Fast Computations in the Lattice of Polynomial Rational Function Fields

Franz Binder

Abstract

By Lüroth's theorem, all intermediate fields of the extension $\Bbbk(x) : \Bbbk$, $\Bbbk$ an arbitrary field, are simple. Those that contain a nonconstant polynomial, the *polynomial rational function fields*, constitute a sublattice (with respect to set inclusion). We give a fast algorithm for computing a generator of $\Bbbk(p, q)$, which is similar to the Euclidean algorithm, and also an extended version, that expresses this generator in terms of p and q. These algorithms work over any computable field, in particular, no assumption on the characteristic is needed.

Additionally, if $\Bbbk$ has zero characteristic, we use a deep result of Ritt to give a fast method to compute the other lattice operation, i.e., a generator of the intersection field $\Bbbk(p) \cap \Bbbk(q)$.

1 Intermediate Fields and Composition

If not specified differently, polynomials have coefficients in some fixed field $\Bbbk$, and are univariate using x as variable, i.e., they usually are elements of $\Bbbk[x]$. Similarly, a rational function is considered an element of the simple transcendental extension $k(x)$. Polynomials and rational functions can be composed by

$$f \circ g(x) := f(g(x)).$$

We also use the notation $f(g)$ for $f \circ g$, if convenient.

For nonconstant rational functions f and p there is at most one rational function r such that $f = r \circ p$; it is denoted by $f \div p$ in the affirmative case. If f is a polynomial and the degree of the numerator of p is greater than its denominator then both r and p are polynomials (see e.g. [6], [4], or [3] for such basic properties).

The degree of a polynomial p will be denoted by $[p]$, and we use the convention $[0] = 0$. This notion can be extended to a rational function $f = \frac{p}{q}$, where p and q have no common divisor, by

$$[f] = \max([p], [q]).$$

If K is a subfield of L, the extension will be denoted by $L : K$, and its degree by $[L : K]$. A rational function is called *indecomposable* or *prime* if it cannot be decomposed into ones of smaller degree. The intermediate fields of the extension $\Bbbk(x) : \Bbbk$ can be described easily:

1.1. Theorem (Lüroth). *All intermediate fields of the extension $\Bbbk(x) : \Bbbk$ are simple, i.e., of the form $\Bbbk(f)$ for some rational function $f \in \Bbbk(x)$.* □

Lüroth's theorem can be found in many advanced text books, e.g. [9] contains an elementary proof, and some more results in this topic. Now we have an intimate relation between the intermediate fields and functional composition. We summarize some important properties:

1.2. Proposition. *Let $p, q, f, t \in \Bbbk(x)$.*

1. *$\Bbbk(f) \subseteq \Bbbk(p)$ iff there exists a rational function r such that $f = r \circ p$.*
2. *$\Bbbk(q) = \Bbbk(p)$ iff there exists a rational function a of degree 1 such that $q = a \circ p$. If both p and q are polynomials then so is a.*
3. *$\Bbbk(p) : \Bbbk(f)$ has no intermediate fields iff $f \div p$ is indecomposable.*
4. *The lattice of intermediate fields of $\Bbbk(x) : \Bbbk$ can be determined completely by all essentially different decompositions of f into prime components. (cf. [2]).*
5. *t generates $\Bbbk(p, q)$ iff it is a greatest common right component of p and q, i.e., if there exist rational functions $\hat{p}$ and $\hat{q}$ such that $p = \hat{p} \circ t$ and $q = \hat{q} \circ t$, and such that for any other $\tilde{t} \in \Bbbk(x)$ with this property there is a rational function $\hat{t}$ with $t = \hat{t} \circ \tilde{t}$.*

6. *f generates $\Bbbk(p) \cap \Bbbk(q)$ iff it is a least common left multiple of p and q, i.e., if there exist rational functions r and s such that $f = r \circ p = s \circ q$, and such that for any other $\tilde{f} \in \Bbbk(x)$ with this property there is a rational function $\hat{f}$ with $\tilde{f} = \hat{f} \circ f$.*

