

AD637127

No. 2032

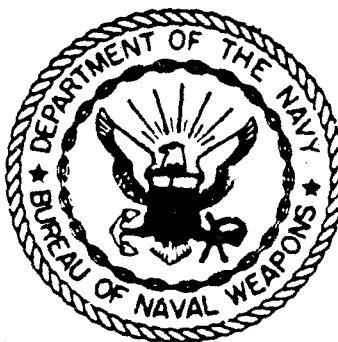
TECHNICAL REPORT

MODELS FOR MATHEMATICAL SYSTEMS

by

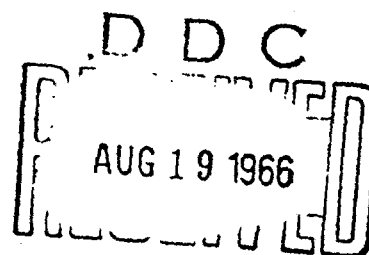
Alfred H. Morris, Jr.
Computation and Analysis Laboratory

CLEARINGHOUSE FOR FEDERAL SCIENTIFIC AND TECHNICAL INFORMATION			
Hardcopy	Microfilm		
		29	pp
ARCHIVE COPY			



U. S. NAVAL WEAPONS LABORATORY
DAHLGREN, VIRGINIA

20050223169



Best Available Copy

U. S. Naval Weapons Laboratory
Dahlgren, Virginia

REF ID: A63127

Models for Mathematical Systems

by

Alfred H. Morris, Jr.
Computation and Analysis Laboratory

NWL REPORT NO. 2032

30 March 1966

Distribution of this document is unlimited.

TABLE OF CONTENTS

	<u>Page</u>
Abstract	ii
Foreword	iii
Introduction	1
Initial Formulation of the Model	2
Functions.	6
Definition of the ALGEBRA System	10
Polynomial Algebras.	12
Series Algebras.	15
Differential Forms	16
On the Nature of Symbol Manipulation Languages	18
References	22
Appendix A	
Distribution	

ABSTRACT

The purpose of this paper is to describe and illustrate a program, written for the IBM 7090, which defines a model for a finite collection of algebras for the computer. It is shown that the program contains a structure broad enough in scope to allow one to perform operations on such diverse mathematical concepts as differential equations, infinite series, and differential forms in a simple yet comprehensive manner, while also serving as a foundation upon which a variety of higher-level symbolic manipulation languages can be developed.

FOREWORD

The work described in this report was performed in the Programming Systems Branch of the Computer Programming Division with Foundational Research funds (R360FR103/210-1/R0110101).

This report was prepared for presentation by the author at the ACM SICSAM Symposium on Symbolic and Algebraic Manipulation held in Washington, D. C. in March 1966.

APPROVED FOR RELEASE

/S/ BERNARD SMITH
Technical Director

Introduction

It has become increasingly evident in recent years that the intricate numerical procedures that have been designed for the computer have not been adequate for handling mathematical and physical problems requiring extensive symbolic analysis; what has been needed is the development of highly sophisticated symbolic techniques treating various analytical problems in a variety of ways. Studies have been made regarding this problem, but because of the many inherent difficulties involved, much confusion has arisen regarding not only the nature of the computational procedures that could best be handled using the computer, but the structure that would be necessary for the development of any higher-level language which would be designed to manipulate such data. In the present paper, we shall attempt to rectify this situation in part by defining a model for a finite collection of (associative) algebras. It will then be seen that the program, called ALGEBRA, will provide us with a structure broad enough in scope to allow us to perform operations on such diverse mathematical concepts as differential equations, infinite series, and differential forms in a fairly simple yet comprehensive manner. Moreover, the model will serve as a foundation upon which a variety of higher-level languages dealing with symbolic manipulation can be constructed, depending only on the types of mathematical computational procedures desired.

Before we proceed, however, several remarks should be made regarding the type of programming language that is appropriate for developing

symbolic computational procedures of any type. In general, assembly languages are unsuitable because of the complexity of even the most elementary operations on complex data structures that are generally required, whereas the algebraic languages, such as FORTRAN and ALGOL, are excluded because of their relatively primitive storage allocation capabilities. The list processing languages, however, are quite adequate for this purpose because of their ability to store and manipulate exceedingly complex data structures. Of the systems in this category, the LISP 1.5 language (see reference 1 or 6) was used in the development of ALGEBRA because of its simplicity and generality.

Initial Formulation of the Model

Let $\mathcal{R} = \{R_1, \dots, R_n\}$ be a set of commutative rings and $\mathcal{A} = \{A_1, \dots, A_n\}$ a collection of algebras, each of which is operated on by a ring in $\mathcal{R}$ (see reference 3). It will be assumed that the rings and algebras have a multiplicative identity "1" and that each algebra A_v is an algebra of some module generated by a recursive set X_v .¹ For convenience of reference, the algebras will be indexed by the variable v and the rings by the variable r . Moreover, for each ring R_r of $\mathcal{R}$, a computable function $s \rightarrow \text{scalar}_r(s)$ will be assumed to exist which evaluates expressions s of the forms $a_1 + \dots + a_p$, $a_1 \cdots a_p$, and a^q , where $a, a_1, \dots, a_p$ are scalars (i.e., elements of the ring) and q is a non-negative integer. In general, the definition of this mapping will be extended to evaluate

¹A set A will be said to be recursive if for any expression x , the predicate $x \in A$ is computable (see reference 4 or 5).

re elaborate expressions containing the scalars, but this will necessarily vary with the structure of R , and how we wish to use it. In any case, the function will always serve as the sole criterion for whatever assumptions are being made regarding this ring.

