COMPUTER ALGEBRA ALGORITHMS*

Erich Kaltofen

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York 12180-3590
Arpa-Net: kaltofen@csv.rpi.edu

January 19, 1987

CONTENTS

* To appear in *Annual Review in Computer Science*, Vol. 2, Annual Reviews Inc., Palo Alto, CA, 1987.

INTRODUCTION

The origins of the discipline of computer algebra can be found in Isaac Newton's *Universal Arithmetic* (1728) [130], where methods for manipulating universal mathematical expressions (i.e. formulas containing symbolic indeterminates) and algorithms for solving equations built with these expressions are systematically discussed. One can interpret the mission of computer algebra as the construction of computer systems that enable scientific or engineering users, for instance, to carry out mathematical manipulation automatically. Indeed, systems with this goal already exist, among them MACSYMA, MAPLE, muMATH, REDUCE, SAC/2, SCRATCHPAD/II, and SMP. These systems carry out scientific computing tasks, whose results are distinguished from numerical computing in two principle aspects. (a) The results are symbolic rather than numerical, as the typical example of the inversion of a symbolic matrix demonstrates.

$$\text{FACTOR}(\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{bmatrix}^{-1}) \rightarrow \qquad 1.$$

$$\begin{vmatrix} \frac{x_2x_3x_4}{(x_2-x_1)(x_3-x_1)(x_4-x_1)} & \frac{-x_1x_3x_4}{(x_2-x_1)(x_3-x_2)(x_4-x_2)} & \frac{x_1x_2x_4}{(x_3-x_1)(x_3-x_2)(x_4-x_3)} & \frac{-x_1x_2x_3}{(x_4-x_1)(x_4-x_2)(x_4-x_3)} \\ \frac{-x_3x_4-x_2x_4-x_2x_3}{(x_2-x_1)(x_3-x_1)(x_4-x_1)} & \frac{x_3x_4+x_1x_4+x_1x_3}{(x_2-x_1)(x_3-x_2)(x_4-x_2)} & \frac{-x_2x_4-x_1x_4-x_1x_2}{(x_3-x_1)(x_3-x_2)(x_4-x_3)} & \frac{x_2x_3+x_1x_3+x_1x_2}{(x_4-x_1)(x_4-x_2)(x_4-x_3)} \\ \frac{x_2+x_3+x_4}{(x_2-x_1)(x_3-x_1)(x_4-x_1)} & \frac{-x_1-x_3-x_4}{(x_2-x_1)(x_3-x_2)(x_4-x_2)} & \frac{x_1+x_2+x_4}{(x_3-x_1)(x_3-x_2)(x_4-x_3)} & \frac{-x_1-x_2-x_3}{(x_4-x_1)(x_4-x_2)(x_4-x_3)} \\ \frac{-1}{(x_2-x_1)(x_3-x_1)(x_4-x_1)} & \frac{1}{(x_2-x_1)(x_3-x_2)(x_4-x_2)} & \frac{-1}{(x_3-x_1)(x_3-x_2)(x_4-x_3)} & \frac{1}{(x_4-x_1)(x_4-x_2)(x_4-x_3)} \end{vmatrix}$$

(b) The results are exact; for instance, the real roots of a polynomial are computed as

$$\text{REALROOTS}(x^5 - x^3 - 2x^2 - 2x - 1, 10^{-20}) \rightarrow [x = \frac{51199051681388495 7689}{2^{68}}], \qquad 2.$$

where the actual real root is guaranteed to lie closer than 10^{-20} to the given rational answer. An important ingredient in such systems is therefore that they can handle expressions with symbolic variables and that there is no digit overflow in the number arithmetic. Hence the basic data objects are arbitrarily long integers, polynomials in several variables with rational coefficients, and matrices with polynomial entries.

Many algorithms can be performed by viewing the data objects as abstract algebraic structures; for example, Gaussian elimination is valid over any field, be it the rational numbers or Laurent series. Such algebraic algorithms and the corresponding complexity theory constitute the subject of computational algebra. Of course, this is an important part of computer algebra. Long integer arithmetic and associated operations such as integer primality testing are perhaps more number theoretical in nature, but also play a significant role in computer algebra.

Operations on approximate data such as truncated power series, and on perhaps more combinatorical objects, such as finite permutation groups, constitute yet another category of computer algebra activity. I think all algorithms discussed in this article have an impact on computerizing mathematical formula manipulation and equation solving, and many of them are available on existing computer algebra systems.

First I discuss the development of efficient arithmetic algorithms. I then list five problem areas that have led to major results and implementations: factorization, computational group theory, integration in finite terms, polynomial system solving, and theorem proving in real closed fields. Then I briefly note linkages of computer algebra to logic programming, high-level language design, artificial intelligence, and application areas outside computer science.

Computer algebra is a field of wide scope with many connections to other areas in computer science. The reader can get a more extensive introduction from several books (Lipson 1981 [111]; Buchberger et al 1982) [22] and survey articles (Yun & Stoutemyer 1981 [184]; Caviness 1986) [31]. References to computational algebra include the book by Borodin & Munro (1975) [15] and the two survey articles (Strassen 1984 [168]; Schönhage 1986b) [154]. Recent research topics are reported in Caviness (1985) [51], in Char (1986) [52], and in the *Journal of Symbolic Computation.* An article in the *Scientific American* (Pavelle et al 1981) [135] gives a less academic look at the subject.

Notation: By **Z** I denote the set of integers, by **Q** the set of rational numbers, and by **C** the set of complex numbers. By $\mathbf{F}_q$ I denote a finite field with q elements. By $D[x]$ I denote the polynomials in the variable x with coefficients from the domain D, and by $D(x)$ I denote their rational functions - that is, fractions of polynomials from $D[x]$.

ARITHMETIC

A prerequisite to carrying out computer algebra efficiently is the ability to perform arithmetic in the basic domains fast. Fortunately, today algorithms are available whose efficiency significantly surpasses the classical methods. Here I report on these algorithms not only for the integer and polynomial domains, but also for power series, floating point numbers, and matrix algebras. A main reference to this subject is Chapter 4 in Knuth's book (1981) [94].

Integer and Polynomial Addition, Multiplication, and Division with Remainder

Integers and polynomials are usually represented as lists or arrays of digits and coefficients, respectively:

$$(\pm 1, a_0, \ldots, a_n) \quad \text{denotes} \quad N = \pm \sum_{i=0}^{n} a_i r^i,\ 0 \le a_i < r,$$

and

$$(x, a_0, \ldots, a_n) \quad \text{denotes} \quad f(x) = \sum_{i=0}^{n} a_i x^i, \quad a_i \in \text{coefficient domain.}$$

The radix r is usually chosen a suitable power of 2, such that each digit fits into a word of computer memory. The school methods for addition and subtraction with running time $O(n)$ bit or coefficient domain operations suffice on a sequential computer, but for multiplication the $O(n^2)$ school algorithm can be significantly improved. The fastest methods today from both a theoretical and a practical point of view are all based on the discrete Fast Fourier Transformation (Cooley & Tukey 1965) [46]. In fact, if the coefficient domain contains primitive roots of unity, as is true for the complex numbers, polynomials can be multiplied in $O(n \log(n))$ arithmetic operations. Otherwise, such roots can be synthetically adjoined to the coefficient ring and the best known running time is the asymptotically slightly slower $O(n \log(n) \log(\log n))$ (Schönhage 1977 [149]; Nussbaumer 1980 [132]; Cantor & Kaltofen 1987) [27]. Even earlier these ideas led to the $O(n \log(n) \log(\log n))$ algorithm for integer multiplication, counting bit operations on a circuit or multitape Turing machine (Schönhage & Strassen 1971) [156]. It is a major unresolved theoretical problem to improve these running times to $O(n \log(n))$.[1] The FFT-based multiplication algorithms for both polynomial and integer multiplication are also of practical significance. In the integer case we refer especially to Pollard's "three primes" algorithm (Lipson 1981 [111], §IX.2.2) or to Schönhage's (1982a) [150] improvements.

Multivariate polynomial multiplication can be accommodated by the ubiquitous algorithmic paradigm of computing by homomorphic imaging. Figure 1 exhibits the particular instance used in that case, the Kronecker homomorphism. Other imaging schemes appear in the FFT-based multiplication algorithms, in general Chinese remaindering and interpolation (Lipson 1981 [111], §VIII.2), in the powerful Hensel lifting technique (Yun 1980 [183], and the references to earlier work there; Lauer 1982 [103]; Kaltofen 1985b [77], 1987b) [83], and in algorithms based on continued fraction (Krishnamurthy et al 1975 [96]; Wang 1981) [178] and Padé approximation (Czapor & Geddes 1984) [48].

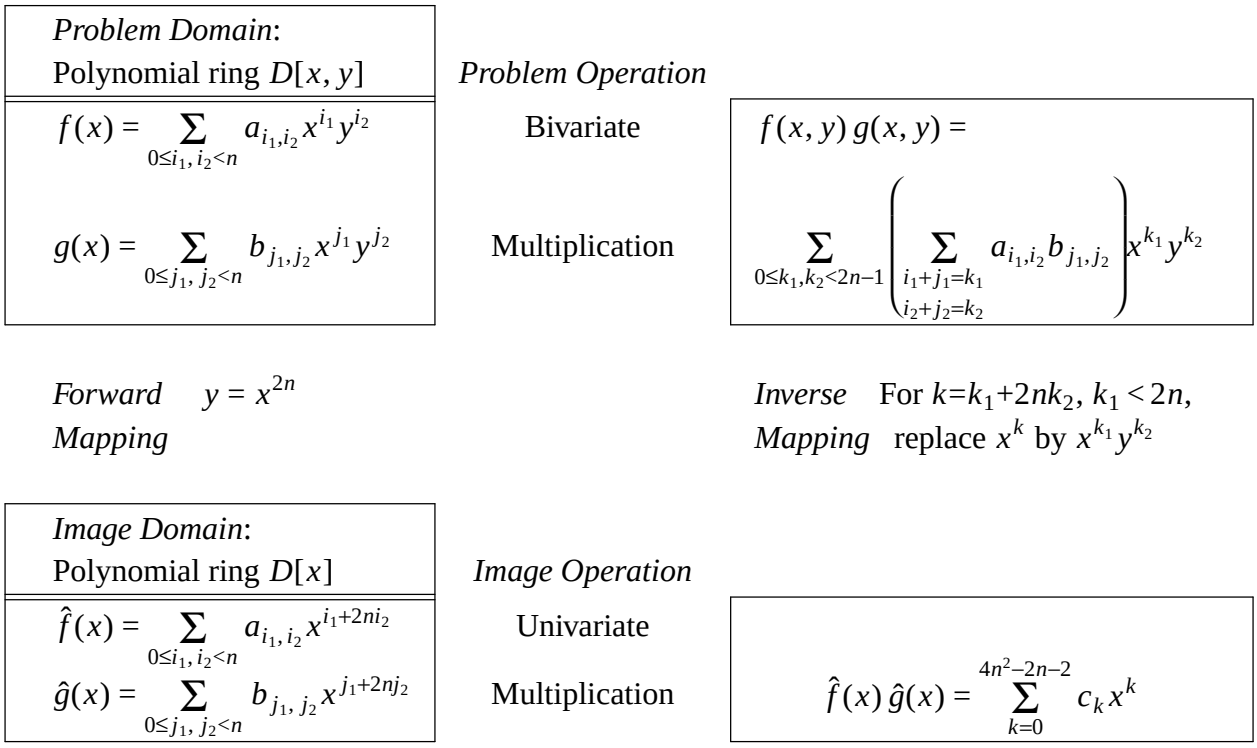
An important improvement over the school method of integer division with remainder are the fast quotient digit prediction techniques (Knuth 1981, §4.3.1) [94]. For polynomial division with remainder over the integers the novel application of Hensel lifting (Lauer 1982) [103] should prove useful.

