Fast construction of a lexicographic Gröbner basis of the vanishing ideal of a set of points

Dahan Xavier

Ochanomizu Univeristy, Faculty of General Educational Research

ACA 2017, July 17-21 — High-performance computing

Introduction

- Setting: Let $V \subset \overline{\mathbf{k}}^n$ a finite set of points
  V is Zariski-closed over $\mathbf{k}$: V is the set of solutions of a polynomial system over $\mathbf{k}$.
- Problem: Compute a lexicographic Gröbner basis of the vanishing polynomials on V.
  - Classical problem: Buchberger-Möller (1982) for any monomial order
  - …and for the lex order, dedicated algorithms: 1995 to 2016.

Introduction

- Setting: Let $V \subset \overline{\mathbf{k}}^n$ a finite set of points
  V is Zariski-closed over $\mathbf{k}$: V is the set of solutions of a polynomial system over $\mathbf{k}$.
- Problem: Compute a lexicographic Gröbner basis of the vanishing polynomials on V.
  - Classical problem: Buchberger-Möller (1982) for any monomial order
  - ...and for the lex order, dedicated algorithms: 1995 to 2016.
- Yet, all those works are somewhat still "incomplete". Why ?
  - research articles tend to address some aspects and ignoring some others.
  - for example, fully explicit interpolation formulas have not appear clearly...
  - ...it is a key for a sharp complexity study.

Non-generic LexGB of Dimension Zero

LexGB = Lexicographic Gröbner Basis: $x_1 \prec x_2 \prec \cdots \prec x_n$.

$$
\begin{aligned}
g_{\ell(n)}(x_1, x_2, x_3, \ldots, x_{n-2}, x_{n-1}, x_n) &= x_n^{d_{\ell(n)}} + \cdots \\
g_{\ell(n)-1}(x_1, x_2, \ldots, x_{n-2}, x_{n-1}) &= \mathrm{lc}_{n-1}(g_{\ell(n)-1}) x_n^{d_{\ell(n)-1}} + \ldots \\
\ddots \quad &\vdots \quad \vdots \\
g_{\ell(n-1)}(x_1, \ldots, x_{n-1}) &= x_{n-1}^{d_{\ell(n-1)}} + \cdots \\
\ddots \quad &\vdots \quad \vdots \\
g_{\ell(2)}(x_1, x_2) &= x_2^{d_{\ell(2)}} + \cdots \\
g_{\ell(2)-1}(x_1, x_2) &= x_1^{n_{\ell(2)-1}} x_2^{d_{\ell(2)-1}} + \cdots \\
\ddots \quad &\vdots \quad \vdots \\
g_1(x_1) &= x_1^{d_1} + \cdots
\end{aligned}
$$

This work $\rightarrow$ "Highly" non-generic lexGB : $|\mathcal{G}| > n$.

Non-generic LexGB of Dimension Zero

LexGB = Lexicographic Gröbner Basis: $x_1 \prec x_2 \prec \cdots \prec x_n$.

$$
\begin{array}{rcl}
g_{\ell(n)}(x_1, x_2, x_3, \ldots, x_{n-2}, x_{n-1}, x_n) & = & x_n^{d_{\ell(n)}} + \cdots \\
g_{\ell(n)-1}(x_1, x_2, \ldots, x_{n-2}, x_{n-1}) & = & \mathrm{lc}_{n-1}(g_{\ell(n)-1}) x_n^{d_{\ell(n)-1}} + \ldots \\
\ddots & \vdots & \vdots \\
g_{\ell(n-1)}(x_1, \ldots, x_{n-1}) & = & x_{n-1}^{d_{\ell(n-1)}} + \cdots \\
\ddots & \vdots & \vdots \\
g_{\ell(2)}(x_1, x_2) & = & x_2^{d_{\ell(2)}} + \cdots \\
g_{\ell(2)-1}(x_1, x_2) & = & x_1^{n_{\ell(2)-1}} x_2^{d_{\ell(2)-1}} + \cdots \\
\ddots & \vdots & \vdots \\
g_1(x_1) & = & x_1^{d_1} + \cdots
\end{array}
$$

This work $\rightarrow$ “Highly” non-generic lexGB : $|\mathcal{G}| > n$.
Non-generic: Shape Lemma, Triangular Set $\rightarrow$ nothing new.

Results

Let $D_i := |V_{\leq i}|$ where $V_{\leq i} = \pi_i(V)$,
$\pi_i : \overline{\mathbf{k}}^n \to \overline{\mathbf{k}}^i, (a_1, \ldots, a_n) \mapsto (a_1, \ldots, a_i)$

1. There is a Gröbner basis $\mathcal{G}'$, non-reduced in general, such that any polynomial $g \in \mathcal{G}'$ can be computed in at most:

$$O(\mathrm{A}(D_1) + \mathrm{A}(D_2) + \cdots + \mathrm{A}(D_n)) < O(nD_n),$$

arithmetic operations in $\mathbf{k}$.

- $\mathrm{A}(d)$ cost to construct Lagrange idempotents of d points.
- $\mathrm{A}(d) = \mathrm{M}(d)\log(d)$, by the subproduct tree technique. ($\mathrm{M}(d) = O(d\log(d)\log\log(d))$ by Schönhage-Strassen, or naively d^2).

Results

Let $D_i := |V_{\leq i}|$ where $V_{\leq i} = \pi_i(V)$,
$\pi_i : \overline{\mathbf{k}}^n \to \overline{\mathbf{k}}^i, (a_1, \ldots, a_n) \mapsto (a_1, \ldots, a_i)$

1. There is a Gröbner basis $\mathcal{G}'$, non-reduced in general, such that any polynomial $g \in \mathcal{G}'$ can be computed in at most:

$$O(\mathrm{A}(D_1) + \mathrm{A}(D_2) + \cdots + \mathrm{A}(D_n)) < O(nD_n),$$

   arithmetic operations in $\mathbf{k}$.
   - $\mathrm{A}(d)$ cost to construct Lagrange idempotents of d points.
   - $\mathrm{A}(d) = \mathrm{M}(d)\log(d)$, by the subproduct tree technique. ($\mathrm{M}(d) = O(d\log(d)\log\log(d))$ by Schönhage-Strassen, or naively d^2).
2. The structure of (non-generic) lexGB allows to recycle computations.
   - But difficult to estimate in general. Simple strategy is still a work in progress.

