Journal of Symbolic Computation 81 (2017) 20–40

Contents lists available at ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Implicitization of hypersurfaces ☆

John Abbott [1], Anna Maria Bigatti, Lorenzo Robbiano

Dip. di Matematica, Università degli Studi di Genova, Via Dodecaneso 35, I-16146 Genova, Italy

ARTICLE INFO

Article history:
Received 12 February 2016
Accepted 20 September 2016
Available online 14 November 2016

MSC:
13P25
13P10
13-04
14Q10
68W30

Keywords:
Hypersurface
Implicitization

ABSTRACT

We present new, practical algorithms for the hypersurface implicitization problem: namely, given a parametric description (in terms of polynomials or rational functions) of the hypersurface, find its implicit equation. Two of them are for polynomial parametrizations: one algorithm, "ElimTH", has as main step the computation of an elimination ideal via a *truncated, homogeneous* Gröbner basis. The other algorithm, "Direct", computes the implicitization directly using an approach inspired by the generalized Buchberger–Möller algorithm. Either may be used inside the third algorithm, "RatPar", to deal with parametrizations by rational functions. Finally we show how these algorithms can be used in a modular approach, algorithm "ModImplicit", for avoiding the high costs of arithmetic with rational numbers. We exhibit experimental timings to show the practical efficiency of our new algorithms.

1. Introduction

Let K be a field and let $P = K[x_1, \ldots, x_n]$ be a polynomial ring in n indeterminates. Then let $f_1, \ldots, f_n$ be elements in the field $L = K(t_1, \ldots, t_s)$, where $\{t_1, \ldots, t_s\}$ is another set of indeterminates which are viewed as parameters. We consider the K-algebra homomorphism

$$\varphi : K[x_1, \ldots, x_n] \longrightarrow K(t_1, \ldots, t_s) \quad \text{given by} \quad x_i \mapsto f_i \quad \text{for } i = 1, \ldots, n$$

☆ This research was partly supported by the "National Group for Algebraic and Geometric Structures, and their Applications" (GNSAGA–INdAM).

E-mail addresses: abbott@dima.unige.it (J. Abbott), bigatti@dima.unige.it (A.M. Bigatti), robbiano@dima.unige.it (L. Robbiano).

[1] J. Abbott is a Marie Curie fellow of the Istituto Nazionale di Alta Matematica.

http://dx.doi.org/10.1016/j.jsc.2016.11.002
0747-7171/

All the polynomial equations satisfied by the f_i's

Its kernel, which will be denoted by Implicit$(f_1, \ldots, f_n)$, is a prime ideal, and the general problem of implicitization is to find a set of generators for this ideal.

The task of computing Implicit$(f_1, \ldots, f_n)$ can be solved by computing a suitable Gröbner basis (see Proposition 3.1). However, in practice this method does not work well since in most non-trivial cases it is far too slow. The poor computational speed is aggravated when computing with rational coefficients (rather than coefficients from a finite field).

There is definitely a big need for new, efficient techniques, and many authors have investigated alternative ways. The literature about implicitization is so vast that it is almost impossible to mention the entire body of research on this topic. This interest derives from the fact that the parametric representation of a rational variety is important for generating points on it, while the implicit representation is used to check whether a point lies on it. Besides its theoretical importance, the double representation of a rational variety is used intensively for instance in Computer Aided Geometric Design. A good source of bibliography up to ten years ago is Bastl and Ježek (2005). More recently, new ideas have emerged. As we said, it is almost impossible to cite all of them, and we content ourselves to mention a few. In particular, new methods for computing implicitizations have been described in Busé and Ba (2010), Busé and Chardin (2005), d'Andrea (2015), and Orecchia and Ramella (2015). Some of these new ideas respond to the fact that in many cases the computation of Implicit$(f_1, \ldots, f_n)$ is too hard, hence one seeks a way to check whether a point lies on the rational variety without actually computing its equations.

So, what is the content of this paper? And what are the novelties and the new algorithms presented here? First of all, we concentrate on the "hypersurface case" where Implicit$(f_1, \ldots, f_n)$ is a principal ideal, and hence generated by an irreducible polynomial which is therefore unique up to an invertible constant factor.

Remark 1.1. Let I be the ideal Implicit$(f_1, \ldots, f_n)$ and let m be the minimum number of generators for I. From the facts that $\dim(K[f_1, \ldots, f_n]) \leq \dim(K[t_1, \ldots, t_s]) = s$ and $\dim(K[f_1, \ldots, f_n]) = \dim(K[x_1, \ldots, x_n]/I) \geq n - m$ it follows that $m \geq n - s$. So, whenever $s \leq n - 1$ then I has at least one generator, and in particular I is non-zero.

The hypersurface case typically arises when $s = n - 1$, in accordance with the remark above. However, this is not always the case, as the following examples show.

Example 1.2. We consider the "atypical" case where $n = s = 2$ and $f_1 = f_2 = t_1 + t_2$. Clearly we have Implicit$(f_1, f_2) = \langle x - y \rangle$, which is obviously principal. This does, however, become a typical case if we use a "better parametrization" in terms of $u = t_1 + t_2$, where we then have $f_1 = f_2 = u$, and consequently also have $s = n - 1$ with this better parametrization.

Another "atypical" example is the following. Let $n = 3$, $s = 2$ and $f_1 = \frac{t_2^2}{t_1^2}$, $f_2 = \frac{t_1^2 + t_2^2}{t_1^2}$, $f_3 = \frac{t_1^2 + t_1 t_2 + t_2^2}{t_1^2}$. Here we do have $s = n - 1$ but the implicitization is not principal, in fact it turns out that Implicit$(f_1, f_2, f_3) = \langle x_1 + x_2 - 1,\ x_2^2 - 2x_2x_3 + x_3^2 - x_2 - 1 \rangle$. The reason here is that there is a "better parametrization" in terms of $u = \frac{t_2}{t_1}$, where we have $f_1 = u^2$, $f_2 = u^2 + 1$, $f_3 = u^2 + u + 1$; and with this better parametrization we have $s \neq n - 1$.

In this paper we do not examine the interesting question of finding a good parametrization, which is a problem of a quite different nature.

The ideas explored in this paper can be summarized in the following way:

- Exploit homogenization to improve elimination (RatPar, ElimTH):
  using elimination is known to be an elegant but impractical way to achieve implicitization. We show that any problem (polynomial or rational parametrization) can be homogenized (see Proposition 3.3 and Theorem 5.3). Thereafter, the result is given by the first polynomial not involving any of the parameters, so the computation can be stopped as soon as it is found, avoiding the remaining "useless reductions".

- Use a direct algorithm which does not need elimination (Direct):
  Wang (2004) described an algorithm based on searching for a linear relationship among the images of the power-products. We refine this idea and make it incremental, thus leading to several important insights and optimizations (see Subsection 4.2).
- When the coefficient field is $\mathbb{Q}$ use modular methods (ModImplicit):
  computing the solution polynomial modulo several primes, and then obtaining the solution over $\mathbb{Q}$ by Chinese Remaindering is a powerful tool, but needs to be fine-tuned to any specific context. In Section 6 we use an incremental approach combined with fault-tolerant rational reconstruction to resolve the problem of how many primes are needed and to "tolerate" computations with bad primes (some of which cannot be detected *a priori*).

Remark 1.3. Regarding the first item, one of the referees pointed out that we made no reference to the *"projective view of the implicitization problem, which is relatively classical"*. The main reason was to avoid complications for the typical computer-algebra people who, generally, are much more familiar with algebra than geometry. Nevertheless, let us give some hints in this direction to the interested reader.

If $f_1, \dots, f_n \in K[t_1, \dots, t_s]$ then the map φ can be seen as the algebraic counterpart of the map of the affine schemes $\Phi : \mathbb{A}_s \longrightarrow \mathbb{A}_n$. We let $d_i = \deg(f_i)$ for $i = 1, \dots, n$, then homogenize the f_i with a new indeterminate h such that $\deg(h) = 1$, and set $\deg(x_i) = \deg(f_i)$. Now we consider the projective space $\mathbb{P}_s$ with coordinates $t_1, \dots, t_s, h$ and the weighted projective space $\mathbb{P}(d_1, \dots, d_n, 1)$ with coordinates $x_1, \dots, x_n, h$ (see for instance Beltrametti and Robbiano, 1986 for an introduction to the theory of weighted projective spaces). The map Φ can be viewed as the restriction to $\mathbb{A}_s$ of the *rational map* $\Psi : \mathbb{P}_s \dashrightarrow \mathbb{P}(d_1, \dots, d_n, 1)$ given by $[t_1 : t_2 : \cdots : t_s : h] \to [f_1^{\hom(h)} : f_2^{\hom(h)} : \cdots : f_s^{\hom(h)} : h]$. Observe that Ψ is a rational map, but not necessarily a map, since it may have a non-trivial base locus. The algebraic explanation of this fact is exactly the proof of Proposition 3.3.

The situation is more complicated when $f_1, \dots, f_n \in K(t_1, \dots, t_s)$. Using a common denominator, we may assume that $f_i = \frac{p_i}{q}$ with $f_i, q \in K[t_1, \dots, t_s]$ for $i = 1, \dots, n$. If we let D_q denote the open subscheme $\mathbb{A}_s \setminus \{q = 0\}$, then φ can be seen as the algebraic counterpart of the map of affine schemes $\Phi : D_q \longrightarrow \mathbb{A}_n$, and the standard way to proceed is to take care of this limitation, as explained in Remark 5.1.

But there is a different way to interpret Implicit$(f_1, \dots, f_s)$. We let $d_i = \deg(p_i)$ for $i = 1, \dots, n$ and $d_0 = \deg(q)$. Then we let $d = \max\{\deg(q), \deg(p_1), \dots, \deg(p_n)\}$, let $Q = q^{\hom_d(h)}$, $P_i = p_i^{\hom_d(h)}$ for $i = 1, \dots, n$ (see Definition 2.5), so that all the polynomials $Q, P_1, \dots, P_s$ are homogeneous of the same degree d. Next we consider the projective space $\mathbb{P}_s$ with coordinates $t_1, \dots, t_s, h$ and the projective space $\mathbb{P}_n$ with coordinates $x_0, x_1, \dots, x_n$. If we let $\mathbb{A}_n$ be the affine open chart of $\mathbb{P}_n$ defined by $x_0 \neq 0$, the map Φ can be interpreted as the restriction to D_q of the corresponding rational map $\Psi : \mathbb{P}_s \dashrightarrow \mathbb{P}_n$ defined by $[t_1 : t_2 : \cdots : t_s : h] \to [Q : P_1 : \cdots : P_s]$. The algebraic explanation of this fact is exactly the proof of Theorem 5.3.

