Signature-based algorithms to compute Gröbner bases

Christian Eder
(joint work with John Perry)

University of Kaiserslautern

June 09, 2011

What is this talk all about?

1. Efficient computations of Gröbner bases using so-called **signature-based algorithms**
2. Explanation of the **criteria** those algorithms are based on in comparison to Buchberger's criteria.
3. Explanation of **termination issues** and how they can be solved
4. Comparison between **different attempts** in the signature-based world

Convention

In this talk $R = K[x_1, \ldots, x_n]$, where K is a field. Moreover, $<$ is a well-order on R.

The following section is about

1. Given a ring R and an ideal $I \lhd R$ we want to answer some question w.r.t. to I.
   $\Rightarrow$ We want to compute a **Gröbner basis** G **of** I.
2. G can be understood as a **nice representation for** I.
   Gröbner bases were discovered by Bruno Buchberger in 1965.
   Having computed G lots of **difficult questions** concerning I are **easier to answer using** G instead of I.
3. This is due to some nice properties of Gröbner bases. The following is very useful to understand how to compute a Gröbner basis.

Main properties of Göbner bases

Definition

$G = \{g_1, \ldots, g_r\}$ is a **Gröbner basis** of an ideal $I = \langle f_1, \ldots, f_m \rangle$ iff $G \subset I$ and $\langle \mathrm{lm}(g_1), \ldots, \mathrm{lm}(g_r) \rangle = \langle \mathrm{lm}(f) \mid f \in I \rangle$.

Main properties of Göbner bases

Definition

$G = \{g_1, \ldots, g_r\}$ is a **Gröbner basis** of an ideal $I = \langle f_1, \ldots, f_m \rangle$ iff $G \subset I$ and $\langle \mathrm{lm}(g_1), \ldots, \mathrm{lm}(g_r) \rangle = \langle \mathrm{lm}(f) \mid f \in I \rangle$.

Theorem (Buchberger's Criterion)

The following are equivalent:

1. *G is a Gröbner basis of an ideal I.*
2. *For all $p, q \in G$ it holds that*

$$\mathrm{Spol}(p, q) \xrightarrow{G} 0,$$

where

- $\mathrm{Spol}(p, q) = \mathrm{lc}(q) u_p p - \mathrm{lc}(p) u_q q$, *and*
- $u_r = \frac{\mathrm{lcm}(\mathrm{lm}(p), \mathrm{lm}(q))}{\mathrm{lm}(r)}$.

Example

Assume the ideal $I = \langle g_1, g_2 \rangle \lhd \mathbb{Q}[x, y, z]$ where $g_1 = xy - z^2$, $g_2 = y^2 - z^2$; $<$ degree reverse lexicographical order.
Computing

$$\begin{aligned}
\mathrm{Spol}(g_2, g_1) &= xg_2 - yg_1 \\
&= \mathbf{xy^2} - xz^2 - \mathbf{xy^2} + yz^2 \\
&= -xz^2 + yz^2,
\end{aligned}$$

we get a new element $g_3 = xz^2 - yz^2$.

The usual **Buchberger Algorithm** to compute G follows easily from Buchberger's Criterion:
Input: Ideal $I = \langle f_1, \ldots, f_m \rangle$
Output: Gröbner basis G of I

1. $G = \emptyset$
2. $G := G \cup \{f_i\}$ for all $i \in \{1, \ldots, m\}$
3. Set $P := \{(g_i, g_j) \mid g_i, g_j \in G, i > j\}$

The usual **Buchberger Algorithm** to compute G follows easily from Buchberger's Criterion:
Input: Ideal $I = \langle f_1, \ldots, f_m \rangle$
Output: Gröbner basis G of I

1. $G = \emptyset$
2. $G := G \cup \{f_i\}$ for all $i \in \{1, \ldots, m\}$
3. Set $P := \{(g_i, g_j) \mid g_i, g_j \in G, i > j\}$
4. Choose $(p, q) \in P$, $P := P \setminus \{p\}$
5. $r := \mathrm{Spol}(p, q)$

The usual **Buchberger Algorithm** to compute G follows easily from Buchberger's Criterion:
Input: Ideal $I = \langle f_1, \ldots, f_m \rangle$
Output: Gröbner basis G of I

1. $G = \emptyset$
2. $G := G \cup \{f_i\}$ for all $i \in \{1, \ldots, m\}$
3. Set $P := \{(g_i, g_j) \mid g_i, g_j \in G, i > j\}$
4. Choose $(p, q) \in P$, $P := P \setminus \{p\}$
5. $r := \mathrm{Spol}(p, q)$
   (a) If $r \xrightarrow{G} 0$
       Go on with the next element in P.

The usual **Buchberger Algorithm** to compute G follows easily from Buchberger's Criterion:
Input: Ideal $I = \langle f_1, \ldots, f_m \rangle$
Output: Gröbner basis G of I

1. $G = \emptyset$
2. $G := G \cup \{f_i\}$ for all $i \in \{1, \ldots, m\}$
3. Set $P := \{(g_i, g_j) \mid g_i, g_j \in G, i > j\}$
4. Choose $(p, q) \in P$, $P := P \setminus \{p\}$
5. $r := \mathrm{Spol}(p, q)$
   (a) If $r \xrightarrow{G} 0$
   Go on with the next element in P.
   (b) If $r \xrightarrow{G} h \neq 0$
   Add h to G.
   Build new s-polynomials with h and add them to P.
   Go on with the next element in P.
6. When $P = \emptyset$ we are done and G is a Gröbner basis of I.