Previous Work

- Buchberger-Möller (1982): linear algebra $O(nD_n^3)$ (but for any monomial order)

Previous Work

- Buchberger-Möller (1982): linear algebra $O(nD_n^3)$ (but for any monomial order)
- Lederer (2008): almost "fully" explicit formulas, no complexity at all.
  - focuses on the reduced lexGB $\mathcal{G}$ which complicates the matter quite a lot.
  - essentially computes the above non-reduced lexGB $\mathcal{G}'$, stops half-way, then withdraw linear combinations of other polynomials built "on-demand" to cancel unwanted monomials.

Previous Work

- Buchberger-Möller (1982): linear algebra $O(nD_n^3)$ (but for any monomial order)
- Lederer (2008): almost "fully" explicit formulas, no complexity at all.
  - focuses on the reduced lexGB $\mathcal{G}$ which complicates the matter quite a lot.
  - essentially computes the above non-reduced lexGB $\mathcal{G}'$, stops half-way, then withdraw linear combinations of other polynomials built "on-demand" to cancel unwanted monomials.
- Finding a separating basis: $\forall v \in V,\ p_v \in k[x_1, \ldots, x_n]$, such that $p_v(w) = 0$ if $v \neq w$, and $p_v(v) = 1$ otherwise.

Previous Work

- Buchberger-Möller (1982): linear algebra $O(nD_n^3)$ (but for any monomial order)
- Lederer (2008): almost "fully" explicit formulas, no complexity at all.
  - focuses on the reduced lexGB $\mathcal{G}$ which complicates the matter quite a lot.
  - essentially computes the above non-reduced lexGB $\mathcal{G}'$, stops half-way, then withdraw linear combinations of other polynomials built "on-demand" to cancel unwanted monomials.
- Finding a separating basis: $\forall v \in V$, $p_v \in k[x_1, \ldots, x_n]$, such that $p_v(w) = 0$ if $v \neq w$, and $p_v(v) = 1$ otherwise.
  - Lundqvist (2010): $O(D^2)$ But using subproduct tree $O(D \log(D))$.

Previous Work

- Buchberger-Möller (1982): linear algebra $O(nD_n^3)$ (but for any monomial order)
- Lederer (2008): almost “fully” explicit formulas, no complexity at all.
  - focuses on the reduced lexGB $\mathcal{G}$ which complicates the matter quite a lot.
  - essentially computes the above non-reduced lexGB $\mathcal{G}'$, stops half-way, then withdraw linear combinations of other polynomials built “on-demand” to cancel unwanted monomials.
- Finding a separating basis: $\forall v \in V,\ p_v \in k[x_1, \ldots, x_n]$, such that $p_v(w) = 0$ if $v \neq w$, and $p_v(v) = 1$ otherwise.
  - Lundqvist (2010): $O(D^2)$ But using subproduct tree $O(D \log(D))$.
  - Lei-Teng-Ren (2016): lexGB for vanishing polynomials of higher order (Hermite interpolation). $(O(\tau + 3)D^2)$ where τ is a dislacement rank.

Previous Work

- Buchberger-Möller (1982): linear algebra $O(nD_n^3)$ (but for any monomial order)
- Lederer (2008): almost "fully" explicit formulas, no complexity at all.
    - focuses on the reduced lexGB $\mathcal{G}$ which complicates the matter quite a lot.
    - essentially computes the above non-reduced lexGB $\mathcal{G}'$, stops half-way, then withdraw linear combinations of other polynomials built "on-demand" to cancel unwanted monomials.
- Finding a separating basis: $\forall v \in V$, $p_v \in k[x_1, \ldots, x_n]$, such that $p_v(w) = 0$ if $v \neq w$, and $p_v(v) = 1$ otherwise.
    - Lundqvist (2010): $O(D^2)$ But using subproduct tree $O(D\log(D))$.
    - Lei-Teng-Ren (2016): lexGB for vanishing polynomials of higher order (Hermite interpolation). $(O(\tau+3)D^2)$ where τ is a dislacement rank. Vandermonde matrix, $\tau = 2 \to O(D^2)$

The lextree: introduction and backgrounds

A key tool to study lexGB is a combinatorial decomposition of V:
One-one correspondence between standard monomials of $\mathcal{G}$ and points of V.

- Macaulay? Lazard in two variables.
- Cerlienco-Muredu (1995 & 01), Marinari Mora (2003 & 06): linear algebra.

The lextree: introduction and backgrounds

A key tool to study lexGB is a combinatorial decomposition of V:
One-one correspondence between standard monomials of $\mathcal{G}$ and points of V.

- Macaulay? Lazard in two variables.
- Cerlienco-Muredu (1995 & 01), Marinari Mora (2003 & 06): linear algebra.
- Lex game (Ronyai et al., 2006) Lextree: Major simplification by using a tree data structure.

The lextree: introduction and backgrounds

A key tool to study lexGB is a combinatorial decomposition of V:
One-one correspondence between standard monomials of $\mathcal{G}$ and points of V.

- Macaulay? Lazard in two variables.
- Cerlienco-Muredu (1995 & 01), Marinari Mora (2003 & 06): linear algebra.
- Lex game (Ronyai et al., 2006) Lextree: Major simplification by using a tree data structure.
- Lederer (2008): Instead of lextree uses a "four-in-a-row"-like operation on the standard monomials.

The lextree: introduction and backgrounds

A key tool to study lexGB is a combinatorial decomposition of V:
One-one correspondence between standard monomials of $\mathcal{G}$ and points of V.

- Macaulay? Lazard in two variables.
- Cerlienco-Muredu (1995 & 01), Marinari Mora (2003 & 06): linear algebra.
- Lex game (Ronyai et al., 2006) Lextree: Major simplification by using a tree data structure.
- Lederer (2008): Instead of lextree uses a “four-in-a-row”-like operation on the standard monomials.
- Lundqvist (2010): complexity of building the lextree to find standard monomials.

The lextree: introduction and backgrounds

A key tool to study lexGB is a combinatorial decomposition of V:
One-one correspondence between standard monomials of $\mathcal{G}$ and points of V.

- Macaulay? Lazard in two variables.
- Cerlienco-Muredu (1995 & 01), Marinari Mora (2003 & 06): linear algebra.
- Lex game (Ronyai et al., 2006) Lextree: Major simplification by using a tree data structure.
- Lederer (2008): Instead of lextree uses a “four-in-a-row”-like operation on the standard monomials.
- Lundqvist (2010): complexity of building the lextree to find standard monomials.

One-one correspondence between standard monomials and leaves

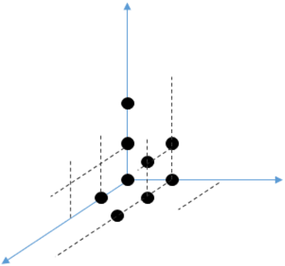
Lextree II

More than standard monomials, we are interested in leading monomials of a lexGB.