7. *If an intermediate field contains a non-constant polynomial, then it can be generated by a polynomial.*

8. *If f is a polynomial then each intermediate field of $\Bbbk(x) : \Bbbk(f)$ is generated by a polynomial. In particular, the generator of $\Bbbk(p, q)$ is a polynomial whenever p and q are polynomials.*

9. *The intersection field of two polynomially generated fields is generated by a polynomial.*

10. $[\Bbbk(x) : \Bbbk(p)] = [p]$.

11. $[f \circ g] = [f]\,[g]$.

A more detailed treatment of such properties is contained e.g. in [3].

2 A Compositional Euclidean Algorithm

The key result used for our algorithm is essentially contained in [4]. We state and prove it here for convenience.

2.1. Proposition. *Let f, p, q, r be polynomials such that*

$$f = q \cdot p + r, \quad [r] < [p]\,.$$

Then

$$\Bbbk(f, p) = \Bbbk(p, q, r).$$

Proof. The $\subseteq$-part is trivial. For the converse assume that $\Bbbk(t) = \Bbbk(f, p)$ for some polynomial $t \in \Bbbk[x]$, thus there exist polynomials $\hat{f}$ and $\hat{p}$ such that

$$f = \hat{f}(t),\ p = \hat{p}(t).$$

Then, by Euclidean division,

$$\hat{f} = \hat{q} \cdot \hat{p} + \hat{r}; \quad [\hat{r}] < [\hat{p}]\,,$$

and substituting t into this equation gives

$$\hat{f}(t) = \hat{q}(t) \cdot \hat{p}(t) + \hat{r}(t); \quad [\hat{r}(t)] < [\hat{p}(t)]\,,$$

i.e.,

$$f = \hat{q}(t) \cdot p + \hat{r}(t); \quad [\hat{r}(t)] < [p]\,.$$

As the quotient and remainder are unique, it follows that $q = \hat{q}(t)$, $r = \hat{r}(t)$, thus $\Bbbk(t) \supseteq \Bbbk(q, r)$, and $\Bbbk(f, p) = \Bbbk(t) = \Bbbk(p, t) \supseteq \Bbbk(p, q, r)$. □

2.2. Remark.

1. Note that $[q] + [r] = [f] - [p] + [r] < [f]$. Therefore the sum of the degrees of the generators decreases if we take p, q, r instead of f and p. This gives the termination of the following algorithm (generators that are constants may be omitted).

2. The degree of a generator of $\Bbbk(p_1, p_2, \dots)$ is a divisor of $\gcd([p_1], [p_2], \dots)$. Thus if this gcd is $= 1$, x is a generator, and no computation is needed for this quite probable case.

2.3. Algorithm. *For any finite set $F = \{p_1, p_2, \dots\}$ of polynomials, the following method computes a generator of $\Bbbk(p_1, p_2, \dots)$.*

Remove all constants from the set F;
if $F = \varnothing$ **then return** 0;
while F contains at least two elements
and the gcd of their degrees is > 1
repeat choose polynomials
$f \in F$ and $p \in \Bbbk[F \setminus \{f\}]$, $[f] \geq [p] > 0$;
use Euclidean division of f by p,
giving q and r;
remove f from F;
add q and r instead, if nonconstant;
return the single element of F
or rather x if the gcd was $= 1$.

2.4. Remark. There is a lot of arbitrariness in this algorithm, involved by the word *choose*.

1. Most simply, we may just choose p in F. If F contains e.g. exactly one polynomial of rather high degree n, we have to expect $O(n)$ repetitions.

