NASA CONTRACTOR REPORT

NASA CR-14

c.1

NASA CR-1455

ON-LINE PARSING OF HAND-PRINTED MATHEMATICAL EXPRESSIONS: FINAL REPORT FOR PHASE II

by Thomas G. Williams

Prepared by
SYSTEM DEVELOPMENT CORPORATION
Santa Monica, Calif.
for Electronics Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • DECEMBER 1969

NASA CR-1455

ON-LINE PARSING OF HAND-PRINTED MATHEMATICAL EXPRESSIONS: FINAL REPORT FOR PHASE II

By Thomas G. Williams

Distribution of this report is provided in the interest of information exchange. Responsibility for the contents resides in the author or organization that prepared it.

Issued by Originator as Report No. TM-4158/000/00

Prepared under Contract Nos. NAS 12-526 and DAHC15-67-C-0149 by
SYSTEM DEVELOPMENT CORPORATION
Santa Monica, Calif.

for Electronics Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – Price $3.00

TABLE OF CONTENTS

1. INTRODUCTION

This document describes a program which analyzes free-hand two-dimensional mathematical expressions. Such expressions can be generated with the hand-printed character recognition program* currently in operation at SDC. The user interacts with the character recognizer to enter characters of any size placed anywhere on the input surface. His hand-printed characters, as recognized, are replaced in a standard form at the same place, same size, same aspect ratio by the character recognizer. The output of the analyzer is also displayed at the top of the writing surface so that the user knows the current interpretation of his input. Figure 1 shows the system in operation.

The analysis or parsing program is a two-phase system. The first phase converts the two-dimensional structure into a linear string. This phase operates concurrently with input and editing, so that the user can always be given the current interpretation of his input. The string also provides editing information. The second phase operates on the linear string in the same manner as a conventional compiler or interpreter. It is dependent upon the ultimate objective of the input, and thus will not be discussed here.

There are three advantages to this two-phase system. First, the first phase can operate correctly on incomplete expressions. The user feedback mentioned above can only be given if incomplete expressions can be analyzed. Second, as mentioned above, the information developed can be used in editing, whether or not the expression is complete. This allows, for example, a group of characters--such as an integral sign with its associated limits of integration--to participate in editing operations without explicit designation by the user. Third, the context-dependencies of mathematical notation can be handled by the second phase, where syntax- or table-driven systems make this easy.

2. PROGRAM STRATEGY

The first phase of the parser analyzes "local syntax," that is, relationships between nearby characters. The assumption is made that these relationships can be analyzed by comparing successive pairs of characters. Thus, for example, rules are formulated as to what constitutes a subscript or superscript, and positions of characters are tested against these rules. The use of such local syntax allows incomplete expressions to be processed.

The current program reprocesses all input characters in the expression after each new set of characters is received from the character recognizer. The characters are kept in a list that is sorted in left-to-right order. This list is used to generate the linear string. In earlier versions, attempts were made to edit the new characters into the existing linear string without rescanning all characters. Thus, the routines used were (and for the most part still are) relatively insensitive to variations in character order. This provides protection in cases where editing operations alter the position of

*The character recognizer is described in TM-3937/000/00.

