automata Theory and Formal Languages, 1975, Springer *Lecture Notes in Computer Science* **33**, pp. 134-183.

[Coll83]
G. E. Collins: "Quantifier Elimination for Real Closed Fields: A Guide to the Literature", in *Computer Algebra - Symbolic and Algebraic Computation*, (B. Buchberger, R. Loos and G.E. Collins, eds.), 2nd edition, Springer Wien - New York, 1983, pp. 79-81.

[Czap86a]
S. R. Czapor, K. O. Geddes: "On Implementing Buchberger's Algorithm for Gröbner Bases", Proc. SYMSAC '86, Waterloo, July, 1986, (B. W. Char, ed.), pp. 233-238.

[Czap86b]
S.R. Czapor: "Solving Algebraic Equations: Combining Buchberger's Algorithm with Multivariate Factorization", to appear in J. Symbolic Comp..

[Czap87]
S. R. Czapor: "Solving Algebraic Equations via Buchberger's Algorithm", Proc. EUROCAL '87, Leipzig, June, 1987, to appear in Springer *Lecture Notes in Computer Science.*

[Dave85]
J. H. Davenport: "Computer Algebra for Cylindrical Algebraic Decomposition", KTH Report TRITA-NADA-8511, Royal Inst. of Tech., Stockholm, 1985.

[Fee87]
G. Fee: Private communication, 1987.

[Geba86]
R. Gebauer, M. Möller: Buchberger's Algorithm and Staggered Linear Bases", Proc. SYMSAC '86, Waterloo, July, 1986, (B. W. Char, ed.), pp. 218-221.

[Gedd83]
K. O. Geddes: Course Notes, CS 687, Winter, 1983, University of Waterloo.

[Gian85]
P. Gianni, B. Trager: "GCD's and Factoring Multivariate Polynomials Using Gröbner Bases", Proc. EUROCAL '85, Vol. 2, Linz, April, 1985, (B. F. Caviness, ed.), Springer *Lecture Notes in Computer Science* **204**, pp. 409-410.

[Gian87]
P. Gianni: "Properties of Gröbner bases under specialization", Proc. EUROCAL '87, Leipzig, June, 1987, to appear in Springer *Lecture Notes in Computer Science.*

[Gius85]
M. Giusti: "A Note on the Complexity of Constructing Standard Bases", Proc. EUROCAL '85, Vol. 2, Linz, April, 1985, (B. F. Caviness, ed.), Springer *Lecture Notes in Computer Science* **204**, pp. 411-412.

[Gonn86]
G. H. Gonnet, M. B. Monagan: "Solving Systems of Algebraic Equations, or the Interface between Software and Mathematics", Computers and Mathematics Conference, Stanford, Calif., July, 1986.

[Gonn87]
G. H. Gonnet: Private communication, 1987.

[Hear79]
A.C. Hearn: "Non-modular Computation of Polynomial GCD's using Trial Divisions", Proc. EUROSAM '79, Marseille, June, 1979, (W. Ng, ed.), Springer *Lecture Notes in Computer Science* **72**, pp. 227-239.

[Herm26]
G. Hermann: "The Question of Finitely Many Steps in Polynomial Ideal Theory (German)", Math. Ann. **95**, 1926, pp. 736-788.

[Hiro64]
H. Hironaka: "Resolution of singularities of an algebraic variety over a field of characteristic zero I, II", Ann. Math. **79**, 1964, pp. 109-326.

[Kalk87]
M. Kalkbrener: "Solving systems of algebraic equations by using Gröbner bases", Proc. EUROCAL '87, Leipzig, June, 1987, to appear in Springer *Lecture Notes in Computer Science.*

[Kand84]
A. Kandri-Rody, D. Kapur: "Algorithms for Computing Gröbner Bases of Polynomial Ideals over Various Euclidean Rings", Proc. EUROSAM 84, Cambridge, July, 1984, (J. Fitch, ed.), Springer *Lecture Notes in Computer Science* **174**, pp. 195-205.

[Kapu86]
D. Kapur: "Geometry Theorem Proving Using Hilbert's Nullstellensatz", Proc. SYMSAC '86, Waterloo, July, 1986, (B. W. Char, ed.), pp. 202-208.

[Knut67]
D. E. Knuth, P. B. Bendix: "Simple word problems in universal algebras", Proc. of the Conf. on Computational Problems in Abstract Algebra (OXFORD '67), (J. Leech, ed.), Pergamon Press, Oxford, 1970, pp. 263-298.

[Knut69]
D. E. Knuth: *The Art of Computer Programming, Vols. 1, 2*, Addison-Wesley, Reading, Mass., 1969.

[Koll78]
C. Kollreider, B. Buchberger: "An Improved Algorithmic Construction of Gröbner Bases For Polynomial Ideals", ACM SIGSAM Bull. **12**, No. 2, 1978, pp. 27-36.

[KuAd69]
S. Y. Ku, R. J. Adler: "Computing Polynomial Resultants: Bezout's Determinant *vs.* Collins' Reduced P.R.S. Algorithm", Comm. ACM **12**, No. 1, 1969, pp. 23-30.

[Laza79]
D. Lazard: "Systems of Algebraic Equations", Proc. EUROSAM '79, Marseille, June, 1979, (W. Ng, ed.), Springer *Lecture Notes in Computer Science* **72**, pp. 88-94.

[Laza81]
D. Lazard: "Résolution des systèmes d'équations algébrique", Theor. Comp. Sciences **15**, 1981, pp. 77-110.

