Application of Symbolic Mathematics to Mathematics

A. M. Odlyzko

Bell Laboratories
Murray Hill, New Jersey 07974

1. INTRODUCTION

This paper is an account of some applications of symbolic algebra systems to mathematics. It is not a general survey of this very wide subject, but rather a small selection from my own work, a selection made so as to illustrate the main point of the paper, which is that the most important applications of symbolic algebra systems is not to compute formulas as such, but to help in formulating and testing hypothesis.

My experience with symbolic mathematics goes back over ten years. Most of this work was with MACSYMA [10], although at various points I or my collaborators used other systems such as ALTRAN [2], MAPLE [5], and SMP [4]. These systems were used in many fields of mathematics, as the examples presented here demonstrate. Still, these extremely varied examples do not cover the full range of applications that have been made, and are a reflection of my research interests. For example, one of the most impressive achievements in the field of computer algebra is the development of algorithms for indefinite integration. However, I have not yet made any serious use of them. The reason is that all of the many integrals I have worked with werre either very easy, so that I was able to do them either directy or with the help of a table of integrals, or else they were so hard that even the algorithms in MACSYMA failed on them. On the other hand, I have made use of most of the other capabilities of computer algebra systems at one time or another. Since my problems are usually quite mathematical to start with, I had a strong advantage over people in some other fields in not having to devote too much effort into translating these problems into a form suitable for symbolic manipulation programs.

The theme of this paper is reflected in the famous aphorism of R. W. Hamming [8]:

‘‘The purpose of computing is insight, not numbers.’’

In the context of computer algebra, this maxim could be modified to say that the purpose of symbolic maniuplation is insight, not formulas. To make it clear what is meant by this remark, consider some of the most successful applications of symbolic algebra, namely those to physics and celestial mechanics, which are listed and summarized briefly in [3]. In those applications, one is usually faced witha tremendously

complicated mathematical descriptions of the system under consideration, and the goal of the symbolic algebra package is to go through a very long series of complicated transformations which result in a relatively simple formula which can then be used for numerical computation of planetary orbits, for example. The validity of the results depends on the correctness of the program that was used, and little or no insight is provided by those results into why they cam out the way they did. Important applications of this type do occur in mathematics, and an example will be presented in Section 2. It is my feeling, though, that such applications are not the most important ones. I see the main role of symbolic algebra systems as that of helping to formulate hypotheses, search for examples and counterexamples, and in general explore ramifications of mathematical models. In other words, the main role of these systems is to obtain mathematical insight. Once that insight is obtained, one can then go on and construct canonical mathematical proofs, in which there might not even be any traces of the use of computer algebra. Several examples of such applications are presented in Sections 3-5.

2. CLASSIFICATION OF FINITE SIMPLE GROUPS

This section presents an application of computer algebra to our small chapter in the big subject of classification of finite simple groups, namely Boulieri's proof [1] of the uniqueness of simple groups of Ree type. This application was chosen as an example where only some computing power was needed, and no special insight was gained from the computation.

Thompson had earlier shown that the uniqueness of simple groups of Ree type would follow if one could show that given any automorphism σ of $GF(3^{2n+1})$, $n \geq 1$, $\sigma \neq 3^{n+1}$, and integers a, s such that a is even, s is odd, and $x^{(\sigma+1)} = x^2$ and $x^{(\sigma+2)s} = x$ for all $x \in GF(3^{2n+1})$, then is an element $x \in GF(3^{2n+1}) \backslash GF(3)$ such that if

$$z = (x + 1)^s , \quad y = (x - 1)^2 , \quad n = (x + 1)^{1-s} - (x - 1)^{1-s} ,$$

then $u \notin GF(3)$ and

$$z(u - 1)^a - (z - y + 1)u^a - y(u + 1)^2 + (u^2 - 1)^a \neq 0 . \qquad (2.1)$$

Bombien succeeded in showing, by a sophisticated argument, that a suitable x for which (2.1) holds can be found except possibly in a finite number of cases. Then the problem was reduced to a finite computation.