Consider now an algebra A defined over the ring R , the elements which will be called vectors. We shall find it useful for a variety of reasons, including input-output and uniqueness considerations, to assume that there exists a linear ordering relation " $<$ " of the set A . We may also wish to presume that the algebra is either free or semisimple, though this will not be necessary. It will be convenient, however, to be able to represent the vectors in various forms, but this so occurs whenever the theory is developed from a mathematical viewpoint.

Now any non-zero vector x in A can always be written in the form $x = c_1 (\varphi_1^1)^{\lambda_{11}} \dots (\varphi_{r_1}^1)^{\lambda_{1r_1}} + \dots + c_n (\varphi_1^n)^{\lambda_{n1}} \dots (\varphi_{r_n}^n)^{\lambda_{nr_n}}$, where each $c_i \neq 0$ is a scalar, λ_{ij} is a positive integer, and $\varphi_j^i \in X_i \cup \{1\}$. If the algebra A is commutative, it will be presumed that the terms of the product $(\varphi_1^i)^{\lambda_{i1}} \dots (\varphi_{r_i}^i)^{\lambda_{ir_i}}$ are ordered so that $\varphi_1^i < \varphi_2^i < \dots < \varphi_{r_i}^i$; otherwise, their ordering will have to be specified in the hypothesis given concerning the algebra. In any case, with the further assumption that $(\varphi_1^1)^{\lambda_{11}} \dots (\varphi_{r_1}^1)^{\lambda_{1r_1}} < \dots < (\varphi_1^n)^{\lambda_{n1}} \dots (\varphi_{r_n}^n)^{\lambda_{nr_n}}$, it follows that associated with every vector is an expression of the form $\bar{x} = ((c_1, (\varphi_1^1)^{\lambda_{11}}, \dots, (\varphi_{r_1}^1)^{\lambda_{1r_1}}), \dots, (c_n, (\varphi_1^n)^{\lambda_{n1}}, \dots, (\varphi_{r_n}^n)^{\lambda_{nr_n}}))$. This form will be called

the canonical representation of x and will be denoted by $\text{part}(x, X_r)$. For completeness, we shall also require $\text{part}(0, X_r)$ to have the value $((0, 1))$.

Whereas the mathematical form of a vector may or may not be useful for an analyst, depending on the nature of the problem under consideration, the form will always be found to be awkward for a computer because of the partitioning that necessarily will ensue. Consequently, the usual procedure for manipulating these elements is to first partition them into their canonical representations, use these representations for carrying out the operations that are needed, and then return the answers in their mathematical forms. In so far as the uniqueness of the canonical forms is concerned, and this will be discussed in detail later, let it suffice to say for the moment that this will most certainly depend on, among other things, the assumptions made regarding the basic structure of A_r .

We now consider the addition and multiplication operations for the vectors of A_r , which, for convenience, will be denoted by the symbols $\oplus$ and $*$ respectively. The addition function is particularly easy to describe, being defined as follows:

1. For any vector x , $x \oplus 0 = 0 \oplus x = x$
2. If x is a non-zero vector having the canonical form $((a, \psi_1^{\mu_1}, \dots, \psi_n^{\mu_n}))$, y is an arbitrary non-zero vector having the canonical form $((b_1, (\phi_1^1)^{\lambda_{11}}, \dots, (\phi_{r_1}^1)^{\lambda_{1r_1}}), \dots,$

$(b_n, (\varphi_1^n)^{\lambda_{n1}}, \dots, (\varphi_r^n)^{\lambda_{rn}}))$, and $\xi_i = (\varphi_1^i)^{\lambda_{i1}} \dots (\varphi_r^i)^{\lambda_{ir}}$
 $(i = 1, \dots, n)$, then

$$= \begin{cases} b_1 \xi_1 + \dots + b_i \xi_i + a \psi_1^{\mu_1} \dots \psi_r^{\mu_r} + b_{i+1} \xi_{i+1} + \dots + b_n \xi_n \\ \quad \text{if } \xi_i < \psi_1^{\mu_1} \dots \psi_r^{\mu_r} < \xi_{i+1} \\ b_1 \xi_1 + \dots + b_{i-1} \xi_{i-1} + \text{scalar}_r(a+b_i) * \xi_i + b_{i+1} \xi_{i+1} + b_n \xi_n \\ \quad \text{if } \xi_i = \psi_1^{\mu_1} \dots \psi_r^{\mu_r} \text{ and } \text{scalar}_r(a+b_i) \neq 0 \\ b_1 \xi_1 + \dots + b_{i-1} \xi_{i-1} + b_{i+1} \xi_{i+1} + \dots + b_n \xi_n \\ \quad \text{if } \xi_i = \psi_1^{\mu_1} \dots \psi_r^{\mu_r} \text{ and } \text{scalar}_r(a+b_i) = 0 \end{cases}$$

