Gröbner Bases computation and Gaussian Elimination

Dissertation

zur Erlangung des akademischen Grades
"Doktor der Naturwissenschaften"

Eingereicht von

Dipl.-Ing. Ana Maria Mandache,

Mai 1995

Erster Begutachter: o.Univ.-Prof. Dr. Bruno Buchberger
Zweiter Begutachter: o.Univ.-Prof. Dr. George E. Collins

Angefertigt am Forschungsinstitut für Symbolisches Rechnen
Technisch-Naturwissenschaftliche Fakultät
Johannes Kepler Universität Linz

Acknowledgements

I would like to thank my thesis advisor, Bruno Buchberger for introducing me to the subject of the connection between Gröbner bases and subresultants, and for all the support he provided during my Ph.D. studies. I also thank him for the efforts he made for creating and developing the institute RISC-Linz, which is a wonderful environment for research in symbolic computation.

I thank George Collins for the useful discussions we had and for the careful refereeing of my thesis, which helped me clarifying several ideas.

I am indebted to Hoon Hong who helped me much at the beginning of my studies at RISC, sharing his experience of research work and of life.

My Ph.D. studies were financially supported by a scholarship of the Austrian Academic Exchange Service (ÖAD) during 1990-1993 and by the EC project PoSSo (ESPRIT III Basic Research Action, no. 6846 and Austrian Fond zur Förderung der wissenschaftlichen Forschung P9181-TEC) during the academic year 1993-1994.

In the summer of 1994 I spent one month and a half as a short-time invited researcher at ISIS, Fujitsu Laboratories Ltd. in Numazu, Japan. During this visit, financially supported by Fujitsu Laboratories Ltd., I appreciated the excellent working environment in the Computer Algebra group.

I spent several months during my Ph.D. studies and especially during the academic year 1994–1995 in the computer algebra team led by Daniel Lazard at LITP, University Paris VI. I thank them for the hospitality and for providing me good conditions for finishing writing my thesis.

Several persons made a special contribution to my mathematical education during high school and university: Marcel Ţena, Horia Georgescu and Ioan Tomescu.

My husband Niculae Mandache and my family were beside me in spite of the geographical distance that separated us.

I also want to thank to all the members of the institute RISC-Linz, which were for me not only good colleagues but real friends.

Abstract

We investigate the connection between Gröbner basis computation and Gaussian elimination. Our main goal is to construct for Gröbner bases a concept similar to the one provided by subresultants for polynomial remainder sequences.

Determinant polynomials are generalised for multivariate polynomials. We associate to Gröbner bases computation Sylvester-like matrices having the columns indexed by terms and containing on each row the coefficients of a polynomial from the input basis, possibly shifted by multiplication by a term.

We show then that Gröbner basis polynomials computed during Buchberger algorithm can not, in general, be expressed as determinant polynomials of matrices of the form above. Even in simple examples where the input basis consists of three bivariate polynomials this is not always possible.

We consider then the more general problem of simulating the Gröbner basis computation by Gaussian elimination. We devised a simple tool, which we called reduction graph, for tracing the computations performed by a Gröbner basis algorithm or the row reductions in a matrix. With this aid we prove that Gröbner basis computation by the classical algorithm can not always be simulated by Gaussian elimination.

Next we generalise the fraction-free Gaussian elimination method (known as Bareiss' method) for matrix triangularisations by an arbitrary sequence of row reductions. We give exact expressions for the factors that appear systematically during the computation. Reduction graphs are used for tracing the sequence of row reductions.

We compare the growth of the integer entries of a matrix during a factor-free Gaussian elimination that performs an arbitrary sequence of row reductions and one that performs the classical sequence. We see that in the first case the entries can reach an exponential length at

intermediate steps whereas in the second case their length can grow at most linearly.

We therefore propose to use Gaussian elimination in the Gröbner basis algorithm in order to slow down the coefficient growth. The Sylvester-like matrix is "guessed" by a modular Gröbner basis computation and then it is triangularised by the classical fraction-free Gaussian elimination.

Zusammenfassung

Wir untersuchen den Zusammenhang zwischen Gröbner Basen Berechnung und Gauß'scher Elimination. Hauptziel ist der Aufbau eines Konzepts für Gröbner Basen, das jenem der Subresultanten für polynomiale Restfolgen ähnlich ist.

Determinantenpolynome werden für multivariate Polynome verallgemeinert. Mit einer Gröbner Basen Berechnung werden Sylvester-ähnliche Matrizen assoziiert, in denen die Spalten durch Terme indiziert werden und die Zeilen die Koeffizienten eines Polynoms der Ausgangsbasis, eventuell durch Multiplikation mit einem Term verschoben, enthalten.

Wir zeigen, daß die im Laufe des Buchberger Algorithmus berechneten Gröbner Basen Polynome im allgemeinen nicht durch Determinantenpolynome der oben beschriebenen Form ausgedrückt werden können. Schon in einfachen Beispielen, in denen die Ausgangsbasis aus 3 Polynomen in 2 Unbekannten besteht, ist das nicht immer möglich.

Wir betrachten dann das allgemeinere Problem der Simulation der Gröbner Basen Berechnung durch Gauß'sche Elimination. Wir entwarfen ein einfaches Werkzeug, genannt Reduktionsgraph, das die Berechnungen während des Gröbner Basen Algorithmus oder die Zeilenreduktionen einer Matrix mitverfolgt. Mit dessen Hilfe zeigen wir, daß die klassische Art der Berechnung einer Gröbner Basis nicht immer durch Gauß'sche Elimination simuliert werden kann.

Dann verallgemeinern wir das ganzzahlige Gauß'sche Eliminationsverfahren, auch als Bareiss'sches Verfahren bekannt, in der Weise, daß es Matrixtriangulierungen durch eine beliebige Folge von Zeilenreduktionen behandeln kann.

Wir vergleichen das Wachstum der Koeffizienten einer Matrix während einer ganzzahligen Gauß'schen Elimination, die beliebige

Zeilenreduktionen vornimmt, mit jener, die die klassische Folge von Reduktionen durchführt. Man sieht, daß in ersterem Fall die Einträge exponentielle Länge erreichen können wohingegen in letzterem Fall die Länge höchstens linear wachsen kann.

Wir schlagen deshalb vor, Gauß'sche Elimination im Gröbner Basen Algorithmus zu verwenden, um das Koeffizientenwachstum zu verlangsamen. Die Sylvester-ähnliche Matrix wird zuerst durch eine modulare Gröbner Basen Berechnung "vermutet" und dann durch die klassische ganzzahlige Gauß'sche Elimination trianguliert.

Contents

Chapter 0

Introduction

We are studying the connections between the Gröbner basis algorithm and Gaussian elimination. There are two main aspects we examine: first, the possibility of simulating Buchberger's algorithm by Gaussian elimination, and second, generalising the fraction free Gaussian elimination such that it can be applied to arbitrary sequences of row reductions, in particular to Gröbner basis computations.

Our initial goal was to investigate the connection between Gröbner bases and subresultants. This subject was considered in several papers and talks (see [Buc83, Laz83, Col86, Buc89, Buc91]) but is far from being completely clarified.

For linear polynomials, the computation of a Gröbner basis amounts to Gaussian elimination, hence it can be interpreted as a triangularisation of the matrix of the coefficients. When the initial coefficients are integers, we can control the growth of the matrix entries by using a fraction-free Gaussian elimination method, usually known as Bareiss' method. This method reduces a doubly exponential growth to a simple exponential one.

Computing a Gröbner basis for two univariate polynomials amounts to computing their polynomial remainder sequence. It was proven by Collins ([Col66, Col67]) and Brown ([BT71, Bro76, Bro78]) that the polynomials in the polynomial remainder sequence are similar to the subresultants of the two initial polynomials. Subresultants are defined as determinant polynomials of certain submatrices of the Sylvester matrix. The similarity coefficients can be expressed explicitly in terms of the leading coefficients of the polynomials from the polynomial remainder sequence. One method to prove these results is by showing that the pseudodivisions performed for the polynomial remainder sequence computation can be simulated by a certain sequence of row reductions in the Sylvester matrix. Using subresultants we can control the coefficient growth during a polynomial remainder sequence computation, by a mechanism similar to the fraction-free Gaussian elimination. Again, the growth is reduced from doubly exponential to simply exponential.

For two variables, it was noticed by Buchberger and Collins (see [Buc83, Col86, Buc91]) that when we start the Gröbner basis computation with two dense polynomials with indeterminate entries, all the intermediate Gröbner basis polynomials can be expressed as determinant polynomials of certain matrices containing on each row the coefficients of one of the input polynomials, shifted by multiplication with a term. Using these determinant polynomials one can also detect a certain S-polynomial which reduces to 0 although it does not satisfy the classical criteria for unnecessary S-polynomials.

All these observations led Buchberger to the conjecture that a subresultant-like construction can be devised for Gröbner bases.

From such a construction we may expect benefits similar to those provided by subresultant sequences for polynomial remainder sequences. Namely, we could obtain a method for easily detecting certain systematic constant factors of the Gröbner basis polynomials without computing the greatest common divisor of their coefficients. This would be useful because coefficients typically grow very much during a Gröbner basis computation, and keeping the polynomials primitive requires many greatest common divisor computations. We could as well get better theoretical bounds for the growth of the coefficients. There are almost no theoretical results in the literature in this direction. Also we may expect to obtain other useful properties which can not appear in the univariate case, for example new criteria for detecting unnecessary S-polynomials.

Buchberger proposed a generalisation of determinant polynomials to the multivariate case and conjectured that any Gröbner basis polynomial can be expressed as determinant polynomial of certain polynomials in the input basis multiplied by certain terms.

We show that indeed the conjecture is false and we give a counterexample. We tried then to express Gröbner basis polynomials as what we called simple nested determinant polynomials, but this approach also failed. Finally, we were led to considering the more general problem of simulating the Gröbner basis computation using Gaussian elimination in a matrix. It seemed to us that there was a general feeling in the scientific comunity that Gröbner basis computation and Gaussian elimination are closely related, and that they are even equivalent in some sense. We think that this equivalence is not completely clarified yet. Understanding it better may eventually improve the algorithms for computing Gröbner bases as well as other related algorithms.

For studying the relationship between the Gröbner basis algorithm and Gaussian elimination we devised a simple tool for tracing the computations, which we called reduction graph. Such graphs allowed us to decide, by simple graph-theoretical considerations, if a Gröbner basis computation can be simulated by row reductions in a matrix in a way similar to the way polynomial remainder sequence computation can be simulated by row reductions in the Sylvester matrix.

The conclusion was rather surprising for us: the classical Gröbner basis algorithm cannot be simulated by Gaussian elimination, unless we allow multiple occurences of some rows in the matrix. Other variants of Gröbner basis algorithms may be more appropriate for this purpose.

Another problem (closely connected to the one discussed so far) was to understand better the mechanism by which constant factors appear systematically during polynomial remainder sequence computation, in order to extend it to Gröbner basis computation. This investigation led us to generalising the fraction-free Gaussian elimination for the case when an arbitrary sequence of row reductions is performed. For this generalisation we used again reduction graphs for tracing the sequence of row reductions.

We show then how to apply this method to the Gröbner basis computation in order to detect intrinsic constant factors that appear during the computation. This is relatively easy to do when the Gröbner basis polynomials can be expressed as determinant polynomials, but it becomes rather intricate in general. Hence, these results are for the moment rather of theoretical interest than of practical one. The existence of such factors and the fact that they are leading coefficients of previously computed Gröbner basis polynomials was observed in practice and heuristically used in [GPT88]. But there was no theoretical explanation of this fact and therefore no possibility of giving the exact expression of these factors. Our results answer this question (at least partially, since we were not able to prove that our method will detect all the constant factors that appear in general).

When Gröbner basis computation can be simulated by row reductions, it is in general an arbitary sequence of row reductions, not the classical Gaussian elimination. It seems therefore worthwhile to compare the classical and the arbitrary fraction-free Gaussian elimination from the point of view of the coefficient growth. It turns out that for arbitrary fraction-free Gaussian elimination in certain cases the intermediate growth can be as bad as doubly exponential, whereas for the classical fraction-free Gaussian elimination it cannot be more than simply exponential. This suggests that in the cases where the intermediate coefficients grow much during the Gröbner basis algorithm it may be useful to use a hybrid method that combines Gröbner basis computation and classical fraction-free Gaussian elimination in order to slow down the coefficient growth.

The thesis is organised in the following way:

In the first chapter, beside recalling some definitions and basic results concerning Gröbner bases, subresultants and Gaussian elimination, we define determinant polynomials for the multivariate case. We discuss the reasons why we considered our definition to be the natural extension of univariate determinant polynomials.

In the second chapter we present first the example studied by Buchberger and Collins of the Gröbner basis computation for two dense bivariate polynomials with indeterminate coefficients. We extend slightly the class of examples having the same properties as the example above. Then we describe the attempts we made for expressing Gröbner basis polynomials using determinant polynomials or what we called simple nested determinant polynomials or, finally, using Gaussian elimination. We define reduction graphs and use them to prove that in general the classical Buchberger algorithm cannot be simulated by Gaussian elimination. Also we show that even in the bivariate case we cannot always express Gröbner basis polynomials as determinant polynomials.

In the third chapter we generalise the fraction-free Gaussian elimination method to Gaussian eliminations that perform arbitrary sequences of row reductions, possibly using more than one pivot for each column. We apply then this method to the detection of constant factors of the polynomials during Gröbner basis computations. We also make a comparison of the coefficient growth in a classical fraction-free Gaussian elimination and a generalised one and design a hybrid method for computing Gröbner bases using Gaussian elimination.

The main results in this thesis appeared in [Man94].

Chapter 1

Preliminaries

1.1 Definitions and Notation for Gröbner bases

We give here a short introduction to Gröbner bases, in order to fix some notation. For a comprehensive presentation of the subject see [BW93] or [Buc85].

Let R be an integral domain, K its quotient field and $K[x_1, \ldots, x_n]$ the ring of n-variate polynomials over K. The set T of terms over $x_1, \ldots, x_n$ (i.e. products of elements in $\{x_1, \ldots, x_n\}$) with multiplication forms a commutative monoid. We will call a set $I \subseteq T$ a monomial ideal if for any $u \in I, v \in T$ we have $u|v$ implies $v \in I$. We call $B \subset T$ a basis for I if $I = \{v \in T | \exists u \in B, u|v\}$.

By the degree of a polynomial f, denoted by $deg(f)$, we mean the total degree. The degree in any variable x_i will be denoted by $deg_{x_i}(f)$. The leading form of f is the polynomial consisting of the monomials of f of degree equal to $deg(f)$. A polynomial is called homogeneous if all its monomials have the same degree.

Definition 1 (Admissible term order)
A total order "$<$" on T is called admissible if for all terms $u, v, w \in T$ we have
1) $u < v \Rightarrow uw < vw$ (compatibility with multiplication)
2) $1 \leq u$ (1 is the minimum element)

Fix an admissible order "$<$" on T. For any polynomial $f \in K[x_1, \ldots, x_n] \setminus \{0\}$, $f = \sum_{t \in T} c_t t$ we define the support of f as $supp(f) = \{t \in T | c_t \neq 0\}$, (finite set), the leading term $lt(f) = max(supp(f))$, the leading coefficient $lc(f) = c_{lt(f)}$ and the leading monomial $lm(f) = lc(f)lt(f)$. We define $lt(0) = 0$ and $lc(0) = 0$. We can extend the relation "$<$" to $T \cup \{0\}$ by putting $0 < t$, for all $t \in T$. For a set F of polynomials, $lt(F) = \{lt(f) | f \in F\}$.

Two polynomials $f, g \in K[x_1, \ldots, x_n]$ are called similar, (denoted $f \simeq g$), if there exist non-zero constants $a, b \in K$ such that $a \cdot f = b \cdot g$. Actually we can take a and b in R.

Definition 2 (Reduction)
Let f, g be polynomials in $K[x_1, \ldots, x_n]$, having the same leading term. The reduction of g by f is defined as $red(f, g) = lc(f) \cdot g - lc(g) \cdot f$.

Remark 1 In fact any polynomial similar to $lc(f) \cdot g - lc(g) \cdot f$ could be taken as $red(f, g)$ and all the subsequent statements, except those referring to the similarity coefficients, remain true. We prefer to take $red(f, g) = lc(f) \cdot g - lc(g) \cdot f$ because then, if $f, g \in R[x_1, \ldots, x_n]$, no fractions are introduced and $red(f, g) \in R[x_1, \ldots, x_n]$.

Definition 3 (S-polynomial, Polynomial reduction)
Let $f_1, f_2 \in K[x_1, \ldots, x_n] \setminus \{0\}$.

1. The S-polynomial of f_1 and f_2 (abbreviated $Spol(f_1, f_2)$) is defined as $Spol(f_1, f_2) = red(f_2 u_2, f_1 u_1)$ where $u_i = lcm(lt(f_1), lt(f_2))/lt(f_i)$, $i = 1, 2$.

2. If $lt(f_2)$ divides a term u which appears with a non-zero coefficient c in f_1 then we say f_1 reduces by f_2 to h where $h = lc(f_2)f_1 - c \cdot (u/lt(f_2)) \cdot f_2$. We write $f \rightarrow_g h$. If u is the leading term of f_1 the reduction is called head reduction.

   We say f_1 reduces to h w.r.t. a set of polynomials F (written $f \rightarrow_F h$ or $f \rightarrow h$ if F is clear from the context) if there exists $f_2 \in F$ with the properties above. Denote by $\rightarrow^m, \rightarrow^+, \rightarrow^*$ respectively the m-th power, the transitive closure, and the reflexive and transitive closure of the relation $\rightarrow$. The similar notations are defined for $\rightarrow_f$ and $\rightarrow_F$. If $f \rightarrow_F^* h$ and h is irreducible w.r.t.

F then we say h is a normal form of f w.r.t. F. If the normal form is unique we write $h = NF(f, F)$ or $h = NF(f)$ when F is clear from the context.

The relation $\rightarrow$ is Noetherian, so a normal form always exists. If moreover $\rightarrow_F$ is confluent then the normal form is unique (up to similarity). (A relation $\rightarrow$ is called confluent if for any f, f_1, f_2 such that $f \rightarrow^* f_1, f \rightarrow^* f_2$ there exists g such that $f_1 \rightarrow^* g$ and $f_2 \rightarrow^* g$.)

Definition 4 (Gröbner bases)
A finite set $G \subset K[x_1, \ldots, x_n]$ is a Gröbner basis for the ideal I generated by G, iff it satisfies (one of) the following equivalent conditions:

1. The set $lt(G)$ generates the monomial ideal $lt(I)$.
2. The relation $\rightarrow_G$ is confluent.
3. For all $f, g \in G, Spol(f, g) \rightarrow_G^* 0$

The notion of Gröbner basis was introduced in 1965 by Bruno Buchberger in his Ph.D. thesis. An algorithm for computing such bases was also given there, based on the third characterization above. The basic form of the algorithm can be described as follows:

Algorithm Gröbner basis
  input : F finite set of non-zero polynomials
  output: G Gröbner basis for $ideal(F)$

  $G \leftarrow F$;
  $B \leftarrow \{\{f, g\} | f, g \in G, f \neq g\}$;
  while $B \neq \emptyset$
    take out a pair $\{f, g\}$ from B
    $h \leftarrow$ some normal form of $Spol(f, g)$ with respect to G
    if $h \neq 0$ **then** $B \leftarrow B \cup \{\{h, g\} | g \in G\}$;
        $G \leftarrow G \cup \{h\}$;
  end while;

The termination of the algorithm follows from the fact that the sequence of leading terms of the polynomials introduced in G is finite, which can be proven either by Dixon's lemma or, alternatively, by the noetherianity of $K[x_1, \ldots, x_n]$, applied to the sequence of ideals generated by $lt(G)$ in $K[x_1, \ldots, x_n]$.

The algorithm was improved in various ways over the years. An important improvement consisted of the criteria for detecting, a priori, certain unnecesary S-polynomials (see [Buc79]). The S-polynomials of pairs $\{f, g\}$ satisfying these criteria can be skipped without affecting the correctness of the algorithm. Actually they reduce to 0 if we choose the S-polynomials in increasing order of the least common multiple of their leading terms of the two polynomials. The main criteria are the following:

Product Criterion: $lcm(lt(f), lt(g)) = lt(f)lt(g)$

Chain Criterion: There exist $m \geq 3$ and $h_i \in G, i = 1, \ldots, m$ with $h_1 = f, h_m = g$ such that $lcm(lt(h_i), lt(h_{i+1}))$ is a proper divisor of $lcm(lt(f), lt(g))$, for $i = 1, \ldots, m-1$.

We can relax the conditions in the chain criterion by fixing a total order on pairs of polynomials having the same least common multiple of leading terms, and allowing equality between $lcm(lt(h_i), lt(h_{i+1}))$ and $lcm(lt(f), lt(g))$, in case the pair $\{h_i, h_{i+1}\}$ is smaller than $\{f, g\}$. In practice only chains with one intermediate element (i.e. $m = 3$) are searched for.

The Gröbner basis algorithm as stated above is non-deterministic. For obtaining a deterministic one we need to specify the strategies of choosing which S-polynomial to process next and which polynomials to use for a reduction (when multiple choices are possible). These strategies also may have a great influence on the efficiency of the algorithm.

When computing in a field of characteristic 0, the coefficients typically grow very much during the algorithm. Obviously, at any step of a Gröbner basis computation, we can replace a polynomial with a similar one. In practice, when R is an unique factorization domain we either compute over K and we usually keep the polynomials monic, or we compute over R and try to keep the polynomials primitive. (Notice that with our definitions, if we start with polynomials in $R[x_1, \ldots, x_n]$ the algorithm will produce only polynomials in $R[x_1, \ldots, x_n]$.) In both cases we have to do many g.c.d. computations over R if we want to keep the rational coefficients in canonical form or if we want to keep the polynomials primitive, respectively.

Convention: From now on, if not otherwise specified, we will always consider a fixed admissible order "$<$". By Gröbner basis algorithm we will mean the classical version of the algorithm, w.r.t. "$<$", using the above criteria and arbitrary but fixed strategies. We will only perform head-reductions. This does not affect the correctness of the algorithm.

1.2 Subresultants and Determinant Polynomials. Multivariate Determinant Polynomials.

Let $R[x]$ be the ring of univariate polynomials over an integral domain R. In general $R[x]$ is not Euclidean but we can perform pseudo-division of polynomials. That is, given $f, g \in R[x]$ there exist $a \in R$ and $q, r \in R[x]$ such that $af = gq + r$, and $r = 0$ or $deg(r) < deg(g)$. Moreover, q and r are unique up to similarity. We call r a pseudo-remainder of f and g. We can take $a = lc(g)^{deg(f)-deg(g)+1}$. The pseudo-remainder obtained for this value of a will be denoted by $prem(f, g)$.