2. To get a more balanced version, we can choose the polynomial $f \in F$ of highest degree n, and p such that $[p] \approx \frac{n}{2}$. Computing appropriate polynomials of any order of magnitude between the smallest and the biggest elements of F can be done with $O(M(n) \log n)$ field operations, where $M(n)$ denotes the number of field operations necessary for multiplying or dividing two polynomials of degree n. With this strategy, a polynomial of degree n is replaced by two ones of degree $\approx \frac{n}{2}$. Thus we get the asymptotic bound of $O(M(n) \log n)$ field operations.

   This technique is somewhat similar to that proposed in [10] to test and compute $f \div p$.

3. Though we have got a quite satisfactory complexity bound for the worst case, there may be even better strategies. This is an open problem.

2.5. Remark. The algorithm should be compared with previous ones, though most of these were designed primarily for use with rational functions (cf. [1]). The method using minimal polynomials uses either matrix algebra (via the companion matrix) or Gröbner bases. Netto's method has to compute the gcd of polynomials in two variables. All these methods have worst case complexities rather high polynomial or even exponential. Sederberg's method uses just one univariate gcd computation, $\gcd(p(x)-p(\alpha), q(x)-q(\alpha))$, thus has about the same complexity as our one, but $\alpha \in \Bbbk$ must be chosen such that $(p(\alpha), q(\alpha))$ is regular, thus it is only probabilistic and works satisfactory only for char 0-fields. In fact, our method can be considered an improvement and extension of Sederbergs method (for polynomials).

The method computing right components of specified degrees is fast only if $\operatorname{char}\Bbbk = 0$ (or if the polynomials are at least tame, cf. [10]), and it may happen that a lot of decompositions have to be tested.

Thus, for polynomials, we have obtained a faster and more general algorithm. Additionally, observe that it is usually particularly fast in the rather probable case that the trivial solution (generator x) comes out.

2.6. Remark. The method used in the proof of Proposition 2.1 also provides an elementary proof, i.e., not depending on Lüroth's theorem, of the essentially unique (cf. 1.2.2) existence of a *greatest common right component* of polynomials p, q with respect to composition. Lüroth's theorem is used only to prove that it also *generates* $\Bbbk(p,q)$ (cf. Section 3). As a corollary, there is always a *least common left multiple*, which generates $\Bbbk(p) \cap \Bbbk(q)$ (cf. Section 4). In particular, one obtains that the polynomial subfields of $\Bbbk(x) : \Bbbk$ form a lattice, and that the generators of $\Bbbk(p,q)$ and of $\Bbbk(p) \cap \Bbbk(q)$ do not depend on the ground field $\Bbbk$, even in the so-called *wild case* (cf. [11]). Note that it may happen (and is quite probable) that $\Bbbk(p) \cap \Bbbk(q) = \Bbbk$; in this case, the generator is just a constant. The details are explained e.g. in [3] or [1].

2.7. Remark. In is not known whether Algorithm 2.3 can be generalized in order to compute a generator of $\Bbbk(\frac{r_1}{s_1}, \frac{r_2}{s_2}, \ldots, \frac{r_k}{s_k})$, i.e., the greatest common right component of general rational functions. But the generator of $\Bbbk(r_1, s_1, r_2, s_2, \ldots, r_k, s_k)$ is the greatest common right *polynomial* component of $\frac{r_1}{s_1}, \frac{r_2}{s_2}, \ldots, \frac{r_k}{s_k}$. Using 1.2.7 we see that our algorithm can also be used to compute the generator of the field $\Bbbk(\frac{r_1}{s_1}, \frac{r_2}{s_2}, \ldots, \frac{r_k}{s_k})$ if it is polynomial, in particular, if at least one of the given generators is a polynomial.