We introduce a new variation of computing standard monomials to compute leading exponents:

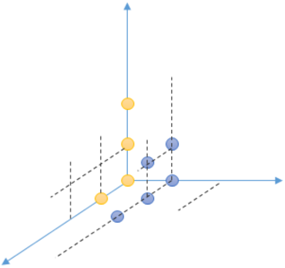
Example interlude

Lextree: construction

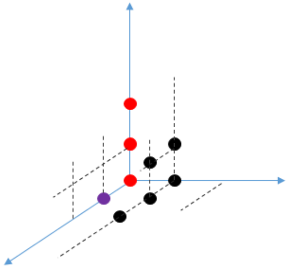


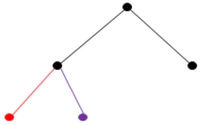
● root

Lextree: construction

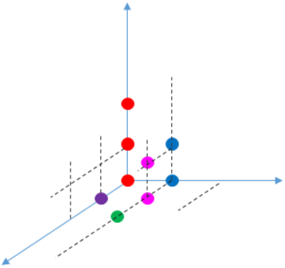


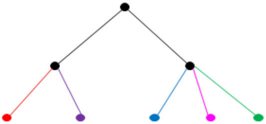
Lextree: construction



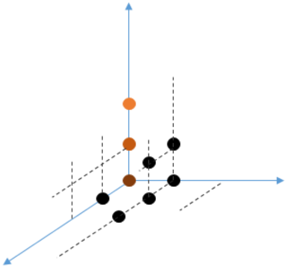


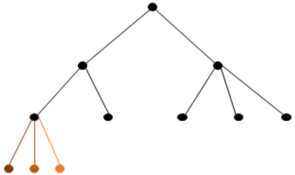
Lextree: construction



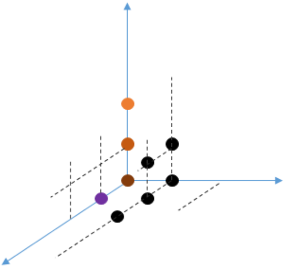


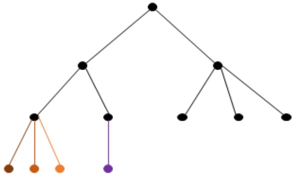
Lextree: construction



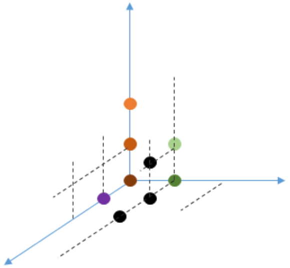


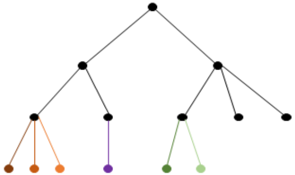
Lextree: construction



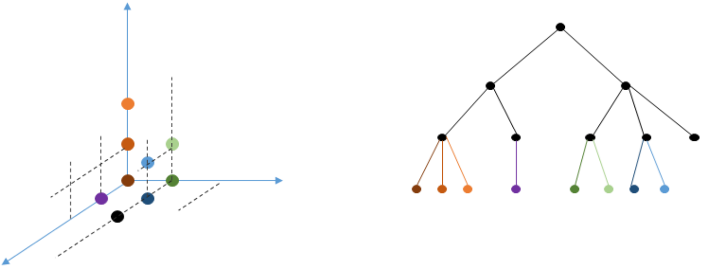


Lextree: construction

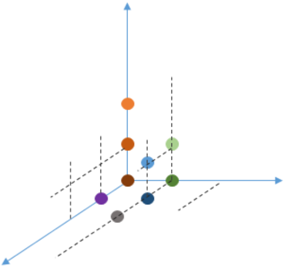


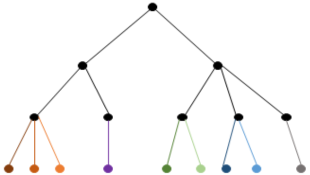


Lextree: construction

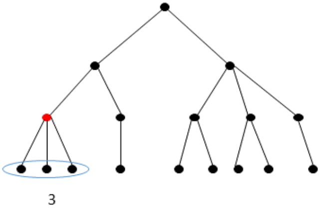


Lextree: From leaves to exponents in the GB

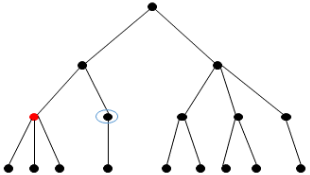




Lextree: From leaves to exponents in the GB

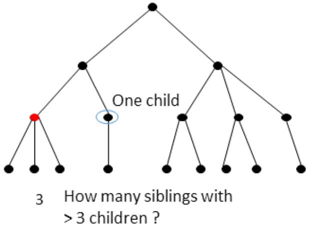


Lextree: From leaves to exponents in the GB

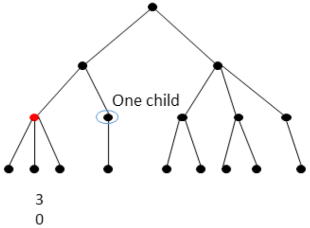


3 How many siblings with
> 3 children ?

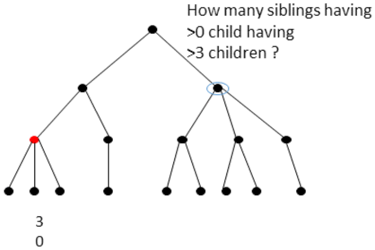
Lextree: From leaves to exponents in the GB



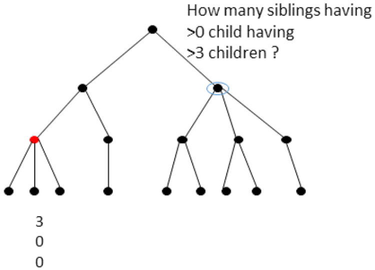
Lextree: From leaves to exponents in the GB



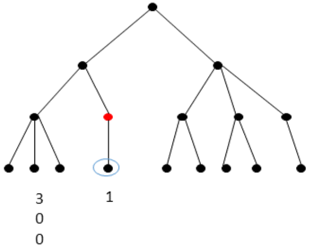
Lextree: From leaves to exponents in the GB



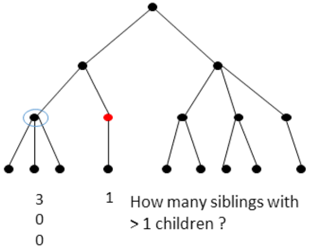
Lextree: From leaves to exponents in the GB