The computations that completed Bombien's proof were carried out by D. Hart and myself (independently) [1]. Given any $x \in GF(3^{2n+1})\backslash GF(3)$, the chances of (2.1) being false seemed very low, so unless there were some unexpected phenomena (such as a counterexample to the conjecture), it seemed likely that choosing a random x and computing the quantity o nthe left side of (2.1) would suffice. That, in fact, turned out to be the case.

The computation of the quantity in (2.1) in a straightforward computer algebra step. It is necessary to generate the field $GF(3^{2n+1})$, which is equivalent to finding an irreducible polynomial of degree $2n + 1$ over $GF(3)$, which can be done easily using known factorization algorithms and then to carry out the computations in the field $GF(3^{2n+1})$. These computations are, in principle, quite straightforward. Their validity (and that of the main result) depend on the correctness of the programs and machines. The fact that (2.1) did hold for the choices of x that were made only confirmed the guess that (2.1) would hold for most x's, and did not provide any insight that would enable one to prove the result without the computations.

As an aside, the computations required to complete Bombien's proof were in principle perfectly suited to the capabilities of the MACSYMA system. I wrote the necessary program in MACSYMA in a fraction of an hour and ran some small cases succesfully through it (and in the process discovered a previously unknown bug in MACSYMA). However, a quick extrapolation of the times needed for those examples showed that the entire computation would have taken on the order of several days of MACSYMA's time. Since I was then a non-paying guest at the MIT MACSYMA machine, I did not feel like straining my host's hospitality to such an extent. Therefore I spent a day or two writing a FORTRAN program especially for their job, which then carried out the entire computation in a few hours. Some of the output of that program was checked using MACSYMA, though.

3. DISCRIMINANTS OF NUMBER FIELDS

In the precision example, the symobolic algebra application stood out on its own, since the final result depended on the correctness of the computation, and no special insight was provided into the reasons the result turned out the way it did. We next consider another application, where the role of computer algebra was quite different. This example is drawn from my dissention, which appeared in [12]. The subject of it

was improved lower bounds for discriminants of number fields. The mehtods used were from analytic number theory. Some of the crucial technical lemmas required showing certain national functions were nonnegative over a wide range. For example, one of these problems was reduced to showing that

$$\begin{aligned} P(v) &= 73v^8 - 20{,}214\,v^7 + 2{,}094{,}217v^6 - 92{,}766{,}496\,v^5 \\ &\quad + 1{,}419{,}515{,}855\,v^4 + 3{,}533{,}810{,}602\,v^3 \\ &\quad - 2{,}837{,}192{,}781\,v^2 - 29{,}267{,}901{,}572\,v + 237{,}342{,}960{,}316 \end{aligned}$$

satisfies $P(v) \geq 0$ for all $v \geq -10$ [12, p. 288]. Now this is a problem that can be solved easily by some computer algebra systems, since there is an algorithm due to Sturm for determining the exact number of zeros of a polynomial in a given interval, and that algorithm is available in MACSYMA. However, the proof presented in [12] does not rely on the use of blot algorithm.

It is easy to see, for example, that for $v \geq 0$,

$$\begin{aligned} P_2(v) &= 10v^8 - 277v^7 + 2{,}868v^6 - 12{,}708v^5 + 19{,}445v^4 \\ &\quad + 4{,}840v^3 - 389v^2 - 401v + 325\ . \end{aligned}$$

Now for $0 \leq v \leq 1/2$, say,

$$389v^2 + 401v \leq 325, \quad 12{,}708v^5 \leq 19{,}445v^4, \quad 277v^7 \leq 2{,}868v^6,$$

and so $P_2(v) \geq 0$ in that range. Other ranges are treated similarly.

During the preparation of [12], the MACSYMA algorithm for finding the number of zeros of a polynomial in an internal was used (and some bugs in it were found). In fact, I used MACSYMA in the initial analysis that led to the reduction of the problem to the nonnegativity of polynomials like $P(v)$ of (3.1), and I used MACSYMA again in constructing the case-by-case proofs of the nonnegativity of the kind outlined above. As a result the proof can be verified even by people without access to computer algebra systems. The role of symbolic manipulation in this case was to spped up my own work by letting me kow which polynomials could be used and in helping to construct paper-and-pencil type proofs.

