(These lecture notes were prepared and presented by Dan Roche.)

11 Multivariate Polynomials

References: *MCA*: Section 16.6 and Chapter 21
Algorithms for Computer Algebra (Geddes, Czapor, Labahn):
Section 3.4 and Chapter 10
Ideals, Varieties, and Algorithms (Cox, Little, O'Shea): Chapters 1 & 2

Solving a linear system is the same as finding a solution to a system of degree-1 multivariate polynomial equations. That is, given an $n \times n$ matrix A and a $n \times 1$ vector $\mathbf{b}$, solving $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$ is the same as finding a set of values for the variables $x_1, x_2, \ldots, x_n$ that satisfy the set of equations

$$\{A_{i1}x_1 + A_{i2}x_2 + \cdots + A_{in}x_n = b_i\}_{1 \le i \le n}.$$

But what if we want to solve a system of *non-linear* multivariate polynomial equations? Such systems arise frequenty in various engineering and physical science applications. We start with some basic terminology for multivariate polynomials:

Definition 11.1. Let F be a field and $n \in \mathbb{N}$.

- Every *exponent vector* $\mathbf{e} = (e_1, e_2, \ldots, e_n) \in \mathbb{N}^n$ defines a *monomial* in $\mathsf{F}[x_1, x_2, \ldots, x_n]$: $\mathbf{x}^{\mathbf{e}} = x_1^{e_1} \cdot x_2^{e_2} \cdots x_n^{e_n}$.
- A *term* in $\mathsf{F}[x_1, x_2, \ldots, x_n]$ is the product of a nonzero coefficient $c \in \mathsf{F} \setminus \{0\}$ and a monomial, i.e. $c \cdot \mathbf{x}^{\mathbf{e}}$.
- A polynomial $f \in \mathsf{F}[x_1, x_2, \ldots, x_n]$ is a finite sum of terms. We write $\#f$ for the number of terms in f.
- The *max degree* of a monomial is the largest exponent: $\operatorname{maxdeg} \mathbf{x}^{\mathbf{e}} := \|\mathbf{e}\|_\infty = \max_{1 \le i \le n} e_i$. The max degree of a polynomial is the largest max degree of a term that appears in the polynomial.

11.1 Representations

The choice of what data structure we use to represent objects is always of crucial importance in computer science. In mathematical computing, there is a tendency to ignore this and focus only on the mathematical structures at hand. Our computer algebra system (e.g. Maple) will usually make a default choice for us, but it isn't always the best. Let's examine a few options for representing multivatiate polynomials.

11.1.1 Completely Dense Representation

This is the extension of the usual dense univariate polynomial representation that we have covered extensively:

Definition 11.2. The *completely dense representation* of a multivariate polynomial $f \in \mathsf{F}[x_1, x_2, \ldots, x_n]$ with $\deg_{x_i} f < d_i$ for $i = 1, \ldots, n$ is a $d_1 \times d_2 \times \cdots \times d_n$ array with entries in F, such that the entry at index $\mathbf{i} = (i_1, i_2, \ldots, i_n)$ is the coefficient of $\mathbf{x}^{\mathbf{i}}$ in f.

When comparing polynomial representations, we will assume that each coefficient is stored in a single machine word. This is useful even when coefficients take up more space, since we might reasonably suppose then that each coefficient is represented by a pointer to the actual storage, and the size of the coefficient storage will be the same no matter what the polynomial representation scheme is. Under this model, the size of the completely dense representation is $d_1 d_2 \cdots d_n$; if $m \in \mathbb{N}$ such that $m > \operatorname{maxdeg} f$, this is at most m^n.

The completely dense representation allows constant-time access to any coefficient, and all "fast" methods for dense univariate arithmetic are easily applied.

