SCROLL—A pattern recording language*

by MURRAY SARGENT III

University of Arizona
Tucson, Arizona

INTRODUCTION

A number of routines have been developed recently to facilitate labeling of computer plotted output. One of the more versatile programs is that written by Freeman1 which is capable of plotting characters in sequence including sub and superscripts, over and underscoring, using italics, changing fonts and returning to a saved coordinate. Programs of related nature have been written specifically for the purpose of text editing such as the IBM TEXT 360 and CALL 360 which are primarily printer oriented. Along these lines, all conversational programming systems have editing facilities. The routines share a common feature: the interpretation of character strings containing substrings specifying the desired output and other substrings specifying control functions. The substrings are separated typically by a break character such as a dollar sign or slash followed by a character representing the purpose of the substring. The use of a single character to represent a word or idea is as old as language itself (& for and, $ for dollar, etc.) and the characters so used are called logograms or logographs. One of the early programming languages to use logograms is APL although the purpose of that language is very different from the string languages involved above. A clear advantage of the logogrammatic language is its brevity. However this can be a confusing factor as well.

In this paper, we present a new logogrammatic language called SCROLL which extends the string language of Freeman to allow nesting of the subsuperscript, over-underscore and backward reference facilities and most important to include recursive procedure and measurement capabilities hitherto absent in plotting languages.* An abstract pattern can be defined by such a procedure and invoked as desired with specification of appropriate arguments to yield a specific pattern. Hence a mathematical fraction procedure measures the dimensions of the plotted output corresponding to its two arguments, one for the numerator and one for the denominator. This procedure then centers the numerator with respect to the fraction bar and denominator, while positioning the pattern elements vertically to prevent intersections. The arguments consist of any allowable sentences of the language and can in particular reference the procedure to which the arguments themselves belong. This allows one to draw, for example, a fraction in the numerator of another fraction.

The semantics and syntax of SCROLL are given in the next section of the paper together with numerous examples of plot output. A detailed discussion of procedures and measuring functions is given with examples in the third section. The utility of the language is discussed in the last section. In the appendices, SCROLL syntax is specified using a meta-language similar to that used in the COBOL report, and definitions of built-in SCROLL procedures given.

SEMANTICS AND SYNTAX

SCROLL sentences are composed of plot and control statements which, syntactically, can be mixed together in any order such that the final statement is a termina-

* Most of the work discussed herein was completed at Bell Telephone Laboratories, Inc., Holmdel, New Jersey.

* SCROLL is an acronym for String and Character Recording Oriented Logogrammatic Language. The language has been incorporated into a general plotting system described in Bell Telephone Laboratories memorandum MM 69-1254-11 by M. Sargent III. Details of the SCROLL implementation can be found there.
### Table I—Semantics of Logograms for IBM 360, CDC 6000 and Machine Independent Versions of SCROLL.

<table>
<thead>
<tr>
<th>Definition</th>
<th>IBM 360</th>
<th>CDC 6000</th>
<th>(machine independent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font change</td>
<td>0-9</td>
<td>A-Z</td>
<td>0-9</td>
</tr>
<tr>
<td>Case Inversion</td>
<td>A-Z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subscript</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Superscript</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Sub-superscript return</td>
<td>=</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>Italics</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>Under-scorecoring</td>
<td>(</td>
<td>(</td>
<td>(</td>
</tr>
<tr>
<td>Ending sentence</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>Carriage control</td>
<td>,</td>
<td>,</td>
<td>,</td>
</tr>
<tr>
<td>Column control</td>
<td>$</td>
<td>=</td>
<td>T (FIV FORMAT)</td>
</tr>
<tr>
<td>Linewidth</td>
<td>$</td>
<td>[</td>
<td>W (Width)</td>
</tr>
<tr>
<td>Character size</td>
<td>@</td>
<td>]</td>
<td>S (Size)</td>
</tr>
<tr>
<td>Omitting output</td>
<td>&lt; &gt;</td>
<td>&lt; &gt;</td>
<td>H (Here) T (There)</td>
</tr>
<tr>
<td>Coordinate changes</td>
<td>;</td>
<td>;</td>
<td>C (Call)</td>
</tr>
<tr>
<td>Procedure calls</td>
<td>?</td>
<td>↓</td>
<td>D (Diagnostics)</td>
</tr>
<tr>
<td>Diagnostics</td>
<td>!</td>
<td>↑</td>
<td>L (Logogram)</td>
</tr>
<tr>
<td>Plotting $</td>
<td>$</td>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>Changing semantics</td>
<td>!</td>
<td>!</td>
<td>*</td>
</tr>
<tr>
<td>Process statements</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Null</td>
<td>%</td>
<td>=</td>
<td>N (Null)</td>
</tr>
<tr>
<td>Backspace</td>
<td>&amp;</td>
<td>∧</td>
<td>B</td>
</tr>
</tbody>
</table>

