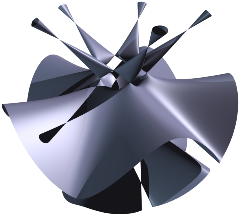


Gröbner basis - What, Why and How?

Tushant Mittal

Agenda

Motivational Problems

- **Ideal Membership Problem**
  Given $f \in k[x_1, x_2, \cdots x_n]$ and an ideal $I =< f_1, f_2, \cdots, f_n >$, determine if $f \in I$.

Motivational Problems

- **Ideal Membership Problem**
  Given $f \in k[x_1, x_2, \cdots x_n]$ and an ideal $I =< f_1, f_2, \cdots , f_n >$, determine if $f \in I$.
- **Solving Polynomial Equations**
  Find all solution in k^n of a system of polynomial equations $f_i(x_1, x_2, \cdots , x_n) = 0$. In other words, given an ideal I, compute $V(I)$.

Motivational Problems

- **Ideal Membership Problem**
  Given $f \in k[x_1, x_2, \cdots x_n]$ and an ideal $I =< f_1, f_2, \cdots, f_n >$, determine if $f \in I$.
- **Solving Polynomial Equations**
  Find all solution in k^n of a system of polynomial equations $f_i(x_1, x_2, \cdots, x_n) = 0$. In other words, given an ideal I, compute $V(I)$.
- **Implicitization Problem**
  Given a parametric solution of x_i's in terms of variables t_i i.e. $x_i = g_i(t_1, t_2, \cdots, t_i)$, find a set of polynomials f_i such that $x_i \in V(< f_1, f_2, \cdots, f_n >)$. It can be easily observed that this is essentially the inverse of the above question i.e given $V(I)$ compute I.

Motivational Problems

- **Ideal Membership Problem**
  Given $f \in k[x_1, x_2, \cdots x_n]$ and an ideal $I =< f_1, f_2, \cdots, f_n >$, determine if $f \in I$.
- **Solving Polynomial Equations**
  Find all solution in k^n of a system of polynomial equations $f_i(x_1, x_2, \cdots, x_n) = 0$. In other words, given an ideal I, compute $V(I)$.
- **Implicitization Problem**
  Given a parametric solution of x_i's in terms of variables t_i i.e. $x_i = g_i(t_1, t_2, \cdots, t_i)$, find a set of polynomials f_i such that $x_i \in V(< f_1, f_2, \cdots, f_n >)$. It can be easily observed that this is essentially the inverse of the above question i.e given $V(I)$ compute I.

But an immediate question arises.
How do we even store these ideals which are possibly of infinite size ?

Noetherian Ring

- A Noetherian ring is a ring that satisfies the ascending chain condition on ideals; that is, given any chain of ideals:

$$I_1 \subseteq \cdots \subseteq I_{k-1} \subseteq I_k \subseteq I_{k+1} \subseteq \cdots$$

there exists an n such that: $I_n = I_{n+1} = \cdots I_{n+k}\ \forall k \geq 0$

Noetherian Ring

- A Noetherian ring is a ring that satisfies the ascending chain condition on ideals; that is, given any chain of ideals:

$$I_1 \subseteq \cdots \subseteq I_{k-1} \subseteq I_k \subseteq I_{k+1} \subseteq \cdots$$

  there exists an n such that: $I_n = I_{n+1} = \cdots I_{n+k}\ \forall k \geq 0$
- Equivalently, every ideal I in R is finitely generated, i.e. there exist elements $a1, ..., an$ in I such that $I =< a_1, a_2, \cdots, a_n >$

Noetherian Ring

- A Noetherian ring is a ring that satisfies the ascending chain condition on ideals; that is, given any chain of ideals:

$$I_1 \subseteq \cdots \subseteq I_{k-1} \subseteq I_k \subseteq I_{k+1} \subseteq \cdots$$

  there exists an n such that: $I_n = I_{n+1} = \cdots I_{n+k} \ \forall k \geq 0$
- Equivalently, every ideal I in R is finitely generated, i.e. there exist elements $a1, ..., an$ in I such that $I =< a_1, a_2, \cdots, a_n >$

Theorem (Hilbert Basis Theorem)

R is Noetherian $\Rightarrow R[x]$ *is Noetherian*

Special Cases

- $R = k[x]$ i.e. $n = 1$.
  We know that $k[x]$ is a PID. Moreover, it is a Euclidean domain and hence, a polynomial $g \in < f >$ *iff* $f|g$.