Definition 5 (Polynomial remainder sequence)
Let $f_1, f_2 \in R[x]$. A sequence of polynomials $f_1, f_2, f_3, \ldots, f_r$ with the properties $f_{i+2} \simeq prem(f_i, f_{i+1})$ for $i = 1, \ldots, r-2$ and $prem(f_{r-1}, f_r) = 0$ is called a polynomial remainder sequence.

Polynomial remainder sequences are mainly used for greatest common divisor computation, real root counting and real root isolation using Sturm's Theorem.

The coefficients may grow exponentially during the computation of a polynomial remainder sequence. In order to decrease the size of the coefficients, we can divide out constant factors when we detect them. One possibility is to compute the content of each polynomial, and divide it out. But this may require much computational effort in the computation of greatest common divisors over R. An ingenious solution was proposed by Collins and Brown (see [Col66, Col67, BT71, Bro76, Bro78]). It is noticed that certain factors appear systematically in the computation, and they can be detected a priori with little effort. The origin of these factors can be better explained by the fundamental theorem of polynomial remainder sequences, which we will present now. For a more detailed presentation as well as for the history of these results see the overview paper [Loo83].

We associate to any finite sequence of polynomials $f_i = \sum_{j=0}^{m_i} a_{ij} x^j \in R[x], i = 1, \ldots, k$ the matrix

$$mat(f_1, \ldots, f_k) = \begin{pmatrix} a_{1,m} & \ldots & a_{1,0} \\ \vdots & & \\ a_{k,m} & \ldots & a_{k,0} \end{pmatrix}$$

where $m = \max(m_i)$ and $a_{ij} = 0$ for $j > m_i$.

Definition 6 (Determinant polynomial) The determinant polynomial of a matrix $M = (a_{i,j})_{\substack{i=1,\ldots,k \\ j=1,\ldots,m}}$ with $k \leq m$ is defined as: $detpol(M) = \sum_{j=k}^{m} d_j x^{m-j}$ where

$$d_j = \begin{vmatrix} a_{1,1} & \ldots & a_{1,k-1} & a_{1,j} \\ \vdots & & & \\ a_{k,1} & \ldots & a_{1,k-1} & a_{k,j} \end{vmatrix}$$

Definition 7 (Subresultants) Let $f, g \in R[x]$, $deg(f) = m, deg(g) = n$. The polynomial $S_k(f,g) = detpol(x^{n-k-1} f, \ldots, f, x^{m-k-1} g, \ldots, g)$, $0 \leq k < \min(m,n)$ is called the k-th subresultant of f and g.

Theorem 1 (Fundamental Theorem of polynomial remainder sequences)
Let $f_1, \ldots, f_r$ be a polynomial remainder sequence and $deg(f_i) = n_i$, $n_1 \geq n_2$. Then, for $j = 3, \ldots, r$ we have $f_j \simeq S_{n_{j-1}-1}(f_1, f_2) \simeq S_{n_j}(f_1, f_2)$ and $S_i(f_1, f_2) = 0$ for all $n_{j-1} - 1 < i < n_j$. The coefficients of similarity are given by the formulas below. Assume $e_i f_i = q_i f_{i+1} + d_i f_{i+2}$, for $i = 1, \ldots, r-2$, $e_i, d_i \in R$ and let $c_i = lc(f_i)$. Then, for $j = 2, \ldots, r-1$

$$\left(\prod_{i=1}^{j-1} e_i^{n_{i+1} - n_j + 1} \right) S_{n_j - 1} =$$

$$= \left(\prod_{i=1}^{j-1}(-1)^{(n_i-n_j+1)(n_{i+1}-n_j+1)} d_i^{n_{i+1}-n_j+1} c_{i+1}^{n_i-n_{i+2}}\right) c_j^{-n_j+n_{j+1}+1} f_{j+1}$$

$$\left(\prod_{i=1}^{j-1} e_i^{n_{i+1}-n_{j+1}}\right) S_{n_{j+1}} =$$

$$= \left(\prod_{i=1}^{j-1}(-1)^{(n_i-n_{j+1})(n_{i+1}-n_{j+1})} d_i^{n_{i+1}-n_{j+1}} c_{i+1}^{n_i-n_{i+2}}\right) c_{j+1}^{n_j-n_{j+1}-1} f_{j+1}$$

We will give here one variant of proof (essentially the proof from [Col66, Col67, Loo83]) which illustrates the technique we have in mind for the generalization of subresultants. The idea of the proof is to take the matrix that defines a subresultant and perform on it a certain Gaussian elimination which simulates the pseudo-division of polynomials in the polynomial remainder sequence; thereby in this matrix we obtain successively the polynomials of the polynomial remainder sequence. In the end the matrix becomes triangular and we can easily compute its determinant polynomial, which will be similar both to the subresultant and to a certain polynomial in the polynomial remainder sequence.

Proof Let $M_k = mat(x^{n_2-k-1}f_1, \ldots, f_1, x^{n_1-k-1}f_2, \ldots, f_2)$. For convenience we will think of this matrix as having two parts: the first part consists of the rows which have as entries the coefficients of f_1, the second consists of the rows which correspond to f_2.

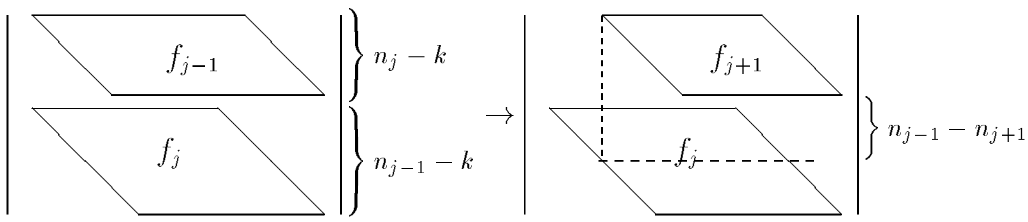
We get zeros in the the first row of the first part in the columns $1, \ldots, n_1 - n_2 + 1$ by elimination using the rows $1, \ldots, n_1 - n_2 + 1$ from the second part. We can do these eliminations exactly the way the pseudo-division of f_1 by f_2 is performed. In this way we multiply the determinant polynomial of the matrix by e_1 and we obtain the coefficients of $d_1 f_3 x^{n_2-k-1}$ in the first row, so we can divide by d_1. We continue for rows $2, \ldots, n_2 - k$ such that in row i of the first part we obtain zeros in columns $i, \ldots, i + n_1 - n_2$ by row reductions using the rows $i, \ldots, i + n_1 - n_2$ of the second part. Let $\overline{M}_k$ be the matrix thus obtained. If we interchange the two parts, the first $n_1 - n_3$ columns are in triangular form, so we get:

$$detpol(M_k)(e_1/d_1)^{n_2-k} = detpol(\overline{M}_k) = (-1)^{(n_1-k)(n_2-k)} c_2^{n_1-n_3} detpol(M_k^{(3)})$$

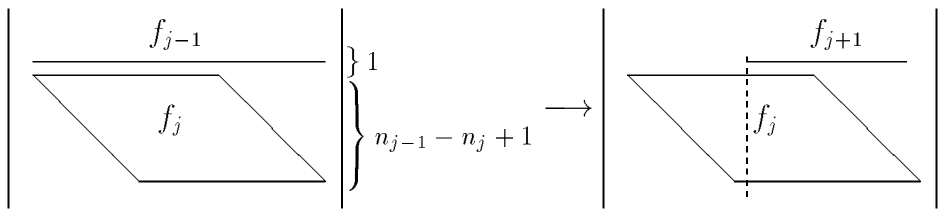
where $M_k^{(3)}$ is the matrix obtained from $\overline{M}_k$ after interchanging the two parts of the matrix and then deleting the first $n_1 - n_3$ rows and columns. Notice that $M_k^{(3)}$ is exactly the matrix that gives the k-th subresultant of f_2 and f_3. We apply recursively the same transformations, obtaining $\overline{M}_k^{(3)}, M_k^{(4)}, \overline{M}_k^{(4)}, \ldots, M_k^{(j)}$. In general we have

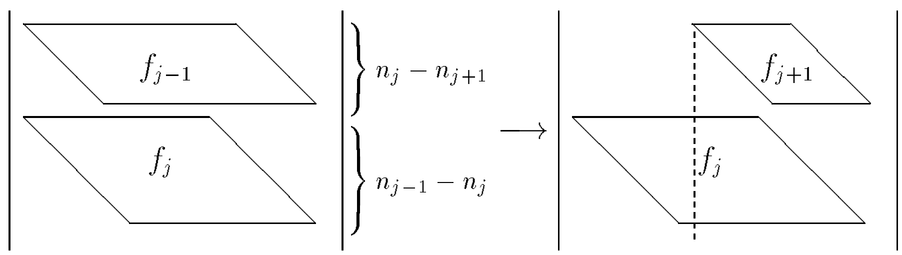
$$detpol(M_k^{(j)}) \prod_{i=1}^{j-2} (-1)^{(n_i-k)(n_{i+1}-k)} c_{i+1}^{n_i - n_{i+2}} d_i^{n_{i+1}-k} = \prod_{i=1}^{j-2} e_i^{n_{i+1}-k} detpol(M_k) \tag{1.1}$$

The matrix $M_k^{(j)}$ contains $n_j - k$ rows with the coefficients of f_{j-1} as entries, and $n_{j-1} - k$ rows with f_j. The shape of the matrices $M_k^{(j)}$ and $\overline{M}_k^{(j)}$ is shown in the next figure:



The interesting cases are when $\overline{M}_k^{(j)}$ becomes triangular. This happens for $k = n_j - 1$ and for $k = n_{j+1}$, as shown in the next pictures.





In the first case,

$$detpol(M^{(j)}_{n_j-1})e_{j-1}/d_{j-1} = detpol(\overline{M}^{(j)}_{n_j-1}) =$$
$$(-1)^{(n_{j-1}-n_j+1)}c_j^{n_{j-1}-n_j+1}f_{j+1}$$

In the second case,

$$detpol(M^{(j)}_{n_{j+1}})(e_{j-1}/d_{j-1})^{n_j-n_{j+1}} = detpol(\overline{M}^{(j)}_{n_{j+1}}) =$$
$$(-1)^{(n_{j-1}-n_{j+1})(n_j-n_{j+1})}c_j^{n_{j-1}-n_{j+1}}c_{j+1}^{n_j-n_{j+1}-1}f_{j+1}$$

For $k = n_j - 2, \ldots, n_{j+1}+1$ we have $detpol(\overline{M}^{(j)}_k) = 0$. By putting in (1.1) in the first case $k = n_j - 1$ and in the second case $k = n_{j+1}$ and replacing $detpol(M^{(j)}_k)$ by the expressions we got above, we obtain the identities in the theorem. □

We generalize now the definition of the determinant polynomial to multivariate polynomials. First we identify polynomials with vectors (rows of matrices). For multivariate polynomials it is more natural to consider these vectors indexed by terms rather than by integers. Let "$<$" be an admissible order on T. Finite subsets of T are totally ordered and will be used as indices for vectors and as indices for the columns of matrices. Given a polynomial $f = \sum_{t\in T} c_t t \in R[x_1, \ldots, x_n]$, for any finite set $A \subset T$ with $supp(f) \subseteq A$ we associate to f the vector $v_A(f) = (c_t)_{t\in A} \in R^A$. Conversely, to any vector $c = (c_t)_{t\in A} \in R^A$ we associate the polynomial $pol(c) = \sum_{t\in A} c_t t$. We define the leading term and the leading coefficient of a vector c as $lt(c) := lt(pol(c))$ and $lc(c) := c_{lt(c)}$. For finite sets $A, B \subset T$ with $A \subseteq B$, a vector $c \in R^A$ can be canonically extended to a vector in R^B by defining $c_t = 0$ for all $t \in B - A$ (natural embedding of R^A in R^B). When A is clear from the context, we will use interchangeably f and $v_A(f)$, as well as c and $pol(c)$.

For any sequence of polynomials $f_1, \ldots, f_r$ we define the matrix

$$mat(f_1, \ldots, f_r) = \begin{pmatrix} v_A(f_1) \\ \ldots \\ v_A(f_r) \end{pmatrix}$$

where $A = \cup_{i=1}^r supp(f_i)$.

Definition 8 (Row reduction, triangular matrix)

Let M be a matrix over R, and let c, d be two distinct non-zero rows of M such that $lt(c) = lt(d)$. We say that we reduce the row c by the row d in the matrix M if we replace the row c by the row $red(d,c) = lt(d)c - lt(c)d$.

We call M triangular iff each of its non-zero rows has a different leading term.

Notice that if we repeatedly perform row reductions (in any order) on a matrix M, the process will finally stop, producing a triangular matrix.

Definition 9 (Determinant polynomial)

Let A be a finite subset of T and $M = (c_{i,j})_{i \in \{1,\ldots,r\}, j \in A}$ a matrix over R. If $rank(M) = k$, we define the determinant polynomial of M as: $detpol(M) = \sum_{t \in A} a_t t$,where

$$a_t = \begin{vmatrix} c_{1,m_1} & \ldots & c_{1,m_{r-1}} & c_{1,t} \\ \vdots & & & \\ c_{r,m_1} & \ldots & c_{r,m_{r-1}} & c_{r,t} \end{vmatrix}$$

and $(m_1, \ldots, m_{r-1})$ is lexicographically the maximum among all tuples $(i_1, \ldots, i_{r-1}) \in A^{r-1}$ for which the columns indexed by $i_1, \ldots, i_{r-1}$ are linearly independent. If $rank(M) < r$ we define $detpol(M) = 0$.

Remark 2

1. The determinant polynomial is invariant, up to similarity, under row reductions. Its leading term is m_r where $(m_1, \ldots, m_r)$ is lexicographically the maximum tuple such that the columns $m_1, \ldots, m_r$ are linearly independent, that is, the minor consisting of these columns is not 0.

2. Our definition of determinant polynomial is a particular case of the one proposed by Buchberger ([Buc91]). He defines the determinant polynomial associated to r polynomials and $r-1$ terms $u_1, \ldots, u_{r-1}$. These terms denote the positions of the columns to be fixed in the determinants that define the coefficients a_t of the determinant polynomial. For the reasons below, we decided that we would not use an arbitrary tuple of terms $u_1, \ldots, u_{r-1}$, but exactly the tuple $(m_1, \ldots, m_{r-1})$ we chose above. Besides, this tuple $(m_1, \ldots, m_{r-1})$ is not given "from outside" but is intrinsic to the matrix.

If we choose as fixed columns $(m_1, \ldots, m_{r-1})$ the $r-1$ highest indices, as in the univariate case, we can encounter (both in the univariate and in the multivariate case) the situation when these columns are linearly dependent, which would make the determinant polynomial a priori zero. This is the case with the zero elements (the gaps) in the subresultant sequence. Whereas in the univariate case they can be managed separately, in the multivariate case these situations can become more intricate and we decided it would be more convenient to avoid them from the beginning. So we will choose linear independent columns.

The reason why we choose $(m_1, \ldots, m_{r-1})$ as above becomes clear if we first triangularize the matrix by row reductions. Notice that triangularization does not change the determinant polynomial (up to similarity), no matter what columns we may decide to fix. Now, for a triangular matrix, it seems to us most natural to fix the columns that correspond to the $r-1$ highest leading terms of rows. The positions of these rows can be expressed w.r.t. the initial matrix as being the lexicographically highest tuple of linear independent columns (To prove this we use the fact that the linear dependence/independence of columns is preserved by triangularization). Notice that in a triangular matrix the row of lowest leading term is similar to the determinant polynomial of the matrix.

1.3 Gaussian elimination

Gaussian elimination transforms a non-singular square matrix over a field to an equivalent triangular matrix (i.e. having the same determinant), by elementary row operations. We can easily extend this method to $k \times l$ matrices $M = (m_{i,j})$, with $k \leq m$, having entries from an integral domain R. Equivalent will mean in this case having similar determinant polynomials, or in case R is a field, equal determinant polynomials.

Algorithm Gaussian elimination

$p \leftarrow 0;$

for $i = 1, \ldots, k-1$ **do**
    if $m_{r,q} = 0$ for all $r \in \{i, \ldots, k\}, q \in \{p+1, \ldots, l\}$ **then return**
    let q be the minimum index greater than p for which there exists
        a row $r \in \{i, \ldots, k\}$ with $m_{r,q} \neq 0$
    interchange rows r and i.
    for $j = i+1, \ldots, k$
        $m_j \leftarrow m_{i,q} m_j - m_{j,q} m_i$
        (or, in case R is a field, do instead $m_j \leftarrow m_j - (m_{j,q}/m_{i,q}) m_i$)
    $p \leftarrow q$;
end for
return

We denoted by m_j the j-th row of the matrix M. The element $m_{r,q} \neq 0$ is called a pivot. Notice that at each step of the outer **for** loop one pivot is chosen.

For matrices over integral domains, it was noticed that at the end of each step $i > 1$ we can perform an exact division of the rows $i+1, \ldots, k$ by the pivot of the previous step, $m_{i-1,p}$. In this case, the row k of the final matrix will be equal to the determinant polynomial (respectively the determinant, if $k = l$) of the initial matrix M.

This method, called fraction-free Gaussian elimination, was discovered and rediscovered several times. It is usually referred to as Bareiss' method ([Bar68]), although it seems it was known earlier to Jordan and to Dodgson ([Dod66]).

We will call *classical* (fraction-free) Gaussian elimination the methods above, which choose one pivot for obtaining zeros on one column.

But, as mentioned in the previous chapter, we can perform other sequences of row reductions that lead as well to a triangular matrix. We will call such an algorithm arbitrary (or non-classical) Gaussian elimination. Notice that there can be columns where more than one pivot is used.

Algorithm Non-classical Gaussian elimination
    while there exist non-zero rows having the same leading term
        choose two non-zero rows i and j having the same leading term
        reduce row j by row i
    end while

Chapter 2

Gröbner Bases and Gaussian Elimination

2.1 Problem Statement and Motivation

Subresultants are a useful tool in the computation of polynomial remainder sequences. They provide formulae for detecting systematic common factors of the coefficients without computing their greatest common divisor. Dividing out these factors improves the efficiency of the algorithm in practice and also leads to better theoretical complexity bounds on the growth of the coefficients.

We know that Gröbner basis computation specializes in the univariate case to polynomial remainder sequence computation. It seems therefore reasonable to expect that the subresultant theory can be generalized to Gröbner bases in such a way that we obtain similar benefits.

Such results would be useful. Many Gröbner basis computations by computer nowadays cannot finish because of the big size of the coefficients of intermediate polynomials. Also, there is basically no theoretical result concerning the growth of the coefficients during Gröbner bases algorithm. Of course, we might also expect to obtain other properties which can not appear in the univariate case, for example new criteria for detecting unnecessary S-polynomials.

Let us now examine in more detail the way in which we expect to generalize subresultants for Gröbner bases.

In the case of a polynomial remainder sequence any polynomial in the sequence is similar to the determinant polynomial of the matrix associated with the two initial polynomials and some of their multiples by terms. This similarity can be explained by the fact that we can simulate the sequence of pseudo-divisions by row reductions in that matrix.

For the Gröbner basis algorithm, we would like each intermediate polynomial f that is adjoined to the basis to be similar to a polynomial obtained from a matrix M associated with some of the polynomials in the input basis and some of their multiples by terms. We expect these polynomials to be those multiples of the polynomials in the input basis that are used for obtaining f. We will call these polynomials the prerequisites of f. We will give a more precise definition.

Convention Throughout this chapter, F will denote the input basis of a fixed Gröbner basis algorithm, and f will denote a Gröbner basis polynomial adjoined to the basis during the algorithm.

Definition 10 (Prerequisites of a polynomial)

The set $prereq(f) \subseteq TF$, of prerequisites of f in F (w.r.t. the given algorithm) is defined in the following way:

If $f \in F$ then $prereq(f) = \{f\}$. If f is obtained as $Spol(f_1, f_2) = g_0 \rightarrow_{h_1} g_1 \rightarrow_{h_2} \ldots \rightarrow_{h_k} g_k = f$ and $Spol(f_1, f_2) = red(v_2 f_2, v_1 f_1)$, $g_i = red(u_i h_i, g_{i-1})$ with $v_i, u_i \in T$ then $prereq(f) = v_1 \cdot prereq(f_1) \cup v_2 \cdot prereq(f_2) \cup \cup_{i=1}^{k} u_i \cdot prereq(h_i)$.

In the univariate case, for a polynomial f_i in a polynomial remainder sequence $f_1, f_2, \ldots f_r$, , the matrix $mat(prereq(f_i))$ we obtain by the definition above has as rows a (possibly proper) subset of the rows of the matrix that defines the corresponding subresultant $S_{deg(f_{i-1})-1}(f_1, f_2)$. (The subset may be proper for sparse polynomials.) The results concerning the systematic constant factors that appear during polynomial remainder sequence computation can be proven using these matrices as well, that is, we have $f_i \simeq detpol(prereq(f_i))$ and we can give exact formulas for the similarity coefficient. The content of $detpol(prereq(f_i))$ is a (possibly proper) divisor of the content of $S_{deg(f_{i-1})-1}(f_1, f_2)$. The polynomial remainder sequence given by $(detpol(prereq(f_i)))_{i=1,\ldots,r}$ may have therefore smaller coefficients than the subresultant polynomial remainder sequence, but we have the overhead of computing $prereq(f)$.

On the other hand, in general even if f_1, f_2 are sparse, the polynomial remainder sequence will soon contain dense polynomials and any advantage is lost. It is therefore better to use the matrices that define the subresultants, which provide a good and easy-to-compute approximation of $mat(prereq(f))$.

Example 1 Compute the polynomial remainder sequence of $f_1 = ax^{12} + bx^9 + cx^3 + dx^2 + e, f_2 = ux^9 + vx^6 + w$, polynomials is x. For dividing f_1 we use $x^3 f_2$ and f_2 and we get f_3 with $lt(f_3) = x^6$, so $prereq(f_3) = \{f_1, x^3 f_2, f_2\}$. We divide f_2 using $x^3 f_3$ and f_3 and we get f_4 and $prereq(f_4) = \{f_2\} \cup x^3\ prereq(f_3) \cup prereq(f_3) = \{x^3 f_1, f_1, x^6 f_2, x^3 f_2, f_2\}$. By direct computation we see that $detpol(prereq(f_4))$ is primitive, whereas the corresponding subresultant $S_5(f_1, f_2)$ has a content of $lc(f_3)^2 = (v(av - bu))^2$.