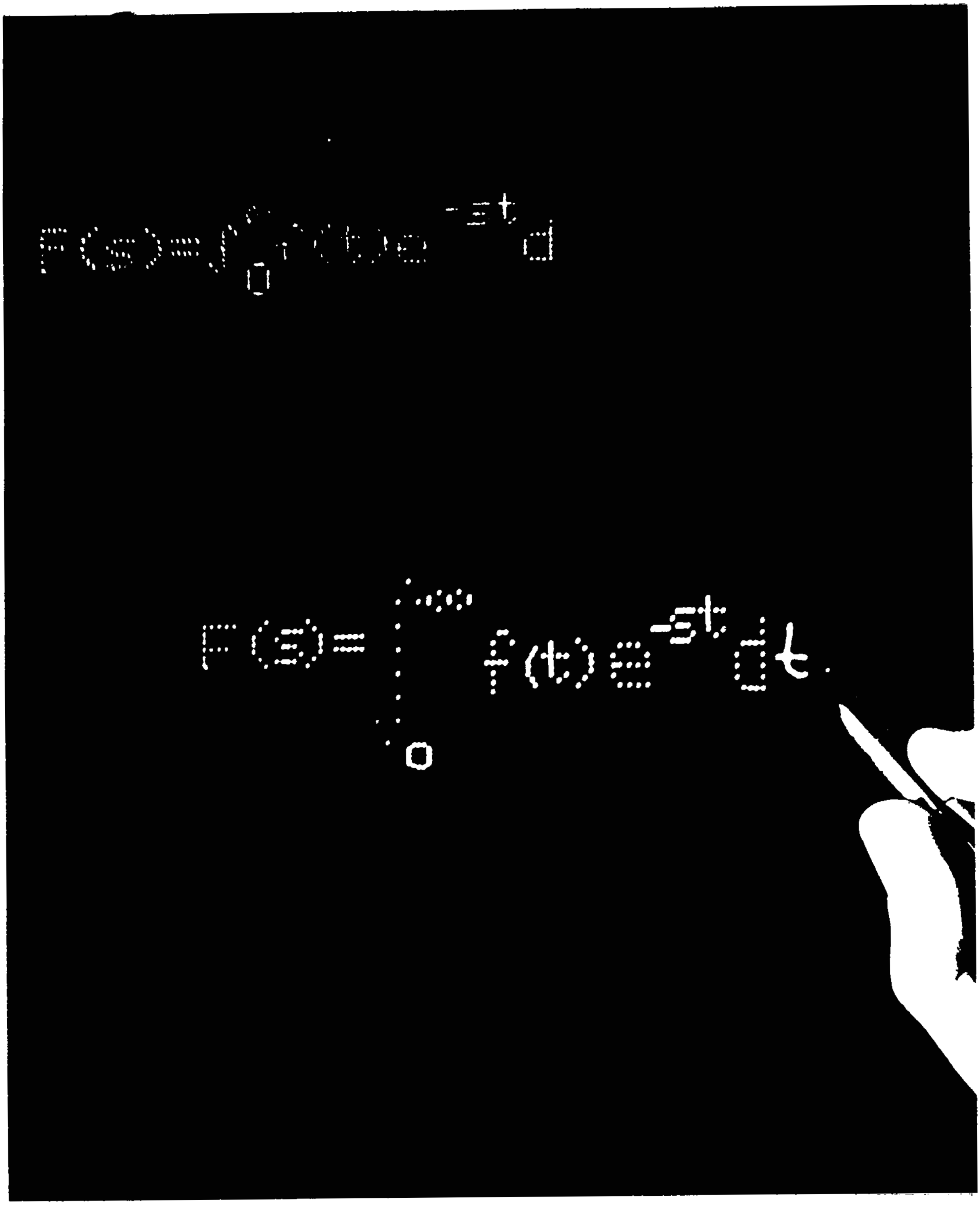

Figure 1. On-Lime Parsing System in Operation

characters. The current approach solves many of the problems that arose in making the analysis completely insensitive to character order.

Under time-sharing, the user does not notice the extra time required to process the entire string of characters. In the present system, representative computation times for the necessary operations are:

10-15 msec per stroke to recognize a character

2 msec per character for the parser

2 msec per character to generate the output messages

Thus, adding a new character to a string which currently has 40 characters would require less than 200 msec. This is insignificant compared to the wait in the time-sharing queue, which is on the order of seconds.

The rest of this document describes in detail the operation of the program. It is divided into sections on the data structures, the program structure, and the editing operations.

3. DATA DESCRIPTIONS

The basic data element in the program is called the CSP block.* A CSP block is generated for each user-supplied character and contains both position and syntax information. The following information is in the CSP (all information is generated by the parser unless otherwise noted):

Character	The character code; supplied by the character recognizer.
X_0,Y_0	The coordinates of the lower left corner of the minimum rectangle surrounding the character; provided by the character recognizer.
DX,DY	The length of the sides of the minimum rectangle; provided by the character recognizer.
X_c,Y_c	The position of the center of the character. This is a weighted center: In X, it is the geometric center; in Y, it may differ from the geometric center as specified in a property string maintained for each character. This is to allow proper analysis of characters like "y" and "g" which extend below the character line.
Display Buffer Location	A pointer to the location of the character in the display buffer; this is generated by the character output routine.

*The name comes from an earlier program; the meaning of the letters has been lost in antiquity.

SORTLIST Link — A pointer to the next CSP on the SORTLIST, which is the list of CSP's sorted into left-to-right order; generated by the sort routine.

Function — The function assigned to the character; functions are mainline (on the same line as another character), subscript, superscript, and unrelated.

Related Character — The character to which this character is related by the function; the leftmost character in a string or substring has no related character, nor does an unrelated character.

Related Fraction Bar — If the character is in the numerator or denominator of a fraction, the CSP location of that bar is stored here.

Mainline Character Forward — If another character follows this one on the same line, its CSP location is stored here.

STRING Pointer — The pointer to the block in the STRING list which contains this CSP.

Expression Begin — The pointer to the block in the STRING list which contains the beginning of the expression for this character. The expression for an ordinary character contains its subscripts and superscripts; for a fraction bar, its numerator and denominator.

Expression End — The pointer to the block in the STRING list which contains the end of the expression for this character.

CSP's are contained on three lists, described in the following sections.