Special Cases

- $R = k[x]$ i.e. $n = 1$.
  We know that $k[x]$ is a PID. Moreover, it is a Euclidean domain and hence, a polynomial $g \in < f > \ iff \ \ f|g$.
- Linear Algebra techniques can be used efficiently when the degree of the polynomials is restricted to 1 irrespective of n.

Special Cases

- $R = k[x]$ i.e. $n = 1$.
  We know that $k[x]$ is a PID. Moreover, it is a Euclidean domain and hence, a polynomial $g \in < f > \ iff \ \ f|g$.
- Linear Algebra techniques can be used efficiently when the degree of the polynomials is restricted to 1 irrespective of n.
- We will generalize both the idea of division and a basis to solve the problem for the general case.

Monomial Ordering

We will use the notation x^α to represent $\prod_i^n x_i^{\alpha_i}$ where $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$.

Definition (admissible ordering of monomials)

A total ordering on all monomials is an ordering for which holds:

- $x^\alpha < x^\beta \Rightarrow \forall\delta\colon\ x^\alpha x^\delta < x^\beta x^\delta$.
- $\forall\alpha\colon\ 1 < x^\alpha$.

Monomial Ordering

We will use the notation x^α to represent $\prod_i^n x_i^{\alpha_i}$ where $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$.

Definition (admissible ordering of monomials)

A total ordering on all monomials is an ordering for which holds:

- $x^\alpha < x^\beta \Rightarrow \forall\delta\colon\ x^\alpha x^\delta < x^\beta x^\delta$.
- $\forall\alpha\colon\ 1 < x^\alpha$.

A few popular orderings are:

1. Lexicographical ordering: In which we compare x^α and x^β thus: if the first $k-1$ indices agree, $\alpha_i = \beta_i, i \leq k-1$ and the kth differ, we decide based on that index $\alpha_k \leq \beta_k \Rightarrow \alpha \leq \beta$, and the reverse.

Monomial Ordering

We will use the notation x^α to represent $\prod_i^n x_i^{\alpha_i}$ where $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$.

Definition (admissible ordering of monomials)

A total ordering on all monomials is an ordering for which holds:

- $x^\alpha < x^\beta \Rightarrow \forall\delta\colon\ x^\alpha x^\delta < x^\beta x^\delta$.
- $\forall\alpha\colon\ 1 < x^\alpha$.

A few popular orderings are:

1. Lexicographical ordering: In which we compare x^α and x^β thus: if the first $k-1$ indices agree, $\alpha_i = \beta_i, i \leq k-1$ and the kth differ, we decide based on that index $\alpha_k \leq \beta_k \Rightarrow \alpha \leq \beta$, and the reverse.
2. Graded lexicographical order: in which the order is by the degree of the monomials and ties are broken using lexicographical ordering.

Preliminary Definitions

Let $f = \sum_i a_i x^{\alpha_i}$ be a polynomial. Associated with it are the following definitions

Preliminary Definitions

Let $f = \sum_i a_i x^{\alpha_i}$ be a polynomial. Associated with it are the following definitions

Definition (Multidegree)

$multideg(f) = max_i \alpha_i$

Preliminary Definitions

Let $f = \sum_i a_i x^{\alpha_i}$ be a polynomial. Associated with it are the following definitions

Definition (Multidegree)

$multideg(f) = max_i \alpha_i$

Definition (Leading Coefficient)

$LC(f) = a_{multideg(f)}$

Preliminary Definitions

Let $f = \sum_i a_i x^{\alpha_i}$ be a polynomial. Associated with it are the following definitions

Definition (Multidegree)

$multideg(f) = max_i \alpha_i$

Definition (Leading Coefficient)

$LC(f) = a_{multideg(f)}$

Definition (Leading Monomial)

$LM(f) = x^{multideg(f)}$

Preliminary Definitions

Let $f = \sum_i a_i x^{\alpha_i}$ be a polynomial. Associated with it are the following definitions

Definition (Multidegree)

$multideg(f) = max_i \alpha_i$

Definition (Leading Coefficient)

$LC(f) = a_{multideg(f)}$

Definition (Leading Monomial)

$LM(f) = x^{multideg(f)}$

Definition (Leading Term)

$LT(f) = LC(f)LT(f)$

Example

Let $f = 7x^3y^2z + 2x^2yz^4 + 9xy^4 + 3yz^7 + 2$.