Note that if the array is stored contiguously in row-major order, the representation of $f(x_1, x_2, \ldots, x_n)$ is the same as that for the dense univariate polynomial over $\mathsf{F}[y]$ given by

$$f\left(y^{d_2 d_3 \cdots d_n}, y^{d_3 d_4 \cdots d_n}, \ldots, y^{d_n}, y\right).$$

If we set the dimensions high enough to make room for the result, converting from multivariate to univariate in this fashion is the most obvious way of using fast univariate arithmetic methods for multivariate computation (known as "Kronecker substitution").

11.1.2 Recursive Dense Representation

This is similar to the completely dense representation, except that zero polynomials are "trimmed" from the storage.

Definition 11.3. A polynomial $f \in \mathsf{F}[x_1, x_2, \ldots, x_n]$ is stored in the *recursive dense representation* as either:

- f, if $n = 0$ (since this means $f \in \mathsf{F}$),
- A null pointer, if $f = 0$, or
- A dense array $(f_0, f_1, f_2, \ldots)$, where $f = f_0 + f_1 x_n + f_2 {x_n}^2 + \cdots$ and each $f_i \in \mathsf{F}[x_1, x_2, \ldots, x_{n-1}]$ is also stored in the recursive dense representation.

Consider the size of the recursive dense representation. Assuming, as before, that each coefficient in F can be stored with a single machine word, and letting $\deg_{x_i} f < d_i$ for each i, the following is an upper bound of the recursive dense representation, easily proven by induction:

$$((((d_1 + 1)\, d_2 + 1) \cdots)\, d_{n-1} + 1)\, d_n \quad = \quad \sum_{i=1}^{n} \prod_{j=i}^{n} d_j$$

Under the reasonable assumption that each variable x_i occurs in at least one term of f, each $d_i \geq 2$. This means the above expression is always less than $2 d_1 d_2 \cdots d_n$, and hence is never twice as large as the size of the completely dense representation.

A polynomial with a single term will have size $d_1 + d_2 + \cdots + d_n$ in the recursive dense representation. If $t = \#f$ is the total number of terms in f, the total size is at most $(d_1 + d_2 + \cdots + d_n)t$.

The following theorem summarizes these two bounds.

Theorem 11.4. *Let $f \in \mathsf{F}[x_1, x_2, \ldots, x_n]$ and $m, t \in \mathbb{N}$ such that $m > \operatorname{maxdeg} f$ and $t = \#f$. Then the size of the recursive dense representation of f is bounded above by* $\min(2m^n, mnt)$.

So the recursive dense representation can be exponentially smaller than the completely dense one. Unfortunately, the recursive dense size is dependent on the order of the indeterminates, as illustrated in the following example.

Example. Let $m \in \mathbb{N}$ be arbitrary and

$$f = y^{m-1} + xy^{m-1} + x^2y^{m-1} + \cdots + x^{m-1}y^{m-1}.$$

(Notice that $m > \operatorname{maxdeg} f$.)

If we treat f as over the ring $\mathsf{F}[x, y]$, then the recursive dense representation corresponds to the m-tuple $(0, 0, \cdots, 0, 1 + x + x^2 + \cdots + x^{m-1})$. The total size in this case is $2m$.

On the other hand, if we reverse the variable order so that $f \in \mathsf{F}[y, x]$, the representation corresponds to $(y^{m-1}, y^{m-1}, \ldots, y^{m-1})$, with total size m^2. $\Diamond$

In fact, the preceding example is as bad as it can get, in terms of variance in size between different variable orderings.

Theorem 11.5. *Let $f \in \mathsf{F}[x_1, x_2, \ldots, x_n]$ with max degree $\operatorname{maxdeg} f < m$. If $a, b \in \mathbb{N}$ are the sizes of the recursive dense representations of f under two different orderings of the n variables, then $a \leq mb$.*

Unfortunately, with the smaller size comes a higher cost of computation. Coefficient access costs $O(n)$ in the recursive dense representation. Polynomial arithmetic again reduces (this time recursively) to the univariate case, but operation speed can be slightly greater due to branching (checking whether polynomials are zero).

11.1.3 Sparse Representation

This corresponds to the default representation of polynomials in most computer algebra systems such as Maple.

