MATHEMATICS OF COMPUTATION
VOLUME 54, NUMBER 190
APRIL 1990, PAGES 797-837

THE COMPUTATIONAL COMPLEXITY OF THE RESOLUTION OF PLANE CURVE SINGULARITIES

JEREMY TEITELBAUM

ABSTRACT. We present an algorithm which computes the resolution of a plane curve singularity at the origin defined by a power series with coefficients in a (not necessarily algebraically closed) field k of characteristic zero. We estimate the number of k-operations necessary to compute the resolution and the conductor ideal of the singularity. We show that the number of k-operations is polynomially bounded by the complexity of the singularity, as measured for example by the index of its conductor ideal. Our algorithm involves calculations over reduced rings with zero divisors, and employs methods of deformation theory to reduce the consideration of power series to the consideration of polynomials.

The problem of resolving singularities is of fundamental interest in modern algebraic geometry. In this paper we make a small step toward approaching this problem from the point of view of computational complexity. We present an algorithm, suitable for machine implementation, which computes the resolution of a plane curve singularity—that is, a singularity at the origin defined by a formal power series F in two variables x and y over a field k. As we describe it, the algorithm requires that k be of characteristic zero (or at least of "large" characteristic) but this hypothesis can certainly be removed at the expense of some complications. The algorithm obtains explicit equations for the blowing-up of the singularity, and therefore yields all of the interesting invariants of the singularity, such as its conductor and its Milnor number. We also provide upper bounds for the number of k-operations needed for the operation of the algorithm.

The problems we consider in this paper have a long history. In [18], Kung and Traub consider the complexity of Newton's method for solving analytic equations. There, they present estimates for the number of times Newton's method must be applied to obtain an approximate solution to an analytic equation before an iterative method can be employed to refine the solution. This process is closely related to the resolution problem. Berry, in [3], considers the complexity of Coates' algorithm [8] for computing Puiseux expansions. Chudnovsky and Chudnovsky [7] have looked at computing Puiseux expansions from the point of view of differential equations. The work of Duval and Dicrescenzo

Received March 14, 1988.
1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 14B05, 68Q40.
This research was supported in part by a Rackham Postdoctoral Research Fellowship at the University of Michigan and by an NSF Postdoctoral Fellowship.

0025-5718/90 $1.00 + $.25 per page

(see [11, 9, 10]), carried out independently from ours, gives another approach to the resolution problem.

Our algorithm for the resolution problem was influenced by two primary considerations. First, we require that our algorithm rely strictly on local data regarding the singularity to be resolved, and we estimate the complexity of the algorithm in terms of this purely local data; and second, we use only arithmetic operations on polynomials over a field, and in particular do not invoke any algorithms for factoring polynomials.

The first restriction was adopted partly for philosophical and partly for practical reasons. From a philosophical point of view, curve singularities (and others) are local phenomena, and therefore one should compute with them using only local data. More practically, it is worth observing that plane algebraic curves—the usual source of curve singularities—typically are far less singular than they can be. In other words, a plane curve of degree, say, d, over a field k, will generically have only a few, rather simple, singularities. It seems reasonable, therefore, that the complexity of an algorithm to resolve those singularities should depend on the complexity of the singularities rather than on the degree of the curve on which they lie.

The decision to work purely locally influences our algorithm in two main ways. First, we assume that our singularity is defined by a power series with coefficients in a field k (assumed for simplicity to be of characteristic zero). We are allowed to ask for any desired coefficient of our power series, but we cannot "know" the entire series at once. Therefore, we must reduce the consideration of an infinite series to the consideration of a polynomial. By applying simple techniques from deformation theory, we show that we can effectively discard all but finitely many terms of the power series. Secondly, we measure complexity in terms of local invariants such as the Milnor number (see §1). We show that the time expended in determining how many coefficients of the power series are necessary to describe the singularity, and the number of terms of the series that we must consider, are polynomial functions of the Milnor number.

To illustrate the consequences of this local approach, suppose that we are attempting to resolve an ordinary double point, at the origin, on a curve of huge degree defined by a polynomial $F(x, y)$. Our method computes quickly that we only need the leading form of F, discards the rest, and therefore the total time needed to blow up this simple singularity is unaffected by the fact that F may have thousands of terms.

To accomplish the reduction of our problem from consideration of power series to polynomials, we apply the theory of Gröbner bases. We find a standard basis for the ideal generated by the partial derivatives F_x and F_y of the power series $F(x, y)$ which defines the singularity. Knowledge of this ideal tells us the Milnor number of F, as well as enabling us, by an application of Tougeron's lemma, to find an integer N such that the polynomial obtained from F by dropping terms of total degree greater than N is analytically equivalent to F.

Our method is related to Mora's method [20] for computing tangent cones, but is complicated by the fact that we work over a power series ring, and also by the fact that we admit coefficient rings with zero divisors (for reasons discussed below).

The second guiding consideration in constructing our algorithm was the desire to avoid the use of polynomial factoring algorithms. The issue of factoring polynomials arises because computers do not naturally compute in algebraically closed fields, so we must begin with singularities defined over finitely generated fields. However, the process of resolution generally introduces field extensions, since the infinitely near points to a singularity need not be defined over the initial field. Some method for handling successive field extensions therefore must be built into the resolution algorithm. We show that, by working with successive extensions of Artinian rings, rather than with successive field extensions, one need never factor a polynomial; rather, whenever the issue of whether a given polynomial factor arises, the necessary factor is already at hand as a by-product of the algorithm. This usually occurs when an element of our coefficient ring needs to be inverted; at such times, we may need to split the coefficient ring into two parts, in one of which our element is zero, and in the other of which it is a unit.

The technique of "passive factorization" which we exploit has been developed independently, in somewhat different settings, by other authors. For example, it is similar to the idea in Lenstra's algorithm for factoring integers using elliptic curves, where one simply treats an integer as prime until one is forced to accept the conclusion that it is not. In addition, Duval and Dicrescenzo (see [9, 10]) have independently developed and implemented ideas similar to ours, and have applied them to the problems of computing Puiseux expansions and to testing absolute irreducibility. Related ideas, under the heading of "flattening stratifications", have apparently also been developed by Schreyer [21].

As with our goal of making the algorithm local, there are both philosophical and practical reasons for adopting this approach. From the practical point of view, by avoiding factoring algorithms we simplify our algorithm, and gain greater generality since we can work over fields where we do not have good factoring algorithms. However, in certain special cases we may pay a price in efficiency.

The philosophical motivation for avoiding factorization comes from the sense that resolution of singularities is properly viewed as a problem in linear algebra. Based on our experience developing this algorithm, we have a hunch, unsupported by evidence, that the lower bound for resolving a plane curve singularity with Milnor number μ is close to the lower bound for solving μ linear equations in μ unknowns—that is, on the order of μ^3 or more likely μ^4. In impressionistic terms, we feel that the two phenomena which contribute to the complexity of resolution—successive field extensions, which increase the number of infinitely near points, and repeated blowings-up, which measure how

singular each such point is—should be treated in a balanced way. Using factorization algorithms puts too much emphasis on the field extension problem. For example, resolving even an ordinary multiple point of multiplicity m could require factoring a polynomial of degree m, and therefore could be quite expensive, even though the singularity is trivial. In addition, use of factorization algorithms will probably not affect the worst case complexity of the blowing-up algorithm, since there are "bad" singularities which are analytically irreducible.

In this paper, we are able to obtain a complexity estimate on the order of μ^6 (see Theorem 49 below) for our version of the blowing-up algorithm. Since we are not convinced by any means that this is a proper lower bound on the complexity of the resolution problem, a more careful study of our algorithm may make it possible to improve our μ^6 estimate. Therefore, we have tried to describe our algorithm in a very detailed manner, in the hope that some future researcher will be able to whittle away at this μ^6 estimate and obtain an estimate closer to μ^4. In any case, the question of a reasonable lower bound on the complexity of resolution is quite open, and deserves attention.

We remark that, since our decision to avoid factoring will force us to work over rings with zero divisors after first blowing up, we assume that our initial singularity is defined over a ring with zero divisors. This way we need not consider the first blowing up as a special case, and so we may design our algorithm to be recursive.

Finally, it is appropriate to give a few comments regarding implementation of this algorithm. As described, it should be relatively straightforward to implement this algorithm in a symbolic computation language such as MAPLE or MACSYMA. To get good performance on other than simple singularities, it would be worth writing a special purpose program. We have, in the course of designing the algorithm, written a host of small test programs, on various different computers, in various languages, at different universities. As a result, while we have considerable experience with implementing small pieces of this algorithm, and have expended considerable effort to make the version of it described in this paper an efficient implementation, we have never had a complete working model. Development of such a model would be a worthwhile project.

We would like to express our appreciation to David Mumford, who originally suggested this problem to us.

The paper is divided into three sections. The first section briefly discusses the various measures of the complexity of a singularity which we will consider, as well as the small amount of deformation theory that we require. The second section presents the algorithm which computes an algebraic deformation of a singularity defined by a power series—that is, it finds a polynomial defining a singularity analytically isomorphic to the original singularity defined by a power series. Much of the time is spent developing a data structure, which we call a "tree of standard bases", which is an efficient way to organize these calculations. The third section describes the resolution algorithm itself.

1. Preliminaries

We begin with a brief discussion of plane curve singularities and their invariants. Let k be a field. A plane curve singularity over k is defined by a power series $F(x, y)$ in the ring $A = k[[x, y]]$ such that the quotient ring A/FA is reduced. Since A is a UFD, this means that F must be square free—in other words, if $F = u\prod \pi_i^{n_i}$ is a factorization of F, where u is a unit and the π_i are distinct prime elements, then all of the n_i must be one. Throughout this paper, we will be working with power series which we assume to be square free. In fact, the question of determining whether or not a power series is square free is uncomputable.

Lemma 1. *There is no algorithm for deciding whether or not a power series* $F = \sum a_{ij}x^i y^j$ *with coefficients in a field* k *of characteristic zero is square free.*

Proof. Suppose we had such an algorithm. Let $f: \mathbf{N} \to \mathbf{Z} \subset k$ be any computable function. Let $F = y^2 + \sum f(i)x^i$. Then it is easy to see that F is not square free if and only if f is identically zero. By applying our square free algorithm to F, we could determine whether or not f was identically zero. But this is not possible—for example, such an algorithm would lead to a solution to Hilbert's tenth problem. □

It is an interesting question to ask what additional hypotheses one can place on formal power series F so that whether or not F is square free becomes computable. For example, if F is in fact a polynomial, then it is well known that one can determine if F is square free by computing its discriminant. Is there a class of power series F, with coefficients given by a computable function, which is larger than the polynomials and for which the predicate "F is square free" is computable?

Let p be a square free polynomial with coefficients in a field k, and let $D(p) = k[T]/p$. Let F be a power series in x and y, with coefficients in D. Geometrically, such a power series represents finitely many plane curve singularities. We will be interested in the following invariants of the ring $D(p)[[x, y]]/F$. For a more detailed discussion of these, see Brieskorn and Knorrer [4, Chapter 8], Serre [22], or Gorenstein's paper [14].

Definition 1. Let $A = D(p)[[x, y]]$ and let $F \in A$ be a square free power series. Let $B = A/FA$. Then we define the following invariants:

1. The *multiplicity* m_F of F is the largest integer such that $F \in (x, y)^{m_F}$.
2. The *Milnor number* of F, μ_F, is the number:
$$\mu_F = \dim_k A/(\partial F/\partial x, \partial F/\partial y).$$
3. The *conductor* c_F is the number
$$c_F = 2\dim_k B^*/B,$$
where B^* denotes the total integral closure of B.

4. The *degree* δ_F of F is the smallest integer δ_F such that $(x, y)^{\delta_F} \subset (\partial F/\partial x, \partial F/\partial y)$.

Since F is square free, all of these invariants are finite. We remark that our c_F is twice what some people call the "conductor"of F; but since we will be using it to measure asymptotic complexity, this factor of two is irrelevant.

Lemma 2. *Let $p(T) = p_1(T)p_2(T)$ be a factorization of p over k. Let $D_i = k[T]/p_i$, and let F_i be the image of $F \in D[[x, y]]$ in the quotient ring $A_i = D_i[[x, y]]$. Then the various invariants of F are related to those of the F_i in the following manner:*

1. $m_F = \min\{m_{F_1}, m_{F_2}\}$.
2. $\mu_F = \mu_{F_1} + \mu_{F_2}$.
3. $c_F = c_{F_1} + c_{F_2}$.
4. $\delta_F = \max\{\delta_{F_1}, \delta_{F_2}\}$.

Proof. All of these properties follow easily from the Chinese remainder theorem. □

We will need the following relationships between these invariants.

Lemma 3. *The following relationships hold between the fundamental invariants:*

$$m_F - 1 \le \delta_F \le \mu_F \le c_F \le 2\mu_F \le C \deg(p)\delta_F^2.$$

Proof. By Lemma 2, we may assume that our coefficient ring is a field. Since m_F is the degree of the leading form of F, we know that $(\partial F/\partial x, \partial F/\partial y)$ is contained in $(x, y)^{m_F - 1}$. Therefore, $\delta_F \ge m_F - 1$. We know from the general theory (again, see [4]) that μ_F and c_F are related by the formula

$$\mu_F = c_F - r + 1,$$

where r is the number of connected components of the resolution of F (over $\bar{k}$). Since $r \ge 1$, we must have $\mu \le c_F$. However, we also know that $r \le m$. It follows from the classical formula for the conductor [4, p. 764] in terms of the multiplicities of infinitely near points that

$$c_F \ge m_F(m_F - 1) \ge r(r - 1) \ge 2(r - 1),$$

since $r \ge 1$. It follows that $2\mu_F \ge c_F$. Now we show that $\delta_F \le \mu_F$. For each element f of $I = (\partial F/\partial x, \partial F/\partial y)$, let f^* be its leading form—that is, the homogeneous part of f of least degree. For each degree i, let

$$M_i = \{f\colon \deg(f^*) = i\}, \qquad M_i^* = \{f^*\colon f \in M_i\}.$$

It is easy to see that if i is such that M_i^* spans all monomials of degree i, then $\delta \le i$. Therefore, we can choose a monomial m_i of degree i, but not belonging to M_i, for each $i < \delta_F$. If $h = \sum a_i m_i$, then the leading form of h is (a multiple of) one of the m_i, and therefore h is not in I. It follows that the m_i are linearly independent mod I, and so $\mu_F \ge \delta_F$.

The last inequality follows from the fact that $A(p)/I_F$ is a subspace of $A(p)/(x, y)^{\delta_F}$. □

Finally, we present the lemma from deformation theory which we will exploit. This lemma, a special case of Tougeron's lemma, shows that, if F is a power series, then the singularity defined by F depends, up to analytic isomorphism, on only finitely many coefficients of F, and that the number of coefficients required to "know" F is polynomially bounded by μ_F.