3.1 STRING LIST

The STRING list is the linear output produced by the parser. It is a two-way list structure, which permits scanning the list in both directions. Two types of cells are contained on this list: (1) external cells, which contain CSP addresses and are used for characters input by the user, and (2) internal cells, which do not contain CSP addresses and are used for characters which must be inserted into the list internally to represent the meaning of the two-dimensional structure. Figure 2 shows the structure for the sub-expression (X) where the X has been supplied by the user and the ( ) are added by the parser.

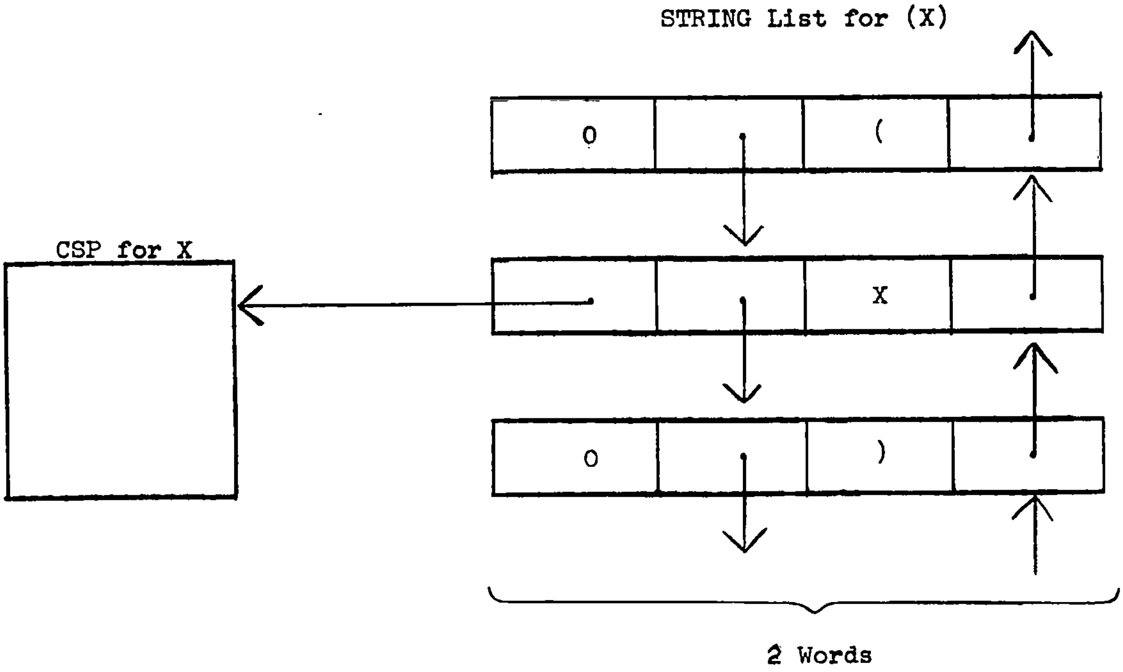

Figure 2. STRING List Structure for a Subexpression (X)

The notation used in the STRING is similar to--but not identical to--the notation used in algebraic compilers such as FORTRAN. It is best given by example.

Two-dimensional
sub- and super-script notation:

$a_i^2 + b_i^{23}$

STRING equivalent:

a[↓i↑2] + b[↓i↑23]

Two-dimensional
fractions:

$$\frac{a^2 + b^2}{c^2}$$

STRING equivalent:

(a[↑2] + b[↑2])/(c[↑2])

3.2 FRACT LIST

The FRACT list is a list of the CSP's for all horizontal bars. It contains CSP's for both fraction bars and minus signs, because a bar is not known to be a minus sign until all characters have been processed. The order on this list is longest bar first.

3.3 SORTLIST LIST

The SORTLIST list contains all CSP's, sorted into left-to-right order.

4. PARSER DESCRIPTION

The parser receives groups of characters from the user through the character recognizer. The following operations are done on each character of a group before entry into the parsing program proper:

1. A check is made to see if the new character overwrites an old character. (See Section 5 on editing for a description of this process.)

2. A CSP block is constructed and set with the information available from the character recognizer.

3. The character is placed in the display buffer for output.

4. The CSP is sorted into the SORTLIST according to the value of X_0.

The parser operates from the SORTLIST. At any time, each character is processed with reference to the characters that have already been processed and placed on the STRING list. The character being processed is called the input character; any previously processed character, a reference character.

The analysis of an input character is in two phases: the first phase finds a substring of the characters on the STRING list that the input character can be related to; the second phase checks for various relations that can exist between the input character and a reference character from the substring.

The relationships that can be found between an input character and a reference character are: subscript, superscript, mainline, or unrelated. Figure 3 shows the positional information used to check such relationships.

