ABSTRACT

Title of thesis: AN ANALYSIS OF IMPROVEMENTS TO BUCHBERGER'S ALGORITHM FOR GRÖBNER BASIS COMPUTATION

Clinton E. McKay, Master of Arts, 2004

Thesis directed by: Professor William W. Adams
Department of Mathematics

Improvements to Buchberger's Algorithm generally seek either to define a criterion for the removal of unnecessary S-pairs or to describe a strategy for improving the choices which one must make in the course of the algorithm. This paper surveys significant improvements to Buchberger's original algorithm for Gröbner basis computation including the Gebauer-Möller Criteria, the "Sugar" strategy, and Jean-Charles Faugère's F4 algorithm. Since Faugère's F4 is generally accepted as being a particularly efficient approach to Gröbner basis computation, we test several variants of the F4 algorithm on a variety of benchmark ideals in an effort to judge the efficiency of the Gröbner basis computation process, while also being mindful of the memory constraint issues occurring in computer algebra.

AN ANALYSIS OF IMPROVEMENTS TO BUCHBERGER'S ALGORITHM FOR GRÖBNER BASIS COMPUTATION

by

Clinton E. McKay

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Arts
2004

Advisory Commmittee:

Professor William W. Adams, Chair/Advisor
Professor Lawrence C. Washington
Assistant Professor Niranjan Ramachandran

ACKNOWLEDGMENTS

I owe my gratitude to all the people who have made this thesis possible. I would especially like to thank my advisor, Professor William Adams, for the insights and guidance he provided throughout this effort. He has always been available to provide help and advice. It has been a pleasure to have had the opportunity to learn from a professor who is extremely knowledgable and also incredibly patient with those who are less knowledgable. Special thanks also to Alyson Reeves for her insights and for providing the implementations of the F4 algorithm, which I used extensively in my research. My most sincere thanks also to Mom, Dad, Jessica Fitzgerald, Boo Barkee, M. E. Yao, Jim Fennell, Charlie Toll, Jacquie Holmgren, Jim Schatz, Jack Clark, Bob Boner, and Harry Rosenzweig.

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1

Introduction

Gröbner bases provide a means for studying polynomial ideals in many variables, and the theory of Gröbner bases may be applied to problems in a variety of disciplines including mathematics, computer science, and engineering. Bruno Buchberger first developed an algorithm for computing Gröbner bases in 1965, and since then, there have been numerous efforts to improve the efficiency of the algorithm. These efforts have in many cases proven successful, and improved versions of Buchberger's Algorithm are utilized in nearly all computer algebra systems.

Before beginning our analysis of these improvements to Buchberger's Algorithm, we first introduce the basic terminology and results necessary to study Gröbner bases. Let $K[x_1, \ldots, x_n]$ be the ring of polynomials in n variables with coefficients from the field K. We denote the set of monomials, also referred to as *power products,* by

$$\mathrm{T}^n = \{x_1^{\beta_1} \ldots x_n^{\beta_n} \mid \beta_i \in \mathrm{N},\, i = 1, \ldots, n\}.$$

Let $x^\alpha = x_1^{\alpha_1} \ldots x_n^{\alpha_n}$, for non-negative integers $\alpha_1, \ldots, \alpha_n$, and let $deg(\alpha) = \sum_{i=1}^n \alpha_i$ be the total degree of the monomial x^α. Given a set of nonzero polynomials $F = \{f_1, \ldots, f_s\}$ in $K[x_1, \ldots, x_n]$, we denote the ideal generated by F as $\langle F \rangle$.

Definition 1.1 *A term order $>_T$ on the monomials of $K[x_1, \ldots, x_n]$ is a total order such that for any exponents α,β,$\gamma \in N^n$, $x^\alpha \geq 1$ and $x^\alpha >_T x^\beta$ implies $x^\alpha x^\gamma >_T x^\beta x^\gamma$.*

We now provide three examples of commonly utilized term orders:

Example 1.2 *Lexicographic (lex): $x^\alpha >_{lex} x^\beta$ if the leftmost nonzero entry of $\alpha - \beta$ is positive.*

Example 1.3 *Degree Lexicographic (degl): $x^\alpha >_{degl} x^\beta$ if $deg(\alpha) > deg(\beta)$, or $deg(\alpha) = deg(\beta)$ and $x^\alpha >_{lex} x^\beta$.*

Example 1.4 *Degree Reverse Lexicographic (rev): $x^\alpha >_{rev} x^\beta$ if $deg(\alpha) > deg(\beta)$, or $deg(\alpha) = deg(\beta)$ and the rightmost nonzero entry of $\alpha - \beta$ is negative.*

Definition 1.5 *Fix a term order $>$, and let f be a polynomial in $K[x_1, \ldots, x_n]$ with $f \neq 0$. We may write*

$$f = a_1 x^{\alpha_1} + a_2 x^{\alpha_2} + \cdots + a_r x^{\alpha_r},$$

where $0 \neq a_i \in K$, $x^{\alpha_i} \in \mathrm{T}^n$, and $x^{\alpha_1} > x^{\alpha_2} > \cdots > x^{\alpha_r}$. Then we define $\mathrm{LT}(f) = a_1 x^{\alpha_1}$, the leading term of f, and $\mathrm{LM}(f) = x^{\alpha_1}$, the leading monomial of f.

Definition 1.6 *Given a set $F = \{f_1, \ldots, f_s\}$ of polynomials in $K[x_1, \ldots, x_n]$. We define the set $\mathrm{LT}(F) = \{\mathrm{LT}(f) \mid f \in F\}$, and similarly we define the set $\mathrm{LM}(F) = \{\mathrm{LM}(f) \mid f \in F\}$.*

Definition 1.7 *Given a subset S of $K[x_1, \ldots, x_n]$, we define the leading term ideal of S to be the ideal*

$$\mathrm{LTI}(S) = \langle \mathrm{LT}(s) \mid s \in S \rangle.$$

Definition 1.8 *A Gröbner basis G of a polynomial ideal I, with respect to $>$, is a finite set of nonzero polynomials $\{g_1, \ldots, g_t\} \subseteq I$ such that $\mathrm{LTI}(G) = \mathrm{LTI}(I)$. G is a reduced Gröbner basis for I if for all $g \in G$, g is monic and no monomial of g lies in $\mathrm{LTI}(G - \{g\})$.*

Proposition 1.9 *Let $I \neq \{0\}$ be an ideal in $K[x_1, \ldots, x_n]$. Then, for a given term order, I has a unique reduced Gröbner basis.*

The purpose of requiring the elements of G to be monic in *Definition 1.8* is to guarantee the uniqueness of the reduced Gröbner basis. The proof of this proposition is given in [1] and [6].

Definition 1.10 *Given $f, g, h \in K[x_1, \ldots, x_n]$, with $g \neq 0$, we say that f reduces to h modulo g in one step, written*

$$f \xrightarrow{g} h,$$

if and only if $\mathrm{LM}(g)$ divides a nonzero term X that appears in f and

$$h = f - \frac{X}{\mathrm{LT}(g)} g.$$

Definition 1.11 *Let $f, h,$ and $f_1, \ldots, f_s$ be polynomials in $K[x_1, \ldots, x_n]$, with $f_i \neq 0$ $(1 \leq i \leq s)$, and let $F = \{f_1, \ldots, f_s\}$. We say that f reduces to h modulo F, denoted*

$$f \xrightarrow{F}_{+} h,$$

if and only if there exist a sequence of indices $i_1, i_2, \ldots, i_t \in \{1, \ldots, s\}$ *and a sequence of polynomials* $h_1, \ldots, h_{t-1} \in K[x_1, \ldots, x_n]$ *such that*

$$f \xrightarrow{f_{i_1}} h_1 \xrightarrow{f_{i_2}} h_2 \xrightarrow{f_{i_3}} \ldots \xrightarrow{f_{i_{t-1}}} h_{t-1} \xrightarrow{f_{i_t}} h.$$

Definition 1.12 *A polynomial* r *is called reduced with respect to a set of non-zero polynomials* $F = \{f_1, \ldots, f_s\}$ *if* $r = 0$ *or no monomial that appears in* r *is divisible by any one of the* $\mathrm{LM}(f_i), i = 1, \ldots, s$. *In other words* r *cannot be reduced modulo* F.

Definition 1.13 *If* $f \xrightarrow{F}_+ r$ *and* r *is reduced with respect to* F, *then we call* r *a remainder for* f *with respect to* F.

The reduction process enables us to define a division algorithm analagous to the well-known Division Algorithm in one variable. Given $f, f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$ with $f_i \neq 0 (1 \leq i \leq s)$, this algorithm returns quotients $u_1, \ldots, u_s \in K[x_1, \ldots, x_n]$, and a remainder $r \in K[x_1, \ldots, x_n]$, such that $f = u_1 f_1 + \cdots + u_s f_s + r$.

Theorem 1.14 *Let* I *be a non-zero ideal of* $K[x_1, \ldots, x_n]$. *The following statements are equivalent for a set of non-zero polynomials* $G = \{g_1, \ldots, g_t\} \subseteq I$.

1. G *is a Gröbner basis for* I.
2. $f \in I$ *iff* f *reduces to 0 modulo* G.
3. $f \in I$ *iff* $f = \sum_{i=1}^{t} h_i g_i$ *with* $\mathrm{LM}(f) = max_{1 \leq i \leq t}(\mathrm{LM}(h_i)\mathrm{LM}(g_i))$.

The preceeding theorem and its proof may be found in Adams and Loustaunau [1].

Definition 1.15 *Let* $0 \neq f, g \in K[x_1, \ldots, x_n]$. *Let* $L = \mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g))$. *The polynomial* $S(f, g) = \frac{L}{\mathrm{LT}(f)} f - \frac{L}{\mathrm{LT}(g)} g$ *is called the S-polynomial of* f *and* g.

Theorem 1.16 (Buchberger) *Let* $G = \{g_1, \ldots, g_t\}$ *be a set of non-zero polynomials in* $K[x_1, \ldots, x_n]$. *Then* G *is a Gröbner basis for the ideal* $I = \langle g_1, \ldots, g_t \rangle$ *if and only if for all* $i \neq j$,

$$S(g_i, g_j) \xrightarrow{G}_+ 0.$$

Again, this theorem along with the proof thereof may be found in [1].

Chapter 2

Buchberger's Algorithm

As a result of Buchberger's Theorem, there is a definitive, though not necessarily desirable, process which one may go through to compute a Gröbner basis for an ideal I. In particular, the process begins with reducing an S-polynomial formed from the generators of the ideal. Then, if non-zero, the remainder is appended to the list of polynomials in our generating set. Additional S-polynomials are formed, and the process then repeats until there are enough polynomials in the generating set to reduce all S-polynomials to zero. Using the Noetherian property of $K[x_1, \ldots, x_n]$, we know that indeed this process will always terminate.