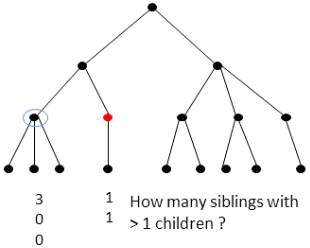
Lextree: From leaves to exponents in the GB



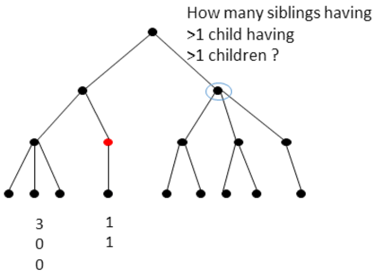
Lextree: From leaves to exponents in the GB



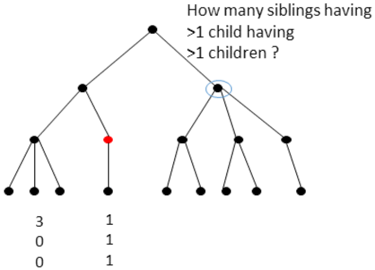
Lextree: From leaves to exponents in the GB



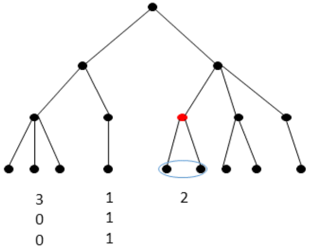
Lextree: From leaves to exponents in the GB



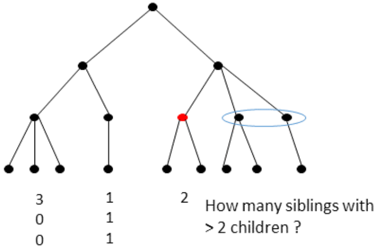
Lextree: From leaves to exponents in the GB



Lextree: From leaves to exponents in the GB

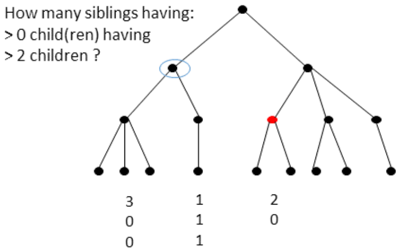


Lextree: From leaves to exponents in the GB

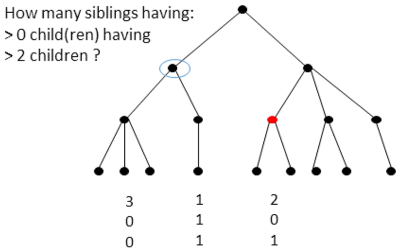


Lextree: From leaves to exponents in the GB

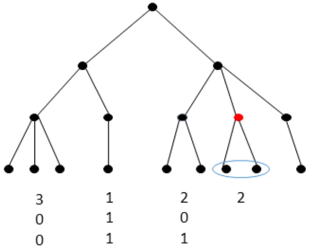
Lextree: From leaves to exponents in the GB



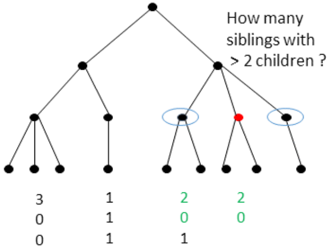
Lextree: From leaves to exponents in the GB



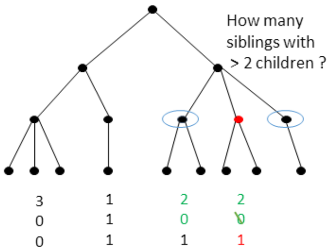
Lextree: From leaves to exponents in the GB



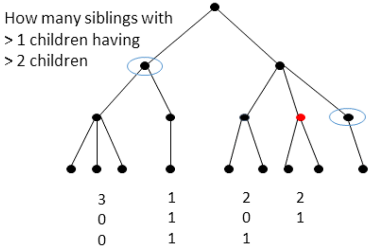
Lextree: From leaves to exponents in the GB



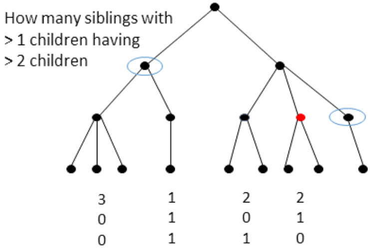
Lextree: From leaves to exponents in the GB



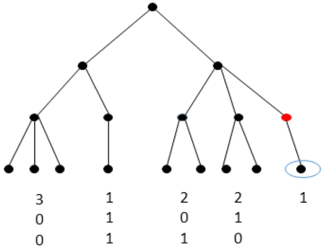
Lextree: From leaves to exponents in the GB



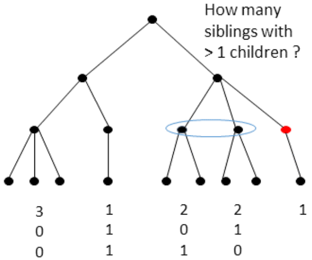
Lextree: From leaves to exponents in the GB



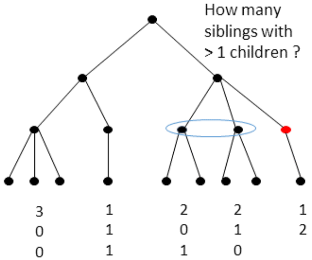
Lextree: From leaves to exponents in the GB



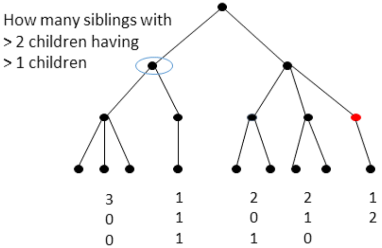
Lextree: From leaves to exponents in the GB



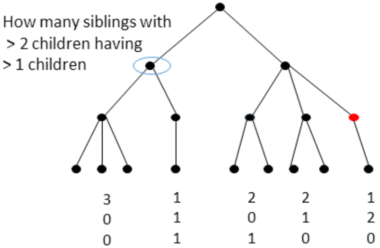
Lextree: From leaves to exponents in the GB



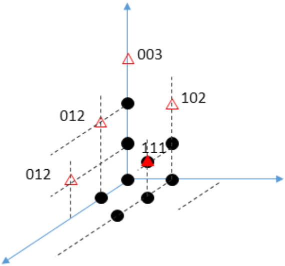
Lextree: From leaves to exponents in the GB