3 An Extended Compositional Euclidean Algorithm

Let d be the gcd of the polynomials p and q. Then Bezout's relation says that there exist polynomials r, s such that

$$d = r \cdot p + s \cdot q.$$

In other words, d can be expressed as a linear function of p and q. For a compositional version of this, we might expect that the greatest common right component of p and q, i.e., the generator of $\Bbbk(p,q)$, should be a nonlinear function of p and q. In fact, as a trivial consequence of Lüroth's theorem, we obtain:

3.1. Corollary. *Let p, q be polynomials. Then there exists a rational function φ in two variables such that*

$$\Bbbk(p,q) = \Bbbk(\varphi(p,q)). \qquad \square$$

The problem now is, how to find such a φ. For example, if $p = x^3 + 3x$ and $q = x^5 + x^4$, we see immediately that $\Bbbk(p,q) = \Bbbk(x)$, simply because $\gcd(5,3) = 1$, but it is considerably less obvious that

$$x = \frac{p^3 - 3p^2 + p \cdot q + 36p + 9q}{7p^2 + p \cdot q - 9p - 3q + 108}.$$

As the usual proofs of the part of Lüroth's theorem needed here can be considered constructive, the problem is solved, in principle. We try to develop an extended version of our compositional Euclidean algorithm. First there is an easy, though surprising result.

3.2. Corollary. *Let $f = q \cdot p + r$ be polynomials, with $[r] < [p]$. Then there exist bivariate rational functions φ, ψ such that*

$$r = \varphi(p,f), \quad q = \psi(p,f). \qquad \square$$

In fact, this follows immediately from proposition 2.1 and 3.1. If we had a direct (constructive) proof of this corollary then we could compute f recursively by a straightforward extension of Algorithm 2.3. Unfortunately we have not, so a slightly different approach has to be used.

We set up an important relation between functional decomposition and factorization. The first part is in [5]. We give a considerably simpler proof than the original one. A slight modification of this proof works also for a multivariate generalization. [2] contains a similar result for rational functions.

3.3. Theorem (Fried&McRae). *Let p,q,f,g be nonconstant polynomials. Then $p(y) - q(x)$ divides $f(y) - g(x)$ if and only if there exists a polynomial r such that*

$$f = r \circ p, \quad g = r \circ q.$$

Proof. The if-part is trivial. Thus assume that

$$f(y) - g(x) = u(x,y) \cdot (p(y) - q(x))$$

We apply Euclidean division of $f(y)$ by $p(y) - q(x)$. Doing this in $\Bbbk(x)[y]$, we see that $g(x)$ is the remainder, as it has degree 0 in y. But both elements are in the polynomial ring $\Bbbk(q)[y]$. Hence by the uniqueness of the remainder, $g(x) \in \Bbbk(q)$. Thus there exists a rational function r such that $g = r \circ q$. As both g and q are polynomials, so is r. Symmetrically we get $f = s \circ p$. It just remains to prove $r = s$.

Suppose $p(\alpha) = q(\beta) = c$ for some α, β, chosen possibly in the algebraic closure $\Bbbk^{\mathrm{alg}}$; thus for every $c \in \Bbbk^{\mathrm{alg}}$ there is a solution. Now substituting α and β in the factorization at the beginning, $r(c) = f(\alpha) = g(\beta) = s(c)$. Because $\Bbbk^{alg}$ is infinite, $r = s$ □

Suppose that $\Bbbk(h) = \Bbbk(f,g)$. Then, from Theorem 3.3, $h(y) - h(x)$ divides $\gcd(f(y) - f(x), g(y) - g(x))$ (trivial part). Conversely, if the gcd *has* the form $\hat{h}(y) - \hat{h}(x)$, then $\hat{h} = h$ (by the non-trivial part). Now we prove that the gcd *always* has this form.

We will write indices (e.g. $\gcd_y$ or $\gcd_{x,y}$) to denote the variables that are considered not to be in the coefficient field.