The basic Buchberger Algorithm is as follows:

INPUT: $F = \{f_1, \ldots, f_s\} \subseteq K[x_1, \ldots, x_n]$ with $f_i \neq 0$ $(1 \leq i \leq s)$
OUTPUT: $G = \{g_1, \ldots, g_t\}$, a Gröbner basis for $\langle f_1, \ldots, f_s \rangle$
INITIALIZATION: $G := F, SP := \{\{f_i, f_j\} \mid f_i \neq f_j \in G\}$
WHILE $SP \neq \emptyset$ DO
Choose any $(f, g) \in SP$
$SP := SP - \{(f, g)\}$
$S(f, g) \xrightarrow{G}_+ h$, where h is reduced with respect to G
IF $h \neq 0$ THEN
$SP := SP \cup \{(u, h) \mid$ for all $u \in G\}$
$G := G \cup \{h\}$

Table 2.1: Buchberger's Algorithm

This algorithm for computing the Gröbner basis of an ideal is often extremely computationally intense, since the number of S-polynomials may grow to be quite large. Also, it is the case that an unfortunate choice as to the order in which the S-polynomials are considered could result in drastically more S-polynomial computations and reductions than would have been the case had a "better" choice been made. Although the reduction of an S-polynomial may be carried out rather quickly, the process of reducing *all* S-polynomials will often be quite time consuming when there are large quantities of S-polynomials needing to be reduced. Thus, Buchberger and other leading

mathematicians sought to develop methods for predicting when S-polynomials would reduce to zero without actually computing them.

Chapter 3

The Gebauer-Möller Criteria

Though nearly twenty years old, Rüdiger Gebauer and H. Michael Möller's paper *On an Installation of Buchberger's Algorithm* presents an improved version of Buchberger's Algorithm which remains an important benchmark against which the most modern algorithms for Gröbner basis computation are compared. Gebauer and Möller build on Buchberger's prior achievements and describe practical criteria for detecting superfluous S-polynomial reductions. Before introducing this criteria, we first define the concept of a syzygy and present another equivalent condition for being a Gröbner basis of an ideal:

Definition 3.1 *Given a set of nonzero polynomials* $F = \{f_1, \ldots, f_r\}$ *in* $K[x_1, \ldots, x_n]$*, we define a syzygy of the tuple* $(\mathrm{LM}(f_1), \ldots, \mathrm{LM}(f_r))$ *to be any tuple* $(h_1, \ldots, h_r)$ *of* $K[x_1, \ldots, x_n]^r$ *such that* $h_1\mathrm{LM}(f_1) + \cdots + h_r\mathrm{LM}(f_r) = 0$*. The set of all syzygies of* $(\mathrm{LM}(f_1), \ldots, \mathrm{LM}(f_r))$ *forms a* $K[x_1, \ldots, x_n]$*-module which we call the first syzygy module of* $(\mathrm{LM}(f_1), \ldots, \mathrm{LM}(f_r))$ *and denote by* $S^{(1)}$*.*

Theorem 3.2 *The first syzygy module has a generating set,* $L^{(1)}$*, which is as follows:*

$$L^{(1)} := \{S_{ij} \mid 1 \leq i < j \leq r\} \text{ with}$$

$$S_{ij} := \frac{\mathrm{LCM}(\mathrm{LM}(f_i), \mathrm{LM}(f_j))}{\mathrm{LM}(f_i)}\mathbf{e_i} - \frac{\mathrm{LCM}(\mathrm{LM}(f_i), \mathrm{LM}(f_j))}{\mathrm{LM}(f_j)}\mathbf{e_j}$$

where e_k *is the* k^{th} *canonical unit vector of* $K[x_1, \ldots, x_n]^r$*.*

The preceding theorem was originally proved by D. K. Taylor in [11]. A proof may also be found in [1].

Theorem 3.3 *Let* $G = \{g_1, \ldots, g_r\} \subset K[x_1, \ldots, x_n] \backslash \{0\}$ *and* $I = \langle G \rangle$*. Then the following conditions are equivalent:*

1. *G is a Gröbner basis of I.*

2. Let L be a generating set of the module of syzygies

$$S^{(1)} := \{(h_1, \ldots, h_r) \in K[x_1, \ldots, x_n]^r \quad \Big| \quad \sum_{i=1}^{r} h_i \mathrm{LM}(g_i) = 0\}.$$

Then for each $(f_1, \ldots, f_r) \in L$,

$$\sum_{i=1}^{r} f_i g_i \xrightarrow{G}_{+} 0.$$

For the proof of this theorem, we refer the reader to H.M. Möller's paper, [10].

Definition 3.4 *An element g_i of a Gröbner basis G is redundant if $G' := G \backslash \{g_i\}$ is also a Gröbner basis and $\langle G \rangle = \langle G' \rangle$.*

Definition 3.5 *Using the result of Theorem 3.3, we will similarly define an element S_{ik} of $L^{(1)}$ to be redundant if $L^{(1)} \backslash \{S_{ik}\}$ still generates $S^{(1)}$.*

Definition 3.6 *A minimal Gröbner basis for a polynomial ideal I is a Gröbner basis G for I such that no element of G is redundant and all elements of G are monic.*

The Gebauer-Möller Criteria, which we present below, applied to a polynomial ideal with a particular term order, enable one to avoid carrying out the reduction of many S-polynomials, which if reduced, would have reduced to zero. From Theorem 3.3 we may seek to eliminate redundant elements in a Gröbner basis, G, by eliminating redundant elements in a basis of the first syzygy module of the the leading monomials of G. This practice of avoiding unnecessary S-polynomial reductions has been experimentally shown to improve the efficiency of a Gröbner basis computation. It should be pointed out, however, that use of the Gebauer-Möller Criteria do not necessarily result in obtaining a minimal Gröbner basis. Nevertheless, these criteria are an extremely effective means of producing a nearly minimal, if not minimal, set of generators for a polynomial ideal.

Definition 3.7 *To find redundant elements in a basis for $S^{(1)}$, we define the module of syzygies for $L^{(1)}$, which we denote $S^{(2)}$. Given a term order $<_T$ on $K[x_1, \ldots, x_n]$ we define an order on the*

$r(r-1)/2$ *syzygies* S_{ij} *by* $<_1$:

$$S_{ij} <_1 S_{kl} \iff \mathrm{LCM}(\mathrm{LM}(f_i), \mathrm{LM}(f_j)) <_T \mathrm{LCM}(\mathrm{LM}(f_k), \mathrm{LM}(f_l)) \text{ or}$$

$$\mathrm{LCM}(\mathrm{LM}(f_i), \mathrm{LM}(f_j)) = \mathrm{LCM}(\mathrm{LM}(f_k), \mathrm{LM}(f_l)), \text{ and } j < l \text{ or}$$

$$\mathrm{LCM}(\mathrm{LM}(f_i), \mathrm{LM}(f_j)) = \mathrm{LCM}(\mathrm{LM}(f_k), \mathrm{LM}(f_l)), \ j = l, \text{ and } i < k.$$

Using this order, we denote the ij^{th} *canonical unit vector in* $K[x_1, \ldots, x_n]^{r(r-1)/2}$ *by* e_{ij}. *Let*

$$S^{(2)} := \left\{ \sum_{\substack{ij=1 \\ i<j}}^{r} h_{ij} e_{ij} \in K[x_1, \ldots, x_n]^{r(r-1)/2} \quad \middle| \quad \sum_{\substack{ij=1 \\ i<j}}^{r} h_{ij} S_{ij} = 0 \right\}.$$

Theorem 3.8 *The module of syzygies* $S^{(2)}$ *has a generating set,* $L^{(2)}$, *defined as follows:*

$$L^{(2)} := \{S_{ijk} \mid 1 \leq i < j < k \leq r\} \text{ with}$$

$$S_{ijk} = \frac{\mathrm{LCM}(\mathrm{LM}(f_i), \mathrm{LM}(f_j), \mathrm{LM}(f_k))}{\mathrm{LCM}(\mathrm{LM}(f_i), \mathrm{LM}(f_j))} e_{ij} - \frac{\mathrm{LCM}(\mathrm{LM}(f_i), \mathrm{LM}(f_j), \mathrm{LM}(f_k))}{\mathrm{LCM}(\mathrm{LM}(f_i), \mathrm{LM}(f_k))} e_{ik} + \frac{\mathrm{LCM}(\mathrm{LM}(f_i), \mathrm{LM}(f_j), \mathrm{LM}(f_k))}{\mathrm{LCM}(\mathrm{LM}(f_j), \mathrm{LM}(f_k))} e_{jk}.$$

A proof of the preceding theorem may be found in [11].

Definition 3.9 *The maximal syzygy involved in* S_{ijk} *is denoted* $\mathrm{MS}(i, j, k)$ *and is defined by*

$$\mathrm{MS}(i, j, k) := max_{<_1}\{S_{ij}, S_{ik}, S_{jk}\}.$$

Let $F = \{f_1, \ldots, f_r\} \in K[x_1, \ldots, x_n]$, and let $S^{(1)}$ be the first syzygy module of $(\mathrm{LM}(f_1), \ldots, \mathrm{LM}(f_r))$. From Theorem 3.2, we know that $L^{(1)} := \{S_{ij} \mid 1 \leq i < j \leq r\}$ is a generating set for $S^{(1)}$, however, some of the S_{ij} in $L^{(1)}$ may be redundant. Through eliminating redundant syzygies in $L^{(1)}$, we are eliminating S-polynomials from our set of S-polynomials to reduce. Indeed, suppose S_{ij} is redundant, where

$$S_{ij} = \frac{\mathrm{LCM}(\mathrm{LM}(f_i), \mathrm{LM}(f_j))}{\mathrm{LM}(f_i)} \mathbf{e_i} - \frac{\mathrm{LCM}(LM(f_i), \mathrm{LM}(f_j))}{\mathrm{LM}(f_j)} \mathbf{e_j}.$$

Recall, e_k is the k^{th} canonical unit vector of $K[x_1, \ldots, x_n]^r$. Using the natural one-one correspondance between unit vectors in $K[x_1, \ldots, x_n]^r$ and polynomials in F, it follows that the S-polynomial

$S(f_i, f_j) = \frac{\text{LCM}\{\text{LM}(f_i),\text{LM}(f_j)\}}{\text{LT}(f_i)} f_i - \frac{\text{LCM}\{\text{LM}(f_i),\text{LM}(f_j)\}}{\text{LT}(f_j)} f_j$ will reduce to zero, and therefore does not need to be reduced. The purpose of the Gebauer-Möller Criteria is to eliminate redundant elements in a generating set for $S^{(1)}$.

