PART II: Research Proposal

Algorithms for the Simplification of Algebraic Formulae

1 Research Area

Computer algebra (CA) or symbolic computation, as my field is known by, consists of two principal research activities, namely, the design, implementation and analysis of algebraic algorithms and, secondly, the design of programming languages, data structures and sofware environments for implementing these algorithms.

The research problem that I propose to work on is the simplification problem. Let f be an algebraic expression (or formula) in n unknowns $(x_1, ..., x_n)$. The *simplification problem* is to compute amongst all expressions which are equal to f (at all values of the unknowns) one which is simplest, that is, of smallest size according to some metric. An important special case of the simplification problem is the *zero-equivalence problem*, that is, testing whether $f(x_1, ..., x_n) = 0$. In terms of importance, I claim that simplification is the single most important functionality provided by a computer algebra system for general users, i.e., scientists and engineers using such systems to compute and manipulate formulae.

To simplify an algebraic expression, some of the tools that we use include solutions to the following computational problems: (i) computing the greatest common divisor (GCD) of two polynomials, (ii) factoring a polynomial into irreducible factors, (iii) decomposing a polynomial, (iv) computing a Gröbner basis for a polynomial ideal, and (v) computing a prime(ary) decomposition of a polynomial ideal. We also use tools from linear algebra.

The long term goal of this proposal is to design effective algorithms and build the best software system for solving the simplification problem. I also propose to do basic research on (i), in particular, computing GCDs of multivariate polynomials over algebraic function fields and triangular sets using modular methods.

2 Background, Literature Review and Recent Progress

Let K be a constant field. We assume, as is customary, that we know how to solve the zero-equivalence problem in K. In practice, in a CA system like Maple, because one can rapidly evaluate a constant as a floating point constant to high precision, say to 50 decimal digits, one can determine if the constant is zero or not with high probability.

Let f be a function of $x_1, ..., x_n$ over K. We permit f to contain elementary and special functions, e.g., $\sqrt{1-x^2}$, e^x, $\log\log x$, $\sin(2x_1 - x_2)$, $\mathrm{erf}(x)$, and also unevaluated sums, products, and integrals. One approach to simplify $f = \log(2-2x^2) - \log(1-x) - \log(1+x)$, the approach that we will take, is to build a basis B for the functions of $x_1, ..., x_n$ appearing in f such that no basis function can be linearly expressed in terms of the others. For example,

$$B = \{\theta_1, \theta_2, \theta_3\} \quad \text{where} \quad \theta_1 = \log(2-2x^2), \theta_2 = \log(1+x), \theta_3 = \log(1-x)$$

is invalid since $\theta_1 = \theta_2 + \theta_3 + \log 2$. This is the approach taken in [15] where Risch develops a structure theorem for the elementary functions of x which gives criteria for when such

linear relations can exist between a set of logarithms, exponentials and algebraic functions, and an algorithm for finding them. Surprisingly, the algorithm to find a linear relation requires only finding a rational solution to a linear system. Risch's simplification algorithm is implemented in many CA systems because it is part of the Risch decision procedure for indefinite integration of elementary functions of x.

In [4], Caviness extends the Risch structure theorem to $n > 1$ variables. Again, the algorithm solves a linear system but this time the number of equations is usually much larger than the number of unknowns. The case $n > 1$ is not implemented in any of the commercially available CA systems as far as we know. We have made a prototype Maple implementation in [12] which uses a modular algorithm for solving the special linear systems that arise.

A problem with the Risch structure theorem is that it requires conversion of trigonometric functions to complex exponentials and arc trigonometric functions to complex logarithms. This means that if the input is a real function the output may be explicitly complex. Since it may be non-trivial to recover a real form from the output, this is user unfriendly. In [2], Bronstein developed a real analogue of the Risch structure theorem for tangents and arc tangents. Receiving output in terms of tangent and arc tangents is still unsatisfactory, however. In current work we have modified the multivariate structure theorem to permit us to work directly with the sine and cosine functions and we propose to use the same approach for other elementary and special functions.

Zero equivalence testing.

For completeness we mention two other approaches for the zero equivalence problem. Although both approaches are effective and highly interesting, we do not pursue them here.

