Faugère's F5 algorithm: variants and termination issues

Christian Eder
(joint work with Justin Gash and John Perry)

University of Kaiserslautern

June 17, 2010

What is this talk all about?

1. Efficient computations of Gröbner bases using Faugère's F5 Algorithm and variants of it
2. Explanation of the F5 Algorithm, its criteria used to detect useless s-polynomials
3. Presentation of the variant F5C which reduce some inefficiencies of F5
4. Explanation and solution of the termination issue of F5

The following section is about

1. Introducing Gröbner bases
   - Gröbner basics
   - Computation of Gröbner bases
   - Problem of zero reduction
2. The F5 Algorithm
3. Optimizations of F5
4. Termination issues of F5

Basic problem

1. Given a ring R and an ideal $I \lhd R$ 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 [Bu65]. 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

Lemma

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

Main properties of Göbner bases

Lemma

$G = \{g_1, \dots, g_n\}$ is a Gröbner basis of an ideal $I = \langle f_1, \dots, f_m \rangle$ iff $G \subset I$ and $\langle \mathrm{lm}(g_1), \dots, \mathrm{lm}(g_n) \rangle = \langle \mathrm{lm}(f_1), \dots \mathrm{lm}(f_m) \rangle$.

Lemma

Let G be a Gröbner basis of an ideal I. It holds that 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*
- $u_k = \frac{\mathrm{lcm}(\mathrm{lm}(p), \mathrm{lm}(q))}{\mathrm{lm}(k)}$.

A lovely example to get to know F5

Example

Assume the ideal $I = \langle g_1, g_2 \rangle \lhd \mathbb{Q}[x,y,z]$ where $g_1 = xy - z^2$, $g_2 = y^2 - z^2$; $x > y > z$.
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 $g_3 = xz^2 - yz^2$.

Computation of Gröbner bases

The standard **Buchberger Algorithm** to compute G follows easily from the previously stated property of G:

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

Output: Gröbner basis G of I

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

Computation of Gröbner bases

The standard **Buchberger Algorithm** to compute G follows easily from the previously stated property of G:
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 $P := \{\mathrm{Spol}(g_i, g_j) \mid g_i, g_j \in G, i > j\}$
4. Choose $p \in P$, $P := P \setminus \{p\}$

Computation of Gröbner bases

The standard **Buchberger Algorithm** to compute G follows easily from the previously stated property of G:

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 $P := \{\mathrm{Spol}(g_i, g_j) \mid g_i, g_j \in G, i > j\}$
4. Choose $p \in P$, $P := P \setminus \{p\}$
   (a) If $p \xrightarrow{G} 0 \Rightarrow$ **no new information**
   Go on with the next element in P.

Computation of Gröbner bases

The standard **Buchberger Algorithm** to compute G follows easily from the previously stated property of G:

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

Output: Gröbner basis G of I

1. $G = \emptyset$
2. $G := G \cup \{f_i\}$ for all $i \in \{1, \dots, m\}$
3. Set $P := \{\mathrm{Spol}(g_i, g_j) \mid g_i, g_j \in G, i > j\}$
4. Choose $p \in P$, $P := P \setminus \{p\}$
   (a) If $p \xrightarrow{G} 0 \Rightarrow$ **no new information**
   Go on with the next element in P.
   (b) If $p \xrightarrow{G} h \neq 0 \Rightarrow$ **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.
5. 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

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 $\langle f_1 \rangle$.

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

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
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.

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:

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:

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

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^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?

The following section is about

Signatures of polynomials

Faugère's idea is to give each polynomial during the computations of the algorithm a so-called **signature**:

Signatures of polynomials

Faugère's idea is to give each polynomial during the computations of the algorithm a so-called **signature**:

1. Assuming a polynomial p its signature is defined to be $\mathcal{S}(p) = (t, \ell)$ where t is its monomial and $\ell \in \mathbb{N}$ is its index.
2. A generating element f_i of I gets the signature $\mathcal{S}(f_i) = (1, \mathrm{i})$.
3. We have an **ordering** $\prec$ on the signatures:

$$(t_1, \ell_1) \succ (t_2, \ell_2) \quad \Leftrightarrow \quad \begin{array}{l} \text{(a)} \ell_1 > \ell_2 \text{ or} \\ \text{(b)} \ell_1 = \ell_2 \text{ and } t_1 > t_2 \end{array}$$