Definition 11.6. Let $f \in \mathsf{F}[x_1, x_2, \ldots, x_n]$ Write $f = c_1\mathbf{x}^{\mathbf{e}_1} + c_2\mathbf{x}^{\mathbf{e}_2} + \cdots + c_t\mathbf{x}^{\mathbf{e}_3}$, with each $c_i \in \mathsf{F} \setminus \{0\}$ and all the $\mathbf{e}_i$'s distinct elements of $\mathbb{N}^n$.

Then the *sparse representation* of f is list of t coefficient-exponent tuples, specifically $((c_1, \mathbf{e}_1), (c_2, \mathbf{e}_2), \ldots, (c_t, \mathbf{e}_t))$.

Notice that the exponents in this case could be very large, and so we should account for their length in this representation. Writing $t = \#f$ and $\operatorname{maxdeg} f < m$, the size of the sparse representation is thus $O(nt \log m)$. Again, this could be exponentially smaller than the size of the recursive dense representation.

As usual, we pay for the decreased representation size with increased operation costs. Typically, operations on sparse polynomials are counted in terms of the number of monomial comparisons required. (The cost of each comparison depends on the specific monomial order chosen, but is usually $O(n \log m)$.) Naïvely, coefficient access then costs $O(t)$ and addition of two polynomials with s and t terms costs $O(st)$ monomial comparisons.

If the terms are stored in some sorted order, however, arithmetic becomes more efficient. Using binary search, coefficient access costs $O(\log t)$, and addition is equivalent to merging two sorted lists, at cost $O(s + t)$ monomial comparisons.

Multiplication of sparse polynomials with s and t terms can be performed by adding (merging) t intermediate products of s terms each. If we always merge polynomials with the same number of terms, this can be performed with $O(st \log t)$ monomial comparisons.

11.2 Monomial Orderings

The question remains as to how exactly we should order the monomials. The following conditions are sufficient for the correctness of the algorithms we present here, and also correspond to intuition.

Definition 11.7. A relation $\prec$ on exponent tuples in $\mathbb{N}^n$ is an *admissible monomial order* iff it satisfies the following conditions:

- For any $\mathbf{a}, \mathbf{b} \in \mathbb{N}^n$ with $\mathbf{a} \neq \mathbf{b}$, either $\mathbf{a} \prec \mathbf{b}$ or $\mathbf{b} \prec \mathbf{a}$. This means that $\prec$ is a *total order*.
- $(0, \ldots, 0) \preceq \mathbf{e}$ for any $\mathbf{e} \in \mathbb{N}^n$. (Note that $(0, \ldots, 0)$ corresponds to the monomial 1). This means that $\prec$ is a *well order*.
- For any $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{N}^n$ with $\mathbf{a} \prec \mathbf{b}$, $(\mathbf{a} + \mathbf{c}) \prec (\mathbf{b} + \mathbf{c})$ also.

The most common admissible monomial order corresponds to the standard alphabetic ordering used in dictionaries.

Definition 11.8. The *pure lexicographic order* $\prec_{\text{lex}}$ is given by: $(a_1, a_2, \ldots, a_n) \prec_{\text{lex}} (b_1, b_2, \ldots, b_n)$ iff there exists $j \in \{1, 2, \ldots, n\}$ such that $a_j < b_j$ and for each $i \in \{1, 2, \ldots, j-1\}$, $a_i = b_i$.

We also use the $\prec$ symbol to apply the order to monomials and terms (not just exponent tuples). So, for instance, over $\mathsf{F}[x, y, z]$, $x^2y^3z^4 \prec_{\text{lex}} x^4z^2$ and $1000xy^2z^3 \prec_{\text{lex}} 3xy^2z^8$.

There are many other monomial orderings used for various reasons, but for simplicity we will just concentrate on $\prec_{\text{lex}}$, and use $\prec$ to denote this ordering from now on.

This leads to some more new terminology for multivariate polynomials.