* A plot statement consists of any string of characters (a Hollerith literal) terminated by and not including a dollar sign. The action implied by a plot statement is the plotting of the characters constituting the statement. Hence the string ‘ABC’ means “plot ‘ABC’.” This statement cannot by itself constitute an entire sentence; it must be followed by a control statement which terminates the sentence.

Control statements are also character strings, but inevitably begin with a dollar sign and end according to context as described below. The first nonblank character following the dollar sign stands for the type of action demanded by the control statement. As such the character is a logogram and gives SCROLL its logogrammatic nature. More characters may be included in the statement depending on its purpose. Blanks occurring between the initial $ and the final character of the control statement are ignored unless they belong to a plot statement which is the argument of a SCROLL procedure or function call. The simplest SCROLL sentence consists of the single control statement ‘$’ which means “terminate the sentence.” Combined with the plot statement above, one has the sentence ‘ABC$’ which means “plot ‘ABC’.” Most control statements have fixed length. Those having variable length are terminated either by a per cent sign (%) of the $ of the next control statement.

In the remainder of this section, the control statements are defined and illustrated by figures containing prints of unretouched output obtained using an IBM 360/65 computer in conjunction with a Stromberg-Carlson 4060 microfilm recorder. The semantics given are for the IBM 360 version; Table I summarizes these semantics, the CDC 6000 version* and a machine independent set. Additional examples of SCROLL sentences are given in Appendix B which gives the definitions of the built-in procedures.

1. Specifying a new type font (see Figure 1a) 
   \$n \text{ change to the font numbered } n (1 \leq n \leq 9) .

Four interpretive fonts** have been used herein: #1, the upper case English font; #2, the lower case English font; #3, the upper case Greek font, and #4, the lower case Greek font. The fonts include the special symbols \% + - = \( \) / * | ; ; ; \( ; ; ; < > [ ] \)

\{ \} and characters representing differentiation, integration, infinity, summation, product and the Yale seal. Characters not

---

* The reader may prefer the schematic definition of SCROLL syntax given in the Appendix A to that given here.

---

* Debugged at the University of Arizona Computer Center.

** An interpretive font consists of a sequence of coordinates and delimiters which is interpreted and scaled to produce characters of desired size.

From the collection of the Computer History Museum (www.computerhistory.org)
on key punch are retrieved by a number sign # followed by a key punch character. 
#A and #B, for example, yield { and } respectively. See the plot system memo for further discussion.
$0$ return to previous font.

2. Shifting case for one character only (Figure 1b) 
$\text{s}_a$ where a is any letter of the alphabet inverts the shift for one character only: if $\text{s}_A$ is encountered and a lower case font is set up "A" will be plotted (instead of "a") and succeeding characters will be plotted in lower case.
Note: The case may be shifted for one or more characters by changing to the appropriate font or by typing parts of SCROLL sentences in lower case when an upper case font is set up.

3. Subscripting and superscripting (Figure 1c) 
$\text{s}_-$ enter subscript mode, 
$\text{s}_+$ enter superscript mode, 
$\text{s}_=$ return to previous sub or superscript mode.

