AN ON-LINE SYMBOLIC MATHEMATICS SYSTEM USING HAND-PRINTED TWO-DIMENSIONAL NOTATION*

Frederick W. Blackwell

Robert H. Anderson

The RAND Corporation
Santa Monica, California

Summary

This paper describes a system that is being developed at The RAND Corporation for the on-line manipulation of symbolic mathematical expressions. The primary input consists of the user's expressions hand-printed on a RAND tablet in ordinary two-dimensional mathematical notation. The system recognizes the characters and interprets the whole expression from the spatial relationships present in accordance with a previously input syntax. The user at the console directs symbolic transformations upon his input expressions by instructing the computer to selectively attempt to apply various rules of mathematics; these rules have been previously entered into the system in the same manner as the expressions. A transformed expression resulting from the application of a rule or group of rules is displayed on the IBM 2250 graphic console. An experimental version of the system is in operation at the present time.

Introduction

Most programming languages employ linear notation for algebraic formulas not only because it is much easier to implement, but also because the relatively few distinct types of two-dimensional configurations which normally occur can be readily represented in linear form. Familiar examples are the use of the slash for division and the use of the double asterisk or some special character for exponentiation. While the programmer and non-programmer alike adapt to this linear notation, we assume that both would usually prefer a language in which it is possible to write in ordinary two-dimensional mathematical notation. Examples of how this can be done for typewriter-like devices are provided by the work of Klerer and May[1] and Wells[2], and for more general input devices by the work of Anderson[3] and Bernstein and Williams[4]. The latter systems are complicated by the fact that the user prints his symbols on a RAND tablet or similar device; the characters must be individually recognized, and then the whole formula must be properly interpreted from the spatial relationships present. The development of formula-manipulating languages, in which the data and results are often inherently two-dimensional, has created additional demand (intensified in an on-line environment) for explicit two-dimensional representations in communicating with the computer.

This paper describes a system that is being developed at The RAND Corporation for the on-line manipulation of symbolic mathematical expressions. The primary input is the user's expressions hand-printed on a RAND tablet, and the primary output consists of transformed expressions which appear on the IBM 2250 graphical display console. Both input and output are in two-dimensional mathematical notation. The person at the console directs the transformations upon his expressions. Thus the system provides a kind of "sophisticated scratchpad" for the user.

System Design

The goals of the system are several. First, it is hoped that it will be a useful mathematical tool for RAND scientists.

*The work described in this paper was supported under Air Force Project RAND and under contract to ARPA.

If it is to do this, it must have the virtues of availability, convenience and ease of use, and of course speed and accuracy. The system provides considerable opportunity for experimentation, and it is intended that this attitude be maintained even as the system becomes more refined. In particular, the design of the system makes it possible to put together complex sequences of relatively simple mathematical transformation rules for later automatic execution. The synthesis of efficient sequences for problem-solving in particular areas can, in part, be determined by experimentation at the console. Finally, the system provides additional experience and guidelines that are useful in understanding the nature and means of solving problems in interactive computer systems, especially those in which the information being manipulated is two-dimensional or graphic in nature.

At present, the system described herein runs on a 256K-byte IBM 360 model 40, to which is attached a RAND tablet, and IBM 2250, a typewriter, four disk files, and several 2260's. Interaction with the programs, which will be described in some detail later, is by means of function keys and 2250 "light buttons", as well as by direct input via the RAND tablet. Most of the system was written in PL/1, using the facilities of SGS (Simultaneous Graphics System); the latter is a RAND-developed time-sharing system which allows multiple users at 2260 terminals to create and update files at the same time that a normal background job is running.[5]

Engeli remarks that "the problem space of formula manipulation is quite large and of such diversity that even today there is relatively little overlap in the goals and achievements of different systems."[6] At the outset, then, it is useful to put our system into perspective by briefly comparing its intent with that of certain other systems. Basically, our system is one for manipulating mathematical expressions in accordance with rules which the user supplies. In this respect it is much akin to the spirit of Fenichel's FAMOUS system[7]. This contrasts somewhat with systems such as Engelman's MATHLAB[8] and Martin's Symbolic Mathematics Laboratory[9], in which there are built-in high-level imperatives like SOLVE. It is also quite different from a system such as FORMAC[10], in which the formula-manipulating capabilities appear as an extension of a general programming language. Our system does give the user the ability to program sequences of transformations at the console, somewhat like that of Blackwell's system[11] that was based on the Culler-Fried system[12]. While this may not result in particularly efficient procedures, it does allow a user a relatively convient method of building toward higher-level operations such as SOLVE.