Lemma 4 (see Artin [1, p. 100]). *Let k be a field of characteristic zero, let p be a square free polynomial over k, and let F be an element of the ideal (x, y) in $A = k[T]/p[[x, y]]$. Let $I = (\partial F/\partial x, \partial F/\partial y)$. Suppose G is another power series in A such that $G \equiv F \pmod{(x, y)I^2}$. Then there are power series*

$$u(x, y) = x + \cdots, \qquad v(x, y) = y + \cdots,$$

such that $u \equiv x \pmod{I(x, y)}$ and $v \equiv y \pmod{I(x, y)}$, and $G(x, y) = F(u(x, y), v(x, y))$.

Proof. Again, we may assume that p is irreducible— which amounts to saying that F has coefficients in k. Let $F_x = \partial F/\partial x$ and $F_y = \partial F/\partial y$. We will try to find a, b, and c in (x, y) so that

$$G(x, y) = F(x + aF_x + bF_y, y + cF_y).$$

Expanding in a Taylor series, we have

$$G(x, y) - F(x, y) = aF_x^2 + bF_xF_y + cF_y^2 + R(a, b, c, x, y).$$

It is not hard to see that $R(a, b, c, x, y)$ can be written as

$$R(a, b, c, x, y) = H_1(a, b, c)F_x^2 + H_2(a, b, c)F_xF_y + H_3(a, b, c)F_y^2,$$

where the H_i are power series in $A[[a, b, c]]$, all terms of which are of degree at least two in a, b, and c. Since $G \equiv F \pmod{I(x, y)}$, we may write

$$G - F = rF_2^2 + sF_xF_y + tF_y^2,$$

where r, s, and t belong to (x, y). Therefore, we will be done if we can solve the system of analytic equations:

$$\begin{aligned} a + H_1(a, b, c) &= r, \\ b + H_2(a, b, c) &= s, \\ c + H_3(a, b, c) &= t. \end{aligned}$$

Clearly, setting $a = r$, $b = s$, and $c = t$ gives an approximate solution to this equation; since the Jacobian matrix of the system is invertible and r, s, and t belong to (x, y), Hensel's lemma gives us an exact solution. This proves the lemma. □

We apply this lemma in the following setting:

Lemma 5. *Let F be a square free power series in $A = (k[T]/p)[[x, y]]$, and let G be the polynomial constructed from F by dropping all terms in F of degree larger than $2\delta_F$. Let $I_F = (\partial F/\partial x, \partial F/\partial y)$ and let I_G be the similar ideal for G. Then $I_G = I_F$ and there is an automorphism $\theta: A \to A$ carrying F to G and inducing the identity map on $A/I_F = A/I_G$.*

Proof. Once again, we may reduce to the field case. It follows immediately from Lemma 4 that there is an automorphism $\theta: A \to A$ which induces the identity on A/I_F and carries F to G. Therefore, we only need to show that $I_F = I_G$. We leave it to the reader to show that this equality follows easily from the assumption that $F \equiv G \pmod{I_F^2(x, y)}$. □

It is worth pointing out that we never actually need to compute the analytic isomorphism referred to in this lemma. It is sufficient for our purposes to know that it exists, and that it is congruent to the identity $\bmod I_F$. Notice also that the number of monomials in a polynomial of degree 2δ is $O(\delta^2)$. This, in turn, is bounded by $O(\mu_F^2)$. Therefore, the amount of data necessary to describe a plane curve singularity is polynomially bounded in the Milnor number of the singularity.

2. Computing an algebraic deformation

Let $p(T)$ be a square free polynomial with coefficients in a field k, and let

$$D(p) = k[T]/p(T).$$

Suppose that $A(p) = D(p)[[x, y]]$ is the ring of formal power series with $D(p)$ coefficients. Let $f \in A(p)$ and let $I_F = (\partial F/\partial x, \partial F/\partial y)$. Our aim in this section is to describe a method for determining the invariants δ_F and μ_F. This is equivalent to finding an algebraic deformation of F, since we know by Lemma 5 that knowledge of δ_F enables us to construct a polynomial G which is analytically equivalent to F. Our method for computing δ_F and μ_F is to compute a special type of generating set for the ideal $I_F = (\partial F/\partial x, \partial F/\partial y)$, from which we can read off the desired invariants.

2.1. Standard bases. In order to describe a standard basis, let us order the monomials in x and y lexicographically within degree, so that

$$1 > x > y > x^2 > xy > y^2 > \cdots.$$

If $f \in A(p)$ is a power series, then we define the leading term $l(f, p)$ to be the largest monomial in f with a nonzero coefficient, and we let $c(f, p)$ denote its coefficient. Then the following definition describes a standard basis when p is prime.

Definition 2 (see [5, 2]). Suppose that $p(T)$ is a prime polynomial over k, and that $I \subset A(p)$ is an ideal. Then a set B of elements $B = \{f_1, \ldots, f_n\}$ of $A(p)$ such that $l(f_1, p) > l(f_2, p) > \cdots$ and such that the f_i generate I is called a *standard basis* for I provided that

$$B^* = \{l(f_1, p), \ldots, l(f_n, p)\}$$

generates the ideal

$$I_F^* = \{l(f, p) | f \in I\}$$

of leading forms of elements of I.

Standard bases for I are useful because they provide a method of constructing canonical representatives for elements of A/I, as we see in the following theorem.

Theorem 6 (Hironaka [15]). *Suppose that p is prime. Suppose for $I \subset A$ that $B = \{f_1, \ldots, f_n\}$ is a standard basis. Then every element $g \in A$ has a unique representation*

$$g = \sum_{i=1}^{n} a_i f_i + a_{n+1}, \qquad a_i \in A, \tag{1}$$

where no monomial m occurring in a_{n+1} with nonzero coefficient is divisible by any $l(f_i, p)$, and if m is a monomial with nonzero coefficient in a_i, $i \leq n$, then $ml(f_i, p)$ is not divisible by $l(f_j, p)$ for any $j < i$.

We adopt some terminology for a representation of the form described in this theorem.

Definition 3. If $f_1, \ldots, f_n$ are a decreasing set of elements of A such that $l(f_k, p)$ is not divisible by $l(f_i, p)$ for all $i < k$, then an expression

$$g = \sum_{i=1}^{n} a_i f_i$$

for $g \in I$ with the property described in Theorem 6 will be called a *normal form* for g.

The difficulty with using this theorem in our situation is caused by the presence of zero divisors in the ring $D(p)$ when p is not irreducible. To deal with this problem, we adopt the following definition.

Definition 4. Let $p(T)$ be a square free polynomial over the field k, and let $I \in A(p)$ be an ideal. Then a set $B = \{f_1, \ldots, f_n\}$ of generators for I will be called a standard basis for I *over* $D(p)$ provided that B^* generates I^* as in the field case, and all $c(f_i, p)$ are units in $D(p)$.

Thanks to the Chinese remainder theorem, we have a normal form theorem in this more general case as well.

Lemma 7. *Let $p(T)$ be a square free polynomial, and $I \subset A(p)$ be an ideal. Suppose that I has a standard basis $B = \{f_1, \ldots, f_n\}$ over $D(p)$. Assuming that the f_i are listed in descending order, every element $g \in A(p)$ can be written uniquely as*

$$g = \sum_{i=1}^{n} a_i f_i + a_{n+1}, \qquad a_i \in A(p),$$

where, as in the field case, no monomial m *occurring in* a_{n+1} *with nonzero coefficient is divisible by any* $l(f_i, p)$, *and if* m *is a monomial with nonzero coefficient in* a_i, $i \leq n$, *then* $ml(f_i, p)$ *is not divisible by* $l(f_j, p)$ *for any* $j < i$.

Proof. Let $p = \prod p_k(T)$ be the factorization of $p(T)$ into irreducibles. Then, by the Chinese remainder theorem, there is an isomorphism

$$A(p) \to \prod A(p_k).$$

Since the f_i have unit leading terms, the image of B in each factor is a standard basis for the image of I. Thus we can obtain an expression of the desired form in each factor. Using the isomorphism, we obtain an expression

$$g = \sum_{i=1}^{n} a_i f_i + a_{n+1}, \qquad a_i \in A(p),$$

which is in normal form in each component. But then suppose that m occurs with coefficient u in a_i and that $ml(f_i, p)$ is divisible by $l(f_j, p)$ for $j < i$. Then the coefficient of $ml(f_i, p)$, which is $uc(f_i, p)$, must reduce to zero in each component. Since $c(f_i, p)$ is a unit, this means that u reduces to zero in each component, and so $u = 0$. We conclude that we have a normal form for g. □

The following corollary, which tells how to determine the Milnor number, is an immediate consequence of the normal form lemma. It says that the index of an ideal is the same as the index of the monomial ideal generated by the leading terms of elements in a standard basis. Computing this index is just a linear algebra problem.

Corollary 8. *Let* F *be a square free power series in* $A(p)$, *and suppose that* I_F *has a standard basis* B *over* $D(p)$. *Let* $B^* = \{l(f, p) : f \in B\}$. *Then*

$$\mu_F = \dim_k A(p)/B^*A(p) = \deg(p) \dim_k A/B^*A.$$

We will explain later how to compute the degree invariant δ.

2.2. Computing standard bases in power series rings. Buchberger has devised an algorithm for computing standard bases of ideals in polynomial rings which is based on a criterion for determining when a set of generators is a standard basis (see [5 and 20]). Our first task is to adapt this criterion slightly so that we can apply it to power series with coefficients in $D(p)$. We begin by introducing the R and S operations.

Definition 5. Suppose that g and h are elements of $A(p)$. Let

$$m = \operatorname{lcm}(l(g, p), l(h, p)).$$

Then

$$S(g, h) = \frac{mc(h, p)}{l(g, p)} g - \frac{mc(g, p)}{l(h, p)} h.$$

Definition 6. Let $G = \{g_1, \ldots, g_n\}$ be a set of elements of $A(p)$ such that $c(g_i, p) = 1$ for all i. Let h be any element of $A(p)$. Then we define $R_n(h, G, p)$ by the following rules:

1. If there is no element $g \in G$ such that $l(g, p)$ divides $l(h, p)$, then $R_1(h, G, p) = h$.
2. Otherwise, let g be the largest element of G such that $l(g, p)$ divides $l(h, p)$ and set $R_1(h, G, p) = S(h, g)$.
3. Define $R_n(h, G, p) = R_1(R_{n-1}(h, G, p), G, p)$.

Finally, we set $R(h, G, p) = \lim_{n\to\infty} R_n(h, G, p)$.

We make the following observations regarding the R-operation.

Lemma 9. *The R-operation is well defined. Let $f = R(h, G, p)$. Then, if f is not zero, $l(f, p)$ is not in the span of the $l(g, p)$ as g runs through G. Furthermore, if $f \neq h$, then there are series $a(g) \in A(p)$ such that*

$$h - f = \sum_{g\in G} a(g)g$$

is in normal form.

Proof. It is easy to see that the sequence $R_n(h, G, p)$ converges in the *adic* topology on A; in fact, if f is not zero, then it will eventually stabilize. Thus, either f is zero or $l(f, p)$ is not in the span of the $l(g, p)$. The normal form claim follows easily by induction and the observation that

$$R_{n-1}(h, G, p) = mg + R_n(h, G, p),$$

where $ml(g, p) = l(R_{n-1}(h, G, p), p)$ and g is maximal among elements of G such that $l(g, p)$ divides $l(R_{n-1}(h, G, p), P)$. □

Having defined these operations, we have the following power series version of Buchberger's criterion.

Theorem 10. *Suppose that $I \subseteq A(p)$ is an ideal, $B = \{f_1, \ldots, f_n\}$ is a descending sequence of distinct elements of $A(p)$ which generate I, and that all $c(f_i, p) = 1$. If, for all pairs (i, j), we may write*

$$S(f_i, f_j) = \sum a_k^{(ij)} f_k,$$

where all monomials m with nonzero coefficient in $a_k^{(ij)}$ satisfy $ml(f_k, p) < \operatorname{lcm}(l(f_i, p), l(f_j, p))$, then B is a standard basis for I over $D(p)$.

Proof. Let g be an element of I. Assume that $l(f_i, p) > l(f_j, p)$ whenever $i < j$. If $e \in A(p)^{\oplus n}$, we will write $e \cdot B = \sum e_i f_i$. With these conventions established, let

$$E(g) = \{(e_0, \ldots, e_n) \in A(p)^{\oplus n+1} : g = e \cdot B\}.$$

If m is a monomial occurring in some component e_i of $e \in E(g)$ such that $ml(f_i, p)$ is divisible by $l(f_j, p)$ for $j < i$, then we will say that $ml(f_i)$ is in

the *wrong place*. Let $\mathrm{err}(e)$ be a largest monomial in some $e_i l(f_i)$ which is in the wrong place if such a one exists, and 0 otherwise. Let $\mathrm{Err}(g)$ be the set of all $\mathrm{err}(e)$ as e runs though $E(g)$.

Suppose that $\mathrm{Err}(g)$ contains arbitrarily small monomials. Then we claim that there is an $e \in E(g)$ such that $g = e \cdot B$ is a normal form for g. In fact, this hypothesis means that we can approximate g arbitrarily closely by elements which can be written in normal form. But it follows easily from the uniqueness of the normal form that if $\{e^n\}$ is a sequence of elements of $E(g)$ such that $e^n \cdot B \to g$, then the e^n converge to an element e with $e \cdot B = g$ giving a normal form.

On the other hand, let t be a minimal nonzero element of $\mathrm{Err}(g)$, and suppose that t occurs in $e_i l(f_i, p)$. For each i, let m_i be the monomial in e_i such that $m_i l(f_i) = t$. Let j be minimal such that $l(f_j)$ divides t. Then we can write, for each i,

$$m_i f_i - m_i' f_j = m_i'' S(f_i, f_j)$$

with monomials m_i' and m_i''. But by assumption, we know that

$$m_i'' S(f_i, f_j) = a^{(i)} \cdot B$$

for some $a^{(i)} = (a_0^{(i)}, \ldots, a_n^{(i)}) \in E(g)$, where all monomials occurring in $a_k^{(i)} l(f_j)$ in $a^{(i)} \cdot B$ are strictly smaller than t. Let $v^{(i)}$ be the vector with $-m_i$ in the ith position, m_i' in the jth position, and zero elsewhere. Then $e' = e - \sum_i (a^{(i)} + v^{(i)})$ belongs to $E(g)$ and $\mathrm{err}(e')$ is by construction smaller than $\mathrm{err}(e)$. It follows that $\mathrm{err}(e')$ is zero. Therefore, every $g \in I$ has a representation in normal form. This clearly implies that B is a standard basis over $D(p)$, as was to be shown. □

In the next subsection, we consider the problem of doing arithmetic on power series.