Integer and Polynomial Greatest Common Divisor (GCD)

A fundamentally important operation in computer algebra is integer and polynomial GCD. The roots of perhaps all algorithms lie in Euclid's remainder sequence scheme. Substantial computational improvements in the case of integer GCDs are due to Lehmer (Knuth 1981, §4.5.2) [94], Knuth (1970) [93], and Schönhage (1971) [148]. These improvements carry over to polynomial GCD (Moenck 1973) [119] in terms of coefficient field operation count. However a critical problem of exponential coefficient size growth arises in case the coefficients are rational numbers or polynomials themselves, the latter in the multivariate case. The beautiful theory of subresultants (Collins 1967 [41]; Brown & Traub 1971) [18] explains that the size growth is not inherent

[1] For integer multiplication, the optimal circuit size appears to be $n \log(n)\, o(\log(\log n))$ as the hypothetical results reported by M. Fürer at the 1986 Oberwolfach complexity conference strongly suggest.

Figure 1: Multivariate polynomial multiplication by Homomorphic Imaging, the Kronecker scheme.



but a result of oversimplified rational coefficient arithmetic. Efficient algorithms are then obtained in various ways: One can keep coefficients small by exactly dividing out known common factors (Knuth 1981, §4.6.1 [94]; Hearn 1979 [68]; Stoutemyer 1985b) [164] or employ Chinese remaindering (Brown 1971) [17] or Hensel lifting (Moses & Yun 1973) [127].

It comes as a surprise that GCDs of integral polynomials can be found efficiently via integer GCDs. Clearly, if $f_1(x),\ f_2(x) \in \mathbf{Z}[x]$ and $g(x) = \mathrm{GCD}(f_1(x),\ f_2(x))$ then for any integer N

$$g(N) \quad \text{divides} \quad M = \mathrm{GCD}(f_1(N),\ f_2(N)).$$

Char et al (1984) [33] first established how to invert this mapping - that is, deduce g from M. Although the inventors viewed their method as an efficient heuristic, a rigorous probabilistic analysis has been recently accomplished (Schönhage 1986a) [155].

The extended Euclidean problem also entails the determination of multipliers s and t such that $\mathrm{GCD}(f, g) = s f + t g$, and most of the mentioned algorithms solve this problem as well. Two immediate applications are the computation of reciprocals modulo N and modulo $g(x)$ from the schemes

$$1 = \mathrm{GCD}(M, N) = SM + TN, \quad S = (M^{-1} \bmod N)$$

and

$$1 = \mathrm{GCD}(f(x), g(x)) = s(x)f(x) + t(x)g(x), \quad s(x) = (f(x)^{-1} \bmod g(x)).$$

These constitute the division operations in the basic domains of integers modulo N and of algebraic number fields in Kronecker representation $\mathbf{Q}[x]$ modulo $g(x)$ (Loos 1982) [113].

For polynomials with coefficients in an integral domain, such as $F[y]$, F a field, the Euclidean scheme can only be solved by multiplying the GCD by a domain element - that is

$$h(y)\,\mathrm{GCD}(f(x, y), g(x, y)) = s(x, y)f(x, y) + t(x, y)g(x, y),$$

where $s(x, y)$, $t(x, y) \in (F[y])[x]$, $0 \neq h(y) \in F[y]$. A particularly important multiplier $h(y)$ occurs when $\mathrm{GCD}(f, g) = 1$, namely the resultant of f and g with respect to x (van der Waerden 1953, §IV) [176]. The importance of the resultant follows from the fact that if f and g have a common zero point (x_0, y_0), then y_0 is a zero of their resultant $h(y)$, because

$$0 = f(x_0, y_0)\, s(x_0, y_0) + g(x_0, y_0)\, t(x_0, y_0) = 1 \cdot h(y_0).$$

Therefore, intersection points of f and g, be they real or complex, are projected to roots of their resultant. Clearly, efficient algorithms to compute resultants are connected to the GCD problem and we refer to Collins (1971) [42], Schwartz (1980) [157], and Rothstein (1984) [145].

Once the GCD algorithms are in place we are in the position of performing exact rational number and function arithmetic ("slash arithmetic") and keep the explicitly found numerators and denominators in lowest terms (Knuth 1981 [94], §4.5.1). However, slash arithmetic is likely to lose in efficiency to computing in homomorphic images such as modular or approximating domains.

Floating Point Numbers and Power Series

Although truncated formal power series are well accepted objects in computer algebra (Lipson 1981 [111], §IX.3; Zippel 1976 [187]; Knuth 1981, §4.7) [94], floating point numbers may be thought to belong to numerical computing, where the exactness principle of computer algebra is replaced by numerical error analysis. Indeed, both the floating point number and power series domains can be considered what Knuth calls seminumerical data. The exactness principle must still apply as such data is manipulated in computer algebra, and that means for the power series domain that the computed Taylor series coefficients are correct up to the indicated truncation point. When using floating point numbers in computer algebra we shall at least require that the numbers returned as answers must be guaranteed correct within an explicitly stated tolerance. Computer algebra systems therefore should support floating point operations with arbitrary settable mantissa length, and indeed several systems do. A good example of a problem for this principle is complex root approximation of a univariate polynomial, where the ability to select the floating point precision dependent on the input polynomial is important to guaranteeing the desired tolerance. Solutions to this problem go as far back as to Sturm in 1835, and Schönhage (1982b) [151] recently developed a comprehensive complexity theory for this fundamental

problem of algebra. Earlier in 1976, Collins & Loos had already established that for real root isolation Newton iteration is computationally superior to the Sturm bisection method, which was used to find the answer to Equation 2, above. I refer to Collins & Loos (1982) [44] for a full account on modern methods.

Several problems in computer algebra, such as computing polynomial GCDs or Jordan normal forms of matrices over **C** represented as floating point numbers are considered "inherently numerically unstable." Schönhage's (1985) [153] quasi-GCD algorithm is a step towards dealing with such problems. Obviously, the semi-numerical approach to computer algebra problem solving must incorporate numerical analysis techniques and is at this time restricted to a few problems. However, if a difference scheme leads to a linear system best solved by a conjugate gradient method, this should be done, keeping in mind that a different notion of error analysis applies.

Power series manipulation is probably better understood, perhaps for the lack of carry propagation in the arithmetic. Here I only add new information to the discussion by Knuth (1981 [94], §4.7). The coefficient growth in the Newton algorithm for power series expansion of algebraic functions (Kung & Traub 1978) [98] has been analyzed (Kaltofen 1985c [78]; Chistov 1986) [37]. It turns out that for a fixed algebraic function the expansion can be computed in linear time in the requested order (Chudnovsky & Chudnovsky 1986) [40]. Closely related to power series are the p-adic numbers (Krisnamurthy et al 1975) [96], which can be used as an alternative to floating point numbers to approximate certain complex numbers.

Matrix Arithmetic

Algorithms for manipulating matrices form the basis to linear system solving. As mentioned, the great Gaussian elimination algorithm is generic in that it can be carried out over an abstract field. As such, Gaussian elimination constitutes the basis for linear system solving in computer algebra, and homomorphic imaging can then be applied to it (McClellan 1973 [116]; Moenck & Carter 1979) [120].

Theoretically, it turns out that the complexity of many matrix problems is reducible to the complexity of n by n matrix multiplication (Aho et al 1974) [5]. Now it is well-known that n by n matrices can be multiplied asymptotically faster than in $O(n^3)$ arithmetic steps. In Knuth (1981 [94], §4.6.4), the developments leading to an $O(n^{2.498})$ algorithm are described. Recently, Strassen (1986) [169] has introduced the new "LASER" technique for aligning a nonmultiplication tensor, which has led to the new world record of $O(n^{2.376})$ by Coppersmith & Winograd (1987) [47]. Although the actual algorithms are purely of theoretical interest, the innovations such as the border rank or the asymptotic spectrum of a tensor can be expected to have an impact in computational algebra.

Linear algebra appears to be an important subject of advanced studies in computer algebra, for it is part of the solution to many problems, be it to compute a Taylor series solution to a differential equation or to factor integers by the continued fraction method, to name but two very

distant ones. The work on computing matrix canonical forms (Kaltofen et al 1986 [85], 1987) [86], as well as Wiedemann's (1986) [180] spectral methods for sparse linear system solving over finite fields demonstrates randomization as a powerful new tool in this area. The control of coefficient growth in diophantine linear system solving (Kannan & Bachem 1981 [88], Chou & Collins 1982) [39], as well as the lattice reduction algorithm (Kannan, in this volume) arises as an important issue. And finally, methods to manipulate sparse matrices must be introduced to computer algebra. Both the combinatorical approaches (Lipton et al 1979 [112]; Gentleman & Johnson 1976 [62]; Pan & Reif 1985) [134] and the algebraic approach (Wiedemann 1986) [180] have high potential to improve significantly the linear system solving capability within computer algebra.

Arithmetic with Concisely Represented Expressions

Sparse polynomials certainly play as important a role in the polynomial calculus as do sparse matrices in large linear system solving. A representation at hand is the vector of nonzero terms together with the corresponding exponents

$$((a_{e_1,\ldots,e_n}, e_1,\ldots,e_n))_{(e_1,\ldots,e_n)\in J} \quad \text{denotes} \quad \sum_{(e_1,\ldots,e_n)\in J} a_{e_1,\ldots,e_n}\, x_1^{e_1}\cdot\cdot\cdot x_n^{e_n},$$

where $a_{e_1,\ldots,e_n} \neq 0$ for $(e_1,\ldots,e_n) \in J$. A main consideration in addition and multiplication is the sorting of the terms in the answer (Horowitz 1975) [71]. However, the problem of deciding whether polynomials have a common nontrivial divisor becomes NP-hard for even univariate inputs, if exponents are represented in binary (Plaisted 1977) [136]. In general, allowing very large exponents spoils the ability to manipulate such "ultrasparse" polynomials. Neither can the polynomial x^{2^d} be evaluated at say $x = 3$, nor can $x^{2^d} - 1$ be divided by $x - 1$, for in both cases the inputs are of size $O(d)$, whereas the outputs require $\Omega(2^d)$ bits.

