Algorithms for Manipulating Algebraic Functions

Master Thesis – M.I.T. (1976)

Barry Marshall Trager

Abstract

Given a base field k, of characteristic zero, with effective procedures for performing arithmetic and factoring polynomials, this thesis presents algorithms for extending those capabilities to elements of a finite algebraic extension of k. The algorithms were designed to be embedded in an algebraic symbolic manipulation system. An algebraic factorization algorithm along with a constructive version of the primitive element theorem is used to construct splitting fields of polynomials. These fields provide a context in which we can operate symbolically with all the roots of a set of polynomials. One application for this capability is rational function integration. Previously presented symbolic algorithms concentrated on finding the rational part and were only able to compute the complete integral in special cases. This thesis presents an algorithm for finding an algebraic extension field of least degree in which the integral can be expressed, and then constructs the integral in that field. The problem of algebraic function integration is also examined, and a highly efficient procedure is presented for generating the algebraic part of integrals whose function fields are defined by a single radical extension of the rational functions.

Contents

Chapter 1

Introduction

Mathematics has always been a natural source of applications for computer science. The earliest computers were usually thought of as mere high speed "number crunchers". This restriction to numerical problems was caused in part by the scarcity of languages suitable for manipulating the complex data structures needed for symbolic calculations. Another limitation was the inability of mathematicians to give complete, algorithmic descriptions of the operations which they routinely performed, e.g. integration. It was felt that these processes were largely heuristic, depending on experience gained from exposure to a wide variety of problems. As this learning process was not well understood, there was not much hope for developing computer systems for performing highly sophisticated mathematics. On the other hand, algorithms for tasks such as polynomial manipulation were fairly well understood, and many special purpose systems were developed for performing these operations. Thus a prerequisite for expanding the domain of applications of symbolic algebraic manipulation systems is the systematic development of effective algorithms for performing these "heuristic" tasks.

The problem of indefinite integration of elementary functions was attacked along these lines and finally solved, at least theoretically, by the work of Risch. The earliest attempt at integration was Slagle's SAINT [21], which used primarily pattern matching and table look up. In view of the way this subject is traditionally taught, this was thought to be the natural, and perhaps only, approach. The first really successful and practical program for integration was Moses' SIN. Its primary advantage was its dependence on special case algorithms, and almost any problem which it could solve at all was solved quite rapidly. Unfortunately the inability of SIN to perform a particular integration did not guarantee that the problem was in fact not integrable.

To perform a mathematical analysis of integration, we need very precise definitions of "elementary" functions and integrable in "finite terms". This was provided by Ritt's formulation of differential algebra [20]. A differential field is

a function field which is closed under differentiation. Following Risch [17] we define θ to be simple elementary over a differential field $\mathcal{D}$ if and only if one of the following conditions holds:

1. θ is algebraic over $\mathcal{D}$
2. There is an $f \in \mathcal{D}$, $f \neq 0$, such that $f' = f\theta'$, i.e. $\theta = \log f$.
3. There is an $f \in \mathcal{D}$ such that $\theta' = \theta f'$, i.e. $\theta = \exp f$.

Let $\mathcal{F} = \mathcal{D}(\theta_1, \ldots, \theta_n)$ where each θ_i is simple elementary over $\mathcal{D}(\theta_1, \ldots, \theta_{i-1})$, then any $g \in \mathcal{F}$ is said to be elementary over $\mathcal{D}$. Intuitively, an elementary function is one defined by a finite tower of algebraic, exponential, and logarithmic extensions of the rational functions. An elementary function is said to be integrable in finite terms if its integral is elementary.

Risch's decision procedure is based on a theorem of Liouville which gives us the form of the integral if it is an elementary function. If f is an element of a differential field $\mathcal{D}$ over $\mathbf{C}$, and there is a g elementary over $\mathcal{D}$ such that $g' = f$, then $\int f = v_0 + \sum c_i \log v_i$ where $v_i \in \mathcal{D}$ and $c_i \in \mathbf{C}$. For functions defined by exponential and logarithmic extensions, Risch's integration algorithm is reasonably efficient in many cases and has already been implemented by Moses and embedded in SIN. For the algebraic case, however, his algorithm requires many constructions from algebraic geometry and the existence of practical algorithms is not yet known.

If we restrict ourselves to purely algebraic functions, Liouville's theorem reduces to a theorem originally proved by Abel which states that the integral of an algebraic function, if it is elementary, will have a purely algebraic part and a purely logarithmic part. In the fifth chapter of this thesis we present a highly efficient algorithm for finding the algebraic part of the integral of algebraic functions involving only a single radical. For this class of functions, we are able to prove a much stronger result about the form of the integral. This allows us to avoid all the costly steps in Risch's approach, and generate directly a linear system of equations for the integral. If the equations are inconsistent, then the original problem was not integrable.

One simplifying assumption that mathematicians often make is that all their operations are being performed over an algebraically closed field. Thus they assume that they can factor any polynomial into linear factors and perform operations directly on the roots. This presents a real problem for implementors of algebraic manipulation systems since we have no effective procedure for constructing the algebraic closure of a field or performing operations there. Attempts to use complex floating point approximations to the elements of the closure can often suffer severely from roundoff errors, especially if many calculations are being performed. Many of the algorithms currently in use in algebraic systems require exact arithmetic since they make decisions based on zero tests

for the coefficients. Operating with floating point numbers, it is nearly impossible to distinguish small non-zero numbers from a representation of zero with roundoff present. Related problems also occur when one use interval arithmetic. Several of the algorithms presented in this thesis provide a partial solution to this problem. We cannot actually construct the algebraic closure, but given any particular set of polynomials we can construct their splitting field. This is the field of least degree over our base field which contains all the roots of all the polynomials. If we later need the roots of some more polynomials, we can extend the current splitting field to include their roots also.

Our approach to this problem is to construct a single extension of our base field in which all the roots are contained. In chapter two we present algorithms for performing arithmetic with algebraic expressions. If the defining polynomial for our extension field is univariate, then the expressions we manipulate are algebraic numbers. If multivariate, then the elements of our field are algebraic functions. The algorithms presented can be used in either domain.

Once we have constructed a splitting field, we need to be able to factor polynomials there. In [25] Paul Wang presents an algorithm for factoring multivariate polynomials over algebraic number fields. His approach is to perform the factorization modulo powers of small primes and then reconstruct the factorization over the integers using Hensel's lemma. His algorithm is somewhat faster than the algebraic factoring algorithm we present, but is restricted to algebraic numbers. Our algorithm has the advantages of increased generality and extreme simplicity, and in the context of splitting field calculations, our factoring algorithm is actually faster than Wang's.

In [10] Loos presents an algorithm using resultant calculations for reducing a multiple extension to a simple extension. The defining polynomials he uses are not irreducible since he wants to avoid factoring. We believe that in practice the factoring cost is dominated by the resultant cost, and our algorithm finds the minimal defining polynomial. We then use this primitive element algorithm iteratively along with our factoring algorithm to compute splitting fields.

In chapter four we use our splitting field algorithm to give a solution to the problem of rational function integration. Previous work in this area concentrated on finding the rational part of the integral. When all the functions in the transcendental part have rational coefficients, the integration can be completed by factoring the denominator over $\mathbf{Q}$ and then doing a partial fraction decomposition. However, in some cases the integral cannot be expressed over $\mathbf{Q}$, but over a finite algebraic extension of $\mathbf{Q}$. The integral can always be expressed in the splitting field of the denominator of the integrand, but often an extension field of much lower degree is also sufficient. Performing operations in an extension field of high degree is very costly, so it becomes important to compute the integral in an extension field of least degree over $\mathbf{Q}$. Tobey [22] claimed that a general procedure for finding the smallest field in which the integral can be expressed was an important research problem. Using the machinery developed in chapters two and three, we present a complete solution to this problem.

Chapter 2

Representations of Algebraic Expressions

Most early algebraic manipulation systems tried to find a single uniform representation for all expressions which they could handle. Simplicity of structure was a primary design criterion for a working system with a minimum programming effort. Since these early systems were primarily designed to solve a particular problem or a very limited class of problems, this requirement was not found too restrictive and aided communication between independent algorithmic modules.

As systems increased in internal complexity and in the complexity of the problems they could handle, designers became willing to trade simplicity for efficiency. This was the advent of multiple representations. Each module could select a representation which was most appropriate for a particular class of expressions. Conversion routines and some mechanism for coercing inputs for each module to the appropriate type were necessary.

In contructing a data type for algebraic expressions, we will carry this approach to its natural conclusion. We select a representation on the basis of the transformation to be performed instead of the data itself.

In this chapter we will examine several alternate representations for expressions which are algebraic over our base field k. We are assuming the capability of representing and manipulating elements of k and will present data structures and algorithms for extending these capabilities to algebraic extension fields of finite degree over k. We use subscripts to indicate an association between a function and some algebraic quantity, but we will drop the subscript when there is no ambiguity.

Let α be algebraic over a base field k. Then α satisfies a monic irreducible polynomial with coefficients in k, $f_\alpha(x) = x^n + a_1 x^{n-1} + \ldots + a_n, a_i \in k$.

$$\text{Let} A = \begin{pmatrix} 0 & 0 & \cdots & 0 & -a_n \\ 1 & 0 & \cdots & 0 & -a_{n-1} \\ 0 & 1 & \cdots & 0 & -a_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_1 \end{pmatrix}$$

Then A is called the companion matrix of $f(x)$, denoted by $C(f(x))$ [11].

The characteristic polynomial of any $n \times n$ matrix B is defined as the determinant of the matrix $xI_n - B$ where I_n is the $n \times n$ identity matrix. We shall frequently abbreviate this as $\text{Det}(x - B)$. It is clearly a polynomial of degree n. By the Cayley-Hamilton theorem, a matrix satisfies its characteristic polynomial. The minimal polynomial for B is the polynomial of least degree which B satisfies, and it therefore divides the characteristic polynomial.

By expanding $\text{Det}(x - A)$ in minors along the last column, it is easily seen that $f_\alpha(x)$ is the characteristic polynomial for $A = C(f(x))$. Since $f(x)$ is irreducible and the minimal polynomial divides it, $f(x)$ must also be the minimal polynomial for A.

Let $k[x]$ be the ring of polynomials in x with coefficients in k. Similarly, let $k[A]$ be the matrix algebra generated by A over k. Any element of $k[A]$ can be represented as a polynomial in A with coefficients from k. Thus there is a canonical surjective homomorphism from $k[x]$ to $k[A]$ which simply maps x to A. The kernel of this homomorphism, the set of all $p(x) \in k[x]$ such that $p(A) = 0$, is the principal ideal generated by $f_\alpha(x)$. The induced map from the quotient ring $k[x]/(f(x))$ to $k[A]$ is an isomorphism. In fact, since $f(x)$ is irreducible, the ideal it generates is prime and therefore the quotient ring is actually a field. We claim that this field is also isomorphic to $k(\alpha)$, the extension field obtained by adjoining α to the field k.