The usual **Buchberger Algorithm** to compute G follows easily from Buchberger's Criterion:
Input: Ideal $I = \langle f_1, \ldots, f_m \rangle$
Output: Gröbner basis G of I

1. $G = \emptyset$
2. $G := G \cup \{f_i\}$ for all $i \in \{1, \ldots, m\}$
3. Set $P := \{(g_i, g_j) \mid g_i, g_j \in G, i > j\}$
4. Choose $(p, q) \in P$, $P := P \setminus \{p\}$
5. $r := \mathrm{Spol}(p, q)$
   (a) If $r \xrightarrow{G} 0 \Rightarrow$ **no new information**
   Go on with the next element in P.
   (b) If $r \xrightarrow{G} h \neq 0$
   Add h to G.
   Build new s-polynomials with h and add them to P.
   Go on with the next element in P.
6. When $P = \emptyset$ we are done and G is a Gröbner basis of I.

The usual **Buchberger Algorithm** to compute G follows easily from Buchberger's Criterion:
Input: Ideal $I = \langle f_1, \ldots, f_m \rangle$
Output: Gröbner basis G of I

1. $G = \emptyset$
2. $G := G \cup \{f_i\}$ for all $i \in \{1, \ldots, m\}$
3. Set $P := \{(g_i, g_j) \mid g_i, g_j \in G, i > j\}$
4. Choose $(p, q) \in P$, $P := P \setminus \{p\}$
5. $r := \mathrm{Spol}(p, q)$
   (a) If $r \xrightarrow{G} 0 \Rightarrow$ **no new information**
   Go on with the next element in P.
   (b) If $r \xrightarrow{G} h \neq 0 \Rightarrow$ **new information**
   Add h to G.
   Build new s-polynomials with h and add them to P.
   Go on with the next element in P.
6. When $P = \emptyset$ we are done and G is a Gröbner basis of I.

Computing Gröbner bases incrementally

A slightly variant of this algorithm is the following computing the Gröbner basis **incrementally**:

Computing Gröbner bases incrementally

A slightly variant of this algorithm is the following computing the Gröbner basis **incrementally**:
Input: Ideal $I = \langle f_1, \ldots, f_m \rangle$
Output: Gröbner basis G of I

A slightly variant of this algorithm is the following computing the Gröbner basis **incrementally**:

Input: Ideal $I = \langle f_1, \ldots, f_m \rangle$

Output: Gröbner basis G of I

1. Compute Gröbner basis G_1 of $\langle f_1 \rangle$.

Computing Gröbner bases incrementally

A slightly variant of this algorithm is the following computing the Gröbner basis **incrementally**:

Input: Ideal $I = \langle f_1, \ldots, f_m \rangle$

Output: Gröbner basis G of I

1. Compute Gröbner basis G_1 of $\langle f_1 \rangle$.
2. Compute Gröbner basis G_2 of $\langle f_1, f_2 \rangle$ by
   (a) $G_2 = G_1 \cup \{f_2\}$,
   (b) computing s-polynomials of f_2 with elements of G_1
   (c) reducing all s-polynomials w.r.t. G_2 and possibly add new elements to G_2

A slightly variant of this algorithm is the following computing the Gröbner basis **incrementally**:

Input: Ideal $I = \langle f_1, \ldots, f_m \rangle$

Output: Gröbner basis G of I

1. Compute Gröbner basis G_1 of $\langle f_1 \rangle$.
2. Compute Gröbner basis G_2 of $\langle f_1, f_2 \rangle$ by
   (a) $G_2 = G_1 \cup \{f_2\}$,
   (b) computing s-polynomials of f_2 with elements of G_1
   (c) reducing all s-polynomials w.r.t. G_2 and possibly add new elements to G_2
3. ...
4. $G := G_m$ is the Gröbner basis of I

Problem of zero reduction

Lots of useless computations

It is very time-consuming to compute G such that $\mathrm{Spol}(p, q)$ **reduces to zero w.r.t.** G for all $p, q \in G$.

Problem of zero reduction

Lots of useless computations

It is very time-consuming to compute G such that $\mathrm{Spol}(p, q)$ **reduces to zero w.r.t.** G for all $p, q \in G$.
When such an s-polynomial reduces to an element $h \neq 0$ w.r.t. G then we get **new information** for the structure of G, namely adding h to G.

Problem of zero reduction

Lots of useless computations

It is very time-consuming to compute G such that $\mathrm{Spol}(p, q)$ **reduces to zero w.r.t.** G for all $p, q \in G$.

When such an s-polynomial reduces to an element $h \neq 0$ w.r.t. G then we get **new information** for the structure of G, namely adding h to G.

But most of the s-polynomials considered during the algorithm reduce to zero w.r.t. G.

$\Rightarrow$ **No new information from zero reductions**

Problem of zero reduction

Lots of useless computations

It is very time-consuming to compute G such that $\mathrm{Spol}(p,q)$ **reduces to zero w.r.t.** G for all $p, q \in G$.
When such an s-polynomial reduces to an element $h \neq 0$ w.r.t. G then we get **new information** for the structure of G, namely adding h to G.
But most of the s-polynomials considered during the algorithm reduce to zero w.r.t. G.
$\Rightarrow$ **No new information from zero reductions**

Let's have a look at the example again:

An example of zero reduction

Example

Given $g_1 = xy - z^2$, $g_2 = y^2 - z^2$,we have computed

$$\mathrm{Spol}(g_2, g_1) = \mathbf{xy^2} - xz^2 - \mathbf{xy^2} + yz^2 = -xz^2 + yz^2.$$

An example of zero reduction

Example

Given $g_1 = xy - z^2$, $g_2 = y^2 - z^2$,we have computed

$$\mathrm{Spol}(g_2, g_1) = \mathbf{xy^2} - xz^2 - \mathbf{xy^2} + yz^2 = -xz^2 + yz^2.$$

We get a new element $g_3 = xz^2 - yz^2$ for G.