The sparse representation becomes more natural if the number of variables n is large, yet the degree stays polynomially bounded. There are $\binom{n+d}{n}$ terms $x_1^{e_1}\cdot\cdot\cdot x_n^{e_n}$ of total degree $e_1 + \ldots + e_n \leq d$ - that is, exponentially many in n, even though the number of nonzero terms can be quite small. GCD and factorization algorithms on such multivariate sparse polynomials were studied both theoretically and empirically as early as a decade ago (Wang 1978) [177], and randomization became one of the major ingredients to sparsity preserving operations (Zippel 1979) [188]. Early on Moses (1971b) [125] pointed out, however, that sparsity is not measured in terms of nonzero coefficients alone, as his example

$$\int (z+1)^{1000}\mathrm{d}z = \frac{(z+1)^{1001}}{1001} \qquad 3.$$

demonstrates. From algebraic complexity theory (Strassen 1973a [166], b [167]; Valiant 1982) [174] one can adopt straight-line programs as a concise representation of polynomials. Figure 2 exhibits the straight-line representation for a very dense multivariate polynomial. Moses's example (Expression 3) can be recast in these terms as: "Given a straight-line program of length l for a

Figure 2: Straight-line program example for the expression

$$\frac{1}{x^6}\left((xa^2+xb^2+xc^2-xw^2-xy^2-xz^2-xa-2xb+3xc-4xw+5xy-6xz+10x^2-5x)^6-x^{18}\right).$$

Following is the program produced for the above expression by the conversion procedure of Freeman et al (1986) [57].

v1:=0	v16:=a	v31:=v28*v28	v46:=v43*v43	v61:=v36+v60
v2:=1	v17:=v16*v3	v32:=v31*v3	v47:=v3*v46	v62:=v32+v61
v3:=x	v18:=v8*v17	v33:=−4	v48:=v8*v47	v63:=v30+v62
v4:=v3*v3	v19:=v16*v16	v34:=w	v49:=−6	v64:=v26+v63
v5:=v3*v4	v20:=v19*v3	v35:=v34*v3	v50:=z	v65:=v24+v64
v6:=v5*v5	v21:=−2	v36:=v33*v35	v51:=v3*v50	v66:=v20+v65
v7:=v2/v6	v22:=b	v37:=v34*v34	v52:=v49*v51	v67:=v18+v66
v8:=−1	v23:=v22*v3	v38:=v37*v3	v53:=v50*v50	v68:=v15+v67
v9:=v4*v4	v24:=v21*v23	v39:=v8*v38	v54:=v3*v53	v69:=v68*v68
v10:=v9*v9	v25:=v22*v22	v40:=10	v55:=v8*v54	v70:=v68*v69
v11:=v3*v10	v26:=v25*v3	v41:=v40*v4	v56:=v52+v55	v71:=v70*v70
v12:=v11*v11	v27:=3	v42:=5	v57:=v48+v56	v72:=v13+v71
v13:=v8*v12	v28:=c	v43:=y	v58:=v45+v57	v73:=v7*v72
v14:=−5	v29:=v28*v3	v44:=v3*v43	v59:=v41+v58	
v15:=v14*v3	v30:=v27*v29	v45:=v42*v44	v60:=v39+v59	

polynomial function $f(z) \in \mathbf{C}[z]$, find another straight-line program of length $l^{O(1)}$ that computes $\int f(z)\mathrm{d}z$." However, it is not even clear that such programs always exist (Strassen 1986 [168], Problem X). Again, the high degree may be held responsible for the difficulty of this integration problem. This is not the case in Valiant's (1982) [174] multivariate example:

$$\frac{\partial^n}{\partial y_1 \cdots \partial y_n}\left(\prod_{i=1}^{n}\left(\sum_{j=1}^{n} x_{i,j}y_j\right)\right) = \mathrm{PERMANENT}\left(\begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & & \vdots \\ x_{n,1} & \cdots & x_{n,n} \end{bmatrix}\right). \qquad 4.$$

Clearly, the argument to the iterated partial derivatives in Expression 4 has a straight-line computation of length $O(n^2)$, whereas the general permanent on the right-hand side constitutes a #P complete problem (Valiant 1979) [173]. Therefore partial derivatives of functions given by straight-line programs are difficult to compute. I like to look upon Expression 4 as a central infeasible computer algebra problem, and Moses's problem may fall into this category as well.

Despite Valiant's permanent example, the polynomial factoring problem of multivariate polynomials in straight-line representation such as the one in Figure 2 could be shown feasible in even the sense of efficient straight-line answer generation (discussed in the next section). Aside from this, there are several other basic operations known to be feasible on multivariate polynomials of polynomially bounded degree given by straight-line programs, such as automatic parallelization (Valiant et al 1983 [175], Miller et al 1986) [118], GCD, separation of numerator and denominator of a rational function, and multiple partial derivatives in a single variable (Kaltofen 1987b) [84].

SEMINAL PROBLEMS

Five problems have led to a substance of work in computer algebra not found for any other of its problem areas, if we exclude the arithmetic discussed before. Each of these has also initiated a major implementation effort.

Factorization

The two domains basic to factorization are the integers **Z**, and multivariate polynomials over a field $F[x_1, \ldots, x_n]$. In the latter, F is typically an algebraic extension of the rationals or the integers modulo a prime number. Integer factorization is, strictly speaking, a problem in computational number theory, and many algorithms utilize number theoretic properties. Refer to the survey by Pomerance (1984) [137] for the work until 1984. Recently, the use of elliptic and hyperelliptic curves has led to a new approach in integer factoring and primality testing (Lenstra 1986 [109]; Goldwasser & Kilian 1986 [63]; Adleman & Huang 1987) [3]. This approach leads to both a randomized primality test that certifies its inputs $\leq N$ to be prime and has $(\log N)^{O(1)}$ expected running time, and to a factorization procedure that has constant space requirement and works well on numbers with fairly many factors (Montgomery 1987) [122]. Previously known primality testing procedures establish the primality of their inputs either with overwhelming probability of being right [that probability being independent of the input number (Solovay & Strassen 1977 [163]; Rabin 1980a)] [138], or with correctness that is subject to an unproven extension of the Riemann hypothesis (Miller 1976 [117]; Bach 1985) [9]. It is fair to say that today prime numbers can be recognized and generated feasibly in both a strong theoretical and practical sense. These developments are definitely useful in computer algebra tasks - for instance, for probabilistically verifying polynomial identities (Schwartz 1980) [157]. On the other hand, the fact remains that factoring an arbitrary integer $\leq N$ by any of the fastest known probabilistic methods requires at least

$$\exp(\sqrt{\log(N)\log(\log N)}\,)^{1+o(1)}$$

steps. This computational difficulty of integer factoring has led to the belief that integer multiplication is an irreversible process with respect to the resources required to undo it. Modern cryptography is based on this hypothesis, and with it encoding schemes could be proven secure (Goldwasser & Micali 1984) [64].

We now come to the problem of factoring polynomials. Isaac Newton's classical approach (see van der Waerden 1953) [176] reduces the problem to integer factorization, but the problem of factoring the arising integers remains computationally difficult. For evidence, see Monagan (1986) [121], who provides an irreducibility test based on Newton's method. It is a fine example of computer algebra work to remove this exponential complexity from the polynomial factoring problem and today multivariate polynomial factorization is theoretically and practically feasible. Consider the following far reaching theorem.

Theorem (Kaltofen 1987a) [80]: *Let F be an algebraic extension of a prime field - that is, either $F = \mathbf{Q}[\theta]$ modulo $m(\theta)$, where $m(\theta) \in \mathbf{Q}[\theta]$ is the defining minimal polynomial of the algebraic number θ, or $F = \mathbf{F}_q$, q a prime power. Assume we are given a straight-line program P of length l that computes a polynomial $f \in F[x_1,\ldots, x_n]$, a failure probability $\varepsilon << 1$, and a maximum number of terms bound t. Then we can find in polynomially many binary steps as a function of*

l; t; $\log(1/\varepsilon)$;
element-size(P), *the binary size of the scalars in P;*
deg(f), *the total degree of f;*
coeff-size(f), *the number of bits needed to represent the coefficients of f in F;*

sparse polynomials that with probability $> 1 - \varepsilon$ constitute all irreducible factors of f with no more than t monomials.

In fact, the algorithm first finds straight-line programs for all irreducible factors of f and then converts those programs to sparse representation, with abortion if the polynomials have more than t terms. A consequence is, for example, that the factorization of the polynomial in Figure 2 can be found automatically even though the polynomial has 14,379 terms. I consider the above theorem to be the culmination to date of work begun in the mid-1950s on the polynomial factoring problem. Instead of referring to the extensive literature available now, I note below what I consider the stepping stones toward the above theorem. For more detailed information, consult the two survey articles (Kaltofen 1982c [74], 1986c) [82].

1. A reduction from factoring in $\mathbf{Q}(\theta)[x]$ to factoring in $\mathbf{Q}[x]$ is given (Kronecker 1882 [97]; Trager 1976 [170]; Landau 1985) [100]. This is the only classical algorithm of polynomial running time, although the lattice method (Lenstra 1982 [106]; Abbot et al 1986) [1] may be more efficient.

2. The factoring problem in $\mathbf{F}_q[x]$ is resolved in polynomial time, using randomization for large characteristic (Butler 1954 [24]; Berlekamp 1967 [11]; 1970 [12]; Rabin 1980b [139]; Cantor & Zassenhaus 1981) [28].

3. The Hensel lemma is applied to reconstructing rational factors from modulo p factors and multivariate factors from univariate ones (Zassenhaus 1969 [186]; Musser 1975 [128]; Yun 1980) [183].

4. An interpolation and lifting scheme for sparse polynomials is introduced using randomization (Zippel 1979 [188], 1981 [189]; von zur Gathen & Kaltofen 1985 [61]; Kaltofen 1985b) [77]

5. Effective versions of the Hilbert irreducibility theorem are employed to guarantee the correctness of the factors reconstructed by Hensel lifting (Heintz & Sieveking 1981 [69]; Kaltofen 1982a [75], 1985a [79], 1985c [78]; von zur Gathen 1985) [59].

6. The lattice algorithm (see Kannan, this volume) is employed to obtain a polynomial-time factorization algorithm in $\mathbf{Q}[x]$ (Lenstra et al 1982 [108]; Schönhage 1984) [152].

7. Factoring multivariate polynomials over the integers is reduced to univariate algorithms. The algorithms are exponential in the number of variables (Kaltofen 1982b [76], 1985c [78]; Chistov & Grigoryev 1982 [38]; Lenstra 1984) [107].

8. Straight-line programs are used to represent dense polynomials as inputs, intermediate results, and final answers (Kaltofen 1986a [81], 1987a) [80].

I have not given the asymptotic complexity associated with the theorem. I believe that any crude upper bound would be quite useless in judging the practical feasibility of any of the algorithms. Fortunately, fine-tuned implementations of most methods are available (Wang 1978 [177]; Moore & Norman 1981 [123]; Freeman et al 1986) [57] and perform remarkably well. If the reader has a polynomial needing to be factored, I estimate that with 98% chance the appropriate procedure will be able to do the job. For instance, the factors of the polynomial in Figure 2 were found by our system in 100 seconds. I believe that no other of our seminal problems, with the exception perhaps of some of the group problems discussed next, is as far advanced towards its computational solution.

Computing with Groups

The discovery of the interconnection between finite group theory and solving a single equation by radicals, known as Galois theory, is already 150 years old (van der Waerden 1953, Ch. 7) [176]. The activity on the polynomial factorization problem led to a polynomial-time solution for deciding whether the Galois group of a rational polynomial is solvable (Landau & Miller 1984) [101]. I hope this result will open a computational approach to classical Galois theory. However, the fundamental question of finding the Galois group of a polynomial in $\mathbf{Q}[x]$ in polynomial-time remains unsolved. Since the group can have exponentially many elements in the degree of the equation, we must ask for a concise representation, say a set of permutations generating the group. Notice that there always exists a generating set whose cardinality is the logarithm of the order of the group. Given a generating set of a permutation group, problems like

Cardinality, Membership, Solvability,

are all solvable in polynomial-time. Such results are precisely the theme of the well developed computational group theory (see Neubüser 1982 [129] and the references there). Computer

systems such as CAYLEY (Cannon 1976) [26] and CAMAC (Leon & Pless 1979) [110] provide sophisticated procedures to actually compute such answers.