Why not try to use other embeddings into suitable projective or weighted projective spaces, as we do in the case of polynomial parametrizations? The reason is explained in all the remarks and examples following Theorem 5.3.

There is a further idea: computing implicitizations with constraints, in particular using a method of "slicing" the variety with suitable parallel hyperplanes. This technique was introduced and used in Robbiano (2015). However, it is rarely better than our new methods when Implicit$(f_1, \dots, f_n)$ is a hypersurface. We shall investigate the "implicitizations with constraints" for the general case in a later paper.

The algorithms described in this paper are implemented in CoCoALib (Abbott and Bigatti, 2016), and are also available in CoCoA 5 (Abbott et al., 2016). With our new methods most of the examples mentioned in the literature become "easy", that is we can compute the implicitization in less than a second — see Table 1 in Section 7. Consequently, we introduce new, challenging examples, and the last table shows the performance of our implementation.

We thank the referees for their useful comments and suggestions which helped us to improve this paper.

2. Notation and terminology

Here we introduce the notation and terminology we shall use.

Definition 2.1. Let K be a field, and let $P = K[x_1, \ldots, x_n]$ be a polynomial ring in n indeterminates. Let $t_1, \ldots, t_s$ be further indeterminates which are viewed as "parameters". Given elements $f_1, ..., f_n$ in $K[t_1, \ldots, t_s]$, we define the ideal $J = \langle x_1 - f_1, \ldots, x_n - f_n \rangle$ of the ring $P[t_1, \ldots, t_s]$ to be the **eliminating ideal** of the n-tuple $(f_1, \ldots, f_n)$.

The kernel of the K-algebra homomorphism $\varphi : P \longrightarrow K[t_1, \ldots, t_s]$ which sends $x_i \mapsto f_i$ for every $i = 1, \ldots, n$ will be called **Implicit**$(\boldsymbol{f_1, \ldots, f_n})$.

Definition 2.2. We extend naturally Definition 2.1 to parametrizations by rational functions. Let $\frac{p_1}{q}, \ldots, \frac{p_n}{q}$ be rational functions in the field $L = K(t_1, \ldots, t_s)$ with common denominator q; so that we have $q, p_1, \ldots, p_n \in K[t_1, \ldots, t_s]$.

We define **Implicit**$(\frac{\boldsymbol{p_1}}{\boldsymbol{q}}, \ldots, \frac{\boldsymbol{p_n}}{\boldsymbol{q}})$ to be the kernel of the K-algebra homomorphism $\varphi : P \longrightarrow L$ which sends $x_i \mapsto \frac{p_i}{q}$ for $i = 1, \ldots, n$.

Definition 2.3. An **enumerative ordering** is a total ordering such that for every element there are only finitely many elements smaller than it. In particular, an **enumerative term-ordering** is a term-ordering which is also enumerative; consequently, an enumerative term-ordering is defined by a matrix with strictly positive entries in the first row.

Example 2.4. Any degree-compatible term-ordering is enumerative because for any power-product $\bar{T}$ all smaller power-products, $T < \bar{T}$, must have $\deg(T) \leq \deg(\bar{T})$, and so they are finite in number. In contrast, the lex-ordering (for 2 or more indeterminates) is not enumerative because if indeterminate x_2 is less than x_1 then all powers x_2^d are smaller than x_1.

Definition 2.5. In section 5 we shall use two different kinds of homogenization:

- traditional **homogenization** and **dehomogenization**: with respect to h we denote them by the superscripts $^{\hom(h)}$ and $^{\deh(h)}$ respectively; and with respect to x_0, by the superscripts $^{\hom(x_0)}$ and $^{\deh(x_0)}$.
- ***d*-shifted-homogenization**: for a non-zero polynomial f and degree $d \geq \deg(f)$ we write $f^{\hom_d(h)}$ to mean $h^{d-\deg(f)} f^{\hom(h)}$, which is a homogeneous polynomial of degree d. As a special case, since $0^{\hom(h)} = 0$, we have $0^{\hom_d(h)} = 0$ for all d.

The following easy properties of the shifted-homogenization will help the reader understand the proof of Theorem 5.3

Lemma 2.6. *Let P be a polynomial ring over the field K, and let $f, g \in P$.*

- *If $d_1 \geq \deg(f)$ and $d_2 \geq \deg(g)$ then $f^{\hom_{d_1}(h)} \cdot g^{\hom_{d_2}(h)} = (fg)^{\hom_{d_1+d_2}(h)}$*
- *If $d \geq \deg(f)$ and $d \geq \deg(g)$ then $f^{\hom_d(h)} + g^{\hom_d(h)} = (f+g)^{\hom_d(h)}$*

Proof. The proofs are elementary exercises in algebra. Observe that the special definition $0^{\hom_d(h)} = 0$ is indeed compatible with this lemma, since from the equality $f - f = 0$ we deduce the equalities $0 = f^{\hom_d(h)} - f^{\hom_d(h)} = 0^{\hom_d(h)}$. □

3. Polynomial parametrizations

In this section we consider a parametrization given by polynomials $f_1, \ldots, f_n$ in the ring $K[t_1, \ldots, t_s]$, where $\{t_1, \ldots, t_s\}$ is a set of indeterminates which are viewed as parameters. We will look at parametrizations by rational functions in section 5.

Proposition 3.1. *In the setting of Definition 2.1:*

(a) We have $\text{Implicit}(f_1, \ldots, f_n) = J \cap P$.
(b) The ideal $\text{Implicit}(f_1, \ldots, f_n)$ *can be computed using an elimination ordering for all the* t_i.
(c) The ideal $\text{Implicit}(f_1, \ldots, f_n)$ *is prime.*

Proof. Claims (a) and (b) are standard results (see for instance book Kreuzer and Robbiano, 2000, Section 3.4). Claim (c) follows from the isomorphism $K[t_1, \ldots, t_s, x_1, \ldots, x_n]/J \cong K[t_1, \ldots t_s]$, whence J is prime, and so $J \cap P$ is prime too. □

Remark 3.2. We shall later find it convenient to assume that in the parametrization no x_i maps to a constant. This is not a restriction because if, say, $f_n \in K$ then we obtain the simple decomposition: $\text{Implicit}(f_1, f_2, \ldots, f_n) = \langle x_n - f_n \rangle + \text{Implicit}(f_1, \ldots, f_{n-1})$. Thus any indeterminates x_i which map to constants can simply be taken out of consideration, letting us concentrate on the interesting part. Henceforth we shall assume that none of the f_i is constant.

The very construction of the eliminating ideal (in Definition 2.1) looks intrinsically non-homogeneous. And it is well-known that the behaviour of Buchberger's algorithm can be quite erratic when the input is not homogeneous: usually the computation for a non-homogeneous input is a lot slower than a "similar" homogeneous computation (though there are sporadic exceptions); for instance, see Example 4.4. We now look quickly at how to use homogenization during implicitization.

A first idea is to give weights to the x_i indeterminates by setting $\deg(x_i) = \deg(f_i)$ for each i. If we do so, and if the original f_i are homogeneous polynomials, then the eliminating ideal J turns out to be a homogeneous ideal. Even when the f_i are not all homogeneous, in the process of ordering and choosing the power-products of a given degree, we may reasonably expect that Buchberger's algorithm will "behave" more similarly to a homogeneous ideal than with the *standard grading*, where all indeterminates have degree 1.

Although this trick improves the computation in most cases, it is not a miraculous panacea. Much better ideas come from the following Proposition 3.3 and Theorem 5.3 which reduce the computation of the implicitization ideal to the case of prime ideals whose generators are homogeneous polynomials.

In the proofs we use the fundamental properties of homogenization and dehomogenization as described in Kreuzer and Robbiano (2005, Section 4.3). A general discussion about the topic treated in the following proposition can be found in Kreuzer and Robbiano (2005, Tutorial 51).

In the proposition below we use a single homogenizing indeterminate h; so, to simplify notation, homogenization and dehomogenization are tacitly taken with respect to h.

Proposition 3.3 *(Implicitizating polynomial parametrizations by homogenization). Let* $P = K[x_1, \ldots, x_n]$, *let* $f_1, \ldots, f_n \in K[t_1, \ldots, t_s] \setminus K$. *Now let* h *be a new indeterminate, and let* $P[t_1, \ldots, t_s, h]$ *be graded by setting* $\deg(x_i) = \deg(f_i)$ *for* $i = 1, \ldots, n$ *and* $\deg(t_1) = \cdots = \deg(t_s) = \deg(h) = 1$. *Finally let* $F_i = f_i^{\text{hom}}$, *and let* $\bar{J}$ *be the eliminating ideal of* $(F_1, \ldots, F_n)$. *Then:*

(a) The ideal $\bar{J} \cap P[h]$ *is prime.*
(b) We have the equality $\text{Implicit}(f_1, \ldots, f_n) = (\bar{J} \cap P[h])^{\text{deh}}$.

Proof. The proof of claim (a) follows immediately from the fact that $\bar{J}$ is an eliminating ideal, hence prime.

Let J be the (non-homogeneous) eliminating ideal of the tuple $(f_1, \ldots, f_n)$. Now, since $\bar{J}$ is prime, it is saturated with respect to h; furthermore we have $\bar{J}^{\text{deh}} = J$, so we can deduce that $\bar{J} = J^{\text{hom}}$. Clearly $(J \cap P)^{\text{hom}} \subseteq J^{\text{hom}} \cap P[h] = \bar{J} \cap P[h]$, hence, by dehomogenizing, we deduce that $J \cap P = ((J \cap P)^{\text{hom}})^{\text{deh}} \subseteq (\bar{J} \cap P[h])^{\text{deh}}$. On the other hand, if $f \in (\bar{J} \cap P[h])^{\text{deh}}$ then we have $f \in \bar{J}^{\text{deh}} \cap P$, but $\bar{J}^{\text{deh}} = (J^{\text{hom}})^{\text{deh}} = J$, and the proof is complete since $\text{Implicit}(f_1, \ldots, f_n) = J \cap P$ by Proposition 3.1.a. □

4. Hypersurfaces parametrized by polynomials

In this section we start to treat the "hypersurface case", namely the case where it is known that the implicitization ideal is principal. In this situation we typically have $s = n-1$, although this equality is not equivalent to the implicitization being a principal ideal, as shown in Example 1.2.

There is no easy way to determine whether the implicitization is going to be a principal ideal, but this information might already be independently known for the particular example under consideration. So this is usually taken as hypothesis by the papers on this topic.

4.1. A truncated homogeneous computation

As already observed, the ideal $\bar{J}$ in Proposition 3.3 is homogeneous, hence the computation of the (elimination) Gröbner basis of $\bar{J}$ can be performed degree by degree. Moreover, using the methods described in Proposition 3.3 we get the following extra bonus in the hypersurface case: as soon as we obtain a Gröbner basis element, G, which does not involve the parameters, we may stop the computation of the Gröbner basis because the solution polynomial is just the dehomogenization of G.