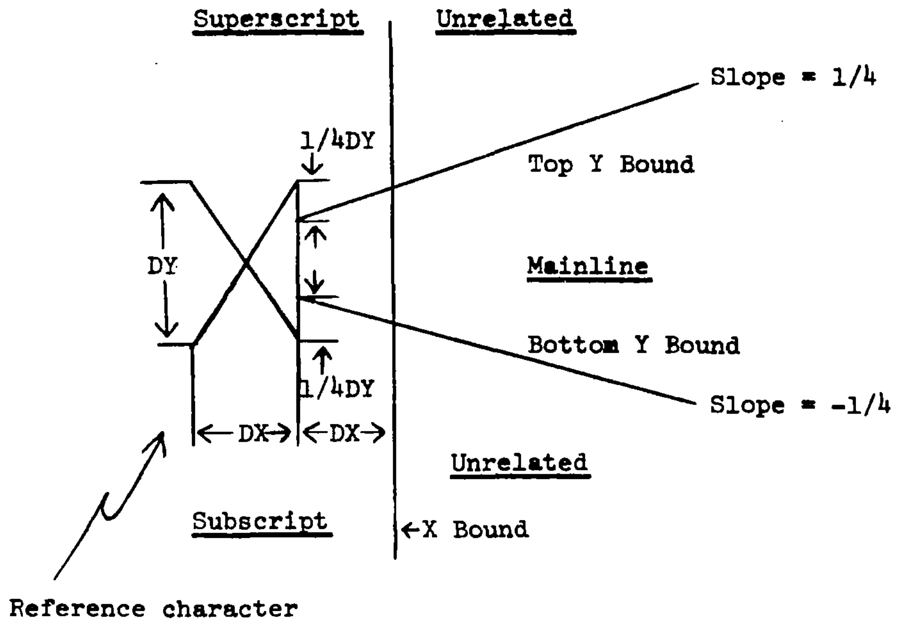

Figure 3. Derivation of Positional Relationship Information

To be considered at all, the left edge of the input character must be to the right of the left edge of the reference character. To be a superscript, the input character must be shorter than the reference character, the center of the input character must be above the top Y bound, and the left edge of the input character must be to the left of the X bound. To be a subscript, the input character must be shorter than the reference character, the center of the input character must be below the bottom Y bound, and the left edge of the input character must be to the left of the X bound. Any input character whose center falls between the top and bottom Y bounds is a mainline character. Any input character not meeting any of the above conditions is unrelated.

The processing performed during the first phase of the parser depends on whether the symbol is a horizontal bar or not. If the input character is not a horizontal bar, a search is made for any horizontal bar already in the string that the input character might be above or below. An input character is related to a horizontal bar if the center of the input character lies within the X extent of the bar. This check is made with the FRACT list. If the input character is related to a horizontal bar, the bar is changed from a minus to a fraction bar if necessary. Then it is determined whether the input character belongs in the numerator or denominator of the bar, by comparing the center of the input character and the center of the bar. The characters already in the appropriate part of the bar are used as the first substring of characters that the input character can be related to. If this subset is not empty, any horizontal bars in the substring are checked to see if the current character lies within their extent. If so, the process is repeated to find a new subset. Eventually, a character substring (possible empty) is found which contains the characters associated with the same fraction bar as the input character. If no fraction bars have been found, the subset is the entire string. The related fraction bar entry in the CSP for the input character is assigned at this point.

Figure 4 shows some two-dimensional structures, input characters, and the subsets of characters that are used. The box (□) indicates the input character.

Expression	Substring Found
$a + b + \boxed{c}$	$a + b +$
$a + \frac{b}{\boxed{c}}$	empty
$a + \frac{b}{a + \frac{b}{d + \frac{b}{e + \boxed{b}}}}$	$e +$
$a + \frac{b + \frac{c}{d} + \boxed{e}}{f}$	$b + \frac{c}{d} +$

Figure 4. Sample Input Structures and Character Subsets

If the substring is empty, the input character is assigned as a mainline character to the numerator or denominator of the fraction bar, as appropriate, and the analysis is complete. Otherwise, the nearest of the characters to the input character whose left edge is to the left of the left edge of the input character is chosen as the first reference character. The two characters are compared. If a relation is found, the function and related character entries in the CSP are set accordingly, and the analysis is finished. If not, the character before the current reference character in the substring is selected as the next reference character. This process continues until either a relation is found or the beginning of the substring is reached. If the beginning of the substring is reached, the input character is unrelated and the CSP is so marked.

If the input character is a horizontal bar, a check is made to see if the bar is a fraction bar, or is in a fraction. This is determined by the presence of characters whose centers are either above or below the input bar and are contained within the X entent of the bar. If no such characters are found, the bar is assumed to be a minus sign, and it is treated as an ordinary character (as above). It is still placed on the FRACT list, as characters encountered later may convert it into a fraction bar.