2.3. Computations with power series. In order to construct a standard basis for an ideal in a polynomial ring, we must develop a method for computing with power series. In this subsection, we examine this problem.

Our concern is computation of a standard basis for the ideal $I_F = (F_x, F_y) \subset A(p)$ for a particular power series F. In practice, such a series may arise in a number of different ways. For example, it may be presented as a large polynomial, all of whose coefficients are known. More interestingly, F may be the output of an iterative procedure which computes the coefficients of F inductively. We wish to deal with this more general situation. We therefore adopt the following representation for F.

Definition 7. Let $M_d = \{q \in k[T]\colon \deg(q) < d\}$. Let $p(T)$ be a square free polynomial of degree d. The power series $F \in A(p)$ is represented by a function

$$\tilde{F}\colon \mathbf{N} \times \mathbf{N} \to M_d,$$

where $\tilde{F}(i, j)$ is interpreted as a representative $a_{ij} \in D(p)$ for the coefficient of $x^i y^j$ in F.

We observe that the function $(\tilde{F}_x)$, which represents $\partial F/\partial x$, is defined by

$$(\tilde{F}_x)(i, j) = (i+1)\tilde{F}(i+1, j),$$

with a similar formula holding for $(\tilde{F}_y)$.

For elements $h \in I_F$ which will arise in the course of our computations, we adopt a different representation. The power series F defines a map $\phi(\cdot, p)$:

$$D(p)[x, y] \oplus D(p)[x, y] \to I_F,$$
$$(h_1, h_2) \overset{\phi(\cdot, p)}{\mapsto} h_1 F_x + h_2 F_y.$$

We will compute with elements h of I_F which are in the image of the map ϕ, representing them as ordered pairs of polynomials with $D(p)$-coefficients. Clearly, we can perform arithmetic on such pairs by operating componentwise.

Lemma 11. *Suppose $\vec{h} = (h_1, h_2)$ is an ordered pair of polynomials with coefficients in $D(p)$. If $d(\vec{h})$ is the larger of the degrees of h_1 and h_2, then the coefficient of any monomial in $h_1 F_x + h_2 F_y = h$ can be computed with $O(d(\vec{h})^2)$ coefficient ring operations.*

Proof. Write $h_i = \sum a_{ijk} x^i y^k$. Then the coefficient b_{rs} of $x^r y^s$ in h is

$$b_{rs} = \sum_{j+j'=r,\, k+k'=s} a_{1jk}\tilde{F}_x(j', k') + a_{2jk}\tilde{F}_y(j', k'),$$

and there are $O(d(\vec{h})^2)$ nonzero terms in this calculation. □

We also remark that if $\phi(\vec{h}, p) = h \neq 0$, then we can compute the leading term $l(h, p)$ of h by computing in sequence the coefficients of the monomials in h, looking for the first nonzero coefficient. However, if $h = 0$, this procedure may not terminate.

Because the determination of leading terms is such an important part of any standard basis algorithm, we expand our abstract data type for elements $\vec{h} \in D(p)[x, y]^{\oplus 2}$ to include the leading term of $\phi(\vec{h}, p)$.

Definition 8. We define an abstract data type *vector*. An element $\vec{h}$ of this type consists of:

1. A monomial, called the leading monomial of $\vec{h}$.
2. A polynomial in T, called the leading coefficient of $\vec{h}$.
3. Two polynomials in x and y, with coefficients in the polynomials in T, called the components of $\vec{h}$.

Vectors are used to represent elements of $A(p)$ as follows. Let $h \neq 0$ be an element of $A(p)$ which can be written $h = \phi((h_1, h_2), p)$, where (h_1, h_2)

is a pair of polynomials. Then h is stored as a vector with the leading monomial field set to $l(h, p)$, the leading coefficient field set to $c(h, p)$, and the components set to h_1 and h_2.

Vectors of this type are added together by adding their components and recomputing the leading monomial and coefficient. They are multiplied by a monomial m by multiplying the components and the leading monomial by m. Scalar multiplication (by a nonzero scalar u) is computed by multiplying the leading coefficient and the components by u. We abuse notation and write $\phi(\vec{h}, p)$ for the $\phi(\cdot, p)$ applied to the components of $\vec{h}$, $l(\vec{h}, p)$ for the leading monomial, and $c(\vec{h}, p)$ for the leading coefficient.

Notice that addition is only defined on $\vec{h}$ and $\vec{g}$ when $\phi(\vec{h} + \vec{g}, p) \neq 0$. If the sum is zero, the search for a new leading coefficient in the sum will never terminate.

Algorithm 12. *Algorithm for* $\vec{R}$.

Input: A square free polynomial $p(T)$, a vector $\vec{h}$, and a finite list $\vec{G}$ of vectors $\vec{g}_i$, representing elements in $A(p)$ in the manner described above, such that $h = \phi(\vec{h}, p)$ and all $g_i = \phi(\vec{g}_i)$ are nonzero. We assume further that the $l(g_i, p)$ are distinct and all $c(\vec{g}_i, p) = 1$. A further condition, discussed in Lemma 13, is necessary to guarantee termination.

Output: A vector $\vec{r}$ representing $R(h, G, p)$ (here $G = \{\phi(\vec{g}_i, p)\}$).

Step 1: Set $\vec{r} := \vec{h}$.

Step 2: While there exists i such that $l(\vec{g}_i, p)$ divides $l(\vec{r}, p)$, do:

Let $\vec{g}_j$ be the element of G which is maximal among all $\vec{g} \in G$ such that $l(\vec{g}, p)$ divides $l(\vec{r}, p)$. Set

$$\vec{r} := \vec{r} - (c(\vec{r}, p)l(\vec{r}, p)/l(\vec{g}_j, p))\vec{g}_j, \tag{2}$$

carrying out arithmetic componentwise.

Determine $l(\vec{r}, p)$ by searching for a nonzero coefficient $\bmod p(T)$.

Step 3: Return $\vec{r}$.

That this algorithm computes what it claims to is a simple consequence of the definition of the R-operation. However, the following lemma clarifies when the procedure actually terminates after finitely many passes through the loop in Step 2.

Lemma 13. *If h has no normal form expression in the g_i, then Algorithm* 12 *terminates.*

Proof. Suppose the algorithm fails to terminate. Then we must have $R(h, G, p) = 0$, and we conclude by Lemma 9 that we can find a_i so that

$$h = \sum a_i g_i,$$

and this expression is in normal form. □

In our discussion of complexity, we will require the following result.

Lemma 14. *Suppose that* $\vec{h}$ *is a vector. Let* $d(\vec{h})$ *be the degree of the largest monomial occurring in a component of* $\vec{h}$, *and let* $dl(\vec{h}) = \deg(l(\vec{h}, p))$. *Also, we define*

$$r(\vec{h}) = \#(\{m\colon m \text{ a monomial with } m > l(\vec{h}, p)\}).$$

Let $\vec{G}$ *be a list of vectors, and set*

$$d(\vec{G}) = \max\{d(\vec{g})\colon \vec{g} \in \vec{G}\}, \qquad r(\vec{G}) = \min\{r(\vec{g})\colon \vec{g} \in \vec{G}\}.$$

Finally, let $\vec{f} = \vec{R}(\vec{h}, \vec{G}, p)$. *Then* $\vec{f}$ *is computed by Algorithm* 12 *using*

$$O((\deg(p)^2)(r(\vec{f}) - r(\vec{h}))(d(\vec{G}) + dl(\vec{h})))$$

field operations.

Proof. Addition of polynomials of degree d in two variables can be computed with $O(d^2)$ coefficient ring operations. Let $\vec{h}_i$ be the vector computed in the ith pass through the loop in Algorithm 12. Since $\vec{h}_i := \vec{h}_{i-1} - m\vec{g}$ for some $\vec{g} \in \vec{G}$, $\vec{h}_i$ can be computed from $\vec{h}_{i-1}$ by $O(\max(d(\vec{G}), d(\vec{h}_{i-1}))^2)$ coefficient ring operations. In addition, since $ml(\vec{g}, p) = l(\vec{h}_{i-1}, p)$, we must have

$$d(\vec{h}_i) \leq \max(d(\vec{g}) + dl(\vec{h}_{i-1}), d(\vec{h}_{i-1})).$$

We conclude from this that

$$d(\vec{h}_i) \leq \max(d(\vec{G}) + dl(\vec{f}) - dl(\vec{G}), d(\vec{h})).$$

Therefore, each computation of h_i in (2) requires at most

$$T_1 = C_1(\max(d(\vec{G}) + dl(\vec{f}), d(\vec{h}))^2)$$

coefficient ring operations.

Again, passing through the loop, we must compute the leading term $l(\vec{h}_i, p)$. This involves verifying that $T_2(i) = (r(\vec{h}_i) - r(\vec{h}_{i-1}))$ monomials have zero coefficients; each verification requiring $C_2(d(\vec{h}_i)^2)$ coefficient operations. To finish the lemma, we sum over passes through the loop:

$$\text{time} < \sum_i (T_1 + C_2 T_2(i)(d(\vec{h}_i)^2)).$$

Using the fact that the number of passes through the loop is bounded by the number of monomials between $l(\vec{h}, p)$ and $l(\vec{f}, p)$, which is $r(\vec{f}) - r(\vec{h})$, and substituting our estimates for T_1, T_2, and $d(\vec{h}_i)$, we see that computing f requires at most

$$O((r(\vec{f}) - r(\vec{h}))\max(d(\vec{h}), d(\vec{G}) + dl(\vec{h})))$$

coefficient ring operations. Converting coefficient ring operations to field operations completes the proof of the lemma. □

We also have a means for computing the S-operation on power series represented as vectors $\vec{h} \in A(p) \oplus A(p)$.

Algorithm 15. *$\vec{S}$-operation.*

Input: Two vectors $\vec{h}$ and $\vec{g}$ representing elements h and g in $A(p)$ such that $S(\phi(\vec{h}, p), \phi(\vec{g}, p)) \neq 0$.

Output: A vector $\vec{s}$ such that $\phi(\vec{s}, p) = S(h, g, p)$.

Step 1: Compute $m = \operatorname{lcm}(l(\vec{h}, p), l(\vec{g}, p))$.

Step 2: Compute

$$\vec{s} := \frac{mc(\vec{h}, p)}{l(\vec{g}, p)}\vec{g} - \frac{mc(\vec{g}, p)}{l(\vec{h}, p)}\vec{h},$$

doing arithmetic componentwise.

Step 3: Search for the leading coefficient of $\vec{r}$.

Step 4: Return $\vec{s}$, together with its leading term and coefficient.

An analysis similar to, but simpler than, the one given for the $\vec{R}$-operation gives the following estimate.

Lemma 16. *Let* $\vec{s} = \vec{S}(\vec{g}, \vec{h}, p)$. *Then*

$$d(\vec{s}) \leq d(m) + \max(d(\vec{g}) - dl(\vec{g}), d(\vec{h}) - dl(\vec{h})),$$

where $m = \operatorname{lcm}(l(\vec{g}, p), l(\vec{h}, p))$. *Computation of* $\vec{s}$ *by Algorithm* 15 *requires*

$$O(\deg(p)^2 d(\vec{s})^2 (r(\vec{s}) - r(m)))$$

field arithmetic operations.

2.4. Trees of standard bases. In this subsection we discuss how to deal with the presence of zero divisors in our ring of coefficients. Our method will consist of "discovering" factors of the polynomial $p(T)$ which defines $D(p)$, and then splitting up $D(p)$ into corresponding factor rings.

In general, if $q(T)$ is a divisor of $p(T)$, there is a natural reduction map $D(p) \to D(q)$, inducing a reduction map $A(p) \to A(q)$. In practice, if F is a power series in $A(p)$ represented by a function $\widetilde{F}$, then the image of F in a quotient $A(q)$ of $A(p)$ is represented by the function obtained by reducing the values of $\widetilde{F} \bmod q$. Therefore, we can carry out arithmetic in any factor ring of $A(p)$. We will therefore freely view any element of $A(p)$ also as an element of $A(q)$ via the reduction map when q divides p. For example, if $f \in A(p)$ and q divides p, then $l(f, q)$ will denote the first coefficient of f which is nonzero $\bmod q$.

It is important to notice that $l(f, q)$ and $l(f, p)$ are not in general equal. Indeed, the coefficient $c(f, p)$ might reduce to zero in $D(q)$. However, whenever $c(f, p)$ is a unit $\bmod p$, then $l(f, q) = l(f, p)$ for all q dividing p.

The following algorithm splits up the polynomial $D(p)$ into factors $p_i(T)$ so that, if F_i is the image of F in $D(p_i)$, then the leading coefficient of $l(F_i, p_i)$ is a unit $\bmod p_i$. Among other things, this guarantees that the multiplicity of F_i at all maximal ideals of $A(p_i)$ is the same.

Algorithm 17. *Constant multiplicity.*

Input: A square free polynomial $p(T)$ and a power series F in $A(p)$, represented as a function in the manner described above. We assume that the image of F in each irreducible component of $D(p)$ is known to be nonzero.

Output: A list of triples $(u_k(T), m, p_k(T))$, each consisting of a polynomial $u_k(T)$, a monomial m, and a polynomial $p_k(T)$ such that

1. $u_k(T)$ is the inverse of the coefficient of $x^i y^j \bmod p_k(T)$.
2. m is the leading term of F viewed $\bmod p_k$—this means that m is the first monomial with a coefficient not divisible by p_k.
3. The coefficient of m is a unit $\bmod p_k$—meaning that $\tilde{F}(i, j)$ is relatively prime to p_k.
4. $p = \prod p_k(T)$.
5. The p_k are pairwise relatively prime.

Step 0: $LIST := \{\}$. $m := x^0 y^0$.
Step 1: While $p \neq 1$ do:
Let u be the coefficient of m in F.
Let $d := \gcd(u, p)$.
If $d \neq 0$ do:
Let $q := p/d$.
Append $\{(u^{-1} \bmod q, m, q)\}$ to $LIST$.
$p := d$.
$m :=$ next monomial in order.