Corollary 4.1. *With the same assumptions as in Proposition 3.3, if* $\text{Implicit}(f_1, \ldots, f_n)$ *is a principal ideal generated by* g *then* $\text{Implicit}(f_1^{\text{hom}}, \ldots, f_n^{\text{hom}})$ *is a principal ideal generated by* g^{hom}.

Proof. We recall the equality $\text{Implicit}(f_1, \ldots, f_n) = (\bar{J} \cap P[h])^{\text{deh}}$ proved in Proposition 3.3.b. This implies that $(\bar{J} \cap P[h])^{\text{deh}} = \langle g \rangle$. Conversely, the ideal $\bar{J} \cap P[h]$ is prime by Proposition 3.3.a, hence it is saturated with respect to h, and so it is generated by g^{hom}. □

Algorithm 4.2 *(ElimTH: truncated homogeneous elimination).*
Input $f_1, \ldots, f_n \in K[t_1, \ldots, t_s] \setminus K$ such that the ideal $\text{Implicit}(f_1, \ldots, f_n)$ is principal.
ElimTH-1 Initialization:
  ElimTH-1.1 Create the polynomial ring $R = K[t_1, \ldots, t_s, h, x_1, \ldots, x_n]$ graded by $[1, \ldots, 1, 1, \deg(f_1), \ldots, \deg(f_n)]$.
  Let σ be an elimination ordering for $\{t_1, \ldots, t_s\}$ on R.
  ElimTH-1.2 Let $F_i = f_i^{\text{hom}(h)} \in R$.
  ElimTH-1.3 Let $J = \langle x_1 - F_1, \ldots, x_n - F_n \rangle$, the eliminating ideal of $(F_1, \ldots, F_n)$.
ElimTH-2 *Main Loop:*
  Start Buchberger's algorithm for the computation of a σ-Gröbner basis GB of J.
  Perform its main loop degree by degree (*i.e.* always choose the lowest degree pair).
  When you add to GB the first polynomial G such that $\text{LT}_\sigma(G)$ is not divisible by any t_i exit from loop.
ElimTH-3 Let $g_{\text{calc}} = G^{\text{deh}(h)}$ mapped into $K[x_1, \ldots, x_n]$.
Output $g_{\text{calc}} \in K[x_1, \ldots, x_n]$.
Then g_{calc} generates $\text{Implicit}(f_1, \ldots, f_n)$.

Proof. *Termination*: The *Main Loop* in the algorithm is simply Buchberger's algorithm, and that terminates in a finite number of steps. Moreover, Corollary 4.1 guarantees that $\bar{J}$ contains a polynomial not involving the t_i indeterminates, and since σ is an elimination ordering for the t_i there is such a polynomial in the Gröbner basis, so the *Main Loop* will set G and exit.

Correctness: In the *Main Loop* we execute Buchberger's Algorithm with respect to an elimination ordering for all the t_i; thus the elements of the Gröbner basis whose leading terms are not divisible by any t_i form a Gröbner basis for the elimination ideal $\bar{J} \cap P[h]$. By Corollary 4.1 this ideal is principal, so Buchberger's Algorithm (computing degree by degree) will produce exactly one polynomial whose leading term is not divisible by any t_i. The *Main Loop* stops as soon as this polynomial is found. In step **ElimTH-3** the polynomial G will be the generator of Implicit$(f_1^{\text{hom}}, \ldots, f_n^{\text{hom}})$, and by Corollary 4.1 we have that $g_{\text{calc}} = G^{\text{deh}(h)}$ is the generator of Implicit$(f_1, \ldots, f_n)$. □

Remark 4.3. We consider briefly what happens if the input to Algorithm 4.2 (ElimTH) does not correspond to a principal implicitization ideal. If Implicit$(f_1, \ldots, f_n)$ is the zero ideal then Buchberger's Algorithm in step **ElimTH-2** will terminate without finding any candidate for G; we could in that case simply set $G = 0$. By Remark 1.1 this cannot happen if $s \leq n - 1$.

If, instead, Implicit$(f_1, \ldots, f_n)$ is non-zero and non-principal then the polynomial G found in step **ElimTH-2** will be a lowest weighted-degree element of a Gröbner basis for that ideal (and consequently a lowest weighted-degree non-zero element of the ideal).

This next example illustrates the good behaviour of the algorithm above.

Example 4.4. We let $K = \mathbb{Z}/(32003)$ and in the ring $K[x_1, x_2, t]$ we consider the eliminating ideal

$$I = \langle x_1 - (t^{15} - 3t^2 - t + 1),\ x_2 - (t^{23} + t^{11} + t^3 - t - 2)\rangle$$

The usual elimination of t takes more than one hour, even if we give the weights 15 and 23 to the indeterminates x_1, x_2 respectively; whereas the truncated homogeneous elimination takes less than a second (this is one of our test cases: see Example A.14). The solution polynomial has 176 power-products in its support.

Remark 4.5. If the eliminating ideal is not homogeneous, the idea of truncating the computation as soon as a polynomial in P is found does not work well, since it may happen that the first such polynomial computed by the algorithm is a proper multiple of the solution polynomial. The phenomenon is similar to the case where the reduced Gröbner basis of an ideal is $\{1\}$, yet before discovering that 1 is in the basis it often happens that many other (non-reduced) Gröbner basis elements are computed.

One could take the polynomial found and factorize it, then substitute into the various irreducible factors to see which factor is the good one. But this is unlikely to be efficient.

4.2. A direct approach

We briefly recall the setting of this paper: we have been given a K-algebra homomorphism $\varphi\colon K[x_1, \ldots, x_n] \longrightarrow K[t_1, \ldots, t_s]$ sending $x_i \mapsto f_i$ and we assume that its kernel is a principal ideal: the problem is to find the generator of $\ker(\varphi) = \text{Implicit}(f_1, \ldots, f_n) = \langle g\rangle$. Following Remark 3.2, we shall find it convenient to assume that each $f_i \in K[t_1, \ldots, t_s]$ is non-constant. In this section we compute the polynomial g via a *direct* approach.

We use the notation $\mathbf{LPP}(\boldsymbol{f})$ to indicate the leading power-product of the polynomial f (also denoted in the literature by $\text{LT}(f)$ of $\text{in}(f)$). If $f = \sum_i a_i T_i$, with distinct power-products T_i, then the **support** of f is $\mathbf{Supp}(\boldsymbol{f}) = \{T_i \mid a_1 \neq 0\}$.

Remark 4.6. First of all notice that, if a polynomial $f = \sum_i a_i T_i \in K[x_1, \ldots, x_n]$ is such that $\varphi(f) = 0$, then $\sum_i a_i \varphi(T_i) = 0$. In other words, there is a K-linear dependency among the image polynomials $\{\varphi(T) \mid T \in \text{Supp}(f)\} \subset K[t_1, \ldots, t_s]$, and the coefficients of the linear relation are exactly the coefficients of f (up to a scalar multiple).

The idea behind our direct approach is to *directly* determine g by searching for a linear dependency among all the $\varphi(T)$: we generate, one by one, the polynomials $\varphi(T)$ as T runs through the power-products in $K[x_1, \ldots, x_n]$ until a dependency exists. We shall now see how to reduce this apparently infinite problem to a finite, tractable one.

Algorithm 4.7 *(Direct: implicitization by direct search).*
Input $f_1, \ldots, f_n \in K[t_1, \ldots, t_s]$ such that the ideal Implicit$(f_1, \ldots, f_n)$ is principal.
Variables The main variables are:
***QB*:** the list of power-products in $K[x_1, \ldots, x_n]$ already considered.
***PPL*:** the list of power-products in $K[x_1, \ldots, x_n]$ yet to be considered.
***PhiQB*:** the list $[\varphi(T_i) \mid T_i \in QB]$; its elements are seen as "sparse vectors" in the infinite dimensional K-vector space $K[t_1, \ldots, t_s]$ with basis comprising all power-products.
Direct-1 *Initialization:*
Direct-1.1 Fix an enumerative term-ordering σ on $K[x_1, \ldots, x_n]$.
Direct-1.2 Set $QB = \emptyset$. Set $PhiQB = \emptyset$. Set $PPL = \{1\}$.
Direct-2 *Main Loop:*
Direct-2.1 Let $T = \min_\sigma(PPL)$. Remove T from *PPL*.
Direct-2.2 Compute $v = \varphi(T) \in K[t_1, \ldots, t_s]$.
Direct-2.3 Is there a linear dependency $v = \sum_i a_i v_i$ with $a_i \in K$ and $v_i \in PhiQB$?
yes exit from loop
no Add to *PPL* the elements of $\{x_1 T, \ldots, x_n T\}$ not already in *PPL*;
append T to the list *QB*; append v to the list *PhiQB*
Direct-3 Let $g_{\text{calc}} = T - \sum_i a_i T_i$ where $T_i \in QB$ corresponds to $v_i \in PhiQB$.
Output $g_{\text{calc}} \in K[x_1, \ldots, x_n]$.
Then g_{calc} generates Implicit$(f_1, \ldots, f_n)$.

Proof. *Termination:* The main loop of the algorithm considers the power-products in the ring $K[x_1, \ldots, x_n]$ in increasing σ-order until the condition in step **Direct-2.3** breaks out; since σ is enumerative, every power-product will be considered at some (finite) time. The initial values for *QB* and *PPL*, and the updates to these two variables in step **Direct-2.3 (no)** guarantee that whenever we enter step **Direct-2.1** the set *PPL* satisfies $PPL = \{x_i T : 1 \le i \le n \text{ and } T \in QB\} \setminus QB$; in other words it comprises those power-products outside *QB* and which border on *QB*. As σ is a term-ordering, *PPL* therefore always contains the σ-smallest power-product outside *QB* (as well as many others).

In step **Direct-2.3** the algorithm looks for a K-linear dependency amongst the polynomials in the set $\{\varphi(\bar{T}) \mid \bar{T} \le_\sigma T\}$. Every such linear dependency corresponds to a monic element of $\ker(\varphi)$. By hypothesis $\ker(\varphi)$ contains the polynomial g (which we may assume to be monic wrt. σ), so if we reach step **Direct-2.3** with $T = \text{LPP}(g)$ then a linear dependency will surely be found (*e.g.* corresponding to the coefficients of g). Since σ is an enumerative ordering, there are only finitely many power-products less than LPP(g); so we will break out of the main loop when $T = \text{LPP}(g)$, if not earlier.

Correctness: We shall show that we do not break out of the main loop until $T = \text{LPP}(g)$, and that when we do break out, the polynomial we construct in step **Direct-3** is g.