Let $\{f_1, \ldots, f_r\}$ be a set of polynomials in $K[x_1, \ldots, x_n]$. The Gebauer-Möller Criteria are as follows:

1. Criterion M holds for a pair (f_i, f_k) if $\exists j < k$, such that

$$\text{LCM}\{\text{LM}(f_j), \text{LM}(f_k)\} \text{ properly divides } \text{LCM}\{\text{LM}(f_i), \text{LM}(f_k)\}.$$

2. Criterion F holds for a pair (f_i, f_k) if $\exists j < i$, such that

$$\text{LCM}\{\text{LM}(f_j), \text{LM}(f_k)\} = \text{LCM}\{\text{LM}(f_i), \text{LM}(f_k)\}.$$

3. Criterion B_k holds for a pair (f_i, f_j) if $\exists j < k$ and

$$\text{LM}(f_k) \mid \text{LCM}\{\text{LM}(f_i), \text{LM}(f_j)\},$$

$$\text{LCM}\{\text{LM}(f_i), \text{LM}(f_k)\} \neq \text{LCM}\{\text{LM}(f_i), \text{LM}(f_j)\}, \text{ and}$$

$$\text{LCM}\{\text{LM}(f_i), \text{LM}(f_j)\} \neq \text{LCM}\{\text{LM}(f_j, f_k).$$

In any cases in which criterion M, F, or B_k hold, the associated S-polynomial will reduce to zero, and therefore does not need to be computed. This result is explained by the following proposition:

Proposition 3.10 *The module of syzygies $S^{(1)}$ is generated by*

$$L^* := \{S_{ij} \mid 1 \leq i < j \leq r, \; M(i,j) \textit{ fails, } F(i,j) \textit{ fails, and } B_k(i,j) \textit{ fails } \forall k > j\}.$$

Proof. We first observe that if $\text{LCM}\{\text{LM}(f_i), \text{LM}(f_j), \text{LM}(f_k)\} =$

$$max\{\text{LCM}\{\text{LM}(f_i), \text{LM}(f_j)\}, \text{LCM}\{\text{LM}(f_i), \text{LM}(f_k)\}, \text{LCM}\{\text{LM}(f_j), \text{LM}(f_k)\}\},$$

then one of the three components of S_{ijk} is a constant, and $\text{MS}(i,j,k)$ can be expressed in terms of lesser syzygies with respect to $<_1$. Suppose that S_{ik} may be written in terms of the syzygies S_{ij} and S_{jk}, then we may remove S_{ik} from $L^{(1)}$ since $L^{(1)} \backslash \{S_{ik}\}$ still generates $S^{(1)}$.

If M holds for (f_i, f_k), then $\exists j < k$, such that $\text{LCM}\{\text{LM}(f_j), \text{LM}(f_k)\}$ properly divides $\text{LCM}\{\text{LM}(f_i), \text{LM}(f_k)\}$ in fact $\text{LCM}\{\text{LM}(f_i), \text{LM}(f_k)\} =$
$\text{LCM}\{\text{LM}(f_i), \text{LM}(f_j), \text{LM}(f_k)\}$ and $j \neq i$. So either S_{ijk} (if $i < j$) or S_{jik} (if $j < i$) has S_{ik} for its maximal syzygy:

$$S_{jk} <_1 S_{ik} \text{ because } \text{LCM}\{\text{LM}(f_j), \text{LM}(f_k)\} <_T \text{LCM}\{\text{LM}(f_i), \text{LM}(f_k)\}$$

$$S_{ij} \text{ or } S_{ji} <_1 S_{ik} \text{ because } j < k, i < k \text{ and}$$

$$\text{LCM}\{\text{LM}(f_i), \text{LM}(f_j)\} \leq_T \text{LCM}\{\text{LM}(f_i), \text{LM}(f_k)\}.$$

So S_{ik} is redundant and therefore not required to be in a generating set for $S^{(1)}$. If F holds for (f_i, f_k), then $\text{LCM}\{\text{LM}(f_j), \text{LM}(f_k)\} = \text{LCM}\{\text{LM}(f_i), \text{LM}(f_k)\}$ for a $j < i$. So considering S_{jik}, we have

$$\text{LCM}\{\text{LM}(f_j), \text{LM}(f_i), \text{LM}(f_k)\} = \text{LCM}\{\text{LM}(f_i), \text{LM}(f_k)\},$$

and by a similar argument as before $\text{MS}(j, i, k) = S_{ik}$. Thus S_{ik} is redundant. If B_k holds for (i, j), then analogously $\text{MS}(i, j, k) = S_{ij}$, and S_{ij} is redundant. □

Example 3.11 *Suppose we wish to compute a Gröbner basis for the ideal $I = \{f_1, f_2, f_3, f_4\}$ with respect to some given term order. Then we initially have six S-polynomials to compute.*

Suppose further that $\text{LM}(f_1) = x^2y^2, \text{LM}(f_2) = y^2z, \text{LM}(f_3) = x^2z$, *and*
$\text{LM}(f_4) = xyz$. *Then*

$$\text{LCM}\{\text{LM}(f_1), \text{LM}(f_2)\} = \text{LCM}\{\text{LM}(f_1), \text{LM}(f_3)\} =$$

$$\text{LCM}\{\text{LM}(f_2), \text{LM}(f_3)\} = \text{LCM}\{\text{LM}(f_1), \text{LM}(f_4)\} = x^2y^2z,$$

$$\text{LCM}\{\text{LM}(f_2), \text{LM}(f_4)\} = xy^2z, \text{ and } \text{LCM}\{\text{LM}(f_3), \text{LM}(f_4)\} = x^2yz.$$

Note that criterion M holds for (f_1, f_4), criterion F holds for (f_2, f_3), and criterion B_4 holds for (f_2, f_3), but in no other cases do any of the above defined criteria hold. Thus $S(f_1, f_4)$ and $S(f_2, f_3)$ do not need to be computed, and we have reduced the number of S-polynomials by one third.

As it turns out, the syzygy S_{13} is also redundant, even though the Gebauer-Möller criteria do not identify this redundancy. To see that S_{13} is indeed redundant, observe

$$S_{13} = S_{14} - yS_{34} \text{ by } S_{134} \in S^{(2)} \text{ and}$$

$$S_{14} = S_{12} + xS_{24} \text{ by } S_{124} \in S^{(2)}.$$

So we have

$$S_{13} = S_{12} + xS_{24} - yS_{34}.$$

Therefore, $\{S_{12}, S_{24}, S_{34}\}$ generate $S^{(1)}$.

Gebauer and Möller's improved version of Buchberger's Algorithm is as follows:

INPUT: $F = \{f_1, \ldots, f_s\} \subseteq K[x_1, \ldots, x_n]$ with $f_i \neq 0$ $(1 \leq i \leq s)$
OUTPUT: $G = \{g_1, \ldots, g_t\}$, a Gröbner basis for $\langle f_1, \ldots, f_s \rangle$
INITIALIZATION: $G := \{f_1\}, SP := \emptyset$
FOR $t := 2$ to s
$SP := updatePairs(SP, f_t)$
$G := G \cup \{f_t\}$
$R := s$
WHILE $SP \neq \emptyset$ DO
Choose any $(f, g) \in SP$
$SP := SP - \{(f, g)\}$
$S(f, g) \xrightarrow{G}_+ h$, where h is reduced with respect to G
IF $h \neq 0$ THEN
$f_{R+1} := h$
$SP := updatePairs(SP, f_{R+1})$
$G := G \cup \{f_{R+1}\}$, $R := R + 1$

Table 3.1: Gebauer and Möller's Improved Buchberger Algorithm

The subalgorithm *updatePairs* is applied to SP by forming all polynomial pairs (f_i, f_t), where $i < t$, that may be formed from the new polynomial at hand, f_t, and then eliminating those pairs in SP for which criterion M, F, or B_k hold.

Even though the Gebauer-Möller criteria do not always produce a minimal set of generators for $S^{(1)}$, it has become the consensus among implementers of Buchberger's Algorithm that applying the Gebauer-Möller criteria results in a set of generators for $S^{(1)}$ that is nearly minimal (see, for example,[5] and [7]). Since generally implementers of Buchberger's Algorithm wish to find a Gröbner basis as efficiently as possible, minimalization techniques which could be used to find a

truly minimal set of generators for $S^{(1)}$ are typically not utilized since employing such techniques would slow down the Gröbner basis computation process. It is worth noting, however, that in [5], M. Caboara et al. describe a procedure for finding a minimal set of generators for $S^{(1)}$, while retaining the same efficiency as the usual application of the Gebauer-Möller criteria. This procedure described in [5] does have the added requirement that the input polynomials for the algorithm must be homogeneous.

Chapter 4

The "Sugar" Strategy

In the course of computing the Gröbner basis of an ideal, one must decide upon a procedure for choosing the order in which S-polynomials are reduced. From experience, this selection strategy may have a very significant effect upon the efficiency of the Gröbner basis computation. Unfortunately, a strategy which proves to be quite efficient for a particular polynomial ideal with a given term order could be extremely inefficient for other ideals and/or other term orders. Although the effectiveness of any strategy will vary, in [4] Buchberger suggests choosing an S-polynomial (f_i, f_j) if the least common multiple of the leading terms is minimal with respect to the given term order. Assuming $i < j$, in the case of a tie, we would give priority to the S-polynomial with the least j. Additional ties are not possible if we apply the Gebauer-Möller criteria to eliminate redundant S-polynomials. Recall that the F criterion of Gebauer-Möller requires that we eliminate pairs (f_i, f_k) if $\exists j < i$, such that $\text{LCM}\{\text{LM}(f_j), \text{LM}(f_k)\} = \text{LCM}\{\text{LM}(f_i), \text{LM}(f_k)\}$. Testing this strategy, known as Buchberger's *normal selection strategy*, has shown that it works rather well in the case of degree-compatible term orders such as degree lexicographic and degree reverse lexicographic. In the case of the lexicographic term order, however, the normal selection strategy has been shown to have a negative effect on the algorithm (see [9]).

Suppose $I = \langle f_1, \ldots, f_s \rangle$ is an ideal in $K[x_1, \ldots, x_n]$, and we wish to carry out Buchberger's Algorithm using the lexicographic term order and the normal selection strategy. Once the algorithm produces two polynomials without the variable x_1, the algorithm will run on these polynomials, disregarding all others, until it has computed a Gröbner basis for the ideal generated by these two polynomials. Then as further polynomials are introduced, we must compute Gröbner bases for further subproblems. In fact it is often the case that we create so many subproblems, that for the full problem of computing a Gröbner basis for I, this strategy is worse than carrying out the Gröbner basis computation with no strategy at all (ie. the arbitrary selection strategy).

Although the lexicographic term order can make the Gröbner basis computation process particularly time-consuming, there are methods for speeding up the process. One option is to *homogenize* the generators of the ideal, and to carry out Buchberger's Algorithm only for *homogeneous ideals,* and in increasing degrees. We will refer to this method as the "homogeneous algorithm." Experiments have shown that the choice of different selection strategies for choosing the order for reducing the S-polynomials of a homogeneous ideal will generally have minimal impact on the overall efficiency of Buchberger's Algorithm (see [9]).