4. Italics (Figure 2a) 
$\text{s}$/ enter italics mode and 
$\text{s}\mid$ leave italics mode.

5. Under and overscoreing (Figure 2c) 
$\text{s}($ remember where to start drawing a line, 
under or overscoring,

Figure 1—Examples of SCROLL sentences illustrating control statements for (a) switching type fonts, (b) shifting case for one character and (c) sub and superscripting

Figure 2—Examples of SCROLL sentences illustrating control statements for (a) italics, (b) bold face, (c) over and underscoring, (d) returning to a saved plot position and (e) plotting a dollar sign

$\text{s}+$ or $\text{s}$—overscore (+) or underscore (−) the characters between this $\text{s}$ and the corresponding $\text{s}$; ($\text{s}$ and $\text{s}$) are treated as a pair the same way right and left parentheses are treated in FORMAT statements,
$\text{s}$ followed by anything else, draw a line between current plot coordinates and those saved by corresponding $\text{s}$. See also $\text{s}$ _ facility for drawing lines.

6. Ending sentence (see next section for examples) $\text{s}$_. (or $\text{s}$;) If encountered in procedure or argument sentence, return to calling sentence; otherwise return to the plot system.

Figure 3—Examples of SCROLL sentences illustrating control statements for (a) skipping to next line, (b) changing column, (c) backspacing and (d) drawing lines

$\text{s}_-$ $\text{s}$remember where to start drawing a line, under or overscoring,
7. Starting a new line (Figure 3a)
   \$n advance “carriage” according to value of n:
   n = plus, go to beginning of current line;
   n = 1, go to next frame; n = 0, skip a line;
   n = 2–9, advance \((1/n)\)th of current frame;
   n = anything else, go to next line. If the
   control requests plotting below the frame,
   the frame is automatically advanced.

8. Specifying plot column (Figure 3b)
   \$n \(n = \text{COLNO} - \text{variable in PLTRM de­}
   \text{scribed in plot system memo)} \) go to column
   n with respect to left side of film (ignore
   XORG and X, positioning variables). The
   variable COLNO determines the number of
   columns assumed on a frame and has the
   default value 80.0. Hence unless COLNO
   is reset larger than 100.0, the control se­
   quence \$6100’ will require plotting outside
   screen boundaries and error messages will
   be issued. The letter n can be the name of a
   SCROLL process variable containing the
   desired column number. Hence on can set
   “tabs” for printing text.

9. Changing linewidth (Figure 2b)
   \$#n for n = 1 to n = 4, use standard line
   density and change linewidth to n where
   width is proportional to \(2^n\) (gives varying
   degrees of bold-face type); for n = 5 to
   n = 8 change linewidth to n – 4 and use
   light density.

10. Changing character size
    \$@n change character size to \(\text{SIZE} \times n/2\), where
    SIZE is the nominal character size;
    \$@0 go back to previous size.

11. Omitting output
    \$\r omitted output until \$\r; is encountered,
    \$\r; restore normal output.

12. Saving and returning to a point (Figure 2d)
    \$< save the current plot coordinates;
    \$> go back to the coordinates saved by the \$<
    corresponding to this \$>; \$< and \$> are
    treated as a pair the way right and left
    parentheses are treated in FORTRAN
    FORMAT statements).

13. Calling and defining SCROLL procedures (see
    next section)
    \$:an call the procedure named by the upper
    case alphanumeric character a and nu­
    meral n (1–9 or null),
    \$:an(list) call the procedure named an with
    arguments given by list,
    \$: (an, sentence) define a procedure named an
    by the SCROLL sentence sentence.
    \$:? read characters on next card in FORT­
    RAN input stream into SCROLL storage
    starting at this \$ and continue interpreta­
    tion. A 0–2–8 punch does the same thing
    and can occur anywhere in a SCROLL
    sentence.

14. Backspacing (Figure 3c)
    \$&n backspace n (1 \(\leq n \leq 9\)) characters exclu­
    sive of control characters,
    \$&0 backspace 10 plotted characters,
    \$& followed by anything else—backspace one
    plotted character and process the next
    character as usual.
    Up to ten characters can be backspaced
    over at a given time unless fewer than ten
    positions have been established. To back­
    space more generally, one must use the
    \$< and \$> facility (see 12).