3. If x is a vector having the canonical form $((a_1, (\psi_1^1)^{\mu_{11}}, \dots, (\psi_1^{\mu_{1s}})^{\mu_{1s}}), \dots, (a_{r1}, (\psi_1^r)^{\mu_{r1}}, \dots, (\psi_r^{\mu_{rs}})^{\mu_{rs}}))$, y is an arbitrary vector,
 $= (\psi_1^i)^{\mu_{i1}} \dots (\psi_r^i)^{\mu_{ir}}$, then $x \oplus y = a_1 \xi_1 \oplus (a_2 \xi_2 \oplus (\dots \oplus (a_r \xi_r \oplus y) \dots))$.
 Similarly, using the associative and distributive properties of the
 operation $*$, it is clear that the definition of $x * y$ reduces to the case
 where x has the canonical form $((a, \psi_1^{\mu_1}, \dots, \psi_r^{\mu_r}))$ and y has the canonical
 form $((b, \varphi_1^{\lambda_1}, \dots, \varphi_r^{\lambda_r}))$. However, since the definition of this product
 agrees with the algebra under consideration, it is evident that it will
 have to be given by a function $x, y \rightarrow \text{vector}_v(x, y)$. This map will always
 be assumed to be computable, of course.

EXAMPLE:

Let R be the standard fixed point arithmetic indexed by the value
 $\neq 0$, so that the function $s \rightarrow \text{scalar}_0(s)$ has as its value the sum
 (product) of the numbers $a_1, \dots, a_p$ whenever $s = a_1 + \dots + a_p$
 $= a_1 \cdots a_p$) and the exponentiation of the number a to the non-negative

integral power n whenever $s = a^n$. If A is the polynomial algebra of the set $X = \{x\}$ over the ring R (see reference 2), it is then evident that the mapping $\lambda, \mu \rightarrow \text{vector}_1(\lambda, \mu)$ can be easily defined by assigning the value $\text{scalar}_0(a \cdot b)x^{\text{scalar}_0(k+l)}$ if $\text{scalar}_0(k+l) \neq 0$ and $\text{scalar}_0(a \cdot b)$ if $\text{scalar}_0(k+l) = 0$, where $\lambda = ax^k$ and $\mu = bx^l$. Consequently, it follows that the expressions $K + L$ and $K \cdot L$ easily reduce to the respective values $7 + 4x + 3x^2$ and $12 + 16x + 9x^2 + 12x^3$ for $K = 3 + 4x$ and $L = 4 + 3x^2$.

Functions

Let A_r be an algebra in $\mathcal{H}$ defined over a ring R_r . A collection of vector-valued functions (with vectors and/or scalars as arguments) is generally associated with this algebra, some resulting directly from the algebraic structure of A_r and others depending on whatever further restrictions are placed on A_r . In any case, since the choice of these maps will necessarily vary with the algebra under consideration and how it is to be used, it follows that the model ALGEBRA will have to contain a general procedure for storing and manipulating any assumptions that will be required.

Before the vector-valued functions are considered, however, certain remarks should be made regarding the structure of the algebra. In particular, although it is true that in certain cases we shall be interested in A_r for its algebraic properties only, it is also clear that at other times the topological aspects of the algebra will be of dominant importance. For example, the assumption that the algebra is a Banach space

is a fairly common requirement, in which case, we would most certainly be interested in the bounded linear operators of this space. In fact, we might even be interested in the vector-valued Borel functions. In many cases, however, the functions that we shall be interested in, whether they are continuous or not, will not be computable. And for that matter, we might be interested in using a computable function for its representation only, and not its value.

Heretofore, it has been assumed that A_v is an algebra of some module M_v , where M_v is generated by the set X_v . If, however, we wish for our algebra to have a topological structure, this assumption may be relaxed by requiring that X_v be a set that is dense in M_v . The set X_v can be modified in other ways also. In particular, consider a function f associated with A_v . Then for any expression¹ s of the form $f(x_1, \dots, x_n)$, where each x_i is a vector or scalar, if it isn't desirable to compute s for the arguments under consideration, or it is not possible to do so, the expression will be regarded as being irreducible relative to the hypotheses being made, and hence will be treated as an element of X_v .

¹ An expression in A_v is defined to be (1) a scalar or vector in A_v , (2) a representation of one of the forms $x_1 + \dots + x_n$ or $x_1 \dots x_n$, where each x_i is an expression, (3) a representation of the form x^n , where x is an expression and n is a positive integer, or (4) a representation of the form $f(x_1, \dots, x_n)$, where f is an associated mapping and each x_i is an expression.

In general, if no assumptions are made concerning the function, we have no choice but to regard it as being irreducible whenever it is used. If, on the other hand, f is to satisfy certain hypotheses, a function h is associated with f . This mapping, called the axiom or hypothesis function of f , acts as a partial evaluation function for f , stating which properties of the function are to be assumed, and how they will be used. In particular, if f has the property $e_i(x_1, \dots, x_n)$ whenever the condition $p_i(x_1, \dots, x_n)$ is fulfilled (for $i = 1, 2, \dots, n$), then h is the function of the variables $x_1, \dots, x_n$, having for its value $e_i(x_1, \dots, x_n)$ if the predicate $p_i(x_1, \dots, x_n)$ is satisfied, and the irreducible expression $f(x_1, \dots, x_n)$ otherwise.