Signatures of polynomials

Faugère's idea is to give each polynomial during the computations of the algorithm a so-called **signature**:

1. Assuming a polynomial p its signature is defined to be $\mathcal{S}(p) = (t, \ell)$ where t is its monomial and $\ell \in \mathbb{N}$ is its index.
2. A generating element f_i of I gets the signature $\mathcal{S}(f_i) = (1, \mathrm{i})$.
3. We have an **ordering** $\prec$ on the signatures:

$$(t_1, \ell_1) \succ (t_2, \ell_2) \quad \Leftrightarrow \quad \begin{array}{l} \text{(a)}\ell_1 > \ell_2 \text{ or} \\ \text{(b)}\ell_1 = \ell_2 \text{ and } t_1 > t_2 \end{array}$$

Example

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

1. $(x^2y, 3) \succ (z^3, 3)$,
2. $(1, 5) \succ (x^{12}y^{234}z^{3456}, 4)$.

Signatures of polynomials

Remark

Note that there are other ways to define the ordering $\prec$ such that it prefers the degree of the monomial and not the index [MMT92].
Implementations of F5 with different orderings:

(a) 2009 Ars and Hashemi [AH09]

(b) 2010 Sun and Wang [SW10]

Signatures of polynomials

Remark

Note that there are other ways to define the ordering $\prec$ such that it prefers the degree of the monomial and not the index [MMT92]. Implementations of F5 with different orderings:

(a) 2009 Ars and Hashemi [AH09]

(b) 2010 Sun and Wang [SW10]

Using the signatures in the F5 Algorithm we also need to define them for s-polynomials:

$$\mathrm{Spol}(p,q) = \mathrm{lc}(q)u_p p - \mathrm{lc}(p)u_q q \text{ where } \mathcal{S}\left(\mathrm{Spol}(p,q)\right) = u_p\mathcal{S}(p)$$

where we assume that $u_p\mathcal{S}(p) \succ u_q\mathcal{S}(q)$.

Example revisited - with signatures

In our example

$$
\begin{aligned}
g_3 &= \mathrm{Spol}(g_2, g_1) = xg_2 - yg_1 \\
\Rightarrow \mathcal{S}(g_3) &= x\mathcal{S}(g_2) = x(1, 2) := (x, 2).
\end{aligned}
$$

Example revisited - with signatures

In our example

$$g_3 = \mathrm{Spol}(g_2, g_1) = xg_2 - yg_1$$
$$\Rightarrow \mathcal{S}(g_3) = x\mathcal{S}(g_2) = x(1,2) := (x, 2).$$

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

$$\mathcal{S}\left(\mathrm{Spol}(g_3, g_1)\right) = y\mathcal{S}(g_3) = (xy, 2).$$

Example revisited - with signatures

In our example

$$g_3 = \mathrm{Spol}(g_2, g_1) = xg_2 - yg_1$$
$$\Rightarrow \mathcal{S}(g_3) = x\mathcal{S}(g_2) = x(1,2) := (x,2).$$

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

$$\mathcal{S}\left(\mathrm{Spol}(g_3, g_1)\right) = y\mathcal{S}(g_3) = (xy, 2).$$

Note that $\mathcal{S}\left(\mathrm{Spol}(g_3, g_1)\right) = (xy, 2)$ and $\mathrm{lm}(g_1) = xy$.

Example revisited - with signatures

In our example

$$g_3 = \mathrm{Spol}(g_2, g_1) = xg_2 - yg_1$$
$$\Rightarrow \mathcal{S}(g_3) = x\mathcal{S}(g_2) = x(1,2) := (x,2).$$

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

$$\mathcal{S}\left(\mathrm{Spol}(g_3, g_1)\right) = y\mathcal{S}(g_3) = (xy, 2).$$

Note that $\mathcal{S}\left(\mathrm{Spol}(g_3, g_1)\right) = (xy, 2)$ and $\mathrm{lm}(g_1) = xy$.
$\Rightarrow$ In F5 we **know** that $\mathrm{Spol}(g_3, g_1)$ will reduce to zero!