15. Rotating output
    \$’ \([\text{exp1}]\,[\text{exp2}]\)% where exp1 and exp2 are
    SCROLL arithmetic expressions. The azi­
    muthal angle (angle of rotation from the x
    axis in the x–y plane) is set equal to the value
    of exp1 if present, and the polar angle
    (angle of rotation from the z-axis towards
    the x–y plane) is set equal to the value of
    exp2 if it appears. The two angles have
    default values 0° and 90° respectively. Note
    that in SCROLL Version I, all previous
    measurement information is destroyed by
    this control statement. Hence one cannot,
    for example, draw an unrotated rectangle
    around a rotated pattern.

16. Shifting plot position and drawing lines (Figure
    3d)
    \$_field[\_field] \ldots \% shift plot position and/or
    draw lines. A field contains one or more co­
    ordinate sets separated by semicolons and
    the sets themselves are SCROLL arith­
    metic expressions separated by commas.
    If a field consists of a single set, the plot
    position is shifted such that the first co­
    ordinate is added to the x position and the
    second to the y position. If a third co­
    ordinate appears it is used as the z (depth)
    coordinate in a projected drawing. If an
    equal sign precedes the x coordinate, the
    set specifies an absolute location on the
    plot screen. If more than one set appear
    in a field, lines are drawn between the
points they determine relative (no =) to the current plot position. If a ‘>’ is the first character of a field, the coordinates are treated as vectors, that is, displacements from the current plot position. Lines are drawn and shifts occur. If a coordinate is omitted, it is assumed to be zero; if a single coordinate (no comma) appears, it is assumed to be x. Examples of shifting are given in another section and examples of line drawing in the “box” procedure. Shifts can be made in process statements also.

17. Writing diagnostic information
   $?; terminate printing plot diagnostics,
   $?n turn on the flag for diagnostic class n (see discussion of PLTDBG in next section of plot system memo for details),
   $? followed by anything else causes descriptions of the type of actions resulting from subsequent control strings to be printed.

18. Plotting dollar sign (Figure 2e)
   $$ plot dollar sign ($).

19. Changing SCROLL semantics
   $!n use control semantics n (1 or 2), where semantics refers to the meanings of the logograms. $!1 causes the semantics described in this section to be used; $!2 causes semantics presumably specified by the user to be used (perhaps second set given in Table 1).

20. Executing process statements (see Appendix B for examples)
   $* ... % execute the process statements given by the ellipsis ( . . . ) (see next section for further details).

21. Null statement
   $% statement is ignored.

Further examples of SCROLL sentences

To demonstrate a little more of the power of SCROLL we consider the first equation in Figure 5. This resulted from interpretation of the sentence

$*:T($4Q$0$.) = -$I[H, $4Q]$- - $ $
   = -$:F1(1$S.2$.)$[5C.Q]S- +$. '}

Here the string $*:T($4Q$0$.)’ is a procedure call (see next Section) to the time derivative procedure T which centers a dot above the plot output given by the argument, ‘$4Q$0$.’. The argument itself is a SCROLL sentence in which ‘$4’ switches to the lower case Greek font, ‘Q’ plots the letter q, ‘$0’ returns to the previous font (the upper case English font) and ‘$’ ends the sentence. The next four characters ‘=’ ‘-’ plot an equal sign with blanks on either side followed by a minus sign. The ‘$I’ plots i (the dollar sign inverts the case for I alone), ‘[H,’ plots itself, ‘$4’ switches to the lower case Greek font and ‘Q’) plots q. The ‘$ - - ‘ subscripts a minus sign, ‘$ = ’ returns to on-line mode, ‘- ‘ plots a minus sign with blanks on either side, ‘$:F1(1$S.2$.)’ calls the simple fraction procedure to plot the fraction one half, the ‘$C’ plots I, ‘$ - + ‘ subscripts a plus and ‘$ ’ ends the SCROLL sentence. In the IBM implementation for fonts 1 through 4, [ is given by #1 and ] by #2.