The test in step **Direct-2.3** gives *true* if and only if there is a polynomial $\tilde{g}$, of the form $T - \sum_i a_i T_i$ with each $T_i <_\sigma T$, satisfying $\varphi(\tilde{g}) = 0$ or equivalently $\tilde{g} \in \ker(\varphi)$. Note that $\tilde{g}$ is monic, thus non-zero by construction.

By hypothesis $\ker(\varphi)$ is a principal ideal (generated by g). So every non-zero element of $\ker(\varphi)$ has leading term σ-greater-than-or-equal to LPP(g), thus step **Direct-2.3** will not find any linear dependency if $T <_\sigma \text{LPP}(g)$.

Let g_{calc} be the polynomial constructed in step **Direct-3**. We have $\text{LPP}(g_{\text{calc}}) = \text{LPP}(g)$, and both polynomials are monic. Suppose $g_{\text{calc}} \neq g$, and set $\hat{g} = g_{\text{calc}} - g$. Then $\text{LPP}(\hat{g}) <_\sigma \text{LPP}(g)$ but also $\varphi(\hat{g}) = \varphi(g_{\text{calc}}) - \varphi(g) = 0$, so $\hat{g} \in \ker(\varphi)$ which contradicts the fact that g is the (non-zero) element of $\ker(\varphi)$ with σ-smallest leading term. This concludes the proof. □

Remark 4.8. We consider briefly what happens if the input to Algorithm 4.7 (Direct) does not correspond to a principal implicitization ideal.

If Implicit$(f_1, \ldots, f_n)$ is the zero ideal then the *Main Loop* never exits (as no non-trivial linear dependency exists). However, if $s \le n - 1$ then the ideal Implicit$(f_1, \ldots, f_n)$ cannot be the zero ideal as proved in Remark 1.1.

If, instead, Implicit$(f_1, \ldots, f_n)$ is non-zero and non-principal then the main loop will exit, and the polynomial g_{calc} found in step **Direct-3** will be the monic polynomial with σ-smallest leading term in the reduced σ-Gröbner basis for that ideal.

Remark 4.9. This approach is inspired by the *Generalized Buchberger–Möller algorithm* (Abbott et al., 2005), and is somewhat simpler (*e.g.* the list G for storing the Gröbner basis is not needed, and the update to the list *PPL* is simpler). But there is an important difference: here we cannot specify *a priori* a finite dimensional vector space as the codomain of the *normal form vector map*. For the generalized Buchberger–Möller algorithm the finiteness of the codomain led to an easy proof of termination; instead here we had to introduce the concept of enumerative ordering.

Remark 4.10 *(Optimizations).* We mention here a few important optimizations which considerably improve the execution time:

(a) The successive linear systems we check in step **Direct-2.3** are very similar: in practice we build up a row-reduced matrix adjoining a new row on each iteration.
(b) The computation of $\varphi(T)$ in step **Direct-2.2** can be effected in several ways. We suggest exploiting the fact that φ is a homomorphism to compute the value cheaply. Apart from the very first iteration when $T = 1$, we always have $T = x_j\, T'$ for some indeterminate x_j and some power-product T' for which we have already computed $\varphi(T')$; so we can calculate with just a single multiplication $\varphi(T) = \varphi(x_j)\,\varphi(T')$. Usually there are several choices for the indeterminate x_j, so we can choose the one which leads to the cheapest multiplication. Note that in step **Direct-2.3(no)** we manipulate just power-products when updating *PPL*.
(c) In step **Direct-1.1** we pick some enumerative ordering on the power-products of the ring $K[x_1, \ldots, x_n]$. Here we describe a specific good choice; the idea is that as we pick (in step **Direct-2.1**) the power-products T in increasing order then the corresponding LPP$(\varphi(T))$ are in non-decreasing order.
We start with a (standard) degree-compatible term-ordering τ on the power-products of $K[t_1, \ldots, t_s]$. Let M_τ be an $s \times s$ integer matrix representing it (so all entries in the first row are 1). We define the **order vector** of a power-product $t_1^{e_1} t_2^{e_2} \cdots t_s^{e_s}$ to be $M_\tau \underline{e}$; the ordering τ is thus equivalent to lex comparison of the order vectors.
Let E be the $s \times n$ integer matrix whose columns are the exponents of $\text{LPP}_\tau(f_i)$; put $M = M_\tau \cdot E$, an $s \times n$ matrix whose i-th column is the order vector of $\text{LPP}_\tau(\varphi(x_i))$. The first row of M is strictly positive: the i-th entry is $\deg(f_i)$. We complete M to a term-ordering matrix M' for the power-products of $K[x_1, \ldots, x_n]$: *i.e.* we remove rows linearly dependent on those above it, and adjoin new rows at the bottom to make M' square and invertible. The term-ordering defined by M' is enumerative since M and M' have the same first row, and it has our desired property.

Example 4.11. An example to illustrate Remark 4.10(b). Let $K[s, t]$ have terms ordered by $\left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right)$. Let $f_1 = s^5 - st^3 - t$, $f_2 = st^2 - s$, $f_3 = s^4 - t^2$ then the order vectors of the LPPs are $(5, 5)$, $(3, 1)$ and $(4, 0)$ respectively. So we obtain the matrix $M' = \left(\begin{smallmatrix} 5 & 3 & 4 \\ 5 & 1 & 0 \\ * & * & * \end{smallmatrix}\right)$ where we can fill the last row freely to make the matrix invertible, *e.g.* (0 0 1).

Remark 4.12. We contrast Algorithm 4.7 (Direct) with the method presented by Wang (2004). The underlying idea is the same: find the generator by searching for a linear relationship among the images of power-products. Wang's method adjoins new power-products in blocks. Each block comprises all power-products of a given standard degree (where each indeterminate has degree 1). Wang observed that the linear systems produced "tend to be almost triangular".

Our approach adjoins new power-products one at a time, and this lets us use several important optimizations (described in Remark 4.10). On each iteration we do a single multiplication to obtain $\varphi(T)$, and then a single "row reduction". Using the term-ordering described in Remark 4.10(c) guar-

antees that our linear system is as triangular as possible, and by adjoining power-products one by one we keep the system small (and avoid computing extraneous images of power-products under φ).

The importance of these optimizations is illustrated by the computation time for Example A.7: our implementation of Algorithm 4.7 (Direct) took less than 8 seconds, while Wang reported about 47000 seconds — no doubt some (but not all) of the speed gain is due to improvements in hardware.

Remark 4.13. The requirement that σ be a term-ordering is stronger than necessary. For instance, it is sufficient that σ orders by degree (how it orders within a fixed degree is unimportant). However, if we use such a general ordering then the list *PPL* will then have be to updated differently in step **Direct-2.3 (no)** (*e.g.* fill it with all power-products of the next degree when it becomes empty, much like Wang's method (Wang, 2004)).

5. Hypersurfaces parametrized by rational functions

In this section we consider a parametrization given by rational functions $f_1, \dots, f_n$ in the field $L = K(t_1, \dots, t_s)$, where $\{t_1, \dots, t_s\}$ is a set of indeterminates which are viewed as parameters. We can write this parametrization with a common denominator q, so that we have $f_i = \frac{p_i}{q}$ with $q, p_1, \dots, p_n \in K[t_1, \dots, t_s]$.

Remark 5.1. We recall here a general method for computing Implicit$(\frac{p_1}{q}, \dots, \frac{p_n}{q})$. We start with the ideal $I = \langle qx_1 - p_1, \dots, qx_n - p_n \rangle \subset K[x_1, \dots, x_n, t_1, \dots, t_s]$, then we introduce a new indeterminate u, and let $J = I + \langle uq - 1 \rangle \subset K[x_1, \dots, x_n, t_1, \dots, t_s, u]$. Now we have Implicit$(\frac{p_1}{q}, \dots, \frac{p_n}{q}) = J \cap P$ which can be computed by eliminating the indeterminates u and $t_1, \dots, t_s$. The following example illustrates the necessity of adding $\langle uq - 1 \rangle$ to I.

Example 5.2. We let $K = \mathbb{Q}$ and we let $f_1 = f_2 = \frac{s}{t}$, $f_3 = s$ in $K[s, t]$. We construct the ideal $I = \langle tx - s,\ ty - s,\ z - s \rangle$, and we get $I \cap K[x, y, z] = \langle z(x - y) \rangle$ which is not prime. The correct result, which may be obtained by including the generator $ut - 1$, is Implicit$(f_1, f_2, f_3) = \langle x - y \rangle$.

In the following we present a "homogeneous" method for the computation of the ideal Implicit$(f_1, \dots, f_n)$ when $f_1, \dots, f_n$ are rational functions; compared to the classical method the big advantage we have is that the Gröbner basis computation (*i.e.* the elimination) is performed on a homogeneous ideal.

Theorem 5.3 *(Implicitizating rational parametrizations by homogenization). Let $q, p_1, \dots, p_n$ be non-zero polynomials in $K[t_1, \dots, t_s]$. Let $R = K[x_0, x_1, \dots, x_n, t_1, \dots, t_s, h]$ be graded by $\deg(h) = \deg(t_i) = 1$ for $i = 1, \dots, s$ and $\deg(x_i) = d$ for $i = 0, \dots, n$ where $d = \max\{\deg(q), \deg(p_1), \dots, \deg(p_n)\}$.*

To make the input polynomials homogeneous and of equal degree d we set $Q = q^{\hom_d(h)}$, and $P_i = p_i^{\hom_d(h)}$ for $i = 1, \dots, n$. Let $\bar{J}$ be the homogeneous eliminating ideal generated by $\{x_0 - Q,\ x_1 - P_1, \dots, x_n - P_n\}$ in the ring R.

(a) The ideal $\bar{J} \cap K[x_0, x_1, \dots, x_n]$ is prime.
(b) We have the equality Implicit$(\frac{p_1}{q}, \dots, \frac{p_n}{q}) = \big(\bar{J} \cap K[x_0, x_1, \dots, x_n]\big)^{\deh(x_0)}$.

Proof. The kernel Implicit$(\frac{p_1}{q}, \dots, \frac{p_n}{q})$ is an ideal in the ring $P = K[x_1, \dots, x_n]$, which we view as a subring of R. We write $P[x_0]$ to denote the subring $K[x_0, x_1, \dots, x_n]$: observe that a polynomial in $P[x_0]$ is homogeneous in the induced grading if and only if it is homogeneous in the standard grading (*i.e.* in the usual sense of the word).

As in Proposition 3.3, the proof of claim (a) follows immediately from the fact that $\bar{J}$ is an eliminating ideal, hence prime.