Let $f_1, \dots, f_n$ be a collection of functions which are to be associated with A , and which will satisfy the axiom functions $h_1, \dots, h_n$, respectively. These functions and their properties are referred to in ALGEBRA by a variable, called AXIOM, whose value is the list $(f_1, h_1, f_2, h_2, \dots, f_n, h_n)$. If no assumptions are to be made, AXIOM will have as its value the empty set $\emptyset$. In any case, whenever a vector-value expression $f(x_1, \dots, x_n)$ is encountered, if it is not a sum, product, or exponential expression of the form x^y where x is a vector and y is a non-negative integer, ALGEBRA first checks to see if f is a member of the set AXIOM. If it is found to be in AXIOM, the expression is evaluated using the corresponding axiom mapping of f ; otherwise, s is regarded as being irreducible and hence, a member of X .

EXAMPLE:

Let A be the polynomial algebra defined in the example on Page 5 and assume that the trigonometric functions sin and cos are associated with A. If they are assigned the respective axiom functions hsin and hcos, where

$$hsin(s) = \begin{cases} 0 & \text{if } s = 0 \\ (-1)*hsin[(-1)*s] & \text{if } s = n \text{ or } n\xi \text{ (} n=-1,-2,\dots \text{ and } \xi \text{ a vector)} \\ hsin(\xi)*hcos[(n-1)\xi] \oplus hcos(\xi)*hsin[(n-1)\xi] & \text{if } s = n\xi \text{ (} n=2,3,\dots \text{ and } \xi \text{ a vector)} \\ hsin(\xi_1)*hcos(\xi_2+\dots+\xi_n) \oplus hcos(\xi_1)*hsin(\xi_2+\dots+\xi_n) & \text{if } s = \xi_1+\dots+\xi_n \text{ (each } \xi_i \text{ a vector)} \\ \sin(s) & \text{otherwise} \end{cases}$$

and

$$hcos(s) = \begin{cases} 1 & \text{if } s = 0 \\ hcos[(-1)*s] & \text{if } s = n \text{ or } n\xi \text{ (} n=-1,-2,\dots \text{ and } \xi \text{ a vector)} \\ hcos(\xi)*hcos[(n-1)\xi] \oplus (-1)*hsin(\xi)*hsin[(n-1)\xi] & \text{if } s = n\xi \text{ (} n=2,3,\dots \text{ and } \xi \text{ a vector)} \\ hcos(\xi_1)*hcos(\xi_2+\dots+\xi_n) \oplus (-1)*hsin(\xi_1)*hsin(\xi_2+\dots+\xi_n) & \text{if } s = \xi_1+\dots+\xi_n \text{ (each } \xi_i \text{ a vector)} \\ \cos(s) & \text{otherwise} \end{cases}$$

it then follows that any expression s of the form $\sum_{i=1}^T a_i [\sin(p_i x)]^{n_i}$.

$[\cos(q_i x)]^{m_i}$ (a_i, p_i , and q_i are arbitrary integers and $m_i, n_i = 0, 1, 2, \dots$)

easily reduces to a vector of the form $\sum_j b_j [\sin(x)]^{r_j} [\cos(x)]^{s_j}$.

Moreover, if it is also assumed that $\sin^2(\xi) + \cos^2(\xi) = 1$ for every vector ξ in A (i.e., the mapping $\lambda, \mu \rightarrow \text{vector}_1(\lambda, \mu)$ is modified by letting $\text{vector}_1[\cos(\xi), \cos(\xi)] = 1 - [\sin(\xi)]^2$), then s completely collapses to an expression of the form $\sum_r \alpha_r [\sin(x)]^{r_1} + \sum_s \beta_s [\sin(x)]^{s_1} \cos(x)$. Thus, we see that the trigonometric polynomials over the scalar ring R are easily manipulated whenever this algebra is used.

Definition of the ALGEBRA System

In the previous two sections a procedure was developed for first formulating the basic structure of an algebra A_v , and then associating with this algebra an arbitrary collection of vector-valued maps, along with any associated assumptions that were required. The generating set X_v of this algebra was seen to be well-defined, consisting entirely of explicitly defined vectors and irreducible expressions of the form $f(x_1, \dots, x_n)$. Since these elements were evaluated to themselves, it followed recursively (by the definitions of the $\oplus, *$, and axiom mappings) that every combination s of expressions of these elements would reduce to a unique value, the form of which would vary only when the underlying structure was changed. We now define the ALGEBRA system to be this reduction procedure, in which case it is then representable as a function of the form $s, X_v, r, v, \text{AXIOM} \rightarrow \text{ALGEBRA}(s, X_v, r, v, \text{AXIOM})$.