For generalizing subresultants for Gröbner bases it seems natural to try to obtain the polynomial f from $mat(prereq(f))$ as determinant polynomial. If this turns out not to be possible, we could try at least to simulate by row reductions the computations done during the Gröbner basis algorithm. This would also clarify the relation between Buchberger's Gröbner basis algorithm and Gaussian elimination.

In this chapter, after studying the particular case of two dense bivariate polynomials as input, we will describe three attempts that we made for achieving the goal above. After briefly presenting the first two we will elaborate more on the third, since this one led us to a construction (which we call a reduction graph) that throws a new light on the first two and seems to provide a useful tool for further research.

2.2 Gröbner Basis Computation in the Bivariate Case

For the beginning, we study the simplest non-trivial Gröbner basis computation, namely the one in 2 variables. We will especially focus on the similarities with subresultants.

We consider the case when the input basis consists of 2 dense polynomials with indeterminate coefficients. This example was examined by Buchberger and Collins in various talks ([Buc83, Col86, Buc91]) and it possesses several remarkable properties:

- All the Gröbner basis polynomials can be expressed as determinant polynomials of multiples of the input polynomials.
- Constant factors appear systematically in the polynomials.
- The last S-polynomial (whose reduction is by far the most time consuming) reduces to 0 although this cannot be detected by the classical product or chain criterion.

The reduction to 0 of the last S-polynomial can be detected by the syzygy criterion (described in [MMT92]), in the variant in which we interreduce all the syzygies. As for principal syzygies, we only need to use the principal syzygy of the input polynomials. Performing all interreductions becomes quite complex as the computation goes on, and it is not clear if in the end it will not become more costly than the S-polynomial computation we want to avoid.

The properties enumerated above motivate a subresultant-like approach to Gröbner basis computation. Since (to our knowledge) they never appeared in print, we thought it worthwhile to give here a more detailed analysis.

Theorem 2 Let f_1, f_2 be two bivariate polynomials with indeterminate coefficients. Let $lt(f_i) = x^{a_i}y^{b_i}, i = 1, 2$ and assume $a_1 > a_2$ and $b_1 < b_2$. Then, if $b_1 > 0$ the computation of a Gröbner basis of $\{f_1, f_2\}$ w.r.t. degree lexicographical order, $(x > y)$, proceeds as follows:

1. $Spol(f_1, f_2) \longrightarrow_{f_2}^{a1-a2} f_3$ and $lt(f_3) = x^{a_2-1}y^{b_2+a_1-a_2+1}$
   $Spol(f_{i+1}, f_{i+2}) \longrightarrow_{f_{i+2}} f_{i+3}$ and $lt(f_{i+3}) = x^{a_2-i-1}y^{b_2+a_1-a_2+2i+1}$, for $i = 1, \ldots, a_2 - 1$
   $Spol(f_{a_2+1}, f_{a_2+2}) \longrightarrow_{f_{a_2+2}} f_{a_2+3}$ and $lt(f_{a_2+3}) = x^{b_2+a_1+a_2-1}$.

   During this part of the algorithm, we always have only one S-polynomial that can be computed, and only one polynomial from the basis that can be used for the reduction. Therefore the flow of the algorithm does not depend on any choice.

2. $f_{i+3} \simeq detpol(f_1 x^i y^{b_2-b_1}, \ldots, f_1 y^{b_2-b_1+i}, \quad f_2 x^{a_1-a_2+i}, \ldots, f_2 y^{a_1-a_2+i})$, $i = 0, \ldots, a_2$

3. Any reduction of $Spol(f_{a_2+3}, f_1)$ leads to 0. The Gröbner basis of $\{f_1, f_2\}$ is $\{f_1, f_2, f_3, \ldots, f_{a_2+3}\}$

4. Constant factors appear as follows: f_4 is divisible by $lt(f_2)^{a_1-a_1+1}$. Put $g_4 = f_4/lt(f_2)^{a_1-a_1+1}$, $g_i = f_i$ for $i = 1, 2, 3$ and denote $lc(g_i)$ by c_i. Then, inductively for $i = 5, \ldots, a_2+3$, $NF(Spol(g_{i-2}, g_{i-1}))$ is divisible by c_{i-2}^2 and we put $g_i = NF(g_{i-2}, g_{i-1})/c_{i-2}^2$.

For the case $b_1 = 0$ all the statements above are true except that we have already $Spol(f_{a_2+1}, f_{a_2+2}) \longrightarrow^* 0$, so f_{a_2+3} does not appear any more and the Gröbner basis is $\{f_1, f_2, f_3, \ldots, f_{a_2+2}\}$

Figure 2.1 shows the leading terms of the polynomials in a Gröbner basis computation like the one above.

For proving the theorem we use some simple facts collected in the following lemma.

Lemma 1 Denote by $(R[x_0, x_1, \ldots, x_n])_d$ the set of homogeneous polynomials of degree d in $x_0, x_1, \ldots, x_n$, and by $(R[x_1, \ldots, x_n])_{\leq d}$ the set of polynomials of degree at most d in $x_1, \ldots, x_n$.

1. The homogenisation and dehomogenisation functions defined as:
   $dehom : (R[x_0, x_1, \ldots, x_n])_d \to (R[x_1, \ldots, x_n])_{\leq d}$,
   $dehom(p) = p(1, x_1, \ldots, x_n)$
   $hom_d : (R[x_1, \ldots, x_n])_{\leq d} \to (R[x_0, x_1, \ldots, x_n])_d$,
   $hom_d(\overline{p}) = x_0^d \overline{p}(x_1/x_0, \ldots, x_n/x_0)$
   are bijective and are the inverse of each other.

2. Let "$<$" be an admissible order on the monoid of terms in $x_1, \ldots, x_n$, and define an order $<'$ on the terms in $x_0, x_1, \ldots, x_n$, which compares two terms by comparing first the degree and in case of equality compares w.r.t. $<$ their dehomogenisation. Then $<'$ is an admissible order and lt commutes with hom_d and $dehom$.

3. The function red commutes with hom_d, $dehom$ and $form_d$, where $form_d$ is defined on $R[x_0, x_1, \ldots, x_n]$ and returns the homogeneous part of degree d of a polynomial.

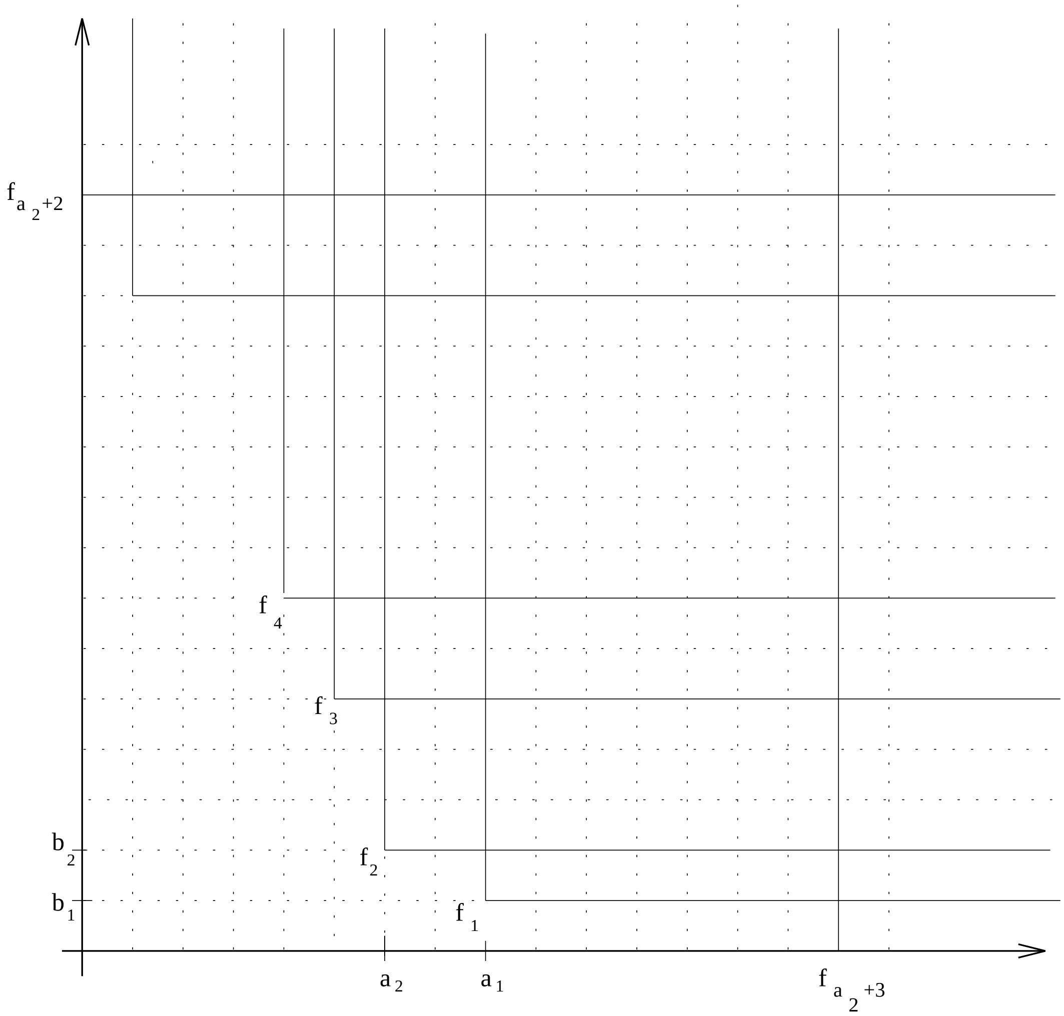
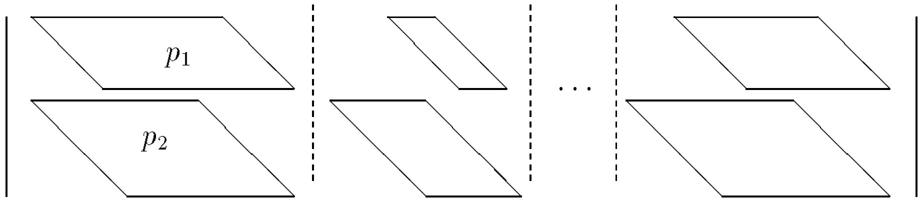


Figure 2.1: Gröbner basis computation for two dense bivariate polynomials

Proof of the theorem. Consider the matrix M_i whose lines consist of $f_1x^iy^{b_2-b_1}, f_1x^{i-1}y^{b_2-b_1+1}, \ldots, f_1y^{b_2-b_1+i}, f_2x^{a_1-a_2+i}, f_2x^{a_1-a_2+i-1}y, \ldots,$ and $f_2y^{a_1-a_2+i}$). All the polynomials in this matrix have the same degree, namely $d = a_1 + b_2 + i$. The matrix has the following shape:



where the parallelograms represent the regions with non-zero entries, and the vertical dashed lines separate the blocks of columns of different degrees.

Notice that the submatrix consisting of columns corresponding to terms of maximal degree has exactly the shape of the matrix defining a subresultant. The idea of the proof is to apply to M_i the same kind of Gaussian elimination we used in the proof of the fundamental theorem of subresultants, obtaining successively on the rows of the matrix the coefficients of $f_3, f_4, \ldots, f_{i+3}$. We will get thus results analogous to those in the fundamental theorem of subresultants, which will prove the points 1,2 and 4 of the theorem.

We apply the lemma above for $(R[x,y])_d$ and $(R[x])_{\leq d}$, where $d = a_1 + b_2 + i$, and y is the variable of homogenisation. The unique admissible order on the monoid of terms in x induces on the monoid of terms in x, y the degree lexicographical order with $x > y$, i.e. the order used in the theorem.

Denote by $lform(f)$ the leading form of a polynomial f. Let $p_i = lform(f_i)$ for $i = 1, 2$ and let $\overline{p}_i$ be the dehomogenisation of p_i. If in the submatrix of M_i consisting of the columns of highest degree we dehomogenise the column indices we get the matrix: $mat(dehom(form_d(f_1y^{b_2-b_1}x^i)), \ldots,$ $dehom(form_d(f_1y^{b_2-b_1}y^i)),$ $dehom(form_d(f_2x^{a_1-a_2+i})), \ldots, dehom(form_d(f_2y^{a_1-a_2+i})))$, which, by abuse of notation we will denote by $dehom(form_d(M_i))$. This matrix coincides with the matrix $\overline{M_i} = mat(\overline{p}_1x^i, \ldots, \overline{p}_1, \overline{p}_2x^{a_1-a_2+i}, \ldots, \overline{p}_2)$ defining the $(a_2 - i - 1)$-th subresultant of $\overline{p}_1$ and $\overline{p}_2$.

From the relation $dehom(form_d(M_i)) = \overline{M_i}$ using the second point of the lemma we see that the j-row of M_i and the j-th row of $\overline{M_i}$ have the same leading coefficient and their leading terms are the image of each other by dehomogenisation, respectively homogenisation. Denote by M_i' and by $\overline{M_i}'$ the matrix obtained by reducing the j-th row by the k-th row in M_i and in $\overline{M_i}$, respectively. Then, due to the commutativity of red and $dehom$ we again have the relationship $dehom(form_d(M_i')) = \overline{M_i}'$. The determinant polynomials of the two matrices M_i and $\overline{M_i}$ are modified during row reduction in the same way, namely they are multiplied by the leading coefficient of the k-th row. Thus, if we perform a sequence of row reductions on one of the matrices M_i or $\overline{M_i}$, we can perform the same sequence of reductions on the other matrix, and the determinant polynomials are multiplied by the same quantities.

We will perform on both $\overline{M_i}$ and M_i the sequence of reductions used in the proof of the fundamental theorem of subresultants. On the rows of $\overline{M_i}$ we obtain successively the coefficients of the polynomial remainder sequence $\overline{p}_1, \overline{p}_2, \overline{p}_3, \ldots, \overline{p}_{i+3}$, where $\overline{p}_{j+2} = prem(\overline{p}_j, \overline{p}_{j+1})$. The degrees of the polynomials are $deg(\overline{p}_1) = a_1$, $deg(\overline{p}_2) = a_2$, and due to the indeterminate coefficients of $\overline{p}_1$ and $\overline{p}_2$, the degree will drop by 1 at each pseudo-remainder computation, i.e. $deg(\overline{p}_{j+2}) = a_2 - j$.

We prove by induction that in the rows of the matrix M_i we obtain successively the coefficients of $f_3, \ldots, f_{i+3}$ multiplied by certain terms, and that $\overline{p_{j+3}} = dehom(lform(f_{j+3}))$ and $lt(f_{j+3}) = x^{a_2-j-1}y^{b_2+a_1-a_2+2j+1}$ for $j = 0, \ldots, i$. When we obtain $\overline{p}_3 = red(\ldots red(red(\overline{p}_1, x^{a_1-a_2}\overline{p}_2), x^{a_1-a_2-1}\overline{p}_2)\ldots, \overline{p}_2)$ in a row of $\overline{M_i}$, the third point of the lemma implies that in the corresponding row of M_i we obtain $red(\ldots red(red(f_1 y^{b_2-b_1}y^i,\ f_2 x^{a_1-a_2}y^i),\ f_2 x^{a_1-a_2-1}y^{i+1})\ldots,$ $f_2 y^{a_1-a_2+i}) = f_3 y^i$. Hence, when the first rows of M_i become $x^i\overline{p}_3, x^{i-1}\overline{p}_3, \ldots, \overline{p}_3$, the first rows of $\overline{M_i}$ become $x^i f_3, x^{i-1}yf_3, \ldots, y^i f_3$. Since the relation $dehom(form_d(M_i)) = \overline{M}_i$ is preserved, we have in particular $dehom(lform(f_3)) = \overline{p}_3$. The leading term of f_3 has the same degree in x as $\overline{p}_3$ and since its total degree is $a_1 + b_2$ we deduce $lt(f_3) = x^{a_2-1}y^{b_2+a_1-a_2+1}$. Similarly, for $j \geq 1$, when we obtain $\overline{p}_{j+3} = red(red(\overline{p}_{j+1}, x\overline{p}_{j+2}), \overline{p}_{j+2})$ in a row of $\overline{M_i}$, by the induction hypothesis and using the third point of the lemma we see that on the corresponding row of M_i we obtain $red(red(f_{i+1}y^2y^{i-j}, f_{j+2}xy^{i-j}), f_{j+2}y^{i-j+1}) = f_{j+3}y^{i-j}$. Since the property $dehom(form_d(M_i)) = \overline{M_i}$ is preserved by row reduction, we have $dehom(lform(f_{j+3})) = \overline{p}_{j+3}$ hence the degree in

x of f_{j+3} is the same as the degree of $\overline{p}_{j+3}$. But from the leading terms of f_{j+1} and f_{j+2} given by the induction hypothesis we deduce that the total degree of f_{j+3} is a_1+b_2+j, hence $lt(f_{j+3}) = x^{a_2-j-1}y^{b_2+a_1-a_2+2j+1}$.

From the fundamental theorem of subresultants we get $detpol(\overline{M_i}) = S_{a_2-i-1}(\overline{p}_1, \overline{p}_2) \simeq \overline{p}_{i+3}$, and the similarity coefficient is given by:

$$\left((c_2^{a_1-a_2+1})^{i+1}\prod_{l=2}^{i+1}(c_{l+1}^2)^{i-l+2}\right) S_{a_2-i-1} = \left(c_2^{a_1-a_2+1}\prod_{i=2}^{i+1}c_{l+1}^2\right)\overline{p}_{i+3}$$

which amounts to:

$$\left((c_2^{a_1-a_2+1})^{i}\prod_{l=2}^{i+1}(c_{l+1}^2)^{i-l+1}\right) S_{a_2-i-1} = \overline{p}_{i+3}$$

The same relations will then hold in M_i, i.e. $detpol(M_i) \simeq f_{i+3}$ and the similarity coefficient is given by:

$$\left((c_2^{a_1-a_2+1})^{i}\prod_{l=2}^{i+1}(c_{l+1}^2)^{i-l+1}\right) detpol(M_i) = f_{i+3}$$

The points 1,2 and 4 of the theorem result now easily for $i = 1, \ldots, a_2-1$. For $i = a_2$ we still have to prove that $lt(f_{a_2+3}) = x^{b_2+a_1+a_2-1}$. Indeed, from the leading terms of f_{a_2+1} and f_{a_2+2} we deduce that the degree of f_{a_2+3} is $b_2 + a_1 + a_2 - 1$. To compute the coefficient of $x^{b_2+a_1+a_2-1}$ in f_{a_2+3} we trace back the computation of f_{a_2+3} up to the beginning of the algorithm, that is up to the computation of $Spol(f_1, f_2)$. At each S-polynomial computation or polynomial reduction during the algorithm, one of the polynomials involved is multiplied by a positive power of y, the other is multiplied by 1 or by a power of x. Hence if the second polynomial has a non-zero coefficient for some pure power of x, then this coefficient cannot be canceled accidentally and the corresponding power of x in the result also has non-zero coefficient. The coefficient of $x^{b_2+a_1+a_2-1}$ in f_{a_2+3} stems from the coefficient of $x^{a_2+b_2-1}$ in f_2, which is non-zero and, as we saw, could not be accidentally canceled during the computation. Hence $lt(f_{a_2+3}) = x^{b_2+a_1+a_2-1}$.

For proving point 3 we need first some simple lemmas:

Lemma 2 If $h = detpol(h_1, \ldots, h_m)$ and there exists a linear combination of $h_1, \ldots, h_m$, $\overline{h} = \sum_{i=1}^m a_i h_i$, such that $lt(\overline{h}) \leq lt(h)$, then $h \simeq \overline{h}$.

Proof. Let $M = mat(h_1, \ldots, h_m)$ and let M' be a matrix obtained from M by replacing by $\overline{h}$ a row h_i for which $a_i \neq 0$. Since $\overline{h}$ is a linear combination of the rows fo M, the determinant polynomial of M' is similar to the one of M. We triangularize now M', obtaining a matrix M''. The row g having the smallest leading term in M'' will have the leading term lower or equal to any leading term of a row of M', in particular $lt(g) \leq lt(\overline{h}) \leq lt(h)$. But, on the other hand g is similar to $detpol(M'')$ and therefore to h, which implies $lt(g) = lt(\overline{h}) = lt(h)$. This means that the row $\overline{h}$ was not modified during triangularization, otherwise its leading term in M'' would be strictly smaller than $lt(g)$, in contradiction with the minimality of $lt(g)$. Hence $\overline{h}$ is the row with the smallest leading term in M'', and is therefore similar to h. □

Lemma 3 Let $F = \{f_1, \ldots, f_m\} \subseteq R[x_1, \ldots, x_n]$, $t \in T$ and let f be a polynomial of the form $f = \sum_{i=1}^m h_i f_i$ with $h_i \in R[x_1, \ldots, x_n]$ and $lt(h_i f_i) \leq t$. Assume that all $Spol(f_i, f_j)$ with $lcm(lt(f_i), lt(f_j)) \leq t$ are reducible to 0 w.r.t. F. Then any reduction of f w.r.t. F leads to 0.

Proof. We may assume that $t = max\{lt(h_i f_i) | i = 1, \ldots, m\}$. The proof is made by induction on t.

Case $t > lt(f)$ Let $i_1, \ldots, i_p$ be the indices for which $lt(h_i f_i)$ has the maximum value t and let $lt(h_i) = u_i, lc(h_i) = a_i$. Then $\sum_{j=1}^p a_{i_j} = 0$ and therefore $\sum_{j=1}^p a_{i_j} u_{i_j} f_{i_j}$ can be written as $\sum_{j=1}^{p-1} b_i v_i Spol(f_{i_j}, f_{i_{j+1}})$ with $b_i \in K$ conveniently chosen and $v_i = t/lcm(lt(f_{i_j}), lt(f_{i_{j+1}}))$. But each S-polynomial $Spol(f_{i_j}, f_{i_{j+1}})$ is reducible to 0, hence can be written as a combination of the form $\sum_{i=1}^m h'_i f_i$ with $lt(h'_i f_i) < lcm(lt(f_{i_j}), lt(f_{i_{j+1}})) = t/v_i$. Summarising, $\sum_{j=1}^p a_{i_j} u_{i_j} f_{i_j}$ can be written in the form $\sum_{i=1}^m h''_i f_i$ with $lt(h''_i f_i) < t$, for appropriately chosen h''_i. Hence, defining $g_i = h_i - a_i u_i + h''_i$ if $i \in \{i_1, \ldots, i_p\}$ and $g_i = h_i + h''_i$ otherwise, we have $f = \sum_{i=1}^m g_i f_i$ and $lt(g_i f_i) < t$ for all i. We can now apply the induction hypothesis and infer that any reduction of f w.r.t. F leads to 0.