Input and Recognition

After loading the symbolic mathematics system and pressing an appropriate function key at the 2250 console, the user can write his expressions on the RAND tablet. (An "expression" as used here may be either an equation or a replacement rule to be used in a simplification process.) As each character is printed (in any size and at any location on the tablet surface) it is sent to Groner's character recognizer [13]; if it is recognized as a valid character, its "ink track" on the CRT is replaced by a vector-drawn stylized character of the same size and at the same location. A scrub character may be used to remove one or more previously recognized characters. The order in which the characters are drawn is not significant.

During the character input phase, a light button labelled PARSE is displayed. When that button is hit with the tablet stylus, the character configuration currently displayed on the screen is syntactically analyzed; if it is a valid equation or replacement rule, a character string equivalent is produced. The two-dimensional syntactic analysis is performed by Anderson's recognition algorithm[3]. This analysis is entirely syntax-directed; that is, it is governed by a set of replacement rules which describes the valid two-dimensional character configurations which can be recognized and converted into character strings.

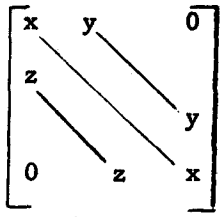
Upon completion of a successful syntactic analysis, the character configuration is enclosed by a displayed rectangle, the character string equivalent is placed beneath the rectangle, and two light buttons are available: EDIT and OK. The user may verify that the character string is a correct interpretation of his hand-printed expression. He may modify the expression by touching EDIT; control is returned to the character input phase, allowing him to scrub and rewrite parts of the existing two-dimensional expression. If the OK button is hit, the character string meaning of the expression is transmitted as input to the system. The system then displays a message asking the user if he wishes to name the expression; the user can reply by writing any alphanumeric name on the tablet. It is usually desirable to name an expression for later reference, unless it is one which is to be used immediately. Figure 1 contains four sample displays generated during the operation of the input phase of the symbolic mathematics system.

As mentioned above, the recognition of two-dimensional character configurations is entirely syntax-directed. The recognition system is therefore extremely flexible; individual users can modify and extend the mathematical notation which the system can recognize merely by changing the syntax which governs the recognition. Although not done at present, we plan to allow the user to make these syntax modifications on-line.

The flexibility of the parser will allow the system to be expanded in several ways by a user:

(1) he may add new symbols denoting operators (or operands), such as defining the symbol $\otimes$ to mean matrix multiplication;

(2) he may define new configurations of symbols, such as parsing the configuration



into the string

TRIDIAG(x, y, z, 0).

That is, the interpretation would be a functional notation describing a tridiagonal matrix (of unspecified size) with upper, center, and lower diagonals filled with x, y, and z respectively (where x, y, z could be arbitrary arithmetic expressions) and zeros filled elsewhere. Syntaxes have, in fact, been written and tested for the parser which handle most matrix notation, and even arbitrary directed line graphs. Therefore, it is possible for the user to print a wide variety of mathematically interesting constructions, to have them parsed into a character string, and to perform groups of manipulations on the resulting string. The symbolic manipulation of matrices and graph structures raises many questions; for example, how are intermediate results shown? How are forms too large to be displayed at once shown? The authors do not at present know the best answers to these questions, but there is sufficient flexibility in the specification of the syntax of input configurations to allow eventual experimentation with symbolic manipulation of these "very two-dimensional" structures.

Transformation and display

The basic expression manipulation capability of the system is that of searching a given expression and attempting to match all or part of it with the left-hand side of a given rule, where both the expression and rule have been input as previously described. A variable in a rule may match an entire subexpression. If the match succeeds, a replacement is made in accordance with the right-hand side of the rule. Substitution of one expression for another is done similarly. Expressions may also be transformed directly through on-line editing operations of insertion, deletion, and rearrangement. While the operations of substitution, editing and rule application are conceptually different, from a user's point of

view they are simply means to the same end: transforming mathematical expressions in meaningful ways.

A simple example that illustrates the basic types of on-line transformations will probably best convey the flavor of the system. Figure 2 shows the sequence of expressions which is generated on the 2250 display graphic console as various operations are performed on a given expression. Previous expressions are not automatically erased, so at the end of the sequence all expressions of Figure 2 (but without identification numbers) will appear on the 2250.