Lemma 18. *Algorithm* 17 *terminates. Further, the list it returns has the claimed properties.*

Proof. Each pass through the algorithm for which $d \neq 0$ holds will reduce the degree of the polynomial p. Therefore, the algorithm can fail to terminate only if there is a factor q of p such that all coefficients u of F are congruent to zero $\bmod q$. This contradicts our assumption that F is nonzero in all factors of $A(p)$. Therefore, the algorithm terminates. The correctness of the algorithm follows by induction. In particular, let m be the first monomial in F with nonzero coefficient $u \bmod p$. If $d = \gcd(u, p)$ is 1, then the algorithm returns $\{(u^{-1} \bmod p, m, p)\}$ which has the desired properties. If $d \neq 1$, then d is a factor of p and u is invertible $\bmod p/d$. Since the degree of d is smaller than p, we may assume that the algorithm works correctly on F and d, computing a list $LIST_0$. Then, since p/d and d are relatively prime, the list $LIST = LIST_0 \cup \{(u^{-1} \bmod p/d, m, p/d)\}$ also has the desired properties. □

Lemma 19. *Algorithm* 17 *uses at most* $O(\deg(p)^2 + c_F \deg(p))$ *field operations.*

Proof. Let $p_1, \ldots, p_k$ be the polynomials which successively play the role of p in the course of the algorithm, where p_1 is the input polynomial p. Suppose that $\{\{v_i, m_i, q_i\}_i\}$ is the output of the algorithm, as i runs from 1 to k. If m is a monomial, let $r(m)$ be the number of monomials greater than m in the

ordering. Since the various polynomial operations $\bmod p$ carried out in each pass through the loop require $O(\deg(p)^2)$ coefficient operations, the number of operations T for the algorithm can be estimated as

$$T \leq C\left(\sum \deg(p_i)^2 (r(m_i) - r(m_{i-1}))\right),$$

where C is a constant. Since all $\deg(p_i) \leq \deg(p)$, we may write

$$T \leq C\left(\deg(p)\left(\sum \deg(p_i)(r(m_i) - r(m_{i-1}))\right)\right).$$

This sum can be rearranged to yield

$$T \leq C\left(\deg(p)\left(\sum \deg(q_i) r(m_i)\right)\right).$$

If $\deg(m) > 1$, there is a constant C' such that $r(m) < C' \deg(m)(\deg(m) - 1)$. Therefore, we may split up our sum into two parts:

$$T \leq \deg(p)\left(C_1\left(\sum_{i \in A} \deg(q_i)\right)\right) + C_2\left(\sum_{i \in B} \deg(q_i)\deg(m_i)(\deg(m_i) - 1)\right),$$

where A is the set of indices where $\deg(m_i) \leq 1$ and B is the set where $\deg(m_i) > 1$. The first sum is bounded by $\deg(p)$ since $\sum \deg(q_i) = \deg(p)$, and the second sum is bounded by $c_F \deg(p)$ by [4, p. 764]. It follows that the time T is bounded by $O(\deg(p)^2 + c_F \deg(p))$ as claimed. □

We are now ready to begin describing our standard basis algorithm. We will organize our data in a tree structure. This tree will be rooted at the top at a node called **root**. Every node will have a finite (possibly empty) set of child nodes, and every nonroot node will have a unique parent node.

Each node $\mathbf{n}$ of the tree will contain the following data:

1. A pointer to its parent node (denoted $\mathbf{n} \to p$).
2. A set of pointers to its child nodes.
3. A square free polynomial $q(T)$ (denoted $\mathbf{n} \to r$).
4. A vector representing an element of $A(\mathbf{n} \to r)$, denoted $\mathbf{n} \to vec$.
5. Two boolean fields (denoted $\mathbf{n} \to x$ and $\mathbf{n} \to y$).

We will write $x(\mathbf{n})$ and $y(\mathbf{n})$ to denote respectively the power of x and y appearing in the leading term of $\phi(\mathbf{n} \to vec, \mathbf{n} \to r) \in A(\mathbf{n} \to r)$. We will abbreviate and write:

$$\phi(\mathbf{n}) = \phi(\mathbf{n} \to vec, \mathbf{n} \to r),$$
$$c(\mathbf{n}) = c(\mathbf{n} \to vec, \mathbf{n} \to r),$$
$$l(\mathbf{n}) = l(\mathbf{n} \to vec, \mathbf{n} \to r).$$

The following definition characterizes the relationship between the various fields in a node and the various nodes in the tree.

Definition 9. Suppose p is a square free polynomial in $k[T]$ and F is a power series in $A(p)$. A tree T of nodes as described above will be called compatible with p if, whenever $\mathbf{n}$ is a nonroot node, and $\mathbf{c}$ is one of its children, the following myriad conditions are met:

1. $\mathbf{root} \to p = \mathbf{root}$.
2. $\mathbf{root} \to r = p(T)$.
3. $\mathbf{root} \to x = \mathbf{root} \to y = \mathbf{false}$.
4. $c(\mathbf{n}) = 1$.
5. $\mathbf{c} \to r$ divides $\mathbf{n} \to r$.
6. $\prod \mathbf{c} \to r = \mathbf{n} \to r$ where the product is taken over all children $\mathbf{c}$ of $\mathbf{n}$.
7. $l(\mathbf{c}) < l(\mathbf{n})$.
8. $\mathbf{n} \to x = \mathbf{true}$ if and only if either $x(\mathbf{n}) = 0$ or $x(\mathbf{n} \to p) = \mathbf{true}$.
9. $\mathbf{n} \to y = \mathbf{true}$ if and only if either $y(\mathbf{n}) = 0$ or $y(\mathbf{n} \to p) = \mathbf{true}$.

To understand all of this, associate to every node $\mathbf{n}$ the ring $D(\mathbf{n} \to r)$. Then the root node **root** corresponds to $D(p)$, and if $\mathbf{c}$ is a child node of **root**, then there is a reduction map $D(\mathbf{root}) \to D(\mathbf{c})$. Even more, by property 6, there is an isomorphism (by the Chinese remainder theorem)

$$D(\mathbf{n} \to r) \to \prod D(\mathbf{c} \to r),$$

where the product is taken over all children $\mathbf{c}$ of $\mathbf{n}$. Then $\mathbf{n} \to vec$ represents an element of $A(\mathbf{n} \to r)$ via the map $\phi(\cdot, \mathbf{n} \to r)$. Notice that our requirement that $c(\mathbf{n}) = 1$ implies that the image of the leading term $l(\mathbf{n})$ in $A(\mathbf{c} \to r)$ is the same as $l(\mathbf{n})$ for any descendant $\mathbf{c}$ of $\mathbf{n}$.

Definition 10. Suppose that T is a tree compatible with F and p. If $\mathbf{n}$ is a node in T, we derive three sets from $\mathbf{n}$.

1. Let $\Pi(\mathbf{n})$ denote the set of nodes on the path between $\mathbf{n}$ and **root**. In more formal terms, let $\Pi^*(\mathbf{n})$ be defined recursively by the rules:
   $\mathbf{n} \in \Pi^*(\mathbf{n})$.
   If $\mathbf{b} \in \Pi^*(\mathbf{n})$, then $\mathbf{b} \to p \in \Pi^*(\mathbf{n})$.
   Then $\Pi(\mathbf{n}) = \Pi^*(\mathbf{n}) - \mathbf{root}$.
2. When $\mathbf{n}$ is a node of T, we let
   $$B(\mathbf{n}) = \{\mathbf{b} \to vec: \mathbf{b} \in \Pi(\mathbf{n})\}.$$
3. Then, if $\mathbf{n}$ is a node of T, we define $I(\mathbf{n})$ to be the list of elements in $A(\mathbf{n} \to r)$ given by the rule
   $$I(\mathbf{n}) = \{\phi(\mathbf{b}) \bmod \mathbf{n} \to r: \mathbf{b} \in \Pi(\mathbf{n})\}.$$

The following definition summarizes the type of T we would like to construct.

Definition 11. A tree T, compatible with p and F, is called a tree of standard bases for F if, for each base node (that is, each node without children) $\mathbf{b}$, the set $I(\mathbf{b})$ is a standard basis for $I_F \bmod \mathbf{b} \to r$ over $D(\mathbf{b} \to r)$.

Notice that the polynomials $\mathbf{b} \to r$, as $\mathbf{b}$ runs through base nodes of T, yield a factorization of p into pairwise relatively prime polynomials. A tree of standard bases T gives us a standard basis $I(\mathbf{b})$ for the image of I_F in each factor ring $A(\mathbf{base} \to r)$. Since

$$A(p) = \prod A(\mathbf{base} \to r),$$

T partitions $A(p)$ into disjoint subrings so that in each subring we have a standard basis for the image, in that subring, of I_F.

The following procedure is the fundamental process by which a compatible tree T is extended downward.

Algorithm 20. *Tree extension.*

Input: A compatible tree T for a power series F and a square free polynomial p, a selected base node $\mathbf{n}$ in T, and a vector $\vec{h}$ representing an element of $A(\mathbf{n} \to r)$. To guarantee termination, $\vec{h}$ must satisfy an additional condition described in Lemma 22.

Output: A compatible tree $\overline{T}$, consisting of T with a set of child nodes attached to $\mathbf{n}$. In addition to the compatibility conditions, these new child nodes of $\mathbf{n}$, $\mathbf{c}_1, \ldots, \mathbf{c}_n$, satisfy

$$\mathbf{c}_j \to vec = \vec{R}(\vec{h}, I(\mathbf{n}), \mathbf{c}_j \to r).$$

Step 1: $\vec{r} := \vec{R}(\vec{h}, I(\mathbf{n}), \mathbf{n} \to r)$.

Step 2: $i := 1$, $p := \mathbf{n} \to r$.

Step 3: Create a new node $\mathbf{c}_i$. Set it up so that $\mathbf{c}_i$ is the ith child of $\mathbf{n}$.

Step 4: Compute polynomials a and b so that $ac(\vec{r}, p) + bp = d$, where $d = \gcd(p, c(\vec{r}, p))$.

Step 5: Compute an inverse $c(\vec{r}, p/d)^{-1}$ for $c(\vec{r}, p) \bmod p/d$.

Set $\mathbf{c}_i \to vec := c(\vec{r}, p/d)^{-1}\vec{r}$.

Set $\mathbf{c}_i \to r := p/d$.

Set $\mathbf{c}_i \to x := \mathbf{c}_i \to y := \mathbf{false}$.

If $x(\mathbf{c}_i \to vec) = 0$ or $\mathbf{n} \to x = 0$ then $\mathbf{c}_i \to x := \mathbf{true}$.

If $y(\mathbf{c}_i \to vec) = 0$ or $\mathbf{n} \to y = 0$ then $\mathbf{c}_i \to y := \mathbf{true}$.

Step 6: If $d = 1$, stop. Otherwise, $i := i + 1$, $p := d$, and $\vec{r} := \vec{R}(\vec{r}, I(\mathbf{n}), p)$.

Go to Step 3.

Lemma 21. *Suppose that Algorithm* 20 *halts when operating on* $\vec{h}$ *and* $\mathbf{n} \in T$. *Then it produces a tree with the claimed properties.*

Proof. Informally, the algorithm proceeds as follows. It reduces $\vec{h}$ by the set of ancestors of $\mathbf{n}$, working $\bmod p$. Calling the result $\vec{r}$, it attempts to invert the lead coefficient $c(\vec{r}, p)$ of $\vec{r} \bmod p$. However, this coefficient may not be relatively prime to p. Therefore, it computes $d = \gcd(c(\vec{r}, p), p)$. It can invert $c(\vec{r}, p) \bmod p/d$, so it does. Then it creates a node $\mathbf{c}_1$ with $\mathbf{c}_1 \to r = p/d$. It sets $\mathbf{c}_1 \to vec = c(\vec{r}, p/d)^{-1}\vec{r} \bmod p/d$. At this stage it has computed the

reduction of $\vec{h}$ by the ancestors of $\mathbf{n}$ in the component of $A(p)$ defined by $p/d = 0$. Now it must consider the places where $d = 0$. The leading term of $\vec{r} \bmod d$ is no longer the leading term $\bmod p$, since this lead coefficient reduces to zero $\bmod d$. Therefore, it sets $p := d$ and continues to reduce. Any new nodes $\mathbf{c}_i$ which the algorithm attaches will have $\mathbf{c}_i \to r$ dividing d. Inductively, this guarantees that all of the $\mathbf{c}_i \to r$ are pairwise relatively prime. Since the algorithm halts when $d = 1$, the product of the $\mathbf{c}_i \to r$ will be p. The remaining properties of the extended tree are clear. □

Lemma 22. *Suppose that Algorithm* 20 *fails to halt when operating on an element* $\vec{h}$ *and a node* $\mathbf{n}$. *Then there exists a polynomial* q *dividing* $\mathbf{n} \to r$ *and a list of power series* a_i *such that*

$$h \equiv \sum a_i f_i \pmod{q}$$

is a normal form for $h = \phi(\vec{h}, q)$ *in the factor ring* $A(q)$ *and the* f_i *are the elements of* $I(\mathbf{n})$.

Proof. Each pass through the loop in Algorithm 20 reduces the degree of the polynomial p, and the algorithm halts if p has degree zero. Therefore, the algorithm can only fail to halt if one of the reduction steps fails to halt. But by Lemma 13, this can only happen if the stated condition holds. □

Now suppose that T is a compatible tree. The following adjacency condition allows us to avoid unnecessary reduction steps in our standard basis algorithm.

Definition 12. Let $\mathbf{n}$ be a node in a compatible tree T. If $\mathbf{adj} \neq \mathbf{n}$ belongs to $\Pi(\mathbf{node})$, then we will say that $\mathbf{adj}$ is *adjacent* to $\mathbf{n}$ if $|x(\mathbf{adj}) - x(\mathbf{n})|$ is minimal among all elements of $\Pi(\mathbf{n})$. The set of nodes adjacent to $\mathbf{n}$ will be denoted $\mathrm{Adj}(\mathbf{n})$. (Notice that adjacency is a relation between the series represented by the nodes, and does not refer to their positions in the tree.)

This algorithm creates a tree of standard bases for F and p.

Algorithm 23. *Standard basis.*

Input: A square free polynomial $p(T)$ and a square free power series $F \in D(p)$ represented by a function in the manner described above.

Output: A tree of standard bases for F and p.

Step 0: Create a root node **root** for T and set $\mathbf{root} \to r := p$. Call Algorithm 17 on F_x and p, yielding a list *LIST*.

For each entry (a, m, q) of *LIST*:

Call Algorithm 17 on F_y and q, obtaining a list *LISTB*.

For each entry (b, n, w) in *LISTB*:

if $l(F_x, w) > l(F_y, w)$ then:

Create a child node **child** of **root** with:

$\mathbf{child} \to r := w$.

$\mathbf{child} \to vec := [a, 0] \bmod w$.

Call Algorithm 20 (Tree extension) on **child** and $[0, b]$.

otherwise:

create a child node **child** of **root** with:

child $\rightarrow r := w$.

child $\rightarrow vec := [0, b] \bmod w$.

Call Algorithm 20 (Tree extension) on **child** and $[a, 0]$.

Step 1: Until all base nodes **base** of T have **base** $\rightarrow x =$ **true** and **base** $\rightarrow y =$ **true**, do:

Let **base** be a base node of T with at least one flag set to false.

Let **prev** be an element of Adj(**base**).

Compute $\vec{h} := \vec{S}(\textbf{base} \rightarrow vec, \textbf{prev} \rightarrow vec, \textbf{base} \rightarrow r)$.

Carry out Algorithm 20 (Tree extension) on **node** and $\vec{h}$.

Step 2: Return T.

We now begin the process of proving that this procedure terminates, and delivers a tree of standard bases.