Case $t = lt(f)$ There exists at least an index for which $lt(h_i f_i) = lt(f)$, which means f is reducible w.r.t. F. Let f_j be some polynomial in F whose leading term divides the leading term of f. We reduce f by f_j and we get f'. We can write $f' = \sum_{i=1}^m h'_i f_i$ with $h'_i = h_i$ for $i \neq j$ and $h'_j = h_j - lm(f)/lm(f_j)$. Then we either are in the situation that

$lt(h'_i f_i) \leq lt(f') < t$ for all i, and we can apply the induction hypothesis, or we are in the situation of the first case and we proceed as above.

Proof of point 3 of Theorem 2. Let p_i be the leading form of f_i, $i = 1, 2$, and let $q_i = f_i - p_i$. We have $Spol(f_{a_2+3}, f_1) = lc(f_1)f_{a_2+3}y^{b_1} - lc(f_{a_2+3})f_1x^{a_2+b_2-1}$. From point 1 we know $f_{a_2+3}y^{b_1} \simeq detpol(f_1x^{a_2}y^{b_2}, \ldots, f_1y^{b_2+a_2}, f_2x^{a_1}y^{b_1}, \ldots, f_2y^{a_1+b_1})$. Notice that in this determinant polynomial f_1 is multiplied by all terms of p_2, and f_2 is multiplied by all terms of p_1. From the obvious identity $f_1f_2 - f_2f_1 = 0$ we see that $p_2f_1 - p_1f_2$, which is a linear combination of the rows of the determinant polynomial above, is equal to $q_1f_2 - q_2f_1$ and has the leading term $lt(p_2f_1 - p_1f_2) = lt(q_1f_2 - q_2f_1) = lt(f_1q_2) = x^{b_2+a_1+a_2-1}y^{b_1} = lt(f_{a_2+3}y^{b_1})$. Hence, using Lemma 2 we infer that $q_1f_2 - q_2f_1$ is similar to the determinant polynomial mentioned above, which means $f_{a_2+3}y^{b_1} = lc(f_{a_2+3})/lc(f_1q_2)(q_1f_2 - q_2f_1)$. Then $Spol(f_{a_2+3}, f_1) = lc(f_{a_2+3})/lc(q_2)(q_1f_2 - q_2f_1) - lc(f_{a_2+3})f_1x^{a_2+b_2-1} = lc(f_{a_2+3})/lc(q_2)(q_1f_2 - (q_2 - lm(q_2))f_1)$. Now the conditions of Lemma 3 are satisfied for $f = Spol(f_{a_2+3}, f_1), t = lt(Spol(f_{a_2+3}, f_1))$ and $F = \{f_1, \ldots, f_{a_2+3}\}$. This means all normal forms of $Spol(f_{a_2+3}, f_1)$ with respect to F are 0.

For the case $b_1 = 0$, the argument is quite similar but f_{a_2+3} is replaced by the result of reducing $Spol(f_{a_2+1}, f_{a_2+2})$ once by f_{a_2+2}. $\square$

Let us study now some sufficient conditions for the theorem above to stay true (possibly with minor changes) when the coefficients are not indeterminates.

Theorem 3 Let $f_1, f_2 \in R[x, y]$. Let $lt(f_i) = x^{a_i}y^{b_i}, i = 1, 2$ and assume $a_1 > a_2$ and $b_1 < b_2$. Then there exist $k \leq a_2 + 2$ and $0 \leq a_k < a_{k-1} < \ldots < a_3 < a_2$ such that the computation of a Gröbner basis of $\{f_1, f_2\}$ w.r.t. degree lexicographical order $(x > y)$ proceeds as follows:

1. $Spol(f_{i+1}, f_{i+2}) \longrightarrow^*_{f_{i+2}} f_{i+3}$ and
   $lt(f_{i+3}) = x^{a_{i+3}}y^{b_2+a_1+a_2-a_{i+2}-a_{i+3}}$, for $i = 0, \ldots, k-3$
   $Spol(f_{k-1}, f_k) \longrightarrow^*_{f_k} f_{k+1}$ and
   $deg(lt(f_{k+1})) < deg(lcm(lt(f_{k-1}), lt(f_k)))$

2. $f_{i+3} \simeq detpol(f_1x^{a_2-a_{i+3}-1}y^{b_2-b_1-a_{i+2}+a_{i+3}+1}, \ldots, f_1y^{b_2-b_1+a_2-a_{i+2}},$
   $f_2x^{a_1-a_{i+3}-1}y^{a_{i+3}-a_i+2+1}, \ldots, f_2y^{a_1-a_{i+2}})$, for $i = 0, \ldots, k-3$

3. Constant factors appear systematically. They are exactly the same factors as would appear in a polynomial remainder sequence computation starting with the 2 polynomials in x obtained by dehomogenizing the leading forms of f_1 and f_2, and can be computed like in the subresultant polynomial remainder sequence.

4. If, moreover, the following two conditions are satisfied:

   a) the greatest common divisor of the leading forms of f_1 and f_2 has degree 0 in x,

   b) the coefficient of $x^{deg(f_2)-1}$ in f_2 is not 0,

   then $a_k = 0$ and, in the case $b_1 > 0$, we have $lt(f_{k+1}) = x^{b_2+a_1+a_2-1}$, $Spol(f_{k+1}, f_1) \rightarrow^* 0$ and the Gröbner basis of $\{f_1, f_2\}$ is $\{f_1, f_2, f_3, \ldots, f_{k+1}\}$ For the case $b_1 = 0$, we have already $Spol(f_{k-1}, f_k) \longrightarrow^* 0$. Thus $f_{k+1} = 0$ and the Gröbner basis is $\{f_1, f_2, f_3, \ldots, f_k\}$

Proof. Similar to the proof above. The a_i's are the degrees in the polynomial remainder sequence of $\overline{p}_1$ and $\overline{p}_2$. The condition a) from the last point implies that the last polynomial in the polynomial remainder sequence is a constant, hence $a_k = 0$. Condition b) ensures that the leading term of f_{k+1} is $x^{b_2+a_1+a_2-1}$. □

Corollary 1 If $f_1, f_2 \in R[x, y]$ satisfy the conditions of the theorem above (including conditions a) and b)) then the ideal generated by f_1 and f_2 is of dimension 0.

2.3 Determinant Polynomials

We saw that the fact that polynomials are similar to the determinant polynomials of their prerequisites holds in the case of polynomial remainder sequences as well as for the Gröbner basis of the example of the previous section. Also notice that the S-polynomial and the reduction of two polynomials can be written as a determinant polynomial:

Lemma 4 Let $f, g, f_1, f_2, h \in R[x_1, \ldots, x_n]$. If $lt(f) = lt(g)$, then $red(f, g) = detpol(f, g)$. As a consequence, we have:

1. $Spol(f_1, f_2) = detpol(u_2 f_2, u_1 f_1)$, where $u_i = lcm(lt(f_1), lt(f_2))/lt(f_i)$.

2. If $f_1 \rightarrow_{f_2} h$ then $h = detpol(uf_2, f_1)$, where $u = lt(f_2)/lt(f_1)$.

Proof We triangularize the matrix $M = mat(f, g)$ reducing the second row by the first one. We obtain the matrix $M' = mat(f, red(f, g))$. When we performed the row reduction on M we multiplied a row by $lc(f)$, hence $detpol(M') = lc(f) \cdot detpol(M)$. On the other hand, since M' is triangular, its determinant polynomial can be easily computed: $detpol(M') = lc(f)red(f, g)$. We get thus $detpol(M) = red(f, g)$.□

We are lead to the first natural conjecture:

Conjecture 1 Any intermediate Gröbner basis polynomial adjoined to the basis during a Gröbner basis algorithm is similar to the determinant polynomial of its prerequisites in the input basis.

Bruno Buchberger formulated this conjecture in [Buc83] and tested it on numerous examples. For most of these examples the conjecture hold, but for some no proof was found either that the conjecture is true or that it is false. We show that the conjecture is false, by giving the following counterexample.

Counterexample 1 Input basis $F = \{f_1, f_2, f_3, f_4\}$, degree lexicographical order, $x > y > z > t$.
$f_1 = yzt + z^2 + x$
$f_2 = xzt + y^2$
$f_3 = xyt + xz + z$
$f_4 = xyz + t^2$

$Spol(f_1, f_2) = xf_1 - yf_2 = xz^2 - y^3 + x^2 =: f_5$
$f_5 = detpol(xf_1, yf_2)$
$Spol(f_3, f_4) = zf_3 - tf_4 = xz^2 - t^3 + z^2 \rightarrow_{f_5} -y^3 + t^3 + x^2 - z^2 =: f_6$
$f_6 = detpol(detpol(xf_1, yf_2), detpol(zf_3, tf_4))$
The prerequisites of f_6 in F are xf_1, yf_2, zf_3, tf_4, but f_6 is not similar to $detpol(xf_1, yf_2, zf_3, tf_4)$, they do not even have the same leading term. This can be seen from the next picture. Notice that even if we would allow to choose differently the fixed columns in the definition of the

determinant polynomial(like in the original definition of Buchberger) or if we would take only three out of the four prerequisite polynomials, the determinant polynomial would still be different from f_6.

$$\begin{array}{cc}
 & \begin{array}{cccccc} xyzt & xz^2 & y^3 & t^3 & x^2 & z^2 \end{array} \\
\begin{array}{c} xf_1 \\ yf_2 \\ zf_3 \\ tf_4 \end{array} & \begin{vmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{vmatrix} \\
 & \begin{array}{cccccc} & & \uparrow & \uparrow & & \\ & & lt(f_6) & lt(detpol(M)) & & \end{array}
\end{array}$$

Roughly speaking , the reason for the failure of the conjecture is that it requires that the number of distinct leading terms whose coefficients are canceled during the computation of a polynomial f from its prerequisites has to be exactly the number of prerequisites minus one. (These terms will correspond to the fixed columns in the determinant polynomial.) But this is not the case in general. In the example above, only the coefficients of $xyzt$ and xz^2 are canceled during the computation of f_6, whereas for the determinant polynomial we would need three terms to be fixed.

Let us see, in the cases where the conjecture above is true, what can be inferred on the growth of the coefficients. Assume the coefficient ring is $\mathbb{Z}$ and the norm of an integer is, as usual, its absolute value. Let d be a bound for the norm of the coefficients of the polynomials in the input basis F. The primitive part of f will be equal to the primitive part of $detpol(prereq(f))$. Therefore the norm of the coefficients of the primitive part of f will be bounded by $D(r, d)$, where r is the number of prerequisites and $D(r, d)$ denotes a bound on the norm of the determinant of a $r \times r$ matrix over $\mathbb{Z}$ with entries of norm at most d. We can take the bound $D(r, d) = r!d^r$, which can be immediately deduced from the definition of the determinant, or the bound $D(r, d) = \sqrt{r}^r d^r$ given by Hadamard's inequality. A finer bound is $r!d_1 \dots d_r$, respectively $\sqrt{r}^r d_1 \dots d_r$, where d_i is a bound on the norm of the elements of the i-th row of the matrix.

2.4 Nested Determinant Polynomials

In the last counterexample we saw that f_6, even if it cannot be written as a determinant polynomial, can be written as $f_6 = detpol(detpol(xf_1, yf_2), detpol(zf_3, tf_4))$ This suggests the following definition.

Definition 11 (Nested determinant polynomial)
A polynomial $f \in R[x_1, \ldots, x_n]$ is called a *nested determinant polynomial* of a set of polynomials F if either $f \in F$ or $f = detpol(f_1, \ldots, f_k)$ where each f_i is again a nested determinant polynomial of F. We call f a *simple nested determinant polynomial* if moreover, each f_i is a simple nested determinant polynomial of a subset $F_i \subseteq F$ such that the F_i's are pair-wise disjoint.

Lemma 4 implies that any intermediate Gröbner basis polynomial f can be expressed as a nested determinant polynomial of its prerequisites. What we would like is to express it as a simple nested determinant polynomial(that is, none of the prerequisites should appear more than once in the expression of the nested determinant polynomial).

Like in the case of determinant polynomial, let us see what can we infer about the growth of the coefficients when we work over integers. For a simple nested determinant polynomial of $prereq(f)$, the bounds $r!d^r$ and $\sqrt{r}^r d^r$ hold again, with r the cardinality of $prereq(f)$ and d a bound on the norm of the coefficients of the input polynomials. We prove this by induction on the structure of the simple nested determinant polynomial. Assume f is a simple nested determinant polynomial of F, $f = detpol(f_1, \ldots, f_k)$ and each f_i is a simple nested determinant polynomial of F_i, where the F_i's form a partition of F. Assume the cardinality of F is r and the cardinality of F_i is r_i, hence $r_1 + \ldots + r_k = r$. By induction hypothesis the coefficients of f_i are bounded by $r_i!d^{r_i}$, respectively $\sqrt{r_i}^{r_i} d^{r_i}$. Then the coefficients of f will be bounded by $k! \prod_{i=1}^k r_i! d^{r_i} = (k! \prod_{i=1}^k r_i!) d^r$, respectively $\sqrt{k}^k \prod_{i=1}^k \sqrt{r_i}^{r_i} d^{r_i} = (\sqrt{k}^k \prod_{i=1}^k \sqrt{r_i}^{r_i}) d^r$. We are left to prove that $k! \prod_{i=1}^k r_i! \leq r!$, respectively $\sqrt{k}^k \prod_{i=1}^k \sqrt{r_i}^{r_i}) \leq \sqrt{r}^r$. These inequalities are proven in the following lemma.

Lemma 5 Let $k, n, n_i \in \mathbb{N}\backslash\{0\}$, for $i = 1, \ldots, k$ such that $n_1 + \ldots + n_k = n$. Then the following inequalities hold:

a) $k!n_1!\ldots n_k! \leq n!$

b) $k^k n_1^{n_1} \ldots n_k^{n_k} \leq n^n$

Proof Without loss of generality we can assume $n_1 \leq n_2 \leq \ldots \leq n_k$.

a) We can rewrite the inequality as $\prod_{i=1}^k i n_i! = \prod_{i=1}^k i \cdot 2 \cdot 3 \cdot \ldots \cdot n_i \leq \prod_{i=1}^k (n_1 + \ldots + n_{i-1} + 1) \ldots (n_1 + \ldots + n_{i-1} + n_i)$. But it is clear that $i \leq n_1 + \ldots + n_{i-1} + 1$ and $j \leq n_1 + \ldots + n_{i-1} + j$ for $j = 2, \ldots n_i$.

b) We prove that for fixed k and n the maximum of the expression $k^k n_1^{n_1} \ldots n_k^{n_k}$ is reached for $n_1 = \ldots = n_{k-1} = 1$, $n_k = n - k + 1$.

First we show that if for some n_i, n_j with $1 < n_i \leq n_j$ we replace in the expression above n_i by $n_i - 1$ and n_j by $n_j + 1$ the value of the expression increases, i.e. $n_i^{n_i} n_j^{n_j} < (n_i - 1)^{n_i - 1}(n_j + 1)^{n_j + 1}$, or equivalently $n_i^{n_i}/(n_i - 1)^{n_i - 1} < (n_j + 1)^{n_j+1}/n_j^{n_j}$. The function $f(x) = x^x/(x-1)^{x-1}$ is strictly increasing for $x \geq 2$, because its derivative $f'(x) = f(x)(\ln x - \ln(x-1))$ is strictly positive. But $n_i < n_j + 1$ implies $f(n_i) < f(n_j + 1)$ which is exactly the inequality we are looking for. We iterate the process above, i.e. for any $1 < n_i \leq n_j$ we replace n_i by $n_i - 1$ and n_j by $n_j + 1$, and the process will stop when $n_1 = \ldots = n_{k-1} = 1$, $n_k = n - k + 1$.

We are now left to prove $k^k (n - k + 1)^{n-k+1} \leq n^n 1^1$. This results by the same argument as above. By putting $\overline{k} = 2, \overline{n} = n + 1$, $\overline{n_1} = min(k, n - k + 1), \overline{n_2} = max(k, n - k + 1)$, the inequality becomes $\overline{n_1}^{\overline{n_1}} \overline{n_2}^{\overline{n_2}} \leq \overline{n}^{\overline{n}} 1^1$, and we know that $\overline{n}^{\overline{n}} 1^1$ is the maximum value of the expression $\overline{n_1}^{\overline{n_1}} \overline{n_2}^{\overline{n_2}}$ over all $\overline{n_1}, \overline{n_2}$ such that $\overline{n_1} + \overline{n_2} = n + 1$. $\square$

In the case of a nested determinant polynomial, bounds of the same form hold, but this time r is replaced by the cardinality of $prereq(f)$ counting for each element multiple occurrences in the expression of the nested determinant polynomial. Hence, the more elements are repeated, the worse the bound is. We are therefore interested in obtaining simple nested determinant polynomials. Another advantage of obtaining them will become clear in the next chapter, where we will see that redundant constant factors can be determined for the polynomials that can be expressed as determinant polynomials or simple nested determinant polynomials.

The nested determinant polynomial obtained by applying Lemma 4 is not always simple nested determinant polynomial. But we might hope to find ways for rewriting it such that we get a similar simple nested determinant polynomial. One such transformation rule is the following:

Lemma 6 If $f, g, h \in R[x_1, \ldots, x_n]$ and $lt(f) = lt(g) = lt(h)$ then $detpol(detpol(f, g), detpol(f, h)) = lc(f)detpol(f, g, h)$.

Proof In the matrix $M = mat(f, g, h)$ we reduce the second and third row by the first one. We get $lc(f)^2 M = mat(f, red(f, g), red(f, h))$, hence $lc(f)^2 detpol(M) = lc(f)detpol(red(f, g), red(f, h))$, which implies the equality we are looking for. □

We state now our second conjecture.

Conjecture 2 Any intermediate Gröbner basis polynomial adjoined to the basis during a Gröbner basis algorithm is similar to a simple nested determinant polynomial of its prerequisites in the input basis.

This conjecture is also false. The counterexample we found is rather elaborate and actually could be skipped since the counterexample to the conjecture of the next section can also serve as a counterexample to Conjecture 2. We give it nevertheless because it will serve as a starting-point of an example in a subsequent chapter.

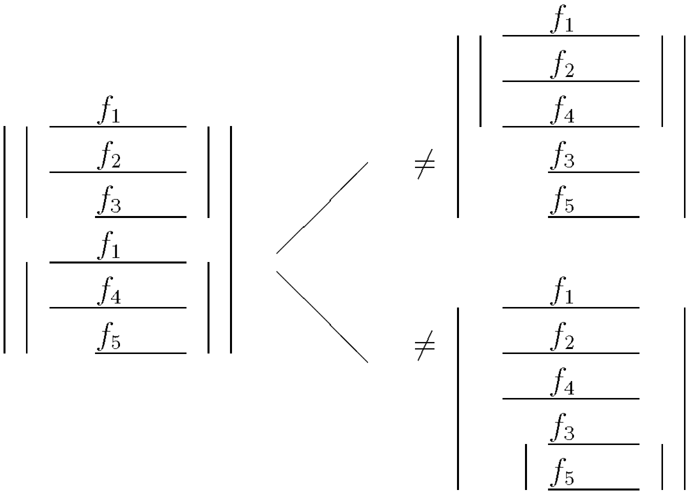
Counterexample 2 Let $t_1, \ldots, t_5 \in T$ with $t_i < t_{i+1}$ and $f_1, \ldots, f_5$ polynomials such that $lt(f_1) = lt(f_2) = lt(f_4) = t_5$, $lt(f_3) = lt(f_5) = t_4$ and the coefficients of $t_1, \ldots, t_5$ in $f_1, \ldots, f_5$ are distinct indeterminates (except the coefficients of t_5 in f_3 and f_5 which are 0). Assume g is an intermediate polynomial in a Gröbner basis computation and $prereq(g) = \{f_1, f_2, f_3, f_4, f_5\}$. Assume that following the Gröbner basis algorithm and using Lemma 4 and possibly Lemma 6 we got the following expression for g: $g = detpol(detpol(f_1, f_2, f_3), detpol(f_1, f_4, f_5))$. Notice that $lt(g) = t_2$ and g depends on all indeterminate coefficients appearing in $f_1, \ldots, f_5$ (we can see this by computing the coefficients of t_2 and t_1 in g by computer, since their size is rather big for a computation by hand.) We will show that g cannot be rewritten as a simple nested determinant polynomial. We examine all the simple nested determinant polynomials with leading term t_2 that can be formed using f_1, f_2, f_3, f_4, f_5, and that depend on all indeterminate coefficients of $f_1, \ldots, f_5$.

If in a matrix $mat(g_1, \ldots, g_m)$ with g_i polynomials with indeterminate coefficients, there exists an index i such that $lt(g_i) > lt(g_j)$ for all $j \neq i$, then the determinant polynomial of this matrix is equal to $\pm lc(g_i) detpol(g_1, \ldots, g_{i-1}, g_{i+1}, \ldots, g_m)$, hence it does not depend on all indeterminate coefficients of g_i. By such an argument we can dismiss many of the simple nested determinant polynomials we can construct using f_1, f_2, f_3, f_4, f_5. We prove first that the three polynomials f_1, f_2, f_4 having the leading term t_5 must appear on the same imbrication level, i.e. in a subexpression of the form $detpol(f_1, f_2, f_4, \ldots)$. If this were not true, one of these three polynomials, say f_1, will belong to a subexpression D of the form $detpol(f_1, h_1, \ldots, h_m)$ with $f_2, f_4 \notin \{h_1, \ldots, h_m\}$. Since no other polynomial in D except f_1 will have leading term t_5, by the argument above D and consequently the whole simple nested determinant polynomial will not depend on all the indeterminate coefficients of f_1.

We are now left with only a few possibilities for the simple nested determinant polynomial: $detpol(f_1, f_2, f_4, f_3, f_5)$, $detpol(f_1, f_2, f_4, detpol(f_3, f_5))$, $detpol(detpol(f_1, f_2, f_4), f_3, f_5) = detpol(detpol(f_1, f_2, f_4), detpol(f_3, f_5))$, $detpol(detpol(f_1, f_2, f_4, f_3), f_5)$. We dismiss the first one because it has the leading term t_1 and the last one because $lt(f_5) > lt(detpol(f_1, f_2, f_4, f_3))$ and therefore it does not depend on all indeterminates. For the other two possibilities, by computing the coefficients of t_2 and t_1 we can check that none of them is similar to g. This can be seen graphically in the next picture.