How does this work?

How does this work?

Let $\mathrm{Spol}(p, q) = \mathrm{lc}(p) u_p p - \mathrm{lc}(q) u_q q$ with
$\mathcal{S}(p) = (s, k), \mathcal{S}(q) = (t, \ell)$.
Then $\mathrm{Spol}(p, q)$ does not need to be computed if

How does this work?

Let $\mathrm{Spol}(p, q) = \mathrm{lc}(p)u_p p - \mathrm{lc}(q)u_q q$ with $\mathcal{S}(p) = (s, k), \mathcal{S}(q) = (t, \ell)$.
Then $\mathrm{Spol}(p, q)$ does not need to be computed if

1. the leading monomial of some element $p \in G$ of index smaller k (ℓ) divides $u_p s$ ($u_q t$) **(Faugère's Criterion)**, or

How does this work?

Let $\mathrm{Spol}(p,q) = \mathrm{lc}(p)u_p p - \mathrm{lc}(q)u_q q$ with $\mathcal{S}(p) = (s,k), \mathcal{S}(q) = (t,\ell)$.
Then $\mathrm{Spol}(p,q)$ does not need to be computed if

1. the leading monomial of some element $p \in G$ of index smaller k (ℓ) divides $u_p s$ $(u_q t)$ **(Faugère's Criterion)**, or
2. the monomial of the signature of an element of index k (ℓ) divides $u_p s$ $(u_q t)$. **(Rewritten Criterion)**

How does this work?

Let $\mathrm{Spol}(p,q) = \mathrm{lc}(p)u_p p - \mathrm{lc}(q)u_q q$ with $\mathcal{S}(p) = (s,k), \mathcal{S}(q) = (t,\ell)$.
Then $\mathrm{Spol}(p,q)$ does not need to be computed if

1. the leading monomial of some element $p \in G$ of index smaller k (ℓ) divides $u_p s$ $(u_q t)$ **(Faugère's Criterion)**, or
2. the monomial of the signature of an element of index k (ℓ) divides $u_p s$ $(u_q t)$. **(Rewritten Criterion)**

Remark

1. F5's criteria are based on the signatures.
2. F5 computes degree-wise in each iteration step.

Difficulty of top-reduction in F5

On the one hand adding signatures to polynomials makes it possible to use these powerful criteria,
on the other hand we have to keep track of the signatures, i.e. we must be very careful when reducing elements.

Difficulty of top-reduction in F5

On the one hand adding signatures to polynomials makes it possible to use these powerful criteria,
on the other hand we have to keep track of the signatures, i.e. we must be very careful when reducing elements.

Remark

We will see in the following example that we do not only need to be careful **if we are allowed to reduce an element**, but also must be able to generate **new polynomials during reduction** when reducing with elements generated in the current iteration step.

Difficulty of top-reduction in F5

On the one hand adding signatures to polynomials makes it possible to use these powerful criteria,
on the other hand we have to keep track of the signatures, i.e. we must be very careful when reducing elements.

Remark

We will see in the following example that we do not only need to be careful **if we are allowed to reduce an element**, but also must be able to generate **new polynomials during reduction** when reducing with elements generated in the current iteration step.

Example

Assume the polynomial $p = xy^2 - z^3$ with $\mathcal{S}(p) = (t_p, \ell)$ and a possible reducer $q = y^2 - xz$ with $\mathcal{S}(q) = (t_q, \ell)$.

Difficulty of top-reduction in F5

On the one hand adding signatures to polynomials makes it possible to use these powerful criteria,
on the other hand we have to keep track of the signatures, i.e. we must be very careful when reducing elements.

Remark

We will see in the following example that we do not only need to be careful **if we are allowed to reduce an element**, but also must be able to generate **new polynomials during reduction** when reducing with elements generated in the current iteration step.