To prove claim (b) we introduce the following sets:

- $S_1 = \{A \in P \mid A(\frac{p_1}{q}, \dots, \frac{p_n}{q}) = 0\}$;
- $S_2 = \{A \in P[x_0] \mid A \text{ is homogeneous and } A(Q, P_1, \dots, P_n) = 0\}$;
- $S_3 = \{A \in P[x_0] \mid A \text{ is homogeneous and } A(q, p_1, \dots, p_n) = 0\}$;
- $S_4 = \{A \in P[x_0] \mid A \text{ is homogeneous and } A^{\deh(x_0)}(\frac{p_1}{q}, \dots, \frac{p_n}{q}) = 0\}$.

Clearly, the conclusion is reached if we prove the following claims.

(1) The set S_1 is the ideal Implicit$(f_1, \dots, f_n)$.
(2) The set S_2 generates the ideal $\bar{J} \cap P[x_0]$.
(3) We have $S_3 = S_2$.
(4) We have $S_4 = S_3$.
(5) We have $S_1 = \{A^{\deh(x_0)} \mid A \in S_4\}$.

Claim (1) is just the definition of Implicit.

To prove claim (2) we recall that the ideal $\bar{J}$ is homogeneous, hence $\bar{J} \cap P[x_0]$ is too. Thus $\bar{J} \cap P[x_0]$ can be generated by homogeneous elements, and S_2 contains all homogeneous elements in $\bar{J} \cap P[x_0]$, and so it surely generates the ideal.

We now prove claim (3). Let $A(x_0, x_1, \dots, x_n)$ be a homogeneous polynomial, and let $D = \deg(A)$. Repeated application of Lemma 2.6 on the monomials in A shows that

$$A(Q, P_1, \dots, P_n) = A(q^{\hom_d(h)}, p_1^{\hom_d(h)}, \dots, p_n^{\hom_d(h)}) = (A(q, p_1, \dots, p_n))^{\hom_{dD}(h)}$$

whence $A(Q, P_1, \dots, P_n) = 0$ if and only if $A(q, p_1, \dots, p_n) = 0$.

Finally, we prove claim (4). We have

$$A = x_0^{\deg(A)} A\big(1, \tfrac{x_1}{x_0}, \dots, \tfrac{x_n}{x_0}\big) = x_0^{\deg(A)} A^{\deh(x_0)}\big(\tfrac{x_1}{x_0}, \dots, \tfrac{x_n}{x_0}\big)$$

consequently $A(q, p_1, \dots, p_n) = q^{\deg(A)} A^{\deh(x_0)}(\frac{p_1}{q}, \dots, \frac{p_n}{q})$, hence the claimed equality follows. Since claim (5) is clear, the proof is complete. □

Corollary 5.4. *If* Implicit$(\frac{p_1}{q}, \dots, \frac{p_n}{q}) = \langle g \rangle$ *then* $\bar{J} \cap K[x_0, x_1, \dots, x_n] = \langle g^{\hom} \rangle$.

Proof. The proof can be done exactly as the proof of Corollary 4.1.a. □

Remark 5.5. We may relax the restriction in the theorem that each p_i be non-zero; it is there just to allow an easy definition of d, the upper bound for the degrees.

Remark 5.6. We could be tempted to use the general method highlighted in Remark 5.1: namely, we homogenize the generators of J given there to get the eliminating ideal $\bar{J}^\dagger$, and then imitate Proposition 3.3. However, even if the ideal $(\bar{J}^\dagger \cap P[h])^{\deh}$ is principal, the ideal $\bar{J}^\dagger \cap P[h]$ need not be principal, as the following example shows. The main drawback is that $\bar{J}^\dagger$ need not be saturated with respect to h.

Example 5.7 *(Ex. 5.2 continued).* We return to Example 5.2, but this time homogenize the generators of $J = I + \langle ut - 1 \rangle$ to produce the following ideal $\bar{J}^\dagger = \langle tx - hs,\ ty - hs,\ z - s,\ ut - h^2 \rangle$. However, elimination yields $\bar{J}^\dagger \cap K[x, y, z, h] = \langle xzh - yzh,\ xh^2 - yh^2 \rangle$ which is not principal. Even if we homogenize the generators and bring them all to the same degree, we get the ideal $\bar{J}^\ddagger = \langle tx - hs,\ ty - hs,\ h(z - s),\ ut - h^2 \rangle$, and again elimination produces $\bar{J}^\ddagger \cap K[x, y, z, h] = \langle xzh - yzh,\ xh^2 - yh^2 \rangle$.

Remark 5.8. In the case of rational functions, we could also be tempted to homogenize the input in a similar way to Theorem 5.3 but applying just $^{\hom(h)}$ instead of equalizing the degrees with $^{\hom_d(h)}$. This does not work because claim (4) of the proof fails, as the following easy example shows.

Example 5.9. In the ring $K(t_1,t_2)[x_1,x_2,x_3]$ we consider the eliminating ideal

$$I = \langle x_1 - \tfrac{t_2^2}{t_1},\ x_2 - \tfrac{t_2^3}{t_1},\ x_3 - \tfrac{t_2^4}{t_1}\rangle$$

The correct answer is Implicit$(\cdots)=\langle x_1x_3-x_2^2\rangle$. However, if we consider the ring $K[x_0,x_1,x_2,x_3,t_1,t_2]$ graded by setting deg(x_0)=1, deg(x_1)=2, deg(x_2)=3, deg(x_3)=4, and deg$(t_1)=$ deg$(t_2)=1$, then the ideal $\bar{I} = \langle x_0 - t_1,\ x_1 - t_2^2,\ x_2 - t_2^3,\ x_3 - t_2^4\rangle$ is homogeneous, but the polynomial of minimal degree in $K[x_0,x_1,x_2,x_3]$ is $x_3 - x_1^2$ whose degree is 4 while the actual solution is the polynomial $x_1x_3 - x_2^2$ whose degree is 6.

We now turn Theorem 5.3 into an explicit algorithm:

Algorithm 5.10 *(RatPar: rational parametrization).*
Input $f_1 = \frac{p_1}{q}, \ldots, f_n = \frac{p_n}{q} \in K(t_1,\ldots,t_s)$ where q is a common denominator.
RatPar-1 Let $d = \max\{\deg(q), \deg(p_1), \ldots, \deg(p_n)\}$, taking deg$(0)=0$ if necessary.
RatPar-2 Create the rings $R_t = K[t_1,\ldots,t_s,\ h]$ graded by $[1,\ldots,1,1]$, and $R_x = K[x_0,\ x_1,\ldots,x_n]$.
RatPar-3 Let $Q = q^{\hom_d(h)} \in R_t$, and let $P_i = p_i^{\hom_d(h)} \in R_t$ for $i = 1,\ldots,n$.
RatPar-4 Compute $\langle G_1,\ldots,G_m\rangle = \text{Implicit}(Q, P_1,\ldots,P_n)$.
RatPar-5 Compute $g_i = G_i{}^{\text{deh}(x_0)}$ for all $i = 1,\ldots,m$.
Output $(g_1,\ldots,g_m) \subseteq K[x_1,\ldots,x_n]$ – satisfying $\langle g_1,\ldots,g_m\rangle = \text{Implicit}(f_1,\ldots,f_n)$.

Remark 5.11. In step **RatPar-4** we may use any algorithm to compute the implicitization from the (homogeneous) polynomial parametrization, *e.g.* Algorithm 4.2 or 4.7.

6. Modular approach for rational coefficients

It is well known that computations with coefficients in $\mathbb{Q}$ can be very costly in terms of both time and space. When possible, it is generally a good idea to perform the computation modulo one or more primes, and then "lift" the coefficients of these modular results to coefficients in $\mathbb{Q}$. There are two general classes of method: Hensel Lifting and Chinese Remaindering. We shall use Chinese Remaindering.

The modular approach has been successfully used in numerous contexts: polynomial factorization (Zassenhaus, 1969), determinant of integer matrices (Abbott et al., 1999), ideals of points (Abbott et al., 2000), and so on. In any specific application there are two important aspects which must be addressed before a modular approach can be adopted:

- knowing how many different primes to consider to guarantee the result (*i.e.* find a realistic bound for the size of coefficients in the answer);
- handling *bad primes*: *i.e.* those whose related computation follows a different route, yielding an answer with the wrong "shape" (*i.e.* which is not simply the modular reduction of the correct non-modular result).

There is no universal technique for addressing these issues. For our particular application there is no useful coefficient bound, and only a partial criterion for detecting bad primes (see Remark 6.8). We shall use *fault-tolerant rational recovery* to overcome our limited knowledge about these two aspects (see section 6.3).

Definition 6.1 *(Reduction modulo p).* Given a prime number p we denote the usual "reduction mod p" ring homomorphism by $\psi_p : \mathbb{Z} \longrightarrow \mathbb{Z}/\langle p\rangle$. We can extend ψ_p naturally to $\mathbb{Z}[t_1,\ldots,t_s]$ by mapping the coefficients but preserving the power-products, and extend it further to rational functions (over $\mathbb{Q}$) by localizing away from its kernel in $\mathbb{Z}[t_1,\ldots,t_s]$.

Our aim is to reconstruct the monic generator of Implicit$(f_1,\ldots,f_n)$ in $\mathbb{Q}[x_1,\ldots,x_n]$ from the modular implicitizations Implicit$\big(\psi_p(f_1),\ldots,\psi_p(f_n)\big)$ in $\mathbb{Z}/\langle p\rangle[x_1,\ldots,x_n]$.

6.1. Bad primes

Let $f_1, \dots, f_n \in \mathbb{Q}(t_1, \dots, t_s)$ be non-constant and such that $\text{Implicit}(f_1, \dots, f_n) = \langle g \rangle$ is a principal ideal, for some $g \in \mathbb{Q}[x_1, \dots, x_n]$. Clearly the generator g is defined only up to a non-zero scalar multiple; we resolve this ambiguity by requiring g to be monic (with respect to some fixed term-ordering on $\mathbb{Q}[x_1, \dots, x_n]$). We can now define $\text{den}(g) \in \mathbb{Z}$ to be the least common denominator of the coefficients of g.

Definition 6.2. We say that the prime p is **unsuitable** if any of the following happens:

(a) there is an index i such that f_i is not in the domain of ψ_p;
(b) there is an index i such that $\psi_p(f_i) = 0$ or
$\deg(\psi_p(\text{numer}(f_i))) < \deg(\text{numer}(f_i))$ or
$\deg(\psi_p(\text{denom}(f_i))) < \deg(\text{denom}(f_i))$.

In other words p is unsuitable if it divides any denominator, or if the degrees of numerator and denominator of some f_i change modulo p. It is easy to check whether p is unsuitable.

We exclude all unsuitable primes from subsequent discussions.

Definition 6.3. We say that the prime p is **bad** if it is suitable but either of the following happens:

(A) g is not in the domain of ψ_p, that is p divides a denominator in g.
(B) $\text{Implicit}(\psi_p(f_1), \dots, \psi_p(f_n)) \neq \langle \psi_p(g) \rangle$.