3.4. Lemma. *Let p, q be polynomials, and suppose that $\Bbbk(p,q) = \Bbbk(x)$. Then*

$$\gcd_y(p(y) - p(x), q(y) - q(x)) = y - x.$$

Proof. According to our convention, we do the computations in the polynomial ring $\Bbbk(x)[y]$. We show that x is the only common zero of $p(y) - p(x)$ and $q(y) - q(x)$. Suppose $\hat{x} \in \Bbbk(x)$ is another one, thus

$$p(\hat{x}) = p(x), \quad q(\hat{x}) = q(x).$$

Now, by 3.1, $x = \varphi(p,q)$. We apply φ to both sides of the equations, giving

$$\varphi(p(\hat{x}), q(\hat{x})) = \varphi(p(x), q(x)),$$

thus $x = \hat{x}$.

If x were a common *multiple* zero then $p' = q' = 0$, which is possible only if $p = \hat{p}(x^c)$ and $q = \hat{q}(x^c)$, $c = \operatorname{char} \Bbbk \neq 0$; but this contradicts $\Bbbk(p,q) = \Bbbk(x)$. □

This lemma is just what we need for our Algorithm. But the following generalization is worth mentioning.

3.5. Corollary. *Let p, q be polynomials and $\Bbbk(h) = \Bbbk(p,q)$. Then*

$$\begin{aligned} h(y) - h(x) &= \gcd_y(p(y) - p(x), q(y) - q(x)) \\ &= \gcd_x(p(y) - p(x), q(y) - q(x)) \\ &= \gcd_{x,y}(p(y) - p(x), q(y) - q(x)). \end{aligned}$$

Proof. Again computations are done in $\Bbbk(x)[y]$. We write p instead of $p(x)$. Let $\hat{p} = p \div h$ and $\hat{q} = q \div h$. Then $\Bbbk(\hat{p}, \hat{q}) = \Bbbk(x)$ and by the lemma, together with Bezout's relation,

$$y - x = r(x,y) \cdot (\hat{p}(y) - \hat{p}) + s(x,y) \cdot (\hat{q}(y) - \hat{q}).$$

We substitute $h(y)$ and $h(x) = h$ for y and x, respectively, thus

$$h(y) - h = r(h, h(y)) \cdot (p(y) - p) + s(h, h(y)) \cdot (q(y) - q).$$

Because $h(y) - h$ is a common divisor and satisfies Bezout's relation, it must be the greatest one.

Of course, the second equality follows by symmetry, and for the third observe that $h(y) - h(x)$ is a common divisor, and that $p(y) - p(x)$ and $q(y) - q(x)$ are primitive. □

3.6. Algorithm. *Given polynomials p and q, the following method computes a birational function φ such that $\Bbbk(p,q) = \Bbbk(\varphi(p,q))$.*

$h :=$ the generator of $\Bbbk(p,q)$;
if $h \neq x$ **then** use $\hat{p} \div h$ and $\hat{q} \div h$ instead.
"Thus we can assume $\Bbbk(p,q) = \Bbbk(x)$."
Introduce new variables α, β;
Apply Euclid's algorithm on
  $p(y) - \alpha$ and $q(y) - \beta$) (in $\Bbbk(\alpha,\beta)[y]$);
The last non-constant remainder must be linear
  (by proposition 3.5);
Let its monic form be $y - \varphi(\alpha, \beta)$;
return φ.

In contrast to the non-extended version the quotients are not added. They seem to be included in the remainders by the appearance of the new variables.

It should be noted, that this algorithm works similar to the method used in Netto's proof of Lüroth's theorem (cf. [8], [1]). The new thing about this is that, for polynomials, Netto's method computes the generator already in the first step.