By a construction identical to the one above, we can show that $k[\alpha]$, the ring of polynomials in α is isomorphic to the field $k[x]/(f(x))$. Thus $k[\alpha]$ is actually a field and must be identical to $k(\alpha)$, the field of rational functions in α.

Thus we have shown that $k[\alpha]$ and $k[A]$ are both canonically isomorphic to the field $k[x]/(f(x))$. Therefore $k[\alpha]$ is isomorphic to $k[A]$ under an isomorphism which maps α to A.

We now have two alternative representations for elements of $k(\alpha)$, either as polynomials in α reduced modulo the minimal polynomial, $f_\alpha(x)$, or as $n \times n$ matrices in $k[A]$. Most previous references to algebraic numbers in the computer science literature have either explicitly or implicitly assumed the polynomial representation. Actually, there are some operations which can be performed more easily in the matrix representation. Not only do multiple representations lead to more efficient algorithms, but we can also gain further insight into this problem domain by considering algebraic expressions from three points of view:

1. A residue field of a polynomial ring (**P-rep**)

2. An n dimensional vector space over k (**V-rep**)

3. A commutative matrix algebra over k (**M-rep**)

In order to utilize all three points of view, we must have algorithms for changing from one representation to another.

P-rep: polynomials in α of degree less than n with coefficients in k.

V-rep: n-tuple or vector of length n.

M-rep: $n \times n$ matrix over k which can be constructed as a polynomial in A.

If we use $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ as the basis for our vector space then conversion between **P-rep** and **V-rep** is merely converting a polynomial into the list of its coefficients.

Before trying to convert to and from matrix form, let us examine the linear transformation associated with the matrix A operating on the elements of the vector space described by **V-rep**. If we take the powers of α as our canonical basis, then we can represent α^k (with $k < n$) as a vector with 1 in the $k+1^{st}$ position and zeros elsewhere. If $k < n-1$ then A maps α^k to α^{k+1}, however $A(\alpha^{n-1}) = (-a_n, -a_{n-1}, \ldots, -a)$, since A maps the j^{th} basis vector to the j^{th} column in A. Thus the effect of applying A to a basis vector is identical to multiplying the vector by α. In fact A is the linear transformation on the **V-rep** vector space corresponding to multiplication by α.

In general, if $\beta \in k(\alpha)$, $\beta = b_0 + b_1\alpha + \ldots + b_{n-1}\alpha^{n-1}$, then $M_\beta = b_0 I_n + b_1 A + \cdots + b_{n-1}A^{n-1}$ is the matrix associated with β and represents multiplication by β as a linear transformation.

If M_β is calculated by its defining equation above using Horner's rule then it requires $\mathrm{O}(n)$ matrix multiplications and $\mathrm{O}(n)$ matrix additions. Matrix addition requires $\mathrm{O}(n^2)$ element additions and matrix multiplication requires $\mathrm{O}(n^3)$ element multiplications and $\mathrm{O}(n^3)$ additions. Thus it would seem that generating the matrix representation for β requires $\mathrm{O}(n * n^3) = \mathrm{O}(n^4)$ multiplications and $\mathrm{O}(n * n^2) + \mathrm{O}(n * n^3) = \mathrm{O}(n^4)$ additions which would tend to make this operation prohibitively expensive.

However, we can use the transformational properties of the matrix M_β to generate a much improved algorithm. Since M_β corresponds to multiplication by β, the j^{th} column of M_β represents $\beta\alpha^{j-1}$, i.e. β times the j^{th} basis vector. In fact we see that the $k+1^{st}$ column is just α times the k^{th} column. The first column is $\beta\alpha^0 = \beta$, i.e. just the coefficient vector for β. To get from the k^{th} to the $k+1^{st}$ column, we multiply by α. This shifts each component except the last up one position. The last component was the coefficient of α^{n-1} and now becomes the coefficient of α^n and must be truncated. Since $\alpha^n = -a_1\alpha^{n-1} + \ldots - a_n$, this truncation involves multiplying the last component by the coefficient vector for α^n and adding the resulting vector to the shifted version of the previous column. Thus generating any column the previous one

requires n multiplications and $n-1$ additions. Since the first column is simply β, we must generate $n-1$ more columns. Therefore generating M_β from the vector representation of β requires $(n-1)n$ multiplications and $(n-1)^2$ additions or $O(n^2)$ operations. To convert from matrix to **V-rep** we simply strip off the first column.

2.1 Elementary Operations

Now that we have fixed our selection of representations, we will discuss algorithms for performing operations on algebraic expressions. The simplest operations are addition and subtraction of algebraic expressions. In all three representations these operations are merely performed element-wise. This requires $O(n)$ operations in **P-rep** and **V-rep** verses $O(n^2)$ operations in **M-rep**.

Multiplication is slightly more complicated. To multiply two algebraic expressions using **M-rep** we simply use formal matrix multiplication on the associated matrices. This is one of the principal advantages of **M-rep**; it contains both the multiplicative and additive structure of the algebraic expressions. Thus to multiply or divide using **M-rep** we simply multiply or invert the elements as matrices. However, we must pay a cost for this algorithmic simplicity. Matrices require $O(n^2)$ space instead of $O(n)$, and multiplying matrices reqires $O(n^3)$ operations whereas we can find a slightly more complicated algorithm for multiplication in the other representations which is $O(n^2)$. Note that since the conversion cost between **V-rep** and matrices is $O(n^2)$, we have the counter intuitive result that for our special polynomial matrix algebra we can multiply matrices faster by converting to another representation, multiplying there, and converting back (total cost = $O(n^2)$).

To multiply elements in **P-rep** we first multiply them as pure polynomials. Then we must truncate so that the highest degree term will be less than n. We accomplish this by using polynomial division with remainder. Our final result is the remainder after dividing the product polynomial by the minimal polynomial for α. Both of these steps requires $O(n^2)$ operations and therefore so does the result.

We could perform multiplication in **V-rep** by converting to and from **P-rep** and using the algorithm outlined above. This would also give us an $O(n^2)$ algorithm. But since the vectors in **V-rep** can be considered one dimensional arrays, we shall present a multiplication algorithm which exploits this structure. **V-rep** is essentially a dense representation for polynomials of bounded degree. Since algebraic expressions are likely to have few non-zero coefficients after even a few operations are performed, and since the exponents of the terms do not have to be stored explicitly in **V-rep**, this is probably the least space consuming representation. If we want to take advantage of the presumed denseness of the operands, we should avoid exponent comparisons and develop algorithms which process the vectors in a linear fashion.

Our multiplication algorithm will be unsymmetric in its treatment of the operands; to eliminate the need for temporary storage the first operand will be destroyed during the multiplication process. The inputs to the procedure are the two multiplicands and a vector supplying the coefficients (except the leading 1) of $P_\alpha(x)$. Each vector has indices running from 0 to $n-1$ and the ith component is the coefficient of α^i in the associated polynomial. The annotated algorithm follows:

ALGORITHM **V-mult**

input: v1,v2 multiplicands; Minp mininum polynomial for α; (All are n-vectors)

output: ov product vector

(1) for i=0 to n-1 do ov[i]:=0 (initialize)

(2) for i=0 to n-1 do

(1.1) for j=0 to n-1 do ov[j]:= ov[j] + v1[j] * v2[i]

(1.2) lastv1 := v1[n-1] (now multiply v1 by α)

(1.3) for j=n-1 to 1 step -1 do v1[j]:= v1[j-1] - lastv1 * Minp[j]

(1.4) v1[0]:= -lastv1 * Minp[0]

(3) return ov □

The only remaining elementary operation is division. Since we already have a multiplication algorithm we can reduce the division problem to inversion, i.e. computing $1/\beta$, however we will be careful to notice when a particular algorithm will allow us to compute γ/β at no extra cost.

As with multiplication we will consider **M-rep** first since it provides us with the simplest algorithm. Given the matrix M_β representing β, we can compute $M_{1/\beta}$ by formally inverting M_β as a matrix. In the next section we will prove that if $\beta \neq 0$ then M_β is nonsingular. Using gaussian elimination matrix inversion is an $O(n^3)$ operation if the elements of the matrix are units, i.e. we are working over a field. Thus using direct matrix operations, multiplication and division of algebraic expressions are $O(n^3)$.

The extended Euclidean Algorithm [8] provides us with an inversion algorithm for algebraic expressions in **P-rep**. Let $f_\alpha(x)$ be the minimal polynomial for α over k, and $P_\beta(\alpha)$ be our representation for β in **P-rep**. Since f is an irreducible polynomial and the degree of P is less than n, the degree of f, f and P are relatively prime polynomials. By the Euclidean algorithm, we can find $A(x)$ and $B(x)$ in $k[x]$ such that $A(x)P(x) + B(x)f(x) = 1$. If we now evaluate at α, f(α)=0 and $A(\alpha) = 1/P(\alpha)$. Thus $A(\alpha)$ is the representation for $1/\beta$ in **P-rep**.

2.2 Computing Norm and Trace

2.2.1 Definitions

A number α which is algebraic over k satisfies an irreducible polynomial with coefficients in k. $k(\alpha)$ is the field obtained by adjoining α to k. Let $f_\alpha(x)$ be the unique monic, irreducible equation of degree n which α satisfies. The conjugates of α over k are the remaining distinct roots of f_α, α_2, α_3, ..., α_n. If β is any element of $k(\alpha)$ then β can be represented uniquely as a polynomial in α, of degree less than n, with coefficients from k, $P_\beta(\alpha)$. The conjugates of β considered as an element of $k(\alpha)$ are $P(\alpha_2)$, $P(\alpha_3)$, ..., $P(\alpha_n)$. If $\beta = P(\alpha)$ we will let β_i represent $P(\alpha_i)$.

Two very useful mappings from $k(\alpha)$ to k are the Norm and Trace. The Norm(β) is the product of all the conjugates of β relative to $k(\alpha)$ over k. If we want to emphasize the fields involved we may also write $\text{Norm}_{[k(\alpha)/k]}(\beta)$. Similarly Trace($\beta$) (with the optional subscript) is the sum of the conjugates of β relative to the extension. Since both trace and norm are symmetric in the α_i, by the fundamental theorem on symmetric functions they both can be expressed in terms of the coefficients of P_α and thus lie in k.

We can extend the definitions of norm and trace to polynomials in several variables with coefficients in $k(\alpha)$. Any such function can be expressed as $G(x_1, x_2, \ldots, x_k, \alpha)$ where G is a polynomial in several variables with coefficients in k. Then the Norm(G) is the product of $G(x_1, x_2, \ldots, x_k, \alpha_i)$ over the n conjugates of α while the Trace(G) is the analogous sum.

The definition given above for the Norm of G coincides with the definition of Resultant$(P_\alpha(y), G(x_1, \ldots, x_k, y))$ as presented in [23, p. 86]. Collins [2] presents a modular resultant algorithm which can be employed to effectively compute norms. Even if the original polynomial is sparse, its norm is likely to be dense; thus a modular algorithm is probably the optimal choice. Note that norm is a multiplicative function by definition, i.e. Norm($A * B$) = Norm(A)*Norm(B). Thus we can extend the norm to rational functions by defining Norm(A/B) = Norm(A)/Norm(B).