The application of computer algebra disucssed here is roughly intermediate between the case discussed in Section 2, where the output of the computation was the result, and the cases to be discussed next, when

computer algebra served only to provide insight. In the case of bounds for discriminants of number fields, the proof is much easier for the reader to check with the use of a computer algebra program than by following the proof given in the paper step-by-step. The point of going to the extra effort was to make the proof accessible even to those without access to such programs. In the cases of the result to be discussed next, even access to a symbolic manipulation system is of no help to the reader in verifying the proofs, although such access was very helpful in figuring out what the results ought be.

4. ENUMERATION OF 2,3-TREES

A 2,3-tree is a rooted, oriented tree each of whose nonleaf nodes has either two or three successors, and all of whom root-to-leaf paths has the same length. 2,3-trees and their generalizations, of *B*-trees, and used as data structures in situations where it is desirable to be able to insert and delete records in time that is logarithmic in the total number of records present. Let a_n denote the number of 2,3-trees of size (i.e., number of leaves) equal to n. In the paper [13], I showed that

$$a_n \sim n^{-1} q^n \; u(\log n) \quad \text{as } n \to \infty \; , \tag{4.1}$$

where $q = (1 + 5^{1/2})/2$ is the ''golden ratio,'' and $u(x)$ is a positive nonconstant continuous function which satisfies $u(x) = u(x + \log(u - q))$ for all real x. The proof contains no trace of the use of a symbolic manipulation program. However, such a program was very helpful in obtaining this result. The proof uses complex analyses techniques to study the generating function

$$f(z) = \sum_{n=1} ot\infty \; a_n z^n \; . \tag{4.2}$$

It is a general principle that the behavior of the coefficient a_n of a generating function of the form (4.2) is determined by the analytic behavior of the function $f(z)$ was its singularities. It is easy to show that the generating function $f(z)$ for 2,3-tries is analytic inside the circle $|z| = q^{-1}$, and also on that circle with the exception of the point $z = q^{-1}$, where it has a singularity. Since it was already known that

$$c_1 n^{-1} q^n < a_n < c_2 n^{-1} q^n \tag{4.3}$$

for some constants c_1 and c_2, I started out trying to prove the natural conjecture, namely that

$$a_n \sim cn^{-1}q^n \quad \text{as } n \to \infty \tag{4.4}$$

for some constant c. If (4.4) were to hold, though, one would expect that

$$f(x) \sim -c\log(qusp - 1 - z) \quad \text{as } z \to q^{-1} \ . \tag{4.5}$$

It was not too difficult to prove (4.5), but I was unable to obtain a good enough estimate for the difference of the two sides in (4.5) to prove (4.4). The reason for that was that (4.4) is false, and (4.1) is the correct result. The crucial analytic result that led to the proof of (4.1) was the relation

$$f(z) = -c\log(q^{-1} - z) + w(\log(q^{-1} - z)) + 0(|q^{-1} - z|) \tag{4.6}$$

as $z \to q^{-1}$, valid in a section of the form $|A \sim g\ (q^{-1} - z)| < \pi/2 + \delta$ for some $\delta > 0$, where $w(z)$ is a nonconstant analytic function which is periodic with period $\log(4 - q)$. The proof (4.6) involves intensive study of the polynomial iteration $T(z) = z^2 + z^3$, since $f(z)$ satisfies the functional equation

$$f(z) = z + f(z^2 + z^3) \ . \tag{4.7}$$

The conjecture that the behavior of $f(z)$ is given by (4.6) was made partially on the basis of computations with MACSYMA. Having failed to prove (4.4), I used the high-precision floating point facility of that system to explore the behavior of $f(x)$ for z mean q^{-1} by using the functional equation (4.7). By using the results of those computations and simultaneously investigating what phenomena could arise from iterating the mass $T(z) = z^2 \in z^3$, I was led to conjecture (4.6) and later to prove it. Also, by studying the polynomials $T(z)$, $T(T(z))$, etc., I was led to an understanding of how the behavior of $f(z)$ near its singularity at $z = q^1$ influenced the behavior of the coefficients a_n, which led to the proof of (4.1).