If the bar is determined to be a fraction bar, a process similar to that used for ordinary characters is entered to find the proper substring of characters in which the input bar should be placed. There are three possible results.

1. The substring is empty. The bar is then a minus sign after all, and is treated as an ordinary character.

2. All of the characters in the substring are in the numerator or the denominator of the new bar. In this case, the characters are copied into the appropriate position in the expression for the input bar, and the function and related character entries of the first character in the subset become the function and related character entries in the CSP of the input bar. The function and related character entries of the first character of the subset are changed to reflect the new situation.

3. The fraction bar may have some (but not all) of the characters in the substring in its numerator or denominator. In this case, the characters in the numerator or denominator are deleted from the substring and placed into an expression containing the input bar. Then the fraction bar is analyzed with the characters left over in the substring.

The placement of an input character into the output STRING list is done from the entries set into the CSP by the analysis process. It is a simple matter of locating the input character with respect to its related character and adding any internal characters necessary to specify the relationship. If the character is unrelated, it is sorted into the STRING list by its position in the two-dimensional expression. This method often results in the correct placement of a character even though it could not be analyzed.

5. EDITING OPERATIONS

There are two classes of editing operations: overwrite and move. The overwrite class consists of the two operations scrub and character overwrite. These are done in the character input mode--the normal mode of operation. The move class consists of horizontal open-up, horizontal close-up, move expression, and move group operations. These operations must be done in the special move mode, which is entered from a light button.

5.1 OVERWRITE OPERATIONS

Overwrite and scrub are used to delete and change characters. Scrub is an input that is analyzed by the character recognizer as having too many features. When this is detected, all characters whose centers are within the rectangle surrounding the scrub character are deleted. The expressions associated with these characters are also deleted. This means that if a character is deleted, its sub- and super-scripts are deleted also; if a fraction bar is deleted, the numerator and denominator are deleted also, as shown in Figure 5.