We are now ready to prove an extremely important property of norms. (See [26, pp. 19–20])

THEOREM 1 *2.1 If $f(x, \alpha)$ is an irreducible polynomial over $k(\alpha)$ then the Norm(f) is a power of an irreducible polynomial over k*

PROOF:Assume Norm(f)=$C(x) * D(x)$ where $\gcd(C, D) = 1$. Let $f_i = f(x, \alpha_i)$, then Norm(f) = $\prod f_i$. The polynomial $f = f_1$ divides Norm(f) and since f is irreducible, either $f|C$ or $f|D$. Assume for concreteness that $f|C$, i.e. $C = f_1 * g_1$. The fields $k(\alpha_1)$ and $k(\alpha_j)$ are canonically isomorphic under a mapping ϕ_j which sends α_1 to α_j and is the identity on k. ϕ_j can be extended to the ring of polynomials in x over those fields and still remain an isomorphism. Since $C(x)$ has all its coefficients in k it is invariant under ϕ_j, but f_1 and g_1

are mapped to f_j and g_j respectively. Thus the equation $C = f_1 * g_1$ becomes $C = f_j * g_j$ under ϕ_j. Therefore $f_j|C$ for $1 \leq j \leq n$, but $\gcd(C, D) = 1$ implies that $\gcd(f_j, D) = 1$ for all j, and in turn $\gcd(\prod f_j, D) = 1$. But we assumed that $D|\text{Norm}(f) = \prod f_j$, so we have shown that $D = 1$. Thus the Norm(f) cannot be split into two relatively prime factors and can only be some power of an irreducible polynomial. □

A simple application of the above theorem is for finding minimal polynomials for elements of $k(\alpha)$. If β is an element of $k(\alpha)$ then, as above, $\beta = Q(\alpha)$. Thus $x - \beta$ divides the Norm($x - Q(\alpha)$)=$B(x)$ and therefore $B(\beta) = 0$. The problem remaining is to determine which irreducible factor of $B(x)$ is actually the minimal polynomial of β. The polynomial $x - \beta$ is linear and thus obviously irreducible. By the above theorem we see that $B(x)$ can only be a power of the minimal polynomial for β, $B(x) = P_\beta(x)^k$. $P_\beta(x)$ can be found directly by calculating the $\gcd(B(x), B'(x))$ where $B'(x)$ is the derivative of $B(x)$.

We will now investigate the relationship between the Norm(β) and the Det($M\beta$) where $\beta \in k(\alpha)$. We have already seen that the Det($x - M_\alpha$) is the minimal polynomial for α. We will show that in general Det($x - M_\beta$) is a power of the minimal polynomial for β. Thus since Det($x - M_\beta$) and Norm($x - \beta$) are both polynomials over k of the same degree and both powers of the same minimal polynomial, they must be equal. In fact as we might suspect, Det(M_β) is always equal to Norm(β) and this will supply us with an alternate algorithm for computing norms.

It is an elementary result of linear algebra [6, pp. 225–226] that the minimal polynomial for a matrix and its characteristic polynomial have the same set of roots, the eigenvalues of the matrix. Because of the representation isomorphism, the minimal polynomial for any M_β is the same as the minimal polynomial for β and is thus irreducible over k. (Note: the minimal polynomial for a general matrix is not necessarily irreducible). Since the characteristic polynomial (Det(x-M_β)) is a polynomial over k and has the same roots as an irreducible polynomial, it must be a power of the minimal polynomial. Thus as explained above, Det(x-M_β) = Norm($x - \beta$).

By definition Norm($x - \beta$) = $\prod(x - \beta_i)$ with the product taken over all the conjugates of β. The constant coefficient (coefficient of x^0) of the norm is $(-1)^n \prod \beta_i$ or $(-1)^n Norm(\beta)$. Similarly the constant coefficient of Det($x - M_\beta$) is $(-1)^n$Det(M_β). Since Norm($x - \beta$) = Det($x - M_\beta$), their trailing coefficients must be identical and therefore Norm(β) = Det(M_β). Thus we now have two ways to compute the norm, either as a resultant or the determinant of an $n \times n$ matrix where $n = \deg(k(\alpha)/k)$.

The coefficient of the x^{n-1} term in Norm($x - \beta$) is $-\sum \beta_j$ or -Trace(β). By expanding Det(x-M_β) in minors we see that the coefficient of the x^{n-1} term is actually the sum of the diagonal elements of the matrix -M_β. Thus we have the not very surprising result that the algebraic Trace(β) is equal to the Matrix

Trace(M_β). We now have two ways to compute the Trace(β), either minus the coefficient of the x^{n-1} term in Norm($x - \beta$) or the matrix Trace(M_β). The second way is by far more efficient, only requiring O(n) operations in **M-rep**. If we start in some other representation the time required to compute the trace is dominated by the time to convert to **M-rep** or O(n^2). **M-rep** requires O(n^2) space, but if we are only interested in computing the trace from **V-rep**, we can modify our conversion algorithm to only use O(n) space and still O(n^2) operations.

ALGORITHM **Trace**

input: Bvec the **V-rep** of $\beta \in k(\alpha)$, Pvec the **V-rep** of the minimal poly. for α

output: $\text{Trace}_{[k(\alpha)/k]}(\beta) \in k$

(1) N = Length (Bvec), Trace = 0

(2) for i=0 thru n-1 do

  (2.1) Trace = Trace + Bvec[n-1]

  (2.2) for j=i+1 thru n-1 do

    (2.2.1) Bvec[j] = Bvec[j] - Bvec[i] * Pvec[j-i]

(3) return Trace □

Chapter 3

Algebraic Factoring and Splitting Fields

3.1 Introduction

For many applications in symbolic mathematics it is necessary to explicitly describe all the roots of a polynomial. One approach is to compute the roots numerically to some predetermined accuracy. This is the approach taken in numerical analysis packages but is generally avoided in symbolic manipulations as the basic routines such as greatest common divisor and factoring would then be useless. Another attempt is to try to express the roots in terms of radicals, but this often cannot be done, and even when it can it leads to great problems when simplifications are required. The approach taken here is to describe an extension field of the rationals by the minimal polynomial for some primitive element and then to express all the roots of the polynomials in terms of that element. If an irreducible polynomial over k is normal then all of its roots are rationally expressible in terms of any one of them. Thus the degree of its splitting field is the same as the degree of the polynomial. For other polynomials, however, the degree of its splitting field may be as high as $n!$. This degree growth tends to make many "computable" problems practical impossibilities. It then becomes very important to operate in as low degree an extension field as possible.

This chapter presents a new, simple, and relatively efficient algorithm for factoring polynomials in several variables over an algebraic number field. This algorithm is then used iteratively to construct the splitting field of a polynomial over the integers. In the next chapter the factorization and splitting field algorithms are applied to the problem of determining the transcendental part of the integral of a rational function. In particular, a constructive procedure is given for finding the least degree extension field in which the integral can be

expressed. Each algorithm is preceded by theorems providing its mathematical basis.

3.2 Norms and Algebraic Factoring

We assume the existence of some base field of characteristic zero (e.g. the rationals) over which we have constructive procedures for factoring polynomials. We also assume the capability is present to perform basic polynomial operations (e.g. division with remainder) in both the base field and some extension field of finite degree. Our approach is to map a polynomial over an extension field to one of higher degree in the base field such that there is an exact correspondence between their factorizations in their respective fields. We use this correspondence to reconstruct the factorization of the original polynomial over the extension field.

We now turn to the problem of factoring polynomials with coefficients in $k(\alpha)$ assuming we have this capability over k. The ability to perform basic arithmetic in $k(\alpha)$ allows one to compute gcd's of polynomials over $k(\alpha)$ by the Euclidean Algorithm. Thus we can perform a square free decomposition on any polynomial and reduce the factoring problem to square free polynomials.

Our approach to the factorization of f(x,α) is first to make a linear substitution for x so that the norm(f) is square free. We then factor the norm(f) over k. Norm(f)=$G_1(x)G_2(x)\cdots G_r(x)$ with each G_i distinct and irreducible over k. We claim that $g_i(x,\alpha) = \gcd(f, G_i)$ is irreducible over $k(\alpha)$ for all i and that $f = \prod g_i$.

THEOREM 2 *3.1 Let $f(x,\alpha)$ be a polynomial over $k(\alpha)$ such that the Norm(f) is square free. Let $\prod G_i(x)$ be a complete factorization of the Norm(f) over k. Then the $\prod \gcd(f(x,\alpha), G_i(x))$ is a complete factorization of f over $k(\alpha)$.*

PROOF:Let $g_i = \gcd(f(x,\alpha), G_i(x))$, then we must show that each g_i is irreducible and that all the irreducible factors of f are among the g_i. Let $v(x)$ be an irreducible factor of f over $k(\alpha)$. By the previous theorem Norm(v) must be a power of an irreducible polynomial over k, but $v|f$ implies Norm(v)—Norm(f) and the Norm(f) is square free. Therefore the Norm(v) is irreducible and must be one of the G_i. Since the Norm(f) is equal to the product of the norms of each of the irreducible factors of f, each G_j must be the norm of some irreducible factor of f. Assume both $v_1(x)$ and $v_2(x)$ divide $\gcd(f, G_i)$ where v_1 and v_2 are distinct irreducible factors of f. $v_1|G_i$ implies norm(v_1)—norm(G_i) , but $G_i(x)$ is a polynomial over k and its norm is $G_i(x)^n$. The norm(v_1) is irreducible over k and divides a power of the irreducible polynomial $G_i(x)$, thus the norm(v_1)=$G_i(x)$. Similarly the norm(v_2)=$G_i(x)$. But $(v_1v_2)|f$ implies Norm(v_1v_2)=$G_i(x)^2$|Norm(f) and this contradicts the assumption that the Norm(f) was square free. Therefore the $\gcd(f, G_i(x))$ must be irreducible for all i. □

The only missing step in the previously outlined factoring procedure is finding a linear substitution that makes Norm(f) square free. We claim that Norm($f(x+s\alpha)$) is square free for some s in k. We will prove this in two stages, first for $f(x)$ a polynomial over k and then extend the result to polynomials over $k(\alpha)$.