3.7. Remark. An alternative approach uses matrix algebra, similar to most elementary proofs of Lüroth's theorem: Note that both $\mathbf{x} := \{1, x, x^2, \ldots, x^{n-1}\}$ and $\mathbf{p} := \{1, p, p^2, \ldots, p^{n-1}\}$ are bases of $\Bbbk(x) = \Bbbk(p,q)$ over $\Bbbk(q)$ ($n := [q]$, $m := [p]$). The matrix $\mathbf{T}$ transforming the basis $\mathbf{p}$ into $\mathbf{x}$ can be computed easily. Because $\mathbf{x} = \mathbf{T}^{-1} \cdot \mathbf{p}$ we can set

$$\varphi(p,q) := \sum_i a_i p^i,$$

where $a_1, \ldots, a_n \in \Bbbk(q)$ constitute the second row of the inverse $\mathbf{T}^{-1}$. Thus with this approach we have to invert an $n \times n$ matrix containing polynomials of degree m.

This seems to be faster then Algorithm 3.6, but the result is generally larger. (Note that the requested birational function is not unique). It is not known, whether the function constructed in Algorithm 3.6 is minimal in some appropriate sense. Additionally, the algorithm for the gcd-computation over $\Bbbk(p,q)$ (we have used Euclid's algorithm with cancellation before the last step) is open for improvements.

4 Faster Computation of Intersection Fields

The computation of a generator of an intersection field given in the form $\Bbbk(p) \cap \Bbbk(q)$ is much more delicate, because, in general, we have no constructive proof of its existence. The problem can be reduced to deciding whether the intersection field contains a non-constant element or not, but a general solution for this problem is not known either.

Things are much better in case $\operatorname{char} \Bbbk = 0$. In particular, Proposition 4.1 and Theorem 4.3 will need this assumption. Therefore:

Throughout this section, we will assume $\operatorname{char} \Bbbk = 0$.

4.1. Proposition. *Let $p, q \in \Bbbk[x]$. Then either*

- $\Bbbk(p) \cap \Bbbk(q) = \Bbbk$ *or*
- $[\Bbbk(x) : \Bbbk(p) \cap \Bbbk(q)] = \operatorname{lcm}([p], [q])$.

In the latter case, we also have

$$[\Bbbk(x) : \Bbbk(p,q)] = \gcd([p], [q]). \qquad \square$$

Again this goes back to [4]. In more abstract terms this means that the lattice of intermediate fields of $\Bbbk(x) : \Bbbk(f)$, f a polynomial, ordered by $\supseteq$, is embedded by the degree function into the lattice of divisors of $[f]$, ordered by divisibility ([3]).

From (the first part of) this proposition we know the degree of a possible non-trivial result in advance, and an approach with indetermined coefficients works and leads to a linear system of equations ([1]). From Ritt's strong result on bidecompositions we will obtain a considerably faster algorithm.

4.2. Definition. A *bidecomposition* consists of polynomials r, p, s, q non-trivially satisfying

$$r \circ p = s \circ q.$$

Note that if a, b, c, d are linear polynomials then $(a \circ r \circ b) \circ (b^{-1} \circ p \circ c) = (a \circ s \circ d) \circ (d^{-1} \circ q \circ c)$ is an *associated* bidecomposition. Of course, the inverses are with respect to composition.

4.3. Theorem (Ritt). *Every bidecomposition with relatively prime $[r] = [q]$ and $[p] = [s]$ is associated to one of the two forms*

$$x^n \circ (x^m \cdot u(x^n)) = (x^m \cdot u^n) \circ x^n$$
$$D_n(x, a^m) \circ D_m(x, a) = D_m(x, a^n) \circ D_n(x, a),$$

where u denotes an arbitrary polynomial, $a \in \Bbbk$, and the $D_n(x,a)$ are the Dickson polynomials (of the first kind), as defined e.g. in [7]. $\square$

This theorem, in the generality used here, is in [8].

Note that a generator of $\Bbbk(p \circ t) \cap \Bbbk(q \circ t)$ is just a generator of $\Bbbk(p) \cap \Bbbk(q)$ composed with t.