The role of computer algebra in this case was purely to provide insight, not to help with the proof itself. The result certainly could have been obtained without the use of symbols mathematics, and perhaps would have been, had I not had access to such systems, but the work would have been harder and would undoubted have taken longer.

5. STRING ENUMERATION

Another example where computer algebra had as its main function providing insight is given by the work on enumeration strings [6,7]. As is shown in those papers and the references cited in them, a surprising variety of problems can be reduced to the question of evaluating the number of strings of length n over some alphabet of q symbols, call it $f(n)$, which contain none of a given set, call them A, B, ..., T of words as subwords (i.e., blocks of consecutive characters). The basic result of [6], from which most of the others follow, is a simple formula for the generating function of $f(n)$,

$$F(z) = \sum_{n=0}^{\infty} f(n) z^{-n} \ . \tag{5.1}$$

(we take $f(0) = 1$. $F(x)$ is written as a power series on z^{-1} in order to obtain formulas.) To explain it, we define the correlation polynomial XY_z of the words $X = x_1 \cdots x_k$ and $Y = y_1 \cdots y_m$ to be

$$XY_z = \sum_{j=1}^{k} c_j z^{j-1} \ , \tag{5.2}$$

where $c_j = 1$ if the suffix of X of length j (i.e., $x_{k-j+1} \cdots x_k$) equals the prefix of Y of length j (i.e., $y_i \cdots y_j$), and $c_j = 0$ otherwise. (These are special values to take care of the case $m < k$.) The first result of [6] is that if the set of excluded words consists of a single word, call it A, then

$$F(z) = \frac{zAA_z}{1 + (z - q)AA_z} \ . \tag{5.3}$$

The proof of (5.3) that we gave is very simple, taken only a couple of lines, and certainly does not require, nor is it even helped by the use of computer algebra. (Moreover, this particular result had been obtained earlier by Solov's [17].) However, computer algebra was extremely helpful in obtaining that proof in the first place. Guibar and I started out by constructing finite state automata, reducing the number of states i them, and obtaining generating functions from them which determined $F(z)$. It was a very laborious process, and we used MACSYMA to carry out the symbolic manipulations that were involved. The formulas for $F(z)$ that came out were surprisingly compact, and inspection of a number of them showed that they were all of the simple form (5.3). Once we saw this elegant formula, we considered that a simple proof had to exist, and it did not take very long to find it. Then in this case the role of computer algebra

was to bring initial approaches. We might have found the result (5.3) by hand, but it would have taken us a lot longer.

Once a simple proof of (5.3) was found, it was a simple matter to generalize it. If the set of excluded words is $\{A, B, ..., T\}$, and no word in this set contains another (as we may assume without loss of generality), then $F(z)$ is determined by the following system of linear equations:

$$\begin{aligned}
&(z - q)\, F(z) + zF_A(z) + zF_p(z) + \cdots zF_T(z) = z \\
&F(z) - zAA_zF_A(z) - zBA_zF_B(z) - \cdots - zTA_zF_T(z) = 0 \\
&\vdots \\
&F(z) - zAT_zF_A(z) - zBT_zF_p(z) - \cdots - zTT_zF_T(z) = 0
\end{aligned} \tag{5.4}$$

Here $F_A(z)$, ..., $F_T(z)$ are generating functions related to $F(z)$. The system (5.4) can easily be shown to be nonsingular [6], and so $F(z)$ is seen to be a national function. Computer algebra is not needed in the proof of (5.4), and was not even used in deriving the proof, as it was fairly easy to generalize the proof of (5.3).