THEOREM 3 *3.2 If $f(x)$ is a square free polynomial with coefficients in k, then there are only a finite number of s in k such that Norm($f(x-s\alpha)$) has a multiple root.*

PROOF:Let the roots of $f(x)$ be $\beta_1, \beta_2, \ldots, \beta_m$, all distinct; then the roots of $f(x-s\alpha_j)$ are $\beta_1+s\alpha_j, \ldots, \beta_m+s\alpha_j$. Let $G(x) = \text{Norm}(f(x-s\alpha)) = \prod_i f(x-s\alpha_i)$. Thus the roots of G are $s\alpha_k + \beta_i$ for $k \leq n$,$i \leq m$. G can have a multiple root only if $s = (\beta_j - \beta_i)/(\alpha_k - \alpha_m)$ where $k \neq m$. Therefore there are only a finite number of such values. □

LEMMA 4 *If $f(x,\alpha)$ is a square free polynomial with coefficients in $k(\alpha)$ then there exists a square free polynomial $g(x)$ over k such that $f|g$.*

PROOF:Let G(x)=Norm(f(x,α)), let $\prod G_i(x)^i$ be a square free decomposition of G. Then $g(x) = \prod G_i(x)$ is a polynomial over k. Since f is square free and we have only discarded the multiple factors of $G(x)$, $g(x)$ is divisible by f. □

COROLLARY 5 *If $f(x,\alpha)$ is a square free polynomial over $k(\alpha)$ then there are only a finite number of s in k such that Norm($f(x-s\alpha)$) has a multiple root.*

PROOF:Let $g(x)$ be a polynomial over k as in the lemma. By theorem 3 there are only a finite number of s such that Norm($g(x-s\alpha)$) has multiple roots. But $f|g$ implies Norm($f(x-s\alpha,\alpha)$) divides Norm($g(x-s\alpha)$) and thus any multiple root of the former is a multiple root of the latter. □

We are now ready to present the entire factoring algorithm, but we will split off the norm computation as a subroutine for later use by other procedures.

ALGORITHM **sqfr_norm**

input: $f(x,\alpha)$ a square free polynomial over $k(\alpha)$

output: a positive integer s, $g(x,\alpha) = f(x-s\alpha,\alpha)$, $R(x)$=Norm($g(x,\alpha)$) a square free polynomial over k.

(1) $s = 0$,$g(x,\alpha) = f(x,\alpha)$ [initialize]

(2) $R(x)$=resultant$(P_\alpha(y), g(x,y), y)$ [Norm,(resultant taken with respect to y)]

(3) If degree(gcd$(R(x), R'(x))$=0 then return (s,g,R) [sqfr check]

(4) $s = s+1$, $g(x,\alpha) = g(x-\alpha, \alpha)$,go to (2) □

ALGORITHM **alg_factor**

input: $f(x,\alpha)$ a square free polynomial over $k(\alpha)$

output: a list of irreducible factors over $k(\alpha)$

(1) (s,g,R)=sqfr_norm$(f(x,\alpha))$

(2) L=factor(R(x)) [over k, returns list of factors]

(3) If length(L)=1 then return (f) [original poly was irreducible]

(4) For each $h_i(x)$ in L Do

(4.1) $h_i(x,\alpha) = \gcd(h_i(x), g(x,\alpha))$

(4.2) $g(x,\alpha) = g(x,\alpha)/h_i(x,\alpha)$ [performed over $k(\alpha)$]

(4.3) $h_i(x,\alpha) = h_i(x+s\alpha, \alpha)$ [undoes linear transformation]

(5) return (L) □

This factoring algorithm is similar to the one presented in van der Waerden [23, pp. 136-7] but computationally more efficient. If one wants to factor a univariate polynomial of degree d over an extension field of degree n, van der Waerden's approach requires computing a norm which is bivariate of degree nd in each variable and then factoring it over k. The algorithm presented above leads to the computation and factoring of a univariate norm of degree nd. It appears we have the additional cost of finding a linear transformation which makes the norm square free. However, the first step in factoring a bivariate polynomial over the k is to find a substitution for one of the variables which makes the result square free. Thus there is no actual additional cost and this algorithm is always superior to van der Waerden's.

In [25] Paul Wang gives another algorithm for factorization over algebraic number fields. His approach is an extension of his algorithm for factoring over the integers [24]. Wang's algorithm utilized van der Waerden's technique when the minimal polynomial for the algebraic number factored over primes, e.g. x^4+1. He now uses our improved algorithm in this case. In other cases, Wang's algorithm appears superior to ours, but not markedly so. We expect to analyze the computing times in both algorithms in the near future.

3.3 Primitive Elements

Next we will present an algorithm for computing a primitive element for a tower of extension fields. All of the algorithms we have presented have assumed that the extension field we operate in can be described by the adjunction of a single element to our base field k. If our current extension field is $k(\alpha)$ and β is algebraic over $k(\alpha)$ with minimal polynomial $Q_\beta(x,\alpha)$ then $k(\alpha,\beta)$ is the field obtained by adjoining β to $k(\alpha)$. We seek some γ which is algebraic over k such that $k(\gamma) = k(\alpha,\beta)$. The following theorem will prove very useful.

LEMMA 6 *Let $P_\alpha(x)$ be the minimal polynomial for α over k and β be a root of $Q(x,\alpha)$, a square free polynomial. If $Norm_{[k(\alpha)/k]}(Q_\beta(x))$ is square free then $\gcd(P_\alpha(x), Q(\beta,x))$ is linear.*

PROOF:α is clearly a root of both $P_\alpha(x)$ and $Q(\beta,x)$ since $Q(\beta,\alpha) = 0$. Let the other roots of $P(x)$ be α_j for $j = 2,\ldots,n$. If $Q(\beta,\alpha_j) = 0$ then β is a root of both $Q(x,\alpha)$ and $Q(x,\alpha_j)$, but Norm(Q) is $\prod Q(x,\alpha_i)$ and then β is a multiple root of the norm. Since the norm is square free this cannot happen and the only common root of $P(x)$ and $Q(\beta,x)$ is α. Therefore the $\gcd(P(x),Q(\beta,x))$ is linear. □

THEOREM 7 *3.3 Let $Q_\beta(x,\alpha)$ be the minimal polynomial for β over $k(\alpha)$ and $P_\alpha(x)$ be the minimal polynomial for α over k. If $Norm_{[k(\alpha)/k]}(Q_\beta(x))$ is square free then $k(\alpha,\beta) = k(\beta)$.*

PROOF:We only need to show that α is representable in $k(\beta)$. By the lemma the gcd(Q(β,x),P(x))=x-c, i.e. is linear. c is the only common root of Q(β,x) and P(x), so c=α. But Q(β,x) and P(x) are both polynomials over $k(\beta)$ and so their gcd is over $k(\beta)$. Thus α=c is in $k(\beta)$. □

Given an arbitrary β the norm($Q_\beta(x,\alpha)$) may not be square free, but algorithm sqfr_norm will find an integer s and a polynomial g(x,α) such that R(x) = Norm($g(x,\alpha)$) is square free. Since Q(x,α) is irreducible over $k(\alpha)$ and sα is in $k(\alpha)$, g(x,α) = Q(x-sα,α) is irreducible over $k(\alpha)$. If we let γ be a root of g(x,α) over $k(\alpha)$ and thus a root of R(x) over k then by theorem 3.3 $k(\alpha,\gamma) = k(\gamma)$. But $\gamma = \beta$+sα so $k(\alpha,\gamma) = k(\alpha,\beta)$. Thus γ is the primitive element we were looking for and R(x) is its minimal polynomial over k.

The algorithm we present for calculating primitive elements is essentially the same as that presented by Loos [10]. We differ in allowing the minimal polynomial for β to have coefficients over $k(\alpha)$ instead of requiring it to be over k. Loos does not guarantee, as we do, that the polynomial returned be irreducible. To achieve this result we must require that the minimal polynomial for β be irreducible over $k(\alpha)$ not just irreducible over k. In fact, if we examine our

algebraic factoring algorithm, we can determine the conditions under which the resultant of two irreducible (over k) polynomials will factor. This will happen if and only if each of the polynomials factors over the extension field determined by the other polynomial. By symmetry, it is sufficient if one of the polynomials factors over the other's extension field.

ALGORITHM **primitive_element**

input: $P_\alpha(x)$ the minimum polynomial for α over k
$Q_\beta(x,\alpha)$ the minimum polynomial for β over $k(\alpha)$

output: R(x) the minimum polynomial for γ over k where $k(\alpha,\beta) = k(\gamma)$.

A(γ) is a representation of α in $k(\gamma)$

B(γ) is a representation of β in $k(\gamma)$

(1) (s,g,R) = sqfr_norm($Q(x,\alpha)$,$P(x)$)

(2) α=linsolve(gcd(g(γ,x),P(x))) [arithmetic over $k(\gamma)$ where γ denotes a root of R(x), linsolve($ax-b$) returns b/a]

(3) $\beta = \gamma - s\alpha$

(4) return (R(x),γ,α,β) □

3.4 Splitting Fields

The last algorithms to be presented in this section calculate splitting fields. We restrict our considerations to irreducible polynomials since the composite field from many such extensions can be found by repeated application of the primitive_element algorithm. Our basic approach will be to alternately adjoin a root of an irreducible factor of the polynomial to the current extension field and then refactor the polynomial in the new extension field. As linear factors are discovered they are put on a separate list and their coefficients are updated as the extension field changes. If n is the degree of the original polynomial, in the worst case $n-1$ iterations will occur and the primitive element for the splitting field will be of degree $n!$.

ALGORITHM **split_field**

input: P(x) a polynomial irreducible over k

output: $R_\gamma(x)$ the defining polynomial for the splitting field of P(x) and a list of the roots of P(x) over $k(\gamma)$.

(1) roots = [], polys = [P(x)]

(2) minpoly = P(x), newminpol = P(x), index = 1, β=γ [γ is a root of minpoly]

(3) replace polys[index] by polys[index]/(x-β), add β to roots, Newfactors = [], k = 1

(4) for each $P_i(x)$ in polys do

(4.1) (g,s,R) = sqfrnorm($P_i(x)$,minpoly)

(4.2) L = factor (R(x))

(4.3) for each $Q_j(x)$ in L do

(4.3.1) $f(x,\gamma) = \gcd(g(x,\gamma), Q_j(x))$
[f is an irred. factor of P_i in $k(\gamma)$]

(4.3.2) if Deg(Q_i)¿Deg(newminpol) then do

(4.3.2.1) newminpol = $Q_j(x)$, index=k, new_s=s, Bpoly(x,γ) = f(x,γ)

(4.3.3) $g(x,\gamma) = g(x,\gamma)/f(x,\gamma)$

(4.3.4) $f(x,\gamma) = f(x + s\gamma, \gamma)$

(4.3.5) If Deg($f(x,\gamma)$) = 1

(4.3.5.1) then add linsolve($f(x,\gamma)$) to roots

(4.3.5.2) else add $f(x,\gamma)$ to Newfactors, $k = k + 1$

[let new_γ be a root of newminpol, now we operate in k(new_γ)]

(5) α = linsolve(gcd(minpoly,Bpoly(new_γ,x)))

(6) β = new_γ-new_sα

(7) subst α for γ in roots [update for new extension]

(8) If Newfactors = [] then return (newminpol,roots)

(9) subst α for γ in Newfactors

(10) polys = Newfactors, minpoly = newminpol, γ = new_γ, go to 3. □

For comparison, we now present a simpler version of the above algorithm which avoids performing any factoring. If the splitting field is actually of degree $n!$ this approach will actually be faster since the attempt at factorization in step 4.2 would always fail and thus be a waste of time. The essence of the splitting field calculation is repeated primitive element calculations. In the algorithm above factorization is attempted in the hope that the degree of the splitting field is much less than $n!$. The algorithm below always returns a polynomial of degree $n!$ known as the resolvent [4]. The irreducible factors of this polynomial over k are all of the same degree, all normal, and are all defining polynomials for the splitting field of $P(x)$.