2.5 Row reductions

When we make a row reduction, or a polynomial reduction, or a S-polynomial computation we compute the reduction of the two polynomials (respectively vectors) involved. The only difference is that when



we make a row reduction or a polynomial reduction we replace one of the elements by the reduction of the two, whereas in an S-polynomial computation both polynomials are kept and possibly used in further computations.

It is a natural question to ask if we can simulate the Gröbner basis computation by performing row reductions in a matrix. This would be a step towards a subresultant-like construction for the Gröbner basis algorithm, and would also help clarifying the relationship between the Gröbner basis algorithm and Gaussian elimination.

Let $F = \{f_1, \ldots, f_m\}$ be the input basis and let $f_{m+1}, \ldots, f_l$ be the polynomials introduced successively in the basis during the Gröbner basis algorithm. As before, f will denote a polynomial introduced in the basis at some step of the algorithm, i.e. $f \in \{f_1, \ldots, f_l\}$. Let $g_1, g_2, \ldots$ be the sequence of polynomials obtained in the intermediate steps of the normal form computations. polynomials that we obtain by succesively reducing an S-polynomial, before we arrive at an irreducible polynomial)

We will give now a more precise meaning to "simulating the Gröbner basis computation by Gaussian elimination", by analogy with the simulation of pseudo-divisions by Gaussian elimination in the univariate case. Recall that since the operation of shifting by multiplication with a term is not available in a matrix, two polynomials of the form g and mg with $m \in T$ have to be represented as different rows in a matrix. We start with a matrix having as rows some of the input polynomials, multiplied by certain terms. We will try to perform on this matrix row reductions which correspond to the reductions performed by the Gröbner basis algorithm, and we expect to obtain successively on the rows of the matrix some of the polynomials $f_{m+1}, \ldots, f_l$ multiplied by certain terms, until finally we obtain f on one of the rows. For simulating the computation of a normal form of an S-polynomial $Spol(f_{i_0}, f_{i_1}) = g_{p+1} \rightarrow_{f_{i_2}} g_{p+2} \rightarrow_{f_{i_3}} \ldots \rightarrow_{f_{i_q}} f$, with $Spol(f_{i_0}, f_{i_1}) = red(u_1 f_{i_1}, u_0 f_{i_0})$, $g_{p+j} = red(u_j f_{i_j}, g_{p+j-1})$, for $j = 2, \ldots, q-1$, $f = red(u_q f_{i_q}, g_{p+q-1})$ and $u_i \in T$, we start from a matrix having the rows $u_0 f_{i_0}, \ldots, u_q f_{i_q}$. First we will reduce the first row by the second. Then, we reduce the first row by the third row, then by the fourth row and so on. In the end, in the first row we obtain the polynomial f. In the subsequent computations that we want to simulate, we may need to use the polynomial uf for some $u \in T$, hence the row uf must be available. The rows $uu_0 f_{i_0}, \ldots, uu_q f_{i_q}$. must therefore be in the initial matrix, and the computation of uf is simulated in an analogous way.

We will now collect together all the operations that we should simulate for obtaining a certain Gröbner basis polynomial f. They are S-polynomial computations and polynomial reductions performed during the Gröbner basis algorithm, possibly shifted by multiplication with a term. We can represent such operations by tuples of the form (u, h_1, h_2, h_3), where $u \in T$ and $h_i \in R[x_1, \ldots, x_n]$. The polynomias h_i are such that the operation $Spol(h_2, h_1) = h_3$ or the operation $h_2 \rightarrow_{h_1} h_3$ was performed by the Gröbner basis algorithm, and u is a term by which we have to shift, that is, we compute instead $Spol(uh_2, uh_1) = uSpol(h_2, h_1) = uh_3$, respectively $uh_2 \rightarrow_{uh_1} uh_3$.

Definition 12 The set $S(f)$ of S-polynomial computations and polynomial reductions that produce a polynomial f during a Gröbner basis algorithm is defined in the following way: If $f \in F$ then $S(f) = \emptyset$. Otherwise, assume f is obtained by a normal form computation $Spol(f_{i_0}, f_{i_1}) = g_{p+1} \rightarrow_{f_{i_2}} g_{p+2} \rightarrow_{f_{i_3}} \ldots \rightarrow_{f_{i_q}} f$, with

$Spol(f_{i_0}, f_{i_1}) = red(u_1 f_{i_1}, u_0 f_{i_0})$, $g_{p+j} = red(u_j f_{i_j}, g_{p+j-1})$, for $j = 2, \ldots, q-1$, $f = red(u_q f_{i_q}, g_{p+q-1})$ and $u_i \in T$. Then

$$\begin{aligned} S(f) &= \{(1, f_{i_1}, f_{i_0}, g_{p+1}), (1, f_{i_2}, g_{p+1}, g_{p+2}), \ldots, (1, f_{i_q}, g_{p+q-1}, f)\} \cup \\ &\quad \cup_{j=0}^{q} u_j S(f_{i_j}) \end{aligned}$$

where $uS(h)$ is the set obtained from $S(h)$ by replacing each operation of the form (w, h_1, h_2, h_3) by (uw, h_1, h_2, h_3).

Notice that if for the computation of some S-polynomial we change $Spol(f_{i_1}, f_{i_0})$ to $Spol(f_{i_0}, f_{i_1})$, the Gröbner basis algorithm remains essentially the same, up to some changes of sign.

The set of polynomials from $T \cdot F$ that appear in the computations in $S(f)$ are exactly the prerequisites of f. (This is an imediate consequence of the way we defined $S(f)$ and $prereq(f)$.)

To execute the set of operations $S(f)$ amounts to performing a set of reductions, namely for each operation (u, h_1, h_2, h_3) we execute the reduction $red(uu_1h_1, uu_2h_2) = uh_3$, for $u_i = lcm(lt(h_1), lt(h_2))/lt(h_i)$, $i = 1, 2$.

We can try to simulate these reductions by row reductions in a matrix. Here is our next conjecture:

Conjecture 3 The set of S-polynomial computations and polynomial reductions that produce f can be ordered such that the resulting sequence of reductions can be performed as row reductions in the matrix $mat(prereq(f))$, and the matrix thus obtained has one of the rows similar to f.

Remark 3 The first natural way to order the set $S(f)$ is to use the order in which the underlying computations were performed by the Gröbner basis algorithm. Namely the operation (u, h_1, h_2, h_3) comes before the operation (u', h'_1, h'_2, h'_3) iff the operation $(1, h_1, h_2, h_3)$ was done by the Gröbner basis algorithm before the operation $(1, h'_1, h'_2, h'_3)$, or, in case $h_i = h'_i$ for $i = 1, 2, 3$, if $u < u'$.

Unfortunately such an order order may bring us in the situation that we modify in the matrix $mat(prereq(f))$ a row that we will need later in its current form. For example, assume we must simulate the computation of $Spol(f_{i_0}, f_{i_1})$, so we reduce $u_0 f_{i_0}$ by $u_1 f_{i_1}$. Then, later in the Gröbner basis computation, assume we need in some S-polynomial computation or polynomial reduction to use $u_0 f_{i_0}$. But this row is not available any more.

We would like therefore to examine all the possible permutations of the computations in $S(f)$ and decide if it is possible to perform one of them on the matrix $mat(prereq(f))$. For this we need a tool for tracing the Gröbner basis algorithm as well as sequences of row reductions in a matrix, in such a way as to capture the essential dependencies between different computations. For any two reduction operations it should become clear if they are independent, and hence can be permuted, or if one of them must necessarily precede the other. We will devise now such a tool, which we will call reduction graph.

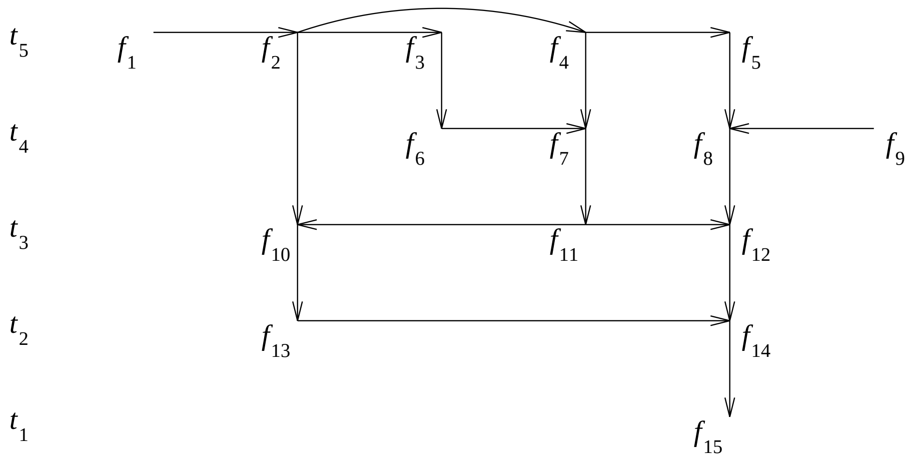
Definition 13 (Reduction Graph)
A reduction graph is a finite labeled directed graph $G = (V, E)$ together with a partition P of the set E of edges, satisfying the following properties:

1. Each vertex is labeled by a polynomial in $R[x_1, \ldots, x_n]$.
2. Each edge is labeled by one of the labels "horizontal" or "vertical".
3. P is a partition of E having the property that each element of P is a set of two edges, a horizontal one (v_1, v_2) and a vertical one (v_3, v_4), such that $v_2 = v_3$ and $label(v_4) = red(label(v_1), label(v_2))$.
4. Any vertex is the end-point of at most one vertical edge.

The subgraph of G corresponding to a term $u \in T$ is the subgraph G_u obtained from G by restricting the set of vertices to those labeled by polynomials with leading term u. The set of inputs and the set of outputs of the graph (denoted by $input(G)$ and $output(G)$) are defined as the set of vertices which are not end-points of vertical edges, respectively are not start-points of vertical edges.

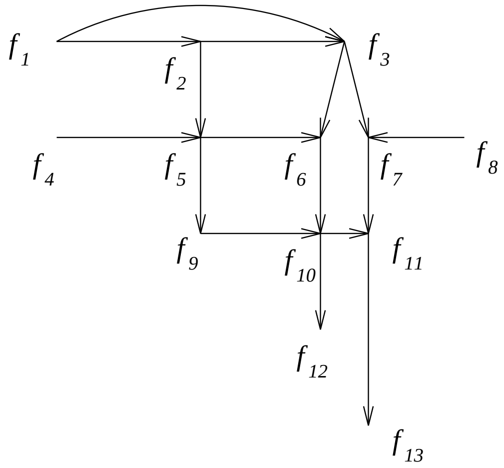
From the way we defined the operation red, it results that any horizontal edge connects vertices labeled by polynomials having the same leading term. For vertical edges, the end-point is labeled by a polynomial with strictly lower leading term than the start-point. The subgraph G_u corresponding to a term u has only horizontal edges. Notice that the partition P establishes a one-to-one correspondence between the set of horizontal edges and the set of vertical edges. When speaking about a "pair of edges" we mean a set of two edges that belongs to P and we will write first the horizontal edge, then the vertical one.

We will draw reduction graphs in the following way: the vertices labeled by polynomials having the same leading term u will appear on the same horizontal line, and these lines will be arranged in decreasing order of the corresponding term. We will usually draw horizontally the edges labeled "horizontal" and vertically the edges labeled "vertical". Here are some examples of reduction graphs:



The f_i's are the polynomials that label the vertices. We have $t_i \in T$, $t_5 > t_4 > \ldots > t_1$, $lc(f_1) = lc(f_2) = lc(f_3) = lc(f_4) = lc(f_5) = t_5$, $lc(f_6) = lc(f_7) = lc(f_8) = lc(f_9) = t_4$, $lc(f_{10}) = lc(f_{11}) = lc(f_{12}) = t_3$, $lc(f_{13}) = lc(f_{14}) = t_2$, $lc(f_{15}) = t_1$. The labels of the inputs of the graph are $f_1, f_2, f_3, f_4, f_5, f_9$ and those of the outputs $f_1, f_6, f_9, f_{11}, f_{13}, f_{15}$. The polynomials f_i are related in the following way: $red(f_1, f_2) = f_{10}, red(f_2, f_3) = f_6$ and so on. In this example there is only one possibility for the partition P. Namely, we see that for each horizontal edge

(v_i, v_j) there is a unique vertical edge having v_j as startpoint. Hence this pair of edges obviously belongs to P.



Here the graph G does not uniquely determine the partition P, hence we have to specify it explicitely. Denote by v_i the vertex labeled by the polynomial f_i. We must define especially which of the vertical edges (v_3, v_6) and (v_3, v_7) correspond to the horizontal edges (v_1, v_3) and (v_2, v_3). So we can have either $\{(v_1, v_3), (v_3, v_6)\}, \{(v_2, v_3), (v_3, v_7)\} \in P$ or $\{(v_1, v_3), (v_3, v_7)\}, \{(v_2, v_3), (v_3, v_6)\} \in P$.

By reduction graphs we intend to represent sets of reduction operations between polynomials (or matrix rows). Namely, each pair of edges $\{(v_1, v_2), (v_2, v_3)\} \in P$ with $label(v_i) = h_i, i = 1, 2, 3$ signifies the fact that the polynomial h_2 is reduced by h_1 and the result is the polynomial $h_3 = red(h_1, h_2)$. If we are given the polynomials that label the vertices in $input(G)$ and we want to perform in a sequential way the reductions described by the reduction graph, it is clear that some reductions must necessarily precede others. Namely, before performing the reduction described by $\{(v_1, v_2), (v_2, v_3)\}$ we must perform the reductions described by pairs $\{(v_1', v_2'), (v_2', v_3')\}$ with $v_3' = v_1$ or $v_3' = v_2$, in order that $label(v_1), label(v_2)$ are computed before we attempt to perform their reduction. We can always find an order for performing the reductions such that the condition above is satisfied. It suffices, for example, to perform the reductions in decreasing order of the leading term of the two operands of $red(.,.)$ (i.e. in a "top-down" order on the picture). We will obtain in the end the polynomials labeling the vertices in $output(G)$.

If after computing a reduction described by $\{(v_1, v_2), (v_2, v_3)\}$ with $label(v_i) = h_i$, instead of keeping all the polynomials h_1, h_2, h_3, we delete h_2 (this is the case for row reductions in matrices and for polynomial reductions), then we must ensure that all reductions of the form $\{(v_2, v_2'), (v_2', v_3')\}$ were already performed (if we try to perform them later, the first operand will be missing). For an arbitrary reduction graph it may happen that this condition is impossible to satisfy.

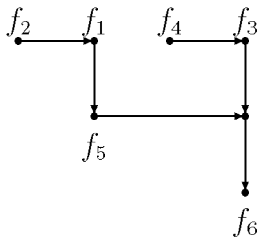
Given a Gröbner basis computation the set of S-polynomials and polynomial reductions that produce f can be traced by a reduction graph constructed in the following way: For each operation (u, h_1, h_2, h_3), with $u_i = lcm(lt(h_1), lt(h_2))/lt(h_i)$, $i = 1, 2$, we check if we already constructed a vertex for uu_1h_1 and a vertex for uu_2h_2. If not, we construct them now and we label them by uu_1h_1, respectively uu_2h_2. (One technical detail should be taken care of. It may happen accidentally that there are some $h_i \neq h_j$ such that $m_ih_i = m_jh_j$ for some $m_i, m_j \in T$. Nevertheless, the vertex constructed for m_ih_i should not be considered as vertex constructed for m_jh_j, we construct distinct vertices for them.) Let v_1 and v_2 be the vertices constructed for uu_1h_1 and uu_2h_2, respectively. We construct a vertex v_3 for uh_3 and assign to v_3 the label uh_3. Then we construct an horizontal edge (v_1, v_2) and a vertical edge (v_2, v_3) and we introduce the pair of edges $\{(v_1, v_2), (v_2, v_3)\}$ in P.

The graph constructed above will be called the reduction graph describing the Gröbner basis algorithm for computing f. Notice that the set of labels of the vertices in $input(G)$ is exactly $prereq(f)$. The labels of the vertices in $output(G)$ are polynomials from the current basis, multiplied by certain terms. Among them, there is a unique polynomial with minimum leading term, and this polynomial is f.

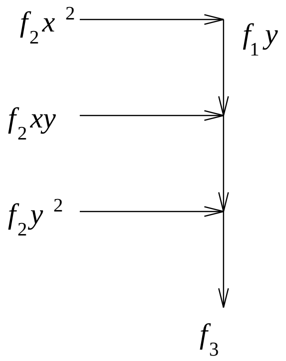
We describe now a variant of constructing a reduction graph for a Gröbner basis computation, which allows the construction of "redundant" reduction graphs, in the sense that some vertices will have intentionally the same label. Namely, for each operation $(u, h_1, h_2, h_3) \in S(f)$, even if some vertices were already constructed for uu_1h_1 and for uu_2h_2, we may construct some more of them. Let $v_1^1, \ldots, v_1^p$ be all the vertices constructed for uu_1h_1 and $v_2^1, \ldots, v_2^q$ all the vertices constructed for uu_2h_2. Then we may construct one or more horizontal edges $(v_1^{i_1}, v_2^{j_1}), \ldots, (v_1^{i_r}, v_2^{j_r})$, with $i_1, \ldots, i_r$ not neccessarily distinct, and also $j_1, \ldots, j_r$ not neccessarily distinct. We construct r vertices $v_3^1, \ldots, v_3^r$ for uh_3, labeled uh_3, and the vertical edges $(v_2^{j_1}, v_3^1), \ldots, (v_2^{j_r}, v_3^r)$. The pairs of edges $\{(v_1^{i_1}, v_2^{j_1}), (v_2^{j_1}, v_3^1)\}, \ldots, \{(v_1^{i_r}, v_2^{j_r}), (v_2^{j_r}, v_3^r)\}$ are introduced in the partition P. Such a graph will be called a redundant reduction graph describing the Gröbner basis algorithm for computing f.

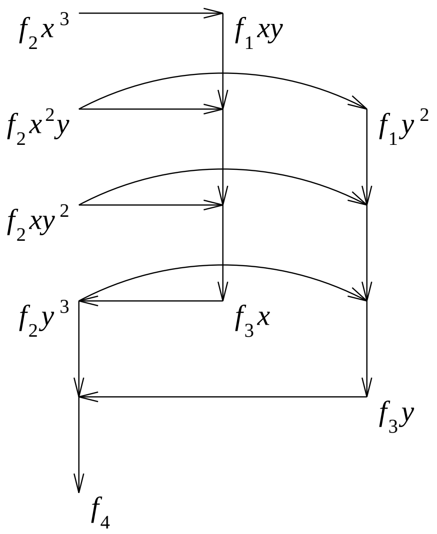
Here are some examples of reduction graphs for Gröbner basis computations.

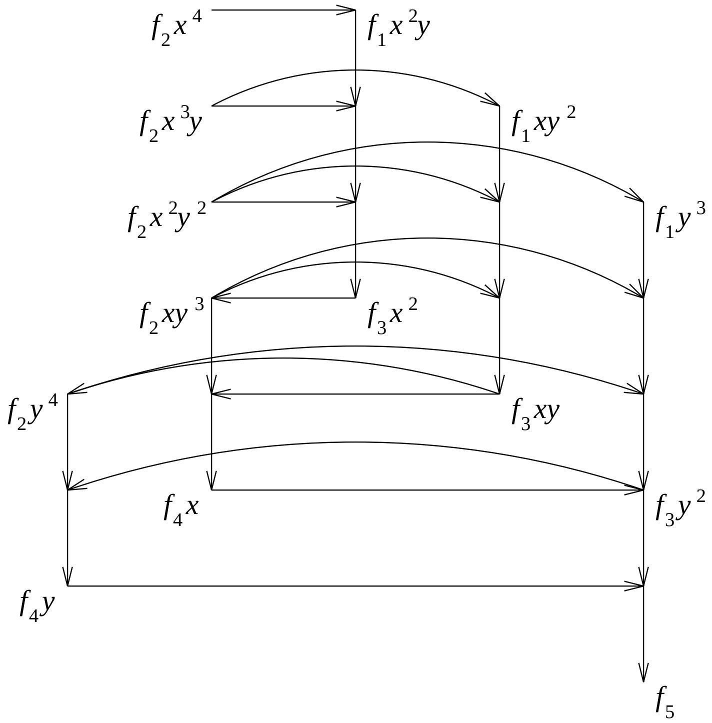
Example 2
1. For the Gröbner basis computation described in Counterexample 1 the reduction graph of f_6 looks as follows:



2. Recall the computation of a Gröbner basis of two bivariate dense polynomials with indeterminate coefficients studied in section 2.2. For the case $lt(f_1) = x^{a+2}y^b$ and $lt(f_2) = x^a y^{b+1}$ the reduction graphs of f_3, f_4 and f_5 are the following:



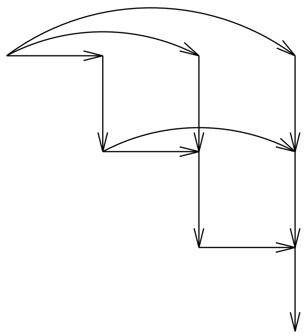




A sequence of row reductions performed on a matrix M for obtaining a matrix M' can be described by a reduction graph as follows. Let r be the number of rows of M and denote by m_i the i-th row of M.

For each $i \in \{1, \ldots, r\}$ we construct the vertex $v_{i,lt(m_i)}$, labeled by m_i. We consider then succesively the row reductions from the sequence. Assume we reduce the j-th row, by the i-th row, and let d and c be the vectors that appear in the current matrix as the j-th and the i-th row, respectively. Then $lt(c) = lt(d)$ and the vertices $v_{i,u}$ and $v_{j,u}$ with $u = lt(c)$ were already constructed. We construct a horizontal edge $(v_{i,u}, v_{j,u})$, a vertex $v_{j,w}$ with $w = lt(red(c,d))$ and a vertical edge $(v_{j,u}, v_{j,w})$. This pair of edges is also introduced in the partition P. Notice that the labels of the inputs of G are exactly the rows of M and the labels of the outputs are the rows of M'.