Using the lex ordering,

- $multideg(f) = (3, 2, 1)$
- $LC(f) = 7$
- $LM(f) = x^3y^2z$
- $LT(f) = 7x^3y^2z$

Example

Let $f = 7x^3y^2z + 2x^2yz^4 + 9xy^4 + 3yz^7 + 2$.

Using the lex ordering,

- $multideg(f) = (3, 2, 1)$
- $LC(f) = 7$
- $LM(f) = x^3y^2z$
- $LT(f) = 7x^3y^2z$

Whereas using the grlex ordering we would get,

- $multideg(f) = (0, 0, 7)$
- $LC(f) = 3$
- $LM(f) = yz^7$
- $LT(f) = 3yz^7$

Division Algorithm

The division algorithm is essentially the same as the one in the univariate case but there is a small change which has to be made. To see this, let us look at an example,

Division Algorithm

The division algorithm is essentially the same as the one in the univariate case but there is a small change which has to be made. To see this, let us look at an example,

$$\begin{array}{rl} a_1: & x+y \\ a_2: & 1 \\ \begin{array}{r} xy+1 \\ y^2+1 \end{array} & \big) \overline{x^2y + xy^2 + y^2} \end{array} \qquad r$$

Division Algorithm

The division algorithm is essentially the same as the one in the univariate case but there is a small change which has to be made. To see this, let us look at an example,

$a_1 :$ $x + y$

$a_2 :$ 1 $\qquad r$

$$\begin{array}{l} xy+1 \\ y^2+1 \end{array} \ \big) \ \overline{x^2y + xy^2 + y^2}$$

$$x^2y - x$$

Division Algorithm

The division algorithm is essentially the same as the one in the univariate case but there is a small change which has to be made. To see this, let us look at an example,

$$
\begin{array}{rll}
 & a_1: & x + y \\
 & a_2: & 1 \\
xy + 1 & & \\
y^2 + 1 &) & \overline{x^2y + xy^2 + y^2} \\
 & & x^2y - x \\
 & & \overline{\qquad xy^2 + x + y^2}
\end{array}
\qquad r
$$

Division Algorithm

The division algorithm is essentially the same as the one in the univariate case but there is a small change which has to be made. To see this, let us look at an example,

$$
\begin{array}{rl}
a_1: & x + y \\
a_2: & 1 \\
xy + 1 & \\
y^2 + 1 & \big) \overline{x^2y + xy^2 + y^2} \\
& x^2y - x \\
\hline
& \quad xy^2 + x + y^2 \\
& \quad xy^2 - y
\end{array}
\qquad
\begin{array}{c}
r \\
\hline
\end{array}
$$

Division Algorithm

The division algorithm is essentially the same as the one in the univariate case but there is a small change which has to be made. To see this, let us look at an example,

$$
\begin{array}{rl}
a_1: & x+y \\
a_2: & 1 \\
\begin{array}{r} xy+1 \\ y^2+1 \end{array} & \big) \overline{x^2y+xy^2+y^2} \\
 & x^2y - x \\
 & \overline{\quad xy^2 + x + y^2} \\
 & \quad xy^2 - y \\
 & \overline{\qquad x + y^2 + y}
\end{array}
\qquad r
$$

Division Algorithm

The division algorithm is essentially the same as the one in the univariate case but there is a small change which has to be made. To see this, let us look at an example,

$$
\begin{array}{rll}
a_1: & x + y & \\
a_2: & 1 & \qquad r \\
\begin{array}{r} xy+1 \\ y^2+1 \end{array} & \big) \overline{x^2y + xy^2 + y^2} & \qquad \underline{\qquad\qquad} \\
 & x^2y - x & \\
\hline
 & xy^2 + x + y^2 & \\
 & xy^2 - y & \\
\hline
 & x + y^2 + y & \quad \to x
\end{array}
$$

Division Algorithm

The division algorithm is essentially the same as the one in the univariate case but there is a small change which has to be made. To see this, let us look at an example,