ALGORITHM **resolvent**

input: $P(x)$ a polynomial irreducible over k

output: $R(x)$ a polynomial of degree $n!$ such that any irreducible factor of R defines a splitting field for P

(1) minpoly = $P(x)$, $\beta = \gamma$ [γ is a root of minpoly]

(2) $P(x, \gamma) = P(x)/(x - \beta)$

(3) If degree(P)=0 then Return minpoly

(4) (g,s,R) = sqfrnorm($P(x, \gamma)$,minpoly)

[Let new_γ be a root of $R(x)$, now operate in k(new_γ)]

(5) α = linsolve(gcd(minpoly,P(new_γ, x)))

(6) β = new_γ - $s\alpha$

(7) subst α for γ in $P(x, \gamma)$

(8) minpoly = $R(x)$, γ = new_γ, go to 2. □

3.5 Extensions and Comments

The algorithms presented in this chapter were designed to operate over an arbitrary base field. If we are interested in factoring univariate polynomials over algebraic number fields then we let our base field be the rational numbers. Given the capability to factor multivariate polynomial norms over Q as in [24], we can extend to factoring multivariate polynomials over algebraic number fields. If we allow our minimal polynomials to have polynomial coefficients then we can factor polynomials over algebraic function fields.

As algebraic manipulation systems expand their problem domains, the need for performing operations with quantities satisfying algebraic relationships will increase. The basic arithmetic operations can be performed by merely using the side relations to keep the expressions reduced. We have extended factoring to these domains by mapping the problem to a simpler domain while still preserving its structure. Then we were able to lift the factorization back to the original expression. That finding such unramified morphisms can lead to efficient algorithms for algebraic manipulation has been amply demonstrated by the recent development of modular and p-adic techniques [27].

Chapter 4

Rational Function Integration

We now turn to an application of the algorithms presented in the previous section. The problem of the symbolic integration of rational functions in the context of algebraic manipulation has been investigated by Manove et al [14], Moses [15], Horowitz [7], Tobey [22], and Mack [12]. By Liouville's theorem the integral of a rational function can be expressed as a rational function plus a sum of complex constants times logs of rational functions.

$$\int R(x)\,dx = v_0(x) + \sum c_i \log(v_i(x))$$

Algorithms for obtaining the rational part of the integral are well known and reasonably efficient [7], [12], but as far as the author knows, no one has actually presented practical and relatively general algorithms for obtaining the transcendental part. Horowitz limited his investigation to algorithms for obtaining the rational part. Manove and by extension Moses' SIN factored the denominator over the integers and obtained logarithmic terms if the factors were linear or myquadratics. Tobey examined the transcendental problem and concluded "There is no generally valid algebraic algorithm for obtaining in a symbolic form the transcendental part of the integral of a rational function." He did however present some algorithms for obtaining the transcendental part in special cases. He also presented as an unsolved problem the problem of obtaining the least degree extension field in which the integration can be done.

Starting where Horowitz leaves off, we are interested in integrating a rational function $S(x)/T(x)$ where $T(x)$ is square free and degree($S(x)$) ¡ degree($T(x)$). If we were willing to go to the splitting field of the denominator then we could perform a partial fraction expansion to get $S(x)/T(x) = \sum c_i/(x - \theta_i)$ where each c_i is an element of the splitting field and θ_i is a root of the denominator.

In fact c_i is just the residue of $R(x)$ at θ_i. This would lead to an expression for the integral, but could require operating in an extension field of very high degree. Thus for the sake of efficiency and to promote a more intelligible result, we propose to find the minimum degree extension field in which the result can be expressed. We claim that any field which contains all the residues is sufficient for expressing the integral. This is a significant result since the residues may be contained in a field of much lower degree than the splitting field of the denominator.

Our approach to this problem is first to constructively find the extension field E determined by all of the residues of the integrand. Then we factor the denominator of the integrand over this field and perform a partial fraction decomposition. This breaks the original integral into a sum of integrals of proper rational functions where each denominator is irreducible over E. We claim that each integral in the sum can be expressed as a single log term.

First we will find a simple expression for the residues in terms of the roots of the denominator T(x). Let θ be a root of the square free polynomial T(x). Since θ must then be a simple root, the residue of $S(x)/T(x)$ at θ is $S(\theta)/T'(\theta)$. Since T(x) is square free, the $\gcd(T(x), T'(x)) = 1$. Thus by the extended Euclidean algorithm we can find polynomials $A(x)$ and $B(x)$ over k such that $A(x)T(x) + B(x)T'(x) = S(x)$, with the degree($B(x)$) ¡ degree($T(x)$). $T(\theta) = 0$ implies $B(\theta) = S(\theta)/T'(\theta)$. Since θ was an arbitrary root of T(x) we have established the following:

LEMMA 8 *Let $S(x)$ and $T(x)$ be polynomials over k, where $T(x)$ is square free. Then there exists a polynomial $B(x)$ over k such that for any root θ of $T(x)$ the residue of $S(x)/T(x)$ at θ is $B(\theta)$.*

THEOREM 9 *4.1 Let $S(x)/T(x)$ be a quotient of polynomials where $T(x)$ is irreducible over some ground field k. If all the residues are contained in k then all the nonzero residues are equal.*

PROOF:Let B(x) be the polynomial described in the lemma. Let the roots of T(x) be θ_i for i=1,...,n. We will operate in the splitting field E of T(x). Since T(x) is irreducible, its Galois group G(E/k) is transitive and there is an automorphism ϕ_j sending θ_1 to θ_j for each j and leaving k unchanged. Let c = B(θ_1), the residue at θ_1. If we apply ϕ_j to this equation, the left side is unchanged since c is in k and the right side is mapped to B(θ_j) since the coefficients of B(x) are in k. Thus c = B(θ_j) for each j and all the residues are equal. □

COROLLARY 10 *With $S(x)$ and $T(x)$ as in Theorem 2.1, $\int S(x)/T(x)\,dx$ is expressible over k.*

PROOF:Let c be the common residue of the θ_j, then the integrand has the partial fraction decomposition, $S(x)/T(x) = \sum c/(x-\theta_j)$. This integrates to the following:

$$\int S(x)/T(x)\,dx = \sum c\log(x-\theta_i) = c\log\prod(x-\theta_i) = c\log T(x)$$

□

All that remains is to transform the integrand so that the corollary applies. If θ is any root of the denominator T(x) then B(θ) is the residue of $S(x)/T(x)$ at γ, therefore the resultant(x-B(y),T(y),y) is a polynomial whose roots are precisely the residues of $S(x)/T(x)$. Using the algorithms presented in the last section we can compute the splitting field of this polynomial and therefore the least degree extension which contains all the residues. Then we factor the denominator of the integrand over this extension field. If we performed a partial fraction decomposition on the resulting rational function, each term in the sum would satisfy the hypotheses of the corollary and thus be directly integrable. But the actual computation of the partial fraction decomposition is unnecessary. Since we know the form of the result as $\sum c_i \log(f_i(x,\gamma))$ where the f_i's are the irreducible factors of the denominator, all we need to be able to do is compute each c_i. But c_i is B(x) evaluated at any root of $f_i(x,\gamma)$. The resultant(B(x),$f_i(x,\gamma)$,x) is the product of B(x) evaluated at each of the roots of f and thus equals c_i^k where k is the degree of f. Therefore g(y) = resultant(y-B(x),$f_i(x,\gamma)$,x) is $(y-c_i)^k$, and $y-c_i$ is $g(x)/\gcd(g(x),g'(x))$.

ALGORITHM **ratint**

input: T(x) a square free polynomial, S(x) a polynomial of lower degree than T(x)

output: R(x) the minimal polynomial for the splitting field of the residues, γ such that R(γ) = 0, and I(x,γ) the integral expressed in terms of γ.

(1) (A,B) = Extended_Euclidean(T(x),T'(x),S(x))

(2) R(x) = split_field (resultant(x-B(y),T(y),y))

(3) L = alg_factor(T(x),R(x),γ)

(4) L = map(Int_log,L) [applies function Int_log to each element in L]

(5) Return(R(x),γ,sum(L)) □

ALGORITHM **Int_log**

input: D(x,γ) an irreducible polynomial over $k(\gamma)$

output: c log D(x,γ)

(1) c(y) = resultant(y-B(x),D(x,γ),x)

(2) c = linsolve($c(y)/\gcd(c(y), c'(y))$) [common residue expressed in terms of γ.]