Lemma 24. *Step* 0 *of Algorithm* 23 *terminates.*

Proof. The calls to Algorithm 17 terminate since we assume F, F_x and F_y do not reduce to zero at any maximal ideal of $k[T]/p[[x, y]]$. If one of the calls to Algorithm 20 fails to terminate, then we conclude that there is a divisor q of p such that F_x and F_y have a common factor $\bmod q$. But this contradicts our assumption that F is square free. Therefore Step 0 terminates. □

Lemma 25. *At the completion of Step* 0, T *is strictly compatible.*

Proof. This is a consequence of the termination of Step 0 and the actions of Algorithm 20. □

Lemma 26. *Suppose, while executing Algorithm* 23, *we are at the top of the loop in Step* 1. *Then every node* $\mathbf{n}$ *of* T *has the following properties*:

1. *The set* Adj($\mathbf{n}$) *contains a unique element. In fact, either*

$$x(\mathbf{n}) = \max_{\Pi(\mathbf{n})} x(\mathbf{n}) \quad \textit{and} \quad y(\mathbf{n}) = \min_{\Pi(\mathbf{n})} y(\mathbf{n})$$

or

$$x(\mathbf{n}) = \min_{\Pi(\mathbf{n})} x(\mathbf{n}) \quad \textit{and} \quad y(\mathbf{n}) = \max_{\Pi(\mathbf{n})} y(\mathbf{n}).$$

2. *Let* $\mathbf{p}$ *and* $\mathbf{a}$ *be respectively the parent node of* $\mathbf{n}$ *and the sole member of* Adj($\mathbf{p}$). *Then*

$$l(\mathbf{n}) < \operatorname{lcm}(l(\mathbf{p}), l(\mathbf{a})).$$

3. *Let* $\mathbf{p}$ *be* $\mathbf{n} \rightarrow p$ *and let* $\mathbf{b}$ *be any element of* $\Pi(\mathbf{n})$. *Then we may write*

$$\vec{S}(\mathbf{p} \rightarrow vec, \mathbf{b} \rightarrow vec) \equiv \sum a(\mathbf{w})\mathbf{w} \rightarrow vec \quad (\bmod \mathbf{n} \rightarrow r),$$

where the sum is over $\Pi(\mathbf{n})$ *and where every monomial* m *occurring with nonzero coefficient* $\bmod \mathbf{n} \rightarrow r$ *in some* $a(\mathbf{w})$ *satisfies*

$$ml(\mathbf{w}) < \operatorname{lcm}(l(\mathbf{p}), l(\mathbf{b})).$$

4. *For any node* $\mathbf{n}$, $l(\mathbf{n})$ *is not in the span of the* $l(\mathbf{b})$ *as* $\mathbf{b}$ *runs through* $\Pi(\mathbf{n})$.

Proof. We proceed by induction. Let us consider the tree prior to the first execution of the loop in Step 1. Each base node **base** has one nonroot ancestor $\mathbf{p}$. Since $\mathbf{base} \to vec$ was obtained by reducing something smaller than $\mathbf{p} \to vec$ by $\mathbf{p} \to vec$, property 4 must hold. Properties 2 and 3 are satisfied vacuously. As for property 1, clearly each base node has at most one adjacent ancestor. The stronger claim holds as well. To see this, observe that if m and n are monomials such that $x(m) < x(n)$ and $y(m) < y(n)$, then obviously m divides n. But the reduction operation guarantees that $l(\mathbf{base})$ is not divisible by $l(\mathbf{base} \to p)$. Therefore, one of the two conditions in 1 holds.

Now suppose that T has the stated properties after some number of iterations of Step 1. Let **base** be a base node of T. We will check that after a pass through Step 1, T still has all of the desired properties. If all base nodes have both flags set, the algorithm halts and we are done. So suppose that $\mathbf{b}$ is a base node of T with at least one flag **false**. By the inductive hypothesis, $\mathrm{Adj}(\mathbf{b})$ contains one element; call this element $\mathbf{a}$. We must check that all of the nodes attached by Algorithm 20 to $\mathbf{b}$ have the claimed properties. Let $\mathbf{c}$ be such a node.

We know that

$$\mathbf{child} \to vec \equiv \vec{R}(\vec{S}(\mathbf{b} \to vec, \mathbf{a} \to vec), B(\mathbf{b}), \mathbf{c} \to r) \pmod{\mathbf{c} \to r}.$$

Since this R-operation halted, claim 4 must hold. Similarly, claim 2 is clear. As for claim 1, let

$$U_+ = \{\mathbf{b} \in \Pi(\mathbf{child}) : x(\mathbf{b}) > x(\mathbf{child})\},$$
$$U_- = \{\mathbf{b} \in \Pi(\mathbf{child}) : x(\mathbf{b}) < x(\mathbf{child})\}.$$

If either set is empty, we have settled claim 1; therefore, let us assume that both are nonempty. Let $\mathbf{u}_+$ be the element of U_+ such that $x(\mathbf{u}_+)$ is minimal, and let $\mathbf{u}_-$ be the element of U_- such that $x(\mathbf{u}_-)$ is maximal. Suppose for the sake of argument that $l(\mathbf{u}_+) > l(\mathbf{u}_-)$. Then $\mathbf{u}_-$ is the unique element of $\mathrm{Adj}(\mathbf{u}_+)$ by the structure of T and the inductive hypothesis. Since $l(\mathbf{c}) < \mathrm{lcm}(\mathbf{u}_+, \mathbf{u}_-)$ by claim 2, we can conclude that $l(\mathbf{child})$ is a multiple of $l(\mathbf{u}_-)$. This contradicts claim 4.

To finish, by properties of the R-operation, we may write

$$h = S(\mathbf{b}, \mathbf{a}) = \sum_{\mathbf{b} \in B} a(\mathbf{b})(\mathbf{b} \to vec) + (\mathbf{c} \to vec)$$

with $\sum a(\mathbf{b})(\mathbf{b} \to vec)$ in normal form. Let $\mathbf{b1}$ be any ancestor of $\mathbf{b}$, and let $g(\mathbf{b1}) = \vec{S}(\mathbf{b} \to vec, \mathbf{b1} \to vec)$. By claim 1, $\mathrm{lcm}(l(\mathbf{b}), l(\mathbf{b1}))$ must be divisible by $\mathrm{lcm}(l(\mathbf{b}), l(\mathbf{a}))$. This means that we can find monomials m_1 and m_2 so that

$$g(\mathbf{b1}) \equiv m_1 S(\mathbf{b} \to vec, \mathbf{a} \to vec) + m_2 S(\mathbf{a} \to vec, \mathbf{b1} \to vec) \pmod{\mathbf{c} \to r}.$$

Claim 3 follows easily from this expression by induction. □

Corollary 27. *The loop in Step* 1 *of Algorithm* 23 *is executed only finitely many times.*

Proof. This follows from the Hilbert basis theorem and property 4. □

Because of this corollary, we see that Algorithm 23 can fail to terminate only if one of the R-operations carried out in the calls to Algorithm 20 fails to terminate. As we will see, the assumption that F is a square free power series will guarantee that this cannot happen.

Theorem 28. *Suppose that* T *is a tree satisfying the conditions of Lemma* 26. *Let* $\mathbf{b}$ *be a base node of* T *and let* $\mathbf{a}$ *be the unique element of* $\mathrm{Adj}(\mathbf{b})$. *If there exists a nontrivial divisor* q *of* $\mathbf{b} \to r$ *such that*

$$\vec{R}(\vec{S}(\mathbf{b} \to vec, \mathbf{a} \to vec), B(\mathbf{b}), q) \equiv 0 \pmod{q},$$

then $I(\mathbf{b})$ *is a standard basis for the image* $I(q)$ *of* I_F *in the factor ring* $A(q)$ *of* $A(p)$.

Proof. Write $I(\mathbf{b}) = \{f_0, \ldots, f_n\}$. It follows from the inductive property 3 of Lemma 26 that $S(f_i, f_j)$ has the property required by the criterion in Theorem 10 for all pairs f_i and f_j with both i and j less than n. The additional hypothesis of this theorem guarantees that the criterion also holds for $S(f_n, g)$, where g is the unique element adjacent to $f_n = \phi(\mathbf{b} \to vec, \mathbf{b} \to r)$. However, as in the proof of Lemma 26, if f_k is a nonadjacent element, then there are monomials m_1 and m_2 so that

$$S(f_n, f_k) \equiv m_1 S(f_n, g) + m_2 S(g, f_k) \pmod{\mathbf{c} \to r}.$$

It is easy to see that this expression also satisfies the criterion of Theorem 10. Therefore, $I(\mathbf{n})$ is a standard basis as claimed. □

Corollary 29. *Algorithm* 23 *terminates.*

Proof. Suppose the algorithm does not terminate. Then, by Lemma 26 and our results on the termination of Algorithm 20, there is a node $\mathbf{n}$ such that the set $I(\mathbf{n})$ is a standard basis for the image of I_F in the factor ring $A(\mathbf{n} \to r)$. However, since either $\mathbf{n} \to x$ or $\mathbf{n} \to y$ is false for all ancestors of $\mathbf{n}$, then either no ancestral node of $\mathbf{n}$ has $x(\mathbf{n}) = 0$ or no ancestral node has $y(\mathbf{n}) = 0$. But this contradicts the assumption that I_F has finite codimension in $A(\mathbf{n} \to r)$, since it implies that no element of I_F has leading term a power of x or y. □

Corollary 30. *Algorithm* 23 *correctly computes a tree of standard bases for* I_F.

Proof. Upon termination, $I(\mathbf{b})$ for every base node $\mathbf{b}$ of T satisfies all of the properties listed in Lemma 26. Furthermore, there is an ancestor $\mathbf{n} \in \Pi(\mathbf{b})$ such that $x(\mathbf{b}) = 0$ and $y(\mathbf{n}) = 0$, or $x(\mathbf{n}) = 0$ and $y(\mathbf{b}) = 0$. Then it follows from Lemma 26, part 1, that, if $h = R(S(\mathbf{b} \to vec, \mathbf{n} \to vec), I(\mathbf{b}), \mathbf{b} \to r) \neq 0$, then $x(h) < 0$ or $y(h) < 0$. Therefore, we must have $h = 0$. But then it follows as in the proof of Lemma 29 that $I(\mathbf{b})$ satisfies the criterion of Theorem 10,

and therefore $I(\mathbf{b})$ is a standard basis for the image of I_F in $A(\mathbf{b} \to r)$. This means that we have constructed a tree of standard bases. □

We are finally able to give a simple method for computing the invariant δ_F.

Lemma 31. *Let T be the tree of standard bases computed by Algorithm 23 for a power series F in $A(p)$. For each base node $\mathbf{b}$ of T, let $\delta(\mathbf{b})$ be the δ invariant for the image of F in $A(\mathbf{b} \to r)$. Let $\mathbf{a}$ be the unique node adjacent to $\mathbf{b}$. Then*

$$\delta(\mathbf{b}) \leq \deg(\mathrm{lcm}(l(\mathbf{b} \to vec, \mathbf{a} \to vec))). \tag{3}$$

Proof. Let m be a monomial, and let

$$U_+ = \{\mathbf{n} \in \Pi(\mathbf{b}) : x(\mathbf{n}) \geq x(m)\},$$
$$U_- = \{\mathbf{n} \in \Pi(\mathbf{b}) : x(\mathbf{n}) \leq x(m)\}.$$

If m is not in the span of the leading terms of $B(\mathbf{b})$, then U_+ must be nonempty. U_- is nonempty, since there is a node $\mathbf{n}$ with $x(\mathbf{n}) = 0$. Let $\mathbf{u}_+$ be the element of U_+ with $x(\mathbf{u}_+)$ minimal. Let $\mathbf{u}_-$ be the element of U_- with $x(\mathbf{u}_-)$ maximal. If m is not divisible by $l(\mathbf{u}_-)$, then $y(m) \leq y(\mathbf{u}_-)$ and therefore $\deg(m) < \deg(\mathrm{lcm}(\mathbf{u}_-, \mathbf{u}_+))$. It is not hard to see, however, that $\mathbf{u}_+$ and $\mathbf{u}_-$ are adjacent in T, and therefore, by Lemma 26, we must have $\deg(m) < \alpha$, where α is the right-hand side of the inequality (3). We conclude that every monomial of degree at least α is in the span of the leading terms of $B(\mathbf{b})$. This implies easily that every monomial of degree at least α is in the ideal I_F and therefore that $\delta \leq \alpha$. □

2.5. Complexity of the standard basis algorithm. We will now estimate the complexity of the algorithm described above for computing a tree of standard bases. We will determine a bound on the number of field operations required to compute such a tree, assuming as always that F is known to be square free.

Let **node** be a base node of a tree of standard bases T constructed by Algorithm 23 from the power series F. Let

$$\Pi(\mathbf{node}) = \{\mathbf{node}_0, \ldots, \mathbf{node}_n = \mathbf{node}\}$$

and, to simplify notation, let $p_i(T) = \mathbf{node}_i \to r$ and $\vec{f}_i = \mathbf{node}_i \to vec$. In the notation of Lemma 14, let $d_i = d(\vec{f}_i)$, $r_i = r(\vec{f}_i)$, and $dl_i = r(\vec{f}_i)$.

Lemma 32. *For any base node* **node**, *and for all $0 \leq i \leq n$, we have $d_i \leq dl_i - dl_0$.*

Proof. As usual, we proceed by induction. We know that there is an element u of $k[T]$ such that $\vec{f}_0 = [u, 0]$. Then $\vec{f}_1 = \vec{R}(\vec{S}([0, 1], [u, 0]), p_1(T))$. Therefore, we may write $[0, 1] \equiv a[1, 0] + \vec{f}_1 \pmod{p_i(T)}$, and every monomial m in a satisfies $ml([1, 0], p_1) > l(\vec{f}_1, p_1)$. This means that $\deg(m) + dl_0 \leq dl_1$ and therefore $\deg(m) \leq dl_1 - dl_0$. We conclude that $d_1 \leq dl_1 - dl_0$ as claimed.

Now suppose $d_j \leq dl_j - dl_0$ for all $0 \leq j \leq i < n$. Then, assuming $\mathbf{node}_t$ is the unique element in $\mathrm{Adj}(\mathbf{node}_i)$, we have

$$\vec{S}(\vec{f}_i, \vec{f}_t) = \sum_{k=0}^{i} a_k \vec{f}_k + \vec{f}_{i+1}.$$

If m_k occurs in a_k, then we have $\deg(m_k) + dl_k \leq dl_{i+1}$, and therefore $\deg(m_k) \leq dl_{i+1} - dl_k$. This gives us $\deg(m_k) + d_k \leq dl_{i+1} - dl_k + d_k \leq dl_{i+1} - dl_0$ by induction. □

We now have enough information to get a bound on the complexity of computing the Milnor number.