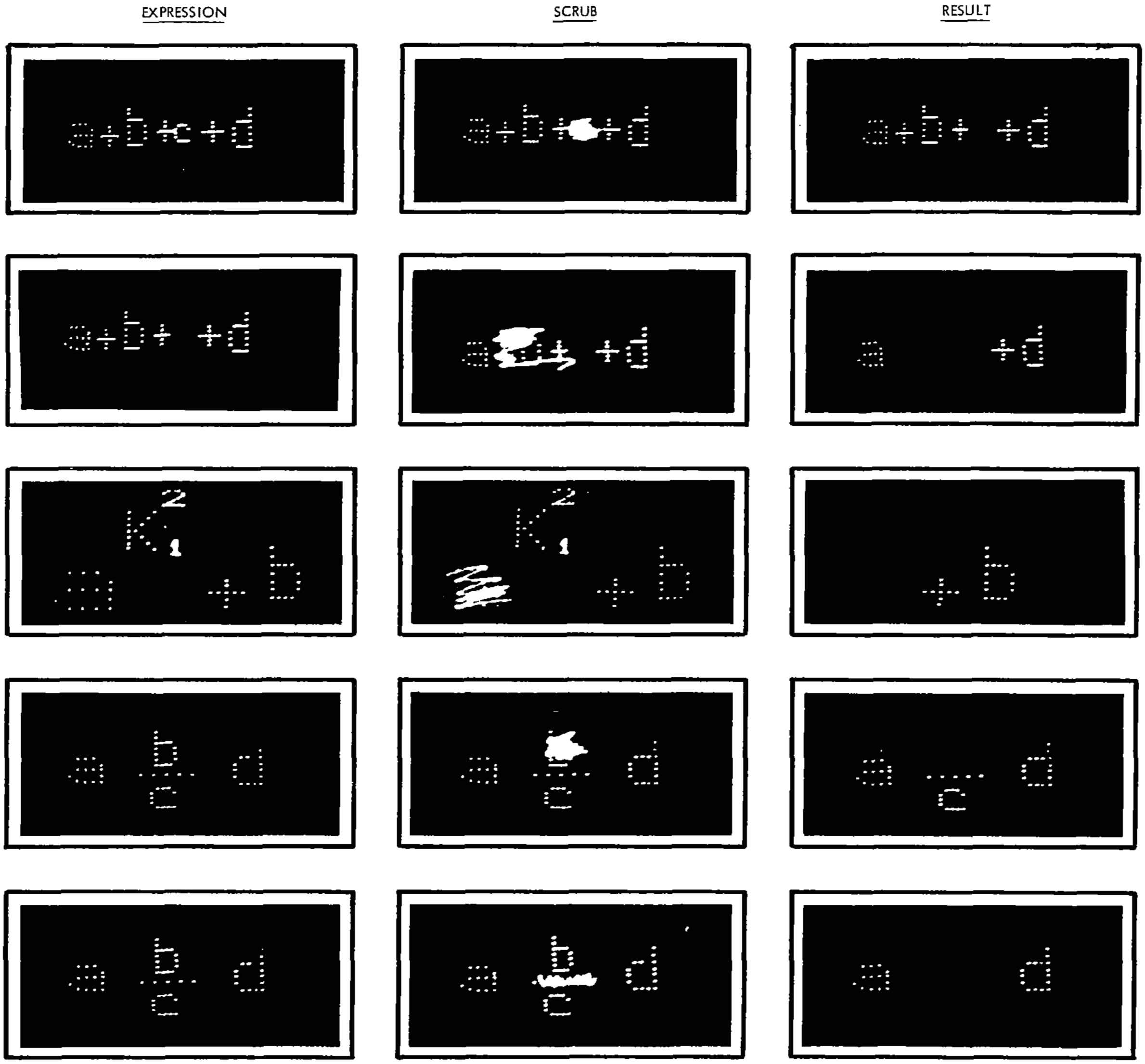

FIGURE 5. SCRUB OPERATIONS

An overwrite is simply a new character written over an old character. For each new character input, all old characters whose centers lie within the rectangle surrounding the new character are deleted. The expressions of these characters are not deleted. This allows characters to be replaced as shown in Figure 6.

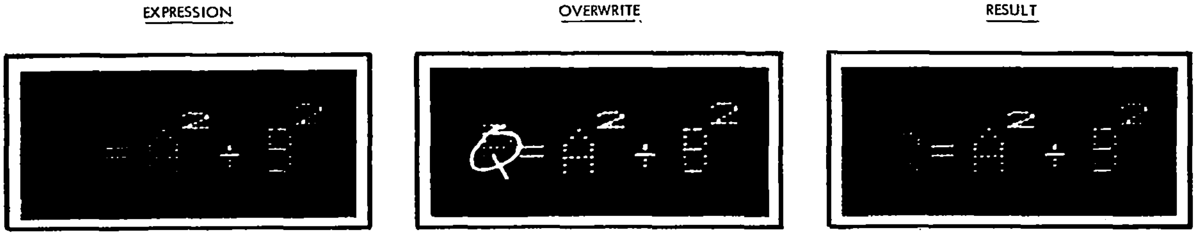

FIGURE 6. OVERWRITE OPERATION

5.2 MOVE OPERATIONS

The move operations are used to move characters around in the user's two-dimensional expression. They are indicated by special characters which are recognized in the move mode. These characters are:

Operation	Character
Horizontal Open-up	Any single stroke beginning in the rectangle of an existing character.
Horizontal Close-up	Two strokes; the beginning and ending points of each stroke must determine a horizontal line.
Move Expression	A period within an existing character followed by any second stroke.
Move Group	A horizontal line followed by a second stroke. The line between the beginning of the first stroke and the beginning of the second stroke must be horizontal.

When arbitrary strokes are allowed, only the beginning and ending points are involved and these are used to determine the increments in X and Y that are used to move the characters.

These four move operations are described in more detail in the following sections.

5.2.1 Horizontal Open-up

The purpose of the horizontal open-up operation is to create a space in the two-dimensional structure. An X increment is calculated from the difference between the end points (from beginning to end) of the stroke used to signify this operation.

The character whose rectangle contains the beginning of the stroke is then found. If the X increment is positive (move right), the specified character and all characters to its right are moved by the X increment. If the X increment is negative, the specified character and all characters to its left are moved left by the X increment. Any character moved off the screen is deleted and cannot be recovered. If the specified character is in a fraction, only characters in the fraction are moved. The Figure 7 illustrates this function.