The first expression (1) is entered directly, or is the result of the previous computation, or is called up by name from the expression storage space. By writing expression (2) on the tablet and pressing the "substitute" function key, expression (3) results. (If (2) had been previously stored, the user could call it by name instead of rewriting it.) The parentheses in expression (1) are superfluous after the substitution, and are automatically removed in (3). Next, if the user wishes to differentiate the expression with respect to t, he has the computer make a copy of the expression and then he simply writes the indicated differentiation on the tablet in the usual manner; this is an example of the editing operation of insertion. The parser recognizes the new expression, and the stylized version (4) appears. By indicating a name (either via typewriter or by writing at the bottom of the tablet) and pressing the "apply" function key, the user calls forth a rule expressing the distributivity of differentation over addition. The result is expression (5), to which is then applied the rule given in Figure 1 (d) and the rule that the derivative of a constant is zero. By pressing another function key, the user instructs the computer to reduce the intermediate form (6) to the final expression (7). This latter simplification is the result of the straightforward application of a sequence of elementary rules of algebra, and occurs only if desired by the user. In fact, the user has the ability to synthesize such sequences of rules, and can therefore maintain as much control over the computer transformations as he desires. The only simplifications which the system automatically performs are arithmetic operations, redundant parenthesis removal, and the placement of a numerical coefficient first in a term. Thus, a term 2(A)3 which might arise in a computation would immediately be converted to 6A. Expressions or parts of expressions are not reordered by the system, with the exception just noted. However, in attempting to apply a rule involving a commutative operator, the system will commute the appropriate operands if this permits the rule to be applied when it would not otherwise apply. The user has the ability to input his own operators and to declare certain of their properties, such as commutativity.

The aggregation of transformation rules into a sequence for later application by just one reference to the set is accomplished by pressing a "group" function key, and then simply naming the rules that constitute the sequence; the group itself is then given a name. It is also possible to read in groups of rules from cards. A given rule may be a member of more than one group, and groups may be nested. At any time the user may ask for a display of rules or group of rules. In addition, he may have a rule or its name displayed as it is applied.

When the user calls for the application of a group of rules, the computer applies these rules cyclically until no further changes occur. Experimentation with the order and structure of rule application, such as that which has been done by Fenichel[7], will permit us to define a broader capability for automatic rule application in groups.

The interpretation of an expression in the computer is left to the user. For example, the equation

$$x^2 - y^2 = (x + y)\ (x - y)$$

may be the current expression being transformed. It may also represent a substitution to be made whenever the particular symbols ($x^2 - y^2$) appear in an expression. Finally, it may represent a general transformation rule, in which x and y themselves stand for expressions. Note that an expression may be numerically evaluated (or its number of variables reduced) by substituting numbers for variables.

The user has control over what he wants displayed; in particular, he can call for display after each transformation, as in the example, or omit the display of intermediate results such as expressions (5) and (6) in Figure 2. He may move forward and backward through his list of expressions, and so his previous work is available to him as long as there is memory space. Named expressions are always saved, but the oldest unnamed ones may be deleted if the space is needed for current expressions. Of course, the user may delete an expression or rule at any time, in which case its storage space is released.

The display of large expressions which cannot be entirely contained on the face of the 2250 is presently handled by having the system write the expression on successive "pages", which are then displayed under user control. Splitting the expression logically (as determined from the Polish form) and varying the character size of parts of the expression ("zooming") are two techniques being considered to further aid in the display of large expressions. An expression could also be displayed in tree form, as in Millen's CHARYBDIS [14]. We have yet to develop a full two-dimensional display capability such as that of Martin[9], but the subscripting, superscripting, and spacing present in our displayed expressions make most of them quite readable. In displays, the user may choose whether or not he wishes to have simple juxtaposition denote multiplication. An explicit multiplication sign between factors is necessary when multi-character names are used; such names must be declared by the user, as well as all names that represent symbolic constants.

Internal structure

This paper is mainly concerned with the system description from a user's point of view, but certain internal features may be of general interest. Expressions are stored in one-way list structures that correspond to a modified delimiter Polish prefix notation. Internally, a binary minus becomes a plus followed by a unary minus; a similar statement holds for the division operation. This kind of representation permits the retention of symmetry which may be present in an expression. As an example of this notation, the external expression a-b+c+d becomes the list structure equivalent of the delimiter Polish form +a-bcd), where the right parenthesis denotes the scope of the "continuing" operator plus. No attempt is made to retain an expression in its external two-dimensional form; the relative ease of manipulation of Polish notation seems to justify the conversion required whenever input/output operations occur. Displayed expressions converted from internal Polish may look slightly different from their original input form, but (as noted before) reordering is not done.

