Implicitization with Gröbner Bases

the well known algorithm
and algorithms which work

John Abbott, Anna M. Bigatti, Lorenzo Robbiano

SC^2: H2020-FETOPEN-2016-2017-CSA project 712689 `www.sc-square.org`

A well known problem

Implicit

K a field, $\quad f_1, ..., f_n \in K[t_1, \ldots, t_s]$

$$\varphi : K[x_1, ..., x_n] \longrightarrow K[t_1, \ldots, t_s] \quad \text{given by} \quad x_i \mapsto f_i$$

Problem: find a set of generators for **Implicit($f_1, \ldots, f_n$)** := ker(φ)

Literature is vast, many different techniques
(Busé, Chardin, D'Andrea, Dickenstein, Emiris, Orecchia, Wang..)

Gröbner Bases → elim: well known solution, simple and elegant 😀

Example 1

$f_1 = t_1^2, \quad f_2 = t_1 t_2, \quad f_3 = t_2^2$ polynomials in $K[t_1, t_2]$

```
/**/ use QQ[t[1..2], x[1..3]];      f := [t[1]^2, t[1]*t[2], t[2]^2];
/**/ elim([t[1],t[2]],  ideal([x[i]-f[i] | i in 1..3]) );
ideal(x[2]^2 -x[1]*x[3])
```

Implicit hypersurface

... but elimination is slow and memory hungry 😩

Indeed it is quite common that an elegant general solution based on Gröbner Bases does not work in practice. Need specialized fine tuning!

Now suppose we know that 🔮
the parametrization $(f_1, ..., f_n)$ gives a **hypersurface**:
i.e. $J = \text{Implicit}(f_1, \ldots, f_n) = (g)$ is principal $\Longrightarrow g$ is (J prime) irreducible

Remark 🔮

Recall: $f_1, ..., f_n \in K[t_1, \ldots, t_s]$
if $s = n - 1$, $J = \text{Implicit}(f_1, \ldots, f_n)$ has *at least one* generator.
Typically if $s = n - 1$, J has *one* generator.

🤔 Can this assumption **help** the Gröbner basis computation?

Implicit hypersurface

... but elimination is slow and memory hungry 😩

Indeed it is quite common that an elegant general solution based on Gröbner Bases does not work in practice. Need specialized fine tuning!

Now suppose we know that 🔮
the parametrization $(f_1, ..., f_n)$ gives a **hypersurface**:
i.e. $J = \text{Implicit}(f_1, \ldots, f_n) = (g)$ is principal $\Longrightarrow$ g is (J prime) irreducible

Remark 🔮

Recall: $f_1, ..., f_n \in K[t_1, \ldots, t_s]$
if $s = n - 1$, $J = \text{Implicit}(f_1, \ldots, f_n)$ has *at least one* generator.
Typically if $s = n - 1$, J has *one* generator.

🤔 Can this assumption **help** the Gröbner basis computation?

1: Homogenization

Proposition

$f_1, \dots, f_n \in K[t_1, \dots, t_s] \setminus K$.
Let h be a new indeterminate, and in $K[t_1, \dots, t_s, h, x_1, \dots, x_n]$
$\deg(t_i) = \deg(h) = 1$ and homogenize $f_i \to f_i^{\text{hom}}$
$\deg(x_i) = \deg(f_i)$ and J be the homog ideal $\langle x_1 - f_1^{\text{hom}}, \dots, x_n - f_n^{\text{hom}} \rangle$.
Then $\text{Implicit}(f_1, \dots, f_n) = (J \cap K[h, x_1, \dots, x_n])^{\text{deh}}$

Example 2

```
/**/ P := NewPolyRing(QQ, "t[1],t[2], h, x[1],x[2],x[3]",
        MakeTermOrd(RowMat([1,  1,  1,  2,  2,  1])), 1);  use P;
/**/ f  := [t[1]^2 -2,      t[1]*t[2] -t[1],    t[2] +1];
/**/ fh := [t[1]^2 -2*h^2,  t[1]*t[2] -t[1]*h,  t[2] +h];
/**/ E  := elim([t[1],t[2]],  ideal([x[i]-f[i]  | i in 1..3]) );
/**/ Eh := elim([t[1],t[2]],  ideal([x[i]-fh[i] | i in 1..3]) );
/**/ E = ideal(subst(gens(Eh), h, 1));  // --> true
```