Computational group theory does not lack of challenging problems such as finding a generating set for the intersection of two permutation groups given by generating sets. Although some applications of this theory are fairly combinatorial, as the relation to the graph isomorphism problem may indicate (Luks 1982) [114], the link to equation solving grows even stronger if we include matrix groups. The theory by Picard & Vessiot relates the question of finding Liouvillian solutions of linear ordinary differential equations to the solvability question of (infinite) matrix groups. Recently, this theory has been exploited to construct the remarkable decision procedures by Kovacic (1986) [95] and Singer (1981) [158]. The computational complexity of problems relating to groups multiplicatively generated by matrices over even finite fields must be expected to be quite high (Babai & Szemerédi 1984) [8], and the same may be true for the general case in the Singer algorithm. Nonetheless, the group theory in Kovacic's algorithm is simple enough to make it worthwhile to implement it (Saunders 1981 [147]; Smith 1984) [161]. For the special ODE $\mathrm{d}y/\mathrm{d}z = f$, or $y = \int f\,\mathrm{d}z$, we need no group theory but the methods are not less beautiful, as we see next.

Integration in Closed Form

The calculation of a "closed form expression" whose derivative equals a given expression, such as

$$\int \log(\exp(z)+1) + \frac{z\exp(z)}{\exp(z)+1}\,\mathrm{d}z \to z\log(\exp(z)+1), \qquad 5.$$

is the subject of extensive study in computer algebra. As the student of ordinary calculus learns, there are several heuristics to aid the solution of such a problem, like substitution or integration by parts. These heuristics entered into the artificial intelligence approach of the pioneers in computerizing integration (Slagle 1961 [160]; Moses 1967) [124], but it is Risch's (1969) [140] ingenious recursive descent argument that replaces the heuristics by a full-fledged algorithm. As is known in the folklore of calculus, certain functions like $\sin(z)/z$ do not allow closed form integration in some sense, which we can make precise. Let $C \subset \mathbf{C}$ and let

$$F_0 = C(z) \subset F_1 \subset F_2 \subset \cdots \subset F_n$$

be a tower of complex function fields such that

$$F_{i+1} = F_i(\theta_i) \text{ where } \begin{cases} \theta_i = \log(\lambda_i),\ \lambda_i \in F_i, \text{ or} \\ \theta_i = \exp(\eta_i),\ \eta_i \in F_i, \text{ or} \\ \theta_i \text{ is algebraic over } F_i. \end{cases}$$

Then F_n is called an *elementary Liouvillian extension* of $C(z)$. If no extensions are by algebraics then we call F_n *purely transcendental*. We call the constant field C *computable* if we have

effective procedures for carrying out arithmetic and testing elements for zero and for being integral. In our Expression 5,

$$C = \mathbf{Q} \subset \mathbf{Q}(z) \subset \mathbf{Q}(z, \exp(z)) \subset \mathbf{Q}(z, \exp(z), \log(\exp(z) + 1)).$$

Here is the theorem.

Theorem (Risch 1969 [140]; Rothstein 1976) [144]: *Given $f \in K$, K a purely transcendental elementary Liouville extension of* $\mathbf{C}(z)$*, and given the constant field of K as a computable subfield of* $\mathbf{C}$*, we can decide in finitely many steps whether there exists an elementary Liouville extension $L \supset K$ and a $y \in L$ such that* $\mathrm{d}y/\mathrm{d}z = f$*. In that case we call f elementary integrable and the algorithm will also produce such a y.*

All trigonometric, hyperbolic, and inverse tangent functions lie in such fields K, and this theorem therefore covers a wide range of functions. Surprisingly, the approach is purely algebraic and in fact uses multivariate polynomial domains. Clearly, K is isomorphic as a field to $C(z, x_1, \ldots, x_n)$ on which we can define a formal derivative

$$z' = 1,\ x_i' = \begin{cases} \lambda_i'/\lambda_i & \text{if } x_i \text{ is the image of } \log(\lambda_i), \\ x_i \eta_i' & \text{if } x_i \text{ is the image of } \exp(\eta_i). \end{cases}$$

With this derivative $C(z, x_1, \ldots, x_n)$ becomes a so-called differential field. An 1834 theorem by Liouville now allows us to restrict L and the structure of y. An entirely algebraic proof was devised by Rosenlicht (1968) [143]. Expanding on these ideas (see also Ostrowski 1946) [133], Risch could reduce the slightly more general ODE problem $y' + f\,y = g$, f, g, and the solution $y \in C(z, x_1, \ldots, x_n)$, to integration and corresponding ODE problems in $C(z, x_1, \ldots, x_{n-1})$. Rothstein (1976) [144] significantly improved Risch's algorithm by showing how to find the minimal algebraic extension of K necessary to express y [in the case $K = C(z)$ see also Trager (1976)] [170]. Once the algorithm is in place, and has been implemented (Moses 1971a [126]; Norman & Moore 1977) [131], it quickly solves Expression 5 or determines $\sin(z)/z = (e^{\mathbf{i}z} - e^{-\mathbf{i}z})/(2z\mathbf{i})$, $\mathbf{i} = \sqrt{-1}$, nonelementary. Attempts have also been made to replace the recursive descent by computing an a priori degree bound (Fitch 1981 [55]; Davenport & Trager 1985) [50], but universally valid such bounds appear excessively high.

If K in the above theorem is allowed also to contain algebraic functions, the theory becomes substantially more involved and algebraic geometric in flavor. However, Risch (1970) [141] announced a finite decision method in this case as well. Davenport (1981) [49] and Trager (1984) [171] have since then extensively studied this case, and a procedure for integrands built by purely quadratic extensions of $\mathbf{Q}(z)$ by Davenport exists for the system REDUCE.

Although $\int \exp(z^2)\mathrm{d}z$ is not elementary we can express it as $-\sqrt{\pi}\,\mathbf{i}\,\mathrm{erf}(\mathbf{i}z)/2$. It is natural to ask which other nonelementary integrals can now be expressed with the additional help of error function extensions. Generally, we may allow special functions as extensions from K to L that satisfy certain differential equations. In case of $y = \mathrm{erf}(g)$ this is $y' = g' \exp(-g^2)$, ignoring the

scaling factor. Decision procedures based on Liouville type theorems and the Risch approach have been devised in this setting, but here we can only refer to Singer et al (1985) [159], Cherry (1985 [36], 1986) [35], and Knowles (1986) [92].

In modeling elementary functions by expressions, the tower of transcendental extensions has to be built before the Risch algorithm can be applied. That means in particular that we must have a transcendence test for exponentials and logarithms. Known algorithms for these are based on so called structure theorems for elementary functions (Epstein & Caviness 1979 [53]; Rothstein & Caviness 1979 [146]; Risch 1979) [142]. The computability of arithmetic in the arising constant fields can become a problem. I note that even in $\mathbf{Q}(\exp(1), \pi)$ zero-testing hinges on the hypothesis of algebraic independence of exp(1) and π. The interpretation of expressions as meromorphic functions must also account for suitable branch selection of multivalued functions (such as a log), and an appropriate representation for this mathematical knowledge remains a challenge. However, once the embedding into a tower of fields is found, the integration algorithms are independent of the particular meromorphic model of these differential fields (Risch 1969 [140], Proposition 2.3).

There is a remarkable similarity between the problem of integration in closed form and the problem of summation in closed form, which we only mention in passing (Gosper 1978 [66]; Karr 1985) [91].

Solving Systems of Polynomial Equations

In the later half of the last century, the theory of invariants constituted a main subject of mathematical research. The fundamental insight to solving systems of polynomial equation was obtained in this setting, the Hilbert Nullstellensatz. Let us first establish a bit of modern terminology. The algebraic variety of a system of polynomial equations $f_1,\ldots, f_r \in \mathbf{C}[x_1,\ldots, x_n]$ is the set of their common zeros,

$$V(f_1,\ldots, f_r) := \{(a_1,\ldots, a_n) \in \mathbf{C}^n \mid f_1(a_1,\ldots, a_n) = \cdots = f_r(a_1,\ldots, a_n) = 0\}.$$

The ideal generated by $f_1 ,\ldots, f_r$ over $\mathbf{C}[x_1 ,\ldots, x_n]$ is the set of all linear combinations

$$(f_1,\ldots, f_r) := \{h_1 f_1 + \cdots + h_r f_r \mid h_1,\ldots, h_r \in \mathbf{C}[x_1,\ldots, x_n]\}.$$

Now a version of the Nullstellensatz states that

$$\forall(a_1,\ldots, a_n) \in V(f_1,\ldots, f_r)\colon g(a_1,\ldots, a_n) = 0$$
$$\Leftrightarrow 1 \in (f_1,\ldots, f_r, x_{n+1}g - 1) \text{ over } \mathbf{C}[x_1,\ldots, x_{n+1}].$$

In particular,

$$V(f_1,\ldots, f_r) = \varnothing \Leftrightarrow 1 \in (f_1,\ldots, f_r) \text{ over } \mathbf{C}[x_1,\ldots, x_n]. \qquad 6.$$

By Expression 6 solvability of polynomial systems is related to polynomial ideal membership, the “Hauptproblem:”

Given $f_1,\ldots, f_r, g \in F[x_1,\ldots, x_n]$, F a field, is $g \stackrel{?}{\in} (f_1,\ldots, f_r)$

☞☞ $\exists\, h_1,\ldots, h_r \in F[x_1,\ldots, x_n]$: $g = \sum_{i=1}^{r} h_i f_i$.

The decidability of the ideal membership is long known (Hermann 1926) [70], because the existence of h_i with $\deg(h_i) \le \deg(g) + (rd)^{2^n}$, where $d = \max\{\deg(f_i)\}$, in case of membership can be guaranteed and reduces the problem to solving a (large) linear system in the unknown term coefficients of the h_i. The degree bound has been improved by Lazard (1977) [104] to $O((r \max\{d, \deg(g)\})^{3^{n/2}})$. In 1965 Buchberger invented an important new approach to polynomial ideal manipulation. First the generating set $f_1 ,\ldots, f_r$ is rewritten into a new basis $f_1^* ,\ldots, f_s^*$ for the same ideal, which has strong completeness properties. In particular, with respect to this so-called Gröbner or standard basis and with a generalized polynomial remainder process we have

$$g \in (f_1,\ldots, f_r) \text{☞☞ REMAINDER}(g;\ f_1^*,\ldots,\ f_s^*) = 0.$$

Another property of Gröbner bases is best demonstrated by an example (Trinks 1978)172 . For the input basis, for instance,

$$\begin{aligned}
f_1 &= 45p + 35s - 165b - 36,\\
f_2 &= 35p + 40z + 25t - 27s,\\
f_3 &= 15w + 251 + 30z - 18t - 165b^2,\\
f_4 &= -9w + 15pt + 20zs,\\
f_5 &= wp + 2zt - 11b^2,\\
f_6 &= 99w - 11sb + 3b^2,\\
f_7 &= b^2 - 33/50b + 2673/10000,
\end{aligned}$$

one obtains under lexicographical ordering with $w > p > z > t > s > b$ a Gröbner basis in which the variables are “diagonalized” such that the finitely many solutions can be computed by back-substitution:

$$\begin{aligned}
f_1^* &= w + 19/120\, b + 1323/20000,\\
f_2^* &= p - 31/18\, b - 153/200,\\
f_3^* &= z + 49/36\, b + 1143/2000,\\
f_4^* &= t - 37/15\, b + 27/250,\\
f_5^* &= s - 5/2\, b - 9/200,\\
f_6^* &= f_7.
\end{aligned}$$

Gröbner bases can be used as a tool for many polynomial ideal theoretic operations, which fortunately has recently been surveyed in the article (Buchberger 1985a) [20].