The system uses floating-point arithmetic for number storage and computation, although an integer or a number which differs from an integer by a prescribed tolerance is displayed as an integer with no decimal point. We plan to incorporate into the system a rational arithmetic using the "decimal arithmetic" feature of System/360.

The programming of the graphics parts of the system was facilitated by the use of the RAND Integrated Graphics System (IGS)[15], a set of subroutines which permit easy on-line communication between the computer, the 2250, and the RAND tablet.

System status

The implementation of almost all of the features described in this paper is complete, and useful system experimentation and resultant modification have begun. In addition, the following improvements and extensions are contemplated: complete two-dimensional output and editing; a better structure for applying rules conditionally so that very complex operators can be built up; a "designation" facility that makes it easier to operate on parts of expressions, including greater use of the tablet stylus for this purpose; and provision for efficiently transferring expressions between core and disk. The system may also be adapted to handle more general algebras. Of particular importance

is the development of provision for smooth dialogue between the user and the system. A possible way of achieving this goal might be an increased use of light buttons and menus and a corresponding decrease in reliance on function keys and the typewriter. The system will be used to test various alternative data input and display techniques, so that the most natural man-machine dialogue might be achieved.

It is intended that this symbolic mathematics system will be implemented on an experimental Videographic system currently being developed at RAND in conjunction with IBM. This system uses television monitors for inexpensive graphic display, with provision for tablet and keyboard input devices at a terminal. The set of subroutines (IGS) currently being used for communication with the IBM 2250 graphic terminal will also provide the interface to the Videographic system, so minimal program conversion will be necessary.

References

1. Klerer, M. and F. Grossman, "Further Advances in Two-Dimensional Input-Output by Typewriter Terminals," Proceedings of the Fall Joint Computer Conference, Vol. 31, Part I, pp. 675-687. Thompson Books, Washington, D.C., 1967.

2. Wells, M. B., "MADCAP: A Scientific Compiler for a Displayed Formula Textbook Language," Communications of the ACM, Vol. 4, pp. 31-36, 1961.

3. Anderson, R. H., "Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics," Interactive Systems for Experimental Applied Mathematics, (Klerer, M. and J. Reinfelds, editors), pp. 436-459. Academic Press, New York, 1968.

4. Bernstein, M. I. and T. G. Williams, "A Two-Dimensional Programming System," Proceedings of IFIP Congress 68. North-Holland Publishing Co., Amsterdam.

5. Balzer, R. and W. Josephs, SGS -- RAND Simultaneous Graphics System, RM-5612-ARPA, The RAND Corporation, Santa Monica, California, February 1969.

6. Engeli, M. E., "Achievements and Problems in Formula Manipulation," Proceedings of IFIP Congress 68. North-Holland Publishing Co., Amsterdam.

7. Fenichel, R. R., An On-Line System for Algebraic Manipulation, Ph.D. dissertation, Division of Engineering and Applied Physics, Harvard University, Cambridge, Massachusetts, July 1966.

8. Engelman, C., "MATHLAB 68," Proceedings of IFIP Congress 68. North-Holland Publishing Co., Amsterdam.

9. Martin, W. A., Symbolic Mathematical Laboratory, Ph.D. dissertation, Dept. of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, January 1967. Also M.I.T. Project MAC report no. MAC-TR-36.

10. Sammet, J. E. and E. R. Bond, "Introduction to FORMAC," IEEE Transactions on Electronic Computers, Vol. EC-13, pp. 386-394, 1964.

11. Blackwell, F. W., "An On-Line Symbol Manipulation System," Proceedings of the 22nd National ACM Conference, pp. 203-209. Thompson Book Co., Washington, D.C., 1967.

12. Culler, G. J. and B. D. Fried, "The TRW Two-Station On-Line Scientific Computer," Computer Augmentation of Human Reasoning, (Sass, M. A. and W. D. Wilkinson, editors), pp. 65-87. Spartan Books, Washington, D.C., 1965.

13. Groner, G. F. "Real-Time Recognition of Handprinted Text," Proceedings of the Fall Joint Computer Conference, Vol. 29, pp. 591-601. Spartan Books, Washington, D.C., 1966.

14. Millen, J. K., "CHARYBDIS: A LIST Program to Display Mathematical Expressions on Typewriter-like

Devices," Interactive Systems for Experimental Applied Mathematics (Klerer, M. and J. Reinfelds, editors), pp. 155-163. Academic Press, New York, 1968.

15. Brown, G. D. and C. H. Bush, The Integrated Graphics System for the IBM 2250, RM-5531-ARPA, The RAND Corporation, Santa Monica, California, October 1968. Also CFSTI report no. AD 677 464.