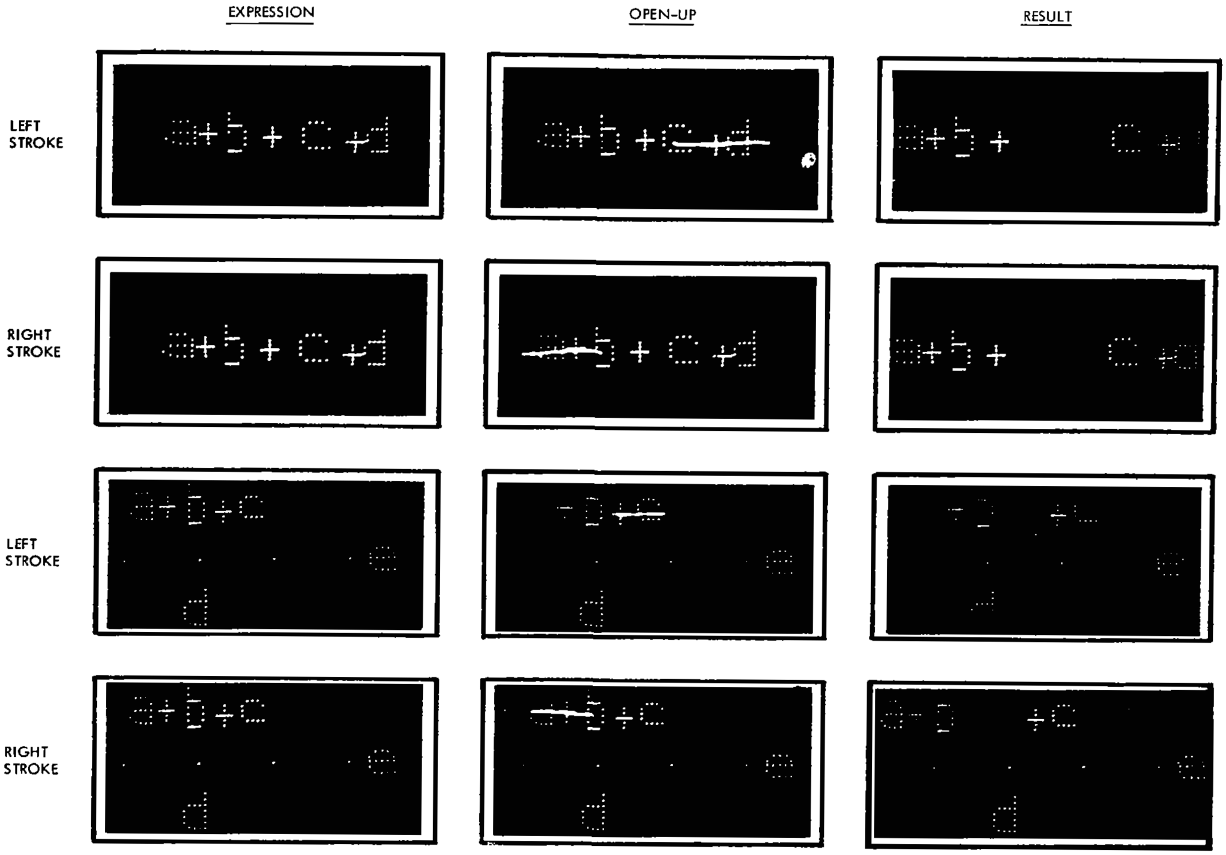

FIGURE 7. HORIZONTAL OPEN-UP OPERATIONS

5.2.2 Horizontal Close-up

This operation is the inverse of open-up. All characters to the right of the close-up symbol are moved left by an amount determined by the average length of the strokes. The same limits on fractions and screen limits apply in close-up. Figure 8 shows some examples.

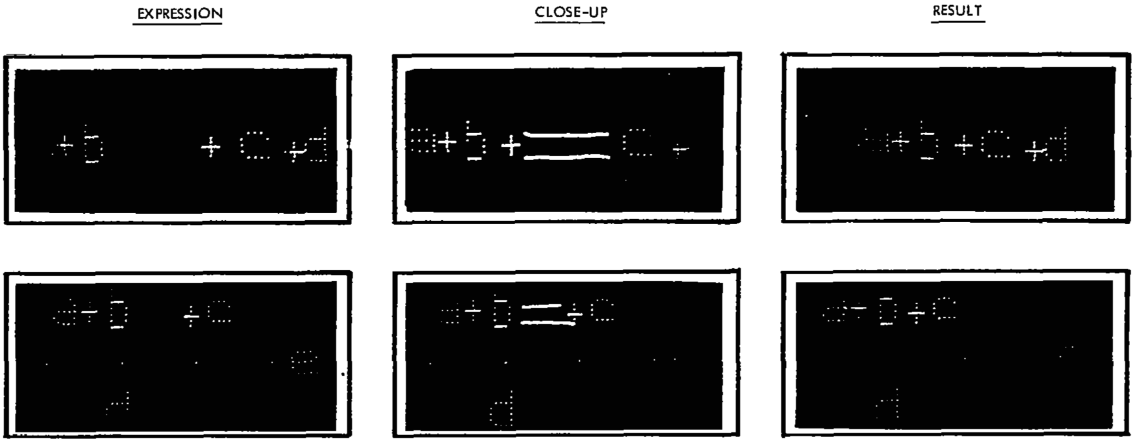

FIGURE 8. HORIZONTAL CLOSE-UP OPERATIONS

5.2.3 Move Expression

This operation allows an expression to be moved arbitrarily in X and Y. The character is specified by the dot, the amount to be moved by the beginning and end of the second stroke. The specified character and its expression are moved; nothing else is changed. As before, any character moved off the screen is deleted. Figure 9 shows an example of this.