Definition 11.9. Let $f \in \mathsf{F}[x_1, x_2, \ldots, x_n]$ and write $f = \sum_{i=1}^{t} c_i \mathbf{x}^{\mathbf{e}_i}$, with $t = \#f$ and each $c_i \in \mathsf{F} \setminus \{0\}$.

- The *multidegree* of f, written $\operatorname{mdeg} f$, is the largest exponent tuple under $\prec$, that is, the unique $\mathbf{e}_j$ such that $\mathbf{e}_i \prec \mathbf{e}_j$ for all $i \neq j$.
- If $\operatorname{mdeg} f = \mathbf{e}_j$ for $j \in \{1, \ldots, t\}$, then the *leading term* of f, denoted $\operatorname{lt}(f)$, is $c_i \mathbf{x}^{\mathbf{e}_j}$. The *leading coefficient* of f, denoted $\operatorname{lc}(f)$, is c_i, and the *leading monomial* of f, denoted $\operatorname{lm}(f)$, is $\mathbf{x}^{\mathbf{e}_j}$.

Example. Let $f \in \mathsf{F}[x, y, z]$ with $f = 2xy^2z + 3y - 5xy^3$. Then f would be stored in the sorted sparse representation as $(-5xy^3, 2xy^2z, 3y)$. This shows that $\operatorname{mdeg}(f) = (1, 3, 0)$, $\operatorname{lt}(f) = -5xy^3$, $\operatorname{lc}(f) = -5$, and $\operatorname{lm}(f) = xy^3$. $\Diamond$

11.3 Reduction and Normal Forms

Consider a single step in the ordinary long division of two univariate polynomials, say f divided by g. We examine the leading terms of f and g. If $\deg f < \deg g$ then we are done, so assume $\deg f \geq \deg g$; this means that $\operatorname{lm}(g)$ divides $\operatorname{lm}(f)$ (since they are both just powers of the single indeterminate). So we write $\operatorname{lt}(f)/\operatorname{lt}(g)$ as the first term of the quotient, and proceed to compute $h = f - (\operatorname{lt}(f)/\operatorname{lt}(g))g$ and then divide h by g.

To generalize this process to multivariate polynomials, we first notice that for $f, g \in \mathsf{F}[x_1, x_2, \ldots, x_n]$, $\operatorname{lm}(g)$ does not necessarily divide $\operatorname{lm}(f)$, even when $\operatorname{mdeg} g \prec \operatorname{mdeg} f$. If, however, this condition does hold, then we can perform a single step of long division as above; this is called a *reduction*.

Definition 11.10. For $f, g, h \in \mathsf{F}[x_1, x_2, \ldots, x_n]$, we say f *reduces to* h *with respect to* g and write $f \underset{g}{\longrightarrow} h$ iff $\mathrm{lm}(g) | \mathrm{lm}(f)$ and

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

Example. Let $g = 3x^2yz^3 + 5xyz^5$. Then

$$15x^3y^3z^3 - 2x^3z \underset{g}{\longrightarrow} -2x^3z - 25x^2y^3z^5$$

$\Diamond$

Having a whole family of polynomials, rather than a single g, can allow us to do more reductions.

Definition 11.11. Let $A \subseteq \mathsf{F}[x_1, x_2, \ldots, x_n]$ be a family of polynomials, and $f, h \in \mathsf{F}[x_1, x_2, \ldots, x_n]$.

- $f \underset{A}{\longrightarrow} h$ iff $\exists g \in A$ such that $f \underset{g}{\longrightarrow} h$.
- $f \overset{*}{\underset{A}{\longrightarrow}} h$ iff $\exists h_1, h_2, \ldots \in A$ s.t. $f \underset{A}{\longrightarrow} h_1 \underset{A}{\longrightarrow} h_2 \underset{A}{\longrightarrow} \cdots \underset{A}{\longrightarrow} h$.