The polynomial ideal membership problem is extremely difficult from a complexity theory point of view. It can be shown that ideal membership is exponentially space hard as a function of

n, the number of variables (Cardoza et al 1976 [29], Mayr & Meyer 1982) [115]. Therefore, the Gröbner basis construction and generalized remaindering is for general inputs a complex procedure. In light of the popularity of the Gröbner basis algorithm, surprisingly little is known on classifying ideals according to their computational complexity, a measure that has yet to be made precise. Bayer & Stillman (1985) [10] have made an attempt to measure ideal complexity in terms of m-regularity, but compared to the notion of straight-line complexity for individual polynomials, we have almost no insight why certain polynomial ideals are difficult and others are easy to manipulate, both of which phenomena have repeatedly been demonstrated by the Gröbner basis method.

The special and from a geometric point of view more important question whether a polynomial system is solvable - that is, membership of 1 (see Expression 6), can be shown to have much smaller complexity. The recently announced degree bounds by Brownawell (1986) [19]

$$1 = \sum_{i=1}^{r} h_i f_i,\ \deg(h_i) \le 3 \min(n, r)\, n\, d^{\min(n, r)},\ d = \max\{\deg(f_i)\},$$

lead to an algorithm of subexponential time with respect to the dense term count of the f_i. This result already followed from the work announced earlier by Grigoryev & Chistov (1984) [67]. For sparse polynomials, it is relatively easy to show co-NP hardness of the solvability problem (Agnarsson et al 1984) [4], and exponential dependence on n will therefore remain.

Even though solvability of polynomial systems is from the complexity point of view an easier problem than ideal membership, the best computerized attack on any such problem to-date appears the Gröbner basis construction. Indeed, well-tuned implementations are available (Boege et al 1986) [13], and further speed-ups may be expected from an increased understanding of its relationship to classical elimination by resultants (van der Waerden 1953) [176] and linear system solving.

Gröbner bases have recently been used in Wu's (1978) [182] algebraic approach of geometrical theorem proving (Kapur 1986 [89]; Kutzler & Stifter 1986) [99]; but when using this method, statements in geometry cannot easily be refuted, since a counterexample has the added restriction that its parameters be real numbers. However, an even larger theory encompassing inequalities and first order quantification still remains decidable, which is our last problem.

Decision Methods for Elementary Algebra and Geometry

Sentences in the elementary theory of real numbers are constructed from polynomials $f_i(x_1, \ldots, x_n) \in \mathbf{Q}[x_1, \ldots, x_n]$ by the predicate symbols $=0$, $\neq 0$, > 0, the Boolean operators **and**, and **or**, and the first order quantifiers $\forall x_j$, and $\exists x_j$. For example, the sentence

$$\forall x, y: (\ a^2 (x - x_0)^2 + b^2 (y - y_0)^2 - a^2 b^2 \neq 0 \ \textbf{or}\ x^2 + y^2 < 1) \qquad 7.$$

expresses the geometrical statement that the ellipsoid given by the real quantities x_0, y_0, a, and b lies entirely within the unit circle. It is Tarski's accomplishment (published in 1948) to show that

the truth or falsehood of a sentence in the theory of real closed fields can be decided by an algorithm in a finite number of steps. One important principle is that of elimination of quantifiers - e.g., that for the above sentence one can find a quantifier free sentence in x_0, y_0, a, and b, whose real solutions, which form a so-called semi-algebraic set, are exactly the quadruples for which the sentence is true.

Tarski's original algorithm has a horrendous complexity, but much better algorithms have been found since then. The sophisticated 1975 approach by Collins is based on decomposing a semi-algebraic set into cylindrical cells. Projections by resultants and exact real root isolations (discussed in the arithmetic section above) are a major tool in this cell decomposition algorithm. I recommended the article by Arnon et al (1984) [7], which gives an introductory account of this method, and the one by Collins (1982) [43] which contains a full set of references to the classical and modern literature. Unfortunately, the general problem again requires at least exponential time in the number of variables (Fischer & Rabin 1974) [54]. Nonetheless, Arnon (1985) [6] has managed to solve selected nontrivial problems by his implementation of the cylindrical algebraic decomposition method. A quantifier-free formula to Expression 7 can be found in Lauer (1977) [102]. Perhaps entire subtheories, like Wu's theory of geometry, can be computerized successfully with these new algorithms.

LINKAGES

Computer algebra has connections to various areas inside and outside computer science. The major ones, which could themselves be considered as sub-disciplines within computer algebra, are now discussed.

Parallel Algorithms

As parallel computers become available, computer algebra systems will benefit from their increased performance. Some algorithms are obviously executable in parallel - the Chinese remainder imaging scheme, or the probabilistic elliptic curve factoring algorithm, to name but two. In these applications of parallel computing the task gets distributed to several processors, each of which is required to possess relatively large computing power. The arithmetic is speeded-up by a finer grain of parallelism available, for example on a vector computer, and this will require the redesigning of some of the basic algorithms. Fortunately, computer algebra can borrow from a significant amount of research done already on this subject. For example, carry save integer addition and multiplication, the systolic algorithms for integer and polynomial GCD (Brent & Kung 1983 [16]; Yun & Zhang 1986) [185], for linear system solving Leiserson's work (1983) [105], or the FFT algorithm performed on a butterfly network can serve as a starting point. It is the challenge of the immediate future to adapt such concepts to the needs of computer algebra and to incorporate them into our systems. On the theoretical side, the theory of problem solving by circuits of polylogarithmic depth has received much attention, refer to the surveys by Cook (1985) [45] and von zur Gathen (1986) [60]. That work may prove useful to computer algebra in the more distant future.

Logic Programming and Simplification

One main task in computer algebra is to transform mathematical expressions into simpler ones. The notion of simplicity itself is subjective, and different approaches are compared by Moses (1971) [125] and Buchberger & Loos (1982) [23]. One such concept prescribes simplification to canonical forms (as in Caviness 1970) [30], which means that the simplified versions of different expressions with the same meaning must be identical. The generalized remainders of multivariate polynomials computed with respect to Gröbner bases discussed earlier turn out to be such canonical normal forms. Loos observed that the Gröbner basis construction is very similar to the so-called Knuth-Bendix completion procedure for equational theories, and much recent work has been directed to formalizing this relationship (e.g. by Kandri-Rody & Kapur 1983 [87]; Winkler 1984) [181]. A sound foundation for algebraic simplification is therefore the theory of term rewriting, which is discussed in the survey article by Buchberger (1985b) [21]

The logic programming approach to simplification deals with expressions purely syntactically, which is both its strength [the algorithms are well suited for computer implementations (Kapur & Sivakumar 1984)] [90], and its weakness [the approach is oblivious to additional mathematical knowledge about the problem]. As an example we offer the problem of simplification of expressions with radicals (Caviness & Fateman 1976 [32]; Borodin et al 1985 [14]; Zippel 1985) [190], which is strongly connected to Galois theory. Nonetheless, term rewriting is the most powerful general tool for algebraic simplification to date.

Canonical simplifiers can be used to test the equality of two expressions. On the other hand, if we represent expressions by straight-line programs, canonical normal forms are difficult to obtain. However, equality testing can be accommodated by a modern randomized algorithm, which evaluates the programs at random points modulo a large random prime number (Schwartz 1980 [157]; Ibarra & Moran 1983 [72]; Gonnet 1984) [65]. Therefore, with respect to the identity problem canonical forms are generally not as important any longer.

Algebraic simplification is perhaps the trickiest problem in computer algebra, and fully satisfactory solutions can only be expected from future research.

Language and System Design

One of the biggest successes in computer algebra is the fact that the algorithms are not just designed by pencil and paper but are available within systems of substantial dimension. The systems MACSYMA, muMATH, and REDUCE are second-generation general-purpose Computer Algebra systems based on LISP; the system SAC/2 is based on FORTRAN; and the systems MAPLE and SMP are based on C; but all of them are programmable in their own language. For example, the problem in Expression 1 (see the introduction) could be carried out by MACSYMA on a SYMBOLICS 3670 in 38 seconds. In my view, SCRATCHPAD/II (Jenks 1984) [73] begins a new generation of computer algebra systems because of its abstract data types (at last, a program for Gaussian elimination can be written, in which the entries lie in an abstractly denoted field) and because of its domain building capabilities. Other such systems are NEWSPEAK

(Foderaro 1983) [56] and VIEWS (Abdali et al 1986) [2], but compared to SCRATCHPAD/II these other efforts are still in their experimental stages. Abstraction mechanisms will find their way into most computer algebra implementation efforts, if not on the user level then at least for library package programming.

Aside from general purpose system design, selected specialized efforts are noteworthy, such as generation of optimized FORTRAN code from symbolic solutions (Gates 1986 [58]; Wang 1986) [179], or the design of user friendly interfaces custom tailored for formula entry (Smith & Soiffer 1986) [162].

Activities Outside Computer Science

It is not unreasonable to conjecture that most users of computer algebra systems are not computer scientists. I am not in a position to survey the application and assess the impact of computer algebra in the mathematical and natural sciences, in engineering, and in mathematical economy, but instead refer the reader to the article by Calmet & van Hulzen (1982) [25] and to the application letters in the *Journal of Symbolic Computation.* As computer algebra systems become available on personal computers (Stoutemyer 1985a) [165], I expect that teaching the universal arithmetic will be influenced by its automation, perhaps as strongly as calculators have changed education in basic arithmetic. For an experimental study, see Char et al (1986) [34].

CONCLUSION

Any intellectual activity that enlists mathematics as one of its tools ultimately needs to manipulate mathematical expressions. Computer algebra puts the burden of "formula crunching" on computing machines. The design of efficient algorithms and systems that can solve such symbolic computation tasks successfully requires great ingenuity, because the classical approach is likely to fail, as modern computation complexity theory can even prove. Here I have listed seminumerical arithmetic and five major problem areas as the hallmarks of today's computer algebra. In order to see these accomplishments at work, the reader is encouraged to test the locally available computer algebra system with his favorite symbolic problem. Eventually, some shortcomings of the algorithms and systems will be revealed to the user. At times, these difficulties are caused by a lack of manpower to refine and implement the best known algorithms. But there are situations for which we have just begun a computerized attack, be it to use parallel computing power or to classify problems according to their intrinsic computational complexity and identify and solve the easy ones. Clearly, many problems are temptingly open, but the state of the art in computer algebra might already impress the ghost of Isaac Newton.

Acknowledgments

Work for this article has been supported in part by the National Science Foundation under Grant No. DCR-85-04391 and by an IBM Faculty Development Award. All computational examples were carried out on the MACSYMA system. I also thank my wife Hoang for her help in typing the manuscript.