The second equation resulted from interpretation of the sentence

   ($$:F(A-B$. B$+Q$-$N$=$=$.)$.) $.’

Here the “box” procedure B plots a rectangle tailored to fit plot output resulting from its argument. Similarly, the partial derivative (P), square root (R) and fraction (F) procedures shift their arguments into place and draw lines of appropriate length.

SCROLL PROCEDURES

In Figure 5 a complex interpretive font character is plotted namely the Yale seal.* One often wants to plot complicated groups of characters, such as a mathematical fraction and could, as for the Yale seal, define the desired pattern as an interpretive font character. In general this is an exceedingly tedious procedure. Instead one would like to define recurrent patterns by procedures and invoke particular patterns using appropriate arguments in the procedure calls. This extremely useful possibility is part of the SCROLL language. Specifically, the call is a control statement having one of the forms

(1) ‘$:an’,
(2) ‘$:an(list)’,

where a (any letter of the alphabet) and n (1-9 or null) identify the procedure, list is a SCROLL paragraph whose sentences comprise the procedure’s arguments. The procedure facility is recursive, that is, a procedure...

\[ \dot{\rho} = -i[H, \rho] - \frac{1}{2}[\Gamma, \rho]. \]

\[ \frac{\partial^m A}{\partial B^m} = \sqrt{\frac{A - B}{A^c}} \frac{\partial^{m-1} A}{\partial B^{m-1}} \]

Figure 4—Plot output resulting from interpretation of SCROLL sentences

can call itself, and procedures can define other procedures. Note that a SCROLL procedure differs from a SCROLL function in that the latter returns a value such as the length of a sentence as contrasted to the execution of plot and control statements.

The fraction procedure

In plotting mathematical equations one pattern which occurs frequently is the fraction. A procedure named \( F \) has been defined to plot a fraction. It is called by the statement

\[ \$:F(n, m) \]

which plots

\[ \frac{n}{m} \]

where \( n \) and \( m \) are SCROLL sentences. In particular the call

\[ \$:F(A - B$. A$ + C$ - $N$ = $ = $.)' \]

plotted the fraction within the square root in Figure 4.

Every SCROLL sentence constitutes a procedure definition. The \( n \)th argument in the procedure call is inserted whenever the characters \&n (1 \( \leq n \leq 9 \)) are encountered. For example, the fraction procedure \( F \) might be defined by the sentence

\[ \$:F(A - B$. A$ + C$ - $N$ = $ = $.)' \]

Here the \$ saves the current \( x \) position for later reference, the \( \$ \) saves the \( x \) position for later reference, the \$ adds half a character size to the vertical \( y \) coordinate, \&1 plots the first argument, \$ returns to the \( x \) coordinate saved by \$, \$ subtracts \( 1.6 \times \text{SIZE} \) from the vertical \( y \) coordinate, \&2 plots the second argument, \$ restores the vertical coordinate to its value at the start of the call and \$ returns to the calling string. Simple fractions can be plotted by the procedure defined by

\[ \$:F(\&1$) - $\&$ = $-$ &2$ = $.$' \]

However, neither definition centers the numerator above the denominator or performs shifts to prevent intersection of lines. For this, one must use the process statement facility described in SCROLL Process Statements section.

Partial derivative and other procedures

An example of a procedure calling another procedure is the partial derivative procedure which plots

\[ \frac{\partial^n y}{\partial x^m} \]

when called by the statements \( \$:P(\&2Y$. X$. N$.)' \). Its definition is

\[ \$:F(\&D$ + $\&3$ = $ = $.)' \]

where \&D retrieves \( \partial \) in fonts 1–4. Use of this procedure was made in drawing Figure 4.