Definition 4.1 *A polynomial $f \in K[x_1, \ldots, x_n]$ is homogeneous if the total degree of every term is the same.*

Definition 4.2 *An ideal $I \subseteq K[x_1, \ldots, x_n]$ is a homogeneous ideal provided that $I = \langle f_1, \ldots, f_s \rangle$ where each f_i is homogeneous.*

Let $F = \{f_1, \ldots, f_s\} \in K[x_1, \ldots, x_n]$, and suppose we wish to compute a Gröbner basis for $\langle F \rangle$ with respect to a given term order via the homogeneous algorithm. The first step is to homogenize each f_i. So let f be any polynomial in $K[x_1, \ldots, x_n]$. Let d be the total degree of f. We define the homogenization of f to be $f^h := Y^d f(\frac{x_1}{Y}, \ldots, \frac{x_n}{Y}) \in K[x_1, \ldots, x_n, Y]$. Note that f^h is homogeneous. We denote the homogenization of each f_i in F by f_i^h, and let $F^h = \{f_1^h, \ldots, f_s^h\} \in K[x_1, \ldots, x_n, Y]$. A Gröbner basis for $\langle F^h \rangle$ may be found through implementing Buchberger's Algorithm on F^h. We denote this Gröbner basis as G^h, and seek to transform G^h into a Gröbner basis for $\langle F \rangle$ through *dehomogenization.* That is, we consider each polynomial g_i^h of G^h and set $Y = 1$ for all terms in which a power of the new variable Y appears. Thus, we create a set of polynomials $G = \{g_1, \ldots, g_t\} \in K[x_1, \ldots, x_n]$. In many cases, it will be the case that G is a Gröbner basis for $\langle F \rangle$; however, this is not always so. It may be that the Gröbner basis of $\langle F^h \rangle$ is much larger than the Gröbner basis of $\langle F \rangle$. So there are cases in which the homogeneous algorithm fails to find a Gröbner basis for $\langle F \rangle$. Therefore, we recognize that even though it is experimentally known that use of the homogeneous algorithm may significantly reduce the number of S-polynomials that must be reduced, this approach is highly problematic because the Gröbner

basis of the ideal generated by the homogenized polynomials can be much larger than the Gröbner basis of the original ideal (see [9]).

In [9], Giovini et al. recommend an improved version of Buchberger's Algorithm, called *The Sugar Strategy* that seeks to exploit the strategic benefits of homogenization without introducing the aforementioned, undesired effect.

Definition 4.3 *Let $I = \langle f_1, \ldots, f_s \rangle \subseteq K[x_1, \ldots, x_n]$. Then for each initial f_i, the sugar of f_i, denoted Sug_{f_i}, is defined to be the total degree of f_i.*

Definition 4.4 *The sugar of a pair (f_i, f_j), denoted $Sug_{f_i f_j}$ is defined as follows:*

$$Sug_{f_i f_j} := max(Sug_{f_i} \text{ - } deg \text{ LM}(f_i),\ Sug_{f_j} \text{ - } deg \text{ LM}(f_j)) +$$

$$deg \text{ LCM}(\text{LM}(f_i), \text{LM}(f_j)).$$

Given $I = \langle f_1, \ldots, f_s \rangle \subseteq K[x_1, \ldots, x_n]$, we may associate with each f_i a homogeneous polynomial of degree equal to the sugar of f_i. As we seek to find a Gröbner basis for I, similar associations are made for each polynomial in the course of the algorithm. The purpose of making these informal associations is to keep track of the degree that each polynomial would have been if we had used the homogeneous algorithm.

The sugar strategy is implemented through comparing all pairs (f_i, f_j) where $i < j$ and reducing first the S-polynomial corresponding to the pair with the least sugar. Ties are then broken using Buchberger's normal selection strategy. To use the sugar strategy to improve the versions of Buchberger's Algorithm given in Table 2.1 and Table 3.1, we replace the "Choose any $(f, g) \in SP$" instruction with instructions to "Choose the $(f, g) \in SP$ with least sugar" and if there are ties, "Break ties by choosing among the $(f, g) \in SP$ with least sugar a pair with least LCM{LM(f),LM(g)}." In the case of the improved algorithm given in Table 3.1 this pair would be unique since we are using the Gebauer-Möller criteria to eliminate redundant pairs. The algorithm in Table 2.1 would require that further ties be broken through making an arbitrary choice among those pairs with least LCM{LM(f),LM(g)}.

Chapter 5

Faugère's F4 Algorithm

The improvements to Buchberger's Algorithm that Jean-Charles Faugère describes in [7] also concern strategies for choosing the order in which S-polynomials will be reduced. Although experiments have shown that *The Sugar Strategy* significantly improves the efficiency of Buchberger's Algorithm for a variety of benchmark problems (see [9]), Faugère has observed an opportunity for further optimization. In particular, Faugère suggests simultaneously reducing several polynomials by a list of polynomials (i.e., reduce $S(f_i, f_j)$ *and* $S(f_k, f_l)$ by $G = \{g_1, g_2, g_3\}$).

Suppose we wish to compute a Gröbner basis for $I = \langle f_1, \ldots, f_s \rangle \subseteq K[x_1, \ldots, x_n]$. Then instead of choosing one pair, (f_i, f_j), for reduction, we select a subset of pairs. Let SP be the set of all pairs (f_i, f_j), with $f_i \neq f_j$. We must define a function, $Sel(SP)$, which selects a subset of SP for reduction. Let

$$d := min\{deg(\text{LCM}\{\text{LM}(f_i), \text{LM}(f_j)\}),\ (f_i, f_j) \in SP\}.$$

Then during each iteration of Faugère's F4 algorithm, we may select the subset of SP :

$$SP_d := \{(f_i, f_j) \in P \mid deg(\text{LCM}\{\text{LM}(f_i), \text{LM}(f_j)\}) = d\}.$$

This is the selection procedure which Faugère recommmends in [7], and for now, we take this to be the definition of $Sel(SP)$.

Definition 5.1 *Let $I = \langle f_1, \ldots, f_s \rangle \subseteq K[x_1, \ldots, x_n]$, and let (f_i, f_j) with $f_i \neq f_j$ be a pair of polynomials. Then we define $Left(f_i) := (t_i, f_i)$ where t_i is the monomial such that $t_i \text{LM}(f_i) = \text{LCM}(\text{LM}(f_i), \text{LM}(f_j))$. Similarly, we define $Right(f_j) := (t_j, f_j)$ where t_j is the monomial such that $t_j \text{LM}(f_j) = \text{LCM}(\text{LM}(f_i), \text{LM}(f_j))$. Given a tuple (t_k, f_k), we define $mult(t_k, f_k) := t_k f_k$.*

Definition 5.2 *Let SP_d be a subset of SP as described above, and define $Left(SP_d) = \{Left(f_i)\}$ for $(f_i, f_j) \in SP_d$. Similarly, define $Right(SP_d) = \{Right(f_j)\}$ for $(f_i, f_j) \in SP_d$.*

INPUT: $F = \{f_1, \ldots, f_s\} \subseteq K[x_1, \ldots, x_n]$ with $f_i \neq 0$ $(1 \leq i \leq s)$,
OUTPUT: $G = \{g_1, \ldots, g_t\}$, a Gröbner basis for $\langle f_1, \ldots, f_s \rangle$
INITIALIZATION: $G := F, SP := \{\{f_i, f_j\} | f_i \neq f_j \in G\}, d := 0$
WHILE $SP \neq \emptyset$ DO
  $d := d + 1$
  $SP_d := Sel(SP)$
  $SP := SP - \{SP_d\}$
  $L_d := Left(SP_d) \cup Right(SP_d)$
  $\tilde{F}_d^+ := Reduction(L_d, G)$
  FOR $h \in \tilde{F}_d^+$ DO
    $SP := SP \cup \{(h, g) \mid g \in G\}$
    $G := G \cup \{h\}$

Table 5.1: Faugère's F4 Algorithm

In Faugère's F4 Algorithm presented above, we must modify the standard reduction procedure used in Buchberger's Algorithm to account for the fact that we are now reducing a subset of $K[x_1, \ldots, x_n]$ by G rather than reducing a single polynomial modulo G. Thus the subalgorithm $Reduction(L_d, G)$ is as follows:

INPUT: L_d, G as defined in Table 5.1
OUTPUT: $\tilde{F}^+$ = A finite subset of $K[x_1, \ldots, x_n]$ (possibly the empty set)
$F := SymbolicPreprocessing(L_d, G)$
$\tilde{F} :=$ Reduction to row echelon form of F with respect to an ordering $<$
$\tilde{F}^+ := \{f \in \tilde{F} \mid \text{LM}(f) \notin \text{LM}(F)\}$

Table 5.2: Reduction Subalgorithm

To construct the matrix F, which will be reduced to row echelon form, we carry out some symbolic preprocessing. Each row in F will correspond to a polynomial, and each column will correspond to a particular monomial. We denote the set of all monomials in F by $M(F)$. The $SymbolicPreprocessing(L_d, G)$ subalgorithm is given below in Table 5.3.

Although F4, as presented above, has been experimentally shown to be an efficient means to compute a Gröbner basis for many benchmark problems (see [7]), there are several ways to modify the F4 algorithm, which could result in greater efficiency. One possible modification would be to eliminate pairs in SP for which the Gebauer-Möller criteria M, F, or B_k hold. Another modification would be to reduce all rows during the reduction subalgorithm.

Faugère's F4 Algorithm including these two modifications is given in Table 5.4.

INPUT: L_d, G as defined in Table 5.1
OUTPUT: A matrix F of polynomials in $K[x_1, \ldots, x_n]$
INITIALIZATION: $F := \{t * f \mid (t, f) \in L_d\}$,
$Done := \mathrm{LM}(F)$
WHILE $M(F) \neq Done$ DO
  Choose an $m \in M(F) - Done$
  $Done := Done \cup \{m\}$
  IF m is reducible modulo G THEN
    $m = m' * \mathrm{LM}(g)$ for some $g \in G$ and some $m' \in K[x_1, \ldots, x_n]$
    $F := F \cup \{m' * g\}$

Table 5.3: SymbolicPreprocessing Subalgorithm

INPUT: $F = \{f_1, \ldots, f_s\} \subseteq K[x_1, \ldots, x_n]$ with $f_i \neq 0$ $(1 \leq i \leq s)$,
OUTPUT: $G = \{g_1, \ldots, g_t\}$, a Gröbner basis for $\langle f_1, \ldots, f_s \rangle$
INITIALIZATION: $G := \emptyset, SP := \emptyset, d := 0$
WHILE $F \neq \emptyset$ DO
  $f := first(F)$
  $F := F - \{f\}$
  $SP := updatePairs(SP, f)$
  $G := G \cup \{f\}$
WHILE $SP \neq \emptyset$ DO
  $d := d + 1$
  $SP_d := Sel(SP)$
  $SP := SP - \{SP_d\}$
  $L_d := Left(SP_d) \cup Right(SP_d)$
  $(\tilde{F}_d^+, F_d) := Reduction(L_d, G, (F_i)_{i=1,\ldots,(d-1)})$
  FOR $h \in \tilde{F}_d^+$ DO
    $SP := updatePairs(SP, h)$
    $G := G \cup \{h\}$

Table 5.4: Modified F4 Algorithm

Here, the subalgorithm *updatePairs* remains as in Table 3.1, and $first(F)$ chooses a polynomial f such that $\mathrm{LM}(f)$ is minimal in F with respect to the given term order. The subalgorithm *Reduction* changes in that it now takes an additional argument, $(F_i)_{i=1,\ldots,(d-1)}$, and also in that it returns the result of *SymbolicPreprocessing*. The *SymbolicPreprocessing* subalgorithm is modified as indicated below in Table 5.5.