Let s be an expression in A_v having the value $\text{ALGEBRA}(s, X_v, r, v, \text{AXIOM}) = x$. Since x and s obviously represent the same element of the algebra, it is clear that the mapping defines an equivalence relation on the set of expressions of A_v , requiring any two, say s_1 and s_2 , to be equivalent whenever $\text{ALGEBRA}(s_1, X_v, r, v, \text{AXIOM}) = \text{ALGEBRA}(s_2, X_v, r, v, \text{AXIOM})$. Consequently, since the hypotheses concerning the structure

of the algebra have been imbedded in the functions $\alpha \rightarrow \text{scalar}, (\alpha)$ and $\lambda, \mu \rightarrow \text{vector}, (\lambda, \mu)$ and in the variable AXIOM, it follows that the function ALGEBRA does indeed characterize the structure of A_v as we have defined it. Moreover, since the values of this mapping are unique, they serve, in effect, as representations for their equivalence classes. Thus, if the definition of the canonical form of a vector s is modified to be that of the vector $\text{ALGEBRA}(s, X_v, r, v, \text{AXIOM})$, we then note that the canonical form of a vector will always be unique.

In reflecting on the structure of the mapping ALGEBRA, it should first be emphasized that all we have succeeded in doing is implementing a model for a finite collection of algebras on the computer; no techniques have been developed for proving theorems about these algebras. The question of theorem-proving in general is a distinct problem, requiring the formulation of various techniques for handling the learning and choosing procedures that are necessarily involved. Secondly, many of the restrictions that were placed on the collection of sets n were not really necessary. It is evident, for example, that ALGEBRA could have easily been defined to handle a collection of groups or monoids instead. In any case, having actually formed a simple but comprehensive model of a collection of mathematical systems, each of which contains a fairly complex underlying structure, we have, in effect, provided ourselves with an extremely powerful tool for treating a vast array of problems in a systematic manner which could not even have been considered otherwise.

Polynomial Algebras

Let R be an arbitrary ring indexed by the value $v = v_0$ and described by the mapping $s \rightarrow \text{scalar}_{r_0}(s)$, X a recursive collection of non-numerical atoms or indexed variables designated by the value $v = v_0$ (see reference 1), and A an extension of the polynomial algebra defined on the set of variables X over the ring R (see reference 2), allowing in addition any appropriate associated functions that are desired. It then follows that A will be well-defined whenever the product mapping $x, y \rightarrow \text{vector}_{v_0}(x, y)$ is evaluated. But this function is easily defined, having for the non-zero vectors $x = a\varphi_1^{r_1} \dots \varphi_n^{r_n}$ and $y = b\psi_1^{s_1} \dots \psi_n^{s_n}$ the value

$$\text{scalar}_{r_0}(a \cdot b) \varphi_1^{r_1} * (\varphi_2^{r_2} * (\dots * (\varphi_n^{r_n} * (\psi_1^{s_1} \dots \psi_n^{s_n}))) \dots),$$

where

$$\varphi^r * (\psi_1^{s_1} \dots \psi_n^{s_n}) = \begin{cases} \psi_1^{s_1} \dots \psi_{\alpha-1}^{s_{\alpha-1}} \varphi^r \psi_{\alpha}^{s_{\alpha}} \dots \psi_n^{s_n} & \text{if } \psi_{\alpha-1} < \varphi < \psi_{\alpha} \\ \psi_1^{s_1} \dots \psi_{\alpha-1}^{s_{\alpha-1}} \psi_{\alpha+1}^{s_{\alpha+1}} \dots \psi_n^{s_n} & \text{if } \varphi = \psi_{\alpha} \text{ and} \\ \text{ALGEBRA}(r+s_{\alpha}, x, r_0, v_0, \text{AXIOM}) = 0 \\ \text{scalar}_{r_0}(\varphi^{r+s_{\alpha}}) \psi_1^{s_1} \dots \psi_{\alpha-1}^{s_{\alpha-1}} \psi_{\alpha+1}^{s_{\alpha+1}} \dots \psi_n^{s_n} & \text{if} \\ \varphi = \psi_{\alpha}, \varphi \text{ and } r+s_{\alpha} \text{ are scalars,} \\ \text{and } \varphi^{r+s_{\alpha}} \text{ is an expression in our} \\ \text{ring of scalars} \\ \psi_1^{s_1} \dots \psi_{\alpha-1}^{s_{\alpha-1}} \varphi^{r+s_{\alpha}} \psi_{\alpha+1}^{s_{\alpha+1}} \dots \psi_n^{s_n} & \text{if } \varphi = \psi_{\alpha} \text{ and} \\ \varphi^{r+s_{\alpha}}, \text{ is irreducible.} \\ \text{ALGEBRA}(\varphi^{r+s_{\alpha}}, X, r_0, v_0, \text{AXIOM}) * \\ (\psi_1^{s_1} \dots \psi_{\alpha-1}^{s_{\alpha-1}} \psi_{\alpha+1}^{s_{\alpha+1}} \dots \psi_n^{s_n}) & \text{if } \varphi = \psi_{\alpha} \\ \text{and } \varphi^{r+s_{\alpha}} \text{ is reducible} \end{cases}$$

We now consider several useful applications of this algebra.