For functions of one variable x, if we are able to develop a power series expansion about a regular point to an arbitrary order, this will readily give us a proof that $f(x)$ is not zero. It is a very interesting problem to bound the order N that we need to compute the series to so that we can correctly decide whether $f(x) = 0$ or not. Results in this area by van der Hoven, Khovanskii, and Schakell are summarized and referenced in [9]. Note, a significant advantage of this approach over the Risch approach is that it does not need to know what the algebraic relations between the elementary and special functions are.

A second approach is to evaluate $f(x_1, ..., x_n)$ at "random" points $(\alpha_1, ..., \alpha_n) \in K^n$. If we can also map the constant field K into a finite field GF(q) then this may give us a random-polynomial-time algorithm of Monte-Carlo type where the output "equals zero" is wrong with low probability. This idea was first developed by Schwartz in [16] to test if large matrices of polynomials over finite fields are singular or not. The approach was extended by Gonnet in [7] to treat non-nested exponentials, logarithms and trigonometric functions.

The problem of branch cuts.

A problem for all these approaches is that identities may not be valid for all x. E.g. $\log(x^2 + x) = \log x + \log(1 + x)$ is not true for all x, e.g. $x = -3/2$. The problem of identifying the domain for which a relation is true appears to be difficult in general. Some results on this are given by Beumont, Bradford and Davenport in [1] and in previous papers.

Another approach is to "fix" the identities so that they are true for all (complex) x. This is the approach taken by Corless and Jeffrey in [5] who introduce the *unwinding number* $K(z)$ which is (implicitly) defined by $\log e^z = z + 2\pi\, iK(z)$. Thus we have

$$\log(xy) = \log x + \log y \; + 2\pi\, iK(\log x + \log y),$$

which is now valid for all complex x, y. Thus when asked to simplify $\log(xy) - \log x - \log y$ the simplifer would return $2\pi\, iK(\log x + \log y)$ instead of 0. In attempting to integrate the unwinding number into the Risch structure theorem we found that a new problem arises, namely, how to test whether a formula involving K is zero or not. This problem is difficult and it may limit the effectiveness of this approach.

Greatest common divisors.

If we are simplifying ratios of polynomials over a field, then gcds exist and can be computed by the monic Euclidean algorithm (or with the subresultant algorithm) and then cancelled out. But an intermediate expression swell occurs which makes these algorithms ineffective for polynomials of even modest degree. In the last 35 years, much progress has been made in the development of efficient algorithms for computing multivariate polynomial GCDs over the integers, finite fields and algebraic number fields using modular methods. We mention the dense modular method of Brown [3], Wang's EEZGCD algorithm, Zippel's sparse modular GCD algorithm, Char, Geddes and Gonnet's heuristic algorithm, and Kaltofen and Trager's black box algorithm as significant contributions.

For polynomials over number fields the modular method has been investigated by Langemyr and McCallum in [10], Encarnacion in [6], and, for more than one field extension, by van Hoeij and Monagan in [8]. As far as we know there is no work in the literature on polynomial gcd computation over algebraic function fields, although, we would expect that the same techniques developed for the integer, finite field, and number field case would apply to the function field case.

In [11], as one step in the process of solving a system of polynomial equations, Lazard describes an algorithm which needs to compute polynomial gcds modulo triangular sets. Let $P = \mathbf{Q}(a_1, ..., a_k)[x_1, ..., x_n]$ and $I = \langle f_1, ..., f_m \rangle \subset P$ be an ideal in P. If each $f_i \in P[x_1, ..., x_i]$ then the set $f_1, ..., f_m$ is called a triangular set. Note, in context, the $a_1, ..., a_k$ are parameters of the polynomial system and $x_1, ..., x_n$ are unknowns being solved for.

Lazard's algorithm needs to compute gcds of univariate polynomials over the quotient ring $R = P/I$ where it will often be the case that R has zero divisors, i.e., I is not prime. In [13], Moreno-Maza and Rioboo show how to compute gcds of univariate polynomials over R. However, their algorithm is non-modular and thus results in an expression swell. One needs needs to develop a modular algorithm.