2: Truncation $\rightarrow$ *ElimTH* algorithm

Input $f_1, \dots, f_n \in K[t_1, \dots, t_s] \setminus K$ such that $\text{Implicit}(f_1, \dots, f_n)$ principal

ElimTH-1 *Initialization:*

- Create the ring $R = K[t_1, \dots, t_s,\ h,\ x_1, \dots, x_n]$
  graded by $[1, \dots, 1,\ 1,\ \deg(f_1), \dots, \deg(f_n)]$
  with σ elimination ordering for $\{t_1, \dots, t_s\}$
- Let $J = \langle x_1 - f_1^{\text{hom}}, \dots, x_n - f_n^{\text{hom}} \rangle$

ElimTH-2 *Main Loop:*

Start Buchberger's algorithm for a σ-Gröbner basis of J
Work degree by degree (*i.e.* always choose pair with min degree)
When you find G such that $\text{LT}_\sigma(G)$ not divisible by any t_i exit loop

Output $g = G^{\text{deh(h)}} \in K[x_1, \dots, x_n] \longrightarrow$ generator of $\text{Implicit}(f_1, \dots, f_n)$.

2: Truncation $\rightarrow$ *ElimTH* algorithm

Input $f_1, \ldots, f_n \in K[t_1, \ldots, t_s] \setminus K$ such that Implicit$(f_1, \ldots, f_n)$ principal

ElimTH-1 *Initialization:*

- Create the ring $R = K[t_1, \ldots, t_s,\ h,\ x_1, \ldots, x_n]$ graded by $[1, \ldots, 1,\ 1,\ \deg(f_1), \ldots, \deg(f_n)]$ with σ elimination ordering for $\{t_1, \ldots, t_s\}$
- Let $J = \langle x_1 - f_1^{\text{hom}}, \ldots, x_n - f_n^{\text{hom}} \rangle$

ElimTH-2 *Main Loop:*

Start Buchberger's algorithm for a σ-Gröbner basis of J
Work degree by degree (*i.e.* always choose pair with min degree)
When you find G such that $\mathrm{LT}_\sigma(G)$ not divisible by any t_i exit loop

Output $g = G^{\text{deh(h)}} \in K[x_1, \ldots, x_n] \longrightarrow$ generator of Implicit$(f_1, \ldots, f_n)$.

3: Linear algebra → *Direct* algorithm

Inspired by Buchberger-Möller algorithm for computing Gröbner bases of *ideal of points*

Direct Method: rough idea

Recall $\varphi : K[x_1, ..., x_n] \longrightarrow K[t_1, \ldots, t_s]$ given by $x_i \mapsto f_i$

$T_1, T_2, T_3, \ldots$ **all** power-products of $K[x_1, ..., x_n]$ in **increasing order**
For $k = 1, 2, 3, \ldots$ do

- Check for **linear dependency:** $\boldsymbol{\sum_{i=1}^{k} a_i\, \varphi(T_i) = 0}$
- If found, return corresponding polynomial $\sum_{i=1}^{k} a_i T_i$

- use **enumerative** term-ordering on $K[x_1, \ldots, x_n]$
  every power-product T appears at a finite position
  Ex: `Lex` is not enumerative `DegRevLex` is enumerative
- detect linear dependency using gaussian elimination
  build “row-reduced echelon form” incrementally

The implementation in CoCoA

These algorithms are implemented in CoCoALib (and CoCoA-5).

Examples	ElimTH 32003	ElimTH 0	Direct 32003	Direct 0	Len
Ex d'Andrea	0	0.009	0	0.003	6
Ex Orecchia	0	0.007	0	0.002	9
Ex Enneper	0	0.0258	0	0.0256	57
Ex Robbiano	0.273	0.597	0.0251	0.118	319
Ex Buse1	0	0.0251	0	0.070	13
Ex Buse2	0	0.228	0	0.083	56
Ex Wang	1.196	16.278	0.159	7.707	715
Ex Dickenstein1	0	0.060	0	0.0252	41
Ex Dickenstein4	0	0.943	0	0.934	161
Ex Bohemian	0	0.011	0	0.004	7
Ex Sine	0	0.012	0	0.010	7

5: Modular Methods

Avoid coefficient swelling in computations over $\mathbb{Q}$ using Modular Methods:

Modular Method: general structure with Chinese Remaindering (CRT)

- Input: $f_1, \ldots, f_n$ with coefficients in $\mathbb{Q}$
- *Main Loop:*
  - Pick a new "suitable" prime p
  - Reduce to $\bar{f}_1, \ldots, \bar{f}_n$ over $\mathbb{F}_p$
  - Compute result modulo $p \longrightarrow$ fast! 👍 👍
  - If not bad prime, CRT-combine result with earlier results
  - If enough primes, exit loop
- Reconstruct answer with coefficients in $\mathbb{Q}$

General problems with (CRT) modular methods 🤔

- How to detect bad primes? (non-compatible results)
- How many CRT iterations?