We say that a prime is **good** if it is neither unsuitable nor bad. We say that p is **very-good** if it is good and $\text{Supp}(g) = \text{Supp}(\psi_p(g))$; in other words, it does not divide the numerator of any coefficient in g.

Example 6.4 *(Bad primes).* Given $f_1 = t_1^3,\ f_2 = t_2^3,\ f_3 = t_1 + t_2 \in \mathbb{Q}[t_1, t_2]$ we have

$$\text{Implicit}(\cdots) = \langle -x_3^9 + 3x_1x_3^6 + 3x_2x_3^6 - 3x_1^2x_3^3 + 21x_1x_2x_3^3 - 3x_2^2x_3^3 + x_1^3 + 3x_1^2x_2 + 3x_1x_2^2 + x_2^3 \rangle$$

but modulo 3 we obtain

$$\text{Implicit}(\psi_3(f_1),\ \psi_3(f_2),\ \psi_3(f_3)) = \langle x_3^3 - x_1 - x_2 \rangle \quad \subseteq \quad \mathbb{Z}/\langle 3 \rangle[x_1, x_2, x_3]$$

So the prime 3 is bad because, even though the modular implicitization is principal, it is not equivalent modulo 3 to the correct result.

Indeed, even when $\text{Implicit}(f_1, \dots, f_n)$ is principal in $\mathbb{Q}[x_1, \dots, x_n]$ we cannot be sure that $\text{Implicit}(\psi_p(f_1), \dots, \psi_p(f_n))$ is principal too. For instance, given the parametrization $f_1 = t_1 + t_2$, $f_2 = t_1 - t_2$ and $f_3 = t_1 - t_2 \in \mathbb{Q}[t_1, t_2]$ we have

$$\text{Implicit}(f_1, f_2, f_3) = \langle x_2 - x_3 \rangle$$

whereas modulo 2 we find that

$$\text{Implicit}(\psi_2(f_1),\ \psi_2(f_2),\ \psi_2(f_3)) = \langle x_1 - x_3,\ x_2 - x_3 \rangle \quad \subseteq \quad \mathbb{Z}/\langle 2 \rangle[x_1, x_2, x_3]$$

From Remarks 4.3 and 4.8, we see that in cases such as this, where the modular inputs do not satisfy the assumption that $\text{Implicit}(\cdots)$ be principal, our Algorithms 4.2 (ElimTH) and 4.7 (Direct) for computing $\text{Implicit}(\psi_p(f_1), \dots, \psi_p(f_n))$ will simply return the first polynomial in the ideal that they find.

Remark 6.5 *(Finitely many unsuitable primes).* Condition (a) is satisfied if and only if p divides the least common denominator for all the f_i; clearly there are only finitely many such primes. Condition (b) is satisfied if and only if p divides the least common multiple of the integer contents of the leading forms of the numerator and denominator of each f_i; again, clearly there are only finitely many such primes.

Remark 6.6 *(Finitely many bad primes).* Clearly condition (A) covers only finitely many primes. For condition (B) we consider what happens when Algorithm 4.7 (Direct) runs. We have a faithful modular implicitization if and only if the check for a linear dependency in step **Direct-2.3** actually finds one on the same iteration that it would have been found while computing over $\mathbb{Q}$. This will happen only if there was no linear dependency in any previous iteration; in other words, if the matrix had been of full rank in the penultimate iteration; and this happens for all primes except those which divide the numerators of all maximal minors — there are clearly only finitely many such primes.

Remark 6.7. Only finitely many primes are good but not very-good. By definition a prime is good but not very-good if it divides the numerator of some coefficient of g, or equivalently if it divides the least common multiple of the numerators of the coefficients of g. Clearly only finitely many primes do so. In conclusion, only finitely many primes are not very-good.

Remark 6.8 *(Detecting bad primes).* We do not have an absolute means of detecting bad primes, but given the implicitizations modulo two different primes we can sometimes detect that one of them is surely bad (without being certain that the other is good). What we can say depends on which algorithm we used to compute the implicitizations — we must use the same algorithm for both modular computations!

If we run Algorithm 4.2 (ElimTH) with a bad prime p to produce the output g_p then we know that $\deg(g_p) \leq \deg(g)$. Thus if we run Algorithm 4.2 with two different primes p_1 and p_2, and if $\deg(g_{p_1}) < \deg(g_{p_2})$ then surely p_1 is a bad prime. Note that even if $\deg(g_p) = \deg(g)$, we need not have $g_p = \psi_p(g)$ as shown by the non-principal ideal in Example 6.4 above.

If we run Algorithm 4.7 (Direct) with a bad prime p to produce the output g_p then we know that $\mathrm{LPP}(g_p) <_\sigma \mathrm{LPP}(g)$ provided we use the same, fixed enumerative term-ordering σ. Thus if we run Algorithm 4.7 with two different primes p_1 and p_2, and if $\mathrm{LPP}(g_{p_1}) <_\sigma \mathrm{LPP}(g_{p_2})$ then surely p_1 is a bad prime.

6.2. Single prime method

Given input $f_1, \ldots, f_n$ we can pick a suitable prime p, and run one of our algorithms to get an output g_p. If p is very-good then $\mathrm{Supp}(g_p) = \mathrm{Supp}(g)$. We can then determine the coefficients of monic(g) by solving a linear system over $\mathbb{Q}$.

Let $N = |\mathrm{Supp}(g_p)|$ and pick N distinct s-tuples of random integers; evaluating all the f_i at each such s-tuple produces a "random point" on the hypersurface, *i.e.* a zero of g. If the $N \times N$ matrix whose (i, j)-entry is the value of the i-th power-product (in $\mathrm{Supp}(g_p)$) at the j-th tuple is of full rank then knowing that every point on the hypersurface is a zero of g, and knowing that the leading coefficient of monic(g) is 1 we can solve the linear system to get all coefficients of g_{calc}, our "informed guess" for the value of monic(g).

We must now verify that g_{calc} is correct; we do this by simply substituting $f_1, \ldots, f_n$ into it. If our choice of prime was very-good then the substitution will verify that g_{calc} is correct. Conversely, if our choice of prime p was not very-good then the candidate "informed guess" for the support of monic(g) was wrong, and g_{calc} will lie outside Implicit$(f_1, \ldots, f_n)$, so the substitution will give a non-zero result; in this case we must start again with a different prime, hoping that this time it will be very-good.

This technique is advantageous when the implicitization is especially sparse (since then the linear system will be small).

Remark 6.9. We can make a cheaper initial verification by picking another random point on the hypersurface, and verifying that that point is a zero of g_{calc}. Naturally, if this "randomized check" passes then a full verification must still be done.

6.3. Multiple prime method

A disadvantage of the single prime method is that if the prime chosen is not very-good then we discover this only at the end of a potentially expensive verification. We can greatly reduce the risk of a failed verification by using several different primes, and combining the corresponding modular answers using Chinese Remaindering. Our strategy must handle bad primes. Using the checks in Remark 6.8 we can detect and discard some bad primes, however it is possible that a few bad primes pass undetected. We use fault-tolerant rational reconstruction to cope with any undetected bad primes; we will find the right answer so long as the good primes sufficiently outnumber the undetected bad ones.

Moreover, when using several primes we do not require that any of the primes be very-good; it is enough for most of the primes to be good and "complementary" (*i.e.* the union of the supports of the answers from all the good primes tried must include the support of the true answer).

The key ingredient in this approach is a *fault-tolerant rational reconstruction* procedure (*e.g.* see Abbott, 2017 and Böhm et al., 2015): this enables rational coefficients to be reconstructed from their modular images even if some of those images are bad. The reconstruction procedure normally returns either the correct rational or an indication of failure, though there is a low probability of it producing an incorrect rational. So for certainty, the reconstructed implicit polynomial must be verified.

We chose the HRR algorithm from Abbott (2017) because it is better suited to our application: compared to ETL from Böhm et al. (2015) it requires fewer primes (and therefore fewer costly modular implicitizations) when reconstructing "unbalanced" rationals, *i.e.* whose numerator and denominator have differing sizes.

Algorithm 6.10 *(ModImplicit).*
Input $f_1, \ldots, f_n \in \mathbb{Q}(t_1, \ldots, t_s)$ such that Implicit$(f_1, \ldots, f_n)$ is principal.
ModImplicit-1 Fix a term-ordering σ on the power-product monoid of $\mathbb{Q}[x_1, \ldots, x_n]$;
choose an enumerative ordering if using Algorithm 4.7 (Direct) in steps 3 and 5.2.
ModImplicit-2 Choose a suitable prime p — see Definition 6.2.
ModImplicit-3 Compute g_p, the monic generator of Implicit$(\psi_p(f_1), \ldots, \psi_p(f_n))$.
ModImplicit-4 Let $g_{\text{crt}} = g_p$ and $\pi = p$.
ModImplicit-5 *Main Loop:*
ModImplicit-5.1 Choose a new suitable prime p so all f_i lie in the domain of ψ_p.
ModImplicit-5.2 Compute the monic generator g_p of Implicit$(\psi_p(f_1), ..., \psi_p(f_n))$.
ModImplicit-5.3 Let $\tilde{\pi} = \pi \cdot p$, and $\tilde{g}_{\text{crt}}$ be the polynomial whose coefficients are obtained by Chinese Remainder Theorem from the coefficients of g_{crt} and g_p.
ModImplicit-5.4 Compute the polynomial $g_{\text{calc}} \in \mathbb{Q}[x_1, \ldots, x_n]$ whose coefficients are obtained as the fault-tolerant rational reconstructions of the coefficients of $\tilde{g}_{\text{crt}}$ modulo $\tilde{\pi}$.
ModImplicit-5.5 Were all coefficients "reliably" reconstructed?
yes if $g_{\text{calc}} \neq 0$ and $g_{\text{calc}}(f_1, \ldots, f_n) = 0$ exit from loop
no Let $g_{\text{crt}} = \tilde{g}_{\text{crt}}$ and $\pi = \tilde{\pi}$
Output $g_{\text{calc}} \in \mathbb{Q}[x_1, \ldots, x_n]$ which generates Implicit$(f_1, \ldots, f_n)$.

Proof. *Correctness:* Let $g \in \mathbb{Q}[x_1, \ldots, x_n]$ be the monic generator of Implicit$(f_1, \ldots, f_n)$.

From the test in step **ModImplicit-5.5** we have that $g_{\text{calc}}(f_1, \ldots, f_n) = 0$, so the value returned is surely an element of Implicit$(f_1, \ldots, f_n)$; consequently, g_{calc} is a non-zero multiple of g.