Theorem 33. *Let F be a square free power series in $A(p)$ and let T be a tree of standard bases for I_F. Suppose that all terms of F of degree at most $\delta(F)$ have been computed. Then T can be computed by Algorithm 23 in less than $O((\delta(F)^4 + 1)\deg(p)^2)$ field operations.*

Proof. It is easy to see that our algorithm does not refer to the coefficients of any monomial in F of degree greater than $\delta(F)$. Therefore, we need only consider the time spent computing a tree of standard bases. The constant multiplicity algorithm requires $O(\deg(p)^2 + c_F \deg(p))$ operations. Since there is a constant C such that $c(F) \leq C\mu(F) \leq C\deg(p)\delta_F^2$, we will be done if we can show that the remainder of the algorithm requires at most $O(\delta(F)^4 \deg(p)^2)$ operations. Let us first determine the time necessary to extend the tree by one step from a node $\mathbf{n}$ in Step 1 of Algorithm 23. Let $\vec{h} = \mathbf{n} \to vec$ and let $\vec{g} = \mathbf{adj} \to vec$ where $\mathbf{adj}$ is the unique node adjacent to $\mathbf{n}$. Let $q = \mathbf{n} \to r$. Then we compute $\vec{s} = \vec{S}(\vec{h}, \vec{g}, q)$, and from Lemma 16 and Lemma 32 we conclude that

$$d(\vec{s}) \leq d(m),$$

where $m = \mathrm{lcm}(l(\vec{g}, q), l(\vec{h}, q))$. The time required for this computation is

$$T_0 = C_0(d(m)^2(r(\vec{s}) - r(m))\deg(q)^2).$$

The next step is to compute $\vec{f} = \vec{R}(\vec{s}, I(\mathbf{n}), q)$. Our estimates from Lemma 14 tell us that this requires

$$T_1 = C_1((r(\vec{f}) - r(\vec{s}))(dl(\vec{h}))^2 \deg(q)^2),$$

where we have used the fact that $dl(I(\mathbf{n})) = dl(\vec{h})$. We conclude from this that the time needed to attach the first child node to $\mathbf{n}$ is bounded by

$$O(\deg(q)^2(r(\vec{f}) - r(\vec{h}))d(\vec{f})^2).$$

In attaching the next child node, we continue to reduce $\vec{f} \bmod I(\mathbf{n})$, working now $\bmod q' = q/d$, where $d = \gcd(c(\vec{f}, q), q)$. Repeating the analysis above, we see that the total time to attach all the child nodes is given by a sum over the child nodes,

$$T(\mathbf{n}) = C_2 \sum (r(\vec{f}_i) - r(\vec{f}_{i-1}))dl(\vec{f}_i)^2 k_i^2,$$

where

$$\vec{f}_i = \mathbf{child}_i \to vec,$$
$$k_1 = \deg(q),$$
$$k_i = k_{i-1} - \deg(\mathbf{child}_i \to r), \qquad i \geq 2,$$

and by convention the subscript 0 refers to $\mathbf{n}$ itself. This sum may be rearranged to yield

$$\text{time} = C_3 \sum (r(\vec{f}_i) - r(\vec{f}_0))(\deg(\mathbf{child}_i \to r)^2 dl(\vec{f}_i)^2).$$

The total time for the algorithm is found by summing this time estimate over all nonbase nodes of the tree. Recall that

$$\delta(F) \geq \max_{\mathbf{n}}(dl(\mathbf{n})),$$
$$\deg(p) \geq \max_{\mathbf{n}} \deg(\mathbf{n} \to r).$$

Then our sum can be estimated by

$$\text{time} \leq C_4 \deg(p)\delta(F)^2 \sum (r(\mathbf{n}) - r(\mathbf{n} \to p)) \deg(\mathbf{n} \to r).$$

Let Q be the sum in the above expression. Let P be the set of nonbase nodes in T, and let us write $\mathbf{n} \mapsto \mathbf{p}$ when $\mathbf{p}$ is the parent node of $\mathbf{n}$. Then

$$\begin{aligned} Q &= \sum (r(\mathbf{n}) - r(\mathbf{n} \to p)) \deg(\mathbf{n} \to r) \\ &= \sum r(\mathbf{n}) \deg(\mathbf{n} \to r) - \sum r(\mathbf{n} \to p) \deg(\mathbf{n} \to r) \\ &= Q_1 - Q_2, \end{aligned}$$

where

$$Q_1 = \sum r(\mathbf{n}) \deg(\mathbf{n} \to r)$$

and

$$Q_2 = \sum_{\mathbf{p} \in T} r(\mathbf{p}) \sum_{\mathbf{n} \mapsto \mathbf{p}} \deg(\mathbf{n} \to r) - r(\mathbf{root}) \deg(\mathbf{root} \to r).$$

Using the fact that

$$\sum_{\mathbf{n} \mapsto \mathbf{p}} \deg(\mathbf{n} \to r) = \deg(\mathbf{p} \to r),$$

which is a property of compatible trees, we conclude that

$$Q = \sum_{T-P} r(\mathbf{n}) \deg(\mathbf{n} \to r) - r(\mathbf{root}) \deg(\mathbf{root} \to r).$$

Since, for each base node $\mathbf{n}$ we must have $r(\mathbf{n}) \leq C\delta_F^2$, we obtain the desired estimate. □

3. Resolving singularities

In this section we apply our results on computing deformations to the problem of blowing up plane curve singularities. Before tackling this problem, we need some algorithms for basic operations in rings with zero divisors.

3.1. Coefficient ring operations. We begin by describing some straightforward generalizations of standard algorithms for computing greatest common divisors, square free factorization, and primitive elements to polynomials over $D(p)$ when p is a square free, but possibly reducible, polynomial. We do not claim any great originality for these techniques, but we spell them out explicitly so that we can use them in careful complexity estimates.

Since $D(p)$ is abstractly isomorphic to a product of fields, $D(p)[x]$ is isomorphic to a product of PID's. Therefore, every ideal in $D(p)[x]$ is principal. The following generalization of the Euclidean algorithm, which splits $D(p)$ up into factor rings, computes the generator of an ideal (u, v) in $D(p)[x]$.

Algorithm 34. *Extended gcd (EGCD).*

Input: A square free polynomial $p \in k[T]$ and two polynomials

$$u = \sum_{i=0}^{m} u_{m-i} x^i, \qquad v = \sum_{i=0}^{n} v_{n-i} x^i,$$

in $D(p)[x] = k[T, x]/p(T)$.

Output: A list of triples $EGCD(u, v, p) = \{(q_j, a_j, b_j, w_j)\}$, where $q_j \in k[T]$ and each of a_j, b_j, and w_j are polynomials in $D(p)[x]$. This list satisfies:

1. The q_j are pairwise relatively prime, and $\prod q_j = p$.
2. For each j, we have the identities

$$u \equiv a_j w_j \pmod{q_j}, \qquad v \equiv b_j w_j \pmod{q_j}.$$

Step 0: If $v = 0$, then return $\{(p, 1, 0, u)\}$.

Step 1: Find polynomials a, b, and d so that $av_0 + bp = d$, where $d = \gcd(p, v_0)$.

$r := 0$.

Carry out the following operations, doing arithmetic $\bmod p/d$:

$t := \deg(u) - \deg(v)$.

while $t \geq 0$ do: (division)

$u := u - (u_0/v_0)x^t v$.

$r := r + (u_0/v_0)x^t$.

$t := \deg(u) - \deg(v)$.

$u' := v$.

$v' := u$.

$LIST_1 := EGCD(u', v', p/d)$.

$LISTU := \{\}$.

For each $(q, a, b, d) \in LIST_1$ do:

$LISTU := LISTU \cup \{(q, b + ar \bmod q, a, d)\}$.

$LISTZ := \{\}$.

If $\deg(d) > 0$, then

$$LISTZ := EGCD(u \bmod d, v \bmod d, d).$$

Step 3: Return $LISTU \cup LISTZ$.

Lemma 35. *Algorithm* 34 *does what it claims to.*

Proof. Suppose that q_1 and q_2 are relatively prime polynomials and that $q_1 q_2 = p$. Let u and v be polynomials in $D(p)[x]$. Then it is easy to see that if $LIST_1$ and $LIST_2$ satisfy the conditions required of $EGCD(u, v, q_1)$ and $EGCD(u, v, q_2)$, respectively, then $LIST = LIST_1 \cup LIST_2$ satisfies the conditions required of $EGCD(u, v, p)$. This fact is the basis for an inductive proof of the correctness of the algorithm.

Let us suppose first of all that $\deg(p) = 1$. Then Algorithm 34 reduces to the Euclidean algorithm, and therefore works properly. Now consider the situation when $\deg(p)$ is bigger than one. Clearly the algorithm works if $\deg(v) = 0$. So suppose that $\deg(v) > 0$. Notice that the lead coefficient v_0 of v is a unit in $D(p/d)$, where $\gcd(p, v_0) = d$. It follows that the ideal $(u', v') \subset D(p/d)$ is the same as the image of (u, v) in $D(p/d)$. Since the degree of v' is smaller than that of v, we may assume by induction that the algorithm computes $EGCD(u', v', p/d)$ correctly. This being the case, it is easy to see that the "change of coordinates" which gives $LISTU$ is a correct derivation of $EGCD(u, v, p/d)$ from $EGCD(u', v', p/d)$. Since p and p/d are relatively prime, the observation made above shows that $LISTU \cup LISTZ$ has the desired properties. □

Lemma 36. *Algorithm* 34 *computes* $EGCD(u, v, p)$ *using*

$$O(\deg(p)^2 \deg(u) \deg(v))$$

field operations.

Proof. Let $T(u, v, p)$ be the number of arithmetic operations necessary to carry out Algorithm 34 on polynomials u, v, and p. The division loop requires $O((\deg(u) - \deg(v)) \deg(v))$ operations in the ring $D(p/d)$, for a total of $T_{div} = O((\deg(u) - \deg(v)) \deg(v) \deg(p/d)^2)$ field operations. The shift operations $b \mapsto b + ar \bmod q$ require, each, at most $C \deg(v)(\deg(u) - \deg(v)) \deg(q)^2$. Since $\sum \deg(q) \leq \deg(p/d)$, we conclude that the time for each pass, T_{pass}, satisfies

$$T_{pass} \leq C \deg(v)(\deg(u) - \deg(v)) \deg(p/d)^2.$$

From the recursion we then obtain

$$T(u, v, p) = T_{pass} + T(u, v, d) + T(u', v', p/d).$$

It follows by induction that

$$T(u, v, p) \leq C(\deg(u) - \deg(v)) \deg(v) \deg(p/d)^2 + C \deg(u) \deg(v) \deg(d)^2 + C \deg(v)^2 \deg(p/d)^2,$$

which is less than $O(\deg(u) \deg(v) \deg(p)^2)$. □

As an application of this extended gcd algorithm, we consider the problem of "square free factorization" in a ring with zero divisors. Suppose that k is a

field of characteristic zero and $p(T)$ is a prime polynomial in $k[T]$. Letting $D(p) = k[T]/p$ as usual, suppose that q is a polynomial in $D(p)[x]$. Then, since $D(p)$ is a principal ideal domain, we can write $q = q_1 q_2^2 q_3^3 \cdots q_n^n$ where the q_i are relatively prime polynomials. The "square free" part $\mathrm{sqfr}(q)$ of q is then the polynomial $q_1 q_2 \cdots q_n$. We can compute the square free part $\mathrm{sqfr}(q)$ very simply since $\mathrm{sqfr}(q) = q/\gcd(q, \partial q/\partial x)$.

As we remarked earlier, every ideal in $D(p)[x]$ is principal. Therefore, it makes sense to speak of the square free part of an element $q \in D(p)[x]$. The following algorithm computes this square free part, by exploiting our extended gcd algorithm.

Algorithm 37. *Square free factorization.*

Input: A square free polynomial $p(T)$ with coefficients in a field k of characteristic zero. A polynomial $q(T, x)$ with coefficients in $D(p)$.

Output: A list of pairs (p_i, q_i) such that

1. The p_i are relatively prime, and $\prod p_i = p$.
2. q_i is the square free part of the image of q in the factor ring $D(p_i)$ of $D(p)$.

Step 1: Compute $LIST = EGCD(q, \partial q/\partial x, p)$.

Step 2: For each quadruple (p_i, a_i, b_i, d_i) in $LIST$, keep only (p_i, a_i).

Lemma 38. *Algorithm* 37 *works as claimed.*

Proof. This is an immediate consequence of the properties of *EGCD*, the Chinese remainder theorem, and the square free factorization algorithm over fields. □

We remark that this algorithm can certainly be modified to work over fields of characteristic $l \neq 0$ by separately handling cases where q is divisible by lth powers. In the interests of space and simplicity, we do not consider this case.

Our blowing up algorithm will depend on one more operation on coefficient rings $D(p)$. Suppose $f(W) \in D(p)[W]$ is a square free polynomial (perhaps computed by the square free factorization algorithm). Then we can find a primitive element for the ring $D(p)[W]/f$—that is, we can find a polynomial $q(T)$ and an isomorphism

$$D(p)[W]/f \to D(q).$$

Again, our method is a generalization of the standard method for finding primitive elements in field extensions.

Algorithm 39. *Primitive elements.*

Input: A square free polynomial $p(T)$ over k, and a square free polynomial $f(T, W)$ over $D(p)$.

Output: A list of triples of polynomials $\{q_i(T), u_{q_i}(T), v_i(T)\}$ such that the

map

$$k[T, W]/(p(T), f(T, W)) \to \prod k[T]/q_i(T),$$
$$T \mapsto (u_i(T)),$$
$$W \mapsto (v_i(T))$$

is an isomorphism.

Step 0: $c := 0$. $RESULT := \{\}$.

Step 1: Let $g(T, Z) := f(T, Z - cT)$ in $D(p)[W]$.

Compute the resultant $r(Z)$ of $g(T, Z)$ with $p(T)$, eliminating T.

If $r(Z)$ is not square free over k, increment c and repeat Step 1.

Step 2: Let $LIST := EGCD(r, p, g)$. Note that, in this invocation of $EGCD$, Z plays the role of T and T plays the role of x.

For each quadruple (p_i, a_i, b_i, d_i) in LIST, do:

Write $d_i(Z, T) = a(Z)T - b(Z)$.

Set $u(T) := b(T)/a(T) \bmod p_i(T)$.

Set $RESULT := \{p_i(T), u(T), Z - cu(T)\} \cup RESULT$.

Step 3: Return $RESULT$.