The modification of F4 involving the reduction of all rows is achieved through the subalgorithm $Simplify$. The $Simplify$ subalgorithm is shown below in Table 5.6.

Example 5.3 *To see how the Modified F4 Algorithm works in practice, we consider the Cyclic 4 problem, using the degree reverse lexicographical term order with* $x_0 > x_1 > x_2 > x_3$.

INPUT: L_d, G as defined in Table 5.1,
The matrices $(F_i)_{i=1,\ldots,(d-1)}$
OUTPUT: A matrix F of polynomials in $K[x_1, \ldots, x_n]$
INITIALIZATION: $F := \{mult(Simplify(m, f, (F_i)_{i=1,\ldots,(d-1)})) \mid (m, f) \in L_d\}$,
$Done := \mathrm{LM}(F)$
WHILE $M(F) \neq Done$ DO
  Choose an $m \in M(F) - Done$
  $Done := Done \cup \{m\}$
  IF m is reducible modulo G THEN
    $m = m' * \mathrm{LM}(g)$ for some $g \in G$ and some $m' \in K[x_1, \ldots, x_n]$
    $F := F \cup \{mult(Simplify(m', f, (F_i)_{i=1,\ldots,(d-1)}))\}$

Table 5.5: Modified SymbolicPreprocessing Subalgorithm

INPUT: t a monomial in $K[x_1, \ldots, x_n]$, f a polynomial in $K[x_1, \ldots, x_n]$
The matrices $(F_i)_{i=1,\ldots,(d-1)}$
OUTPUT: A tuple (a, b) where a is a monomial in $K[x_1, \ldots, x_n]$
and b is a polynomial in $K[x_1, \ldots, x_n]$
FOR $u \in$ list of divisors of t DO
  IF $\exists j (1 \leq j < d)$ such that $(u * f) \in F_j$ THEN
  $\tilde{F}_j$ is the row echelon form of F_j
  with respect to the ordering $<$
  there exists a unique $p \in \tilde{F}_j^+$ such that
  $\mathrm{LM}(p) = \mathrm{LM}(u * f)$
  IF $u \neq t$ THEN
    RETURN $Simplify(\frac{t}{u}, p, (F_i)_{i=1,\ldots,(d-1)})$
  ELSE
    RETURN $(1, p)$
RETURN (t, f)

Table 5.6: Simplify Subalgorithm

Let $F = \{f_1 = x_0x_1x_2x_3 - 1, f_2 = x_0x_1x_2 + x_0x_1x_3 + x_0x_2x_3 + x_1x_2x_3, f_3 = x_0x_1 + x_1x_2 + x_0x_3 + x_2x_3, f_4 = x_0 + x_1 + x_2 + x_3\}$.

After two iterations through the first WHILE loop, we enter the second WHILE loop and have $G = \{f_4\}$, $SP_1 = \{(f_3, f_4)\}$, *and* $L_1 = \{(1, f_3), (x_1, f_4)\}$. *We now enter into SymbolicPreprocessing*$(L_1, G, \emptyset)$; $F_1 = L_1$, $Done = \mathrm{LM}(F_1) = \{x_0x_1\}$ *and* $M(F_1) = \{x_0x_3, x_0x_1, x_1^2, x_1x_2, x_1x_3, x_2x_3\}$, *we choose an element in* $M(F_1) - Done$, *say* x_0x_3, *but* x_0x_3 *is reducible modulo* G; *we now have* $Done = \{x_0x_1, x_0x_3\}$, $F_1 = F_1 \cup \{x_3f_4\}$, *and* $M(F_1) = M(F_1) \cup \{x_3^2\}$. *Since all other elements of* $M(F_1)$ *are not reducible modulo* G, *SymbolicPreprocessing returns* $F_1 = \{x_3f_4, f_3, x_1f_4\}$. *In matrix form, we have:*

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

where the rows correspond to the elements of F_1 *and the columns correspond to the terms in* $M(F_1)$. *Since* x_3f_4 *is the first polynomial listed in* F_1, *the first row in the matrix corresponds to the polynomial* x_3f_4. *Similarly, since* x_0x_1 *is the greatest term in* $M(F_1)$ *with respect to our degree reverse lexicographical term order, the first column in the matrix corresponds to the* x_0x_1 *term. Now we reduce our matrix to row echelon form and obtain:*

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & 0 & 2 & 0 & 1 \end{pmatrix}$$

which implies that $\widetilde{F}_1 = \{f_5 = x_0x_3 + x_1x_3 + x_2x_3 + x_3^2, f_6 = x_0x_1 + x_1x_2 - x_1x_3 - x_3^2,\ f_7 = x_1^2 + 2x_1x_3 + x_3^2\}$. *Since* $x_0x_1, x_0x_3 \in \mathrm{LM}(F_1)$, *we have* $(\widetilde{F}_1^{+} = \{f_7\}$, *and now* $G = \{f_4, f_7\}$. *Note that we do not need to reduce the pair* (f_4, f_7) *because the least common multiple of the leading monomials is* $x_0x_1^2$, *which is a multiple of* x_0x_1. *Recall that* x_0x_1 *is the least common multiple of the leading monomials of the pair* (f_3, f_4). *Thus, criterion* M *of Gebauer-Möller holds for* (f_4, f_7). *At this point, we temporarily exit the second WHILE loop, but remain within the first WHILE loop.*

We now consider f_2 *for the first time, and as we re-enter the second WHILE loop, we have* $SP_2 = \{(f_2, f_4)\}$. *Thus* $L_2 = \{(1, f_2), (x_1x_2, f_4)\}$. *Then during SymbolicPreprocessing, we attempt to simplify* $(1, f_2)$ *and* (x_1x_2, f_4) *with* F_1. *We see that* $x_1f_4 \in F_1$ *and* $\mathrm{LM}(f_6) = \mathrm{LM}(x_1f_4) = x_0x_1$, *so* $Simplify(x_1x_2, f_4, F_1) = (x_2, f_6)$. *At this point,* $F_2 = \{f_2, x_2f_6\}$ *and* $M(F_2) = \{x_0x_1x_2, x_1x_2^2, x_0x_1x_3,\ x_0x_2x_3, x_1x_2x_3,\ x_2x_3^2\}$. *Done* $= \mathrm{LM}(F_2) = x_0x_1x_2$. *Now we choose any element of* $M(F_2) - Done$. *We select* $x_0x_1x_3$, *which is reducible modulo* G. *Indeed,* $x_0x_1x_3 = x_1x_3\mathrm{LM}(f_4)$, *so we* $Simplify(x_1x_3, f_4, F_1)$. *In the Simplify subalgorithm, we initially choose* $u = d$, *a choice that results in the subalgorithm returning* $Simplify(x_1, f_5, F_1)$. *Then after carrying out* $Simplify(x_1, f_5, F_1)$, *we return the tuple* (x_1, f_5). *The mult function is immediately applied to this tuple, thereby yielding* x_1f_5, *which is appended to* F_2. *After testing the remaining elements of* $M(F_2)$ *for reducibility modulo* G, *we find that only* $x_0x_2x_3$ *and* $x_1^2x_3$ *are reducible. After going through the simplification process for each, we ultimately obtain* $F_2 = \{x_2f_5, x_3f_7, x_1f_5, f_2, x_2f_6\}$. *Because of the new elements appended to* F_2, $x_2^2x_3, x_2x_3^2$, *and*

x_3^3 *are added to* $M(F_2)$. *These new elements of* $M(F_2)$ *are not reducible modolo* G. *Therefore, SymbolicPreprocessing returns* $F_2 = \{x_2f_5, x_3f_7, x_1f_5, f_2, x_2f_6\}$, *or in matrix form:*

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \end{pmatrix}$$

where the rows correspond to the elements of F_2 *and the columns correspond to the terms in* $M(F_2)$. *Since* x_2f_5 *is the first polynomial listed in* F_2, *the first row in the matrix corresponds to the polynomial* x_2f_5. *Similarly, since* $x_0x_1x_2$ *is the greatest term in* $M(F_2)$ *with respect to our degree reverse lexicographical term order, the first column in the matrix corresponds to the* $x_0x_1x_2$ *term. Now we reduce our matrix to row echelon form and obtain:*

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & -1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 \end{pmatrix}$$

which implies that $\widetilde{F}_2 = \{f_8 = x_0x_2x_3 + x_1x_2x_3 + x_2^2x_3 + x_2x_3^2, f_9 = x_1^2x_3 + 2x_1x_3^2 + x_3^3,\ f_{10} = x_0x_1x_3 + x_1x_2x_3 - x_1x_3^2 - x_3^3,\ f_{11} = x_0x_1x_2 - x_1x_2x_3 - x_2^2x_3 + x_1x_3^2 - x_2x_3^2 + x_3^3,\ f_{12} = x_1x_2^2 + x_2^2x_3 - x_1x_3^2 - x_3^3\}$. *Since* $x_0x_2x_3, x_1^2x_3, x_0x_1x_3, x_0x_1x_2 \in \mathrm{LM}(F_2)$, *we have* $(\widetilde{F}_2^{+} = \{f_{12}\}$, *and now* $G = \{f_4, f_7, f_{12}\}$. *Note that we do not need to reduce the pair* (f_4, f_{12}) *because the least common multiple of the leading monomials is* $x_0x_1x_2^2$. *As was the case for the pair* (f_4, f_7), *criterion* M *of Gebauer-Möller holds for* (f_4, f_{12}). *At this point, we temporarily exit the second WHILE loop, but remain within the first WHILE loop.*

In the next step, we consider f_1 *for the first time, and as we re-enter the second WHILE loop, we have* $SP_3 = \{(f_1, f_4), (f_7, f_{12})\}$. *Thus* $L_3 = \{(1, f_1), (x_1x_2x_3, f_4),\ (x_2^2, f_7), (x_1, f_{12})\}$. *Then during SymbolicPreprocessing, we recursively call* $Simplify$: $Simplify(x_1x_2x_3, f_4) = Simplify(x_2x_3, f_6) = Simplify(x_3, f_{11}) = (x_3, f_{11})$. *We now have* $F_3 = \{f_1, x_3f_{11}, x_2^2f_7, x_1f_{12}\}$. *Then after carrying out the remainder of the SymbolicPreprocessing, we ultimately obtain* $F_3 = \{f_1, x_3f_{11}, x_2^2f_7,\ x_1f_{12},\ x_3f_{12},\ x_3f_9\}$. *After constructing the matrix* F_3 *and reducing to row echelon form, as was done for* F_1 *and* F_2, *we see that* $\widetilde{F}_3 = \{f_{13} = x_1^2x_2^2 - x_2^2x_3^2 + 2x_1x_3^3 + 2x_3^4,\ f_{14} = x_0x_1x_2x_3 - 1,\ f_{15} = -x_1x_2x_3^2 - x_2^2x_3^2 + x_1x_3^3 - x_2x_3^3 + x_3^4 + 1,\ f_{16} = x_1x_2^2x_3 + x_2^2x_3^2 - x_1x_3^3 - x_3^4,$

$f_{17} = x_1^2x_3^2 + 2x_1x_3^3 + x_3^4\}$. *Thus, we have that an entire row in the matrix* F_3 *reduces to zero. In other words, the rank of* F_3 *is five, as is demonstrated by the fact that* $\widetilde{F}_3$ *contains five polynomials. After comparing the leading monomials of* $\widetilde{F}_3$ *with* $\mathrm{LM}(F_3)$, *we see that* $\widetilde{F}_3^+ = \{f_{15}, f_{17}\}$, *and now* $G = \{f_4, f_7, f_{12}, f_{15}, f_{17}\}$. *At this point,* $F = \emptyset$, *but the algorithm continues since new elements are added to* SP *via the* $updatePairs$ *subalgorithm. Indeed, many more iterations through the "WHILE* $SP \neq \emptyset$*" loop are required before a Gröbner basis for* F *will be obtained.*

In addition to the two modifications discussed above, another modification to F4 that could improve the efficiency of a Gröbner basis computation, would be to redefine the $Sel(SP)$ function. An obvious way to redefine $Sel(SP)$ would to have the function select the subset of pairs in SP with least sugar.