Example. Let $A = \{g_1, g_2, g_3\} \subseteq \mathsf{F}[x, y, z]$ with

$$\begin{aligned} g_1 &= x^2y - 2yz + 1 \\ g_2 &= xy^2 + 2x - z^2 \\ g_3 &= y^2z - y^2 + 5 \end{aligned}$$

Now write $f = 3x^2y^2 + 7y - 1$. We have

$$f \underset{g_1}{\longrightarrow} 6y^2z + 4y - 1 \underset{g_3}{\longrightarrow} 6y^2 + 4y - 31 := h_1$$

$$f \underset{g_2}{\longrightarrow} -6x^2 + 3xz^2 + 7y - 1 := h_2$$

$\Diamond$

Note that both h_1 and h_2 cannot be reduced any further with respect to A. Of course we have a name for this condition:

Definition 11.12. Let $A \subseteq \mathsf{F}[x_1, x_2, \ldots, x_n]$. Then the set of *normal forms with respect to* A, denoted $\mathcal{NF}_A$, is the set of polynomials which cannot be reduced by any polynomial in A. That is,

$$\mathcal{NF}_A = \{f \in \mathsf{F}[x_1, x_2, \ldots, x_n] : \nexists h \in \mathsf{F}[x_1, x_2, \ldots, x_n] \text{ s.t. } f \underset{A}{\longrightarrow} h\}.$$

Since each reduction decreases the multidegree of the polynomial with respect to $\prec$, it should be clear that for any $f \in \mathsf{F}[x_1, x_2, \ldots, x_n]$ and a *finite* subset $A \subseteq \mathsf{F}[x_1, x_2, \ldots, x_n]$, we can always find an $h \in \mathsf{F}[x_1, x_2, \ldots, x_n]$ such that $f \overset{*}{\underset{A}{\longrightarrow}} h$ and $h \in \mathcal{NF}_A$, just by repeatedly reducing f by elements of A until we can't anymore.

If we further specify some algorithmic way of choosing the order of reductions (say by ordering the polynomials in A), then the h we find will always be the same. Without giving a formal definition, let's assume some such algorithm is understood, and write $\mathcal{NF}_A(f) = h$ in this case.

11.4 Ideals and Bases

Recall the notion of an ideal from algebra:

Definition 11.13. Let R with operations $(+, \cdot)$ be a ring. An *ideal* I of R is a subset of R satisfying:

- $a + b \in I$, for all $a, b \in I$
- $a \cdot r \in I$, for all $a \in I$ and $r \in \mathsf{R}$

Note that having some elements of R in an ideal requires that many other elements must be in the ideal also. Hence we have the following:

Definition 11.14. A subset $S = \{s_1, s_2, \ldots\}$ of a ring R is a *generator* for an ideal, which we denote as $\langle S \rangle$ or $\langle s_1, s_2, \ldots \rangle$:

$$\langle S \rangle := \{r_1 s_1 + r_2 s_2 + \cdots : r_1, r_2, \ldots \in \mathsf{R}\}.$$

The set S is called a *basis* for the ideal $\langle S \rangle$.

Note that $\langle S \rangle$ is the smallest ideal in R which contains S. Also, a single ideal can have many different bases.

For example, over $\mathbb{Z}$, you can convince yourself that $\langle 4, 6 \rangle$ is just the set of even integers, and so is the same as $\langle 2 \rangle$. In fact, from the extended Euclidean algorithm, we can see that any ideal in $\mathbb{Z}$ is generated by a single integer, which is just the gcd of all the integers in the ideal.

Here, we're interested in the ring $\mathsf{F}[x_1, x_2, \ldots, x_n]$, where things are not so simple, and most ideals don't have a single-element basis. But it's not too bad:

Theorem 11.15 (Hilbert, 1888)**.** *Every ideal I of $\mathsf{F}[x_1, x_2, \ldots, x_n]$ has a finite basis.*

This is known as "Hilbert's basis theorem" and is crucial for the correctness of our methods here (but says nothing about the complexity because there's no concrete limit on the size of the basis).