1) The SIMPLIFY Algebra

This algebra is the algebra most generally used for manipulating symbolic mathematical expressions, having for its scalar ring the standard fixed and floating point arithmetic and for its generator set X the collection of non-numerical atoms which are not being used as vectors for other algebras. For convenience, we shall denote the value of its implementation mapping $s \rightarrow \text{ALGEBRA}(s, X, r_0, v_0, \text{AXIOM})$ by $\text{SIMPLIFY}(s)$.

Assume that the standard differentiation function $\text{diff}: s, x \rightarrow \frac{\partial s}{\partial x}$ is now associated with SIMPLIFY (see reference 2). If $\text{AXIOM} = \emptyset$ (the empty set), it is then clear that although $\text{SIMPLIFY}(K + x) = 5x + 5y + 3xy$ for the vector $K = 4x + 5y + 3xy$ (where $x < y < xy$), since no assumptions are being made regarding the differentiation mapping, $\text{SIMPLIFY}[\text{diff}(K, x)]$ has for its value $\text{diff}(4x + 5y + 3xy, x)$, considering this expression as being irreducible and hence an element of X . If, on the other hand, $h(s, x)$ is the standard axiom mapping for diff^1 and $\text{AXIOM} = (\text{diff}, k)$, then $\text{SIMPLIFY}[\text{diff}(K, x)]$ would have as its value $h(K, x) = 4 + 3y$. Thus, if the two trigonometric functions $\sin$ and $\cos$ are also associated with SIMPLIFY and $h[\sin(x), x]$ is defined to have the value $\cos(x)$, it is evident that whereas this algebra is appropriate for computing expressions of the form $\cos(x) + \text{diff}[\sin(x), x]$, it is totally inadequate for handling either differential equations or polynomials with symbolic coefficients.

¹ $h(s, x)$ is defined for any expression s and any atomic vector x by requiring that (1) $h(s, x) = 0$ whenever s is a scalar or atomic vector different from x , (2) $h(x, x) = 1$, (3) $h(s, x) = h(s_1, x) + \dots + h(s_p, x)$ for $s = s_1 + \dots + s_p$, and (4) $h(s, x) = h(s_1, x)s_2 \dots s_p + \dots + s_1 \dots s_{p-1} h(s_p, x)$ for $s = s_1 \dots s_p$.

2) The MULT and MULTILINEAR Algebras

The MULT algebra uses as its scalar ring the SIMPLIFY algebra and as its generator set a finite collection $\{x_1, \dots, x_n\}$ of non-numerical atoms or indexed variables. The standard differentiation mapping $\text{diff}: s, x \rightarrow \frac{\partial s}{\partial x}$ is again associated with this algebra, being defined as before whenever $x = x_i$ for some x_i . The definition of diff, however, can be extended to also cover the case where x is a non-numerical atom in the ring SIMPLIFY. This is done by requiring that: (1) if s is a scalar, then $\text{diff}(s, x)$ will be computed as an expression in SIMPLIFY, (2) $\text{diff}(s, x)$ will be regarded as being irreducible for any irreducible vector s in MULT, so that in effect, each x_i is a function of x , (3) $\text{diff}(s, x) = \text{diff}(s_1, x) + \dots + \text{diff}(s_p, x)$ for any $s = s_1 + \dots + s_p$, and (4) $\text{diff}(s, x) = \text{diff}(s_1, x)s_2 \cdots s_p + \dots + s_1 \cdots s_{p-1} \text{diff}(s_p, x)$ for any $s = s_1 \cdots s_p$. Noting then that the expression $xx_1 + \text{diff}[\sin(x)x_1, x]$ reduces to $[x + \cos(x)]x_1 + \sin(x)\text{diff}(x_1, x)$, it is evident that any differential equation with numerical coefficients is easily handled by this algebra.

In order to manipulate differential equations with symbolic coefficients, the MULTILINEAR algebra must be used. This algebra is a modification of the MULT algebra, having as its scalar ring the MULT algebra (with generating set $\{x_1, \dots, x_n\}$) and as its generator set a finite sequence $\{y_1, \dots, y_n\}$ of non-numerical atoms or indexed variables each of which is assumed to be a function of the variables $\{x_1, \dots, x_n\}$. The definition of the differentiation mapping $\text{diff}(s, x)$ is then modeled after that of the MULT algebra, allowing x to be either x_i or y_j and requiring that the expression $\text{diff}(s, x_i)$ be irreducible for any irreducible

expression s in the algebra. Thus, it would follow that the expression $\text{diff}[ax_1 \sin(y_j), x_1] + \text{diff}[y_k, y_j]$ would reduce to $a \cdot \sin(y_j) + ax_1 \cos(y_j) \text{diff}(y_j, x_1)$ for $j \neq k$.

Series Algebras

Let R be a commutative ring with multiplicative identity 1 , Z the ring of integers, and A the collection of all functions $f: Z \rightarrow R$ such that for some integer n_f , $f(n) = 0$ for all $n < n_f$ (the value of n_f will necessarily vary with the choice of f). Denoting the value of $f(n)$ also by f_n , it then is clear that A is an algebra over R whenever $(af)_n = af_n$, $(f + g)_n = f_n + g_n$, and $(fg)_n = \sum_{i+j=n} f_i g_j$ for any scalar a and any vectors f and g . This algebra is, of course, isomorphic to the algebra of series of the form $\sum_{n=k}^{\infty} a_n x^n$ for some variable x .