Example

Given $g_1 = xy - z^2$, $g_2 = y^2 - z^2$, we have computed

$$\mathrm{Spol}(g_2, g_1) = \mathbf{xy^2} - xz^2 - \mathbf{xy^2} + yz^2 = -xz^2 + yz^2.$$

We get a new element $g_3 = xz^2 - yz^2$ for G.

Let us compute $\mathrm{Spol}(g_3, g_1)$ next:

An example of zero reduction

Example

Given $g_1 = xy - z^2$, $g_2 = y^2 - z^2$, we have computed

$$\mathrm{Spol}(g_2, g_1) = \mathbf{xy^2} - xz^2 - \mathbf{xy^2} + yz^2 = -xz^2 + yz^2.$$

We get a new element $g_3 = xz^2 - yz^2$ for G.

Let us compute $\mathrm{Spol}(g_3, g_1)$ next:

$$\mathrm{Spol}(g_3, g_1) = \mathbf{xyz^2} - y^2z^2 - \mathbf{xyz^2} + z^4 = -y^2z^2 + z^4.$$

Example

Given $g_1 = xy - z^2$, $g_2 = y^2 - z^2$, we have computed

$$\mathrm{Spol}(g_2, g_1) = \mathbf{xy^2} - xz^2 - \mathbf{xy^2} + yz^2 = -xz^2 + yz^2.$$

We get a new element $g_3 = xz^2 - yz^2$ for G.
Let us compute $\mathrm{Spol}(g_3, g_1)$ next:

$$\mathrm{Spol}(g_3, g_1) = \mathbf{xyz^2} - y^2z^2 - \mathbf{xyz^2} + z^4 = -y^2z^2 + z^4.$$

Now we can reduce further with z^2g_2:

$$-y^2z^2 + z^4 + y^2z^2 - z^4 = 0.$$

An example of zero reduction

Example

Given $g_1 = xy - z^2$, $g_2 = y^2 - z^2$, we have computed

$$\mathrm{Spol}(g_2, g_1) = \mathbf{xy^2} - xz^2 - \mathbf{xy^2} + yz^2 = -xz^2 + yz^2.$$

We get a new element $g_3 = xz^2 - yz^2$ for G.
Let us compute $\mathrm{Spol}(g_3, g_1)$ next:

$$\mathrm{Spol}(g_3, g_1) = \mathbf{xyz^2} - y^2z^2 - \mathbf{xyz^2} + z^4 = -y^2z^2 + z^4.$$

Now we can reduce further with $z^2 g_2$:

$$-y^2z^2 + z^4 + y^2z^2 - z^4 = 0.$$

$\Rightarrow$ **How to detect zero reductions in advance?**

Known ideas for optimizing computations

- **Predict zero reductions** (Buchberger, Gebauer-Möller, Möller-Mora-Traverso, etc.)
- **Selection strategies:** Pick pairs in a clever way (Buchberger, Giovini et al., Möller et al.)
- **Homogenization:** d-Gröbner bases
- **Involutive bases:** Forbid some top-reductions (Gerdt, Blinkov)

The following section is about

Signatures of polynomials

Let $I = \langle f_1, \ldots, f_m \rangle$. The idea is to give each polynomial during the computations of the algorithm a so-called **signature**:

Signatures of polynomials

Let $I = \langle f_1, \ldots, f_m \rangle$. The idea is to give each polynomial during the computations of the algorithm a so-called **signature**:

1. Let $e_1, \ldots, e_m \in R^m$ be canonical generators such that $\pi : R^m \to R$: $\pi(e_i) = f_i$ for all i.

Let $I = \langle f_1, \ldots, f_m \rangle$. The idea is to give each polynomial during the computations of the algorithm a so-called **signature**:

1. Let $e_1, \ldots, e_m \in R^m$ be canonical generators such that $\pi : R^m \to R$: $\pi(e_i) = f_i$ for all i.
2. Any polynomial $p \in I$ can be written as $p = h_1 f_1 + \ldots + h_m f_m$.

Let $I = \langle f_1, \dots, f_m \rangle$. The idea is to give each polynomial during the computations of the algorithm a so-called **signature**:

1. Let $e_1, \dots, e_m \in R^m$ be canonical generators such that $\pi : R^m \to R$: $\pi(e_i) = f_i$ for all i.
2. Any polynomial $p \in I$ can be written as $p = h_1 f_1 + \ldots + h_m f_m$.
3. Let k be the greatest index such that h_k is not zero.
   $\Rightarrow$ **A signature** $\mathcal{S}(p) = \mathrm{lm}(h_k) e_k$.

Let $I = \langle f_1, \ldots, f_m \rangle$. The idea is to give each polynomial during the computations of the algorithm a so-called **signature**:

1. Let $e_1, \ldots, e_m \in R^m$ be canonical generators such that $\pi : R^m \to R$: $\pi(e_i) = f_i$ for all i.
2. Any polynomial $p \in I$ can be written as $p = h_1 f_1 + \ldots + h_m f_m$.
3. Let k be the greatest index such that h_k is not zero.
   $\Rightarrow$ **A signature** $\mathcal{S}(p) = \mathrm{lm}(h_k) e_k$.
4. A generating element f_i of I gets the signature $\mathcal{S}(f_i) = e_i$.