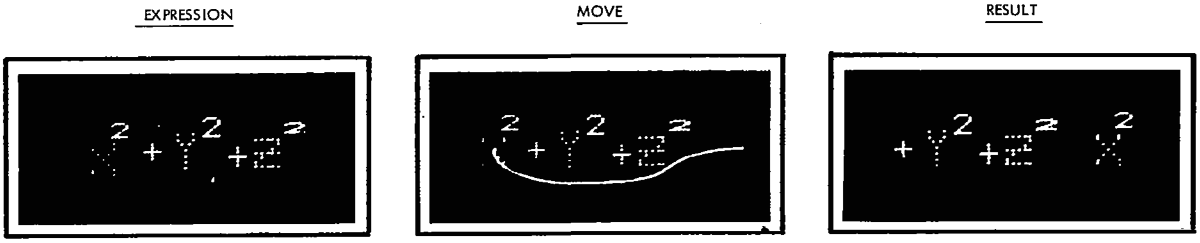

FIGURE 9. MOVE EXPRESSION OPERATION

5.2.4 Move Group

This operation allows a group of characters to be moved. The horizontal line delimits the characters to be moved; the second stroke determines the increments for the move. If the group is to be moved up (Y increment is positive) then all characters above the horizontal line are moved up; if down, all characters below the line are moved down. Motion in the X direction is also done as specified by the second stroke. No other characters are changed. Any characters moved off the screen are deleted. See Figure 10.

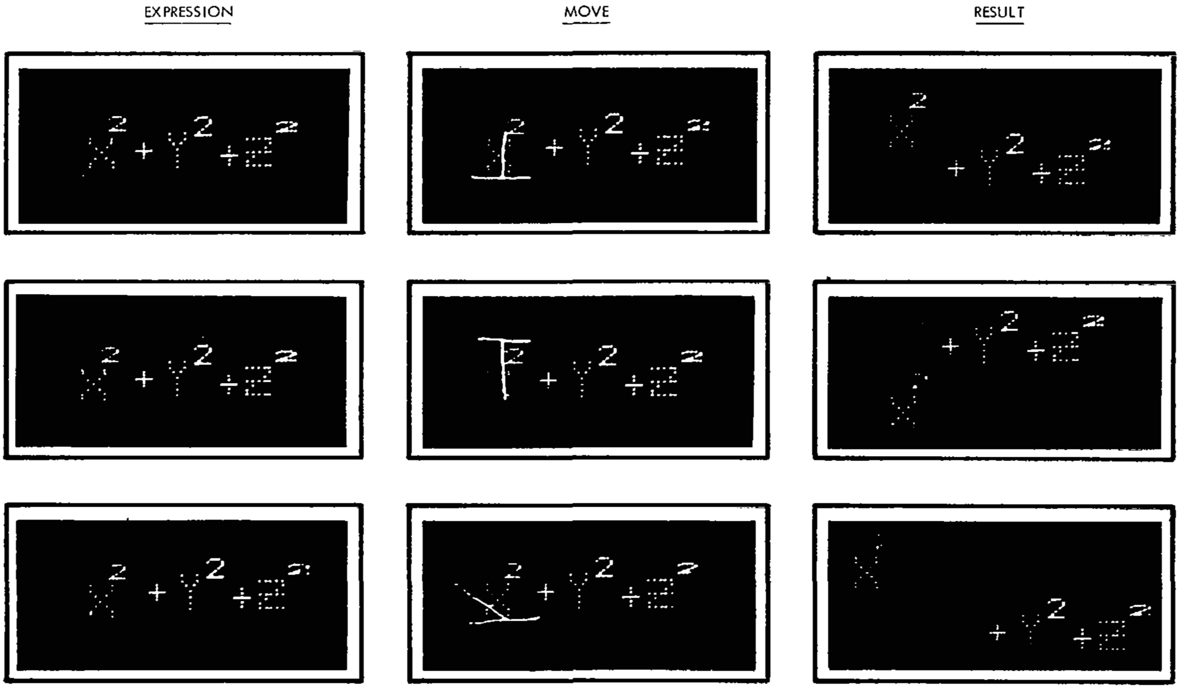

FIGURE 10. MOVE GROUP OPERATIONS

6. CONCLUSION

The notation accepted by this version of the parser is only a subset of the notation used in mathematics. In particular, expressions such as $\dot{X}$, $\bar{X}$, ${}^{d}_{c}X^{a}_{b}$ and $\binom{a}{b}$ cannot be analyzed.

However, the range of notation accepted is sufficiently broad so that useful work can be done without resorting to linear equivalents as input. Future work will include removal of these limitations.

This program demonstrates the usefulness of allowing ordinary hand-printed mathematical notation to be used as a computer language. The most important conclusion to be drawn at this time is that the techniques for utilizing common two-dimensional notations should be extended to encompass all commonly used notations in all fields of science and engineering.

NASA-Langley, 1969 —— 8 CR-1455