Example 3 (Bad prime: wrong degree)

Implicit$(t_1^3,\ t_2^3,\ t_1+t_2) =$

$\mathbb{Q}$: $\langle -x_3^9+3x_1x_3^6+3x_2x_3^6-3x_1^2x_3^3+21x_1x_2x_3^3-3x_2^2x_3^3+x_1^3+3x_1^2x_2+3x_1x_2^2+x_2^3\rangle$

$\mathbb{F}_3$: $\langle x_3^3 - x_1 - x_2\rangle$

Example 4 (Bad prime: not principal)

Implicit$(t_1+t_2,\ t_1-t_2,\ t_1-t_2) =$ $\mathbb{Q}$: $\langle x_2 - x_3\rangle$
$\mathbb{F}_2$: $\langle x_1 - x_3,\ x_2 - x_3\rangle$

General problems with (CRT) modular methods 🤔

- How to detect bad primes? (non-compatible results)
  💬 *Answer for Implicit*: use fault-tolerant rational reconstruction (Abbott **HRR**: Heuristic Rational Reconstruction)
- How many CRT iterations?
  💬 *Answer for Implicit*: verify result $(g(f_1, \ldots, f_n) = 0)$

Example 3 (Bad prime: wrong degree)

$\mathrm{Implicit}(t_1^3,\ t_2^3,\ t_1 + t_2) =$

$\mathbb{Q}$: $\langle -x_3^9+3x_1x_3^6+3x_2x_3^6-3x_1^2x_3^3+21x_1x_2x_3^3-3x_2^2x_3^3+x_1^3+3x_1^2x_2+3x_1x_2^2+x_2^3 \rangle$

$\mathbb{F}_3$: $\langle x_3^3 - x_1 - x_2 \rangle$

Example 4 (Bad prime: not principal)

$\mathrm{Implicit}(t_1+t_2,\ t_1-t_2,\ t_1-t_2) =$ $\mathbb{Q}$: $\langle x_2 - x_3 \rangle$; $\mathbb{F}_2$: $\langle x_1 - x_3,\ x_2 - x_3 \rangle$

General problems with (CRT) modular methods 🤔

- How to detect bad primes? (non-compatible results)
  💬 *Answer for Implicit*: use fault-tolerant rational reconstruction (Abbott **HRR**: Heuristic Rational Reconstruction)
- How many CRT iterations?
  💬 *Answer for Implicit*: verify result ($g(f_1, \ldots, f_n) = 0$)

Implicit: New tests

Examples	ElimTH 32003	ElimTH 0	Direct 32003	Direct 0	Len
Ex 13-Poly	2.1	6.9 (3)	0.1	0.4 (3)	471
Ex 14-Poly	∞	∞	8.41	58.2 (5)	6398
Ex 15-Poly	20.3	55.7 (5)	0.9	3.4 (5)	1705
Ex 16-Poly	∞	∞	58.4	204.1 (3)	4304
Ex 17-Poly	1.4	4.8 (3)	9.1	27.9 (3)	1763
Ex 18-Poly	60.8	∞	228.0	∞	9360
Ex 19-Poly	2.2	9.3 (3)	47.3	148.9 (3)	5801
Ex 20-Poly	5.0	71.5 (6)	∞	∞	6701
Ex 21-Poly	10.2	121.0 (11)	36.4	418.5 (11)	2356
Ex 1-RatFun	0.1	1.370 (4)	0.1	1.2 (4)	62
Ex 2-RatFun	0.6	2.8 (2)	1.1	3.8 (2)	57
Ex 3-RatFun	0.6	13.4 (3)	2.1	17.9 (3)	115
Ex 4-RatFun	10.4	159.1 (3)	64.8	335,0 (3)	189
Ex 5-RatFun	63.3	141.7 (2)	46.2	101.6 (2)	149
Ex 6-RatFun	116.4	761.4 (6)	202.7	1214.4(6)	2692

∞ = more than 20 minutes