Let $I = \langle f_1, \ldots, f_m \rangle$. The idea is to give each polynomial during the computations of the algorithm a so-called **signature**:

1. Let $e_1, \ldots, e_m \in R^m$ be canonical generators such that $\pi : R^m \to R$: $\pi(e_i) = f_i$ for all i.
2. Any polynomial $p \in I$ can be written as $p = h_1 f_1 + \ldots + h_m f_m$.
3. Let k be the greatest index such that h_k is not zero.
   $\Rightarrow$ **A signature** $\mathcal{S}(p) = \mathrm{lm}(h_k) e_k$.
4. A generating element f_i of I gets the signature $\mathcal{S}(f_i) = e_i$.
5. Extend the monomial order on the signatures
   (a) Well-order $\prec$ on the set of all signatures
   (b) Existence of **the minimal signature** of a polynomial p

Remark

Note that there are various ways to define the order $\prec$ depending on different preferences of the monomial resp. the index of the signature

1. 2002 Faugère [Fa02]
2. 2009 Ars and Hashemi [AH09]
3. 2010 Gao, Volny, and Wang [GVW11]
4. 2010 / 2011 Sun and Wang [SW10, SW11]

We use Faugère's variant:

$$t_k e_k \succ t_\ell e_\ell \quad \Leftrightarrow \quad \begin{array}{l}\text{(a)}k > \ell \text{ or} \\ \text{(b)}k = \ell \text{ and } t_k > t_\ell\end{array}$$

Orders on signatures

We use Faugère's variant:

$$t_k e_k \succ t_\ell e_\ell \quad \Leftrightarrow \quad \begin{array}{l} \text{(a)} k > \ell \text{ or} \\ \text{(b)} k = \ell \text{ and } t_k > t_\ell \end{array}$$

Example

Assume $\mathbb{Q}[x, y, z]$ with degree reverse lexicographical order. Then

1. $x^2 y e_3 \succ z^3 e_3$,
2. $1 \cdot e_5 \succ x^{12} y^{234} z^{3456} e_4$.

Signatures of s-polynomials

Using **signatures** in a Gröbner basis algorithm we clearly need to define them **for s-polynomials**, too:

$$\mathrm{Spol}(p,q) = \mathrm{lc}(q)u_p p - \mathrm{lc}(p)u_q q$$

such that

$$\mathcal{S}\left(\mathrm{Spol}(p,q)\right) = u_p\mathcal{S}(p)$$
$$u_p\mathcal{S}(p) \succ u_q\mathcal{S}(q).$$

Example revisited - with signatures

In our example

$$\begin{aligned} g_3 &= \mathrm{Spol}(g_2, g_1) = xg_2 - yg_1 \\ \Rightarrow \mathcal{S}(g_3) &= x\mathcal{S}(g_2) = xe_2. \end{aligned}$$

Example revisited - with signatures

In our example

$$g_3 = \mathrm{Spol}(g_2, g_1) = xg_2 - yg_1$$
$$\Rightarrow \mathcal{S}(g_3) = x\mathcal{S}(g_2) = xe_2.$$

It follows that $\mathrm{Spol}(g_3, g_1) = yg_3 - z^2 g_1$ has

$$\mathcal{S}\left(\mathrm{Spol}(g_3, g_1)\right) = y\mathcal{S}(g_3) = xye_2.$$

Example revisited - with signatures

In our example

$$g_3 = \mathrm{Spol}(g_2, g_1) = xg_2 - yg_1$$
$$\Rightarrow \mathcal{S}(g_3) = x\mathcal{S}(g_2) = xe_2.$$

It follows that $\mathrm{Spol}(g_3, g_1) = yg_3 - z^2 g_1$ has

$$\mathcal{S}\left(\mathrm{Spol}(g_3, g_1)\right) = y\mathcal{S}(g_3) = xye_2.$$

Note that $\mathcal{S}\left(\mathrm{Spol}(g_3, g_1)\right) = (xy\, e_2)$ and $\mathrm{lm}(g_1) = xy$.

Example revisited - with signatures

In our example

$$g_3 = \mathrm{Spol}(g_2, g_1) = xg_2 - yg_1$$
$$\Rightarrow \mathcal{S}(g_3) = x\mathcal{S}(g_2) = xe_2.$$

It follows that $\mathrm{Spol}(g_3, g_1) = yg_3 - z^2 g_1$ has

$$\mathcal{S}\left(\mathrm{Spol}(g_3, g_1)\right) = y\mathcal{S}(g_3) = xye_2.$$

Note that $\mathcal{S}\left(\mathrm{Spol}(g_3, g_1)\right) = (xy\, e_2)$ and $\mathrm{lm}(g_1) = xy$.
$\Rightarrow$ We **know** that $\mathrm{Spol}(g_3, g_1)$ will reduce to zero!

The main idea is to check if the next element $\mathrm{Spol}(p, q)$ has the **minimal signature**.

How does this work?

The main idea is to check if the next element $\mathrm{Spol}(p, q)$ has the **minimal signature**.
If $\mathcal{S}\big(\mathrm{Spol}(p, q)\big)$ is not minimal $\Rightarrow \mathrm{Spol}(p, q)$ can be discarded.

How does this work?

The main idea is to check if the next element $\mathrm{Spol}(p, q)$ has the **minimal signature**.
If $\mathcal{S}\big(\mathrm{Spol}(p, q)\big)$ is not minimal $\Rightarrow \mathrm{Spol}(p, q)$ can be discarded.

Question

How do we know, if the signature of a polynomial / critical pair is not minimal?

Computing Gröbner bases using signatures

Input: $G_{i-1} = \{g_1, \ldots, g_{r-1}\}$, a Gröbner basis of $\langle f_1, \ldots, f_{i-1} \rangle$
Output: Gröbner basis G of $\langle f_1, \ldots, f_i \rangle$

Computing Gröbner bases using signatures

Input: $G_{i-1} = \{g_1, \ldots, g_{r-1}\}$, a Gröbner basis of $\langle f_1, \ldots, f_{i-1} \rangle$
Output: Gröbner basis G of $\langle f_1, \ldots, f_i \rangle$

1. $g_r := f_i$

Computing Gröbner bases using signatures

Input: $G_{i-1} = \{g_1, \ldots, g_{r-1}\}$, a Gröbner basis of $\langle f_1, \ldots, f_{i-1} \rangle$
Output: Gröbner basis G of $\langle f_1, \ldots, f_i \rangle$