$a_1 :\quad x + y$

$a_2 :\quad 1 \qquad\qquad r$

$xy + 1$

$y^2 + 1 \quad) \; \overline{x^2y + xy^2 + y^2}$

$x^2y - x$

$xy^2 + x + y^2$

$xy^2 - y$

$x + y^2 + y \qquad \rightarrow x$

$y^2 + y$

Division Algorithm

The division algorithm is essentially the same as the one in the univariate case but there is a small change which has to be made. To see this, let us look at an example,

$$
\begin{array}{rl}
a_1: & x + y \\
a_2: & 1 \\
xy + 1 & \\
y^2 + 1 & \overline{)\,x^2y + xy^2 + y^2} \\
& x^2y - x \\
\hline
& xy^2 + x + y^2 \\
& xy^2 - y \\
\hline
& x + y^2 + y \quad \rightarrow x \\
\hline
& y^2 + y \\
& y^2 - 1 \\
\hline
& y + 1
\end{array}
\qquad r
$$

Division Algorithm

The division algorithm is essentially the same as the one in the univariate case but there is a small change which has to be made. To see this, let us look at an example,

$$a_1: \quad x + y$$
$$a_2: \quad 1 \qquad\qquad r$$

$$\begin{array}{l}
xy+1 \\
y^2+1
\end{array}
\;\Big)\;
\begin{array}{rl}
\overline{x^2y + xy^2 + y^2} & \\
x^2y - x & \\
\hline
xy^2 + x + y^2 & \\
xy^2 - y & \\
\hline
x + y^2 + y & \rightarrow x \\
\hline
y^2 + y & \\
y^2 - 1 & \\
\hline
y + 1 & \\
\hline
1 & \rightarrow x + y
\end{array}$$

Division Algorithm

The division algorithm is essentially the same as the one in the univariate case but there is a small change which has to be made. To see this, let us look at an example,

$$
\begin{array}{rlr}
a_1: & x + y & \\
a_2: & 1 & r \\
xy + 1 &) \overline{x^2y + xy^2 + y^2} & \underline{\qquad\qquad} \\
y^2 + 1 & & \\
 & x^2y - x & \\
 & \overline{\quad xy^2 + x + y^2} & \\
 & \quad xy^2 - y & \\
 & \overline{\qquad x + y^2 + y} & \to x \\
 & \overline{\qquad\quad y^2 + y} & \\
 & \qquad\quad y^2 - 1 & \\
 & \overline{\qquad\qquad y + 1} & \\
 & \overline{\qquad\qquad\quad 1} & \to x + y \\
 & \overline{\qquad\qquad\quad 0} & \to x + y + 1
\end{array}
$$

Division Algorithm

Algorithm 1: Multi_Divide$(f, f_1, f_2, \cdots f_n)$

$a_1 := 0; a_2 := 0; \cdots a_n := 0; r = 0$
$p := f$
while $p \neq 0$ **do**
    $i := 1$
    divisionoccured := false
    while $i \leq s$ *AND divisionoccured* := *false* **do**
        if $LT(f_i) | p$ **then**
            $a_i := a_i + LT(p)/LT(f_i)$
            $p := p - (LT(p)/LT(f_i))f_i$
            divisionoccured := true
        else
            $i := i + 1$
    if *divisionoccured* := *false* **then**
        $r := r + LT(p)$
        $p := p - LT(p)$
return $a_1, a_2, \cdots, a_n, r$;

Are we done? NO!!

- The natural algorithm to check if f belongs to the ideal generated by f_is would be to check if remainder of $f = 0$ on division with the basis elements.

Are we done? NO!!

- The natural algorithm to check if f belongs to the ideal generated by f_is would be to check if remainder of $f = 0$ on division with the basis elements.
- Although this gives us a sufficient condition, it is not a necessary one. To see this, observe that the output of the algorithm depends on the order of input and the ordering used.

Are we done? NO!!

- The natural algorithm to check if f belongs to the ideal generated by f_is would be to check if remainder of $f = 0$ on division with the basis elements.
- Although this gives us a sufficient condition, it is not a necessary one. To see this, observe that the output of the algorithm depends on the order of input and the ordering used. For example,

$$Multi\_Divide(xy^2 - x, xy + 1, y^2 - 1) = (y, 0, -(x + y))$$
$$Multi\_Divide(xy^2 - x, y^2 - 1, xy + 1) = (y^2 - 1, 0, 0)$$

Are we done? NO!!