Example

Assume the polynomial $p = xy^2 - z^3$ with $\mathcal{S}(p) = (t_p, \ell)$ and a possible reducer $q = y^2 - xz$ with $\mathcal{S}(q) = (t_q, \ell)$.
In Buchberger-like implementations the top-reduction would take place, i.e. we would compute $p - xq$.

Difficulty of top-reduction in F5

Example

In F5 the following can happen:

1. If $x\mathcal{S}(q)$ satisfies Faugère's Criterion $\Rightarrow$ **no reduction**!

Difficulty of top-reduction in F5

Example

In F5 the following can happen:

1. If $x\mathcal{S}(q)$ satisfies Faugère's Criterion $\Rightarrow$ **no reduction**!
2. If $x\mathcal{S}(q)$ satisfies the Rewritten Criterion $\Rightarrow$ **no reduction**!

Difficulty of top-reduction in F5

Example

In F5 the following can happen:

1. If $x\mathcal{S}(q)$ satisfies Faugère's Criterion $\Rightarrow$ **no reduction**!
2. If $x\mathcal{S}(q)$ satisfies the Rewritten Criterion $\Rightarrow$ **no reduction**!
3. None of the above cases holds and $x\mathcal{S}(q) \prec \mathcal{S}(p) \Rightarrow p - xq$ is computed and gets the signature $\mathcal{S}(p)$.

Difficulty of top-reduction in F5

Example

In F5 the following can happen:

1. If $x\mathcal{S}(q)$ satisfies Faugère's Criterion $\Rightarrow$ **no reduction**!
2. If $x\mathcal{S}(q)$ satisfies the Rewritten Criterion $\Rightarrow$ **no reduction**!
3. None of the above cases holds and $x\mathcal{S}(q) \prec \mathcal{S}(p) \Rightarrow p - xq$ is computed and gets the signature $\mathcal{S}(p)$.
4. None of the first two cases holds and $x\mathcal{S}(q) \succ \mathcal{S}(p)$:

Difficulty of top-reduction in F5

Example

In F5 the following can happen:

1. If $x\mathcal{S}(q)$ satisfies Faugère's Criterion $\Rightarrow$ **no reduction**!
2. If $x\mathcal{S}(q)$ satisfies the Rewritten Criterion $\Rightarrow$ **no reduction**!
3. None of the above cases holds and $x\mathcal{S}(q) \prec \mathcal{S}(p) \Rightarrow p - xq$ is computed and gets the signature $\mathcal{S}(p)$.
4. None of the first two cases holds and $x\mathcal{S}(q) \succ \mathcal{S}(p)$:
   (a) p is **not reduced**, but searching for another possible reducer of it.

Difficulty of top-reduction in F5

Example

In F5 the following can happen:

1. If $x\mathcal{S}(q)$ satisfies Faugère's Criterion $\Rightarrow$ **no reduction**!
2. If $x\mathcal{S}(q)$ satisfies the Rewritten Criterion $\Rightarrow$ **no reduction**!
3. None of the above cases holds and $x\mathcal{S}(q) \prec \mathcal{S}(p) \Rightarrow p - xq$ is computed and gets the signature $\mathcal{S}(p)$.
4. None of the first two cases holds and $x\mathcal{S}(q) \succ \mathcal{S}(p)$:
   - (a) p is **not reduced**, but searching for another possible reducer of it.
   - (b) A new **s-polynomial** $r := xq - p$ where $\mathcal{S}(r) = x\mathcal{S}(q)$ is computed.

Redundant polynomials

Example

Assuming one of the first two cases of the previous example and moreover that there exists no other top-reducer of p we would end up with both, p and q being in G whereas clearly $\mathrm{lm}(q) \mid \mathrm{lm}(p)$. Thus p is **redundant** for G **at the moment it is added**.

Redundant polynomials

Example

Assuming one of the first two cases of the previous example and moreover that there exists no other top-reducer of p we would end up with both, p and q being in G whereas clearly $\mathrm{lm}(q) \mid \mathrm{lm}(p)$. Thus p is **redundant** for G **at the moment it is added**.

But...