1. $g_r := f_i$
2. $G = \big\{(e_1, g_1), \ldots, (e_{r-1}, g_{r-1}), (e_r, g_r)\big\}$ (monic)

Computing Gröbner bases using signatures

Input: $G_{i-1} = \{g_1, \dots, g_{r-1}\}$, a Gröbner basis of $\langle f_1, \dots, f_{i-1} \rangle$
Output: Gröbner basis G of $\langle f_1, \dots, f_i \rangle$

1. $g_r := f_i$
2. $G = \big\{(e_1, g_1), \dots, (e_{r-1}, g_{r-1}), (e_r, g_r)\big\}$ (monic)
3. Set $P := \big\{ \big(\frac{\mathrm{lcm}(g_r, g_j)}{\mathrm{lm}(g_r)} e_r, g_r, g_j \big), j < r \big\}$

Computing Gröbner bases using signatures

Input: $G_{i-1} = \{g_1, \ldots, g_{r-1}\}$, a Gröbner basis of $\langle f_1, \ldots, f_{i-1} \rangle$
Output: Gröbner basis G of $\langle f_1, \ldots, f_i \rangle$

1. $g_r := f_i$
2. $G = \big\{(e_1, g_1), \ldots, (e_{r-1}, g_{r-1}), (e_r, g_r)\big\}$ (monic)
3. Set $P := \big\{\big(\frac{\mathrm{lcm}(g_r, g_j)}{\mathrm{lm}(g_r)} e_r, g_r, g_j\big), j < r\big\}$
4. While $P \neq \emptyset$
   (a) Choose $(\lambda e_r, p, q) \in P$ such that λe_r is minimal.
   (b) Delete $(\lambda e_r, p, q)$ from P.

Computing Gröbner bases using signatures

Input: $G_{i-1} = \{g_1, \ldots, g_{r-1}\}$, a Gröbner basis of $\langle f_1, \ldots, f_{i-1} \rangle$
Output: Gröbner basis G of $\langle f_1, \ldots, f_i \rangle$

1. $g_r := f_i$
2. $G = \big\{(e_1, g_1), \ldots, (e_{r-1}, g_{r-1}), (e_r, g_r)\big\}$ (monic)
3. Set $P := \big\{\big(\frac{\operatorname{lcm}(g_r, g_j)}{\operatorname{lm}(g_r)} e_r, g_r, g_j\big), j < r\big\}$
4. While $P \neq \emptyset$
   (a) Choose $(\lambda e_r, p, q) \in P$ such that λe_r is minimal.
   (b) Delete $(\lambda e_r, p, q)$ from P.
   (c) (λe_r) not minimal for $up - vq \Rightarrow$ goto 4.

Computing Gröbner bases using signatures

Input: $G_{i-1} = \{g_1, \ldots, g_{r-1}\}$, a Gröbner basis of $\langle f_1, \ldots, f_{i-1} \rangle$
Output: Gröbner basis G of $\langle f_1, \ldots, f_i \rangle$

1. $g_r := f_i$
2. $G = \big\{(e_1, g_1), \ldots, (e_{r-1}, g_{r-1}), (e_r, g_r)\big\}$ (monic)
3. Set $P := \big\{\big(\frac{\mathrm{lcm}(g_r, g_j)}{\mathrm{lm}(g_r)} e_r, g_r, g_j\big), j < r\big\}$
4. While $P \neq \emptyset$
   - (a) Choose $(\lambda e_r, p, q) \in P$ such that λe_r is minimal.
   - (b) Delete $(\lambda e_r, p, q)$ from P.
   - (c) (λe_r) not minimal for $up - vq \Rightarrow$ goto 4.
   - (d) $\big(S(h), h\big) = \mathrm{reduce}\big((\lambda e_r, up - vq), G\big)$

Computing Gröbner bases using signatures

Input: $G_{i-1} = \{g_1, \ldots, g_{r-1}\}$, a Gröbner basis of $\langle f_1, \ldots, f_{i-1} \rangle$
Output: Gröbner basis G of $\langle f_1, \ldots, f_i \rangle$

1. $g_r := f_i$
2. $G = \big\{(e_1, g_1), \ldots, (e_{r-1}, g_{r-1}), (e_r, g_r)\big\}$ (monic)
3. Set $P := \big\{\big(\frac{\mathrm{lcm}(g_r, g_j)}{\mathrm{lm}(g_r)} e_r, g_r, g_j\big), j < r\big\}$
4. While $P \neq \emptyset$
   (a) Choose $(\lambda e_r, p, q) \in P$ such that λe_r is minimal.
   (b) Delete $(\lambda e_r, p, q)$ from P.
   (c) (λe_r) not minimal for $up - vq \Rightarrow$ goto 4.
   (d) $\big(S(h), h\big) = \mathrm{reduce}\big((\lambda e_r, up - vq), G\big)$
   (e) $h \neq 0$ & $\nexists (\mathcal{S}(g), g) \in G$, $t \in M$ s.t. $t\mathcal{S}(g) = \mathcal{S}(h)$ and $t\,\mathrm{lm}(g) = \mathrm{lm}(h)$
      (i) For all $g \in G$ add $(\sigma e_r, h, g)$ to P.
      (ii) Add $\big(\mathcal{S}(h), h\big)$ to G.
5. When $P = \emptyset$ we are done and G is a Gröbner basis of $\langle f_1, \ldots, f_i \rangle$.