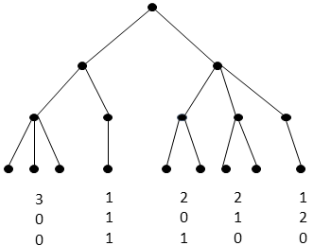


Lextree: From leaves to exponents in the GB

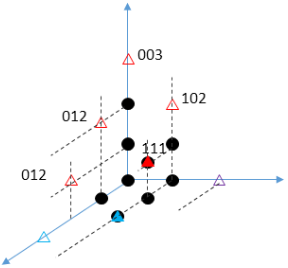


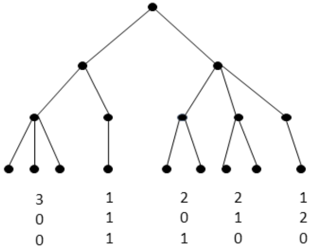
Lextree: From leaves to exponents in the GB



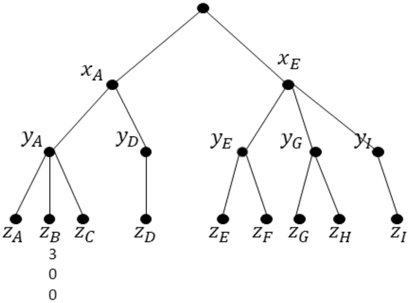


Lextree: From leaves to exponents in the GB

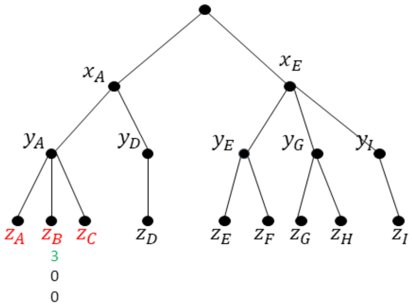




Lextree: From leaves to interpolation

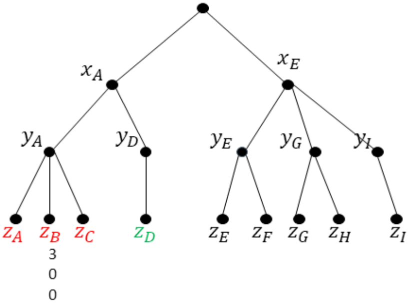


Lextree: From leaves to interpolation



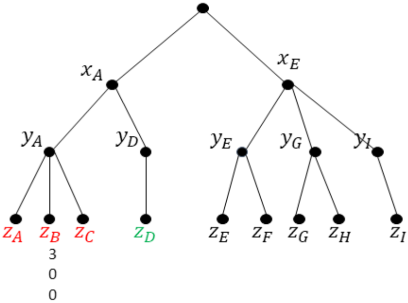
$(z - z_A)(z - z_B)(z - z_C)$ z^3

Lextree: From leaves to interpolation



$$(z-z_A)(z-z_B)(z-z_C)\,\frac{y-y_D}{y_A-y_D} + (z-z_D)\,\frac{y-y_A}{y_D-y_A}$$

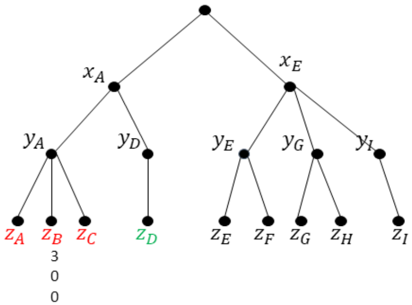
Lextree: From leaves to interpolation



$$(z - z_A)(z - z_B)(z - z_C)\,\frac{y - y_D}{y_A - y_D} + (z - z_D)\,\frac{y - y_A}{y_D - y_A}$$

Lead. Mon. $\neq z^3$

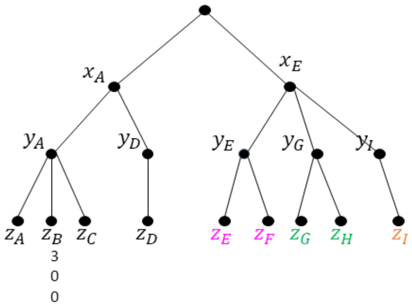
Lextree: From leaves to interpolation



$$(z - z_A)(z - z_B)(z - z_C)\,\frac{y - y_D}{y_A - y_D} + z^2(z - z_D)\,\frac{y - y_A}{y_D - y_A}$$

Lead. Mon. $= z^3$

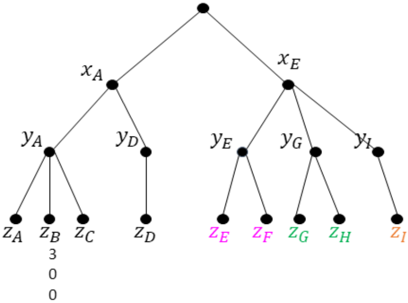
Lextree: From leaves to interpolation



$(z - z_E)(z - z_F)\, z \quad + z(z - z_G)(z - z_H) \quad +$

$z^2(z - z_I)$

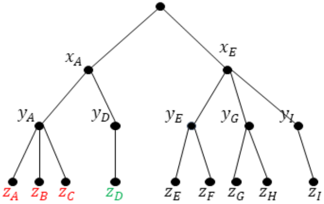
Lextree: From leaves to interpolation



$$(z - z_E)(z - z_F)\, z \frac{y-y_G}{y_E-y_G}\frac{y-y_I}{y_E-y_I} + z(z - z_G)(z - z_H)\frac{y-y_G}{y_E-y_G}\frac{y-y_I}{y_E-y_I} + z^2(z - z_I)\frac{y-y_G}{y_E-y_G}\frac{y-y_I}{y_E-y_I}$$

Lead. Mon. is z^3

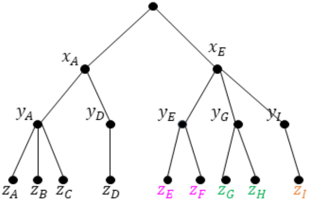
Lextree: From leaves to interpolation



3
0
0

$$(z - z_A)(z - z_B)(z - z_C)\,\frac{y - y_D}{y_A - y_D} + z^2(z - z_D)\,\frac{y - y_A}{y_D - y_A}$$