(3) return (c log D(x,γ)) $\square$

On page III-10 of his thesis, Tobey presents a rational function which he demonstrates is integrable over $\mathbf{Q}(\sqrt{2})$. He asks how one determines a priori the extension of least degree in which the integral can be expressed. Using the MACSYMA [13] system and the ideas presented in this section, we solve his problem below:

```
(C1) 'INTEGRATE(N(X)/D(X),X); /* THIS IS THE PROBLEM TO BE SOLVED.
    N(X) AND D(X) ARE RESPECTIVELY THE NUMERATOR AND DENOMINATOR
    OF TOBEY'S RATIONAL FUNCTION. */
       /    13     8      7      6      3         2
       [ 7 X   + 10 X  + 4 X  - 7 X  - 4 X  - 4 X  + 3 X + 3
(D1)   I ------------------------------------------------- DX
       ]    14      8      7      4         3      2
       /   X   - 2 X  - 2 X  - 2 X  - 4 X  - X  + 2 X + 1

(C2) D1(X):=''(DIFF(D(X),X));
                  13       7       6      3       2
(D2) D1(X) := 14 X   - 16 X  - 14 X  - 8 X  - 12 X  - 2 X + 2

(C3) (ALGEBRAIC:RATALGDENOM:TRUE,TELLRAT(D(C)))$
       /* LET C BE A ROOT OF D(X) */

(C5) B(C):=''(RATSIMP(N(C)/D1(C)));
       /* B(X) IS THE POLY COMPUTED IN STEP 1 OF RATINT */
             12    11    10    9    8    7    6      2
            C   - C   + C   - C  + C  - C  - C  - 2 C  - 2 C + 2
(D5) B(C) := ------------------------------------------------
                                    2

(C6) RESULTANT(Y-B(X),D(X),X);
           14           13           12           11           10           9
(D6) 16384 Y   - 114688 Y   + 315392 Y   - 401408 Y   + 164864 Y   + 121856 Y

          8          7          6         5         4         3        2
 - 109312 Y  - 23552 Y  + 27328 Y  + 7616 Y  - 2576 Y  - 1568 Y  - 308 Y

 - 28 Y - 1

(C7) SQFR(%); /* SQUARE FREE DECOMPOSITION IS NOT NECESSARY BUT
       MAKES THE STRUCTURE MORE EVIDENT IN THIS EXAMPLE. */
```

```
                          2          7
(D7)                   (4 Y  - 4 Y - 1)

(C8) PART(%,1);
     /* ALREADY WE HAVE FOUND THE MINIMAL POLY. FOR THE INTEGRAL */
                            2
(D8)                     4 Y  - 4 Y - 1

(C10) MP(X):=''(SUBST(X/2,Y,%));
      /* MINIMAL POLYNOMIALS ARE REQUIRED TO BE MONIC FOR EFFICIENCY */
                              2
(D10)                 MP(X) := X  - 2 X - 1

(C11) FACTOR(D(X),MP(ALG));
       7              2                7              2
(D11) (X  + (1 - ALG) X  - ALG X - 1) (X  + (ALG - 1) X  + (ALG - 2) X - 1)

(C12) (F1:PART(%,1),F2:PART(%,2))$ /* F1 AND F2 ARE THE FACTORS OF THE
                   DENOMINATOR */

(C14) TELLRAT(MP(ALG))$ /* NEXT WE CALCULATE RESIDUES OVER k(ALG) */

(C16) ALGNORM(Y-B(X),F1,X);
              7           6                    5                        4
(D16)/R/ (128 Y  - 448 ALG Y  + (1344 ALG + 672) Y  + (- 2800 ALG - 1120) Y

                     3                       2
 + (3360 ALG + 1400) Y  + (- 2436 ALG - 1008) Y  + (980 ALG + 406) Y - 169 ALG

 - 70)/128

(C17) SQFR(%);
                           7
                 (2 Y - ALG)
(D17)            ------------
                     128

(C18) SOLVE(%,Y);
SOLUTION

                       ALG
(E18)              Y = ---
                        2
MULTIPLICITY 7
(D18)                    [E18]

(C19) C1:RHS(E18); /* C1 IS THE RESIDUE AT ANY ROOT OF F1 */
```

```
                      ALG
(D19)                 ---
                       2

(C20) ALGNORM(Y-B(X),F2,X);
              7                   6                        5
(D20)/R/ (128 Y  + (448 ALG - 896) Y  + (- 1344 ALG + 3360) Y

                     4                         3                      2
 + (2800 ALG - 6720) Y  + (- 3360 ALG + 8120) Y  + (2436 ALG - 5880) Y

 + (- 980 ALG + 2366) Y + 169 ALG - 408)/128

(C21) SQFR(%);
                                7
                   (2 Y + ALG - 2)
(D21)              ----------------
                         128

(C22) SOLVE(%,Y);
SOLUTION

                      ALG - 2
(E22)           Y = - -------
                        2
MULTIPLICITY 7
(D22)                  [E22]

(C23) C2:RHS(E22); /* C2 IS RESIDUE AT ANY ROOT OF F2 */
                    ALG - 2
(D23)             - -------
                       2

(C24) C1*LOG(F1)+C2*LOG(F2); /* FINALLY WE CAN EXPRESS THE INTEGRAL */
              7             2
      ALG LOG(X  + (1 - ALG) X  - ALG X - 1)
(D24) ---------------------------------------
                        2

                                7             2
                (ALG - 2) LOG(X  + (ALG - 1) X  + (ALG - 2) X - 1)
              - ------------------------------------------------
                                        2


(C28) /* WE CAN ALSO EXPRESS THE ANSWER IN RADICALS SINCE MP(X)
    IS QUADRATIC */
```

```
    (ALGEBRAIC:FALSE,SOLVE(MP(ALG)));
SOLUTION

(E28)                ALG = 1 - SQRT(2)

(E29)                ALG = SQRT(2) + 1
(D29)                   [E28, E29]

(C31) EV(D24,E29);
                          7            2
      (SQRT(2) + 1) LOG(X  - SQRT(2) X  - (SQRT(2) + 1) X - 1)
(D31) ------------------------------------------------------
                                2

                                  7            2
              (SQRT(2) - 1) LOG(X  + SQRT(2) X  + (SQRT(2) - 1) X - 1)
            - ------------------------------------------------------
                                       2
```

Chapter 5

Integration of a Class of Algebraic Functions

In this chapter we investigate the problem of integrating algebraic functions. We define $K = k(x, y)$ to be an algebraic function field if it is a finite extension of $k(x)$ the field of rational functions in x. Any element of K is an algebraic function and satisfies a polynomial over $k(x)$ whose degree divides that of the extension.

By Liouville's Theorem, if $z \in K$ has an elementary integral, it can be expressed as the sum of an element of K and constant multiples of logs of such elements. This permits us to construct a general form for the integral. If the derivative of this form cannot be made to match the integrand, then the original problem was not integrable. Hardy [5] conjectured that there was no effective general procedure for determining the integrability of algebraic functions if log terms were present. In 1969, Risch [18] gave an algorithmic procedure for determining the integrability of functions constructed using exponential and logarithmic as well as algebraic operations. Risch constructed the pattern from local information at the singularities of the integrand and was able to reduce the problem of integration to a problem in algebraic geometry, determining a bound on the order of the torsion subgroup of the divisor class group, which he called the Points of Finite Order Problem. In 1970, he outlined a procedure for computing this bound [19], but an efficient, practical implementation is still lacking. One problem with Risch's algorithm is that the need to calculate power series expansions forces one to operate in a splitting field of very high degree even when the integral can be expressed in a field of much lower degree. The author is currently investigating methods for constructing the integral avoiding the costs of Risch's intermediate steps.

We restrict ourselves here to the class of algebraic functions which can be expressed using only a single radical. By this we mean that the function field

can be expressed as k(x,y) where $y^n = p(x)$ for some polynomial P over k whose roots have multiplicities less than n. For this class of functions we will present an algorithm for obtaining the algebraic part of the integral while operating completely in the field of definition for the integrand. Although this is a very special case of the general problem addressed by Risch, it encompasses elliptic and hyperelliptic functions and represents the most frequently encountered class of algebraic integrals.

The first obstacle to integrating algebraic functions is their multivalued nature. Instead of trying to pick a particular branch of the function, we will consider the entire function on its Riemann surface. This surface can be viewed as an n-fold covering of the extended complex plane. For our restricted class of functions, whenever $p(x) \neq 0$, the sheets are all distinct over x. Over a simple root of $p(x)$ all n sheets coalesce to form an n sheeted branch point, i.e. any path on the surface must circle this point n times before closing. If γ is a root of p of multiplicity m and d=gcd(m,n) then over γ there are d distinct branch points, each composed of n/d sheets. One of the principal advantages of considering functions on their Riemann surface is that they become single valued and meromorphic there. Thus for any particular algebraic funtion the only singularities we must contend with are poles, and these are finite in number.

We shall use the local properties of functions and their integrals as a key to their global behavior. Our principal localization technique will be power series computation. The expansion of an algebraic function at an unramified point on the surface has integer exponents and is simply a Laurent expansion. However, if we want to expand at branch points, we must permit fractional exponents. In fact if x_0 is a k-sheeted branch point, an expansion at x_0 will be in terms of $(x - x_0)^{1/k}$. These expansions with fractional exponents are known as Puiseux expansions [1]. To get a consistent definition of order for Puiseux expansions, we will express our series in terms of a uniformizing parameter. Instead of expanding in $(x - x_0)^{1/k}$, we let $x = t^k + x_0$ and expand in terms of t. The parameter t is only determined up to a kth root of unity, ω, so we define two expansions to be equivalent if one transforms into the other by substituting ωt for t. The order of an algebraic function at a place (point on the Riemann surface) **p**is the degree of the first non-zero term in its power series expansion (in t). We define $\mathrm{Ord}_{\mathbf{p}}(y)$ to be the order of y at **p**and the branch index to be the number of sheets which coalesce at **p**.

Every neighborhood of a Riemann surface is locally homeomorphic to a neighborhood in the complex plane. The uniformizing parameter supplies this mapping and thus a local coordinate system, but we have no single global coordinate system for the Riemann surface. The integral must contain the information necessary to transform between local coordinate systems along our path of integration. This is done by considering the integrand to be a differential and not a simple algebraic function. A differential is a form $f\,dx$ where f is an algebraic function, and the order of the form at a place **p**with uniformizing parameter t is defined to be $\mathrm{Ord}_{\mathbf{p}}(f) + \mathrm{Ord}_{\mathbf{p}}(dx/dt)$. Since x is locally a rational function in

t, its derivative, dx/dt, is also rational and thus has a well defined order. At a finite k-sheeted branch point dx/dt is kt^{k-1} and so $\mathrm{Ord}(f\,dx) = \mathrm{Ord}(f) + k - 1$.

Algebraic functions on their Riemann surface have many properties in common with rational functions in the extended complex plane. In particular, the following is true in both domains:

1. $\mathrm{Ord}_{\mathbf{p}}(f)$ is nonzero for only a finite set of places.

2. $\sum \mathrm{Ord}_{\mathbf{p}}(f) = 0$ if the sum is taken over all the places on the surface. This is another way of saying an algebraic function has the same number of poles and zeroes, counting multiplicities.

3. $\sum Res_{\mathbf{p}}(fdx) = 0$ with the sum again taken over all places on the surface. $Res_{\mathbf{p}}(fdx)$ is the residue of fdx and is defined to be the coefficient of t^{-1} in the series for $f\,dx/dt$.

The differentials on a Riemann surface are placed in three categories on the basis of their singularities. A differential of the first kind has no singularities on the entire Riemann surface. Differentials of the second kind have poles, but always with zero residue. Differentials of the third kind have only simple poles and thus non-zero residue.

We define a divisor on a Riemann surface to be an element of the free abelian group generated by the points of the surface, i.e. a divisor is a formal sum $\sum n_{\mathbf{p}}\mathbf{p}$ where $n_{\mathbf{p}}$ is a rational integer and is non-zero for only a finite number of points $\mathbf{p}$. The degree of a divisor is $\sum n_{\mathbf{p}}$. The divisor of an algebraic function f is defined to be $\sum \mathrm{Ord}_{\mathbf{p}}(f)\mathbf{p}$ and is denoted by (f). Since any algebraic function has the same number of poles and zeros, its divisor is of degree 0. There is a bijection between divisors of degree 0 over the extended plane and rational functions, determined up to constant multiple. On the other hand, given a divisor of degree 0 on a Riemann surface, there may be no algebraic function with precisely those poles and zeros. If there is such a function the divisor is called a principal divisor.

The operation of computing Puiseux expansions at a place provides us with a differential isomorphism from our function field into the field of power series over $\mathbf{C}$. This means that computing power series commutes with differentiation, i.e. the series for the derivative of f is the derivative of the series for f. Therefore, to obtain local information about the integral, we merely compute the power series expansion of the integrand and integrate the series termwise. If a power series has negative order, the operation of integration increases its order by one, or equivalently decreases the order of the pole by one. Integration of a power series with non-negative order produces a series with non-negative order. Thus we see that integration will never create a pole, and always decreases the order of the pole of the integrand by one.

By Liouville's theorem, $\int f\,dx = w_0 + \sum c_i \log(w_i)$ where $f, w_j \in \mathbf{C}(x, y)$, i.e. algebraic functions, and $c_i \in \mathbf{C}$. After differentiating we get $f = w_0' + \sum c_i w_i'/w_i$.