Regarding n now as a fixed variable, represent the elements of A by the expressions $f(n)$ instead of f . A can then be considered as a modification of the MULT algebra (whenever R is taken to be the SIMPLIFY algebra) having as its generating set X the collection of maps f_n such that $f(Z - \{0\}) = \{0\}$ and as its basic product the value $\sum_{i+j=n} f_i g_j$ instead of $f_n g_n$. However, if A is considered as an algebra over the polynomial ring defined on the set $\{x\}$, in which case $[(ax^k)f]_n = af_{n-k}$ for any scalar of the form ax^k and any vector f , it is then evident that the algebra is a modification of the MULTILINEAR algebra with generator set X , whose scalar ring is the MULT algebra with generating set $\{x\}$. In either case, since the notation being used to represent our series will most likely differ from that of the elements $f(n)$ of A , a short input-output package will have to be supplied to convert from one system of notation to the other. But this is always the case, of course, whenever we deal with isomorphic algebras.

Assuming that the standard differentiation map $\frac{d}{dx} : f_n \rightarrow (n+1)f_{n+1}$ is associated with A, it then follows that the algebra will provide a direct procedure for reducing ordinary differential equations (in x) to their difference equation components. Moreover, if A and MULTILINEAR are used in conjunction with one another for handling these equations, first transforming them and then reducing them, we shall find that we have developed a highly useful procedure for examining their underlying structure.

Differential Forms

Let R be the MULTILINEAR algebra, having as its generating set a sequence of the form $\{y_1, \dots, y_n\}$, the choice of which may vary with the problem under consideration, and as its scalar ring the MULT algebra with generator set $\{x_1, \dots, x_n\}$. Assuming also that the functions being associated with MULT and MULTILINEAR are differentiable, we then consider the exterior algebra A of the free module M generated by the set $\{dx_1, \dots, dx_n\}$ over R, where $\{dx_1, \dots, dx_n\}$ is a collection of n atoms or indexed variables (see reference 2).

Noting that the non-scalar elements of this algebra are of the form $\omega = \sum_{i_1 < \dots < i_r} a_{i_1 \dots i_r} dx_{i_1} \wedge \dots \wedge dx_{i_r}$ (r may vary), it follows that the algebra is a model of the algebra of differential forms for some n-manifold (see reference 7). This in turn implies, of course, that the differential $d(a) = \sum_{i=1}^n \frac{\partial a}{\partial x_i} dx_i$ of an expression a in R is an element of this algebra.

We now extend the definition of this differentiation mapping in the standard manner so as to have the value $d(w) = \sum_{i_1 < \dots < i_r} d(a_{i_1 \dots i_r})$

$\wedge dx_{i_1} \wedge \dots \wedge dx_{i_r}$ for any vector of the form $w = \sum_{i_1 < \dots < i_r} a_{i_1 \dots i_r} dx_{i_1} \wedge \dots \wedge dx_{i_r}$, in which case:

(1) $d(\alpha w_1 + \beta w_2) = \alpha d(w_1) + \beta d(w_2)$ for any vectors w_1 and w_2 and any elements α and β which are in the SIMPLIFY ring.

(2) $d(d(w)) = 0$ for any vector w of class c^r ($r \geq 2$).

(3) $d(w_1 \wedge w_2) = d(w_1) \wedge w_2 + (-1)^r w_1 \wedge d(w_2)$ for any vector w_1 of the form $w_1 = \sum_{i_1 < \dots < i_r} a_{i_1 \dots i_r} dx_{i_1} \wedge \dots \wedge dx_{i_r}$ (r fixed) and any arbitrary vector w_2 .

This algebra is an extremely important application of the MULTILINEAR ring R , allowing us not only to be able to transform many highly complex partial differential equations in a systematic, yet simple manner, but to transform and in many cases reduce multiple integrals also. Moreover, if the collection of algebraic homomorphisms of the form $f: A \rightarrow A$, where $f(M)$ M_i are associated with A , we then have a direct procedure for studying the collection of matrices associated with the corresponding restricted linear transformations $f: M \rightarrow M$.

On the Nature of Symbol Manipulation Languages

During the last few years a number of studies have been made on the structure required for higher-level languages designed primarily for an analyst desiring to manipulate various mathematical expressions in a variety of ways (e.g., See reference 3). From these studies it has become evident that the general list processing languages (such as LISP) are not appropriate for processing complex mathematical data. This is due not only to the extreme complexity of the coding that necessarily ensues in the analysis of an intricate problem, but also to the fact that the input-output procedures most ideally suited for these languages are totally inadequate for the individual primarily interested in the analysis of a problem and not the underlying mathematical and input-output structure involved. On the other hand, it has become equally clear that in order for such a symbol manipulation language to be constructed, a list processing language, or equivalently a set of list processing routines will necessarily have to be employed, along with a highly sophisticated input-output procedure for allowing the analyst to communicate with the machine and a mathematical structure which would be broad enough in scope to interpret a diverse collection of expressions in a fairly simple manner. We shall now consider briefly the underlying mathematical structure that any such language would need.