In [8] and in current work, we have developed and implemented a modular GCD algorithm for the case $k = 0$ and $m = n$. We propose to develop a modular GCD algorithm for the general case. If we succeed, this will cover almost all gcd problems that arise in practice. It includes number fields ($k = 0, m = n$, I maximal) and function fields ($k > 0, m = n$, I maximal) as special cases, for example.

What is done in practice?

All of the computer algebra (CA) systems provide facilities for zero recognition, simplification, and polynomial GCD computation. In Maple, the `normal` command provides zero recognition and simplification for rational functions and the `simplify` command performs ad-hoc simplifications on expressions involving elementary and special functions. Maple has good algorithms for multivariate polynomial GCD computation over the integers, finite fields, and number fields with one field extension. Maple is typical in this regard.

A serious design problem with existing CA systems is that, for the most part, their simplifiers are black boxes; they cannot be extended by the user to understand new mathematical objects. Thus several authors have written their own "normalizers" and "simplifiers." For example, in Maple there is the `radnormal` command (by Marc Rybowicz) and in Macsyma the `radcan` command (by Richard Fateman). Both are attempts to treat formulae containing radicals. However, these routines are not used automatically by the system, and consequently, most of the system's code does not benefit from them. In Mathematica and REDUCE users may define simplification rules which are implemented by pattern matching and applied automatically. This mechanism, although very helpful, does not guarantee zero recognition in general. The design problem for this research proposal is to to design the simplifier so that we can easily add new information (algebraic identities) about new and existing mathematical functions and still guarantee zero recognition.

3 Research Proposal

For the simplification problem we propose to build a design for a good simplifier which allows us to incrementally add new knowledge about special functions and identities to the simplifier. Our basic approach will be to incrementally build a basis for the functions in an expression. Note, the choice of the basis is usually not unique and some optimization is needed to choose the best one. For example, one basis the exponentials $\{e^{2x}, e^x, e^{x/2}\}$ is $\{\theta_1 = e^x, \theta_2 = \sqrt{\theta_1}\}$, but a better basis is $\{\theta_1 = e^{x/2}\}$.

In order that we work with the user's functions, e.g., trigonometric functions instead of converting to complex exponentials or tangents, we need to modify the Risch structure theorem to support the users representation of functions.

For the polynomial GCD problem we propose to design and implement efficient algorithms for computing the GCD of two (multivariate) polynomials over an algebraic function field $K(x_1, ..., x_n)$, and, more generally, modulo a triangular set T (involving one or more parameters). Our approach will be to map the inputs modulo one or more primes, evaluate out all parameters and polynomial variables except one, and to recover polynomial variables using (sparse) interpolation, then recover the parameters using (sparse) rational function interpolation, and finally recover the rational numbers using rational reconstruction.

The technical difficulty is to show that the algorithm will terminate correctly even when R has zero divisors. This will involve a careful analysis of the different failure cases that can occur. The new tool that needs to be developed to make this algorithm efficient is a good sparse rational function interpolation algorithm. (Good algorithms for sparse polynomial interpolation have been developed by Ben-Or, Tiwari, and Zippel. See [17].)

An Exploration for non UFDs.

Consider the trigonometric polynomial ring $D = \mathbf{Q}[s,c]/\langle s^2+c^2-1\rangle$ where $s = \sin x$ and $c = \cos x$. This ring is an integral domain but it not a unique factorization domain for $s^2 = (1-c)(1+c)$. Consequently, gcds do not always exist. Given non-zero $a, b \in D$, to simplify a fraction a/b, one could look for a common divisor of $a(s,c)$ and $b(s,c)$ of maximum total degree in (s,c) and cancel it out. In [14], Monagan and Mulholland found that this does not always lead to the simplest form. We gave an algorithm for finding a simplest form which used the tan half angle substitution. This yields a very efficient algorithm but it is not general. The extension to multiple angles will be very "messy" and the algorithm is specific to D. Perhaps there is a better way?