For example, the reduction graph describing the classical Gaussian elimination on a matrix with 4 rows will look as follows:



As a side remark, we note that for tracing a Gröbner basis computation we could define graphs having a simpler form than the reduction graphs, namely, we could store only once polynomials which are actually the same Gröbner basis polynomial multiplied by different terms. But we chose the representation by reduction graphs, for the purpose of comparison with matrix row reductions.

With the help of reduction graphs we can reformulate Conjecture 3 in the following way:

Conjecture 4 The reduction graph describing the Gröbner basis algorithm for computing f coincides with the reduction graph describing a sequence of row reductions performed on the matrix $mat(prereq(f))$.

Remark 4 A sufficient condition for Conjecture 1 (respectively Conjecture 2) to be true is that the reduction graph describing the Gröbner basis algorithm for computing f coincides with a reduction graph describing a sequence of reductions performed for the computation of a determinant polynomial(respectively simple nested determinant polynomial).

The computation of a determinant polynomial can be performed, up to similarity, using row reductions. Namely, we triangularize the matrix involved and we obtain the determinant polynomial, up to similarity, as the row with lowest leading term in the matrix.

The computation of a (simple) nested determinant polynomial $g = detpol(g_1, \ldots, g_j)$ up to similarity can be done by row reductions by computing, recursively $g_1, \ldots, g_j$, then triangularising the matrix $mat(g_1, \ldots, g_j)$. This sequence of reductions can be described by a reduction graph.

The sufficient condition mentioned above for Conjecture 1 (respectively Conjecture 2) is probably not a necessary condition, because it may happen accidentally that $f \simeq detpol(prereq(f))$ even if the reductions performed for obtaining f in the Gröbner basis algorithm do not correspond to a triangularization of the matrix $mat(prereq(f))$. Nevertheless, this situation seems quite unlikely to happen, and hard to check other than by direct computation.

For deciding if Conjecture 3 is true we need characterizations of the graphs describing a sequence of matrix row reductions and in particular of those describing a triangularization.

Theorem 4 *(Characterization of the graph of a sequence of row reductions)* Let (G, P) be a reduction graph. The following assertions are equivalent:

1. (G, P) describes a sequence of row reductions applied to the matrix $mat(input(G))$.

2. In each subgraph of G corresponding to a term there are no cycles and any vertex has in-degree at most one.

3. Each subgraph of G corresponding to a term consists of disjoint trees with edges oriented from the root towards the leaves.

Proof $1 \Rightarrow 2$ Assume (G, P) is the graph of a sequence of row reductions. Then for each $u \in T$ the subgraph $G_u = (V_u, E_u)$ corresponding to a term u consists of the horizontal edges describing reductions between rows having the same leading term. Each vertex can have the in-degree at most one because each row can be reduced at most once by a row with leading term u. Assume G_u has a cycle consisting of the edges $(v_1, v_2), \ldots, (v_{p-1}, v_p), (v_p, v_1)$ and let (v_j, v_{j+1}) be the reduction which is performed last among those in this cycle. But by this time v_j was already reduced by v_{j-1}, hence the row $label(v_j)$ was already replaced by a row with lower leading term, and the reduction $(v_j, v_{j+1}) \in E_u$ cannot be performed.

$2 \Rightarrow 1$ Assume each G_u has no cycles and each vertex has in-degree at most one. We prove that it is possible to perform in the matrix $mat(input(G))$ the row reductions described by G. For each $u \in T$ for which $G_u \neq \emptyset$, in decreasing order of the u's, we perform the reductions described by G_u as follows: We choose always an edge $(v_1, v_2) \in E_u$ such that all edges having v_2 as start-point were already considered. The absence of cycles guarantees that as long as there are edges not considered yet we can find one with the desired property. We reduce then the row $label(v_2)$ by the row $label(v_1)$. This reduction is possible because no edge (v, v_1) could be considered before (v_1, v_2), so $label(v_1)$ was not yet reduced, and $label(v_2)$ was not reduced yet because there are no edges except (v_1, v_2) having v_2 as end-point. Both rows have therefore u as leading term and we can perform the reduction. A technical detail: for correctly choosing the rows $label(v_1), label(v_2)$ above in case multiple identical rows appear in the matrix, we assign to each vertex an additional label indicating the index of the row in the matrix. For the vertices in $input(G)$ we assign the labels $1, 2, \ldots, |input(G)|$ (where $|input(G)|$ denotes the cardinality of $input(G)$) and then we propagate these labels by assigning to each vertex v the label of the unique vertex v' for which (v', v) is a vertical edge.

The equivalence $2 \Leftrightarrow 3$ is a simple graph theoretical result. Notice that the roots of the trees are the vertices with in-degree zero. $\Box$

Theorem 5 *(Characterization of the graph describing a matrix triangularization)* Let (G, P) be a reduction graph. Then (G, P) describes a triangularization of the matrix $mat(input(G))$ iff each non-empty subgraph of G corresponding to a term consists of exactly one tree with edges oriented from the root towards the leaves.

If these conditions are satisfied, then $output(G)$ has a unique vertex whose label has minimum leading term. The polynomial labeling this vertex is similar to the determinant polynomial of the matrix associated to the polynomials labeling the input vertices.

Proof According to the previous theorem (G, P) corresponds to row reductions in a matrix iff each subgraph G_u corresponding to a term u consists of disjoint trees with edges oriented from the root towards the leaves. The roots of the trees in G_u are exactly the vertices which have in-degree equal to zero, so they are not start-points of vertical edges and are therefore output vertices. This means that the rows labeling them are not reduced and are exactly the rows which will have the leading term equal to u in the final matrix. Hence, the final matrix is triangular iff each such graph G_u consists of exactly one tree.

Theorem 6 *(Characterization of the reduction graph of a simple nested determinant polynomial)* A reduction graph (G, P) with $G = (V, E)$ describes the computation of a simple nested determinant polynomial iff it consists of a single vertex or for some p there exist for $i = 0, \ldots, p$ reduction graphs (G_i, P_i) with $G_i = (V_i, E_i)$ satisfying the properties:

1. (G_i, P_i) form a decomposition of (G, P), in the sense that $\cup_{i=0}^{p} V_i = V$, $\cup_{i=0}^{p} E_i = E$, $\cup_{i=0}^{p} P_i = P$.

2. For $i = 1, \ldots, p$, the graphs (G_i, P_i) describe the computation of a simple nested determinant polynomial, and have therefore a unique vertex v_i whose label has minimum leading term among all labels of vertices in V_i.

3. (G_0, P_0) describes a matrix triangularization. The input vertices of G_0 are exactly $v_1, \ldots, v_p$.

4. The reduction graphs (G_i, P_i) are disjoint, more precisely $V_i \cap V_j = \emptyset$ for $1 \leq i, j \leq p$ and $V_0 \cap V_i = \{v_i\}$.

5. The labels of the vertices $input(G)$ are all distinct.

If these conditions are satisfied, then $output(G)$ has a unique vertex whose label has minimum leading term. The polynomial labeling this vertex is similar to a simple nested determinant polynomial of the set of polynomials labeling $input(G)$.

Proof From the definition of a simple nested determinant polynomial we can see that if the reduction graph corresponds to a simple nested determinant polynomial $h = detpol(h_1, \ldots, h_p)$, then the graph consists of the reduction graphs (G_i, P_i) that describe the computation of the simple nested determinant polynomial h_i and the graph (G_0, P_0) which describes the triangularization of the matrix $mat(h_1, \ldots, h_p)$. This gives a decomposition of (G, P) having the properties mentioned above.

Conversely, if a reduction graph has the properties in the theorem, denote by h_i the label of the vertex v_i. Then, point 3 implies that $h_0 \simeq detpol(h_1, \ldots, h_p)$. Point 2, 4 and 5 imply that the h_i's for $i = 1, \ldots, p$ are similar to simple nested determinant polynomials of disjoint sets. Hence h_0 is similar to a simple nested determinant polynomial. □

We can now decide, by only graph theoretical considerations, if a polynomial f satisfies Conjecture 1 or if it satisfies sufficient conditions for Conjecture 2 or 3.

Remark 5 Let (G, P) be the reduction graph describing a Gröbner basis algorithm for computing f. Then, for checking if G satisfies the conditions of Theorem 4 or 5, only the condition of having no cycles (respectively of consisting of one tree) is essential. Once this condition is satisfied, we can, if necessary, change the orientations of some horizontal edges (v_1, v_2) corresponding to S-polynomials(and change the corresponding vertical edges (v_2, v_3) to (v_1, v_3)) such that all vertices have in-degree at most one. This amounts to changing in the Gröbner basis algorithm some S-polynomial computations of the form $Spol(f_i, f_j)$ to $Spol(f_j, f_i)$, which only changes the signs of some polynomials.

Let us see in more detail in which way we should change the orientation of some horizontal edges in each tree of G_u, $u \in T$ such as to obtain a graph with all vertices having in-degree at most one. Notice first that if an edge (v_1, v_2) was constructed for a polynomial reduction $ug_i \rightarrow_{f_j} uh_3$, then the vertex v_2 has in-degree 1 and out degree 0 in the graph G_u. This is because the label of v_2 is a multiple of a polynomial g_i which, being an intermediate result in the computation of a normal form, appears only once in the Gröbner basis computation. Now, we choose as root for the tree a vertex having in-degree 0 and we change the orientation of some edges such that all edges become oriented from the root towards the leaves. But the vertices (v_1, v_2) constructed for polynomial reductions have the property that v_2 is a leaf, hence they have already the correct orientation. All the edges that we reorient are therefore edges corresponding to S-polynomial computations.

The following example will show that Conjecture 3 is not true. Namely, there exists a Gröbner basis computation in which there exists an intermediate Gröbner basis polynomial such that its reduction graph has cycles of horizontal edges. Hence, it cannot be the graph of a matrix reduction. The idea for constructing this counterexample is to have in the basis three polynomials f, g, h such that each of the three S-polynomials $Spol(f,g), Spol(g,h), Spol(h,f)$ have to be computed and produce a non-zero normal form. Then, in the normal form computation of some other S-polynomial, all the three normal forms above are used, specifically $NF(Spol(f,g))$ is multiplied by $m/lcm(lt(f), lt(g))$, where $m = lcm(lt(f), lt(g), lt(h))$, $NF(Spol(g,h))$ is multiplied by $m/lcm(lt(g), lt(h))$ and $NF(Spol(h,f))$ by $m/lcm(lt(h), lt(f))$. This will yield a cycle in the subgraph corresponding to m.

Counterexample 3 We compute the Gröbner basis of the set of polynomials below, w.r.t. degree lexicographical order, $x > \overline{x} > \overline{\overline{x}} > y > \overline{y} > z > \overline{z} > t > \overline{t} > u > v > w > s > \overline{s}$.

The choice of the pair of polynomials whose S-polynomial will be processed next is made in increasing order of the least common multiple of the leading terms. In case of equality we choose the pair which contains the oldest element. If again we get more than one pair, we choose the pair whose other element is the oldest. We also use the product criterion and the chain criterion to avoid computing unnecessary S-polynomials.

The input basis:

$$\begin{aligned} f &= xt + \overline{x}\overline{t} \\ g &= yt + \overline{y}\overline{t} \end{aligned}$$

$$\begin{aligned}
h &= zt + \overline{z}\overline{t} \\
h_1 &= x\overline{y}\overline{t} + w \\
h_2 &= \overline{x}y\overline{t} \\
g_1 &= x\overline{z}\overline{t} + v \\
g_2 &= \overline{x}z\overline{t} \\
f_1 &= y\overline{z}\overline{t} + u \\
f_2 &= \overline{y}z\overline{t} \\
q_1 &= xy + us \\
q_2 &= x^2 + \overline{\overline{x}}\overline{s} \\
q_3 &= \overline{\overline{x}}y\overline{s} + yvs + zws + \overline{s}
\end{aligned}$$

The flow of the algorithm will be as follows:
$Spol(g,h) = zg - yh \to^*_{f_1,f_2} u =: \overline{f}$
$Spol(h,f) = xh - zf \to^*_{g_1,g_2} v =: \overline{g}$
$Spol(f,g) = yf - xg \to^*_{h_1,h_2} w =: \overline{h}$
$Spol(f,q_1) \to^* 0$
$Spol(g,q_1) \to^* 0$
$Spol(f,q_2) = x\overline{x}\overline{t} - \overline{\overline{x}}t\overline{s} =: p_1$
$Spol(q_1,q_2) = \overline{\overline{x}}y\overline{s} - xus \to_{q_3} xus + yvs + zws + \overline{s} \to_{xs\overline{f}} yvs + zws + \overline{s} \to_{ys\overline{g}}$
$zws + \overline{s} \to_{zs\overline{h}} \overline{s} =: p_2$
The graph corresponding to this intermediate polynomial p_2 has cycles of horizontal edges (see Figure 2.2).

Notice that if we would have chosen the S-polynomials in another order such that $Spol(f_1,g_1) = xu + yv$ would be already computed at this point and $Spol(g,h)$ not, then $xu + yv$ would be used in the reduction of $Spol(q_1,q_2)$ and there would be no cycles.

Remark 6

1. We saw that the counterexample for Conjecture 3 only works for a certain order of choosing the S-polynomials. It is not clear yet if given an input basis and an admissible term order we can always find a strategy such that we never get intermediate polynomials having cycles in the associated graphs, which therefore satisfy Conjecture 3.

2. Let us examine what kind of horizontal edges can produce cycles in a reduction graph. Assume in a reduction graph (G,P) there

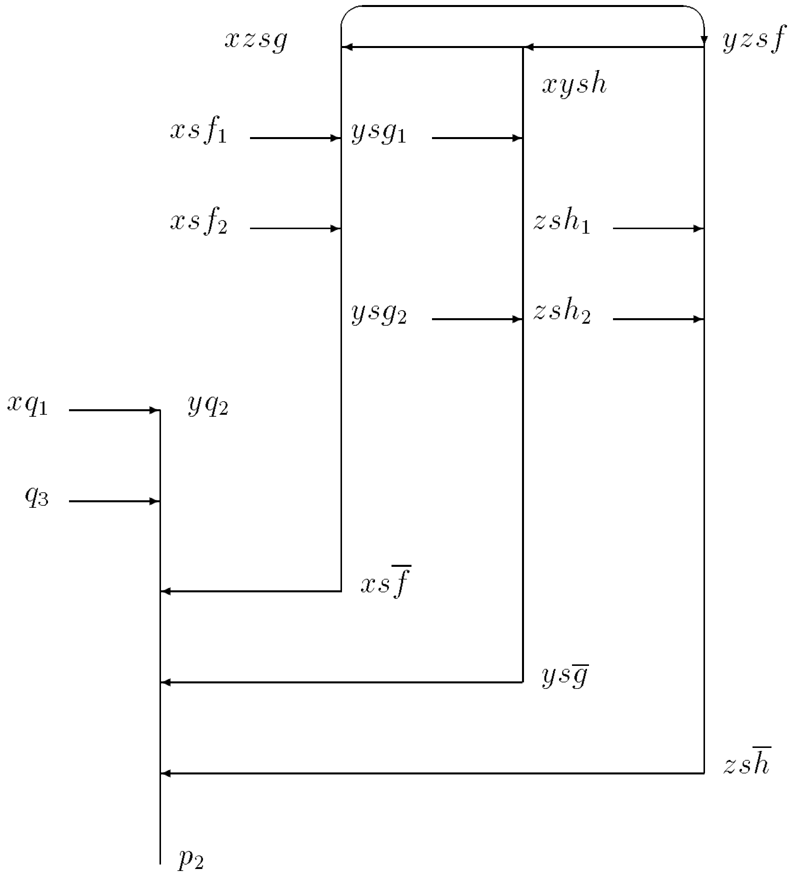


Figure 2.2: Reduction graph with cycles

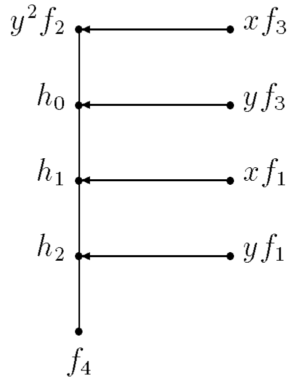
is a cycle in the subgraph G_u corresponding to the term $u \in T$. An edge describing a polynomial reduction cannot belong to a cycle because the degree of its end-point is one. (see the discussion in Remark 5). Hence all the edges in the cycle describe S-polynomial computations shifted by multiplication with a term, i.e. computations of the form $wSpol(f_i, f_j) = wh$. Moreover, we will show that at least one of these S-polynomial computations is actually shifted i.e. at least one w is different from 1. Let $w_1 Spol(f_{i_1}, f_{i_2})$, $w_2 Spol(f_{i_2}, f_{i_3}), \ldots$, $w_{r-1} Spol(f_{i_{r-1}}, f_{i_r})$,$w_r Spol(f_{i_r}, f_{i_1})$ be the S-polynomial computations in the cycle. We have $lcm(lt(f_{i_j}), lt(f_{i_{j+1}})) = u/w_j$, for $j = 1, \ldots, r$, with the convention $i_{r+1} = i_1$. If all w_j were equal to one, then the chain criterion would discard one of the S-polynomials above.

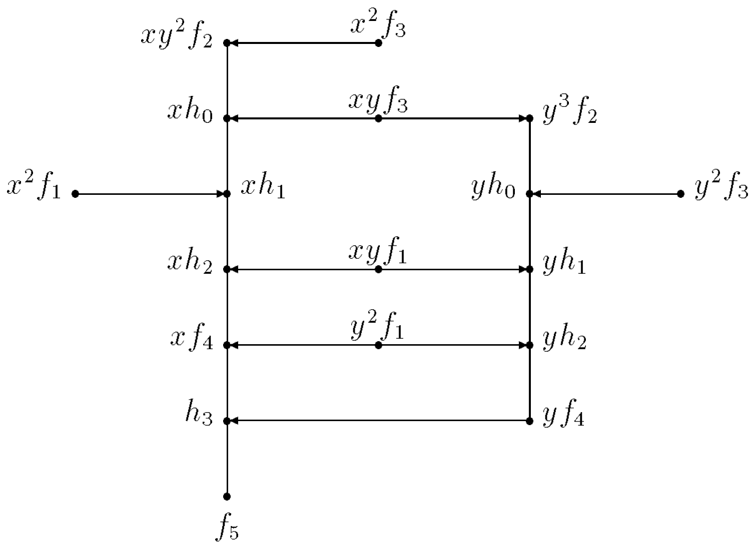
Now, having reduction graphs as tools, let us come back to the conjectures and counterexamples from previous sections. We will give an counterexample in the spirit of counterexample 2. This time we start from an explicit Gröbner basis computation in only 2 variables and 3 input polynomials. So we will prove that even in two variables the Gröbner basis computation does not always correspond to a determinant polynomial computation, and not even to a simple nested determinant polynomial.

Counterexample 4 We compute a Gröbner basis of $\{f_1, f_2, f_3\}$ with respect to degree lexicographical order, $x > y$. The S-polynomials are selected in increasing order of the least common multiple of their leading terms.

$$\begin{aligned} f_1 &= 5x^4y^6 - 7x^2y^8 - 6xy^9 \\ f_2 &= 7x^7y^2 + 7x^6y^3 - 5x^4y^5 \\ f_3 &= 3x^6y^4 - 5x^5y^5 + 11x^4y^6 \end{aligned}$$

The computation will proceed as follows:
$Spol(f_2, f_3) = red(xf_3, y^2 f_2) := h_0 \rightarrow_{yf_3} h_1 \rightarrow_{xf_1} h_2 \rightarrow_{yf_1} f_4$ and $lt(f_4) = x^3y^8$
$Spol(f_4, f_1) = red(y^2 f_1, xf_4) := h_3 \rightarrow_{yf_4} f_5$ and $lt(f_5) = x^2y^{10}$
The reduction graphs of f_4 and f_5 are shown in the next figure and the leading terms in Figure 2.3.



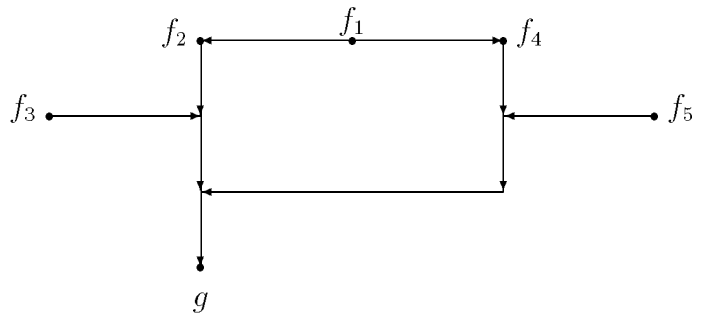


The graph of f_5 does not have the properties required for a graph corresponding to a determinant polynomial computation, because the subgraph corresponding to $lt(x^2f_1)$ does not consist of one but of two disjoint trees, namely the one consisting of the edge (x^2f_1, xh_1) and the one consisting of the edge (y^2f_3, yh_0).

We prove that the graph G describing the Gröbner basis algorithm for computing f_5 does not have the properties required for the computation of a simple nested determinant polynomial either. If we assume it does, from Theorem 6 we see that the two edges (x^2f_1, xh_1) and

(y^2f_3, yh_0) should belong to two disjoint subgraphs of G each of which correspond to the computation of some simple nested determinant polynomial. Then also the edges (xyf_3, xh_0) and (xyf_3, y^3f_2) should belong to these two disjoint subgraphs, which is not true since they have the vertex labeled xyf_3 in common.

The same problem appears in any reduction graph for Counterexample 2. One of the possible reduction graphs for the computation of the nested determinant polynomial described there is shown in the next picture.



As for the size of coefficients of polynomials that can be obtained by Gaussian elimination, in general we can give only a doubly exponential bound and not a simple exponential one like in the case of determinant polynomials or simple nested determinant polynomials. More precisely, we have the bound that can be obtained straightforwardly by bounding at each step of the computation the entries of the rows of the matrix. That is, if the row f_i has integer entries bounded by d_i, for $i = 1, 2$, then by reducing the row f_2 using the row f_1 we obtain a row $red(f_1, f_2)$ with entries bounded by $2d_1d_2$. In this way we may obtain in general bounds of the order d^{2^r}, where r is the number of rows and d a bound on the norm of the entries of the initial matrix. See section 3.4 for further discussion of the issue.