Lemma 40. *Algorithm* 39 *terminates, returning a list* (q_i, u_i, v_i) *with the claimed properties.*

Proof. First, Step 1 terminates. To see this, interpret the resultant $r(Z)$ computed in each pass through Step 1 as the projection of the points in the (T, W) plane determined by the equations $p(T) = 0$ and $f(T, W) = 0$ onto the Z-line, where $Z = W + cT$. Since the ground field k is infinite and the equations $p(T) = 0$ and $f(T, W) = 0$ determine a set of points with multiplicity one, there must be a line such that the projection consists of distinct points. Therefore, we eventually reach a c such that the polynomial $r(Z)$ is square free. It follows that the map

$$k[T, Z]/(p, r, g) \to k[T, W]/(p, f),$$
$$T \mapsto T,$$
$$Z \mapsto W - cT$$

is an isomorphism. However, since the dimension of $k[Z]/r(Z)$ is equal to that of $k[T, W]/(p, f)$, the ideal (p, g) in $k[Z, T]/r(Z)$ must be of the form $(aT - b)$ where a is a unit. Therefore, we have an isomorphism

$$k[Z, T]/(r, p, g) \to k[Z]/r,$$
$$Z \mapsto Z,$$
$$T \mapsto b(Z)/a(Z).$$

Recalling that we constructed r by setting $Z = W + cT$, we see that $W = Z - cb(Z)/a(Z)$ under this isomorphism. In Step 2, the call to $EGCD$ computes the generator $aT - b$ for the ideal (p, g), possibly splitting up the ring r in the process. Then, for each factor, we compute the $b(Z)/a(Z)$ and $W =$

$Z - cb(Z)/a(Z)$. Thus, the list *RESULT* that we eventually return has the desired properties. □

We conclude this subsection with a discussion of the complexity of the primitive element algorithm.

Lemma 41. *Let $p(T)$ and $f(T, W)$ be polynomials such that $p(T)$ is square free over k and $f(T, W)$ is square free over $D(p)$. Let*

$$n = \dim k[T, W]/(p, f).$$

Then the primitive element algorithm requires $O(n^4)$ field operations.

Proof. Let n_1 be the degree of $p(T)$ as a polynomial in T, and n_2 be the degree of f as a polynomial in W. Then $n = n_1 n_2$. Each substitution of $Z - cT$ for W in Step 1 requires $O(n_2^2)$ operations in the ring $D(p)$, for a total of $O(n^2)$ field operations. Each resultant can also be calculated in $O(n^2)$ operations. The square free check requires $O(n^2)$ operations since it amounts to computing the gcd of r with r' over k. In addition, this loop need be executed at most $O(n^2)$ times. To see this, recall that $g(T, Z)$ will not be square free if the line $W - cT = 0$ intersects two or more of the points satisfying $p(T) = 0$ and $f(T, W) = 0$. Since there are n points, there are less than n^2 lines meeting two or more of them. Thus, Step 1 requires $O(n^4)$ operations.

In Step 2, the *EGCD* operation requires

$$O(\deg(r)^2 \deg(p) \deg(g))$$

operations, and this is simply $O(n^3)$. The various arithmetic operations in the quotient rings $D(p_i)$ produced by *EGCD* are all $O(\deg(p_i)^2)$, and since the sum of these degrees is n, the total time for them is $O(n^2)$. It follows that the grand total number of operations for this algorithm is $O(n^4)$. □

We point out that this result is very much a worst case. If, in Step 1, a random c is picked and used to change coordinates, the chances are excellent that the loop in Step 1 will only need to be executed once. Therefore, this algorithm is "generically" an $O(n^3)$ algorithm.

3.2. The blowing up algorithm. For a detailed description of the process of blowing up, we refer the reader to the algebraic geometry literature, and in particular to [4, Chapter 8]. However, we will walk through the process a step at a time, describing it algorithmically as we go.

We begin with a polynomial $F \in (x, y)D(p)[[x, y]] = (x, y)A(p)$. Assume that the multiplicity of F at each maximal ideal of $A(p)$ is m. Let $\tilde{F}_1 \in A(p)[y/x]$ and $\tilde{F}_2 \in A(p)[x/y]$ be defined as follows:

$$\begin{aligned} \tilde{F}_1(x, y/x) &= x^{-m} F(x, x(y/x)), \\ \tilde{F}_2(x/y, y) &= y^{-m} F(y(x/y), y). \end{aligned} \tag{4}$$

Let B_1 and B_2 be the rings

$$B_1 = A(p)[y/x]/\tilde{F}_1, \qquad B_2 = A(p)[x/y]/\tilde{F}_2.$$

Then the blowing up $\tilde{X}$ of $X = \operatorname{Spec} A(p)/F$ at the ideal (x, y) is the scheme constructed by gluing $\operatorname{Spec} B_1$ to $\operatorname{Spec} B_2$, identifying $\operatorname{Spec} B_1[x/y]$ with $\operatorname{Spec} B_2[y/x]$ by means of the obvious isomorphism between these rings.

It is easy to see that $\operatorname{Spec} B_1$ covers all of $\tilde{X}$ except possibly for the point on $\operatorname{Spec} B_2$ where $x/y = 0$. The function x/y is not a unit in B_2 precisely when $\tilde{F}_2 \in (y, x/y)$. If this is the case, a standard theorem [19, p. 175] shows that the localization B_∞ of B_2 at the ideal $(x/y, y)$ is a direct factor of B_2, and is isomorphic in the obvious way to $A(p)[[y, x/y]]/\tilde{F}_2$. Of course, if x/y is a unit in B_2, then B_∞ is the zero ring. If we write $B_2 = B_2' \times B_\infty$, the maximal ideals of B_∞ are the "points at infinity" on the exceptional divisor, and the maximal ideals of B_2' are covered by $\operatorname{Spec} B_1$. We conclude from this discussion and the definition that the blowup of $A(p)/F$ is the ring $B_1 \times B_\infty$.

Notice that B_∞ is in the form $A(q)/G$ for a polynomial q and power series G. Our computation of the blow up of X amounts to finding an isomorphism of $B_1 = B$ with a ring of the form $\prod A(q_i)/G_i$. The following lemma explains how this is done.

Lemma 42. *Let* $B = A(p)[y/x]/\tilde{F}_1(x, y/x)$. *Let* $f(W) = \tilde{F}_1(0, W)$, *and let* (p_i, f_i) *be the pairs returned by the square free factorization algorithm applied to* $p(T)$ *and* $f(W)$ *over* k. *Then there is an isomorphism* α:

$$\begin{aligned} B &\xrightarrow{\alpha} \prod (D(p_i)[W_i]/f_i)[[x, y_i]]/\tilde{F}_1(x, y_i + W_i), \\ x &\mapsto x, \\ y/x &\mapsto y_i + W_i. \end{aligned} \tag{5}$$

Proof. Every maximal ideal of B must contain x. Since

$$B/xB = A(p)[y/x]/(x, \tilde{F}_1) = D(p)[y/x]/f(y/x),$$

we see that the radical of the ideal $(x, f(y/x))$ is the Jacobson radical of B_1. The square free factorization algorithm returns a list of polynomials (p_i, f_i) such that f_i generates the radical of the ideal $fD(p_i) \subset D(p)$. Let B_i be the factor of B corresponding to p_i. Then $(x, f_i(y/x))$ is the Jacobson radical of B_i.

Since $f_i(y/x)$ is in the Jacobson radical of B_i, B_i contains an approximate solution to the equation $f_i(W_i) = 0$. Since $f_i(y/x)$ is square free $\bmod x$, we know that $f_i'(y/x)$ is a unit in B_i. It follows by Hensel's lemma that there exists a $W_i \in B_i$ with $W_i \equiv y/x \bmod (x, f_i(y/x))$ and $f_i(W_i) = 0$. It is easy to see that $(x, y/x - W_i) = (x, f_i(y/x))$. We conclude that there is a surjection

$$D(p_i)[W_i]/f_i[[x, y/x - W_i]] \to B_i.$$

It then follows easily that the kernel of this surjection is generated by the form $\tilde{F}_1(x, (y/x - W_i) + W_i)$. Since $B = \prod B_i$, the lemma follows. □

The lemma shows us how to split $A(p)[y/x]/\tilde{F}_1$ up into a product of rings of the form $D(p)[W]/(f_i(W))[[x, y]]/G$. As a final stage in finding a standard form for the blowup of $A(p)/F$, we convert the coefficient rings $D(p_i)[W_i]/f_i$ into the form $D(h_i)$. This is the problem of computing a primitive element, which we have discussed in Subsection 3.1. As we saw there, we may find polynomials q_i, u_i, and v_i such that the map β_i

$$
\begin{aligned}
k[T, W]/(p_i(T), f_i(T, W))[[x, y/w - W_i]] &\xrightarrow{\beta} k[T']/q_i[[x, y/x - v_i]], \\
T &\mapsto u_i(T'), \\
W &\mapsto v_i(T')
\end{aligned} \tag{6}
$$

is an isomorphism. If we let $\gamma_i = \beta_i \circ \alpha$, then we obtain an isomorphism

$$
\begin{aligned}
A(p)[y/x]/\tilde{F}_1 &\xrightarrow{\prod \gamma_i} \prod A(q_i)[[x, y_i]]/G_i, \\
x &\mapsto (x)_i, \\
y/x &\mapsto (y_i + v_i)_i, \\
T &\mapsto u_i, \\
G_i(x, y_i) &= \tilde{F}_i(x, y_i + v_i).
\end{aligned} \tag{7}
$$

The maps γ_i, which we would like to compute, are determined by quintuples $(var, G_i, q_i, u_i, v_i)$, where G_i, q_i, u_i, and v_i define a map as in (7), and "*var*" is either x or y. We use this piece of information to identify the equation of the exceptional divisor in the image ring determined by the other data. Thus, if *var* is x, then the remaining data describes a map exactly as in (7), but if *var* is y, then we interpret that data as giving a map from $A(p)[x/y]/\tilde{F}_2$ to $A(q_i)[[x_i, y]]$ defined as in (7) but with the roles of x an y interchanged. This convention allows us to handle the points at infinity without special consideration. Indeed, the ring B_∞ discussed above, which is the coordinate ring of the points at infinity, is determined by the data $(y, \tilde{F}_2, p(T), 0, T)$.

We represent the blowup of $A(p)/F$ at (x, y) by supplying a list of quintuples $\{v_i, G_i, q_i, u_i, v_i\}$. The product of the rings $A(q_i)/G_i$ constructed from this data will be identified with the coordinate ring of the blowup of $A(p)/F$ via the product of the isomorphisms constructed from the u_i, v_i as in (7).

In terms of this data structure, the blowing up algorithm takes the following form.

Algorithm 43. *Blowing up.*

Input: A square free polynomial $p(T)$ and a polynomial

$$F(x, y) = \sum a_{ij} x^i y^j$$

with coefficients a_{ij} in $D(p)$. We assume that F is square free and that the leading coefficient $c(F)$ of F (relative to the lexicographic within degree ordering) is invertible in $D(p)$.

Output: A list of quintuples representing the blowup of $A(p)/F$ as described above.

Step 1: $m := \text{mult}_{(x,y)}(F)$.

$\tilde{F}_1 := x^{-m}F(x, xy)$.

$f(W) := F_1(0, W)$.

Step 2: Apply the square free factorization algorithm to the pair $f(W)$ and $p(T)$. Let $LIST := \{(f_i, p_i)\}$ be the result.

Step 3: Set $RESULT = \{\}$. For each pair $(f_i(W, T), p_i(T)) \in LIST$ do:

Compute a primitive element Z for the ring $k[T, W]/(p_i(T), f_i(W, T))$.

Let *PRIMLIST* be the resulting list $(q_{ij}(Z), u_{ij}(Z), v_{ij}(Z))$.

For each element of *PRIMLIST*, do:

Let G_{ij} be the polynomial obtained from $\tilde{F}_1$ by substituting:

$$T \mapsto v_{ij}(T),$$
$$y \mapsto y + u_{ij}(T),$$

computing $\bmod q_{ij}(T)$.

Set $RESULT := \bigcup_{i,j}\{(x, G_{ij}, q_{ij}, u_{ij}, v_{ij})\} \cup RESULT$.

Step 4: If $a_{0,m} = 0$ then:

Let $G := y^{-m}F(xy, y)$.

Set $RESULT := \{(y, G, p, 0, T)\} \cup RESULT$.

Step 5: Return *RESULT*.

Theorem 44. *The blowing up algorithm works as claimed.*

Proof. First of all, we point out that our assumption that the leading coefficient of F is a unit in $D(p)$ guarantees that the multiplicity of F at all maximal ideals of $A(p)$ is the same. Therefore, $\tilde{F}_1$ is indeed the equation of the strict transform of F. The correctness of the remainder of Steps 2 and 3 follows from the discussion preceding the algorithm. Step 4 computes the coordinate ring of the points at infinity. □

It is now a straightforward matter to determine the complexity of the blowing up operation. We write the complexity in a way that will be convenient when we consider the complexity of the resolution problem.

Lemma 45. *Let m be the multiplicity of F at the ideal (x, y), let r be the degree of the coefficient polynomial $p(T)$, and let d be the degree of the polynomial F representing the singularity. Then Algorithm* 43 *requires $O(m^4r^4 + r^2m^2d^4)$ field operations.*

Proof. Step 1 requires time proportional to d^2. Step 2, a square free factorization, requires r^2m^2 operations. The primitive element calculations require $O(\sum r_i^4 m^4)$, where r_i is the degree of the p_i returned by the square free algorithm. Since $\sum r_i = r$, this sum is $O(r^4m^4)$. Let r_{ij} be the degree of q_{ij}. Then the coordinate changes in Step 3 require $\sum r_{ij}^2 d^4$ field operations. But

$\sum r_{ij}^2 < r^2m^2$ since $\sum r_{ij} < rm$. Therefore, the coordinate changes require $O(r^2m^2d^4)$ operations. Combining these totals yields the desired result. □

3.3. Resolution of singularities. We have finally assembled all of the necessary apparatus for the resolution of curve singularities. We will combine the blowing up procedure with the deformation computation to accomplish the resolution.

Algorithm 46. *Resolution.*

Input: A square free polynomial p and a square free power series F in $D(P)$ represented by a function as discussed in §2.

Output: A list of lists representing the resolution of the singularity of F at the origin, in a manner discussed more fully below.

Step 1: Apply the standard basis algorithm to F and p. From the resulting tree of standard bases, we obtain a factorization $p = \prod p_i(T)$ such that F has constant multiplicity (and constant Milnor number) on each factor $A(p_i)[[x, y]]$ of $A(p)$. We also obtain from this computation the invariants $\delta_i(F)$, that is, the smallest power of (x, y) such that $(x, y)^{\delta_i} \subset (F_x, F_y)$ in $A(p_i)$. For each i, let G_i be the polynomial obtained by dropping all terms of F_i of total degree greater than or equal to $2\delta_i + 1$.

Step 2: Let $RESOLUTION = \{\}$.

For each i, do the following: $RES_i = \{\}$.

If $\text{mult}(G_i) > 1$, then

Compute the blowup of G_i using Algorithm 43. Let *NEWPTS* be the resulting list.