It appears that the tools of algebraic geometry, in particular, the ability to compute a primary decomposition of a polynomial ideal solve this problem and may be much more general. Suppose we compute the primary decomposition of the ideals

$$I = \langle a, s^2+c^2-1\rangle \text{ and } J = \langle b, s^2+c^2-1\rangle,$$

that is, we obtain

$$I = K_1 \cap ... \cap K_k, \text{ and } J = L_1 \cap ... \cap L_l.$$

Because I and J are zero-dimensional their primary decomposition is unique. Suppose we identify and remove the primary ideals in common between I and J. Suppose we obtain

$$\bar{I} = \langle f_1, ..., f_m\rangle \text{ and } \bar{J} = \langle g_1, ..., g_n\rangle.$$

Consider solving the linear system that one obtains by equating

$$\frac{c_1 f_1 + ... + c_m f_m}{g_1 + c_{m+1} g_2 + ... + c_{m+n-1} g_m} = \frac{a(s,c)}{b(s,c)} \bmod \langle s^2+c^2-1\rangle$$

We find that the solutions for $c_1, ..., c_{m+n-1}$ yield the simplest forms of $a(s,c)/b(s,c)$. We propose to explore this approach, to see how general it is and how efficient we can make it. Note, *ideal quotients* can also be used here.

4 Training Aspect of the Proposal

I presently have six graduate students and am supervising two MITACS research personnel and expect to be co-supervising a PDF in January 2004. I have budgeted (80%) of the funds for support of graduate students and travel support for them to attend conferences in the field. (Over 70% of my NSERC grant from 2000-2004 was spent on HQP). The research problems in this proposal will provide graduate students the opportunity to develop their mathematical skills and algorithm design and implementation skills.

References

[1] J. Beaumont, R. Bradford, and J. Davenport, Better Simplification of Elementary Functions Through Power Series. *Proceedings of ISSAC '03*, ACM Press, pp. 30–36, 2003.

[2] Bronstein, M. (1989), Simplification of Real Elementary Functions. *Proceeding of ISSAC '89*, ACM Press, pp. 207–211, 1989.

[3] W. S. Brown, On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors, *J. ACM* **18** (1971), pp. 476–504.

[4] Caviness B.F. On canonical forms and simplification, *J. ACM* **17**, pp. 385–396, 1970.

[5] Corless, R.M., Jeffrey D.J. The Unwinding Number. SIGSAM Bulletin, Volumne 30, No 2, pp. 15–19, 1996.

[6] M. J. Encarnacion, Computing GCDs of Polynomials over Algebraic Number Fields, *J. Symbolic Computation* **20** (1995), pp. 299–313.

[7] Gonnet G.H. Determining Equivalence of Expressions in Random Polynomial Time. *Proceedings of STOC '84*, ACM Press, 334-341, 1984.

[8] M. van Hoeij, M. B. Monagan, A Modular GCD Algorithm over Number Fields with Multiple Field Extensions. *Proceedings of ISSAC '02*, ACM Press, pp. 109–116, 2002.

[9] B. van der Hoven, A new zero test for formal power series. *Proceedings of ISSAC '02*, ACM Press, 117–122, 2002.

[10] L. Langemyr, S. McCallum, The Computation of Polynomial GCD's over an Algebraic Number Field, *J. Symbolic Computation* **8** (1989), pp. 429–448.

[11] D. Lazard, Solving Zero-dimensional Algebraic Systems. *J. Symbolic Computation,* **13**, 117–131, 1992.

[12] M. Monagan, P. Lisonek, H. Bauck, Simplification of Algebraic Expressions. MITACS Research Report: http://www.cecm.sfu.ca/CAG/products.html, November 2000.

[13] M. Moreno Maza, R. Rioboo (1995). Polynomial Gcd Computations over Towers of Algebraic Extensions, *Proc. of AAECC-11* S-V LNCS **948** (1995), pp. 365–382.

[14] J. Mulholland, M. Monagan, Algorithms for Trigonometric Polynomials. *Proceedings of ISSAC '2001*, ACM Press, pp. 245–252, 2001.

[15] Risch R. Algebraic properties of elementary functions in analysis. American J. of Math, **4**, pp. 743–759, 1975.

[16] Schwartz J.T. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, **27**, 701-717, 1980.

[17] Zippel R., Interpolating Polynomials from Their Values. *J. Symbolic Computation,* **9**, pp. 375–403, 1990.