References

1. Abbott, J. A., Bradford, R. J., and Davenport, J. H., "The Bath algebraic number package," *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 250-253 (1986).

2. Abdali, S. K., Cherry, G. W., and Soiffer, N., "An object-oriented approach to algebra system design," *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 24-30 (1986).

3. Adleman, L. M. and Huang, M.-D. A., "Recognizing primes in random polynomial time," *Proc. 19th ACM Symp. Theory Comp.*, pp. 462-469 (1987).

4. Agnarsson, S., Kandri-Rody, A., Kapur, D., Narendran, P., and Saunders, B. D., "Complexity of testing whether a polynomial ideal is nontrivial," *Proc. 1984 MACSYMA Users' Conf.*, pp. 452-458, General Electric (1984).

5. Aho, A., Hopcroft, J., and Ullman, J., *The Design and Analysis of Algorithms*, Addison and Wesley, Reading, MA (1974).

6. Arnon, D. S., "On mechanical quantifier elimination for elementary algebra and geometry: Solution of a non-trivial problem," *Proc. EUROCAL '85, Vol. 2, Springer Lec. Notes Comp. Sci.* **204**, p. 271 (1985).

7. Arnon, D. S., Collins, G. E., and McCallum, S., "Cylindrical algebraic decomposition I: The basic algorithm," *SIAM J. Comp.* **13**, pp. 865-877 (1984).

8. Babai, L. and Szemeredi, E., "On the complexity of matrix group problems," *Proc. 25th IEEE Symp. Foundations Comp. Sci.*, pp. 229-240 (1984). To appear in Combinatorica.

9. Bach, E., *Analytic methods in the analysis and design of number theoretic algorithms*, ACM Distinguished Dissertation Series, MIT Press, Cambridge, MA (1985).

10. Bayer, D. and Stillman, M., "A criterion for detecting m-regularity," Manuscript (May 1985).

11. Berlekamp, E. R., "Factoring polynomials over finite fields," *Bell Systems Tech. J.* **46**, pp. 1853-1859 (1967). Republished in revised form in: E. R. Berlekamp, *Algebraic Coding Theory*, Chapter 6, McGraw-Hill Publ., New York 1968.

12. Berlekamp, E. R., "Factoring polynomials over large finite fields," *Math. Comp.* **24**, pp. 713-735 (1970).

13. Boege, W., Gebauer, R., and Kredel, H., "Some examples for solving systems of algebraic equations by calculating Groebner bases," *J. Symbolic Comp.* **2**, pp. 83-98 (1986).

14. Borodin, A., Fagin, R., Hopcroft, J. E., and Tompa, M., "Decreasing the nesting depth of expressions involving square roots," *J. Symbolic Comp.* **1**, pp. 169-188 (1985).

15. Borodin, A. and Munro, I., *Computational Complexity of Algebraic and Numeric Problems*, American Elsevier, New York, N.Y. (1975).

16. Brent, R. P. and Kung, H. T., "Systolic VLSI arrays for linear-time GCD computation," *Proc. VLSI '83*, pp. 145-154, IFIP (1983).

17. Brown, W. S., "On Euclid's algorithm and the computation of polynomial greatest common divisors," *J. ACM* **18**, pp. 478-504 (1971).

18. Brown, W. S. and Traub, J. F., "On Euclid's algorithm and the theory of subresultants," *J. ACM* **18**, pp. 505-514 (1971).

19. Brownawell, W. D., "Bounds for the degrees in the Nullstellensatz," Manuscript (October 1986).

20. Buchberger, B., “Gröbner bases: An algorithmic method in polynomial ideal theory” in *Recent Trends in Multidimensional Systems Theory*, ed. N. K. Bose, pp. 184-232, D. Reidel Publ. Comp., Dordrecht (Holland) (1985).

21. Buchberger, B., “Basic features and development of the critical-pair/completion procedure,” *Proc. Conf. Rewriting Techniques Appl., Springer Lec. Notes Comp. Sci.* **202**, pp. 1-45 (1985).

22. Buchberger, B., Collins, G. E., and (eds.), R. Loos, *Computer Algebra: Symbolic and Algebraic Computation*, Springer Verlag, Vienna (1982).

23. Buchberger, B. and Loos, R., “Algebraic simplification” in *Computer Algebra, 2nd ed.*, ed. B. Buchberger et al., pp. 11-43, Springer Verlag, Vienna (1982).

24. Butler, M. C. R., “On the reducibility of polynomials over a finite field,” *Quart. J. Math., Oxford Ser. (2)* **5**, pp. 102-107 (1954).

25. Calmet, J. and Hulzen, J. A. van, “Computer algebra applications” in *Computer Algebra, 2nd ed.*, ed. B. Buchberger et al., pp. 245-258, Springer Verlag, Vienna (1982).

26. Cannon, J. J., “A draft description of the group theory language Cayley,” *Proc. 1976 ACM Symp. Symbolic Algebraic Comp.*, pp. 66-84 (1976).

27. Cantor, D. G. and Kaltofen, E., “Fast multiplication of polynomials with coefficients from an arbitrary ring,” Manuscript (March 1987).

28. Cantor, D. G. and Zassenhaus, H., “A new algorithm for factoring polynomials over finite fields,” *Math. Comp.* **36**, pp. 587-592 (1981).

29. Cardoza, E., Lipton, R., and Meyer, A. R., “Exponential space complete problems for Petri nets and commutative semigroups,” *Proc. 8th ACM Symp. Theory Comp.*, pp. 50-54 (1976).

30. Caviness, B. F., “Canonical forms and simplification,” *J. ACM* **17**, pp. 385-396 (1970).

31. Caviness, B. F., “Computer Algebra: Past and Future,” *J. Symbolic Comp.* **2**, pp. 217-236 (1986).

32. Caviness, B. F. and Fateman, R., “Simplification of radical expressions,” *Proc. 1976 ACM Symp. Symbolic Algebraic Comp.*, pp. 329-338 (1976).

33. Char, B. W., Geddes, K. O., and Gonnet, G. H., “GCDHEU: Heuristic polynomial GCD algorithm based on integer GCD computation,” *Proc. EUROSAM ’84, Springer Lec. Notes Comp. Sci.* **174**, pp. 285-296 (1984). To appear in J. Symbolic Comp.

34. Char, B. W., Geddes, K. O., Gonnet, G. H., Marshman, B. J., and Ponzo, P. J., “Computer algebra in the undergraduate mathematics classroom,” *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 135-138 (1986).

35. Cherry, G. W., “Integration in finite terms with special functions: the error function,” *J. Symbolic Comp.* **1**, pp. 283-302 (1985).

36. Cherry, G. W., “Integration in finite terms with special functions: the logarithmic integral,” *SIAM J. Comp.* **15**, pp. 1-21 (1986).

37. Chistov, A. L., “Polynomial complexity of the Newton-Puiseux algorithm,” *Proc. MFCS ’86, Springer Lec. Notes Comp. Sci.* **233**, pp. 247-255 (1986).

38. Chistov, A. L. and Grigoryev, D. Yu., “Polynomial-time factoring of multivariable polynomials over a global field,” *LOMI preprint E-5-82*, Steklov Institute, Leningrad (1982).

39. Chou, T. J. and Collins, G. E., “Algorithms for the solution of systems of diophantine linear equations,” *SIAM J. Comp.* **11**, pp. 687-708 (1982).

40. Chudnovsky, D. V. and Chudnovsky, G. V., “Computer assisted number theory with applications,” Tech. Report, IBM T. J. Watson Research Center, Yorktown Heights, N.Y. (1986).

41. Collins, G. E., “Subresultants and reduced polynomial remainder sequences,” *J. ACM* **14**, pp. 128-142 (1967).

42. Collins, G. E., “The calculation of multivariate polynomial resultants,” *J. ACM* **18**, pp. 515-532 (1971).

43. Collins, G. E., “Quantifier elimination for real closed fields: A guide to the literature” in *Computer Algebra, 2nd ed.*, ed. B. Buchberger et al, pp. 173-187, Springer Verlag, Vienna (1982).

44. Collins, G. E. and Loos, R., “Real zeros of polynomials” in *Computer Algebra, 2nd ed.*, ed. B. Buchberger et al, pp. 83-94, Springer Verlag, Vienna (1982).

45. Cook, S. A., “A taxonomy of problems with fast parallel algorithms,” *Inf. Control* **64**, pp. 2-22 (1985).

46. Cooley, J. W. and Tuckey, J. W., “An algorithm for the machine calculation of complex Fourier series,” *Math. Comp.* **19**, pp. 297-301 (1965).

47. Coppersmith, D. and Winograd, S., “Matrix multiplication via arithmetic progressions,” *Proc. 19th Annual ACM Symp. Theory Comp.*, pp. 1-6 (1987).

48. Czapor, S. R. and Geddes, K. O., “A comparison of algorithms for the symbolic computation of Padé approximants,” *Proc. EUROSAM ’84, Springer Lec. Notes Comp. Sci.* **174**, pp. 248-259 (1984).

49. Davenport, J. H., *On the integration of algebraic functions*, Springer Lec. Notes Comp. Sci. **102** (1981).

50. Davenport, J. H. and Trager, B. M., “On the parallel Risch algorithm (II),” *ACM Trans. Math. Software* **11**, pp. 356-362 (1985).

51. (ed.), B. F. Caviness, *Proc. EUROCAL ’85, Vol. 2*, Springer Lec. Notes Comp. Sci. **204** (1985).

52. (ed.), B. W. Char, “SYMSAC ‘86,” *Proc. 1986 Symp. Symbolic Algebraic Comp.*, Assoc. Comp. Machinery, New York, N.Y. (1986).

53. Epstein, H. I. and Caviness, B. F., “A structure theorem for the elementary functions and its application to the identity problem,” *Internat. J. Comp. Inf. Sci.* **8**, pp. 9-37 (1979).

54. Fischer, M. J. and Rabin, M. O., “Super-exponential complexity of Presburger arithmetic” in *Complexity of Computation*, ed. R. M. Karp, pp. 27-41, Amer. Math. Soc. (1974).

55. Fitch, J., “User-based integration software,” *Proc. 1981 ACM Symp. Symbolic Algebraic Comp.*, pp. 245-248 (1981).

56. Foderaro, J., “Newspeak,” Ph.D. Thesis, Univ. California-Berkeley (1983).

57. Freeman, T. S., Imirzian, G., Kaltofen, E., and Yagati, Lakshman, “DAGWOOD: A system for manipulating polynomials given by straight-line programs,” Tech. Report 86-15, Dept. Comput. Sci., RPI. Preliminary version in *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 169-175, 1986.

58. Gates, B. L., “A numerical code generation facility for Reduce,” *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 94-99 (1986).

59. Gathen, J. von zur, “Irreducibility of multivariate polynomials,” *J. Comp. System Sci.* **31**, pp. 225-264 (1985).

60. Gathen, J. von zur, “Parallel arithmetic computation: A survey,” *Proc. MFCS ’86, Springer Lec. Notes Comp. Sci.* **233**, pp. 93-112 (1986).

61. Gathen, J. von zur and Kaltofen, E., "Factoring sparse multivariate polynomials," *J. Comp. System Sci.* **31**, pp. 265-287 (1985).