In general, any symbol manipulation language must always contain at least two modes of computation, namely the ring of rational numbers and a polynomial algebra over a set of variables using only numerical coefficients (in which case, we have just an application of the SIMPLIFY algebra, of course). However, because of the diversity of the mathematical concepts encountered and the techniques that are needed, these modes will obviously never suffice. On the other hand, since SIMPLIFY was found to be a very simple application of an extremely general model, namely ALGEBRA, and such a model in some form will always be needed for defining this algebra, it is evident that one can implement the model itself so that we then have at our disposal not only the SIMPLIFY algebra, but a host of other mathematical procedures also. This being the case, an experimental programming language called FLAP (for FORTRAN-like algebraic list processor) was developed, using the ALGEBRA system as its underlying structure.

The FLAP language is a higher-level language (written in LISP) which contains not only the model ALGEBRA but also a complete system of input-output routines for interpreting the data under consideration. The language contains many computational modes, including the standard fixed and floating point arithmetic and the SIMPLIFY, MULT, and MULTILINEAR algebras. Its format is similar to that of the FORTRAN system, in that it makes use of the same notation of the statements and formulas, the major distinction being the existence of a mathematical mode setting statement, appropriately called the MODE statement. This statement, which has for its argument the name of an algebra, in effect tells the interpreter that until another mode statement is made, every statement

is to be regarded as being dominated by this algebra. If however, no MODE statements are made, the standard fixed and floating point arithmetic will prevail throughout the program so that, in effect, we then have a FORTRAN-like program.

The language FLAP, though still in the experimental stage, has already been found to be extremely useful. This is due not only to the list-processing capability which it inherits from LISP, but also to the fact that it has been formulated on a general analytical structure (namely ALGEBRA) rather than a set of distinct computational procedures. Using the model we then have the capability of being able to easily define any other computational mode (by setting the arguments X, r, v, AXIOM, etc.) as well as being able to modify any existing ones.

EXAMPLE:

Suppose that we wished to write a FLAP program for computing the expression $\frac{\partial}{\partial x_i} [ay \cdot \sin(xy)]$, where y is a variable depending on the variable x and a is a constant. Since we would then be using the MULTILINEAR algebra with the generator set {y} and the scalar ring MULT (which would, in turn, have the generator set {x}), the program could be written as follows:

CALCULATE

INPUT A,X,Y

DUMMY K

MODE MULTILINEAR (X),(Y)

K = DIFF(A*Y*SIN(X*Y),X)

RETURN K

END

CALCULATE(A,X,Y)

The following value would then be printed:

$A * X * Y * \cos(X * Y) * \text{DIFF}(Y, X) + A * Y^2 * \cos(X * Y) + A * \sin(X * Y) * \text{DIFF}(Y, X)$

The three statement cards beginning with the MODE statement contain the instructions for computing and printing the expression; the other cards are used only for declaring and binding the variables concerned.

REFERENCES

1. Berkeley, E. C., etal. The Programming Language LISP: An Introduction and Appraisal. Computers and Automation, Vol. 13, No. 9. Sep, 1964, pp. 16-23.
2. Chevalley, Claude. Fundamental Concepts of Algebra. Academic Press, New York, 1956.
3. Engelman, C. MATHLAB: A Program for On-Line Machine Assistance in Symbolic Computations. AFIPS Conference Proceedings, 1965 Fall Joint Computer Conference, Vol. 27, Part 1, 1965, pp. 413-421.
4. Kleene, S. C. Introduction to Metamathematics. D. Van Nostrand Co., Princeton, N. J. 1952.
5. McCarthy, John. A Basis for a Mathematical Theory of Computation. Proceedings of the 1961 Western Joint Computer Conference, May 1965. pp. 225-237.
6. McCarthy, John, etal. LISP 1.5 Programmer's Manual. MIT Press, Cambridge, Mass. 1962.
7. Sternberg, Shlomo. Lectures on Differential Geometry. Prentice-Hall, Englewood Cliffs, N. J., 1964.

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)		2a. REPORT SECURITY CLASSIFICATION	
Naval Weapons Laboratory		UNCLASSIFIED	
		2b. GROUP	
3. REPORT TITLE			
MODELS FOR MATHEMATICAL SYSTEMS			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates)			
5. AUTHOR(S) (Last name, first name, initial)			
Morris, Alfred H.			
6. REPORT DATE		7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
30 March 1966		25	
8a. CONTRACT OR GRANT NO.		8a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO.		2032	
c.		8b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.			
10. AVAILABILITY/LIMITATION NOTICES			
Distribution of this document is unlimited.			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY	
13. ABSTRACT			
The purpose of this paper is to describe and illustrate a program, written for the IBM 7090, which defines a model for a finite collection of algebras for the computer. It is shown that the program contains a structure broad enough in scope to allow one to perform operations on such diverse mathematical concepts as differential equations, infinite series, and differential forms in a simple yet comprehensive manner, while also serving as a foundation upon which a variety of higher-level symbolic manipulation languages can be developed.			