Computer algebra was also useful in deriving other string enumeration results, but again at the level of providing insight, rather than providing proof. As an example, we might consider the asymptotic estimation of $f(x)$. If the set of excluded words consists of A only, then (5.3) holds, and it can be shown that the behavior of $f(n)$ is determined almost completely by a first order pole of $F(z)$ near $z = q$ [6,7]. No computer algebra is involved there, as only basic complex analysis is used. However, when several words are excluded, the situation can become much more complicated. For example, it can happen that $f(n) = 0$ for all sufficiently large n, as occurs for the set {000, 111, 10} of $g = 2$, in which case $F(z)$ is a polynomial on z^{-1}. Thus in general we cannot say too much about the generating function $F(z)$ determined by (5.4). However, something can be done when we consider repetition patterns. Suppose that B is a nonperiodic word of length m [7], so it cannot be written as $B = CC...C$ for any word C shorter than B. A B^*-run of length k is a word $A = B''BB\cdots BB'$ of length k where B'' is a suffix of B and B' is a prefix of B. Thus, if $B = 010$, then $A = 001001$ in a $B^* - run$ of length σ. Once can ask about the distribution of lengths of maximal B^*-runs in random sequences. Now the number of strings of length n that contains no B^*-run of length k is exactly $f(n)$, the number of strings of length n that contain none of

$A(1), \ldots, A(m)$, where

$$A(n) = b_{m-r+1}\ b_{m-r+2}\ \ldots b_n\ BB\ \cdots\ BB\prime\ ,$$

and $B\prime = B\prime(r)$ is a prefix of B of the appropriate length. It can then be shown [7] that (for $q = 2$, say)

$$F(z) = \frac{z^{(k+1)m+1}\ (z^m - 1)^{m-1} + g_{1(z)}}{(z-2)z^{(k+1)m}\ (z^m - 1)^{m-1} + g_2(z)}\ , \tag{5.5}$$

where $g_1(z)$ and $g_2(z)$ are polynomials of small degrees and small coefficients that satisfy some additional conditions (see [7] for details). From (5.5) one can then show that a good approximation to $f(n)$ is given by

$$f(n) \approx 2^n\ \exp(-nm2^{-k-1})\ . \tag{5.6}$$

Also, $E_B^*(n)$, the expected length of the maximal B^*-run in a random binary sequence can be shown to satisfy

$$E_B^*(n) = \log n - 3/2 + \sigma(\log 2)^{-1} + \log m \tag{5.7}$$
$$+ v(\log n + \log m) + \sigma(1) \text{ as } n \to \infty\ ,$$

when lgx is the logarithm of x to base 2, γ is Eulen's construct (0.577...), and $v(x)$ is a certain nonconstant continuous periodic function of period 1 and near value 0.

The proof of (5.6) and (5.7) did not rely in any way on computer algebra. Even (5.5), which is a very algebraic result, and was proved by some very messy algebraic manipulations, was proved without the help of symbolic manipulation programs; since the algebra involve exponents and as k and m as variables, as small as z. However, computer algebra was very helpful in guessing that something like (5.5) might hold. The function $F(z)$ was computed for various B*-runs, and from their algebraic forms and their singularities, the correct conjectures were derived.

6. MISCELLANEOUS APPLICATIONS AND CONCLUSIONS

Many more examples of the application of computer algebra to mathematics could be presented. I will note a few minor applications. In recent work on discrete logarithms [15], it was desirable to verify that for various degrees n, there are polynomials of the form $x^n + f(x)$ which an irreducible over $GF(2)$ and

such that the degree of $f(x)$ as small. Since about one out of n polynomials of degree n is irreducible over $GF(2)$, it was expected and confirmed by computation that one can find such $f(x)$ with dig $f(x) \leq \log_2 n$. This was a straightforward problem perfectly suited for a symbolic manipulation program but too tedious to do by hand. In the work on cellular automata [9], computer algebra was used to explore the behavior of automata with so-called linear evolution rules. Once the data was collected, conjectures were made and proved by ordinary methods. In the work on knapsack cryptosystems described in [14], the theoretical attacks that were proposed were tested on a symbolic manipulation system mainly because it was easy to program all the required operations (random number generation, modular multiplication of large integers, reduction of lattice cases) in it. In the role on new bounds for ‘‘kissing numbers,’’ [16], computer algebra was used to generate the required Jacobi polynomials. In all of the above applications, computer algebra was not essential, but it was very useful by taking care of most of the drudgery.