We will now examine w_i'/w_i locally to see how f decomposes. Let $\mathbf{p}$be an arbitrary place on the Riemann surface. If $\mathrm{Ord}_{\mathbf{p}}(w_i) = n \neq 0$, then $\mathrm{Ord}_{\mathbf{p}}(w_i') = n - 1$ and thus $\mathrm{Ord}(w_i'/w_i) = -1$. If $\mathrm{Ord}(w_i) = 0$, then $\mathrm{Ord}(w_i') \geq 0$ and $\mathrm{Ord}(w_i'/w_i) \geq 0$. Since $\mathrm{Ord}(f+g) \geq \min(\mathrm{Ord}(f), \mathrm{Ord}(g))$, $\mathrm{Ord}(\sum c_i w_i'/w_i) \geq -1$ everywhere on the Riemann surface.

We define the principal part of a function at a place to be the sum of all terms with negative degree in the power series expansion at that place. The principal part of a differential at a place is the sum of all terms of degree less than -1 in its expansion. Thus termwise integration of the principal part of a differential produces the principal part of the integral.

The main result of this chapter is that for our restricted class of algebraic functions, we can get a very complete description of the form of the algebraic part of the integral. In general the algebraic part would involve all powers of y less than n. By considering only function fields defined by $y^n = p(x)$, we were able to prove the following key theorem.

THEOREM 11 *5.1 The algebraic part of $\int R(x)ydx$, where $y^n = p(x)$ and $R(x)$ is a rational function in x, is of the form $yA(x)$ where $A(x)$ is a rational function in x.*

Although the statement seems very intuitive, the proof is fairly long and we will prove it in several stages. The proof is based on the relationship between the series expansions for y at all the places above a given x value. By examining the formula for computing nth roots of power series [28], we see that if $p(x_0) \neq 0$ then the n distinct expansions for y can be generated by taking any one of them and multiplying it successively by the nth roots of unity. We make the conventions that if $p(x_0) \neq 0$, we call x_0 a root of multiplicity 0 and define $\gcd(0, n) = n$. Now in general, if x_0 is a root of p(x) of multiplicity m and $d = \gcd(m, n)$ then there are d distinct places over x_0, each with branch index n/d. Two expansions of y over x_0 are equivalent if their quotient is an (n/d)th root of unity. Again starting with any particular expansion over x_0, we can generate n different expansions by multiplying by all the nth roots of unity. These n expansions can be grouped into d sets each with n/d elements, such that all the elements within a given set are equivalent, but any two expansions from distinct sets are inequivalent. This gives us a coset decomposition of the group of nth roots of unity; we consider two nth roots equivalent if their quotient is an (n/d)th root of unity. If ω is a primitive nth root of unity, we can let ω^0, ω^1, $\ldots$, ω^{d-1} be our canonical representatives of the equivalence classes. Now we can form d inequivalent expansions of y, over x_0, by multiplying any particular expansion by these d representative elements. The exponents in the terms of these expansions are all congruent to m/d modulo n/d. We summarize these results in the following lemma:

LEMMA 12 *5.1 Let x_0 be a root of $p(x)$ of multiplicity m, $g(t)$ be the power series expansion of y at some place over x_0, $d = \gcd(m, n)$. Then d inequivalent series*

expansions of y over x_0 are $g(t), \omega g(t), \ldots, \omega^{d-1} g(t)$ where ω is a primitive nth root of unity. The exponents in the expansions are all congruent to m/d modulo n/d.

In order to be able to compare the expansion of y with that of a rational function in x and y at a place, we assume the series for the rational function is computed by formal arithmetic operations on the series for y and x, as in [28]. Since a rational function of x, $R(x)$, has the same expansion at all places over x_0, we can multiply by $R(x)$ without changing the coefficient relationship among the different sheets. At an n/d sheeted branch point, the series for R(x) is a series in $t^{n/d}$, thus multiplying by R(x) won't change the congruency class, modulo n/d, of the exponents of a power series. Since the expansion for y^k is the kth power of the expansion for y, with the same assumptions as in Lemma 5.1 above, we get the following analogous result at the corresponding sheets.

LEMMA 13 *5.2 Let $R(x)$ be a rational function in x and $h(t)$ be the power series expansion for $R(x)y^k$ at the same place as $g(t)$ in Lemma 5.1. Then d inequivalent series expansions of $R(x)y^k$ over x_0 are $h(t)$, $\omega^k h(t)$, ..., $\omega^{(d-1)k} h(t)$, where ω is a primitive nth root of unity. The exponents in these expansions are all congruent to km/d modulo n/d.*

The main step in the proof of Theorem 5.1 is the following Lemma:

LEMMA 14 *5.3 Let $f \in k(x, y)$, $f = a_0 + a_1 y + \cdots + a_{n-1} y^{n-1}$. If for every x value the ratios of the principal parts of the expansions of f at any two places over x are constants which are equal to the ratios of the expansions of y at those places, and if the exponents in the expansions of f are congruent, modulo the branch index, to the exponents in the expansions of y then $f - a_1 y$ is constant, i.e. $a_2 = \cdots = a_{n-1} = 0$ and $a_0 \in k$.*

PROOF:Examine the expansions over a particular point, x_0, where we assume t is a uniformizing parameter. We will first consider the case when there are no branch points over x_0 and determine what the constraints are on the coefficients of a particular term, e.g. t^{-s}, in the expansions of the $a_j y^j$. Let b_j be the coefficient of t^{-s} in $a_j y^j$ and let c be its coefficient in the expansion of f at the first sheet. Then their respective coefficients at the (k+1)st sheet will be $\omega^{jk} b_j$ and $\omega^k c$. Thus we get a system of n linear equations in the b'_js. The coefficients of this system form a van der Monde matrix whose determinant is thus $\prod / (\omega^i - \omega^j)$ for $i \neq j$ and $i, j < n$. But since ω is a primitive nth root of unity, each term in the product is non-zero and so is the determinant. Therefore there must be a unique solution. But by inspection we can find a solution where $b_1 = c$ and all the other b_j's are 0, and it must be the only solution. Since this is true for an arbitrary term in the principal part of f, we see that the principal parts of $a_j y^j$ for $j \neq 1$ must be zero at any unramified point.

Now assume there are d distinct branch places above x_0, each with branch index $r = n/d$. Again examine the coefficients of the t^{-s} term in all the expansions. Let m be the multiplicity of x_0 as a root of $p(x)$. Then $d = \gcd(m, n)$, and, by assumption, the exponents of all the terms of the principal part of the expansions of f at x_0 are congruent to m/d modulo r. By lemma 5.2 the exponents in the expansions of $a_j y^j$ are congruent to jm/d. For a particular s there will be d values of j such that jm/d will be congruent to s, thus there are at most d monomials $a_j y^j$ whose expansions can include a term in t^{-s}. At each of the d places over x_0 we can equate the sum of these d coefficients to the coefficient of t^{-s} in f. This gives us d linear equations in d of the b_j's. If we examine the coefficient matrix and factor out the first element of each row, then the matrix we have left is van der Monde with determinant $\prod(\omega^{rj} - \omega^{rk})$ with j,k¡d. Since ω is a primitive nth root of unity, ω^r is a primitive dth root of unity, and just as in the previous paragraph, the determinant is non-zero. If s is congruent to m/d then we have the unique solution that the coefficient of t^{-s} in f is b_1 and the $d - 1$ other b's congruent to 1 mod r are all zero. If s is not congruent to m/d then the obvious unique solution is to let the d b_j's be zero. Since this is true for all terms in the principal part of f, we have shown that the principal parts of $a_j y^j$ for $j \neq 1$ are zero at all branch places. Together with the previous paragraph, this shows that their principal parts are zero everywhere, including at infinity. But this means that they have no poles, and the only algebraic functions with no poles are constants. Thus we have shown that if f is algebraic, it can differ from $a_1 y$ by at most a constant. □

With this last lemma under our belt, we are now ready to finish the proof of theorem 5.1.

PROOF:We must show that the algebraic part of $\int R(x)y\,dx$ satisfies the hypotheses of lemma 5.3. By lemma 5.2, the integrand $R(x)y$ has the required coefficient and exponent properties. It remains to show that these properties are preserved after multiplication by dx and termwise integration of the principal parts of the differential.

Since dx/dt is the same at all sheets above a given x value, multiplication by dx/dt does not disturb the coefficient relationships. The subsequent termwise integration will divide the coefficient of t^s by $s + 1$ on all the sheets over x_0, thus again preserving the ratios between sheets. If x_0 is a branch point of index k, then $dx/dt = kt^{k-1}$ and multiplication by dx/dt will change the congruency class of the exponents modulo k. But the subsequent integration will increase all the exponents by 1. Thus we have a net effect of increasing them by k and leaving their congruency class unchanged. Therefore the hypotheses of lemma 5.3 are satisfied and the algebraic part of the integral must differ from $yA(x)$ by at most a constant. □

Note that the principal part of the integral is generated by the terms of degree ¡ -1 in the expansion of the integrand while the logs arise from the rest of the expansion. Thus by concentrating on the principal parts we were able to determine the structure of the algebraic part without the possibility of interference from the log terms.

COROLLARY 15 *The derivative of the transcendental part of the* $\int R(x)y\,dx$ *is of the form* $yB(x)$ *where* $B(x)$ *is a rational function in* x.

PROOF:If the algebraic part is $yA(x)$, then its derivative is $(P'(x)A(x)/(nP(x))+A'(x))y$ and thus still of the form y times a rational function in x. Subtracting this from the integrand gives the desired result. □

By theorem 5.1 and its corollary we have shown that if $\int R(x)y\,dx$ is expressible in elementary functions, then it can be written as $A(x)y+\int B(x)y\,dx$, where the second term has no algebraic part, i.e. integrates completely into logs. Now we must investigate what the restrictions are on the rational functions $A(x)$ and $B(x)$. All the poles of the algebraic part are derived from poles in the original integrand with their order increased by one. The differential $y\,dx$ has no finite poles, so all the finite poles of $R(x)y\,dx$ arise from poles of $R(x)$. At each of these poles we have that $\mathrm{Ord}_{\mathbf{p}}(A(x)y) = Ord_{\mathbf{p}}(R(x)ydx)+1$. Since the order of a product is the sum of the orders, this reduces to $\mathrm{Ord}_{\mathbf{p}}(A) = \mathrm{Ord}_{\mathbf{p}}(Rdx)+1$. Now let $A(x)$ be $S(x)/T(x)$ and $R(x)$ be $U(x)/V(x)$ where S, T, U, and V are polynomials over k. If $\mathbf{p}$ is a root of $V(x)$ of multiplicity r, it must be a root of $T(x)$ of multiplicity $r-1$. Let $\prod V_i^i$ be a square free decomposition of V, i.e. each V_i has no multiple roots and they are pairwise relatively prime. Then $T(x)$ can be expressed as a similar product with each of the exponents decreased by one, i.e. $T(x) = \prod V_i^{i-1}$.