An ordinary derivative procedure is available and is named \( D \). A “time derivative” procedure named \( T \)
Figure 6—Examples of SCROLL procedure calls illustrating the “bead” (oval), box, circle, diamond, hexagon and arrow procedures. The bead, box, circle, diamond and hexagon procedures produce figures of just the correct size to enclose the plot output resulting from their arguments

centers a dot above the character(s) specified by the argument. These and other procedures are built into the language (and plot system), are illustrated in Figures 6–8 and are defined in Appendix B. The user can define his own procedures simply by including

Figure 7—Examples of SCROLL procedure calls illustrating the fraction and simple fraction procedures, the square bracket, curly brace and parenthesis procedures, the ordinary and partial derivative procedures. The fraction and bracketing procedures are constructed with precisely the correct sizes to fit the plot output resulting from their arguments

Figure 8—Examples of SCROLL procedure calls illustrating the summation, product and square root procedures

statements of the form

in a call to the plot system. Here a and n identify the procedure and sentence is the SCROLL sentence comprising the procedure’s definition.

SCROLL process statements

Particularly in the execution of procedures it may be necessary to calculate the dimensions of a substring or argument. For example, if the numerator of a fraction has different length than the denominator, the bar of the fraction should be as long as the larger of the two and the smaller should be centered with respect to the larger. Furthermore both should be shifted far enough away from the bar to prevent intersection. Clearly the dimension of the numerator and denominator vary from call to call and cannot be successfully defaulted ahead of time. One needs a processing facility to calculate dimensions on the spot. The $\ast \ldots \%$ control statement serves this purpose and has the form

Here “statement” can be

(1) a branch statement—

\[ \pm n \] meaning branch to the present character position \( \pm n \),

\[ n \] meaning branch to the \( n \)th position in current sentence, where \( n \) can be either an integer constant, e.g., 10, or a SCROLL variable (defined below) whose value has been assigned by a \( q: \) (see (2)); followed by anything else causes a return to the calling sentence (procedure or argument);

(2) an assignment statement

\[ [q:] a [b] \ldots = \text{expression} \]
where the (optional) \( q \) assigns the SCROLL variable \( q \) the current location counter (for later branching), where \( a_{[n]} \ldots \) can be SCROLL variables (see below), \( X, Y \) or \( Z \) which add the value of the expression to the current horizontal, vertical and depth plot positions respectively, \( nP \) which stores into the \( n \)th variable in the labeled common PLTPRM (see plot system memo, op. cit.), or 1, 2 or 3 which store the expression's value into the plot limit variables containing the \( x \) maximum, \( y \) maximum and \( y \) minimum respectively if that value exceeds the one already established (allows one to measure plot output dimensions quickly)

(3) a conditional statement

\[
? \text{logical expression} \{, \text{statement} \} \ldots
\]

where the logical expression has a form described below and the statements are any process statements. The logical expression dominates all statements represented by the ellipsis. A null logical expression is true if and only if plotting is not suppressed.

**SCROLL variables**

SCROLL variables are named by single letters of the alphabet, \( A \) through \( W \), and letters \( A-Z \) preceded by \# (i.e., \#A). Those named by the letters \( A \) through \( W \) are local and automatic, that is, the values they assume in different procedures are unrelated and are lost upon return from the procedure in which they were set. Those named by \#A through \#Z are global and static, that is, their values are known to all procedures and can be modified in any procedure.

When a SCROLL variable is used in an arithmetic context, it is assumed to be floating-point. When used in a logical context, a non-zero value is interpreted as true; a zero value as false.

A SCROLL arithmetic expression consists of a single arithmetic primary, a unary arithmetic operator followed by an arithmetic primary or two or more such expressions separated by binary arithmetic operators. An arithmetic primary can be a fixed point constant, e.g., 0.5 or 3, a SCROLL variable, a variable in common PLTPRM (\( nP \) retrieves the \( n \)th variable), the letters \( X, Y, Z \) which return the current \( x, y \) and \( z \) coordinates, a function reference or an arithmetic expression. In arithmetic context, fixed-point constants and SCROLL variables are assumed to have floating-point format. Values of SCROLL variables and fixed point constants have units of the plot screen. However if a fixed point constant is followed by the letter \( S \), it is multiplied by the current character size before being used. There are four binary arithmetic operators, \( + \) (addition), \( - \) (subtraction), \( * \) (multiplication) and \( / \) (division). There are two unary operators, \( + \) (used for emphasis only) and \( - \) (negation). Binary operators must occur between two primaries or between a primary and a unary operator which precedes a primary.