For the F5 Algorithm itself and the criteria based on the signatures p could be necessary **in this iteration step**!
$\Rightarrow$ Disrespecting the way F5 top-reduces polynomials would harm the correctness of F5 **in this iteration step**!

The following section is about

Points of inefficiency

The difficulty of top-reduction in F5 leads to an **inefficiency**, namely we have way too many polynomials in the intermediate G_is

1. which are possible reducers, i.e. more checks for divisibility and the criteria have to be done, and
2. with which we compute new s-polynomials, i.e. more (for the resulting Gröbner basis redundant) data is generated.

Points of inefficiency

The difficulty of top-reduction in F5 leads to an **inefficiency**, namely we have way too many polynomials in the intermediate G_is

1. which are possible reducers, i.e. more checks for divisibility and the criteria have to be done, and
2. with which we compute new s-polynomials, i.e. more (for the resulting Gröbner basis redundant) data is generated.

Question

How can these two points be avoided as far as possible?

F5C: Computations with reduced GB

In 2009 John Perry & Christian Eder have implemented a new variant of the F5 Algorithm, called **F5C**.

F5C: Computations with reduced GB

In 2009 John Perry & Christian Eder have implemented a new variant of the F5 Algorithm, called **F5C**.
F5C uses the reduced Gröbner basis not only for reduction purposes, but also for the generation of new s-polynomials:

F5C: Computations with reduced GB

In 2009 John Perry & Christian Eder have implemented a new variant of the F5 Algorithm, called **F5C**.

F5C uses the reduced Gröbner basis not only for reduction purposes, but also for the generation of new s-polynomials:

1. Compute a Gröbner basis G_i of $\langle f_1, \ldots, f_i \rangle$.
2. Compute the reduced Gröbner basis B_i of G_i.
3. Compute a Gröbner basis G_{i+1} of $\langle f_1, \ldots, f_{i+1} \rangle$ where
   (a) B_i is used to build new s-polynomials with f_{i+1},
   (b) B_i is used to reduce polynomials.

F5C: Computations with reduced GB

In 2009 John Perry & Christian Eder have implemented a new variant of the F5 Algorithm, called **F5C**.
F5C uses the reduced Gröbner basis not only for reduction purposes, but also for the generation of new s-polynomials:

1. Compute a Gröbner basis G_i of $\langle f_1, \ldots, f_i \rangle$.
2. Compute the reduced Gröbner basis B_i of G_i.
3. Compute a Gröbner basis G_{i+1} of $\langle f_1, \ldots, f_{i+1} \rangle$ where
   (a) B_i is used to build new s-polynomials with f_{i+1},
   (b) B_i is used to reduce polynomials.

$\Rightarrow$ **Fewer reductions** and **fewer polynomials generated and considered** during the algorithm

How to use B_i for computations?

We have seen that **if we interreduce** G_i then **the current signatures are useless** in the following.

How to use B_i for computations?

We have seen that **if we interreduce** G_i then **the current signatures are useless** in the following.
$\Rightarrow$ If the current signatures are useless, then **throw them away** and **compute new useful ones**!

How to use B_i for computations?

We have seen that **if we interreduce** G_i then **the current signatures are useless** in the following.
$\Rightarrow$ If the current signatures are useless, then **throw them away** and **compute new useful ones**!

Recomputation of signatures

How to use B_i for computations?

We have seen that **if we interreduce** G_i then **the current signatures are useless** in the following.
$\Rightarrow$ If the current signatures are useless, then **throw them away** and **compute new useful ones**!

Recomputation of signatures

1. Delete all signatures.
2. Interreduce G_i to B_i.
3. For each element $g_k \in B_i$ set $\mathcal{S}(g_k) = (1, \mathrm{k})$.
4. For all elements $g_j, g_k \in B_i$ **recompute signatures** for $\mathrm{Spol}(g_j, g_k)$.
5. Start the next iteration step with f_{i+1} by computing all s-polynomials with elements from B_i.

How to use B_i for computations?

We have seen that **if we interreduce** G_i then **the current signatures are useless** in the following.
$\Rightarrow$ If the current signatures are useless, then **throw them away** and **compute new useful ones**!