- The natural algorithm to check if f belongs to the ideal generated by f_is would be to check if remainder of $f = 0$ on division with the basis elements.
- Although this gives us a sufficient condition, it is not a necessary one. To see this, observe that the output of the algorithm depends on the order of input and the ordering used. For example,

$$Multi\_Divide(xy^2 - x, xy + 1, y^2 - 1) = (y, 0, -(x + y))$$
$$Multi\_Divide(xy^2 - x, y^2 - 1, xy + 1) = (y^2 - 1, 0, 0)$$

- We want to find a "good" basis for a given ideal which preserves the property that nonzero remainder implies non-membership also called the *remainder property*

Are we done? NO!!

- The natural algorithm to check if f belongs to the ideal generated by f_is would be to check if remainder of $f = 0$ on division with the basis elements.
- Although this gives us a sufficient condition, it is not a necessary one. To see this, observe that the output of the algorithm depends on the order of input and the ordering used. For example,

$$Multi\_Divide(xy^2 - x, xy + 1, y^2 - 1) = (y, 0, -(x + y))$$
$$Multi\_Divide(xy^2 - x, y^2 - 1, xy + 1) = (y^2 - 1, 0, 0)$$

- We want to find a "good" basis for a given ideal which preserves the property that nonzero remainder implies non-membership also called the *remainder property*

Does such a basis exist ? Is it computable ?

Gröbner basis

Definition

Fix a monomial order. A finite subset $G = \{g_1, g_2, \cdots, g_n\}$ of an ideal I is said to be a Gröbner basis (or standard basis) if

$$< LT(g_1), LT(g_2) \cdots , LT(g_n) > = < LT(I) >$$

Gröbner basis

Definition

Fix a monomial order. A finite subset $G = \{g_1, g_2, \cdots, g_n\}$ of an ideal I is said to be a Gröbner basis (or standard basis) if

$$< LT(g_1), LT(g_2) \cdots, LT(g_n) > = < LT(I) >$$

Theorem

Let G be a Gröbner basis for an ideal I and let $f \in k[x_1, \cdots, x_n]$. Then there is a unique remainder r on division by G with the following two properties:

1. *No term of r is divisible by any of $LT(g_1), \cdots LT(g_n)$.*
2. *There is $g \in I$ such that $f = g + r$.*

Syzygy Polynomials

Definition

For two monomials $x^\alpha, x^\beta, LCM(x^\alpha, x^\beta) = x^\gamma$ where $\gamma_i = max(\alpha_i, \beta_i)$

Syzygy Polynomials

Definition

For two monomials $x^\alpha, x^\beta, LCM(x^\alpha, x^\beta) = x^\gamma$ where $\gamma_i = max(\alpha_i, \beta_i)$

Definition

If $LCM(LM(f), LM(G)) = x^\gamma$, S-polynomial is defined as,

$$S(f,g) = \frac{x^\gamma}{LT(f)} f - \frac{x^\gamma}{LT(g)} g$$

Syzygy Polynomials

Definition

For two monomials x^α, x^β, $LCM(x^\alpha, x^\beta) = x^\gamma$ where $\gamma_i = max(\alpha_i, \beta_i)$

Definition

If $LCM(LM(f), LM(G)) = x^\gamma$, S-polynomial is defined as,

$$S(f,g) = \frac{x^\gamma}{LT(f)} f - \frac{x^\gamma}{LT(g)} g$$

Lemma

Suppose we have a sum $\sum_{i=1}^{n} c_i f_i$, where $c_i \in k$ and $multideg(f_i) = \alpha$. If $multideg(\sum_{i=1}^{n} c_i f_i) < \alpha$, then

$$\sum_{i=1}^{n} c_i f_i = \sum_{i=1}^{n} c'_{ij} S(f_i, f_j)$$

Buchberger's Criterion

Theorem (Buchberger '65)

Let I be a polynomial ideal. Then a basis $G = g_1, \cdots g_n$ for I is a Gröebner basis for I if and only if for all pairs $i \neq j$, the remainder on division of $S(g_i, g_j)$ by G is zero.

Buchberger's Criterion

Theorem (Buchberger '65)

Let I be a polynomial ideal. Then a basis $G = g_1, \cdots g_n$ for I is a Gröebner basis for I if and only if for all pairs $i \neq j$, the remainder on division of $S(g_i, g_j)$ by G is zero.