Computing Gröbner bases using signatures

Input: $G_{i-1} = \{g_1, \ldots, g_{r-1}\}$, a Gröbner basis of $\langle f_1, \ldots, f_{i-1} \rangle$
Output: Gröbner basis G of $\langle f_1, \ldots, f_i \rangle$

1. $g_r := f_i$
2. $G = \big\{(e_1, g_1), \ldots, (e_{r-1}, g_{r-1}), (e_r, g_r)\big\}$ (monic)
3. Set $P := \big\{\big(\frac{\mathrm{lcm}(g_r, g_j)}{\mathrm{lm}(g_r)} e_r, g_r, g_j\big), j < r\big\}$
4. While $P \neq \emptyset$
   (a) Choose $(\lambda e_r, p, q) \in P$ such that λe_r is minimal.
   (b) Delete $(\lambda e_r, p, q)$ from P.
   (c) (λe_r) not minimal for $up - vq \Rightarrow$ goto 4.
   (d) $\big(S(h), h\big) = \text{reduce}\big((\lambda e_r, up - vq), G\big) \Leftarrow$ sig-safe!
   (e) $h \neq 0$ & $\nexists (\mathcal{S}(g), g) \in G$, $t \in M$ s.t. $t\mathcal{S}(g) = \mathcal{S}(h)$ and $t\,\mathrm{lm}(g) = \mathrm{lm}(h)$
      (i) For all $g \in G$ add $(\sigma e_r, h, g)$ to P.
      (ii) Add $\big(\mathcal{S}(h), h\big)$ to G.
5. When $P = \emptyset$ we are done and G is a Gröbner basis of $\langle f_1, \ldots, f_i \rangle$.

Reductions w.r.t. signatures

Let $\big(\mathcal{S}(p), p\big)$, $\big(\mathcal{S}(q), q\big)$ such that $\lambda \mathrm{lm}(q) = \mathrm{lm}(p)$.

Let $\big(\mathcal{S}(p), p\big)$, $\big(\mathcal{S}(q), q\big)$ such that $\lambda \mathrm{lm}(q) = \mathrm{lm}(p)$.

1. **Sig-safe:** $\mathcal{S}(p - \lambda q) = \mathcal{S}(p)$.

Let $\big(\mathcal{S}(p), p\big)$, $\big(\mathcal{S}(q), q\big)$ such that $\lambda \mathrm{lm}(q) = \mathrm{lm}(p)$.

1. **Sig-safe:** $\mathcal{S}(p - \lambda q) = \mathcal{S}(p)$.
2. **Sig-unsafe:** $\mathcal{S}(p - \lambda q) = \lambda \mathcal{S}(q)$.

Let $\big(\mathcal{S}(p), p\big)$, $\big(\mathcal{S}(q), q\big)$ such that $\lambda \mathrm{lm}(q) = \mathrm{lm}(p)$.

1. **Sig-safe:** $\mathcal{S}(p - \lambda q) = \mathcal{S}(p)$.
2. **Sig-unsafe:** $\mathcal{S}(p - \lambda q) = \lambda \mathcal{S}(q)$.
3. **Sig-cancelling:** $\mathcal{S}(p) = \lambda \mathcal{S}(q) \Rightarrow \mathcal{S}(p - \lambda q) = ?$

Let $(\mathcal{S}(p), p)$, $(\mathcal{S}(q), q)$ such that $\lambda \mathrm{lm}(q) = \mathrm{lm}(p)$.

1. **Sig-safe:** $\mathcal{S}(p - \lambda q) = \mathcal{S}(p)$.
2. **Sig-unsafe:** $\mathcal{S}(p - \lambda q) = \lambda \mathcal{S}(q)$.
3. **Sig-cancelling:** $\mathcal{S}(p) = \lambda \mathcal{S}(q) \Rightarrow \mathcal{S}(p - \lambda q) =?$

Example

$\mathcal{S}(p) = xy^2 e_1$, $\mathcal{S}(q) = xye_1$, $x > y > z$

1. **Sig-safe:** $\mathcal{S}(p - zq) = xy^2 e_1$.

Let $\left(\mathcal{S}(p), p\right)$, $\left(\mathcal{S}(q), q\right)$ such that $\lambda \mathrm{lm}(q) = \mathrm{lm}(p)$.

1. **Sig-safe:** $\mathcal{S}(p - \lambda q) = \mathcal{S}(p)$.
2. **Sig-unsafe:** $\mathcal{S}(p - \lambda q) = \lambda \mathcal{S}(q)$.
3. **Sig-cancelling:** $\mathcal{S}(p) = \lambda \mathcal{S}(q) \Rightarrow \mathcal{S}(p - \lambda q) =?$

Example

$\mathcal{S}(p) = xy^2 e_1$, $\mathcal{S}(q) = xye_1$, $x > y > z$

1. **Sig-safe:** $\mathcal{S}(p - zq) = xy^2 e_1$.
2. **Sig-unsafe:** $\mathcal{S}(p - xq) = x^2 y e_1$.

Let $(\mathcal{S}(p), p)$, $(\mathcal{S}(q), q)$ such that $\lambda \mathrm{lm}(q) = \mathrm{lm}(p)$.

1. **Sig-safe:** $\mathcal{S}(p - \lambda q) = \mathcal{S}(p)$.
2. **Sig-unsafe:** $\mathcal{S}(p - \lambda q) = \lambda \mathcal{S}(q)$.
3. **Sig-cancelling:** $\mathcal{S}(p) = \lambda \mathcal{S}(q) \Rightarrow \mathcal{S}(p - \lambda q) = ?$

Example

$\mathcal{S}(p) = xy^2 e_1$, $\mathcal{S}(q) = xye_1$, $x > y > z$

1. **Sig-safe:** $\mathcal{S}(p - zq) = xy^2 e_1$.
2. **Sig-unsafe:** $\mathcal{S}(p - xq) = x^2 y e_1$.
3. **Sig-cancelling:** $\mathcal{S}(p - yq) = ?$

Computing Gröbner bases using signatures

Termination?

1. No new s-polynomials for $\big(\mathcal{S}(h), h\big) = \lambda\big(\mathcal{S}(g), g\big)$
2. Each new element expands $\big\langle\big(\mathcal{S}(h), \mathrm{lm}(h)\big)\big\rangle$

Termination?