Although F4 is known to be an efficient algorithm for Gröbner basis computation, it is rather unclear what effect various modications to F4 will have on the efficiency of the algorithm. F4 is now the default algorithm used for Gröbner basis computation in most computer algebra systems, so certainly Faugère's algorithm has been well received. But, as we mention above, there are several variants of F4. We will now turn our attention to investigating how different variants of F4 compare through testing each variant on a variety of benchmark ideals.

Chapter 6

Variants of F4 - Test Data Analysis

For this analysis, we consider ten benchmark ideals and eight variants of Faugère's F4 algorithm while always choosing GF(31991) for the coefficient field. The author carried out all tests using an UltraSparc 650Mz Processor with 1024M of memory. The ten ideals studied are: Cyclic 6, Cyclic 7, Cyclic 8, Cyclic 9, Katsura 7, Katsura 8, Katsura 9, Katsura 10, hCyclic 8, and f633. Each may be found in Appendix A. Note that hCyclic 8 and f633 are homogeneous ideals, whereas the others are not. The eight variants of F4 that we compare are all combinations of the three modifications discussed in the previous section. We number the variants as follows:

1. F4 with no modification
2. Reduce all rows
3. Use sugar
4. Check for deletable pairs
5. Reduce all rows and use sugar
6. Reduce all rows and check for deletable pairs
7. Use sugar and check for deletable pairs
8. Reduce all rows, use sugar, and check for deletable pairs

Note that in Example 5.3 discussed in the previous section, we use variant 6. For each test case, we observe the following:

time: the time (in seconds) required for the Gröbner basis computation
spairs: the number of S-pairs considered
maxr: the maximum number of rows appearing in the matrix at any time during the computation
maxc: the maximum number of columns appearing in the matrix at any time during the computation
GBsz: the Gröbner basis size (i.e., the number of polynomials in the Gröbner basis found)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
time:	0.168	0.231	0.165	0.166	0.230	0.233	0.151	0.226
spairs:	377	377	380	377	380	377	380	380
maxr:	144	139	148	144	139	139	148	139
maxc:	173	168	173	173	168	168	173	168
GBsz:	111	111	104	111	104	111	104	104

Table 6.1: Cyclic 6

From the data in Table 6.1, we observe that all eight variants of F4 produce a Gröbner basis in a fraction of a second. Although the timings are extremely close for all eight variants, those variants that include use of the sugar strategy each produce a smaller Gröbner basis for Cyclic 6. It is also the case that variants of F4, which include the reduce all rows option, result in a Gröbner basis while requiring less computer memory. We can conclude that the variants using the reduce all rows option require less computer memory, because the reduce all rows option reduces the maximum size of the matrix required to carry out the computation.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
time:	9.986	14.87	9.200	9.325	13.16	14.93	8.724	13.31
spairs:	2682	2682	2678	2682	2678	2682	2678	2678
maxr:	954	842	1097	954	883	842	1097	883
maxc:	1061	875	1211	1061	877	875	1211	877
GBsz:	592	592	556	592	556	592	556	556

Table 6.2: Cyclic 7

For the Cyclic 7 problem, we observe that a Gröbner basis is computed in less than 15 seconds for all eight variants of F4. Again we observe that the variants that include use of the sugar strategy produce a smaller Gröbner basis, whereas variants that include the reduce all rows option require a significantly smaller matrix to carry out the Gröbner basis computation. Although variants that make use of the reduce all rows option consistently use less computer memory, this advantage is offset by the fact that such variants are noticeably slower than other variants.

In the case of the Cyclic 8 problem, our results are consistent with the results previously seen in the Cyclic 6 and Cyclic 7 problems; however, our observed differences between the variants have grown in number and are increasingly magnified. Use of the sugar strategy now results in a Gröbner basis with 300+ fewer polynomials than other variants. The matrices used to compute a

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
time:	261.2	443.2	190.1	237.2	309.7	443.7	157.2	309.3
spairs:	7588	7588	6466	7586	6466	7588	6415	6462
maxr:	4908	3733	4659	4908	3087	3733	4659	3087
maxc:	5614	4394	5863	5614	4169	4394	5863	4169
GBsz:	1516	1516	1212	1515	1212	1517	1172	1208

Table 6.3: Cyclic 8

Gröbner basis for variants using the reduce all rows option have 1000+ fewer rows and columns than the matrices of other variants. Although checking for deletable pairs did not seem to improve the performance of the algorithm in the cases of Cyclic 6 and Cyclic 7, here we observe that this modified version of F4 can have positive effects on the algorithm, particularly when combined with sugar and/or reduce all rows. Note that variant 7 is the most efficient variant, finding a Gröbner basis 32.9 seconds faster than variant 3, the second most efficient variant for this benchmark problem. Note also that the computer memory saving effect of the reduce all rows option is greatest in variants 5 and 8, the two variants where the reduce all rows option is used in conjunction with checking for deletable pairs.

We do not include a table for the tests on the Cyclic 9 problem, since all eight variants of F4 failed to find a Gröbner basis for Cyclic 9 due to insufficient memory. Relative to Cyclic 7, we observed above a very substantial increase in the size of the matrix used to compute a Gröbner basis for Cyclic 8. As one would expect, when transitioning from the Cyclic 8 problem to the Cyclic 9 problem, the matrix grows considerably larger once more. Although the reduce all rows option helps to limit the growth of the matrix, the variants of F4 that include this option all failed to contain the size of the matrix enough to allow for a Gröbner basis computation for the Cyclic 9 problem within the constraints of the available 1024M of computer memory.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
time:	0.889	1.174	0.939	0.897	1.168	1.171	0.901	1.226
spairs:	377	377	377	377	377	377	377	377
maxr:	806	510	806	806	510	510	806	510
maxc:	832	554	832	832	554	554	832	554
GBsz:	76	76	76	76	76	76	76	76

Table 6.4: Katsura 7

For the Katsura 7 problem, all eight variants of F4 compute a Gröbner basis in about one second. The different variants have absolutely no effect on the total number of S-pairs considered or the size of the Gröbner basis. All variants that include the reduce all rows option require a significantly smaller matrix to carry out the computation; however, the effect of the reduce all rows option on the size of the matrix is neither enhanced nor lessened through combining this option with the sugar strategy and/or checking for deletable pairs.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
time:	6.783	7.637	6.831	6.867	7.626	7.722	6.820	7.799
spairs:	881	881	881	881	881	881	881	881
maxr:	1934	1136	1934	1934	1136	1136	1934	1136
maxc:	1970	1172	1970	1970	1172	1172	1970	1172
GBsz:	145	145	145	145	145	145	145	145

Table 6.5: Katsura 8

Like the Katsura 7 problem, we observe little difference between the eight variants of F4 in the case of the Katsura 8 problem. The number of S-pairs computed and the Gröbner basis size are exactly the same for all eight variants. The time required for the Gröbner basis computation is consistantly about 7 seconds. Variants which include the reduce all rows option compute a Gröbner basis for Katsura 8 while containing the growth of the matrix necessary to carry out the computation. Indeed, approximately 800 fewer rows and columns are present in the matrices for variants 2, 5, 6, and 8, than is the case for the variants that do not use the reduce all rows option.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
time:	50.75	53.79	51.32	51.21	53.87	54.60	51.22	53.66
spairs:	1974	1974	1974	1974	1974	1974	1974	1974
maxr:	4528	2436	4528	4528	2436	2436	4528	2436
maxc:	4555	2525	4555	4555	2525	2525	4555	2525
GBsz:	274	274	274	274	274	274	274	274

Table 6.6: Katsura 9

Again, the number of S-pairs computed and the Gröbner basis size are the same for all eight variants of F4. Variants that include the reduce all rows option compute a Gröbner basis for Katsura 9 while using matrices with 2000+ fewer rows and columns than other variants. The very modest differences in the time that each variant required to carry out the Gröbner basis

computation for Katsura 7 grew slightly larger in Katsura 8, and a bit larger again with Katsura 9. It is worth noting that variant 1 has been the most efficient in all three cases, and variant 7 has consistently been the second most efficient. Nevertheless, the slowest variant, variant 6, computes a Gröbner basis for Katsura 9 while requiring only 3.85 additional seconds than variant 1.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
time:	403.7	403.9	403.6	405.1	403.4	404.4	404.2	406.3
spairs:	4464	4464	4464	4464	4464	4464	4464	4464
maxr:	11777	5406	11777	11777	5406	5406	11777	5406
maxc:	11767	5362	11767	11767	5362	5362	11767	5362
GBsz:	539	539	539	539	539	539	539	539

Table 6.7: Katsura 10

As we would expect based on the results of testing Katsura 7, 8, and 9, all eight variants of F4 reduce the same number of S-pairs during the Gröbner basis computation for Katsura 10, and the size of the Gröbner basis found is the same for all variants. Although timings continue to differ only slightly between the eight variants, the variants which do not utilize the reduce all rows option use considerably more computer memory than the variants that do employ the reduce all rows option. Indeed variants 2, 5, 6, and 8 carry out the Gröbner basis computation with matrices less than half the size of other variants, using 5800+ less rows and columns while retaining the same efficiency the other four variants.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
time:	255.1	377.6	255.4	256.4	376.4	378.7	255.9	380.1
spairs:	7025	7025	7025	7025	7025	7025	7025	7025
maxr:	8583	3611	8583	8583	3611	3611	8583	3611
maxc:	10762	4285	10762	10762	4285	4285	10762	4285
GBsz:	1189	1189	1189	1189	1189	1189	1189	1189

Table 6.8: hCyclic 8

For the homogeneous Cyclic 8 problem, we see that the reduce all rows option has a very positive effect on the size of matrix necessary for the Gröbner basis computation but also a negative effect on the efficiency of the computation process. All four variants that use the reduce all rows option require matrices that are less than half the size of the matrices required by the other variants, yet the variants using the reduce all rows option need 50% more time to compute the

Gröbner basis.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
time:	0.636	0.560	0.637	0.637	0.562	0.565	0.645	0.563
spairs:	392	392	392	392	392	392	392	392
maxr:	2778	762	2778	2778	762	762	2778	762
maxc:	3110	985	3110	3110	985	985	3110	985
GBsz:	74	74	74	74	74	74	74	74

Table 6.9: f633

In the case of the f633 problem, which is also a homogeneous ideal, the Gröbner basis computation is carried out in less than one second for all eight variants. Although all eight variants are essentially equally efficient, it is interesting to note that the four variants which use the reduce all rows option are modestly faster than the four variants that do not utilize the reduce all rows option. The matrices used by variants that do not include the reduce all rows option grow to be over three times the size of matrices needed to compute a Gröbner basis by variants that do use the reduce all rows option. Like the Katsura n probems and the homogeneous Cyclic 8 problem, all variants of F4 find a Gröbner basis for the f633 ideal that is of the same size and results from the reduction of precisely the same number of S-pair reductions.