Recomputation of signatures

1. Delete all signatures.
2. Interreduce G_i to B_i.
3. For each element $g_k \in B_i$ set $\mathcal{S}(g_k) = (1, \mathrm{k})$.
4. For all elements $g_j, g_k \in B_i$ **recompute signatures** for $\mathrm{Spol}(g_j, g_k)$.
5. Start the next iteration step with f_{i+1} by computing all s-polynomials with elements from B_i.

Implementations

Three free available implementations:

1. F5 & F5C as a SINGULAR library (Perry & Eder)
2. F5 & F5C implemented in Python for Sage (Perry & Albrecht): **F4-ish** reduction possible.
3. F5 & F5C implementation in the SINGULAR kernel: **under development**

Comparing F5 and F5C

We are comparing F5 and F5C in the way that we use the **same implementation** of the **core algorithm** for all variants.

Comparing F5 and F5C

We are comparing F5 and F5C in the way that we use the **same implementation** of the **core algorithm** for all variants.

Moreover we do not only compare

1. **timings**, but also
2. the **number of reductions**, and
3. the **number of polynomials generated**.

Comparing F5 and F5C

Instead of the timings themselves we present the ratios of the timings comparing the two variants.

Comparing F5 and F5C

Instead of the timings themselves we present the ratios of the timings comparing the two variants.

system	F5C / F5
Katsura 7	1.06
Katsura 8	0.83
Katsura 9	0.62
Schrans-Troost	0.71
Cyclic 6	0.60
Cyclic 7	0.49
Cyclic 8	0.62

SINGULAR 3.1.0, kernel implementation; Linux-gentoo-r8 2009 x86_64, Intel Xeon @ 3.16 GHz, 64 GB RAM

Number of reductions

system	# red in F5	# red in F5C
Katsura 4	774	222
Katsura 5	14,597	3,985
Katsura 6	1,029,614	58,082
Cyclic 5	512	446
Cyclic 6	41,333	14,167

Sage 3.2.1, Python implementation; Ubuntu Linux 8.10, Intel Core 2 Quad @ 2.66 GHz, 3 GB RAM

Number of polynomials generated

In the following we present internal data from the computation of Katsura 9.

Number of polynomials generated

In the following we present internal data from the computation of Katsura 9.

i	# G_i in F5	# G_i in F5C
2	2	2
3	4	4
4	8	8
5	16	15
6	32	29
7	60	51
8	132	109
9	524	472
10	1,165	778

Sage 3.2.1, Python implementation; Ubuntu Linux 8.10, Intel Core 2 Quad @ 2.66 GHz, 3 GB RAM

▸ Skip

The following section is about

Difficulty of top-reduction revisited

Remember that due to F5's criteria **redundant elements** are added to G.

Difficulty of top-reduction revisited

Remember that due to F5's criteria **redundant elements** are added to G.
$\Rightarrow$ What happens if F5 computes infinitely many redundant elements?

Difficulty of top-reduction revisited

Remember that due to F5's criteria **redundant elements** are added to G.

$\Rightarrow$ What happens if F5 computes infinitely many redundant elements?

Termination of F5?

Do we need those redundant elements?

Do we need those redundant elements?

Yes, we do!

Do we need those redundant elements?

Yes, we do!

F5 works degree-wise in each iteration step.

Do we need those redundant elements?

Yes, we do!

F5 works degree-wise in each iteration step.
Assume that $\mathrm{lm}(p_k) \mid \mathrm{lm}(p_i)$, but the corresponding reductions have not taken place when p_i was computed.

Do we need those redundant elements?

Yes, we do!

F5 works degree-wise in each iteration step.
Assume that $\mathrm{lm}(p_k) \mid \mathrm{lm}(p_i)$, but the corresponding reductions have not taken place when p_i was computed.

1. If there is an element p_m such that $\mathrm{lm}(p_i) \mid \mathrm{lm}(p_m)$, we possibly need p_i as reducer.

Do we need those redundant elements?

Yes, we do!

F5 works degree-wise in each iteration step.
Assume that $\mathrm{lm}(p_k) \mid \mathrm{lm}(p_i)$, but the corresponding reductions have not taken place when p_i was computed.