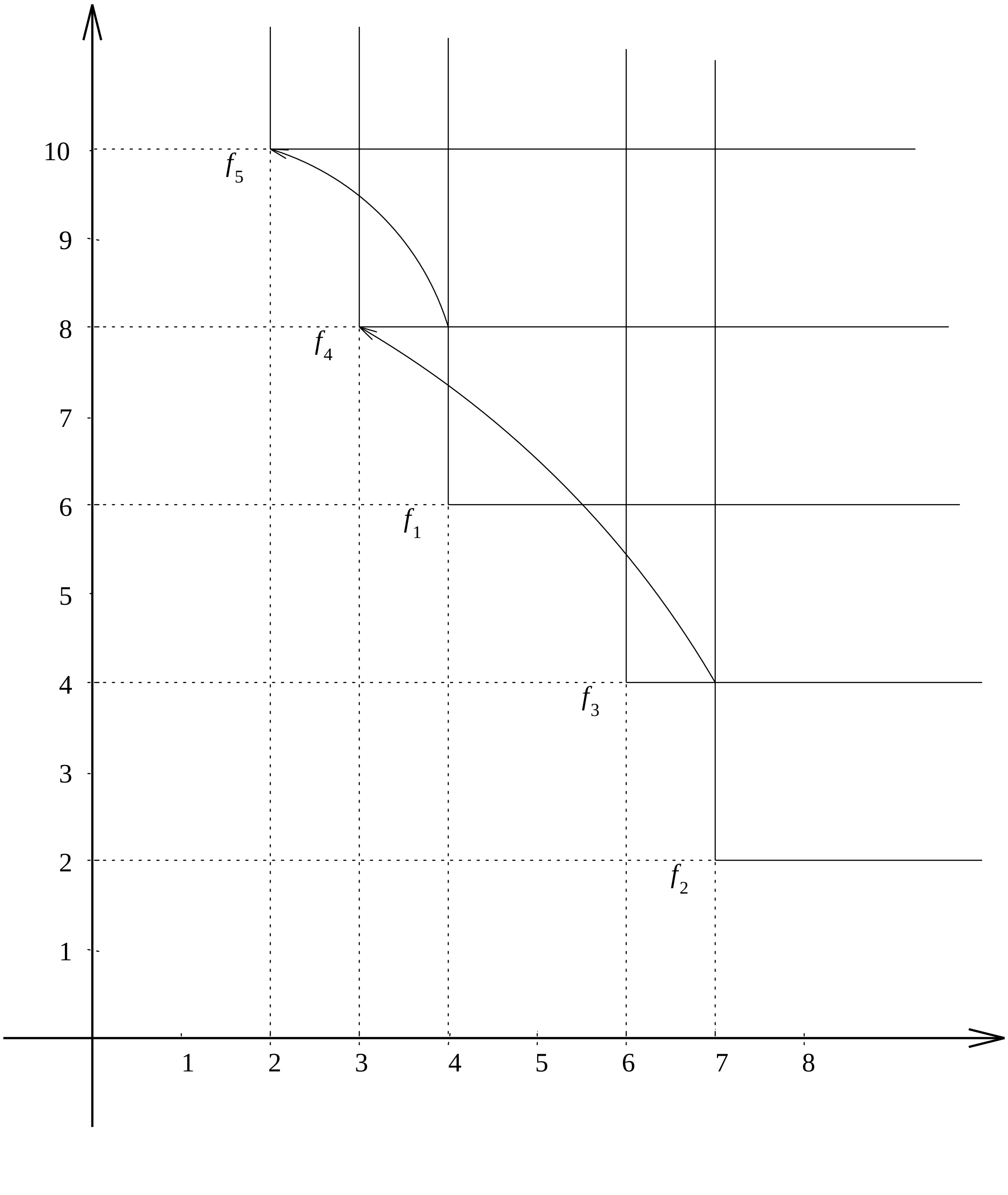


Figure 2.3:

Chapter 3

Generalized Fraction-free Gaussian Elimination and Applications

3.1 Generalized Fraction-free Gaussian Elimination

We recall here the fraction-free Gaussian elimination method, reformulating it slightly for the purpose of generalization we are interested in. Usually the proof is based on Sylvester's determinant identity. We give here a direct proof (based on the technique used in [Col66, Col67, Loo83]). Actually the proof is very similar to the proof of the fundamental theorem of subresultants, the difference being that there we use more pivots for each column, whereas here we use only one.

Theorem 7 (Fraction-free Gaussian elimination)
Let $M = (m_{i,j})_{i\in\{1,\ldots,k\},j\in A}$ be a matrix with indeterminate entries, $A = \{t_1, \ldots, t_l\} \subseteq T, t_i > t_{i+1}$ and $k \leq l$. Let $M' = (m'_{i,j})$ be an equivalent triangular matrix, and h the polynomial corresponding to the row of M' of minimum leading term.

- Assume M' is obtained from M by classical Gaussian elimination (that is, by reducing the rows $2, 3, \ldots, k$ by the row 1, then the rows $3, \ldots, k$ by the row 2 and so on). Then:

$$h = detpol(M) \cdot (m'_{1,t_1})^{k-2} \cdot (m'_{2,t_2})^{k-3} \cdot \ldots \cdot (m'_{k-2,t_{k-2}})^1.$$

- Assume M' is obtained by row reductions as above, but additionally, after reducing the rows $i+1, \ldots, k$ using row i, we divide each of them by $m'_{i-1,t_{i-1}}$ (with the convention $m'_{0,t_0} = 1$). Then all divisions are exact and $h = detpol(M)$.

Proof Let $M^{(1)} = M$ and for $i = 1, \ldots, k-1$ define inductively $M^{(i+1)} = (m^{(i+1)}_{p,j})$ as the matrix obtained from $M^{(i)}$ by reducing the rows $i+1, \ldots, k$ by the row i and then dividing them by an element e_i. (We will have $e_i = 1$ for the first part of the theorem and $e_i = m'_{i-1,t_{i-1}}$ for the second.) When we reduce a row h_1 by a row h_2 we modify the determinant polynomial of the matrix by multiplying it by the leading coefficient of the row h_2. We have thus $detpol(M^{(i+1)}) = detpol(M^{(i)})(m^{(i)}_{i,t_i}/e_i)^{k-i} = detpol(M^{(1)}) \prod_{j=1}^{i} (m^{(j)}_{j,t_j}/e_j)^{k-j}$. The first i rows of $M^{(i)}$ coincide with the first i rows of M'. Hence for $i = k-1$ we have:

$$detpol(M^{(k)}) = detpol(M^{(1)}) \prod_{j=1}^{k-1} (m'_{j,t_j}/e_j)^{k-j} \tag{3.1}$$

But $M^{(k)}$ is exactly the matrix M' which is triangular and its determinant polynomial can be easily evaluated: $detpol(M^{(k)}) = m^{(k)}_k \prod_{j=1}^{k-1} m'_{j,t_j}$, where $m^{(k)}_k$ denotes the k-th row of $M^{(k)}$. Using this relation, the equality (3.1) becomes

$$m^{(k)}_k = detpol(M^{(1)}) \prod_{j=1}^{k-2} (m'_{j,t_j})^{k-j-1} / \prod_{j=1}^{k-1} (e_j)^{k-j} \tag{3.2}$$

The first point of the theorem results now immediately by putting $e_j = 1$ for $j = 1, \ldots, k-1$ in the previous relation.

For the second point we prove that for $i \geq 1$ the rows of $M^{(i)}$ do not contain fractions, which means that all divisions are exact.

For $i = 1$ the result is trivially true. Let $p \in \{i+1, \ldots, k\}$. We will prove that the p-th row of $M^{(i+1)}$ does not contain fractions. Let B be the submatrix of M consisting of the first i rows and the p-th row. If we triangularize B the same way as we triangularize M, i.e. by doing a classical Gaussian elimination and dividing at each step j all the rows of index greater than j by e_j, we obtain a sequence of matrices $(B^{(j)})_{j=1,\ldots,i+1}$, with the property that $B^{(j)}$ coincides with the submatrix of $M^{(j)}$ consisting of the first i rows and the p-th row. We write the identity (3.2) for B taking into account that B has $i+1$ rows and $e_j = m'_{j-1,t_{j-1}}$.

$$\begin{aligned} b_{i+1}^{(i+1)} &= detpol(B^{(1)}) \prod_{j=1}^{i-1} (m'_{j,t_j})^{i-j} / \prod_{j=1}^{i} (e_j)^{i+1-j} = \\ &= detpol(B) \prod_{j=1}^{i-1} (m'_{j,t_j})^{i-j} / \prod_{j=1}^{i} (m'_{j-1,t_{j-1}})^{i+1-j} = \\ &= detpol(B) \end{aligned}$$

But $m_p^{(i+1)}$, the p-th row of $M^{(i+1)}$ coincides with $b_{i+1}^{(i+1)}$, the $(i+1)$-th row of $B^{(i+1)}$. This row, being equal to $detpol(B)$, does not contain fractions. In particular, for $i = k-1$ and $p = k$ we get $h = m_k^{(k)} = detpol(M)$, since B coincides in this case with M. $\square$

A matrix can be triangularised by many different sequences of row reductions. Moreover, when the matrix entries are not indeterminates, we might get during the computation some zero entries which make certain row reductions superfluous or impossible. We will generalize therefore the previous theorem to include these situations.

Theorem 8 (Generalized fraction-free Gaussian elimination)

Let $M = (m_{i,j})_{i\in\{1,\ldots,k\},j\in A}$ be a matrix over R, and M' a triangular matrix obtained from M by a sequence of row reductions described by the reduction graph (G, P), with $G = (V, E)$. Let h be the polynomial corresponding to the row of M' with smallest leading term, and let v_0 be the unique vertex of G labeled by this polynomial. Then:

$$h = detpol(M) \cdot \prod_{v\in V\setminus\{v_0\}} (lc(label(v)))^{vdeg(v)-1}$$

where we denoted by $vdeg(v)$ the number of horizontal edges having v as one of the extremities.

Proof Each horizontal edge (v_1, v_2) of the graph G describes the reduction of the row $label(v_2)$ by the row $label(v_1)$, operation which multiplies the determinant polynomial of the matrix by $lc(label(v_1))$. Hence

$$detpol(M') = detpol(M) \cdot \prod_{v \in V} (lc(label(v)))^{out\_deg(v)} \tag{3.3}$$

where we denote by $out\_deg(v)$ the number of horizontal edges having v as start-point.

All horizontal edges having a vertex v as one of the extremities belong to the graph $G_{lt(label(v))}$.

From theorem 5 we know $G_{lt(label(v))}$ is a tree and all vertices have exactly one horizontal incoming edge except for the root of the tree which has none. Hence, $out\_deg(v) = vdeg(v) - 1$ for all vertices except the root for which $out\_deg(v) = vdeg(v)$. The label of the root v' of the tree $G_{lt(label(v))}$ is the only row which has the leading term $lt(label(v)))$ and is not reduced by other rows, hence it is the only row of M' having the leading term $lt(label(v)))$. Since M' is triangular we have

$$detpol(M') = h \cdot \prod_{\substack{v \in V \setminus \{v_0\} \\ v \ root \ of \ G_{lt(label(v))}}} lc(label(v)). \tag{3.4}$$

Equating the two expressions for $detpol(M')$ from (3.3) and (3.4) we get the identity in the theorem. □

If during a generalized Gaussian elimination we detect that some row f has a common factor c of its entries, then we would like to divide out this factor and use f/c in the subsequent computations. We will generalize Theorem 8 in order to deal with this situation.

We will represent a polynomial (or a matrix row) f as a pair $f = (c, \overline{f})$ where $f = c \cdot \overline{f}$ and $c \in R$, $\overline{f} \in R[x_1, \ldots, x_n]$. Since we want to use only $\overline{f}$ in further computations, we will define the reduction of such polynomials differently: $red((c_1, \overline{f_1}), (c_2, \overline{f_2})) = (c, \overline{f})$ where c and $\overline{f}$ are such that $c \cdot \overline{f} = lc(\overline{f_1}) \cdot \overline{f_2} - lc(\overline{f_2}) \cdot \overline{f_1}$. (Notice that c and $\overline{f}$ are not uniquely defined.) When we use this representation we will assume that as soon as we obtain a new polynomial f, we compute (by some method) a constant factor c of f and divide it out immediately, using $\overline{f} = f/c$ in the subsequent computations.

Theorem 9 Assume we are in the hypothesis of Theorem 8 and we use the representation above for polynomials. Then:

$$h = detpol(M) \cdot \prod_{\substack{v \in V \setminus \{v_0\} \\ label(v)=(c,\overline{g})}} ((lc(\overline{g}))^{vdeg(v)-1}/c)$$

Proof We follow the proof of Theorem 8. After computing a row g and detecting that it has a constant factor c, we divide the row by c. In this way the determinant polynomial of the matrix is also divided by c. The formula (3.3) becomes then

$$detpol(M') = detpol(M) \cdot \prod_{\substack{v \in V \\ label(v)=(c,\overline{g})}} ((lc(\overline{g}))^{out\_deg(v)}/c)$$

$\square$

Remark 7 Until now we only reduced one matrix row by another in such a way as to cancel the leading term of the first one using the leading term of the second. Let us examine the reductions that cancel some other entry of a row using the leading term of another row. Such reductions appear when we want to bring a matrix to a diagonal form rather than a triangular one.

We will see that Theorem 8 still holds.

First, we extend the definition of the operation $red(f,g)$ to the case where $lt(f) \leq lt(g)$ and the coefficient c of $lt(f)$ in g is not zero. We put $red(f,g) = lc(f)g - cf$. We leave unchanged the definition of reduction graphs, but notice that now for horizontal edges the leading term of the label of the start-point will be smaller or equal to the leading term of the label of the end-point (and not necessarily equal, as it was before). For horizontal edges, the leading term of the label of the start-point will be greater or equal than the one of the end-point.

To prove Theorem 8, note first that formula (3.3) still holds. When we reduce a row g by a row f we replace the row g by $red(f,g)$, hence g will not be available any more. This means that in a reduction graph describing a sequence of row reductions in a matrix, any vertex can have at most one horizontal in-coming edge. The vertices that have none correspond to rows that are never reduced and appear therefore in the final matrix. Since the final matrix is triangular we have

$$detpol(M') = h \cdot \prod_{\substack{v \in V \setminus \{v_0\} \\ in\_deg(v)=0}} lc(label(v)).$$

where $in\_deg(v)$ denotes here the number of horizontal edges having v as end-point. Combining the relation above with (3.3) and using the fact that $vdeg(v) = in\_deg(v) + out\_deg(v)$ we obtain the relation from Theorem 8.

3.2 Intrinsic Constant Factors

During a polynomial remainder sequence computation and also during the Gröbner basis computation, two kinds of constant factors of polynomials appear: intrinsic ones (which appear also when the coefficients are indeterminates) and accidental ones (that are due to particular relations between the coefficients of the polynomials in the input basis). A subresultant-like theory can only detect the first kind of factors.

Let us define more precisely what we will mean by "intrinsic constant factors". Consider an algorithm that starts from a sequence of polynomials $f_1, \ldots, f_m$ and constructs a sequence of polynomials $f_{m+1}, \ldots, f_l$, such that each polynomial $f_i, i > m$ is obtained either by multiplying with a term a polynomial f_j with $j < i$, or by reducing two polynomials f_j, f_k with $lt(f_j) = lt(f_k)$ and $j, k < i$. This type of algorithm includes the polynomial remainder sequence computation, the Gröbner basis algorithm and the row reductions in a matrix. Constant factors that appear "in general" in the polynomials $f_i, i > m$ can be defined only if we use indeterminate coefficients for the initial polynomials. Assume therefore that the initial polynomials are not neccessarily dense, but their non-zero coefficients are indeterminates, not necessarily distinct. Let V be the set of these indeterminates. Then an *intrinsic constant factor* of a polynomial f_i will be a factor of f_i belonging to $\mathbb{Z}[V]$. We can define "the" intrinsic constant factor of f_i as being the content of f_i (considering f_i as a polynomial with coefficients in $\mathbb{Z}[V]$).

The reductions that we can perform during the algorithm are determined by the leading terms of the polynomials. For certain specialisations of the coefficients, it may happen that the leading coefficient of a polynomial vanishes, and hence its leading term is modified. To include these cases we can consider the coefficients of the input polynomials as being elements in $\mathbb{Z}[V]/J$, where J is the ideal generated by the leading coefficients that vanish during the algorithm. In this case we can still define "a" intrinsic constant factor of f_i as being a common factor of all coefficients of f_i, but we cannot define "the" intrinsic constant factor, because the greatest common divisor need not exist in $\mathbb{Z}[V]/J$. For example, in $\mathbb{Z}[a_1, a_2, b_1, b_2, c_1, c_2]/(a_1b_1 - a_2b_2, a_1c_1 - a_2c_2)$, the two elements a_1b_1 and a_1c_1 have the common divisors a_1 and a_2 but not a_1a_2, and they do not have a greatest common divisor.

For classical and for arbitrary Gaussian elimination, if all the entries of the initial matrix are distinct indeterminates, then Theorem 7 respectively Theorem 8, give the intrinsic constant factor of the row of lowest leading term in the final triangular matrix. This results immediately from the fact that, as shown in the next theorem, the determinant polynomial of the initial matrix is in this case primitive.

Theorem 10 Let $M = (m_{i,j})_{i\in\{1,\ldots,k\},j\in A}$ with $A \subset T$ be a matrix having as entries distinct indeterminates and let $V = \{m_{i,j} | i \in \{1,\ldots,k\}, j \in A\}$. Then $detpol(M)$ is primitive (as polynomial in $x_1,\ldots,x_n$ with coefficients in $\mathbb{Z}[V]$).

Proof We will prove a more general fact, namely that any two minors of M are relatively prime, as polynomials in $\mathbb{Z}[V]$. The assertion in the theorem will follow then easily, since the coefficients of $detpol(M)$ are distinct minors of M.

Let M_1, M_2 be two distinct square submatrices of M, of order k_1 and k_2, respectively. Let $g = \gcd(det(M_1), det(M_2))$ and assume $k_1 \leq k_2$. We will prove by induction on k_2 that $g = 1$.

If $k_1 = k_2 = 1$, the assertion is obvious because $det(M_1)$ and $det(M_2)$ are just two distinct indeterminates in V.

Assume we proved the assertion for minors of order at most $k_2 - 1$. Since M_1 and M_2 are distinct submatrices of M, and M_2 has the order greater or equal to the order of M_1, there must exist either a row or a column of M_2 such that none of its entries is an entry of M_1. Let V_2 be the set of indeterminates that are entries of that row or column of

M_2. The indeterminates in V_2 do not appear in $det(M_1)$, hence they do not appear in g either. This means g is a common divisor of the coefficients of $det(M_2)$ viewed as polynomial in the variables V_2. But if we expand the determinant of M_2 along the above mentioned row or column, we see that these coefficients are minors of M of order $k_2 - 1$, hence, according to the induction hypothesis they are relatively prime. We proved thus that $g = 1$. $\square$

3.3 Detection of constant factors of Gröbner basis polynomials

We will use now the generalized fraction-free Gaussian elimination for detecting certain intrinsic constant factors of the polynomials obtained during the Gröbner basis algorithm. We assume that for each polynomial introduced in the basis we construct the corresponding reduction graph.

Theorem 11 Let f be a Gröbner basis polynomial and (G, P) the reduction graph describing the Gröbner basis algorithm for computing f. Assume (G, P) coincides with a reduction graph describing the row reductions performed on the matrix $mat(prereq(f))$ for computing (up to similaity) a determinant polynomial or simple nested determinant polynomial $\overline{f}$ of the polynomials labeling the vertices in $input(G)$. Let v_0 be the unique vertex of G labeled by f. Then:

$$f = \overline{f} \cdot \prod_{v \in V \setminus \{v_0\}} (lc(label(v)))^{vdeg(v)-1}$$

and all polynomials $label(v)$ appearing in the product above are polynomials introduced in the basis before f.

Proof If (G, P) is the reduction graph describing the computation of a determinant polynomial, i.e. a matrix triangularisation, then Theorem 8 gives immediately the formula we are looking for.

If the reduction graph describes the computation of a simple nested determinant polynomial, then we apply Theorem 8 recursively, by induction on the structure of the reduction graph described in Theorem 6. We know (G, P) can be decomposed into reduction graphs

$(G_0, P_0), (G_1, P_1), \ldots, (G_p, P_p)$, satisfying the properties in Theorem 6. Each of these reduction graphs has a unique vertex v_i whose label has minimum leading term among all labels of vertices of G_i. The label of v_0 is f, and for each $i = 1, \ldots, p$, denote by h_i the label of v_i. Each h_i is similar to a simple nested determinant polynomial $\overline{h_i}$ of the labels of $input(G_i)$.

Since (G_0, P_0) is the reduction graph of a matrix triangularisation, we have by Theorem 8:

$$\begin{aligned} f &= detpol(\{label(v) | v \in input(G_0)\}) \prod_{v \in V_0 \setminus \{v_0\}} (lc(label(v)))^{vdeg(v)-1} \\ &= detpol(h_1, \ldots, h_p) \prod_{v \in V_0 \setminus \{v_0\}} (lc(label(v)))^{vdeg(v)-1} \end{aligned}$$

By induction, for each $i = 1, \ldots, p$ we have

$$h_i = \overline{h_i} \prod_{v \in V_i \setminus \{v_i\}} (lc(label(v)))^{vdeg(v)-1}$$

Substituting these expressions for h_i in the previous equalities we get:

$$\begin{aligned} f &= detpol(\overline{h_1}, \ldots, \overline{h_p}) (\prod_{i=1}^{p} \prod_{v \in V_i \setminus \{v_i\}} (lc(label(v)))^{vdeg(v)-1}) \cdot \\ &\quad \cdot \prod_{v \in V_0 \setminus \{v_0\}} (lc(label(v)))^{vdeg(v)-1} = \\ &= \overline{f} \prod_{v \in V \setminus \{v_0\}} (lc(label(v)))^{vdeg(v)-1} \end{aligned}$$

We used the linearity of $detpol$ and the fact that $V \setminus \{v_0\} = \cup_{i=0}^{p} V_i \setminus \{v_i\}$ and the sets $V_i \setminus \{v_i\}$ are pairwise disjoint, for $i = 0, \ldots, p$. The simple nested determinant polynomial $detpol(\overline{h_1}, \ldots, \overline{h_p})$ is exactly the simple nested determinant polynomial whose computation is described by the reduction graph (G, P), hence it coincides with $\overline{f}$.