We show by contradiction that g_{calc} is a scalar multiple of g. Suppose not, then $g_{\text{calc}} = f\, g$ for some non-constant polynomial f. Let σ denote the enumerative term-ordering used inside Algorithm 4.7 (Direct); and let $\deg^*$ denote the weighted degree used inside Algorithm 4.2 (ElimTH) — note that condition (b) in our definition of "unsuitable" makes sure that the same weighted degree is used every time.

Let $T_1 = \text{LPP}_\sigma(g_{\text{calc}})$, then clearly $T_1 >_\sigma \text{LPP}_\sigma(g)$. Let T_2 be a term of g_{calc} of maximal weighted degree; then $\deg^*(T_2) > \deg^*(g)$. Note that T_1 and T_2 could be the same term. Since step **ModImplicit-5.4** succeeded in reconstructing g_{calc} more than half the modular implicitizations

had non-zero coefficients for the term T_1, and similarly for the term T_2. So at least one modular implicitization, g_p, had non-zero coefficients for both T_1 and T_2, but this g_p cannot have been produced by Algorithm 4.7 (Direct) because it has $\mathrm{LPP}_\sigma(g_p) \geq_\sigma T_1 >_\sigma \mathrm{LPP}_\sigma(g)$, and it cannot have been produced by Algorithm 4.2 (ElimTH) because $\deg^*(g_p) \geq \deg^*(T_2) > \deg^*(g)$. Thus g_{calc} is just a scalar multiple of g.

Termination: The HRR algorithm in Abbott (2017) for fault-tolerant rational reconstruction guarantees to produce the correct output when the product of the good primes is sufficiently greater than the square of the product of the bad primes (see Corollary 3.2 in that article).

As there are only finitely many bad primes (see Remark 6.6), the product of the good primes chosen in the *Main Loop* will eventually become arbitrarily large compared to the square of the product of all bad primes (which is an upper bound for the square of the product of all bad primes encountered in the *Main Loop*). Thus the reconstruction in step **ModImplicit-5.4** will eventually produce $g_{\mathrm{calc}} = g$. □

Remark 6.11. We can use the comments in Remark 6.8 to discard some bad primes. If we always use Algorithm 4.7 (Direct) to compute g_p then we may insert the following step:

ModImplicit-5.2a If $\mathrm{LPP}(g_p) <_\sigma \mathrm{LPP}(g_{\mathrm{crt}})$ then go to step **5.1**.
If $\mathrm{LPP}(g_{\mathrm{crt}}) <_\sigma \mathrm{LPP}(g_p)$ then set $g_{\mathrm{crt}} = g_p$ and $\pi = p$; go to step **5.1**.

If we always use Algorithm 4.2 (ElimTH) to compute g_p then we may insert the following step:

ModImplicit-5.2a If $\deg(g_p) < \deg(g_{\mathrm{crt}})$ then go to step **5.1**.
If $\deg(g_{\mathrm{crt}}) < \deg(g_p)$ then set $g_{\mathrm{crt}} = g_p$ and $\pi = p$; go to step **5.1**.

Remark 6.12. Since each g_p is defined only up to a scalar multiple, we normalize the polynomial by making it monic; this guarantees that for every good prime p, the corresponding polynomial g_p is equal to $\psi_p(g)$.

7. Timings

In this section we show the practical merits of our algorithms. We conducted two series of experiments, which we report in the two tables below.

The experiments were performed on a MacBook Pro 2.9GHz Intel Core i7, using our implementation in CoCoA 5. The columns headed "ElimTH" and "Direct" report the computation times for the respective algorithms: in each case there are separate columns for computations over a finite field (char 32003), and over the rationals (char 0). The column headed "Len" says how many terms there are in the resulting polynomial. The symbol ∞ in the tables means that the computation was interrupted after 20 minutes, and 0 means that the computation takes less than 0.001 seconds. A horizontal line in the middle of the tables separates examples with polynomial parametrizations from examples with rational parametrization.

7.1. Examples from the literature

Table 1 contains statistics related to examples taken from the literature, which we list in Appendix A.1. It shows that, with the sole exception of Example A.7, they are computed in almost no time.

We found only two examples which defeated us: listed in our Appendix as Examples A.9 and A.10 — originally they were Examples 5.2 and 5.3 in Botbol and Dickenstein (2016). We suspect they are essentially incalculable because the implicitizations are almost certainly polynomials of high degree (over 100) having very many terms (over 100000).

Table 1
Examples from the literature (Appendix A.1).

Examples	ElimTH		Direct		Len
	32003	0	32003	0	
Ex. A.1	0	0.009	0	0.003	6
Ex. A.2	0	0.007	0	0.002	9
Ex. A.3	0	0.028	0	0.026	57
Ex. A.4	0.273	0.597	0.031	0.118	319
Ex. A.5	0	0.021	0	0.070	13
Ex. A.6	0	0.228	0	0.083	56
Ex. A.7	1.196	16.278	0.159	7.707	715
Ex. A.8	0	0.060	0	0.032	41
Ex. A.11	0	0.943	0	0.934	161
Ex. A.12	0	0.011	0	0.004	7
Ex. A.13	0	0.012	0	0.010	7

Table 2
More challenging examples (Appendix A.2).

Examples	ElimTH			Direct			Len
	32003	0		32003	0		
Ex. A.14	0.1	(5)	0.9	0	(5)	0.3	176
Ex. A.15	2.1	(3)	6.9	0.1	(3)	0.4	471
Ex. A.16	∞		∞	8.41	(5)	58.2	6398
Ex. A.17	20.3	(5)	55.7	0.9	(5)	3.4	1705
Ex. A.18	∞		∞	58.4	(3)	204.1	4304
Ex. A.19	1.4	(3)	4.8	9.1	(3)	27.9	1763
Ex. A.20	60.8		∞	228.0		∞	9360
Ex. A.21	2.2	(3)	9.3	47.3	(3)	148.9	5801
Ex. A.22	5.0	(6)	71.5	∞		∞	6701
Ex. A.23	10.2	(11)	121.0	36.4	(11)	418.5	2356
Ex. A.24	0.1	(4)	1.370	0.1	(4)	1.2	62
Ex. A.25	0.6	(2)	2.8	1.1	(2)	3.8	57
Ex. A.26	0.6	(3)	13.4	2.1	(3)	17.9	115
Ex. A.27	10.4	(3)	159.1	64.8	(3)	335.0	189
Ex. A.28	63.3	(2)	141.7	46.2	(2)	101.6	149
Ex. A.29	116.4	(6)	761.4	202.7	(6)	1214.4	2692

7.2. Our own examples

Table 2 contains statistics related to our own examples, which we list in Appendix A.2. The small numbers in brackets in the columns for characteristic 0 are the number of moduli used in Algorithm 6.10 (ModImplicit) for reconstructing the rational coefficients. The time to compute the answer is essentially the product of the number of moduli and the time for a single finite field; the rest of the time is for verification, which can represent more than half the total time as in Example A.27.

Appendix A. Implicitization examples

In this appendix we list the test examples we used. The symbol K is used to denote either the field $\mathbb{F}_{32003}$ or the field $\mathbb{Q}$. The examples are of different types: in the first subsection there are examples taken from the literature; in the second there are our own examples.

A.1. Examples from the literature

Here we collect examples taken from some papers mentioned in the references.

Example A.1. (d'Andrea, 2015, Example 3.4) In the polynomial ring $K[t_0, t_1]$ we let

$$f_1 = t_0^4, \quad f_2 = 6t_0^2t_1^2 - 4t_1^4, \quad f_3 = 4t_0^3t_1 - 4t_0t_1^3$$

Example A.2. (Orecchia and Ramella, 2015, Example 3.1) In the polynomial ring $K[t_1, t_2, t_3]$ we let

$$f_1 = t_1t_2^2 - t_2t_3^2 \quad f_2 = t_1t_2t_3 + t_1t_3^2, \quad f_3 = 2t_1t_3^2 - 2t_2t_3^2, \quad f_4 = t_1t_2^2$$

Example A.3. (Emiris et al., 2012, Enneper's Surface, in Table 4) In the polynomial ring $K[s, t]$ we let

$$f_1 = t - \tfrac{1}{3}t^3 + s^2t, \quad f_2 = 2 - \tfrac{1}{3}s^3 + st^2, \quad f_3 = t^2 - s^2$$

Example A.4. (Robbiano, 2015, Example 1.22) In the polynomial ring $K[s, t]$ we let

$$f_1 = s^5 - st^3 - t, \quad f_2 = st^2 - s, \quad f_3 = s^4 - t^2$$

Example A.5. (Busé and Chardin, 2005, Example 3.3.2) In the polynomial ring $K[s, t, u]$ we let

$$\begin{aligned}
f_1 &= s^2t + 2t^3 + s^2u + 4stu + 4t^2u + 3su^2 + 2tu^2 + 2u^3,\\
f_2 &= -s^3 - 2st^2 - 2s^2u - stu + su^2 - 2tu^2 + 2u^3,\\
f_3 &= -s^3 - 2s^2t - 3st^2 - 3s^2u - 3stu + 2t^2u - 2su^2 - 2tu^2,\\
f_4 &= s^3 + s^2t + t^3 + s^2u + t^2u - su^2 - tu^2 - u^3
\end{aligned}$$

Example A.6. (Busé and Chardin, 2005, Example 3.3.4) In the polynomial ring $K[s, t]$ we let

$$\begin{aligned}
f_1 &= s^3 - 6s^2t - 5st^2 - 4s^2u + 4stu - 3t^2u,\\
f_2 &= -s^3 - 2s^2t - st^2 - 5s^2u - 3stu - 6t^2u,\\
f_3 &= -4s^3 - 2s^2t + 4st^2 - 6t^3 + 6s^2u - 6stu - 2t^2u,\\
f_4 &= 2s^3 - 6s^2t + 3st^2 - 6t^3 - 3s^2u - 4stu + 2t^2u
\end{aligned}$$

Example A.7. (Wang, 2004, Example 13, p. 913) In the polynomial ring $K[s, t]$ we let

$$\begin{aligned}
f_1 &= s^3 + 3t^3 - 3s^2 - 6t^2 + 6s + 3t - 1, \quad f_2 = 3s^3 + t^3 - 6s^2 + 3s + 3t,\\
f_3 &= -3s^3t^3 - 3s^3t^2 + 15s^2t^3 + 6s^3t - 18s^2t^2 - 15st^3 + 9s^2t + 27st^2 - 3s^2\\
&\quad -18st - 3t^2 + 3s + 3t
\end{aligned}$$

Example A.8. (Botbol and Dickenstein, 2016, Example 5.1) In the field $K(s, t)$ we let

$$f_1 = \frac{st^6+2}{2+s^2t^6}, \quad f_2 = \frac{st^5-3st^3}{2+s^2t^6}, \quad f_3 = \frac{st^4+5s^2t^6}{2+s^2t^6}$$