Algorithm 3: Buchberger(F)

```
Start with G:= F
do
    G' := G
    for pair of polynomials f1, f2 ∈ G' do
        h := remainder[G, S(f1, f2)]
        if h ≠ 0 then
            G = G ∪ {h}
while G ≠ G';
output G
```

Using Gröner Basis

- **System of polynomials**

Using Gröner Basis

- **System of polynomials** - It can be shown that computing Gröbner basis using the lex ordering gives a basis where the variables are eliminated successively. Also, the order of elimination seems to correspond to the ordering of the variables.

Using Gröner Basis

- **System of polynomials** - It can be shown that computing Gröbner basis using the lex ordering gives a basis where the variables are eliminated successively. Also, the order of elimination seems to correspond to the ordering of the variables.Example, the Gröbner basis corresponding to

$$I = (x^2 + y^2 + z^2 - 1, x^2 + Z^2 - y, x - z)$$

$$G = (x - z, -y + 2z^2, z^4 + \frac{1}{2}z^2 - \frac{1}{4})$$

Using Gröner Basis

- **System of polynomials** - It can be shown that computing Gröbner basis using the lex ordering gives a basis where the variables are eliminated successively. Also, the order of elimination seems to correspond to the ordering of the variables.Example, the Gröbner basis corresponding to

$$I = (x^2 + y^2 + z^2 - 1, x^2 + Z^2 - y, x - z)$$

$$G = (x - z, -y + 2z^2, z^4 + \frac{1}{2}z^2 - \frac{1}{4})$$

- **The Implicitization Problem**

Using Gröner Basis

- **System of polynomials** - It can be shown that computing Gröbner basis using the lex ordering gives a basis where the variables are eliminated successively. Also, the order of elimination seems to correspond to the ordering of the variables.Example, the Gröbner basis corresponding to

$$I = (x^2 + y^2 + z^2 - 1, x^2 + Z^2 - y, x - z)$$

$$G = (x - z, -y + 2z^2, z^4 + \frac{1}{2}z^2 - \frac{1}{4})$$

- **The Implicitization Problem** Similarly, we can eliminate the t variables and the rest of the equations define the ideal we require.

Using Gröner Basis

- **System of polynomials** - It can be shown that computing Gröbner basis using the lex ordering gives a basis where the variables are eliminated successively. Also, the order of elimination seems to correspond to the ordering of the variables.Example, the Gröbner basis corresponding to

$$I = (x^2 + y^2 + z^2 - 1, x^2 + Z^2 - y, x - z)$$

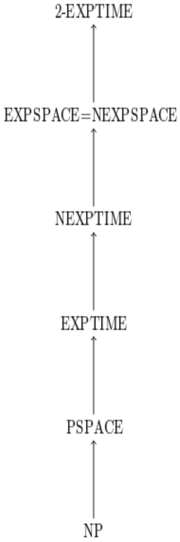
$$G = (x - z, -y + 2z^2, z^4 + \frac{1}{2}z^2 - \frac{1}{4})$$

- **The Implicitization Problem** Similarly, we can eliminate the t variables and the rest of the equations define the ideal we require. Example,

$$I = (t^4 - x, t^3 - y, t^2 - z)$$
$$G = \{t^2 + z, ty - z^2, tz - y, x - z^2, y^2 - z^3\}$$

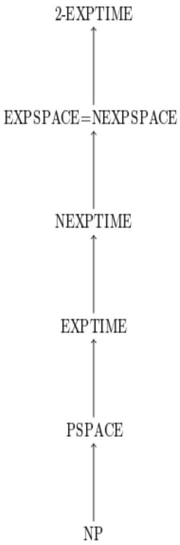
Thus, $(x - z^2, y^2 - z^3)$ is the required ideal.

Complexity



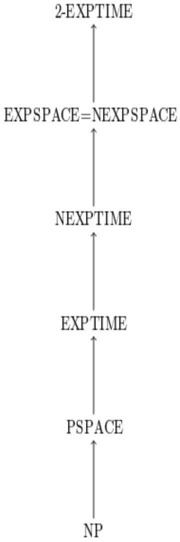
- The worst case time complexity of Buchberger's algorithm is $O(2^{2^n})$ time which restricts its usage.