Lead. Mon. $= z^3$



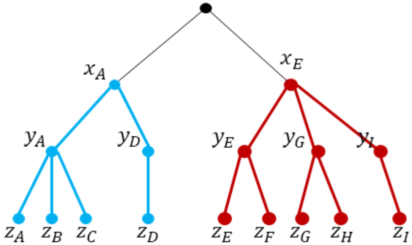
$$(z - z_E)(z - z_F)\,z\,\frac{y - y_G}{y_E - y_G}\,\frac{y - y_I}{y_E - y_I} +$$

$$z(z - z_G)(z - z_H)\,\frac{y - y_G}{y_E - y_G}\,\frac{y - y_I}{y_E - y_I} +$$

$$z^2(z - z_I)\,\frac{y - y_G}{y_E - y_G}\,\frac{y - y_I}{y_E - y_I}$$

Lead. Mon. is z^3

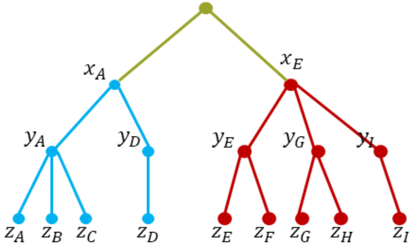
Lextree: From leaves to interpolation



$$(z-z_E)(z-z_F)\, z\, \frac{y-y_G}{y_E-y_G}\frac{y-y_I}{y_E-y_I} + z(z-z_G)(z-z_H)\frac{y-y_G}{y_E-y_G}\frac{y-y_I}{y_E-y_I} + z^2(z-z_I)\frac{y-y_G}{y_E-y_G}\frac{y-y_I}{y_E-y_I}$$

$$(z-z_A)(z-z_B)(z-z_C)\frac{y-y_D}{y_A-y_D} + z^2(z-z_D)\frac{y-y_A}{y_D-y_A}$$

Lextree: From leaves to interpolation



$$\left[(z-z_E)(z-z_F)\,z\,\frac{y-y_G}{y_E-y_G}\frac{y-y_I}{y_E-y_I} + z(z-z_G)(z-z_H)\frac{y-y_G}{y_E-y_G}\frac{y-y_I}{y_E-y_I} + z^2(z-z_I)\frac{y-y_G}{y_E-y_G}\frac{y-y_I}{y_E-y_I}\right] \times \frac{x-x_A}{x_E-x_A}$$

$$+ \left[(z-z_A)(z-z_B)(z-z_C)\frac{y-y_D}{y_A-y_D} + z^2(z-z_D)\frac{y-y_A}{y_D-y_A}\right] \times \frac{x-x_E}{x_A-x_E}$$

Lead. Mon. is z^3

Relation with interpolation: account of the current status

All proofs of the correspondence $\{\text{leaves}\} \leftrightarrow \{\text{std. mononmials}\}$ rely on some sort of interpolation formulas, more or less explicit.

Relation with interpolation: account of the current status

All proofs of the correspondence $\{\text{leaves}\} \leftrightarrow \{\text{std. mononmials}\}$ rely on some sort of interpolation formulas, more or less explicit.

$\rightarrow$ construct polynomials that vanish on some parts of V, and whose leading monomials are standard monomials.

- "Hidden" in case of purely linear algeba method (Cerlienco-Mureddu, Marinari-Mora, Lei-Teng-Ren)

Relation with interpolation: account of the current status

All proofs of the correspondence {leaves} $\leftrightarrow$ {std. mononmials} rely on some sort of interpolation formulas, more or less explicit.

$\rightarrow$ construct polynomials that vanish on some parts of V, and whose leading monomials are standard monomials.

- "Hidden" in case of purely linear algeba method (Cerlienco-Mureddu, Marinari-Mora, Lei-Teng-Ren)
- Explicit: (Lederer,2008) recursive foormula. (Lexgame, 2006): quite coarse.

Relation with interpolation: account of the current status

All proofs of the correspondence {leaves} $\leftrightarrow$ {std. mononmials} rely on some sort of interpolation formulas, more or less explicit.

$\rightarrow$ construct polynomials that vanish on some parts of V, and whose leading monomials are standard monomials.

- "Hidden" in case of purely linear algeba method (Cerlienco-Mureddu, Marinari-Mora, Lei-Teng-Ren)
- Explicit: (Lederer,2008) recursive foormula. (Lexgame, 2006): quite coarse.
- Here: expliciteness of the interpolation formula for polynomials in $\mathcal{G}'$
  - $\rightarrow$ Exploit fully the simplcity supplied by the lextree.
  - $\Rightarrow$ Prone to complexity analysis.

Relation with interpolation: account of the current status

All proofs of the correspondence {leaves} $\leftrightarrow$ {std. mononmials} rely on some sort of interpolation formulas, more or less explicit.

$\rightarrow$ construct polynomials that vanish on some parts of V, and whose leading monomials are standard monomials.

- "Hidden" in case of purely linear algeba method (Cerlienco-Mureddu, Marinari-Mora, Lei-Teng-Ren)
- Explicit: (Lederer,2008) recursive foormula. (Lexgame, 2006): quite coarse.
- Here: expliciteness of the interpolation formula for polynomials in $\mathcal{G}'$
  - $\rightarrow$ Exploit fully the simplcity supplied by the lextree.
  - $\Rightarrow$ Prone to complexity analysis.

Easy Fact: algebraic complexity depends only on the shape of the tree (number of children of nodes) and not on labels at each node.

Interpolating & Discarding points in the lextree

Given an exponent at a leaf $\mathbf{x^e}$, from (parent of the) leaf to the root, perform bottom-up test on siblings of the current node to identify:

1. branches that must be interpolated:
2. branches that must be discarded:

   the branch has a leaf of larger exponent than the current one.

Interpolating & Discarding points in the lextree

Given an exponent at a leaf $\mathbf{x}^{\mathbf{e}}$, from (parent of the) leaf to the root, perform bottom-up test on siblings of the current node to identify:

1. branches that must be interpolated:
   - Lagrange interpolation with eventually additional monomials to guarantee that the leading monomial is 1.
2. branches that must be discarded:

   the branch has a leaf of larger exponent than the current one.

   - Remark: The exponent $\mathbf{x}^{\mathbf{e}}$ comes from polynomials that discard some branches at each level.

Interpolating & Discarding points in the lextree

Given an exponent at a leaf $\mathbf{x}^{\mathbf{e}}$, from (parent of the) leaf to the root, perform bottom-up test on siblings of the current node to identify:

1. branches that must be interpolated:
   - Lagrange interpolation with eventually additional monomials to guarantee that the leading monomial is 1.

2. branches that must be discarded:

   the branch has a leaf of larger exponent than the current one.
   - simple product over the labels at those siblings.
   - Remark: The exponent $\mathbf{x}^{\mathbf{e}}$ comes from polynomials that discard some branches at each level.

Interpolating & Discarding points in the lextree

Given an exponent at a leaf $\mathbf{x}^{\mathbf{e}}$, from (parent of the) leaf to the root, perform bottom-up test on siblings of the current node to identify:

1. branches that must be interpolated:
   - Lagrange interpolation with eventually additional monomials to guarantee that the leading monomial is 1.