Expressions are evaluated left to right subject to the following hierarchy. Functions are evaluated first, negation (unary \( - \)) second, multiplication and division third, and addition and subtraction last. The unary plus is ignored. Primaries consisting of parenthesized expressions are evaluated before arithmetic operations are performed and can be used as in algebra to override the hierarchy.

Examples of legal arithmetic expressions are

\[
\begin{align*}
A*B/(C-D) \\
F+E(A\#-Q\#=X\#).*-D \\
B(\$1\#-Q\#8.)
\end{align*}
\]

The second and third examples contain function references described below. Note that \( A*-D \) is a legal expression, for the minus sign is interpreted as a unary operator signifying arithmetic negation rather than the binary operator indicating subtraction. Examples of illegal expressions are

\[
AB-C \quad \text{either an operator is missing between} \quad A \quad \text{and} \quad B, \quad \text{or SCROLL variable is misnamed (only one letter is allowed)},
\]

\[
(A*B*(C+D) \quad \text{unpaired parenthesis},
\]

\[
A*-B \quad \text{two binary operators must be separated by a primary.}
\]

A SCROLL logical expression consists of a single logical primary, a logical negation operator followed by a logical primary or two or more such expressions separated by binary logical operators. A logical primary has the value true or false and can be a fixed-point constant as above, a SCROLL variable, a logical expression enclosed in parentheses or two arithmetic expressions separated by a relational operator. In logical context, fixed-point constants and SCROLL variables are considered true if and only if they are nonzero. Zero values are false. The relational operators are \( > \) (greater than), \( < \) (less than), \( >= \) (greater than or equal to), \( <= \) (less than or equal to), \( = \) (equal to), and \( != \) (not equal to). The binary logical operators are \& (logical and) and | (logical or). There is one unary logical operator, ~ (logical negation). The total hierarchy of operations for logical expressions is as
follows

<table>
<thead>
<tr>
<th>Operation</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation of functions</td>
<td>8 (highest)</td>
</tr>
<tr>
<td>Unary + and -</td>
<td>7</td>
</tr>
<tr>
<td>* and /</td>
<td>6</td>
</tr>
<tr>
<td>+ and -</td>
<td>5</td>
</tr>
<tr>
<td>&gt;, &lt;, &gt;=, &lt;=, = and ≠</td>
<td>4</td>
</tr>
<tr>
<td>¬</td>
<td>3</td>
</tr>
<tr>
<td>&amp;</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As with arithmetic expressions, parentheses can be used to override the hierarchy. Examples of valid logical expressions are as follows

\[ A \geq B(ABC\%) \neg C \]

\[ F \]

\[ G\&'(A \mid B) \]

**Process functions**

Five functions are defined for the measurement of the plotted output resulting from SCROLL sentence interpretation: B returns the bottom and length of its argument; D returns the length, height, bottom difference height-bottom of its argument; E returns the length alone, G is the same as D except that plotting can take place concurrently with measurement, and H returns the height and length of the argument. When used in an arithmetic statement other than a simple assignment, only one value (the first) is returned; in simple assignment statements, the first value returned is stored in the variable indicated, the second in the next variable in the alphabet, etc. The dimensions are all given in plot screen units. Vertical measurements (height and bottom) are made relative to the bottom of an on-line (not sub-superscripted) period. The functions are recursive.

In addition a maximum function M is defined which returns the maximum value given by the SCROLL variables appearing as arguments (see example below).

**Example of use**

As an example of process statement use, consider an extended definition of the fraction procedure F

\[ \text{F} \]