Complexity



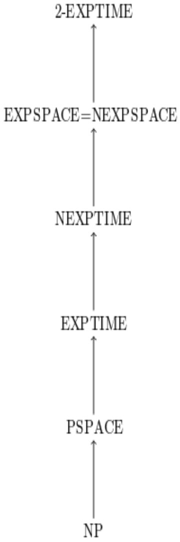
- The worst case time complexity of Buchberger's algorithm is $O(2^{2^n})$ time which restricts its usage.
- Ideal membership problem is EXPSPACE-complete [Mayr-Meyer'82]

Complexity



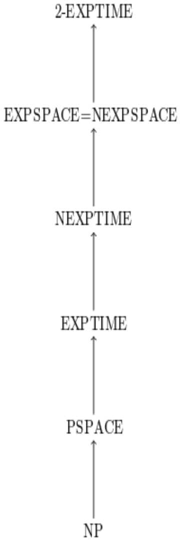
- The worst case time complexity of Buchberger's algorithm is $O(2^{2^n})$ time which restricts its usage.
- Ideal membership problem is EXPSPACE-complete [Mayr-Meyer'82]
- Polynomial System solving is in PSPACE . [Koll´ar'88, Fitchas-Galligo'90]

Complexity



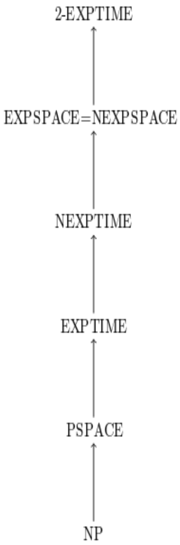
- The worst case time complexity of Buchberger's algorithm is $O(2^{2^n})$ time which restricts its usage.
- Ideal membership problem is EXPSPACE-complete [Mayr-Meyer'82]
- Polynomial System solving is in PSPACE . [Koll´ar'88, Fitchas-Galligo'90]
- However, better algorithms can be constructed for specific purposes. For example, computing a Gröbner basis for the radical of a zero dimensional Ideal takes randomized $O(d)$, deterministic $O(d^n)$ time. [Lakshman '90]

Complexity



- The worst case time complexity of Buchberger's algorithm is $O(2^{2^n})$ time which restricts its usage.
- Ideal membership problem is EXPSPACE-complete [Mayr-Meyer'82]
- Polynomial System solving is in PSPACE . [Koll´ar'88, Fitchas-Galligo'90]
- However, better algorithms can be constructed for specific purposes. For example, computing a Gröbner basis for the radical of a zero dimensional Ideal takes randomized $O(d)$, deterministic $O(d^n)$ time. [Lakshman '90]
- Linear Algebra can also be used to compute Gröbner Basis by using Macaulay Matrices [Macaulay 1902].

Complexity



- The worst case time complexity of Buchberger's algorithm is $O(2^{2^n})$ time which restricts its usage.
- Ideal membership problem is EXPSPACE-complete [Mayr-Meyer'82]
- Polynomial System solving is in PSPACE . [Koll´ar'88, Fitchas-Galligo'90]
- However, better algorithms can be constructed for specific purposes. For example, computing a Gröbner basis for the radical of a zero dimensional Ideal takes randomized $O(d)$, deterministic $O(d^n)$ time. [Lakshman '90]
- Linear Algebra can also be used to compute Gröbner Basis by using Macaulay Matrices [Macaulay 1902].
- Faster Algorithms by Jean-Charles Faugére (F_4, F_5) for a certain (broad) class of systems called *regular sequences* in singly exponential time. Quite fast in the general case as well, used in computer algebra systems.

Applications

- Effective computation with (holonomic) special functions
- Solving Diophantine equations (Pell)
- Automated geometry theorem proving.
- Coding theory
- Signal and image processing
- Robotics
- Graph coloring problems e.g. Sudoku puzzles
- Extrapolating "missing links" in palaeontology, and phylogenetic tree construction

References

Ali Ayad. "A Survey on the Complexity of Solving Algebraic Systems". In: *International Mathematical Forum* 5.7 (2010), pp. 333–353.

Donal O' Shea David Cox John Little. *Ideals, Varieties and Algorithms*. Springer, 2007.

William Fulton. *Algebraic Curves, An Introduction to Algebraic Geometry*. 2008.

Madhu Sudan. "Algebra and Computation". In: (1998).