   At a given level ℓ : $\leq O(\mathrm{A}(D_1) + \mathrm{A}(D_2) + \cdots + \mathrm{A}(D_\ell))$

2. branches that must be discarded:

   the branch has a leaf of larger exponent than the current one.
   - simple product over the labels at those siblings.
   - Remark: The exponent $\mathbf{x}^{\mathbf{e}}$ comes from polynomials that discard some branches at each level.

Interpolating & Discarding points in the lextree

Given an exponent at a leaf $\mathbf{x}^{\mathbf{e}}$, from (parent of the) leaf to the root, perform bottom-up test on siblings of the current node to identify:

1. branches that must be interpolated:
   - Lagrange interpolation with eventually additional monomials to guarantee that the leading monomial is 1.

   At a given level ℓ : $\leq O(\mathsf{A}(D_1) + \mathsf{A}(D_2) + \cdots + \mathsf{A}(D_\ell))$

2. branches that must be discarded:

   the branch has a leaf of larger exponent than the current one.
   - simple product over the labels at those siblings.
   - Remark: The exponent $\mathbf{x}^{\mathbf{e}}$ comes from polynomials that discard some branches at each level.

   At a given level ℓ: $\leq O(\mathsf{A}(D_\ell))$

Non-reduced Gröbner basis $\mathcal{G}'$? Bit-size

- . . . but minimal: $\textsc{lm}(\mathcal{G}) = \{\textsc{lm}(g) \mid g \in \mathcal{G}\} = \textsc{lm}(\mathcal{G}')$.
- Bad: More coefficients . . . ,
  Good: Coefficients of smaller bit-size

Non-reduced Gröbner basis $\mathcal{G}'$? Bit-size

- . . . but minimal: $\text{LM}(\mathcal{G}) = \{\text{LM}(g) \mid g \in \mathcal{G}\} = \text{LM}(\mathcal{G}')$.
- Bad: More coefficients . . . ,
  Good: Coefficients of smaller bit-size
- More precisely let $h(g)$ be “roughly” the max bit-size among all coefficients on $g \in GB'$:

$$h(g) \leq O(nD^2 h_{pts}^2)$$

  where h_{pts} is the max bit-size among coordinates of all points in V.
- all in all, this is not a bad choice. . . it has moreover good properties. . .

Non-reduced Gröbner basis $\mathcal{G}'$? Stability I

- Let $\phi_{\mathbf{a}} : \overline{\mathbf{k}}[x_1, \ldots, x_n] \to \overline{\mathbf{k}}[x_{m+1}, \ldots, x_n]$, $m < n$ evaluation map at $\mathbf{a} = (a_1, \ldots, a_m)$.
- an ideal I is stable under $\phi_{\mathbf{a}}$ if $\mathrm{LM}(\phi(I)) = \phi_{\mathbf{a}}(\mathrm{LT}_m(I))$.

Non-reduced Gröbner basis $\mathcal{G}'$? Stability I

- Let $\phi_{\mathbf{a}} : \overline{\mathbf{k}}[x_1, \ldots, x_n] \to \overline{\mathbf{k}}[x_{m+1}, \ldots, x_n]$, $m < n$ evaluation map at $\mathbf{a} = (a_1, \ldots, a_m)$.
- an ideal I is stable under $\phi_{\mathbf{a}}$ if $\mathrm{LM}(\phi(I)) = \phi_{\mathbf{a}}(\mathrm{LT}_m(I))$.
- A Gröbner basis G of I for an m-elimination order $\prec_m$ specializes well at $\mathbf{a}$:

$$\phi_{\mathbf{a}}(G) \quad \text{is a Gröbner basis of} \quad \phi_{\mathbf{a}}(I).$$

Non-reduced Gröbner basis $\mathcal{G}'$? Stability I

- Let $\phi_{\mathbf{a}} : \overline{\mathbf{k}}[x_1, \ldots, x_n] \to \overline{\mathbf{k}}[x_{m+1}, \ldots, x_n]$, $m < n$ evaluation map at $\mathbf{a} = (a_1, \ldots, a_m)$.
- an ideal I is stable under $\phi_{\mathbf{a}}$ if $\mathrm{LM}(\phi(I)) = \phi_{\mathbf{a}}(\mathrm{LT}_m(I))$.
- A Gröbner basis G of I for an m-elimination order $\prec_m$ specializes well at $\mathbf{a}$:

$$\phi_{\mathbf{a}}(G) \quad \text{is a Gröbner basis of} \quad \phi_{\mathbf{a}}(I).$$

- Stronger condition is: $\mathrm{LT}(\phi_{\mathbf{a}}(g)) \prec_m \phi_{\mathbf{a}}(\mathrm{LT}_m(g))$ then $\phi_{\mathbf{a}}(g) = 0$.

Non-reduced Gröbner basis $\mathcal{G}'$? Stability I

- Let $\phi_{\mathbf{a}} : \overline{\mathbf{k}}[x_1, \ldots, x_n] \to \overline{\mathbf{k}}[x_{m+1}, \ldots, x_n]$, $m < n$ evaluation map at $\mathbf{a} = (a_1, \ldots, a_m)$.
- an ideal I is stable under $\phi_{\mathbf{a}}$ if $\mathrm{LM}(\phi(I)) = \phi_{\mathbf{a}}(\mathrm{LT}_m(I))$.
- A Gröbner basis G of I for an m-elimination order $\prec_m$ specializes well at $\mathbf{a}$:

$$\phi_{\mathbf{a}}(G) \quad \text{is a Gröbner basis of} \quad \phi_{\mathbf{a}}(I).$$

- Stronger condition is: $\mathrm{LT}(\phi_{\mathbf{a}}(g)) \prec_m \phi_{\mathbf{a}}(\mathrm{LT}_m(g))$ then $\phi_{\mathbf{a}}(g) = 0$.
  - whereas stability is: $\mathrm{LT}(\phi_{\mathbf{a}}(g)) \prec_m \phi_{\mathbf{a}}(\mathrm{LT}_m(g)) \Rightarrow \phi_{\mathbf{a}}(g) \equiv 0 \bmod \phi_{\mathbf{a}}(G \setminus \{g\})$.

Non-reduced Gröbner basis $\mathcal{G}'$? Stability II

Previous work: Stability for *m*-elimination order (includes lex order)

- Gianni (Kalkbrener): $m = n - 1$ for 0-dimensional ideals. Strong version of stability.

Non-reduced Gröbner basis $\mathcal{G}'$? Stability II

Previous work: Stability for m-elimination order (includes lex order)

- Gianni (Kalkbrener): $m = n - 1$ for 0-dimensional ideals. Strong version of stability.
- Becker: Stability for radical m-elimination Gröbner bases. (not strong).