Chapter 7

Conclusion

The data above demonstrates that there is not one particular variant of F4 that is always the best choice. For the benchmark ideals considered, it appears that F4 with no modifications often computes a Gröbner basis in the least amount of time; however, in the case of Cyclic 8, this variant is much slower than variant 7. For all test problems considered, variant 7 is never a bad choice with respect to efficiency. Indeed in cases where variant 7 is not the most efficient, it is only marginally slower than the fastest variant. Thus, when efficiency of the Gröbner basis computation process is of particular concern, the results above suggest that using F4 together with the sugar strategy and also checking for deletable pairs is likely to be a good choice.

However, we must recall that all tests were conducted using the degree-reverse-lexicographic term order, and results will likely differ considerably for other term orders. Also, it is generally a rash assumption to assume that efficiency is the only concern during a Gröbner basis computation. As we observe in the case of the Cyclic 9 problem above, the amount of computer memory required to compute a Gröbner basis is also an important issue. Indeed if the amount of computer memory is insufficient for the computation, a Gröbner basis will not be found. The variants of F4 that include the reduce all rows option typically use less computer memory than other variants of F4, sometimes drastically less. Thus, if one uses a variant of F4 that does not include the reduce all rows option and finds that the Gröbner basis computation of a particular ideal with a given term order requires more computer memory than available resources will allow, the computation might be successful for an F4 variant that does include the reduce all rows option. Unfortunately, in the case of the Cyclic 9 problem, even the memory savings provided through use of the reduce all rows option did not allow for the computation of Gröbner basis within the bounds of the available 1024M of computer memory.

Based on the results of the Cyclic n problems, we have reason to believe that variants of F4

that use the sugar strategy will often produce a Gröbner basis that is closer to a minimal Gröbner basis (i.e., has fewer redundant polynomials) than other variants. Yet, further tests should be done to investigate, in the case of other ideals and/or term orders, whether variants of F4 that include the sugar strategy are always the closest to minimal Gröbner bases.

The Cyclic 8 problem for large coefficients was an intractable problem less than a decade ago [7]. Through finding a Gröbner basis for Cyclic 8 over GF(31991) for all eight variants of F4, we demonstrate that Faugère's F4 algorithm is indeed a particularly powerful algorithm for computing Gröbner bases, and F4 has rightly become the default algorithm for most major computer algebra systems.

Appendix A

Benchmark Polynomial Ideals

$x_0x_1x_2x_3x_4x_5 - 1$

$x_0x_1x_2x_3x_4 + x_1x_2x_3x_4x_5 + x_2x_3x_4x_5x_0 + x_3x_4x_5x_0x_1 + x_4x_5x_0x_1x_2 + x_5x_0x_1x_2x_3$

$x_0x_1x_2x_3 + x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5x_0 + x_4x_5x_0x_1 + x_5x_0x_1x_2$

$x_0x_1x_2 + x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_0 + x_5x_0x_1$

$x_0x_1 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_0$

$x_0 + x_1 + x_2 + x_3 + x_4 + x_5$

Table A.1: Cyclic6

$x_0x_1x_2x_3x_4x_5x_6 - 1$

$x_0x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_6x_0 + x_3x_4x_5x_6x_0x_1 + x_4x_5x_6x_0x_1x_2 + x_5x_6x_0x_1x_2x_3 + x_6x_0x_1x_2x_3x_4$

$x_0x_1x_2x_3x_4 + x_1x_2x_3x_4x_5 + x_2x_3x_4x_5x_6 + x_3x_4x_5x_6x_0 + x_4x_5x_6x_0x_1 + x_5x_6x_0x_1x_2 + x_6x_0x_1x_2x_3$

$x_0x_1x_2x_3 + x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5x_6 + x_4x_5x_6x_0 + x_5x_6x_0x_1 + x_6x_0x_1x_2$

$x_0x_1x_2 + x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_6 + x_5x_6x_0 + x_6x_0x_1$

$x_0x_1 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_0$

$x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6$

Table A.2: Cyclic7

$x_0x_1x_2x_3x_4x_5x_6x_7 - 1$

$x_0x_1x_2x_3x_4x_5x_6 + x_1x_2x_3x_4x_5x_6x_7 + x_2x_3x_4x_5x_6x_7x_0 +$
$x_3x_4x_5x_6x_7x_0x_1 + x_4x_5x_6x_7x_0x_1x_2 + x_5x_6x_7x_0x_1x_2x_3 +$
$x_6x_7x_0x_1x_2x_3x_4 + x_7x_0x_1x_2x_3x_4x_5$

$x_0x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_6x_7 + x_3x_4x_5x_6x_7x_0 +$
$x_4x_5x_6x_7x_0x_1 + x_5x_6x_7x_0x_1x_2 + x_6x_7x_0x_1x_2x_3 + x_7x_0x_1x_2x_3x_4$

$x_0x_1x_2x_3x_4 + x_1x_2x_3x_4x_5 + x_2x_3x_4x_5x_6 + x_3x_4x_5x_6x_7 +$
$x_4x_5x_6x_7x_0 + x_5x_6x_7x_0x_1 + x_6x_7x_0x_1x_2 + x_7x_0x_1x_2x_3$

$x_0x_1x_2x_3 + x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5x_6 + x_4x_5x_6x_7 +$
$x_5x_6x_7x_0 + x_6x_7x_0x_1 + x_7x_0x_1x_2$

$x_0x_1x_2 + x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_6 + x_5x_6x_7 + x_6x_7x_0 + x_7x_0x_1$

$x_0x_1 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_7 + x_7x_0$

$x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$

Table A.3: Cyclic8

$x_0x_1x_2x_3x_4x_5x_6x_7x_8 - 1$

$x_0x_1x_2x_3x_4x_5x_6x_7 + x_1x_2x_3x_4x_5x_6x_7x_8 + x_2x_3x_4x_5x_6x_7x_8x_0 +$
$x_3x_4x_5x_6x_7x_8x_0x_1 + x_4x_5x_6x_7x_8x_0x_1x_2 + x_5x_6x_7x_8x_0x_1x_2x_3 +$
$x_6x_7x_8x_0x_1x_2x_3x_4 + x_7x_8x_0x_1x_2x_3x_4x_5 + x_8x_0x_1x_2x_3x_4x_5x_6$

$x_0x_1x_2x_3x_4x_5x_6 + x_1x_2x_3x_4x_5x_6x_7 + x_2x_3x_4x_5x_6x_7x_8 +$
$x_3x_4x_5x_6x_7x_8x_0 + x_4x_5x_6x_7x_8x_0x_1 + x_5x_6x_7x_8x_0x_1x_2 +$
$x_6x_7x_8x_0x_1x_2x_3 + x_7x_8x_0x_1x_2x_3x_4 + x_8x_0x_1x_2x_3x_4x_5$

$x_0x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_6x_7 + x_3x_4x_5x_6x_7x_8 +$
$x_4x_5x_6x_7x_8x_0 + x_5x_6x_7x_8x_0x_1 + x_6x_7x_8x_0x_1x_2 +$
$x_7x_8x_0x_1x_2x_3 + x_8x_0x_1x_2x_3x_4$

$x_0x_1x_2x_3x_4 + x_1x_2x_3x_4x_5 + x_2x_3x_4x_5x_6 + x_3x_4x_5x_6x_7 +$
$x_4x_5x_6x_7x_8 + x_5x_6x_7x_8x_0 + x_6x_7x_8x_0x_1 + x_7x_8x_0x_1x_2 +$
$x_8x_0x_1x_2x_3$

$x_0x_1x_2x_3 + x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5x_6 + x_4x_5x_6x_7 +$
$x_5x_6x_7x_8 + x_6x_7x_8x_0 + x_7x_8x_0x_1 + x_8x_0x_1x_2$

$x_0x_1x_2 + x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_6 + x_5x_6x_7 + x_6x_7x_8 +$
$x_7x_8x_0 + x_8x_0x_1$

$x_0x_1 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_7 + x_7x_8 + x_8x_0$

$x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8$

Table A.4: Cyclic9

$$-x_0 + 2x_7^2 + 2x_6^2 + 2x_5^2 + 2x_4^2 + 2x_3^2 + 2x_2^2 + 2x_1^2 + x_0^2$$

$$-x_1 + 2x_7x_6 + 2x_6x_5 + 2x_5x_4 + 2x_4x_3 + 2x_3x_2 + 2x_2x_1 + 2x_1x_0$$

$$-x_2 + 2x_7x_5 + 2x_6x_4 + 2x_5x_3 + 2x_4x_2 + 2x_3x_1 + 2x_2x_0 + x_1^2$$

$$-x_3 + 2x_7x_4 + 2x_6x_3 + 2x_5x_2 + 2x_4x_1 + 2x_3x_0 + 2x_2x_1$$

$$-x_4 + 2x_7x_3 + 2x_6x_2 + 2x_5x_1 + 2x_4x_0 + 2x_3x_1 + x_2^2$$

$$-x_5 + 2x_7x_2 + 2x_6x_1 + 2x_5x_0 + 2x_4x_1 + 2x_3x_2$$

$$-x_6 + 2x_7x_1 + 2x_6x_0 + 2x_5x_1 + 2x_4x_2 + x_3^2$$

$$-1 + 2x_7 + 2x_6 + 2x_5 + 2x_4 + 2x_3 + 2x_2 + 2x_1 + x_0$$

Table A.5: Katsura7

$$-x_0 + 2x_8^2 + 2x_7^2 + 2x_6^2 + 2x_5^2 + 2x_4^2 + 2x_3^2 + 2x_2^2 + 2x_1^2 + x_0^2$$

$$-x_1 + 2x_8x_7 + 2x_7x_6 + 2x_6x_5 + 2x_5x_4 + 2x_4x_3 + 2x_3x_2 + 2x_2x_1 + 2x_1x_0$$

$$-x_2 + 2x_8x_6 + 2x_7x_5 + 2x_6x_4 + 2x_5x_3 + 2x_4x_2 + 2x_3x_1 + 2x_2x_0 + x_1^2$$

$$-x_3 + 2x_8x_5 + 2x_7x_4 + 2x_6x_3 + 2x_5x_2 + 2x_4x_1 + 2x_3x_0 + 2x_2x_1$$

$$-x_4 + 2x_8x_4 + 2x_7x_3 + 2x_6x_2 + 2x_5x_1 + 2x_4x_0 + 2x_3x_1 + x_2^2$$

$$-x_5 + 2x_8x_3 + 2x_7x_2 + 2x_6x_1 + 2x_5x_0 + 2x_4x_1 + 2x_3x_2$$

$$-x_6 + 2x_8x_2 + 2x_7x_1 + 2x_6x_0 + 2x_5x_1 + 2x_4x_2 + x_3^2$$

$$-x_7 + 2x_8x_1 + 2x_7x_0 + 2x_6x_1 + 2x_5x_2 + 2x_4x_3$$

$$-1 + 2x_8 + 2x_7 + 2x_6 + 2x_5 + 2x_4 + 2x_3 + 2x_2 + 2x_1 + x_0$$

Table A.6: Katsura8

$-x_0 + 2x_9^2 + 2x_8^2 + 2x_7^2 + 2x_6^2 + 2x_5^2 + 2x_4^2 + 2x_3^2 + 2x_2^2 + 2x_1^2 + x_0^2$