One of the important functions of computer algebra is to carry out little computational tasks which can be done by hand, but are tedious and not interesting. In these cases computer algebra can make mathematical research somewhat more efficient and much more pleasant. In some cases, such as the computation described on Section 2, computer algebra can carry out giant computations that appear to be unavoidable, but an impractical to do by hand. The most important application of computer algebra, though, are to situations such as those in section 4 and 5, where the symbolic computation serve to provide insight and lead to standard proofs being constructed.

In summary,I feel that symbolic manipulation programs are already very useful, and will become even more useful in mathematical research. Yet I see their role as essential that of a ‘‘bigger scratchpad,’’ to quote P. Halmor. I feel safe in predicting that they will not change basic

REFERENCES

[1] J. Calmet and J. A. van Hulzen, Computer algebraic applications, pp. 245-258 in *Computer Algebra*, B. Buchberger, G. E. Collins and R. Loos, eds., Computing Suppl. **4** (1982), Springer

[2] W. S. Brown, *ALTRAN User's Manual*, 4th ed., Bell Laboratories 1977.

[3] MATHLAB Group; *MACSYMA Reference Manual*, MIT Laboratory for Computer Service 1977.

[4] C. A. Cole, S. Wolfson, et al., *SMP-Handbook*, Caltech 1981.

[5] K. O. Geddes, G. H. Gonnet, and B. W. Chan, *MAPLE User's Manual* Univ. of Waterloo, 1983.

[6] E. Bombien, Thompson]s problem $(\sigma^2 = 3)$, (will appendices by A. Odlyzko and D. Hurt), Inventions math. **58** (1980), 77-100.

[7] A. M. Odlyzko, Lower bounds for discriminants of number fields, Act. Arith. **29** (1976), 275-297.

[8] A. M. Odlyzko, Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on Sahmis's signature scheme, *IEEE Trans. Inform. Theory IT-30* (1984), 594-600. nature of mathematical research.

[9] A. M. Odlyzko and N. J. A. Sloane, New bound son the number of unit spheres that can tough a unit sphere in a dimension, *J. Conab. Theory (A)* **26** (1979), 210-214.

[10] R. W. Hamming, *Numerical Methods for Scientists and Engineer*s 2nd ed., McGraw Hill 1973.

[11] A. M. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, Proc. Eurpcrypt 1984, Springer Lecture Notes in Computer Science to appear.

[12] A. D. Solviger, A combinatorial identity and its application to the problem concerning the first occurrence of a more event, *Theroy Prob. Appl* **11** (1966), 276-282.

[13] O. Martin, A. M. Odlyzko, and S. Wolfson, Algebraic properties of cellular automata, Commun. Math. Physics **93** (1984), 219-258.

[14] L. J. Guibar and A. M. Odlyzko, string overlap, pattern matching, and constrictive gammas. J. Comb. Theory (A) **30** (1981), 183-708.

[15] L. J. Guiban and A. M. Odlyzko, Long repetitive patterns in random sequences, Z. Wahrscheinlichkeits v. Gib. **53** (1980), 241-262.

[16] A. M. Odlyzko, Periodic oscillations of coefficients of power series that satisfy functional equations, Advances in Math, **44** (1982), 180-205.

Application of Symbolic Mathematics to Mathematics

A. M. Odlyzko

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

Some of the most interesting applications of symbolic mathematics are in mathematics itself. Areas of both pure and applied mathematics, including cryptography, coding theory, probablity theory, analysis, combinatorics, and number theory, have all gained from the availability of the new symbolic manipulation tools. These tools have been used to prove a number of results directly. Their main application, however, has been to obtain insight into behavior of various mathematical objects, which then led to conventional proofs being constructed.