Non-reduced Gröbner basis $\mathcal{G}'$? Stability II

Previous work: Stability for m-elimination order (includes lex order)

- Gianni (Kalkbrener): $m = n - 1$ for 0-dimensional ideals. Strong version of stability.
- Becker: Stability for radical m-elimination Gröbner bases. (not strong).
- Kalkbrener: extension and clarification to the notion of stability.

Non-reduced Gröbner basis $\mathcal{G}'$? Stability II

Previous work: Stability for m-elimination order (includes lex order)

- Gianni (Kalkbrener): $m = n - 1$ for 0-dimensional ideals. Strong version of stability.
- Becker: Stability for radical m-elimination Gröbner bases. (not strong).
- Kalkbrener: extension and clarification to the notion of stability.
- Gröbner basis $\mathcal{G}'$ of this talk: strong stability property.

Non-reduced Gröbner basis $\mathcal{G}'$? Stability II

Previous work: Stability for m-elimination order (includes lex order)

- Gianni (Kalkbrener): $m = n - 1$ for 0-dimensional ideals. Strong version of stability.
- Becker: Stability for radical m-elimination Gröbner bases. (not strong).
- Kalkbrener: extension and clarification to the notion of stability.
- Gröbner basis $\mathcal{G}'$ of this talk: strong stability property.
- Not the case of the reduced Gröbner basis $\mathcal{G}$.

Summary of the interpolation of one polynomial

Step 1. Identify leaves that yields a leading monomial in the Gröbner basis.
$\rightarrow$ purely combinatorial (complexity: only comparisons)

Summary of the interpolation of one polynomial

Step 1. Identify leaves that yields a leading monomial in the Gröbner basis.
$\rightarrow$ purely combinatorial (complexity: only comparisons)

Step 2. Bottom-up interpolation or discard sibling branches.

- This creates an arithmetic circuit. It depends only on the shape of the tree.
- In both case, subproduct tree can be used
$\rightarrow$ many times similar products must be perform.
- About the upper bound an arithmetic complexity $\rightarrow$
Can we do better ? Reuse already computed polynomials.

Recycling already computed components

The more polynomials in $\mathcal{G}'$, have been computed the more it is likely possible to recycle

Two ways to recycle:

1. Detect a product already computed in a subproduct

Recycling already computed components

The more polynomials in $\mathcal{G}'$, have been computed the more it is likely possible to recycle
Two ways to recycle:

1. Detect a product already computed in a subproduct
2. Use structure: some polynomials naturally divides other

Recycling already computed components

The more polynomials in $\mathcal{G}'$, have been computed the more it is likely possible to recycle

Two ways to recycle:

1. Detect a product already computed in a subproduct
   - Too much memory? Is it possible to know in advance which products to store ?
2. Use structure: some polynomials naturally divides other

Recycling already computed components

The more polynomials in $\mathcal{G}'$, have been computed the more it is likely possible to recycle

Two ways to recycle:

1. Detect a product already computed in a subproduct
    - Too much memory? Is it possible to know in advance which products to store ?
2. Use structure: some polynomials naturally divides other
    - Work in progress: structural results. . .

Work in progress – Structure

Assume that $f \in \mathcal{G}'$, with $\mathrm{LM}(g) = x_1^{d_1} \cdots x_n^{d_n}$.

$$f = \sum_{\alpha \in A} \mathcal{L}_\alpha(x_1, \ldots, x_{n-1}) f_{1,\alpha}(x_1) \cdots f_{n,\alpha}(x_n) \cdot m_\alpha$$

- where $f_{i,\alpha}$ depends only on the $i-1$-th first coordinates $(\alpha_1, \ldots, \alpha_{i-1})$ of α,
- $\mathcal{L}_\alpha$ is a multivariate Lagrange idempotent on a grid of points $A \subset V$.
- $m_\alpha = x_1^{d_1 - \delta_1(\alpha)} \cdots x_n^{d_n - \delta_n(\alpha)}$ $(\delta_i(\alpha) = \deg_i(f_{i,\alpha}))$

Work in progress – Structure

Assume that $f \in \mathcal{G}'$, with $\mathrm{LM}(g) = x_1^{d_1} \cdots x_n^{d_n}$.

$$f = \sum_{\alpha \in A} \mathcal{L}_\alpha(x_1, \ldots, x_{n-1}) f_{1,\alpha}(x_1) \cdots f_{n,\alpha}(x_n) \cdot m_\alpha$$

- where $f_{i,\alpha}$ depends only on the $i-1$-th first coordinates $(\alpha_1, \ldots, \alpha_{i-1})$ of α,
- $\mathcal{L}_\alpha$ is a multivariate Lagrange idempotent on a grid of points $A \subset V$.
- $m_\alpha = x_1^{d_1-\delta_1(\alpha)} \cdots x_n^{d_n-\delta_n(\alpha)}$ $(\delta_i(\alpha) = \deg_i(f_{i,\alpha}))$

Kind of generalization of Lazard's structural theorem (1985) full result for lexGB in two variables.

Perspective

1. Find a simple formulation of the previous problems.

Perspective

1. Find a simple formulation of the previous problems.
   - find a simple way, on the lextree, to address a (sharp) complexity analysis of the recycling phase.

Perspective

1. Find a simple formulation of the previous problems.
   - find a simple way, on the lextree, to address a (sharp) complexity analysis of the recycling phase.
2. The ideal of vanishing polynomials is radical.

Perspective

1. Find a simple formulation of the previous problems.
   - find a simple way, on the lextree, to address a (sharp) complexity analysis of the recycling phase.
2. The ideal of vanishing polynomials is radical.
   - Generalization to non-radical ideals: Hermite conditions attached to each point in V.

Perspective

1. Find a simple formulation of the previous problems.
   - find a simple way, on the lextree, to address a (sharp) complexity analysis of the recycling phase.
2. The ideal of vanishing polynomials is radical.
   - Generalization to non-radical ideals: Hermite conditions attached to each point in V.
   - Lei-Zheng-Ruen 2014: investigated using Lederer formulation (four-in-a-row). From their own account, dautingly complicated to estimate complexity in this way.

Perspective

1. Find a simple formulation of the previous problems.
   - find a simple way, on the lextree, to address a (sharp) complexity analysis of the recycling phase.
2. The ideal of vanishing polynomials is radical.
   - Generalization to non-radical ideals: Hermite conditions attached to each point in V.
   - Lei-Zheng-Ruen 2014: investigated using Lederer formulation (four-in-a-row). From their own account, dautingly complicated to estimate complexity in this way.
   - the key is simplicity. It is the case when the Hermite conditions are triangular: the highest order in the derivative of Taylor expansions appear to the largest (single) variable.