Course "The Power of Polynomials and How To Use Them“, JASS 2007

GCD and factorization of multivariate polynomials

Rosa Freund

Computer Science Department
TU München

March 23, 2007
Definition 1
Let $R$ be a ring. $R[x_1, \ldots, x_k] = R[x]$ is the set of all multivariate polynomials over $R$. We write $a(x) \in R[x]$ as

$$a(x) = \sum_{e \in \mathbb{N}^k} a_e x^e$$

To work with multivariate polynomials, we need some basic arithmetic concepts such as an ordering.
Definition 2

Lexicographical ordering: Let $d, e \in \mathbb{N}^k$ be two exponent vectors. Let $j < k$ be the smallest integer such that $d_j \neq e_j$. Define an ordering as follows:

$d < e$ if $d_j < e_j$

$d > e$ if $d_j > e_j$

The coefficient of the first term of a lexicographically ordered polynomial is called leading coefficient and denoted by $\text{lcoeff}(a(x))$.

Example 3

The following polynomial $\in \mathbb{Z}[x, y, z]$ is arranged in lexicographically decreasing order:

$$A(x) = 2x^3y^3z^7 + 3x^3y^2z^8 - 5x^2y^7 + z^3$$
Definition 2
Lexicographical ordering: Let \( d, e \in \mathbb{N}^k \) be two exponent vectors. Let \( j < k \) be the smallest integer such that \( d_j \neq e_j \). Define an ordering as follows:
\[
d < e \text{ if } d_j < e_j \\
d > e \text{ if } d_j > e_j
\]
The coefficient of the first term of a lexicographically ordered polynomial is called leading coefficient and denoted by \( \text{lcoeff}(a(x)) \).

Example 3
The following polynomial \( \in \mathbb{Z}[x, y, z] \) is arranged in lexicographically decreasing order:
\[
A(x) = 2x^3y^3z^7 + 3x^3y^2z^8 - 5x^2y^7 + z^3
\]
Definition 4
The degree vector $\delta(A(x))$ of a multivariate polynomial is the exponent vector of its leading term. The total degree of a multivariate polynomial is the maximum degree of any of its summands. The degree of a summand is the sum of all exponents of its terms.
Problem of Euclidean Algorithm with polynomials: Growth of Remainders, even when adjusted to work only in rings. Consider the following example:

**Example 5**

Let $A(x), B(x) \in \mathbb{Z}[x]$ be defined as

\[
A(x) = x^8 + x^6 - 3x^4 - 3x^3 + x^2 + 2x - 5
\]

\[
B(x) = 3x^6 + 5x^4 - 4x^2 - 9x + 21
\]
Running the Euclidean algorithm in $\mathbb{Q}$ yields the following remainder sequence:

\begin{align*}
R_2(x) & = -\frac{5}{9}x^4 + \frac{1}{9}x^2 - \frac{1}{3} \\
R_3(x) & = -\frac{117}{25}x^2 - 9x + \frac{411}{25} \\
R_4(x) & = \frac{233150}{19773}x - \frac{102500}{6591} \\
R_5(x) & = -\frac{1288744821}{543589225}
\end{align*}

Since $R_5(x)$ is a unit in $\mathbb{Q}$, $A$ and $B$ are relatively prime.
Algorithm MGCD works as follows:

- Use ring homomorphisms to map polynomials from \( D \) to simpler UFDs \( D' \)
- Solve for GCD in new UFD (e.g. by Euclidean Algorithm)
- It can be shown that deg(GCD in \( D \)) \( \leq \) deg(GCD in \( D' \)). We thus have an upper bound for the degree of the GCD in \( D \).
- Information loss is compensated by using several different homomorphisms
- Multivariate polynomials are handled recursively by viewing \( R[x_1, \ldots, x_k] \) as \( R[x_1, \ldots, x_{k-1}][x_k] \)
Algorithm MGCD works as follows:

- Use ring homomorphisms to map polynomials from \( D \) to simpler UFDs \( D' \)
- Solve for GCD in new UFD (e.g. by Euclidean Algorithm)
- It can be shown that \( \deg(\text{GCD in } D) \leq \deg(\text{GCD in } D') \). We thus have an upper bound for the degree of the GCD in \( D \).
- Information loss is compensated by using several different homomorphisms
- Multivariate polynomials are handled recursively by viewing \( R[x_1, \ldots, x_k] \) as \( R[x_1, \ldots, x_{k-1}][x_k] \)
Algorithm MGCD works as follows:

- Use ring homomorphisms to map polynomials from $D$ to simpler UFDs $D'$
- Solve for GCD in new UFD (e.g. by Euclidean Algorithm)
- It can be shown that $\deg(\text{GCD in } D) \leq \deg(\text{GCD in } D')$. We thus have an upper bound for the degree of the GCD in $D$.
- Information loss is compensated by using several different homomorphisms
- Multivariate polynomials are handled recursively by viewing $R[x_1, \ldots, x_k]$ as $R[x_1, \ldots, x_{k-1}][x_k]$
Algorithm MGCD works as follows:

- Use ring homomorphisms to map polynomials from $D$ to simpler UFDs $D'$
- Solve for GCD in new UFD (e.g. by Euclidean Algorithm)
- It can be shown that $\deg(\text{GCD in } D) \leq \deg(\text{GCD in } D')$. We thus have an upper bound for the degree of the GCD in $D$.
- Information loss is compensated by using several different homomorphisms
- Multivariate polynomials are handled recursively by viewing $R[x_1, \ldots, x_k]$ as $R[x_1, \ldots, x_{k-1}][x_k]$
Algorithm MGCD works as follows:

- Use ring homomorphisms to map polynomials from $D$ to simpler UFDs $D'$
- Solve for GCD in new UFD (e.g. by Euclidean Algorithm)
- It can be shown that $\deg(\text{GCD in } D) \leq \deg(\text{GCD in } D')$. We thus have an upper bound for the degree of the GCD in $D$.
- Information loss is compensated by using several different homomorphisms
- Multivariate polynomials are handled recursively by viewing $R[x_1, \ldots, x_k]$ as $R[x_1, \ldots, x_{k-1}][x_k]$
Modular GCD algorithm MGCD
Input: $A, B \in \mathbb{Z}[x_1, \ldots, x_k]$
Example 6
Consider the following polynomials \( \in \mathbb{Z}[x, y, z] \):

\[
A(x, y, z) = 9x^5 + 2x^4yz - 189x^3y^2z + 117x^3yz^2 + 3x^3 - 42x^2y^4z^2 + 26x^2y^2z^3 + 18x^2 - 63xy^3z + 39xyz^2 + 4xyz + 6
\]

\[
B(x, y, z) = 6x^6 - 126x^4y^3z + 78x^4yz^2 + x^4y + x^4z + 13x^3 - 21x^2y^4z - 21x^2y^3z^2 + 13x^2y^2z^2 + 13x^2yz^3 - 21xy^3z + 13xyz^2 + 2xy + 2xz + 2
\]

Use 3 moduli in which to work: 11, 13 and 17.
In \( \mathbb{Z}_{11} \) we now work with the polynomials

\[
A_{11}(x, y, z) = -2x^5 + 2x^4yz - 2x^3y^2z - 4x^3yz^2 + 3x^3 + 2x^2y^4z^2 + 4x^2y^2z^3 - 4x^2 + 3xy^3z - 5xyz^2 + 4xyz - 5
\]

and

\[
B_{11}(x, y, z) = -5x^6 - 5x^4y^3z + x^4yz^2 + x^4y + x^4z + 2x^3 + x^2y^4z + x^2y^3z^2 + 2x^2y^2z^2 + 2x^2yz^3 + xy^3z + 2xyz^2 + 2xy + 2xz + 2
\]

Now evaluate polynomials at four arbitrary points and compute GCD recursively.
Problems with MGCD:

- Need to throw away "unlucky homomorphisms"
- Number of domains which have to be used is exponential in the number of variables of the polynomials.
- Ineffective, when the polynomials have a "sparse" rather than a "dense" structure
- Hence: Especially useless for multivariate polynomials!
Problems with MGCD:

- Need to throw away "unlucky homomorphisms"
- Number of domains which have to be used is exponential in the number of variables of the polynomials.
- Ineffective, when the polynomials have a "sparse" rather than a "dense" structure
- Hence: Especially useless for multivariate polynomials!
Problems with MGCD:

- Need to throw away "unlucky homomorphisms"
- Number of domains which have to be used is exponential in the number of variables of the polynomials.
- Ineffective, when the polynomials have a "sparse" rather than a "dense" structure
- Hence: Especially useless for multivariate polynomials!
Problems with MGCD:

- Need to throw away "unlucky homomorphisms"
- Number of domains which have to be used is exponential in the number of variables of the polynomials.
- Ineffective, when the polynomials have a "sparse" rather than a "dense" structure
- Hence: Especially useless for multivariate polynomials!
Algorithm SparseMod (Zippel, 1979) works as follows:

- Constructs alternating sequence of dense and sparse interpolations
Algorithm EZ-GCD (Moses, Yun 1973) works as follows:

- Uses Hensel’s lemma to reduce polynomials to a univariate representation, determine GCD in simpler domain
- Requires just one homomorphism for each variable
- As with MGCD, relatively prime polynomials are discovered quickly
Algorithm EZ-GCD (Moses, Yun 1973) works as follows:

- Uses Hensel’s lemma to reduce polynomials to a univariate representation, determine GCD in simpler domain
- Requires just one homomorphism for each variable
- As with MGCD, relatively prime polynomials are discovered quickly
Algorithm EZ-GCD (Moses, Yun 1973) works as follows:

- Uses Hensel’s lemma to reduce polynomials to a univariate representation, determine GCD in simpler domain
- Requires just one homomorphism for each variable
- As with MGCD, relatively prime polynomials are discovered quickly
Extended Zassenhaus GCD algorithm EZ-GCD
Input: $A, B \in \mathbb{Z}[x]$
Multivariate factoring problems over $\mathbb{Z}$ can be reduced to univariate factoring problems modulo a prime

**Definition 7**

$a(x) \in R[x]$ is called **square-free** if it has no repeated factors.

**Definition 8**

The **square-free factorization** of $a(x)$ is $a(x) = \prod_{i=1}^{k} a_i(x)^i$, where each $a_i(x)$ is square-free, and $\text{GCD}(a_i(x), a_j(x)) = 1$ for $i \neq j$. 
Multivariate factoring problems over $\mathbb{Z}$ can be reduced to univariate factoring problems modulo a prime

**Definition 7**
$a(x) \in R[x]$ is called square-free if it has no repeated factors.

**Definition 8**
The square-free factorization of $a(x)$ is $a(x) = \prod_{i=1}^{k} a_i(x)^i$, where each $a_i(x)$ is square-free, and $\gcd(a_i(x), a_j(x)) = 1$ for $i \neq j$. 
Algorithm SquareFree determines the square-free factorization of a polynomial \( a(x) \in R[x], \) \( R \) UFD with \( \text{char}(R) = 0 \)

Improvement by Yun (19??): One more differentiation than SquareFree, but much simpler GCD calculations.

Similar algorithm determines square-free factorization over finite fields \( GF(q) \)
Algorithm by Berlekamp (1967) works as follows: Factors polynomials in $GF(q)[x]$ where $q = p^m$
Berlekamp’s Factoring Algorithm
Input: $A, B \in \mathbb{Z}[x]$
Multivariate Factoring: Accomplished by factoring of univariate polynomials over a finite field and Hensel liftings.
<table>
<thead>
<tr>
<th>Introduction</th>
<th>GCD</th>
<th>Factorization</th>
</tr>
</thead>
</table>

Rosa Freund: GCD and Factorisation of multivariate polynomials