4.4. Algorithm. *A generator of $\Bbbk(p) \cap \Bbbk(q)$ for polynomials p, q over a field of characteristic 0 can be computed in the following way.*

Compute a generator t of $\Bbbk(p,q)$;
if $[t] \neq \gcd([p], [q])$ **then return** 0;
if $t \neq x$ **then** compute the generator of
$\Bbbk(p \div t) \cap \Bbbk(q \div t)$
instead and compose it with t.
"Thus we can assume $\Bbbk(p,q) = \Bbbk(x)$."
Test whether p and q match Theorem 4.3
if there is no match
then return 0
else we get corresponding polynomials r and s and **return** $r \circ p$.

4.5. Remark. The computation of the generator of $\Bbbk(p,q)$ at the beginning of the algorithm can be canceled, as soon as it is detected that its degree is $< \gcd(p,q)$. This optimization is particularly important if $[p] \approx [q]$.

4.6. Remark. The complexity of this algorithm depends on the costs for computing the generator of $\Bbbk(p,q)$, for which we have got an $O(\log n\, M(n))$-algorithm in Section 2, and for testing whether a polynomial is associated to a (generalized) power or to a Dickson polynomial. Comparing coefficients this can be done within at most $O(M(n))$ operations. In particular, this algorithm is considerably faster than solving a linear system of equations.

5 Further Remarks

Let us outline some applications.

The rational function φ computed in our extended compositional Euclidean algorithm can be used to do rational computations in $\Bbbk(p, q, \dots)$ more efficiently. After computing the generator t of $\Bbbk(p, q, \dots)$, one can transform the requested computation into one expressed in $\Bbbk(t)$, which is isomorphic to $\Bbbk(x)$, and transform the result back using φ, in order to get the result in terms of p, q, $\dots$ again.

Corollary 3.5 can be used to compute the gcd of polynomials of the form $p(y) - p(x)$ efficiently.

A parameterization (p, q) of a plain curve is faithful iff $\Bbbk(p,q) = x$. For polynomial parameterizations we can use our algorithm to test this quickly. Here it is

quite useful that our algorithm is particularly fast in the affirmative case.

All these algorithms have already been implemented. Improved publicly available packages are planned at least for *Mathematica*, and will be posted for anonymous `ftp` on `bruckner.stoch.uni-linz.ac.at`, in the directory `/pub/decomposition`

References

[1] Alonso, C. *Desarrollo, Análisis e Implementación de Algoritmos para la Manipulación de Variedades Paramétricas*. PhD thesis, Universidad de Cantabria, Santander, 1994.

[2] Alonso, C., Gutierrez, J., and Recio, T. A rational function decomposition algorithm by near-separated polynomials. *Journal of Symbolic Computation* (1996).

[3] Binder, F. Polynomial decomposition. Master's thesis, University of Linz, June 1995.

[4] Engström, H. T. Polynomial substitutions. *American Journal of Mathematics 63* (1941), 249–255.

[5] Fried, M., and MacRae, R. On curves with separated variables. *Math. Ann. 10* (1969), 220–226.

[6] Lausch, H., and Nöbauer, W. *Algebra of Polynomials*, vol. 5 of *North-Holland Mathematical Library*. North Holland, Amsterdam, 1973.

[7] Lidl, R., Mullen, G. L., and Turnwald, G. *Dickson Polynomials*, vol. 65 of *Pitman Monographs and Surveys in Pure and Applied Mathematics*. Longman Scientific & Technical, London, 1993.

[8] Schinzel, A. *Seclected Topics on Polynomials*. Ann Arbor, University of Michigan press, 1982.

[9] van der Waerden, B. L. *Algebra*, 5 ed., vol. II. Springer-Verlag, Berlin Heidelberg New York, 1967.

[10] von zur Gathen, J. Functional decomposition of polynomials: the tame case. *Journal of Symbolic Computation 9* (1990), 281–299.

[11] von zur Gathen, J. Functional decomposition of polynomials: the wild case. *Journal of Symbolic Computation 10* (1990), 439–452.