$-x_1 + 2x_9x_8 + 2x_8x_7 + 2x_7x_6 + 2x_6x_5 + 2x_5x_4 + 2x_4x_3 + 2x_3x_2 + 2x_2x_1 + 2x_1x_0$

$-x_2 + 2x_9x_7 + 2x_8x_6 + 2x_7x_5 + 2x_6x_4 + 2x_5x_3 + 2x_4x_2 + 2x_3x_1 + 2x_2x_0 + x_1^2$

$-x_3 + 2x_9x_6 + 2x_8x_5 + 2x_7x_4 + 2x_6x_3 + 2x_5x_2 + 2x_4x_1 + 2x_3x_0 + 2x_2x_1$

$-x_4 + 2x_9x_5 + 2x_8x_4 + 2x_7x_3 + 2x_6x_2 + 2x_5x_1 + 2x_4x_0 + 2x_3x_1 + x_2^2$

$-x_5 + 2x_9x_4 + 2x_8x_3 + 2x_7x_2 + 2x_6x_1 + 2x_5x_0 + 2x_4x_1 + 2x_3x_2$

$-x_6 + 2x_9x_3 + 2x_8x_2 + 2x_7x_1 + 2x_6x_0 + 2x_5x_1 + 2x_4x_2 + x_3^2$

$-x_7 + 2x_9x_2 + 2x_8x_1 + 2x_7x_0 + 2x_6x_1 + 2x_5x_2 + 2x_4x_3$

$-x_8 + 2x_9x_1 + 2x_8x_0 + 2x_7x_1 + 2x_6x_2 + 2x_5x_3 + x_4^2$

$-1 + 2x_9 + 2x_8 + 2x_7 + 2x_6 + 2x_5 + 2x_4 + 2x_3 + 2x_2 + 2x_1 + x_0$

Table A.7: Katsura9

$-x_0 + 2x_{10}^2 + 2x_9^2 + 2x_8^2 + 2x_7^2 + 2x_6^2 + 2x_5^2 + 2x_4^2 + 2x_3^2 +$
$2x_2^2 + 2x_1^2 + x_0^2$

$-x_1 + 2x_{10}x_9 + 2x_9x_8 + 2x_8x_7 + 2x_7x_6 + 2x_6x_5 + 2x_5x_4 + 2x_4x_3 +$
$2x_3x_2 + 2x_2x_1 + 2x_1x_0$

$-x_2 + 2x_{10}x_8 + 2x_9x_7 + 2x_8x_6 + 2x_7x_5 + 2x_6x_4 + 2x_5x_3 + 2x_4x_2 +$
$2x_3x_1 + 2x_2x_0 + x_1^2$

$-x_3 + 2x_{10}x_7 + 2x_9x_6 + 2x_8x_5 + 2x_7x_4 + 2x_6x_3 + 2x_5x_2 + 2x_4x_1 +$
$2x_3x_0 + 2x_2x_1$

$-x_4 + 2x_{10}x_6 + 2x_9x_5 + 2x_8x_4 + 2x_7x_3 + 2x_6x_2 + 2x_5x_1 + 2x_4x_0 +$
$2x_3x_1 + x_2^2$

$-x_5 + 2x_{10}x_5 + 2x_9x_4 + 2x_8x_3 + 2x_7x_2 + 2x_6x_1 + 2x_5x_0 + 2x_4x_1 +$
$2x_3x_2$

$-x_6 + 2x_{10}x_4 + 2x_9x_3 + 2x_8x_2 + 2x_7x_1 + 2x_6x_0 + 2x_5x_1 + 2x_4x_2 +$
x_3^2

$-x_7 + 2x_{10}x_3 + 2x_9x_2 + 2x_8x_1 + 2x_7x_0 + 2x_6x_1 + 2x_5x_2 + 2x_4x_3$

$-x_8 + 2x_{10}x_2 + 2x_9x_1 + 2x_8x_0 + 2x_7x_1 + 2x_6x_2 + 2x_5x_3 + x_4^2$

$-x_9 + 2x_{10}x_1 + 2x_9x_0 + 2x_8x_1 + 2x_7x_2 + 2x_6x_3 + 2x_5x_4$

$-1 + 2x_{10} + 2x_9 + 2x_8 + 2x_7 + 2x_6 + 2x_5 + 2x_4 + 2x_3 + 2x_2 + 2x_1 + x_0$

Table A.8: Katsura10

$x_0x_1x_2x_3x_4x_5x_6x_7 - x_8^8$

$x_0x_1x_2x_3x_4x_5x_6 + x_1x_2x_3x_4x_5x_6x_7 + x_2x_3x_4x_5x_6x_7x_0 +$
$x_3x_4x_5x_6x_7x_0x_1 + x_4x_5x_6x_7x_0x_1x_2 + x_5x_6x_7x_0x_1x_2x_3 +$
$x_6x_7x_0x_1x_2x_3x_4 + x_7x_0x_1x_2x_3x_4x_5$

$x_0x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_6x_7 + x_3x_4x_5x_6x_7x_0 +$
$x_4x_5x_6x_7x_0x_1 + x_5x_6x_7x_0x_1x_2 + x_6x_7x_0x_1x_2x_3 + x_7x_0x_1x_2x_3x_4$

$x_0x_1x_2x_3x_4 + x_1x_2x_3x_4x_5 + x_2x_3x_4x_5x_6 + x_3x_4x_5x_6x_7 +$
$x_4x_5x_6x_7x_0 + x_5x_6x_7x_0x_1 + x_6x_7x_0x_1x_2 + x_7x_0x_1x_2x_3$

$x_0x_1x_2x_3 + x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5x_6 + x_4x_5x_6x_7 +$
$x_5x_6x_7x_0 + x_6x_7x_0x_1 + x_7x_0x_1x_2$

$x_0x_1x_2 + x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_6 + x_5x_6x_7 + x_6x_7x_0 + x_7x_0x_1$

$x_0x_1 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_7 + x_7x_0$

$x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$

Table A.9: hCyclic8

$x_0x_5 - x_{10}^2$

$x_1x_6 - x_{10}^2$

$x_2x_7 - x_{10}^2$

$x_3x_8 - x_{10}^2$

$x_4x_9 - x_{10}^2$

$-4x_1x_5 - 4x_2x_5 - 4x_3x_5 - 4x_4x_5 + 4x_0x_6 - 4x_2x_6 - 4x_3x_6 - 4x_4x_6 +$
$4x_0x_7 + 4x_1x_7 - 4x_3x_7 - 4x_4x_7 + 4x_0x_8 + 4x_1x_8 + 4x_2x_8 - 4x_4x_8 +$
$4x_0x_9 + 4x_1x_9 + 4x_2x_9 + 4x_3x_9 + 2x_0x_{10} + 2x_1x_{10} + 2x_2x_{10} +$
$2x_3x_{10} + 2x_4x_{10} + x_{10}^2$

$2x_0x_{10} + 2x_1x_{10} + 2x_2x_{10} + 2x_3x_{10} + 2x_4x_{10} + x_{10}^2,$
$4x_1x_5 + 4x_2x_5 + 4x_3x_5 + 4x_4x_5 - 4x_0x_6 + 4x_2x_6 + 4x_3x_6 +$
$4x_4x_6 - 4x_0x_7 - 4x_1x_7 + 4x_3x_7 + 4x_4x_7 - 4x_0x_8 - 4x_1x_8 -$
$4x_2x_8 + 4x_4x_8 - 4x_0x_9 - 4x_1x_9 - 4x_2x_9 - 4x_3x_9 + 2x_5x_{10} +$
$2x_6x_{10} + 2x_7x_{10} + 2x_8x_{10} + 2x_9x_{10}x_{10}^2$

$2x_0 + 2x_1 + 2x_2 + 2x_3 + 2x_4 + x_{10}$

$2x_5 + 2x_6 + 2x_7 + 2x_8 + 2x_9 + x_{10}$

Table A.10: f633

BIBLIOGRAPHY

[1] W. W. Adams and P. Loustaunau, *An Introduction to Gröbner Bases*, Graduate Studies in Mathematics, vol. 3, American Mathematical Society, Providence, RI, 1994.

[2] D. Bayer, M. Stillman, *The design of Macaulay: A system for computing in algebraic geometry and commutative algebra*, 1986 ACM Symposium on Symbolic and Algebraic Computation, University of Waterloo, Ontario, 157-162, 1986.

[3] B. Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, PhD thesis, Innsbruck, 1965.

[4] B. Buchberger, *A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Bases*, in "EUROSAM 1979," Lecture Notes in Computer Science 72, Springer Verlag, Berlin-Heidelberg-New York, 1979, 3-21.

[5] M. Caboara, M. Kreuzer, and L. Robbiano, *Efficiently Computing Minimal Sets of Critical Pairs*, Preprint submitted to Journal of Symbolic Computation, 2003.

[6] D. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Springer Verlag, Berlin and New York, 1992.

[7] J. C. Faugere, *A New Efficient Algorithm for Computing Gröbner Bases (F4)*, Journal of Pure and Applied Algebra, 139(1-3):61-88, June 1999.

[8] R. Gebauer and H. M. Möller, *On an Installation of Buchberger's Algorithm*, J. Symbolic Computation, 6: 257-286, 1987.

[9] A. Giovini, T. Mora, G. Niesi, L. Robbiano, and C. Traverso, *"One Sugar Cube, Please" or Selection Strategies in the Buchberger Algorithm*, in Proc. International Symposium on Symbolic and Algebraic Computation ISSAC'91 (S.M. Watt ed.), ACM Press, New York, 1991, 49-54.

[10] H. M. Möller, *A Reduction Strategy for the Taylor Resolution*, Proc. EUROCAL 85, Springer L.N. in Comp. Sci. 162, 526-534.

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