For each quintuple $(w_{ij}, G_{ij}, q_{ij}, u_{ij}, v_{ij})$ in *NEWPTS*, do:

Call this algorithm recursively on G_{ij} and q_{ij}. Let RES_{ij} be the resulting list.

Set

$$RES_i := \{(w_{ij}, G_{ij}, q_{ij}, u_{ij}, v_{ij}), RES_{ij}\} \cup RES_i.$$
$$RESOLUTION := RESOLUTION \cup \{p_i, RES_i\}.$$

Step 3: Return *RESOLUTION*.

Lemma 47. *The resolution algorithm terminates.*

Proof. The algorithm replaces p and F by polynomials p_i and G_i. The p_i and G_i are such that $A(p)$ is isomorphic to $\prod A(p_i)$ and, by Tougeron's lemma (Lemma 4 and Lemma 5), there is an analytic change of coordinates in $A(p_i)$ for each i carrying the image F_i of F in $A(p_i)$ to the polynomial G_i. It suffices then to prove that the algorithm terminates for each p_i and G_i. If $m = 1$, this is clear. If $m > 1$, then the blowing up step replaces p_i and G_i by a set p_{ij} and G_{ij} such that the conductors of the rings $A(p_{ij})/G_{ij}$ have smaller index. It follows by induction on this index that the algorithm terminates. □

The list *RESOLUTION* produced by the algorithm consists of entries of the form $\{p_i, LIST\}$, where p_i is a factor of p. The list *LIST* is null if the points

in $A(p)$ defined by the zeros of p_t all have multiplicity 1. Otherwise, the list *LIST* consists of pairs of the form $\{(w, G, q, u, v), RESLIST\}$, where (w, G, q, u, v) is a quintuple defining a factor of the blowing up of $A(p_t)/G_i$ according to (7). *RESLIST*, which is of the same form as *RESOLUTION*, describes the resolution of the infinitely near points described by the quintuple.

To construct the blowing up of F at (x, y) from the list resolution, we proceed as follows. Let $\{p_t, LIST\}$ be an element of *RESOLUTION*. If *LIST* is empty, then F is nonsingular at the zeros of p_t. Otherwise, let $\{(w, G, q, u, v), RESLIST\}$ be an element of *LIST*. Then, by recursion, *RESLIST* describes the resolution of the ring $A(q)/G$. Define a map

$$\rho: A(p_i)/F \to A(q)/G$$

so that ρ is the composition of the analytic isomorphism constructed by Tougeron's lemma and the map determined by the quintuple (w, G, q, u, v). The product of the ρ defined in this way, for all elements of *LIST*, combined with the data constructed recursively from *RESLIST*, gives the resolution of $A(p_i)/F$. The product of these resolutions over all elements in *RESOLUTION* gives the complete resolution of $A(p)/F$.

Since we are using the conductor to measure complexity, we need the following result from the general theory.

Theorem 48 (see [4, p. 764]). *Let $F \in A(p)$ be a square free power series such that the multiplicity of F at all maximal ideals of $A(p)$ is m. Let $\{P_1, \ldots, P_n\}$ be the set of infinitely near points to the origin, and let c_i be the conductor of the strict transform of F at P_i. Then*

$$\mathrm{c}_F = \sum \mathrm{c}_i + \deg(p)m(m-1).$$

Theorem 49. *Let p be a square free polynomial over k, F a square free power series in $(k[T]/p)[[x, y]]$, and let c be the index of the conductor of F. Then the resolution algorithm computes the resolution of F using at most $O(\deg(p)^2(1+\mathrm{c}^6))$ field operations.*

Proof. We proceed by induction on the number c. If F is nonsingular at all points of $\operatorname{Spec} A(p)$, then, since $\delta_F = \mathrm{c} = 0$, the Milnor number algorithm will establish this in $O(\deg(p)^2)$ operations. Otherwise, the algorithm will require $O(\deg(p)^2\delta_F^4 + \deg(p)^2)$ operations to split F up into a set of pairs (p_t, G_i). Since $\delta_F \le \mathrm{c}$, we may estimate this by $O(\deg(p)^2(\mathrm{c}^4+1))$. The algorithm then considers those of the (p_i, G_i) where G_i has multiplicity greater than one. Here the algorithm requires

$$\sum_i O(m_i^4 \deg(p_i)^4 + m_i^2 \deg(p_t)^2 \delta_i^4)$$

operations, where m_i is the multiplicity of G_i and δ_i is δ_{G_i}. Since $m_i > 1$, we may estimate $\deg(p_i)m_i^2 < O(\mathrm{c})$, and therefore this sum is bounded by

$\sum O(\deg(p_i)^2 c_i^2 + c_i^5 \deg(p_i))$. Since each $c_i < c$ and $\sum \deg(p_i) < \deg(p)$, we may bound the first steps of the algorithm by

$$O(\deg(p)^2(1 + c^2 + c^4 + c^5)) < O(\deg(p)^2(1 + c)^5)$$

field operations. By the inductive hypothesis, the recursive computation applied to the p_{ij} and G_{ij} requires at most $\sum_{i,j} O(\deg(p_{ij})^2(1 + c_{ij})^6)$ operations. Since $\deg(p_{ij}) < \deg(p)$, this is bounded by $\deg(p)^2 \sum(1 + c_{ij})^6$. However, since all of the points represented by the pairs (p_{ij}, G_{ij}) were obtained by blowing up (p_i, G_i), by Theorem 48 we must have $\sum_j 1 + c_{ij} \leq c_i$, from which we conclude that our time is bounded by $O(\deg(p)^2(\sum_i c_i)^6)$, which in turn is at most $O(\deg(p)^2 c^6)$. Therefore, the total time required is

$$O(\deg(p)^2 c^6) + O(\deg(p)^2(1 + c^5))$$

operations. For a suitable choice of constants, this is bounded by

$$O(\deg(p)^2(1 + c^6))$$

as claimed. □

We make no pretense to claiming that this bound is sharp. The primary significance of the result is the conclusion that the complexity of resolution can be measured by local data—namely the index of the conductor and the number of connected components of the singularity—and that the complexity is polynomial. The important problem of understanding the exact complexity of the resolution of singularities is completely open.

3.4. Computing the conductor. Our final results show how the conductor of a singularity can be extracted from the resolution data generated by the resolution algorithm. We apply the classical method of adjoints, described in the following theorem, to compute the conductor ideal.

Definition 13. Let p be a square free polynomial over k and let F be a square free power series in $A(p)$. Then an element f of $A(p)$ is an adjoint for F if, at each infinitely near point to F of multiplicity m, f vanishes to order $m - 1$.

Theorem 50 (see [4, p. 797ff]). *The conductor ideal of $A(p)/F$ is the ideal in $A(p)/F$ generated by the adjoints.*

We also make use of the following result.

Lemma 51 (see [4, p. 797ff]). *Let p and F be as above. Then $\partial F/\partial x$ and $\partial F/\partial y$ belong to the conductor of F.*

Since the ideal $I_F = (F_x, F_y)$ has finite index in $A(p)$, we can describe the conductor by computing the kernel of the linear map

$$A(p)/(F_x, F_y) \to A(p)/c. \tag{8}$$

expressed in terms of the basis of $A(p)/(F_x, F_y)$ given by the monomials. The following procedure shows how to extract this information from the list *RESOLUTION* which is generated by the resolution algorithm.

Algorithm 52. *Conductor.*

Input: A square free polynomial p, a square free power series F, and the output *RESOLUTION* from the resolution algorithm.

Output: A system of linear equations describing a linear subspace of $A(p)/I_F$; this subspace is the kernel of the map in (8).

Step 0: For each $\{q, LIST\}$ in *RESOLUTION*, do:

Construct a "generic" polynomial

$$E = \sum_{i,j \in Q} \left(\sum_{k=0}^{\deg(q)-1} a_{ijk} T^k \right) x^i y^j$$

so that a_{ijk} are unknowns and Q is the set of i, j such that $x^i y^j$ is not in $I_F \subset A(q)$. (Q is just the set of monomials which are not divisible by the leading terms of elements of the standard basis for I_F.)

Call the conductor-reduction algorithm below on E and $\{q, LIST\}$.

Return the resulting list of linear equations in the a_{ijk}.

Here is the conductor-reduction algorithm we refer to:

Algorithm 53. *Conductor-reduction.*

Input: A polynomial E in x, y, and T, with coefficients that are linear functions in a_{ijk}; and an element $\{p, LIST\}$ from the resolution list *RESOLUTION* for a power series F.

Output: A list of linear equations in the a_{ijk}.

Step 0: Set $EQLIST := \{\}$.

Step 1: Let m be the multiplicity of the point represented by $\{q, LIST\}$. If $m \leq 1$ (in which case *LIST* is null) then return. Otherwise, for each term $e(ijk)T^k x^i y^j$ occurring in E with $i+j < m-1$, add the equation $e(ijk) = 0$ to the list *EQLIST* and delete this term from E.

Step 2: For each element $\{(w, G, q, u, v), RES\}$ in *LIST*, make the change of coordinates represented by the quintuple in E. (See (7).)

Set $E := w^{1-m}E$ (this is an exact division).

For each $\{p, LIST\}$ reduce $E \bmod p$ and call this routine recursively on $E \bmod p$ and $\{p, LIST\}$. Append the resulting list to *EQLIST*.

Step 3: Return *EQLIST*.

Lemma 54. *The list of linear equations returned by the conductor algorithm defines the conductor of F, in the sense that if E is a polynomial $\sum a_{ijk} T^k x^i y^j$ with $a_{ijk} \in k$, then the a_{ijk} satisfy the equations in EQLIST if and only if E belongs to the conductor of F. Furthermore, the equations in EQLIST are independent.*

Proof. First of all, the algorithm actually computes the conductor of the ring $A(p)/G$, where G is the polynomial obtained by dropping terms in F of degree larger than $2\delta_F + 1$. However, by Lemma 5, the analytic isomorphism τ between $A(p)/G$ and $A(p)/F$ induces the identity on $A(p)/I_G = A(p)/I_F$. Therefore, we may replace F by G without loss of information.

Now we proceed by induction on the conductor of F. If F has multiplicity 1, then the algorithm will return an empty list, which is correct. Otherwise, the algorithm will add the set of linear equations in the coefficients of E to *EQLIST* which are equivalent to the condition "E vanishes to order at least $m-1$ at (x, y)." It then makes the necessary changes of coordinates to find the equation of E near each infinitely near point. By induction, we may conclude that the recursive call will adjoin to *EQLIST* the conditions which force the blowup $\tilde{E}$ of E to belong to the conductors of all the infinitely near points (notice that here, too, we are applying Tougeron's lemma). It follows by the theorem on adjoints that this combined list defines the conductor. The independence of the equations in *EQLIST* follows from Noether's theorem that the vanishing conditions which define the conductor are independent. □

Lemma 55. *The computation of the conductor requires* $O(\deg(p)^2 c^6)$ *field operations.*

Proof. The algorithm adjoins some conditions to the list, then makes a change of coordinates of a polynomial of degree $O(\delta_F)$, in two variables, with coefficients that are polynomials in T of degree at most $\deg(p)$. These polynomials in turn have coefficients that are unknowns. The total time for these changes of coordinates is therefore $O(\delta_F^4 \deg(p)^2 c)$, where we have used the fact that the number of unknown coefficients is bounded by the Milnor number of F, which, by Lemma 3, is in turn bounded by a multiple of the conductor. Using the estimate $\delta_F \leq c$ from Lemma 3, we find that the changes of coordinates require $O(c^5 \deg(p)^2)$. It then follows by a reasoning similar to that used to derive the complexity of the resolution algorithm that the total time for the algorithm, including the recursion, is $O(\deg(p)^2 c^6)$ as desired. □

Bibliography

1. M. Artin, *Deformations of singularities*, Tata Institute of Fundamental Research, Bombay, 1976.
2. D. Bayer, *The division algorithm and the Hilbert scheme*, Ph.D. thesis, Harvard University, 1982.
3. T. Berry, *On Coates' algorithm*, SIGSAM Bull. **17** (1983), 12–17.
4. E. Brieskorn and H. Knorrer, *Ebene algebraische Kurven*, Birkhäuser, Boston, 1981.
5. B. Buchberger, *A criterion for detecting unnecessary reductions in the construction of Gröbner bases*, Lecture Notes in Comput. Sci., vol. 72, Springer, 1979, pp. 3–21.
6. B. Buchberger, G. E. Collins, and R. Loos (editors), *Computer algebra: Symbolic and algebraic computation*, Springer, 1983.
7. D. V. Chudnovsky and G. V. Chudnovsky, *On expansion of algebraic functions in power and Puiseux series.* I, J. Complexity **2** (1986), 271–294.

8. J. Coates, *Construction of rational functions on a curve*, Proc. Cambridge Philos. Soc. **68** (1970), 105–123.
9. C. Dicrescenzo and D. Duval, *Computations on curves*, Lecture Notes in Comput. Sci., vol. 174, Springer, 1984, pp. 100–107.
10. ____, *Algebraic computations on algebraic numbers*, in Informatique et Calcul, Wiley-Masson, 1985, pp. 54–61.
11. D. Duval, *Diverses questions relatives au calcul formel avec des nombres algébriques*, Ph.D. thesis, L'Université Scientifique, Technologique, et Médicale de Grenoble, 1987.
12. W. Fulton, *Algebraic curves*, Benjamin/Cummings, 1974.
13. A. Galligo, *Apropos du théorème de préparation de Weierstrass*, Lecture Notes in Math., vol. 409, Springer, 1974, pp. 543–579.
14. D. Gorenstein, *An arithmetic theory of adjoint plane curves*, Trans. Amer. Math. Soc. **72** (1952), 414–436.
15. H. Hironaka, *Resolution of singularities of an algebraic variety over a field of characteristic zero*, Ann. of Math. (2) **79** (1964), 109–326.
16. E. Kaltofen, *Fast parallel absolute irreducibility testing*, J. Symbolic Comput. **1** (1985), 57–67.
17. D. Knuth, *The art of computer programming: Seminumerical algorithms*, Addison-Wesley, 1971.
18. H. T. Kung and J. F. Traub, *All algebraic functions can be computed fast*, J. Assoc. Comput. Mach. **25** (1978), 245–260.
19. H. Matsumura, *Commutative algebra*, Benjamin/Cummings, 1980.
20. F. Mora, *An algorithm to compute the equations of tangent cones*, Lecture Notes in Comput. Sci., vol. 144, Springer, 1982, pp. 158–165.
21. F. O. Schreyer, Ph.D. thesis, University of Hamburg, 1980.
22. J.-P. Serre, *Groupes algébriques et corps de classes*, Hermann, Paris, 1959.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF MICHIGAN, ANN ARBOR, MICHIGAN 48109.
E-mail: jeremy@math.lsa.umich.edu