Next we determine the denominator of B(x). The order of the differential $B(x)y\,dx$ is ≥ -1 everywhere on the Riemann surface. If $\mathbf{p}$ is not a zero of $P(x)$, then $\mathrm{Ord}_{\mathbf{p}}(ydx) = 0$. Thus at an unramified place $B(x)$ can have at most a simple pole. If $\mathbf{p}$ has branch index r and is a zero of $P(x)$ of multiplicity m, then $\mathrm{Ord}_{\mathbf{p}}(ydx) = (m/n+1)r-1$. Thus $\mathrm{Ord}(B(x)) \geq -(m/n+1)r$. Let B(x) be C(x)/D(x) where C and D are polynomials and let $\mathrm{Mult}_{\mathbf{p}}(D(x))$ be the multiplicity of **p**as a root of D(x). Then $\mathrm{Ord}_{\mathbf{p}}(B(x)) = r(\mathrm{Mult}_{\mathbf{p}}(C(x)) - \mathrm{Mult}_{\mathbf{p}}(D(x)))$. Since we are interested in the case where $\mathrm{Mult}_{\mathbf{p}}(D(x)) \neq 0$, we can assume that $\mathrm{Mult}_{\mathbf{p}}(C(x)) = 0$. Thus we have $-r(\mathrm{Mult}_{\mathbf{p}}(D(x))) \geq -(m/n+1)r$ or equivalently $\mathrm{Mult}_{\mathbf{p}}(D(x)) \leq 1+m/n$. But m¡n and the multiplicity is always an integer, so again **p**can only be a simple root of $D(x)$. The differential $B(x)y\,dx$ can only have poles where the original integrand had poles, thus all the zeros of $D(x)$ were zeros of $T(x)$. Since $D(x)$ is square free we have that $D(x) = \prod V_i(x)$.

At this point we have determined all of the algebraic part except its numerator, $S(x)$. Using the integral equation for the algebraic and transcendental

parts, we shall derive a system of linear equations which the coefficients of the numerator polynomials, $S(x)$ and $C(x)$, must satisfy. We start with the equation:

$$\int \frac{U(x)y}{V(x)}dx = \frac{S(x)y}{T(x)} + \int \frac{C(x)y}{D(x)}dx$$

Then we differentiate both sides yielding:

$$\frac{U(x)y}{V(x)} = \frac{S'(x)y}{T(x)} - \frac{S(x)T'(x)y}{T(x)^2} + \frac{S(x)P'(x)y/n}{T(x)P(x)} + \frac{C(x)y}{D(x)}$$

Now we see the power of theorem 5.1. The radical, y, can be divided out from both sides of the equation leaving us with only rational functions. We also multiply through by $V(x) = T(x)D(x)$ and put the right side over a common denominator. To simplify the derivation, we suppress the functional notation for our polynomials.

$$U = \frac{S'TPD - ST'PD + STP'D/n + CT^2P}{TP}$$

Our next goal is to completely eliminate the denominator. We could do this by merely multiplying out both sides, but notice that $T(x)$ divides each term on the right side except the second. We now examine more closely the structure of T, D, and T'.

$$T = \prod_{i=1}^{k} V_i^{i-1}$$

$$T' = \left(\prod_{i=3}^{k} V_i^{i-2}\right)\left(\sum_{i=1}^{k}(i-1)V_2\cdots V_{i-1}V_i'V_{i+1}\cdots V_k\right)$$

$$D = \prod_{i=1}^{k} V_i$$

Now we see that we can divide T into $T'D$. We define $W(x)$ to be the quotient.

$$W = \frac{T'D}{T} = \sum_{i=1}^{k}(i-1)V_1V_2\cdots V_{i-1}V_i'V_{i+1}\cdots V_k$$

Using this definition, the computation of W appears to require both a polynomial multiplication and a polynomial division. But we can give another derivation of W which eliminates the multiplication in favor of a subtraction. Starting with the definition $V = TD$ we have:

$$\frac{V'}{T} = \frac{T'D}{T} + D'$$

$$W = \frac{T'D}{T} = \frac{V'}{T} - D'$$

The other factor of the denominator, $P(x)$, will not, in general, divide out completely, but we can cancel the $\gcd(P(x), P'(x))$ from the numerator and denominator. If we let $E = P/\gcd(P, P')$ and $F = P'/\gcd(P, P')$ then we can muliply both sides of the equation by $E(x)$ yielding the following polynomial equation:

$$U(x)E(x) = S'(x)E(x)D(x)+S(x)\left(F(x)D(x)/n - E(x)W(x)\right)+C(x)T(x)E(x)$$

The only unknowns in this equation are $S(x)$, $S'(x)$, and $C(x)$. If we let $S(x)$ and $C(x)$ be polynomials with undetermined coefficients, then we get a system of linear equations in the coefficients. All that remains is to get degree bounds on $S(x)$ and $C(x)$. We do this by examining our original integral equation at a place above ∞.

$$\mathrm{Ord}_\infty((\text{trans.part})') = \mathrm{Ord}_\infty(C(x)ydx/D(x)) \geq -1$$

$$\Longrightarrow \deg(C) \leq \deg(D) - \deg(P)/n - 1$$

$$Ord_\infty(\text{Algebraicpart}) \geq \min(\mathrm{Ord}_\infty(\text{integrand}) + 1, 0)$$

$$\Longrightarrow \mathrm{Ord}_\infty(S(x)y/T(x)) \geq \min(\mathrm{Ord}_\infty(U(x)ydx/V(x)) + 1, 0)$$

$$\Longrightarrow \deg(S) \leq \max(1 + \deg(U) - \deg(D), \deg(T) - \deg(P)/n)$$

These degree bounds complete the specification of our integration algorithm. If the linear system is inconsistent, then the original problem was not integrable. If we happen to know that our integrand has zero residue everywhere, then we can set the polynomial $D(x)$ to zero since there will be no log terms. In this case our linear system is triangular and can be solved very easily. It is interesting to note that our procedure for integrating algebraic functions reduces to Horowitz's algorithm for rational function integration if we set $P(x)$ to one.

Chapter 6

Suggestions for Future Work

The algorithms specified in chapters two and three provide a very complete nucleus for extending current algebraic manipulation systems to handle algebraic numbers and functions. The splitting field routines can be used to extend the domain of applicability of many procedures, such as Laplace transforms and definite integration as well as our application in chapter four to rational function integration. Moreover, algebraists and number theorists will find splitting field calculations an aid in determining the structure of the Galois groups of polynomials. Efficient procedures for computing these groups is a very difficult research problem. Another associated problem, is the question of whether a polynomial is solvable in radicals. Even when the structure of the Galois group is known, e.g. cyclotomic polynomials, this is a non-trivial problem.

The rational function integration algorithms in chapter four could be extended to always give real solutions by converting complex logs to arctans and arcsins. If the minimal polynomial for the result has odd degree, then the logs are already real, otherwise we can adjoin a root of $x^2 + 1$ to our splitting field so we can recognize complex conjugates.

The algebraic integration algorithm would be more useful, if it could be accompanied by a procedure for computing the log terms. This is a very difficult problem, since the integrand only contains partial information for reconstructing the logs. The author is currently investigating the possibility of finding efficient alternatives to Risch's proposed solution to this problem.

Finally, precise computing time analyses of all the algorithms presented here would be very useful. Not only would this be a way of comparing our algorithms with others, but also would provide a measure of the true costs of computing with algebraic expressions, and thus their feasibility for specific computational applications.

Bibliography

[1] Bliss, G.A., *Algebraic Functions*, American Mathematical Society Colloquium Publications XVI, 1933, reprinted Dover Pub. Inc., N.Y., 1966.

[2] Collins, G.E., "The Calculation of Multivariate Polynomial Resultants", *JACM*, vol. 18, no. 4, Oct. 1971, pp. 515-532.

[3] Eichler, M., *Introduction to the Theory of Algebraic Numbers and Functions*, tr. George Striker, Academic Press, N.Y., 1966.

[4] Gaal, L., *Classical Galois Theory with Examples*, Markham, Chicago, 1971, reprinted by Chelsea, New York.

[5] Hardy, G.H., *The Integration of Functions of a Single Variable*, Cambridge U. Press, Cambridge, England, 1916.

[6] Herstein, I.N., *Topics in Algebra*, Blaisdell, 1964.

[7] Horowitz, E.,*Algorithms for Symbolic Integration of Rational Functions*, Ph.D. Thesis, U. of Wisconsin, 1970.

[8] Knuth, D.E., *The Art of Computer Programming*, vol II, Addison-Wesley, New York, 1971.

[9] Lang, S.,*Algebra*, Addison-Wesley,Reading, MA.

[10] Loos, R. G. K., "A Constructive Approach to Algebraic Numbers", Computer Science Dept., Stanford University, Palo Alto, Calif.

[11] MacDuffee, C.C., *An Introduction to Abstract Algebra*, Dover, New York, 1966.

[12] Mack, D., *On Rational Integration*, Computer Science Dept., Utah Univ., UCP-38, 1975.

[13] *MACSYMA Reference Manual*, Mathlab Group, Project MAC, M.I.T., Cambridge, Mass., November 1975.

[14] Manove, M., Bloom, S., and Engelman, C., "Rational functions in MATHLAB", *Proc. IFIP Conf. on Symbolic Manipulation Languages*, Pisa, Italy, 1968.

[15] Moses, J., "Symbolic Integration: The Stormy Decade", *Communications of the ACM*, vol 14, no 8, pp. 548-560, 1971.

[16] Rauch, H.E. and Lebowitz, A., *Elliptic Functions, Theta Functions, and Riemann Surfaces*, Williams and Wilkins Co., Baltimore Md., 1973.

[17] Risch, R.H., "The Problem of Integration in Finite Terms", *Trans. Amer. Math. Soc.*, Vol. 139, pp. 167-189, 1969.

[18] Risch, R.H., "On the Integration of Elementary Functions which are built up using Algebraic Operations", Rep. SP-2801/002/00, System Development Corp., Santa Monica, Calif., June 1968.

[19] Risch, R.H., "The Solution of the Problem of Integration in Finite Terms", *Bull. Amer. Math. Soc.*, vol. 76, pp 605-608, 1970.

[20] Ritt, J.R., *Integration in Finite Terms*, Columbia U. Press, N.Y., 1948.

[21] Slagle, J., *A heuristic program that solves symbolic integration problems in freshman calculus*, Ph.D. diss., MIT, May 1961.

[22] Tobey, R.G., *Algorithms for Antidifferentiation of Rational Functions*, Ph.D. Thesis, Harvard, 1967.

[23] van der Waerden, B.L., *Modern Algebra*, vol 1, tr. Fred Blum, Frederick Ungar Publishing Co., New York, 1953.

[24] Wang, P.S. and Rothschild, L.P., "Factoring Multivariate Polynomials Over the Intgers," *Mathematics of Computation*, vol 29, no. 131, pp 935-950, 1975.

[25] Wang, P.S., "Factoring Multivariate Polynomials over Algebraic Number Fields", *Mathematics of Computation*, vol. 30, no. 134, April 1976.

[26] Weyl, Hermann, *Algebraic Theory of Numbers*, Princeton University Press, 1940.

[27] Yun, D.Y.Y., *The Hensel Lemma in Symbolic Manipulation*, Ph.D. Thesis, M.I.T., MAC TR-138, 1974.

[28] Zippel, R. E., "Univariate Power Series Expansions in Algebraic Manipulation", *Proceedings of SYMSAC 76*, ACM N.Y., 1976.