1. If there is an element p_m such that $\mathrm{lm}(p_i) \mid \mathrm{lm}(p_m)$, we possibly need p_i as reducer.
2. If $\mathrm{Spol}(p_i, p_j) = \lambda \mathrm{Spol}(p_k, p_j) + \sum_s \lambda_s p_s$ and $\mathrm{Spol}(p_k, p_j)$ **is not computed**, we need $\mathrm{Spol}(p_i, p_j)$.

Resolving the termination issue

Definition

An s-polynomial $\mathrm{Spol}(p, q)$ is called an **F5-s-polynomial** if either $\mathrm{lm}(p)$ or $\mathrm{lm}(q)$ is redundant in G.

Resolving the termination issue

Definition

An s-polynomial $\mathrm{Spol}(p, q)$ is called an **F5-s-polynomial** if either $\mathrm{lm}(p)$ or $\mathrm{lm}(q)$ is redundant in G.

Otherwise $\mathrm{Spol}(p, q)$ is called a **GB-s-polynomial**.

Resolving the termination issue

Definition

An s-polynomial $\mathrm{Spol}(p, q)$ is called an **F5-s-polynomial** if either $\mathrm{lm}(p)$ or $\mathrm{lm}(q)$ is redundant in G.
Otherwise $\mathrm{Spol}(p, q)$ is called a **GB-s-polynomial**.

Example

Recall the last slide: Assume that $\mathrm{lm}(p_j)$ and $\mathrm{lm}(p_k)$ are non-redundant in G.
Then $\mathrm{Spol}(p_i, p_j)$ is an F5-s-polynomial as $\mathrm{lm}(p_k) \mid \mathrm{lm}(p_i)$, whereas $\mathrm{Spol}(p_k, p_j)$ is an GB-s-polynomial.

Resolving the termination issue

1. After finitely many steps only F5-s-polynomials are left.

Resolving the termination issue

1. After finitely many steps only F5-s-polynomials are left.
2. No GB-s-polynomial can be generated from this point onwards.

Resolving the termination issue

1. After finitely many steps only F5-s-polynomials are left.
2. No GB-s-polynomial can be generated from this point onwards.
3. We can go on with the next iteration step / terminate F5.

Resolving the termination issue

1. We label each computed polynomial by a boolean value to distinguish redundant and non-redundant ones.

Resolving the termination issue

1. We label each computed polynomial by a boolean value to distinguish redundant and non-redundant ones.
2. We add a global variable d in the implementation storing the highest known degree GB-s-polynomials.

Resolving the termination issue

1. We label each computed polynomial by a boolean value to distinguish redundant and non-redundant ones.
2. We add a global variable d in the implementation storing the highest known degree GB-s-polynomials.
3. When computing new s-polynomials we have to check and possibly change d's value.

Resolving the termination issue

1. We label each computed polynomial by a boolean value to distinguish redundant and non-redundant ones.
2. We add a global variable d in the implementation storing the highest known degree GB-s-polynomials.
3. When computing new s-polynomials we have to check and possibly change d's value.
4. If the degree of the next bunch of s-polynomials to be computed is greater than d, we go to the next iteration step / terminate the algorithm.

References

G. Ars and A. Hashemi.
Extended F5 Criteria

B. Buchberger.
Ein Algorithmus zum Auffinden der Basiselement des Restklassenrings nach einem nulldimensionalen Polynomideal

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

R. Gebauer and H.M. Möller.
On an Installation of Buchberger's Algorithm

W. Decker, G.-M. Greuel, G. Pfister and H. Schönemann.
SINGULAR 3-1-1. *A computer algebra system for polynomial computations*, University of Kaiserslautern, 2010, `http://www.singular.uni-kl.de`.

H. M. Möller, T. Mora and C. Traverso.
Gröbner bases computation using syzygies

W. A. Stein et al.
Sage Mathematics Software (Version3.2.1), The Sage Development Team, 2008, `http://www.sagemath.org`.

T. Stegers.
Faugère's F5 Algorithm Revisited

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