1. No new s-polynomials for $\big(\mathcal{S}(h), h\big) = \lambda\big(\mathcal{S}(g), g\big)$
2. Each new element expands $\big\langle\big(\mathcal{S}(h), \mathrm{lm}(h)\big)\big\rangle$

Correctness?

1. Proceed by minimal signature in P
2. All s-polynomials considered:
   sig-unsafe reduction $\Rightarrow$ new critical pair next round
3. All nonzero elements added besides $\big(\mathcal{S}(h), h\big) = \lambda\big(\mathcal{S}(g), g\big)$

Non-minimal signature (NM)
$\mathcal{S}(h)$ not minimal for h? $\Rightarrow$ discard h

Non-minimal signature (NM)

$\mathcal{S}(h)$ not minimal for h? $\Rightarrow$ discard h

Proof.

1. There exists syzygy s with $\mathrm{lm}(s) = \mathcal{S}(h)$.
2. We can rewrite h using a lower signature.
3. We proceed by increasing signatures.
   $\Rightarrow$ Those reductions are already considered.

□

Allowed criteria?

Rewritable signature (RW)
$\mathcal{S}(g) = \mathcal{S}(h)$? $\Rightarrow$ discard either g or h

Rewritable signature (RW)
$\mathcal{S}(g) = \mathcal{S}(h)$? $\Rightarrow$ discard either g or h

Proof.

1. $\mathcal{S}(g - h) < \mathcal{S}(h), \mathcal{S}(g)$.
2. We proceed by increasing signatures.
   $\Rightarrow$ Those reductions are already considered.
   $\Rightarrow$ We can rewrite $h = g +$ terms of lower signature.

□

The following section is about

What are the differences?

1. Different implementations of (NM) and (RW)

What are the differences?

1. Different implementations of (NM) and (RW)
2. Different implementations of the sig-safe reduction

What are the differences?

1. Different implementations of (NM) and (RW)
2. Different implementations of the sig-safe reduction

Remark

The presented criteria (NM) and (RW) are also used during the (sig-safe) reduction steps. This usage is quite **soft in GGV** and quite **aggressive in F5**.

What are the differences?

1. Different implementations of (NM) and (RW)
2. Different implementations of the sig-safe reduction

Remark

The presented criteria (NM) and (RW) are also used during the (sig-safe) reduction steps. This usage is quite **soft in GGV** and quite **aggressive in F5**.

$\Rightarrow$ **Termination:** GGV ☺ – F5 ☹

F5's implementation of (NM)

If

$$\mathcal{S}(g) = \lambda e_{<i},$$
$$\mathcal{S}(h) = \sigma e_i, \text{ and}$$
$$\mathrm{lm}(g) \mid \sigma,$$

then discard *h*.

If there exists $\left(\mathcal{S}(g), g\right)$ such that

$$\begin{aligned} \mathcal{S}(g) &= \lambda e_r, \\ \mathcal{S}(h) &= \sigma \mathcal{S}(f) = \sigma\left(\tau e_r\right), \\ &\lambda \mid \sigma\tau, \text{ and} \\ &g \text{ computed after } f, \end{aligned}$$

then discard h.

F5's implementation of (RW)

If there exists $\big(\mathcal{S}(g), g\big)$ such that

$$\begin{aligned}\mathcal{S}(g) &= \lambda e_r,\\ \mathcal{S}(h) &= \sigma\mathcal{S}(f) = \sigma\big(\tau e_r\big),\\ \lambda &\mid \sigma\tau, \text{ and}\\ g &\text{ computed after } f,\end{aligned}$$

then discard h.

Remark

This is an aggressive implementation of (RW) changing "equality" to "divisibility" in the criterion.

Initially $H = \{\mathrm{lm}(g_1), \ldots, \mathrm{lm}(g_{r-1})\}$.

GGV's implementation of (NM)

Initially $H = \{\mathrm{lm}(g_1), \ldots, \mathrm{lm}(g_{r-1})\}$.
Whenever p reduces to zero

$$\Rightarrow H = H \cup \{\lambda\} \text{ where } \mathcal{S}(p) = \lambda e_r.$$

GGV's implementation of (NM)

Initially $H = \{\mathrm{lm}(g_1), \ldots, \mathrm{lm}(g_{r-1})\}$.
Whenever p reduces to zero

$$\Rightarrow H = H \cup \{\lambda\} \text{ where } \mathcal{S}(p) = \lambda e_r.$$

If

$$\mathcal{S}(g) = \sigma e_r,$$
$$\exists h \in H \text{ such that } h \mid \sigma,$$

then discard g.

GGV's implementation of (NM)

Initially $H = \{\mathrm{lm}(g_1), \ldots, \mathrm{lm}(g_{r-1})\}$.
Whenever p reduces to zero

$$\Rightarrow H = H \cup \{\lambda\} \text{ where } \mathcal{S}(p) = \lambda e_r.$$

If

$$\mathcal{S}(g) = \sigma e_r,$$
$$\exists h \in H \text{ such that } h \mid \sigma,$$

then discard g.

Remark

This is F5's NM criterion with additional criteria added during the computation.

GGV's implementation of (RW)

If

$$\mathcal{S}(g) = \mathcal{S}(h),$$

then consider only g or h.

GGV's implementation of (RW)

If

$$\mathcal{S}(g) = \mathcal{S}(h),$$

then consider only g or h.

Remark

This is used when creating new critical pairs.

F5E – Combine the ideas

Behaviour depending on number of zero reductions

- GGV actively uses zero reductions to improve (NM).
- F5 does not do this, but possible incorporates some of this data in (RW).
- Checking by F5's (RW) costs much more time than checking by (NM).