62. Gentleman, W. M. and Johnson, S. C., "Analysis of algorithms, a case study: Determinants of matrices with polynomial entries," *ACM Trans. Math Software* **2**, pp. 232-241 (1976).

63. Goldwasser, S. and Kilian, J., "A provable correct and probably fast primality test," *Proc. 18th ACM Symp. Theory Comp.*, pp. 316-329 (1986).

64. Goldwasser, S. and Micali, S., "Probabilistic encryption," *J. Comp. System Sci.* **28**, pp. 270-299 (1984).

65. Gonnet, G. H., "Determining equivalence of expressions in random polynomial time," *Proc. 16th ACM Symp. Theory Comp.*, pp. 334-341 (1984).

66. Gosper, R. W., "Decision procedure for indefinite hypergeometric summation," *Proc. Natl. Acad. Sci. USA* **75**, pp. 40-42 (1978).

67. Grigoryev, D. Yu. and Chistov, A. L., "Fast decomposition of polynomials into irreducible ones and the solution of systems of algebraic equations," *Soviet Math. Dokl. (AMS Translation)* **29**, pp. 380-383 (1984).

68. Hearn, A. C., "Non-modular computation of polynomial GCDs using trial division," *Proc. EUROSAM '79, Springer Lec. Notes Comp. Sci.* **72**, pp. 227-239 (1979).

69. Heintz, J. and Sieveking, M., "Absolute primality of polynomials is decidable in random polynomial-time in the number of variables," *Proc. ICALP '81, Springer Lec. Notes Comp. Sci.* **115**, pp. 16-28 (1981).

70. Hermann, G., "Die Frage der endlich vielen Schritte in der Theorie der Polynomideale," *Math. Ann.* **95**, pp. 736-788 (1926). (In German).

71. Horowitz, E., "A sorting algorithm for polynomial multiplication," *J. ACM* **22**, pp. 450-462 (1975).

72. Ibarra, O. H. and Moran, S., "Probabilistic algorithms for deciding equivalence of straight-line programs," *J. ACM* **30**, pp. 217-228 (1983).

73. Jenks, R. D., "A primer: 11 keys to new SCRATCHPAD," *Proc. EUROSAM '84, Springer Lec. Notes Comp. Sci.* **174**, pp. 123-147 (1984).

74. Kaltofen, E., "Polynomial factorization" in *Computer Algebra, 2nd ed.*, ed. B. Buchberger et al, pp. 95-113, Springer Verlag, Vienna (1982).

75. Kaltofen, E., "A polynomial reduction from multivariate to bivariate integral polynomial factorization," *Proc. 14th Annual ACM Symp. Theory Comp.*, pp. 261-266 (1982).

76. Kaltofen, E., "A polynomial-time reduction from bivariate to univariate integral polynomial factorization," *Proc. 23rd IEEE Symp. Foundations Comp. Sci.*, pp. 57-64 (1982).

77. Kaltofen, E., "Sparse Hensel lifting," *Proc. EUROCAL '85, Vol. 2, Springer Lec. Notes Comp. Sci.* **204**, pp. 4-17 (1985).

78. Kaltofen, E., "Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization," *SIAM J. Comp.* **14**, pp. 469-489 (1985).

79. Kaltofen, E., "Effective Hilbert irreducibility," *Information and Control* **66**, pp. 123-137 (1985).

80. Kaltofen, E., "Factorization of polynomials given by straight-line programs," *Math. Sci. Research Inst. Preprint* **02018-86**, Berkeley, CA (1986). To appear in: ''Randomness in Computation,'' Advances in Computing Research, S. Micali ed., JAI Press Inc., Greenwich, CT, January 1987.

81. Kaltofen, E., “Uniform closure properties of p-computable functions,” *Proc. 18th ACM Symp. Theory Comp.*, pp. 330-337 (1986).

82. Kaltofen, E., “Polynomial Factorization 1982-1986,” Tech. Report, Dept. Comp. Sci., Rensselaer Polytech. Inst. (September 1986).

83. Kaltofen, E., “Single-factor Hensel lifting and its application to the straight-line complexity of certain polynomials,” *Proc. 19th Annual ACM Symp. Theory Comp.*, pp. 443-452 (1987).

84. Kaltofen, E., “Greatest common divisors of polynomials given by straight-line programs,” *J. ACM* **35**(1), pp. 231-264 (1988).

85. Kaltofen, E., Krishnamoorthy, M. S., and Saunders, B. D., “Fast parallel algorithms for similarity of matrices,” *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 65-70 (1986).

86. Kaltofen, E., Krishnamoorthy, M. S., and Saunders, B. D., “Fast parallel computation of Hermite and Smith forms of polynomial matrices,” *SIAM J. Alg. Discrete Meth.* **8**, pp. 683-690 (1987).

87. Kandri-Rody, A. and Kapur, D., “On the relationship between Buchberger’s Gröbner basis algorithm and the Knuth-Bendix completion procedure,” Tech. Rep. 83 CRD 286, General Electric Research Development Center, Schenectady, N.Y. (1983).

88. Kannan, R. and Bachem, A., “Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix,” *SIAM J. Comp.* **8**, pp. 499-507 (1981).

89. Kapur, D., “Geometry theorem proving using Hilbert’s Nullstellensatz,” *J. Symbolic Comp.* **2**, pp. 399-408 (1986).

90. Kapur, D. and Sivakumar, G., “Architecture of and experiments with RRL, a rewrite rule laboratory” in *Proc. NSF Workshop Rewrite Rule Laboratory*, Tech. Rep. 84 GEN 008, pp. 33-56, General Electric Research Development Center, Schenectady, N.Y. (1984).

91. Karr, M., “Theory of summation in finite terms,” *J. Symbolic Comp.* **1**, pp. 303-315 (1985).

92. Knowles, P. H., “Integration of Liouvillian functions with special functions,” *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 179-184 (1986).

93. Knuth, D. E., “The analysis of algorithms,” *Actes du congr`es international des Mathématiciens* **3**, pp. 269-274, Nice, France (1970).

94. Knuth, D. E., *The Art of Programming, vol. 2, Semi-Numerical Algorithms, ed. 2*, Addison Wesley, Reading, MA (1981).

95. Kovacic, J. J., “An algorithm for solving second order linear homogeneous differential equations,” *J. Symbolic Comp.* **2**, pp. 3-43 (1986).

96. Krishnamurthy, E. V., Rao, T. M., and Subramanian, K., “Finite segment p-adic number systems with applications to exact computation,” *Proc. Indian Acad. Sci.* **81, sec. A, no. 2**, pp. 58-79 (1975).

97. Kronecker, L., “Grundzüge einer arithmetischen Theorie der algebraischen Grössen,” *J. reine angew. Math.* **92**, pp. 1-122 (1882).

98. Kung, H. T. and Traub, J. F., “All algebraic functions can be computed fast,” *J. ACM* **25**, pp. 245-260 (1978).

99. Kutzler, B. and Stifter, S., “On the application of Buchberger’s algorithm to automated geometry theorem proving,” *J. Symbolic Comp.* **2**, pp. 389-397 (1986).

100. Landau, S., “Factoring polynomials over algebraic number fields,” *SIAM J. Comp.* **14**, pp. 184-195 (1985).

101. Landau, S. and Miller, G. L., “Sovability by radicals,” *J. Comp. System Sci.* **30**, pp. 179-208 (1985).

102. Lauer, M., “A solution to Kahan’s problem (SIGSAM problem no. 9),” *ACM SIGSAM Bulletin* **11**, pp. 16-20 (1977).

103. Lauer, M., “Computing by homomorphic images” in *Computer Algebra, 2nd ed.*, ed. B. Buchberger et al, pp. 139-168, Springer Verlag, Vienna (1982).

104. Lazard, D., “Alg`ebre linéaire sur $K[X_{1,ts,X\_n]}$ et élimination,” *Bull. Soc. Math. France* **105**, pp. 165-190 (1977). (In French).

105. Leiserson, C. E., *Area-efficient VLSI computation*, The ACM Doctoral Dissertation Award, M.I.T. Press, Cambridge, MA (1983).

106. Lenstra, A. K., “Lattices and factorization of polynomials over algebraic number fields,” *Proc. EUROCAM ’82, Springer Lec. Notes Comp. Sci.* **144**, pp. 32-39 (1982).

107. Lenstra, A. K., “Factoring multivariate integral polynomials,” *Theoretical Comp. Sci.* **34**, pp. 207-213 (1984).

108. Lenstra, A. K., Jr., H. W. Lenstra, and Lovász, L., “Factoring polynomials with rational coefficients,” *Math. Ann.* **261**, pp. 515-534 (1982).

109. Lenstra, H. W., Jr., “Factoring integers with elliptic curves,” Manuscript (May 1986).

110. Leon, J. S. and Pless, V., “CAMAC 79,” *Proc. EUROSAM ’79, Springer Lec. Notes Comp. Sci.* **72**, pp. 249-257 (1979).

111. Lipson, J., *Elements of Algebra and Algebraic Computing*, Addison-Wesley Publ., Reading, Mass. (1981).

112. Lipton, R., Rose, D., and Tarjan, R. E., “Generalized nested dissection,” *SIAM J. Numer. Anal.* **16**, pp. 346-358 (1979).

113. Loos, R., “Computing in algebraic extensions” in *Computer Algebra, 2nd ed.*, ed. B. Buchberger et al, pp. 173-187, Springer Verlag, Vienna (1982).

114. Luks, E. M., “Isomorphism of graphs of bounded valence can be tested in polynomial time,” *J. Comp. System Sci.* **25**, pp. 42-65 (1982).

115. Mayr, E. W. and Meyer, A. R., “The complexity of the word problem for commutative semigroups and polynomial ideals,” *Advances Math.* **46**, pp. 305-329 (1982).

116. McClellan, M. T., “The exact solution of systems of linear equations with polynomial coefficients,” *J. ACM* **20**, pp. 563-588 (1973).

117. Miller, G. L., “Riemann’s hypthesis and tests for primality,” *J. Comp. System Sci.* **13**, pp. 300-317 (1976).

118. Miller, G. L., Ramachandran, V., and Kaltofen, E., “Efficient parallel evaluation of straight-line code and arithmetic circuits,” *Proc. AWOC ’86, Springer Lec. Notes Comp. Sci.* **227**, pp. 236-245 (1986).

119. Moenck, R. T., “Fast computation of GCDs,” *Proc. 5th ACM Symp. Theory Comp.*, pp. 142-151 (1973).

120. Moenck, R. T. and Carter, J. H., “Approximate algorithms to derive exact solutions to systems of linear equations,” *Proc. EUROSAM ’79, Springer Lec. Notes Comp. Sci.* **72**, pp. 65-73 (1979).

121. Monagan, M. B., “A heuristic irreducibility test for univariate polynomials,” *J. Symbolic Comp.* **submitted** (1986).

122. Montgomery, P. L., “Speeding the Pollard and elliptic curve methods of factorization,” *Math. Comp.* **48**, pp. 243-264 (1987).

123. Moore, P. M. A. and Norman, A. C., “Implementing a polynomial factorization problem,” *Proc. 1981 ACM Symp. Symbolic Algebraic Comp.*, pp. 109-116 (1981).