Let us prove now that all the polynomials $label(v)$ appearing in the formula in the theorem are indeed polynomials already adjoined to the basis. Let g be a polynomial that appears in the algorithm but is not introduced in the basis. Then g must be a polynomial obtained at some intermediate step of a normal form computation. This polynomial is used only once, hence the corresponding vertex is extremity for exactly one horizontal edge. Therefore the exponent of $lc(g)$ in the formula is 0. □

If during the Gröbner basis algorithm we divide out constant factors of the polynomials we introduce in the basis, then we need to represent the polynomials as discussed at the end of section 3.1, namely as pairs $g = (c, \overline{g})$, where $g = c\overline{g}$ and $c \in R$. With this representation of polynomials, and assuming we divide each polynomial $g = c\overline{g}$ by c as soon as g is computed, the formula in the theorem above will become:

$$f = \overline{f} \cdot \prod_{\substack{v \in V \setminus \{v_0\} \\ label(v) = (c, \overline{g})}} ((lc(\overline{g}))^{vdeg(v)-1}/c)$$

The proof is similar, but uses Theorem 9 instead of Theorem 8.

We see thus that once we know the reduction graph of a Gröbner basis polynomial that can be written as a determinant polynomial or as a simple nested determinant polynomial, it is relatively easy to compute an intrinsic constant factor. Now we examine the case when the reduction graph does not have any particular property, i.e. it need not correspond to the computation of a determinant polynomial or simple nested determinant polynomial.

A reduction graph (G', P') will be called a subgraph of the reduction graph (G, P) if G' is a subgraph of G and $P' \subseteq P$.

Corollary 2 Let f be an intermediate Gröbner basis polynomial and (G, P) its reduction graph. Consider (G', P') a subgraph of (G, P) such that (G', P') is a reduction graph corresponding to the computation of a determinant polynomial or simple nested determinant polynomial and f is the unique label of least leading term among all labels of vertices of G'. Denote by $c_{G'}$ the intrinsic constant factor computed for f and G' using Theorem 11. Then the least common multiple of the $c_{G'}$, over all the subgraphs G' having the properties above, divides f.

So for the general case it becomes quite costly to compute an intrinsic constant factor by the method above. (Computing *all* subgraphs is an algorithm of exponential complexity.) Of course we could compute only a few $c_{G'}$'s and their least common multiple will still be a constant factor of f, but it may be a smaller one.

Summarizing, we can modify the Gröbner basis algorithm such that we detect certain intrinsic constant factors of the polynomials. Namely, we keep track of the reductions graphs of the polynomials introduced in the basis. For each new polynomial we can decide by graph theoretical considerations, using Theorems 5 and 6 if its reduction graph corresponds or not to the computation of a determinant polynomial or a simple nested determinant polynomial. In the affirmative case, we compute an intrinsic constant factor using Theorem 11. Otherwise, we compute it by Corollary 2. It is not clear if these are all the intrinsic constant factors that can appear, and it seems very difficult to prove such a result.

The algorithm above is for the moment rather of theoretical interest than of practical one. Beside the overhead of keeping track of the reduction graphs, the computation of the intrinsic constant factors becomes very inefficient when we are not in the favourable case of a determinant polynomial or simple nested determinant polynomial. Hence we think that for the moment this algorithm cannot compete with the direct computation of the content by greatest common divisor computation. Further developments may bring these ideas to an efficiently implementable algorithm.

The fact that constant factors appear systematically during the Gröbner basis computation and that these factors are products of leading coefficients of previously computed Gröbner basis polynomials, was observed in practice and heuristically used in [GPT88]. An explanation for the reason why these factors appear and which exactly is their structure is asked for in the aforementioned paper. Our results answer this question (at least partially, since we are not able to prove that this method detects all the intrinsic factors).

Remark 8 If during the Gröbner basis algorithm we perform not only head reductions but also total reductions, we can still prove the same kind of results concerning the intrinsic constant factors. As we saw in Remark 7, Theorem 8 is still valid for total reductions, so we can directly apply it for Gröbner basis polynomials that can be expressed as determinant polynomials or simple nested determinant polynomials. Otherwise we can apply it as in Corollary 2.

The only additional problem is that in the case of total reductions, to characterise by graph-theoretical properties the reduction graphs corresponding to Gröbner basis computation, sequences of row reductions, matrix triangularisation becomes more intricate than it was for head reductions. We will not enter here in details.

3.4 Classical versus Generalized Fraction-free Gaussian Elimination

We will compare now the classical Gaussian elimination and the generalised Gaussian elimination from the point of view of the growth of the entries (which crucially influences the efficiency of the algorithm).

Assume $M = (m_{i,j})_{i\in\{1,\dots,k\},j\in A}$ is a matrix with distinct indeterminate entries and $A = \{t_1,\dots,t_l\} \subset T$, with $t_i > t_{i+1}$ and $k \leq l$. We will use this matrix as input both for the classical Gaussian elimination algorithm and the generalised Gaussian elimination algorithm.

If we do not divide out common factors of the entries of a row, then for both algorithms the entries will have the same growth rate. Namely, assume that for $i = 1, 2, \dots$ we reduce at step i all but one of the rows having the leading term t_i. We use one pivot for the first algorithm and possibly more pivots for the second. Then in both algorithms the entries of the rows computed at step i will be homogeneous polynomials of degree 2^i in the indeterminates. Hence, if we specialise the indeterminates to integers of norm d, entries of norm d^{2^i} are to be expected after the i-th step.

But, as we saw in section 3.1, intrinsic constant factors appear systematically during the triangularisation and can be divided out in order to slow down the growth of the entries. For the classical Gaussian elimination this is easy, because at each step the factor that appears is just the pivot of the previous step hence we do not need any computational effort to determine it. By dividing out systematically this factor, the entries after step i will have degree $i+1$, hence when specializing to integers of norm d only a growth of order d^{i+1} can be expected.

For the generalised Gaussian elimination, the intrinsic constant factors are harder to compute. Theorem 8 gives for the final triangular matrix the intrinsic constant factor of the row of lowest leading term. But what we want now is to determine intrinsic constant factors that appear during the algorithm. Let g be a row obtained at some intermediate step of the triangularisation. From the reduction graph (G, P) describing all row reductions performed so far, we will extract the subgraph describing those row reductions which were actually needed for obtaining g. If g is a row of the initial matrix M, then no row reductions are needed. If g was obtained by reducing the row g_1 by the row g_2, then the reductions needed for obtaining g will be the reductions needed for obtainig g_1, those needed for obtaining g_2, and the reduction of g_1 by g_2. The reduction graph (G', P') describing these row reductions describes, obviously, a sequence of row reductions performed on the submatrix $\overline{M} = mat(label(input(G'))$ of M. We can decide if this graph describes a matrix triangularisation, either using Theorem 5, or checking if $\overline{M}$ has become triangular at this stage of the computation and g is its row of lowest leading term. If this is the case then we can detect an intrinsic constant factor of g applying Theorem 8 for the matrix $\overline{M}$ and the reduction graph (G', P'). Otherwise we can, like in Corollary 2, construct all reduction graphs (G'', P'') which are subgraphs of (G, P), describe a matrix triangularisation, and g is the unique label of a vertex of G'' having lowest leading term. For each such graph we compute an intrinsic constant factor of g, and the least common multiple of these factors will be a factor of g.

Hence, in general it is difficult to compute intrinsic constant factors appearing at intermediate steps of an arbitrary Gaussian elimination. Notwithstanding this difficulty, let us see now if by dividing out these factors the size of the entries will grow as slow as for the classical fraction-free gaussian elimination.

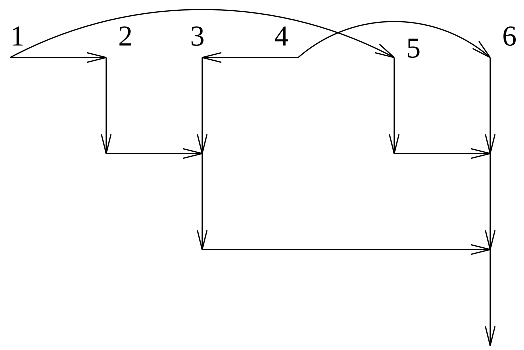
The following example shows that this is not necessarily so. Namely, a triangularization by an arbitrary sequence of row reductions can produce much bigger intermediate growth of the entries than the classical Gaussian elimination. This is due to the fact that, whereas in the classical Gaussian elimination we obtain at each step a constant factor that can be divided out, for an arbitrary Gaussian elimination it may happen that the factors appear only at later steps. This example is a generalisation of the construction from Counterexample 2.

Example 3 Assume the matrix $M = (m_{i,j})_{i\in\{1,\ldots,k\},j\in A}$ with indeterminate entries has an even number of rows $k = 2p$.

We perform row reductions in the matrix M in p steps in the following way: at the i-th step, for $i < p-1$, the row i reduces the rows $i+1$ and $p+i+1$ and the row $p+i$ reduces the rows $i+2,\ldots,p$ and $p+i+2,\ldots,2p$; at step $p-1$ the row $p-1$ reduces the row p and the row $2p-1$ reduces the row $2p$; finally, at step p the row p reduces the row $2p$. We will see later that indeed it is possible to perform this sequence of reductions, that is, before the i-th step the rows $i, i+1,\ldots,p$ and $p+i, p+i+1,\ldots,2p$ have all the same leading term, namely t_i.

Roughly speaking, we have at each step two pivots. The row of one of them reduces the two rows on which the pivots will be at the next step, the row of the other pivot reduces all the other rows. The rows of the pivots themselves are not reduced.

The reduction graph (G, P) describing these reductions has the property that each subgraph corresponding to a term $t_i, i \in \{1,\ldots,p-1\}$ consists of two disjoint trees, (having the roots labeled by the rows of the two pivots used at step i). For $p = 3$ the reduction graph is shown in the next picture.



After these reductions the matrix is not yet triangular, more precisely it has a row of leading term t_{p+1}, one row of leading term t_p, and for each $i \in \{1,\ldots,p-1\}$ two rows with leading term t_i. The sequence of reductions above may be continued afterwards in an arbitrary way, until we reach a triangular matrix. We will examine now what is the situation at this very stage of the triangularization.

We will prove that the row $2p$, which we modified last and has minimum leading term, has no intrinsic constant factor detectable by our method.

Assume there exists such a factor. Then there exists a subgraph (G', P') of (G, P) with the property that (G', P') is the reduction graph of a determinant polynomial computation and the constant factor computed by Theorem 8 for this graph is not 1. This means that G' has at least one vertex v which is extremity for at least two horizontal edges.

By Theorem 5 each subgraph of G' corresponding to a term consists of exactly one tree. Let $t_i = lt(label(v))$ and G_{t_i}, G'_{t_i} be the subgraphs of G, respectively G' corresponding to the term t_i. Only two vertices of G_{t_i} are extremities of more than one edge namely the vertex corresponding to the row i and the vertex corresponding to the row $p+i$. Hence v can only be one of these vertices.

Case 1. The vertex v corresponds to the row i. The row i reduces the rows $i+1$ and $p+i+1$. At the next step, $i+1$, these two rows are the rows of the pivots and the corresponding vertices are therefore the roots of to two disjoint trees of horizontal edges, which is impossible.

Case 2. The vertex v corresponds to the row $p+i$. At least two horizontal edges $(v, v_1), (v, v_2)$ belong to G'. The vertices v_1 and v_2 are labeled by two rows of the current matrix. If one of these rows becomes the row of a pivot at an earlier step (say at step j) than the other, then at step $j-1$ the two rows are reduced by different rows hence the corresponding vertices belong to disjoint trees of horizontal edges, which is impossible. If the two rows are the rows of pivots at the same step j, then the corresponding vertices are the roots of two disjoint trees, which is, again, impossible.

Now let us examine the growth of the entries. Since we start with a matrix with distinct indeterminate entries, any sequence of row reductions produces only entries which are homogeneous polynomials (in the indeterminates). All non-zero entries of a row have the same degree. For our sequence of reductions, the row $2p$ has at least one non-zero entry and the degree of the non-zero entries is 2^p. If we specialize all indeterminates to integers of norm d, we will get (for certain specializations) entries of norm d^{2^p} after the p-th step, hence a doubly exponential growth, compared to a simple exponential growth for the classical fraction-free Gaussian elimination.

One technical detail is left to prove, namely that before the i-th step the rows $i, i+1, \ldots, p$ and $p+i, p+i+1, \ldots, 2p$ have the leading term t_i. We will prove a more general result, namely that no matter what sequence of reductions we perform on the matrix M with indeterminate entries, after reducing a row having the leading term t_i by another row having the same leading term, the result will have the maximum possible leading term, that is t_{i+1}. Let $M = M^{(1)}, M^{(2)}, \ldots, M^{(q)}$ be the sequence of matrices obtained by this sequence of reductions, and assume that in $M^{(q-1)}$ we reduce two rows with the leading term t_i. Without loss of generality we can assume we reduce the first row by the second. Hence $m_1^{(q)} = red(m_2^{(q-1)}, m_1^{(q-1)})$. We pove that $lt(m_1^{(q)}) = t_{i+1}$. Assume that $lt(m_1^{(q)}) = t_j$, with $j > i+1$. Consider then the submatrix P of the matrix M, consisting of the columns $t_1, \ldots, t_i, t_{i+1}$ of M and the submatrix Q consisting of the columns $t_1, \ldots, t_i, t_j$ of M. We perform on both submatrices the same sequence of row reductions that we performed on M, and let $P^{(1)}, P^{(2)}, \ldots, P^{(q)}$, respectively $Q^{(1)}, Q^{(2)}, \ldots, Q^{(q)}$ be the sequence of matrices thus obtained. The matrices P and Q both have indeterminate entries, and we can therefore obtain one from the other by a simple renaming of the indeterminates. Namely, if we define $\varphi(m_{r,t_s}) = m_{r,t_s}$ and $\varphi(m_{r,t_{i+1}}) = m_{r,t_j}$ for $r = 1, \ldots, k$ and $s = 1, \ldots, i$, then $\varphi(P) = Q$. Intuitively speaking, due to the indeterminate entries, it should be equivalent to compute in P or in Q, modulo the renaming of the indeterminates. More precisely, we see immediately by induction that $\varphi(P^{(r)}) = Q^{(r)}$ for $r = 1, \ldots, q$. But we get now a contradiction, because the element $i+1$ of the first row of P is $m_{1,t_{i+1}}^{(q)} = 0$, whereas the element $i+1$ of the first row of Q is $m_{1,t_j}^{(q)} \neq 0$, hence it is not the image by φ of $m_{1,t_{i+1}}^{(q)}$. $\square$

If our method detects all intrinsic constant factors, then this example also shows that even if we compute and divide out at each step the greatest common divisor of the entries of each row, certain sequences of row reductions will inevitably produce a doubly exponential growth of the size of the entries during the computation. For $p = 3$ we verified (by computer) that indeed, in the example above there is no common factor of the entries of the sixth row.

3.5 Combining the Gröbner Basis algorithm with Gaussian Elimination

In this section we will devise a hybrid method for computing Gröbner bases for polynomials with integer coefficients. We combine the Gröbner basis algorithm with Gaussian elimination, with the goal of slowing down the growth of the coefficients.

The ideea is quite simple. We "guess" by a modular Gröbner basis computation which polynomials in the input basis, multiplied by which terms, are needed for obtaining the Gröbner basis polynomials. We triangularize then the matrix associated to these multiples of the polynomials in the input basis, using the classical fraction-free Gaussian elimination. The non-zero rows of the triangular matrix thus obtained are a (redundant) Gröbner basis, (if out prime was lucky). Compared to the Gröbner basis algorithm this algorithm looses, on one hand, in efficiency by not exploiting the special structure of the matrix, and possibly performing the same computation many times, for rows that differ just by shifting. But on the other hand we are likely to gain efficiency in what concerns the coefficient growth. The Gröbner basis computation can be simulated by a non-classical sequence of row reductions performed on a matrix having the same rows as the one above, possibly with multiple occurencess of certain rows. But, as we saw in the previous section, an arbitrary sequence of row reductions can produce much bigger intermediate coefficients than the classical fraction-free Gaussian elimination. Hence this algorithm seems promissing especially for examples in which the coefficients grow very much during the Gröbner basis algorithm.

The hybrid algorithm is based on the following simple theoretical result.

Theorem 12 Let $F = \{f_1, \ldots, f_m\} \subset R[x_1, \ldots, x_n]$ and let $G = \{g_1, \ldots, g_l\}$ be a Gröbner basis of the ideal $I = ideal(F)$. Express the polynomials in G in terms if the initial basis: $g_i = \sum_{j=1}^{m} h_{i,j} f_j$, with $i = 1, \ldots, l$ and $h_{i,j} \in R[x_1, \ldots, x_n]$. Define the set

$$H = \{u f_j | u \in T \text{ and } \exists i \in \{1, \ldots, l\} \text{ such that } coeff(h_{i,j}, u) \neq 0\}$$

where $coeff(h,u)$ denotes the coefficient of the term u in the polynomial h. Triangularize the matrix $M = mat(H)$ and let $\overline{G} = \{\overline{g}_1, \ldots, \overline{g}_r\}$ be the non zero rows of the matrix thus obtained. Then $\overline{G}$ is a (possibly redundant) Gröbner basis for I.

Proof Let V be the vector space over K generated by H. All polynomials g_i belong to this space because they are linear combinations of its generators. The set $\overline{G}$ obtained by elementary row operations in the matrix M, is a system of generators of V. Actually $\overline{G}$ is a basis of V, because all its elements have different leading terms and are therefore linearly independent (such bases are called in [GM88] staggered linear bases). Any g_i can be written as a linear combination of the elements in $\overline{G}$. Since the elements of $\overline{G}$ have distinct leading terms, for each g_i there must exist an element in $\overline{G}$ having the leading term equal to $lt(g_i)$. Hence the monomial ideal generated by $lt(\overline{G})$ includes the monomial ideal generated by $lt(G)$, which is exactly the ideal $lt(I)$. On the other hand all elements of $\overline{G}$ are in the ideal I. Hence $lt(\overline{G})$ generates the monomial ideal $lt(I)$, which means that $\overline{G}$ is a Gröbner basis for I. $\square$

The algorithm will proceed in the following way. We start from a basis $F \subset \mathbb{Z}[x_1, \ldots, x_n]$ and we "guess" the set H by a modular Gröbner basis computation. Namely, for a prime p let $F_p = \{f_1^{(p)}, \ldots, f_m^{(p)}\}$ be the image of F modulo p. Compute a Gröbner basis $G_p = \{g_1^{(p)}, \ldots, g_s^{(p)}\}$ for F_p, over $\mathbb{Z}/(p)$. Assume $g_i^{(p)} = \sum_{j=1}^{m} h_{i,j}^{(p)} f_j^{(p)}$ with $h_{i,j}^{(p)} \in \mathbb{Z}/(p)[x_1, \ldots, x_n]$ and define the set

$$H_p = \{uf_j | u \in T \text{ and } \exists i \in \{1, \ldots, s\} \text{ such that } coeff(h_{i,j}^{(p)}, u) \neq 0\}$$

Notice that $H_p \subset \mathbb{Z}[x_1, \ldots, x_n]$. We triangularize the matrix $M_p = mat(F \cup H_p)$ and let $\overline{G}_p$ be the set of non zero rows of the matrix obtained. Let $\overline{\overline{G}}_p$ be the subset of $\overline{G}_p$ consisting of those polynomials whose leading term is not divisible by the leading term of any other polynomial in $\overline{G}_p$. We check if $\overline{\overline{G}}_p$ is a Gröbner basis. If so, $\overline{\overline{G}}_p$ is a Gröbner basis basis for I and we are done. Otherwise, we restart the computation for the basis $F \cup \overline{\overline{G}}_p$, using another prime.

Bibliography

[Bar68] E.H. Bareiss. Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination. *Mathematics of Computation*, 22:565–578, 1968.

[Bro76] W.S. Brown. On the subresultant prs algorithm. In *SYMSAC proceedings*, page 271, 1976.

[Bro78] W.S. Brown. The subresultant prs algorithm. *ACM Transactions on Mathematical Software*, 4:237–249, 1978.

[BT71] W.S. Brown and J.F Traub. On euclid's algorithm and the theory of subresultants. *Journal of the ACM*, 18:505–514, 1971.

[Buc79] B. Buchberger. A criterion for detecting unnecessary reductions in the construction of Gröbner bases. In *Proceedings of EUROSAM*, pages 3–21, 1979.

[Buc83] Bruno Buchberger. Gröbner Bases, Gaussian Elimination and Euclidean Algorithm. Invited coloquium talk at University of Grenoble, IMAG Institute, 1983.

[Buc85] B. Buchberger. Gröbner bases : An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Multidimensional Systems Theory*, chapter 6. D. Reidel, 1985.

[Buc89] Bruno Buchberger. Gröbner bases and determinant polynomials. Talk at COCOA workshop, Genova, May 29–June 3 1989.

[Buc91] Bruno Buchberger. Gröbner Bases and Determinant Polynomials. Invited talk, Workshop on Symbolic Computation in honour of E. Engeler's 60–th birthday, Mar 1991.

[BW93] T. Becker and V. Weispfenning. *Gröbner Bases*. Springer Verlag, 1993.

[Col66] G. E. Collins. Polynomial remainder sequences and determinants. *Am. Math. Monthly*, 73:708–712, 1966.

[Col67] G. E. Collins. Subresultants and polynomial remainder sequences. *JACM*, 14:128–142, 1967.

[Col86] G. E. Collins. Regularity in Gröbner Basis Construction. Talk at the Seminar on Complexity, Oberwolfach, Nov 1986.

[Dod66] C.L. Dodgson. Condensation of determinants. *Proceedings of the Royal Society of London*, 15:150–155, 1866.

[GM88] R. Gebauer and H.M. Möller. Buchberger's algorithm and staggered linear bases. In *Proceedings of SYMSAC'86*, pages 218–221, 1988.

[GPT88] A. Galligo, L. Pottier, and C. Traverso. Greater Easy Common Divisor and Standard Basis Completion Algorithms. In P. Gianni, editor, *Proceedings of ISSAC88*, pages 162–176. Springer Verlag, 1988.

[Laz83] D. Lazard. Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations. In *Proceedings of EUROCAL*, pages 146–156, 1983.

[Loo83] R. Loos. Generalized Polynomial Remainder Sequences. In B. Buchberger, G. E. Collins, and R. Loos, editors, *Computer Algebra, Symbolic and Algebraic Computation*. Springer Verlag, 1983.

[Man94] A. M. Mandache. The gröbner basis algorithm and subresultant theory. In *Proceedings of ISSAC'94*, pages 123–128. Oxford, UK, ACM press, July 1994.

[MMT92] H.M. Möller, T. Mora, and C. Traverso. Gröbner bases computation using syzygies. In *Proceedings of ISSAC'92*, pages 320–328. Berkley, USA, ACM press, July 1992.