F5E – Combine the ideas

Behaviour depending on number of zero reductions

- GGV actively uses zero reductions to improve (NM).
- F5 does not do this, but possible incorporates some of this data in (RW).
- Checking by F5's (RW) costs much more time than checking by (NM).

The following combination is straightforward:

- Use the F5 Algorithm.
- Add GGV's (NM) to it:
  Whenever g reduces to zero, add $\mathcal{S}(g)$ to H.

The following section is about

Test environments

All examples are computed in the following setting:

1. $\mathbb{F}_{32003}$,
2. graded reverse lexicographical order.

Test environments

All examples are computed in the following setting:

1. $\mathbb{F}_{32003}$,
2. graded reverse lexicographical order.

All examples are computed on the following machine:

1. MacBook Pro 7,1 (Intel Core 2 Duo P8800),
2. 4GB Ram,
3. 5,400 rpm HDD,
4. 64-bit Ubuntu 10.10.
5. Singular 3-1-3 Developer Version

Test environments

All examples are computed in the following setting:

1. $\mathbb{F}_{32003}$,
2. graded reverse lexicographical order.

All examples are computed on the following machine:

1. MacBook Pro 7,1 (Intel Core 2 Duo P8800),
2. 4GB Ram,
3. 5,400 rpm HDD,
4. 64-bit Ubuntu 10.10.
5. Singular 3-1-3 Developer Version

Remark

All algorithms use **the same underlying structure**, differing only in the implementation of the criteria presented in this talk.

Number of critical pairs and zero reductions

System	**F5**		**F5E**		**GGV**	
Katsura 9	886	0	886	0	886	0
Katsura 10	1,781	0	1,781	0	1,781	0
Eco 8	830	322	**565**	**57**	2,012	**57**
Eco 9	2,087	929	**1,278**	**120**	5,794	**120**
F744	1,324	342	**1,151**	**169**	2,145	**169**
Cyclic 7	1,018	76	**978**	**36**	3,072	**36**
Cyclic 8	7,066	244	**5,770**	**244**	24,600	**244**

Number of critical pairs and zero reductions

System	**F5**		**F5E**		**GGV**	
Katsura 9	886	0	886	0	886	0
Katsura 10	1,781	0	1,781	0	1,781	0
Eco 8	830	322	**565**	**57**	2,012	**57**
Eco 9	2,087	929	**1,278**	**120**	5,794	**120**
F744	1,324	342	**1,151**	**169**	2,145	**169**
Cyclic 7	1,018	76	**978**	**36**	3,072	**36**
Cyclic 8	7,066	244	**5,770**	**244**	24,600	**244**

Remark

Besides considering more critical pairs, GGV does a lot more single reduction steps than F5 does.

Timings in seconds

System	**F5**	**F5E**	**GGV**
Katsura 9	14.98	**14.87**	17.63
Katsura 10	153.35	**152.39**	192.20
Eco 8	2.24	**0.38**	0.49
Eco 9	77.13	**8.19**	13.51
F744	19.35	**8.79**	26.86
Cyclic 7	**7.01**	7.22	33.85
Cyclic 8	7,310.39	**4,961.58**	26,242.12

The following section is about

Outlook

- **Implementing F4F5:**
  Gaussian Elimination done by Bradford Hovinen

Outlook

- **Implementing F4F5:**
  Gaussian Elimination done by Bradford Hovinen
- **Inhomogeneous case:**
  Working, but slow

Outlook

- **Implementing F4F5:**
  Gaussian Elimination done by Bradford Hovinen
- **Inhomogeneous case:**
  Working, but slow
- **Orders on signatures:**
  Lots of tests, heuristics

Outlook

- **Implementing F4F5:**
  Gaussian Elimination done by Bradford Hovinen
- **Inhomogeneous case:**
  Working, but slow
- **Orders on signatures:**
  Lots of tests, heuristics
- **Parallelization:**
  On criteria checks, needs thread-safe omalloc

Outlook

- **Implementing F4F5:**
  Gaussian Elimination done by Bradford Hovinen
- **Inhomogeneous case:**
  Working, but slow
- **Orders on signatures:**
  Lots of tests, heuristics
- **Parallelization:**
  On criteria checks, needs thread-safe omalloc
- **Syzygy computations:**
  Needs implementation

Outlook

- **Implementing F4F5:**
  Gaussian Elimination done by Bradford Hovinen
- **Inhomogeneous case:**
  Working, but slow
- **Orders on signatures:**
  Lots of tests, heuristics
- **Parallelization:**
  On criteria checks, needs thread-safe omalloc
- **Syzygy computations:**
  Needs implementation
- **Generalizing criteria:**
  Using more data, combining with Buchberger's criteria, etc.

References

[AH09] G. Ars and A. Hashemi. Extended F5 Criteria

[EP10] C. Eder and J. Perry. F5C: A variant of Faugère's F5 Algorithm with reduced Gröbner bases

[EGP11] C. Eder, J. Gash, and J. Perry. Modifying Faugère's F5 Algorithm to ensure termination

[EP11] C. Eder and J. Perry. Signature-based algorithms to compute Gröbner bases

[Fa02] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero F_5

[GGV10] S. Gao, Y. Guan, and F. Volny IV. A New Incremental Algorithm for Computing Gröbner Bases

[GVW11] S. Gao, F. Volny IV, and M. Wang. A New Algorithm For Computing Grobner Bases

[SIN11] W. Decker, G.-M. Greuel, G. Pfister and H. Schönemann. SINGULAR 3-1-3. *A computer algebra system for polynomial computations*, University of Kaiserslautern, 2011, `http://www.singular.uni-kl.de`.

[SW10] Y. Sun and D. Wang. A new proof of the F5 Algorithm

[SW11] Y. Sun and D. Wang. A Generalized Criterion for Signature Related Gröbner Basis Algorithms