Now let's try an extend the idea of modular equivalence to multivariate polynomials. Over $\mathbb{Z}$, $a \equiv b \bmod m$ iff $m|(a - b)$. This is the same as saying that $(a - b) \in \langle m \rangle$. So for $f, h \in \mathsf{F}[x_1, x_2, \ldots, x_n]$ and $A \subseteq \mathsf{F}[x_1, x_2, \ldots, x_n]$, we say that $f \equiv h \bmod \langle A \rangle$ iff $(f - h) \in \langle A \rangle$.

From this definition, it follows immediately that $f \underset{A}{\longrightarrow} h$ implies $f \equiv h \bmod A$, and in particular $f \equiv \mathcal{NF}_A(f)$.

11.5 Gröbner Bases

Consider the *ideal membership problem*: given polynomials $f, g_1, g_2, \ldots, g_k \in \mathsf{F}[x_1, x_2, \ldots, x_n]$, can f be written as a sum of multiples of g_i's, i.e. is $f \in \langle g_1, g_2, \ldots, g_k \rangle$? Setting $A = \{g_1, \ldots, g_k\}$, we know this is the case iff $f \equiv 0 \bmod \langle A \rangle$. And clearly $0 \in \mathcal{NF}_A$. But there may be many elements of $\mathcal{NF}_A$ which are equivalent to f, and there is no guarantee that our deterministic strategy for choosing $\mathcal{NF}_A(f)$ will produce 0, or even that $f \underset{A}{\overset{*}{\longrightarrow}} 0$.

Gröbner bases provide such a guarantee.

Definition 11.16. A set $G \subseteq \mathsf{F}[x_1, x_2, \ldots, x_n]$ is a *Gröbner Basis* for an ideal I if and only if

$$\forall f, h \in \mathcal{NF}_G, f \equiv h \bmod I \Rightarrow f = h.$$

This implies immediately that for any $f \in \mathsf{F}[x_1, x_2, \ldots, x_n]$, there is only one choice for $\mathcal{NF}_G(f)$, regardless of the order of reductions. And it also proves that G is actually a basis for I: for any $h \in I$, since $h \equiv 0 \bmod I$ and $0 \in \mathcal{NF}_G$, $h \xrightarrow[G]{*} 0$, which means that $h \in \langle G \rangle$.

Since the choice of $\mathcal{NF}_G(f)$ is unique, we now have a *canonical normal form* for polynomials modulo I. So having a *finite* Gröbner Basis for an ideal I gives us a way to determine whether $f \equiv g \bmod I$ for any $f, g \in \mathsf{F}[x_1, x_2, \ldots, x_n]$, and in particular solves the ideal membership problem.

But how can we compute a finite Gröbner basis for an ideal $I = \langle f_1, f_2, \ldots, f_k \rangle$? We'll breifly outline the approach taken by Bruno Buchberger, who actually coined the term "Gröbner Basis" in his 1965 Ph.D. thesis. We start with the following definition:

Definition 11.17. Let $f, g \in \mathsf{F}[x_1, x_2, \ldots, x_n] \backslash \{0\}$ and $u = \operatorname{lcm}(\operatorname{lt}(f), \operatorname{lt}(g))$. Then the *s-polynomial* of f and g is

$$\text{s-poly}(f, g) = \frac{u}{\operatorname{lt}(f)} f - \frac{u}{\operatorname{lt}(g)} g.$$

Intuitively, s-poly(f, g) is the smallest polynomial in $\langle f, g \rangle$ with the leading terms of f and g stripped out. s-polynomials also give a straightworward test for Gröbner bases:

Theorem 11.18 (*MCA*, Theorem 21.31)**.** *$G \subseteq \mathsf{F}[x_1, x_2, \ldots, x_n]$ is a Gröbner basis for the ideal $\langle G \rangle$ iff $\mathcal{NF}_G(\text{s-poly}(g_1, g_2)) = 0$ for all $g_1, g_2 \in G$.*

See the book for a proof. Given a finite set $G \subseteq \mathsf{F}[x_1, x_2, \ldots, x_n]$, Buchberger's algorithm to compute a Gröbner basis for the ideal $\langle G \rangle$ is based around this theorem. Choose the first pair $g_i, g_j \in G$ in the basis such that $\mathcal{NF}_G(\text{s-poly}(g_i, g_j)) \neq 0$. If no such pair exists, then G is a Gröbner basis and we are done. Otherwise, add $\mathcal{NF}_G(\text{s-poly}(g_i, g_j))$ to G and repeat. (This is presented more formally as Algorithm 21.33 in *MCA*.)