Example A.9. (Botbol and Dickenstein, 2016, Example 5.2) In the field $K(s, t)$ we let

$$f_1 = \frac{s^7+s^{47}}{1+st+s^{37}}, \quad f_2 = \frac{s^{37}+s^{59}}{1+st+s^{37}}, \quad f_3 = \frac{s^{61}}{1+st+s^{37}}$$

Example A.10. (Botbol and Dickenstein, 2016, Example 5.3) In the field $K(s, t)$ we let

$$f_1 = \frac{-s^{36}t+1}{1+st}, \quad f_2 = \frac{-t\,(-s^{38}+t)}{1+st}, \quad f_3 = \frac{s^{37}-t}{1+st}$$

Example A.11. (Botbol and Dickenstein, 2016, Example 5.4) In the field $K(s_1, s_2)$ we let

$$f_1 = \frac{3s_1^2s_2 - s_1^2 - 3s_1s_2 - s_1 + s_2 + s_1^2 + s_2^2 + s_1^2s_2^2}{3s_1^2s_2 - 2s_1s_2^2 - s_1^2 + s_1s_2 - 3s_1 - s_2 + 4 - s_2^2}, \quad f_2 = \frac{2s_1^2s_2^2 - 3s_1^2s_2 - s_1^2 + s_1s_2 + 3s_1 - 3s_2 + 2 - s_2^2}{3s_1^2s_2 - 2s_1s_2^2 - s_1^2 + s_1s_2 - 3s_1 - s_2 + 4 - s_2^2},$$
$$f_3 = \frac{2s_1^2s_2^2 - 3s_1^2s_2 - 2s_1s_2^2 + s_1^2 + 5s_1s_2 - 3s_1 - 3s_2 + 4 - s_2^2}{3s_1^2s_2 - 2s_1s_2^2 - s_1^2 + s_1s_2 - 3s_1 - s_2 + 4 - s_2^2}$$

Example A.12. (Emiris et al., 2015, Example 3 and Emiris et al., 2012, Table 4, Bohemian Dome) In the field $K(s, t)$ we let

$$f_1 = \frac{1-t^2}{1+t^2}, \quad f_2 = \frac{1+2t+t^2-s^2-s^2t^2+2ts^2}{(1+t^2)(1+s^2)}, \quad f_3 = \frac{2s}{1+s^2}$$

Example A.13. (Emiris et al., 2012, Table 4, Sine Surface) In the field $K(s, t)$ we let

$$f_1 = \frac{2t}{1+t^2}, \quad f_2 = \frac{2s}{(1+s^2)}, \quad f_3 = \frac{2s}{1+s^2}\frac{1-t^2}{1+t^2} + \frac{2t}{1+t^2}\frac{1-s^2}{1+s^2}$$

A.2. Our own examples

Here are several examples we used while exploring the behaviour of our algorithms.

Example A.14. In the polynomial ring $K[t]$ we let

$$f_1 = t^{15} - 3t^2 - t + 1, \quad f_2 = t^{23} + t^{11} + t^3 - t - 2$$

Example A.15. In the polynomial ring $K[s, t]$ we let

$$f_1 = st^5 - st^3 - t, \quad f_2 = s^3 - st - t^2 - 1, \quad f_3 = s^2t^2 - s$$

Example A.16. In the polynomial ring $K[s, t]$ we let

$$f_1 = s^7 - st^3 - t, \quad f_2 = st^3 - s, \quad f_3 = s^{13} - t^2$$

Example A.17. In the polynomial ring $K[s, t, u]$ we let

$$f_1 = s^2 - st - tu, \quad f_2 = st^2 - su, \quad f_3 = s^3 - t^2 + u, \quad f_4 = s + u^2$$

Example A.18. In the polynomial ring $K[s, t, u]$ we let

$$f_1 = s^3 - t^2 + u, \quad f_2 = s^3 - t^2 + u^2 + s + u, \quad f_3 = s^5 - tu, \quad f_4 = st^2 - su$$

Example A.19. In the polynomial ring $K[s, t, u, w]$ we let

$$f_1 = s^2 - t^2 + w,\ f_2 = s^2 - u - w,\ f_3 = s^2 - tu,\ f_4 = t^2 - su,\ f_5 = s^3 + t - u - w$$

Example A.20. In the polynomial ring $K[s, t, u, w]$ we let

$$f_1 = s^2 - t - u + w,\ f_2 = t^2 - u - w,\ f_3 = s - tu,\ f_4 = u^2 - sw,\ f_5 = s^2 + t - u - w^2$$

Example A.21. In the polynomial ring $K[s, t, u, v, w]$ we let

$$f_1 = s^2 - t - u,\ f_2 = u^2 - sw,\ f_3 = s^2 - v,\ f_4 = u^2 - v - w,\ f_5 = t - u^2,\ f_6 = v^2 - w$$

Example A.22. In the polynomial ring $K[s,t,u,v,w]$ we let

$$f_1 = s^3 - u^2 - t - 3s - u + w, \quad f_2 = u^2 - sw - 11, \quad f_3 = s^2 - 5u - v,$$
$$f_4 = u^2 - s - v - w, \quad f_5 = u^2 + 7s + t, \quad f_6 = v^2 + s^2 - s - t - w$$

Example A.23. In the polynomial ring $K[s,t,u]$ we let

$$f_1 = s^3 - s^2 - t^2 + 3s - 21, \quad f_2 = t^5 - s^5 + st^4 - st + t^2 - s - t - 21,$$
$$f_3 = t^3 - 2t^2 s - 5ts^2 - t^2 + 5s - 12u, \quad f_4 = s + t - u$$

Example A.24. In the field $K(t)$ we let

$$f_1 = \frac{2t^2-t-3}{1+t^{17}} \quad f_2 = \frac{t^4-t+1}{t^2-t-1}$$

Example A.25. In the field $K(s,t)$ we let

$$f_1 = \frac{s^3-t}{t^2-s-t}, \quad f_2 = \frac{s-t}{s^3-2}, \quad f_3 = \frac{s}{s^2+t}$$

Example A.26. In the field $K(s,t)$ we let

$$f_1 = \frac{s^2-t^2-s}{t^2-s-t}, \quad f_2 = \frac{s-t-4}{s^3-2t-5}, \quad f_3 = \frac{s-2}{s^2+t}$$

Example A.27. In the field $K(t_1,t_2,t_3)$ we let

$$f_1 = \frac{t_1t_3-t_2^2}{t_2-t_3}, \quad f_2 = \frac{-t_2+t_3-4}{t_1-2t_2-5}, \quad f_3 = \frac{t_1-2}{t_1^2+t_3}, \quad f_4 = \frac{t_3}{t_1+t_3}$$

Example A.28. In the field $K(s,t,u)$ we let

$$f_1 = \frac{s^3-t-u}{t^2-s-t}, \quad f_2 = \frac{t-u}{s^3-2}, \quad f_3 = \frac{s^2}{s^2+t}, \quad f_4 = \frac{u}{s^2+t}$$

Example A.29. In the field $K(s,t,u,v)$ we let

$$f_1 = \frac{s-t-u}{(t^2-s-t-u-v)(s+u)}, \quad f_2 = \frac{t^3-v^2}{(t^2-s-t-u-v)(s+u)}, \quad f_3 = \frac{u-v}{(t^2-s-t-u-v)(s+u)},$$
$$f_4 = \frac{s+u-v}{((t^2-s-t-u-v)(s+u)}, \quad f_5 = \frac{s^2-5u-6v}{(t^2-s-t-u-v)(s+u)}$$

References

Abbott, J., 2017. Fault-tolerant modular reconstruction of rational numbers. J. Symb. Comput. 80, 707–718.

Abbott, J., Bigatti, A.M., 2016. CoCoALib: a C++ library for doing Computations in Commutative Algebra. Available at http://cocoa.dima.unige.it/cocoalib.

Abbott, J., Bigatti, A.M., Lagorio, G., 2016. CoCoA-5: a system for doing Computations in Commutative Algebra. Available at http://cocoa.dima.unige.it.

Abbott, J., Bigatti, A., Kreuzer, M., Robbiano, L., 2000. Computing ideals of points. J. Symb. Comput. 30 (4), 341–356.

Abbott, J., Bronstein, M., Mulders, T., 1999. Fast deterministic computation of determinants of dense matrices. Proc. ISSAC 1999, 197–204.

Abbott, J., Kreuzer, M., Robbiano, L., 2005. Computing zero-dimensional schemes. J. Symb. Comput. 39, 31–49.

Beltrametti, M., Robbiano, L., 1986. Introduction to the theory of weighted projective spaces. Expo. Math. 4, 111–162.

Böhm, J., Decker, W., Fieker, C., Pfister, G., 2015. The use of bad primes in rational reconstruction. Math. Comput. 84, 3013–3027.

Bastl, B., Ježek, F., 2005. Comparison of implicitization methods. J. Geom. Graph. 9, 11–29.

Botbol, N., Dickenstein, A., 2016. Implicitization of rational hypersurfaces via linear syzygies: a practical overview. J. Symb. Comput. 74, 493–512.

Busé, L., Ba, T.L., 2010. Matrix-based implicit representations of rational algebraic curves and applications. Comput. Aided Geom. Des. 27 (9), 681–699.
Busé, L., Chardin, M., 2005. Implicitizing rational hypersurfaces using approximation complexes. J. Symb. Comput. 40 (4/5), 1150–1168.
d'Andrea, C., 2015. Moving Curve Ideals for Rational Plane Parametrizations. Lect. Notes Comput. Sci., vol. 8942, pp. 30–49.
Emiris, I., Kalinka, T., Konaxis, C., 2012. Implicitization of curves and surfaces using predicted support. In: Proceedings of the 2011 International Workshop on Symbolic-Numeric Computation. ACM, New York, pp. 137–146.
Emiris, I., Konaxis, C., Zafeirakopoulos, Z., 2015. Minkowski decomposition and geometric predicates in sparse implicitization. In: Proceedings of the 2015 International Workshop on Symbolic and Algebraic Computation. ISSAC '15. ACM, New York, pp. 157–164.
Kreuzer, M., Robbiano, L., 2000. Computational Commutative Algebra 1. Springer, Heidelberg.
Kreuzer, M., Robbiano, L., 2005. Computational Commutative Algebra 2. Springer, Heidelberg.
Orecchia, F., Ramella, I., 2015. A new algorithm for implicitizing a parametric algebraic surface. Int. J. Pure Appl. Math. 98 (4), 471–475.
Robbiano, L., 2015. Hyperplane sections, Gröbner bases, and Hough transforms. J. Pure Appl. Algebra 219 (6), 2434–2448.
Wang, D., 2004. A simple method for implicitizing rational curves and surfaces. J. Symb. Comput. 38 (1), 899–914.
Zassenhaus, H., 1969. On Hensel factorization I. J. Number Theory 1, 291–311.