[Laza83]
D. Lazard: "Gröbner Bases, Gaussian Elimination, and Resolution of Systems of Algebraic Equations", Proc. EUROCAL '83, London, March, 1983, (H. van Hulzen, ed.), Springer *Lecture Notes in Computer Science* **162**, pp. 146-156.

[Laza85]
D. Lazard: "Ideal Bases and Primary Decomposition: Case of Two Variables", J. Symbolic Comp. **1**, No. 3, 1985, pp. 261-270.

[LeCh83]
P. Le Chenadec: "Canonical forms in finitely presented algebras (French), Ph.D. Thesis, Univ. of Paris-Sud, Centre d'Orsay, 1983.

[Loos83a]
R. Loos: "Generalized Polynomial Remainder Sequences", in *Computer Algebra - Symbolic and Algebraic Computation*, (B. Buchberger, R. Loos and G.E. Collins, eds.), 2nd edition, Springer Wien - New York, 1983, pp. 115-137.

[Loos83b]
R. Loos: "Computing in Algebraic Extensions", in *Computer Algebra - Symbolic and Algebraic Computation*, (B. Buchberger, R. Loos and G.E. Collins, eds.), 2nd edition, Springer Wien - New York, 1983, pp. 173-187.

[Mart71]
W. A. Martin, R. J. Fateman: "The MACSYMA-System", Proc. SYMSAM '71, Los Angeles, March, 1971, (S. R. Petrick, ed.), pp. 59-75.

[Mayr81]
E. Mayr, A. Meyer: "The complexity of the word problems for commutative semigroups and polynomial ideals", Report LCS/TM-199, M.I.T. Lab. of Computer Science, 1981.

[Moll84]
H. M. Möller, F. Mora: "Upper and Lower Bounds for the Degree of Gröbner Bases", Proc. EUROSAM 84, Cambridge, July, 1984, (J. Fitch, ed.), Springer *Lecture Notes in Computer Science* **174**, pp. 172-183.

[Mona86]
M.B. Monagan: Private communication.

[Mona87]
M. B. Monagan: "Heuristic Irreducibility Test", submitted to J. Symbolic Comp., 1987.

[Mose66]
J. Moses: "Solution of Systems of Polynomial Equations By Elimination", Comm. ACM **9**, No. 8, 1966, pp. 634-637.

[Pohs81]
M.E. Pohst, D.Y.Y. Yun: "On Solving Systems of Algebraic Equations via Ideal Bases and Elimination Theory", Proc. SYMSAC '81, (P.S. Wang, ed.), Utah, August, 1981, pp. 206-211.

[Rich68]
D. Richardson: "Some Unsolvable Problems Involving Elementary Functions of a Real Variable", J. Symbolic Logic **33**, 1968, pp. 511-520.

[Rime84]
K. Rimey: "A System of Polynomial Equations and a Solution by an Unusual Method", ACM SIGSAM Bull. **18**, No. 1, 1984, pp. 30-32.

[Robb85]
L. Robbiano: "Term Orderings on the Polynomial Ring", Proc. EUROCAL '85, Vol. 2, Linz, April, 1985, (B. F. Caviness, ed.), Springer *Lecture Notes in Computer Science* **204**, pp. 513-517.

[Roth84]
M. Rothstein: "On Pseudo-Resultants", in Proc. EUROSAM 84, Cambridge, July, 1984, (J. Fitch, ed.), Springer *Lecture Notes in Computer Science* **174**, pp. 387-394.

[Savi87]
P. Savitch: "The Chistov and Grigoryev Algorithm for Solving Systems of Polynomial Equations", M. Math. Thesis, Univ. of Waterloo, Dept. of Comp. Sci., 1987.

[Tars51]
A. Tarski: *A Decision Method for Elementary Algebra and Geometry*, Univ. of California Press, 1951.

[Trin78]
W. Trinks: "On B. Buchberger's Method for Solving Systems of Algebraic Equations (German)", J. Number Theory **10**, No. 4, 1978, pp. 475-488.

[Trin84]
W. Trinks: "On Improving Approximate Results of Buchberger's Algorithm by Newton's Method", ACM SIGSAM Bull. **18**, No. 3, 1984, pp. 7-11.

[Waer70]
B. L. van der Waerden: *Modern Algebra, Vol. 1*, Frederick Ungar Publishing Co., New York, 1970.

[Waer53]
B. L. van der Waerden: *Modern Algebra, Vol. 2*, Frederick Ungar Publishing Co., New York, 1953.

[Wang78]
P. Wang: "An Improved Multivariate Polynomial Factoring Algorithm", Math. Comp. **32**, 1978, pp. 1215-1231.

[Watt85]
S. M. Watt: "Bounded Parallelism in Computer Algebra", Ph.D. Thesis, Univ. of Waterloo, Dept. of Comp. Sci., 1985.

[Will62]
L. H. Williams: "Algebra of Polynomials in Several Variables for a Digital Computer", J. ACM **9**, 1962, pp. 29-40.

[Wink84]
F. Winkler: "On the Complexity of the Gröbner-Bases Algorithm over $K[x,y,z]$", Proc. EUROSAM 84, Cambridge, July, 1984, (J. Fitch, ed.), Springer *Lecture Notes in Computer Science* **174**, pp. 184-194.

[Wink87]
F. Winkler: "p-adic methods for the computation of Gröbner bases", Proc. EUROCAL '87, Leipzig, June, 1987, to appear in Springer *Lecture Notes in Computer Science.*

[Yun73]
D. Y. Y. Yun: "On Algorithms For Solving Systems Of Polynomial Equations", ACM SIGSAM Bull. **27**, 1973, pp. 19-25.