Example. Let's use Buchberger's algorithm to find a Gröbner basis for $\{g_1, g_2\}$ with

$$g_1 = 3x^2y + 5y - z,$$

$$g_2 = yz.$$

Initially there is just one pair of polynomials in the basis. So compute

$$\mathcal{NF}_G(\text{s-poly}(g_1, g_2)) = \mathcal{NF}_G(5yz - z^2) = -z^2$$

So set $g_3 = z^2$ (dividing out the content) and update $G = G \bigcup g_3$. Then we just need to confirm that

$$\mathcal{NF}_G(\text{s-poly}(g_1, g_3)) = \mathcal{NF}_G(5yz^2 - z^3) = 0,$$
$$\mathcal{NF}_G(\text{s-poly}(g_2, g_3)) = \mathcal{NF}_G(0) = 0.$$

So $\{g_1, g_2, g_3\}$ is a Gröbner basis for the original set G. ◊

Theorem 11.19. *Buchberger's algorithm terminates.*

Proof. Let M_i be the ideal generated by the leading monomials of G after the i'th iteration. Since each new polynomial added to G is in normal form, each $M_i \subsetneq M_{i+1}$. Hence we have an *ascending chain* of ideals. By Hilbert's basis theorem, the ideal $\bigcup_{i \geq 0} M_i$ has a finite basis, say G'. We must have $G' \subseteq M_i$ for some $i \geq 0$. But then $M_i = M_{i+1}$, a contradiction unless there are only i iterations. □

Unfortunately, Gröbner bases can be very large, especially under the lexicographical ordering. We can reduce the size of G somewhat by removing any basis element that is not in $\mathcal{NF}_G$, and further reduction is possible by so-called "interior reductions" that do not always involve the leading monomials.

We've already seen how a Gröbner basis allows us to test ideal membership and equivalence modulo an ideal. Another very useful application is solving a multivariate polynomial system.

Consider the system $\mathcal{S} = (f_i = v_i)_{i=1,2,\ldots,k}$ with each $f_i \in \mathsf{F}[x_1, x_2, \ldots, x_n]$ and $v_i \in \mathsf{F}$. This is of course equivalent to the system $(f_i - v_i = 0)_{i=1,2,\ldots,k}$, so without loss of generality assume each $v_i = 0$.

Let G be a Gröbner basis for the ideal generated by $\{f_i : i = 1, \ldots, k\}$, under the pure lexicographical ordering $\prec_{\text{lex}}$. I claim (without proof) that if the system is zero-dimensional (i.e. it has finitely many solutions), then G contains a polynomial which only contains x_n, one containing only x_{n-1} and x_n, and so forth. There is a name for such a set:

Definition 11.20. A set $T = \{t_1, t_2, \ldots, t_n\} \subseteq \mathsf{F}[x_1, x_2, \ldots, x_n]$ is a *triangular set* iff $t_i \in \mathsf{F}[x_i, x_{i+1}, \ldots, x_n]$ for $i = 1, 2, \ldots, n$.

With this triangular set, we can solve the univariate polynomial t_n for x_n, then use back-substitution into t_{n-1} to solve for x_{n-1}, and so forth. So Gröbner bases can be used to solve zero-dimensional systems of multivariate polynomials.

It is worth noting that Buchberger's algorithm has been more or less superceded by more efficient approaches, especially the F4 and F5 algorithms by Jean-Charles Faugère. There are alternative methods to compute triangular sets for a given polynomial system as well.