124. Moses, J., “Symbolic integration,” Ph.D. Thesis and Project MAC Tech. Rep. 47, MIT (1967).

125. Moses, J., “Algebraic simplification: A guide for the perplexed,” *Commun. ACM* **14**, pp. 548-560 (1971).

126. Moses, J., “Symbolic integration: The stormy decade,” *Commun. ACM* **14**, pp. 548-560 (1971).

127. Moses, J. and Yun, D. Y. Y., “The EZ-GCD algorithm,” *Proc. 1973 ACM National Conf.*, pp. 159-166 (1973).

128. Musser, D. R., “Multivariate polynomial factorization,” *J. ACM* **22**, pp. 291-308 (1975).

129. Neubüser, J., “Computing with groups and their character tables” in *Computer Algebra, 2nd ed.*, ed. B. Buchberger et al, pp. 45-56, Springer Verlag, Vienna (1982).

130. Newton, I., *Arithmetica Universalis, 2nd ed.*, London (1728). In English. Reprinted in *The Mathematical Works of Isaac Newton*, vol. 2, D. T. Whiteside, ed., Johnson Reprint Corp., New York, 1967.

131. Norman, A. C. and Moore, P. M. A., “Implementing the new Risch algorithm,” *Proc. Conf. Adv. Comp. Methods Theor. Physics at St. Maximin*, pp. 99-110 (1977).

132. Nussbaumer, H. J., “Fast polynomial transform algorithms for digital convolutions,” *IEEE Trans. ASSP* **28**, pp. 205-215 (1980).

133. Ostrowski, M. A., “Sur l’intégrabilité élémentaire de quelches classes d’expressions,” *Comment. Math. Helv.* **28**, pp. 283-308 (1946). (In French).

134. Pan, V. and Reif, J., “Efficient parallel solution of linear systems,” *Proc. 17th ACM Symp. Theory Comp.*, pp. 143-152 (1985).

135. Pavelle, R., Rothstein, M., and Fitch, J., “Computer algebra,” *Scientific American* **245**, pp. 135-152 (1981).

136. Plaisted, D. A., “Sparse complex polynomials and polynomial reducibility,” *J. Comp. System Sci.* **14**, pp. 210-221 (1977).

137. Pomerance, C., “Analysis and comparison of some integer factoring algorithms” in *Computational Methods in Number Theory, Part I*, ed. H. W. Lenstra, Jr. and R. Tijdeman, Mathematical Centre Tracts **154**, pp. 89-139, Mathematisch Centrum, Amsterdam (1982).

138. Rabin, M. O., “Probabilistic algorithms for testing primality,” *J. Number Theory* **12**, pp. 128-138 (1980).

139. Rabin, M. O., “Probabilistic algorithms in finite fields,” *SIAM J. Comp.* **9**, pp. 273-280 (1980).

140. Risch, R. H., “The problem of integration in finite terms,” *Trans. Amer. Math. Soc.* **139**, pp. 167-189 (1969).

141. Risch, R. H., “The solution of the problem of integration in finite terms,” *Bull. Amer. Math. Soc.* **76**, pp. 605-608 (1970).

142. Risch, R. H., “Algebraic properties of the elementary functions of analysis,” *Amer. J. Math.* **101**, pp. 743-759 (1979).

143. Rosenlicht, M., “Liouville’s theorem on functions with elementary integrals,” *Pacific J. Math.* **24**, pp. 153-161 (1968).

144. Rothstein, M., “Aspects of symbolic integration and simplification of exponential and primitive functions,” Ph.D. Thesis, Univ. Wisconsin-Madison (1976).

145. Rothstein, M., “On pseudo-resultants,” *Proc. EUROSAM ’84, Springer Lec. Notes Comp. Sci.* **174**, pp. 386-396 (1984).

146. Rothstein, M. and Caviness, B. F., “A structure theorem for exponential and primitive functions,” *SIAM J. Comp.* **8**, pp. 357-367 (1979).

147. Saunders, B. D., “An implementation of Kovacic’s algorithm for solving second order linear differential equations,” *Proc. 1981 ACM Symp. Symbolic Algebraic Comp.*, pp. 105-108 (1981).

148. Schönhage, A., “Schnelle Kettenbruchentwicklungen,” *Acta Inf.* **1**, pp. 139-144 (1971). (In German).

149. Schönhage, A., “Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2,” *Acta Inf.* **7**, pp. 395-398 (1977). (In German).

150. Schönhage, A., “Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients,” *Proc. EUROCAM ’82, Springer Lec. Notes Comp. Sci.* **144**, pp. 3-15 (1982).

151. Schönhage, A., “The fundamental theorem of algebra in terms of computational complexity,” Tech. Report, Univ. Tübingen (1982).

152. Schönhage, A., “Factorization of univariate integer polynomials by diophantine approximation and an improved basis reduction algorithm,” *Proc. ICALP ’84, Springer Lec. Notes Comp. Sci.* **172**, pp. 436-447 (1984).

153. Schönhage, A., “Quasi-GCD computations,” *J. Complexity* **1**, pp. 118-137 (1985).

154. Schönhage, A., “Equation solving in terms of computational complexity,” *Proc. Internat. Congr. Math.* (1986).

155. Schönhage, A., “Probabilistic computation of integer GCDs,” *J. Algorithms* (to appear).

156. Schönhage, A. and Strassen, V., “Schnelle Multiplikation grosser Zahlen,” *Computing* **7**, pp. 281-292 (1971). (In German).

157. Schwartz, J. T., “Fast probabilistic algorithms for verification of polynomial identities,” *J. ACM* **27**, pp. 701-717 (1980).

158. Singer, M. F., “Liouvillian solutions of nth order homogeneous linear differential equations,” *Amer. J. Math.* **103**, pp. 661-682 (1981).

159. Singer, M. F., Saunders, B. D., and Caviness, B. F., “An extension of Liouville’s theorem on integration in finite terms,” *SIAM J. Comp.* **14**, pp. 966-990 (1985).

160. Slagle, J. R., “A heuristic program that solves symbolic integration problems in freshman calulus, symbolic automatic integrator (SAINT),” Ph.D. Thesis, MIT (1961).

161. Smith, C. J., “A discussion and implementation of Kovacic’s algorithm for ordinary differential equations,” Research Rep., Dept. Comp. Sci., Univ. Waterloo (Oct. 1984).

162. Smith, C. J. and Soiffer, N. M., “MathScribe: A user interface for computer algebra systems,” *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 7-12 (1986).

163. Solovay, R. M. and Strassen, V., “A fast Monte-Carlo test for primality,” *SIAM J. Comp.* **6**, pp. 84-85 (1977). Correction: vol. 7, p. 118 (1978).

164. Stoutemyer, D. R., “Polynomial remainder sequences: Greatest common divisors revisited” in *Proc. Second RIKEN Internat. Symp. Symbolic Algebraic Comp. by Computers*, ed. N. Inada and T. Soma, Ser. Comp. Sci. **2**, pp. 1-12, World Scientific Publ., Philadelphia, PA (1985).

165. Stoutemyer, D. R., "A preview of the next IBM-PC version of muMATH," *Proc. EUROCAL '85, Vol. 1, Springer Lec. Notes Comp. Sci.* **203**, pp. 33-44 (1985).

166. Strassen, V., "Vermeidung von Divisionen," *J. reine u. angew. Math.* **264**, pp. 182-202 (1973). (In German).

167. Strassen, V., "Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten," *Numer. Math.* **20**, pp. 238-251 (1973). (In German).

168. Strassen, V., "Algebraische Berechnungskomplexität" in *Anniversary of Oberwolfach 1984,* Perspectives in Mathematics, pp. 509-550, Birkhäuser Verlag, Basel (1984). (In German).

169. Strassen, V., "The asymptotic spectrum of tensors and the exponent of matrix multiplication," *Proc. 27th Annual Symp. Foundations Comp. Sci.*, pp. 49-54 (1986).

170. Trager, B. M., "Algebraic factoring and rational function integration," *Proc. 1976 ACM Symp. Symbolic Algebraic Comp.*, pp. 219-228 (1976).

171. Trager, B. M., "Integration of algebraic functions," Ph.D. Thesis, MIT (1984).

172. Trinks, W., "Uber B. Buchbergers Verfahren, Systeme algebraischer Gleichungen zu lösen," *J. Number Theory* **10**, pp. 475-488 (1978).

173. Valiant, L., "The complexity of computing the permanent," *Theoretical Comp. Sci.* **8**, pp. 189-201 (1979).

174. Valiant, L., "Reducibility by algebraic projections," *L'Enseignement mathématique* **28**, pp. 253-268 (1982).

175. Valiant, L., Skyum, S., Berkowitz, S., and Rackoff, C., "Fast parallel computation of polynomials using few processors," *SIAM J. Comp.* **12**, pp. 641-644 (1983).

176. Waerden, B. L. van der, *Modern Algebra,* F. Ungar Publ. Co., New York (1953).

177. Wang, P. S., "An improved multivariate polynomial factorization algorithm," *Math. Comp.* **32**, pp. 1215-1231 (1978).

178. Wang, P. S., "A p-adic algorithm for univariate partial fractions," *Proc. 1981 ACM Symp. Symbolic Algebraic Comp.*, pp. 212-217 (1981).

179. Wang, P. S., "FINGER: A symbolic system for automatic generation of numerical programs in finite element analysis," *J. Symbolic Comp.* **2**, pp. 305-316 (1986).

180. Wiedemann, D., "Solving sparse linear equations over finite fields," *IEEE Trans. Inf. Theory* **IT-32**, pp. 54-62 (1986).

181. Winkler, F., "The Church-Rosser property in computer algebra and special theorem proving," Doctoral Thesis, Univ. Linz (Austria) (1984).

182. Wu, W.-T., "On the decision problem and the mechanization of theorem proving in elementary geometry" in *Theorem Proving: After 25 Years*, ed. Bledsoe, W. W. Loveland, D. W., Contemporary Mathematics **29**, pp. 235-241, AMS, Providence, RI (1984). Originally published in *Scientia Sinica* vol. 21, 150-172 (1978).

183. Yun, D. Y. Y., "The Hensel lemma in algebraic manipulation," Ph.D. Thesis, M.I.T. (1974). Reprint: Garland Publ., New York 1980.

184. Yun, D. Y. Y. and Stoutemyer, D. R., *Symbolic mathematical computation,* Encyclopedia of Computer Science and Technology **15 (Supplement)**, Marcel Dekker, Inc., New York, N.Y. (1981).

185. Yun, D. Y. Y. and Zhang, C. H., "A fast carry-free algorithm and hardware design for extended integer GCD computation," *Proc. 1986 ACM Symp. Symbolic Algebraic Comp.*, pp. 82-84 (1986).

186. Zassenhaus, H., “On Hensel factorization I,” *J. Number Theory* **1**, pp. 291-311 (1969).

187. Zippel, R. E., “Univariate power series expansions in algebraic manipulation,” *Proc. 1976 ACM Symp. Symbolic Algebraic Comp.*, pp. 198-208 (1976).

188. Zippel, R. E., “Probabilistic algorithms for sparse polynomials,” *Proc. EUROSAM ’79, Springer Lec. Notes Comp. Sci.* **72**, pp. 216-226 (1979).

189. Zippel, R. E., “Newton’s iteration and the sparse Hensel algorithm,” *Proc. ’81 ACM Symp. Symbolic Algebraic Comp.*, pp. 68-72 (1981).

190. Zippel, R. E., “Simplification of expressions by radicals,” *J. Symbolic Comp.* **1**, pp. 189-210 (1985).