Here for the sake of readability, numerous blanks have been included; ordinarily one deletes blanks to save space and execution time. The first statement stores in A and B the bottom and length of the first argument (the numerator), the second statement stores in C and D the height and length of the second argument (the denominator) and the third statement adds 0.2" to C. The fourth statement stores in E the larger of the lengths stored in B and D. The control statement \$_{\text{5S}}$ in line 2 shifts the plot position up one half a character size. The two \$_{\text{S}}$ statements save the plot position for future reference and the \$_{-0.5S}$ in line 2 shifts the plot position so that the numerator will be centered with respect to the length E and the bottom of the numerator will be 0.2 of a character size above the bar of the fraction. The percent sign in line 3 terminates the shift field, and \& plots the numerator. The \$_{\text{S}}$ then returns to the plot position established before the numerator was shifted into place and the \$_{\text{F}}$ draws the bar of the fraction (length E) and then shifts so that the denominator is 0.2 of a character size below the bar of the fraction. The percent sign in line 4 terminates the shift field, \& plots the denominator, \$_{\text{S}}$ returns to the saved plot position and \$_{\text{E}}$ shifts to the original vertical position and a horizontal position on the right of the fraction. Note that by saving and returning to a known reference position, one does not have to know where the numerator and denominator finish.

**CONCLUSIONS**

The SCROLL language has been used extensively in the preparation of lecture slides such as that printed
in Figure 4 and in the labeling of graphs for publication as shown in Figure 9. The cost involved is dramatically less than that incurred when a draftsman is used. Specifically, the first equation if Figure 4 ran on an IBM 360/65 (using FORTRAN IV level 0) 0.226 seconds; the second for 0.64 seconds and the complete labeled graph in Figure 9 for one second. A second costs about ten cents on a typical IBM 360/65, and the cost of a frame on the Stromberg-Carlson 4060 microfilm recorder varies between eight and sixty cents according to load. This yields a cost for either slide of about thirty cents. Furthermore, the turn around time for the slides is generally less than a day. Compared to a draftsman, this is orders of magnitude cheaper and faster. Inasmuch as SCROLL with its recursive procedure and measurement capabilities now provides the user with the power hitherto only available from a draftsman, it is felt that SCROLL represents an important advance in the preparation of figures. It is felt that in general SCROLL represents a substantial advance in the computer preparation of manuscripts for publication.

ACKNOWLEDGMENTS

The author is indebted to numerous colleagues at Bell Telephone Laboratories for helpful discussions. In particular, he would like to thank A. G. Fox, C. T. Schlegel, D. A. Vaughan and D. P. Allen for editorial suggestions and J. F. Gimpel and I. P. Polonsky for illuminating conversations about programming languages. The author is especially grateful to R. E. Griswold for helpful suggestions of all kinds and for finding numerous errors in the SCROLL implementation, to G. C. Freeman of the Yale Computer Center whose MAP-coded plotting routines served as starting points for some of the principal subprograms, to J. D. Beyer for very helpful discussions about syntactic analysers and to D. Barth of Yale University for the coordinates of most characters in the interpretive type fonts.

APPENDIX A—SCROLL SYNTAX

In this appendix, a formal definition of SCROLL syntax is given in terms of a meta-language much like that used in the IBM PL/I Reference Manual (C28-8201).

A.1 The Syntax Meta-language

The meta-language used below to define the syntax of SCROLL consists itself of literals, variables, expressions and operators much as SCROLL or any other language does. More specifically, a literal consists of any character which is preceded by a bar (\), a blank, a left square bracket ([) or a left curly brace ({) and is followed by a bar, a blank, a right square bracket (]) or a right curly brace (}). A variable is named by a lower-case letter of the English alphabet followed by any non-zero combination of such letters and underscores (_). A primary is a constant or a variable or a bracketed expression. Square brackets are used when the expression is optional; curly braces when required. A unit is a primary optionally preceded by an ellipsis (\ldots), the latter indicating that the primary is repeated zero or more times. An expression is one of three things: 1) a single unit, 2) a unit followed first by one or more blanks and then by another expression, or 3) a unit followed first by a bar (\) optionally preceded and followed by blanks and then by another expression. In the second case the values of the unit and expression are concatenated; in the third case they are considered to be alternatives. A variable is given the value of an expression when its name is followed first by a colon (:), then by one or more blanks and finally by the expression. A variable is given the value of an English phrase when the variable is followed by a colon, then one or more blanks and finally by the phrase. A phrase is distinguished from an expression in that the former contains undefined variables and always makes sense as a phrase. When a syntactic symbol I \[J {} is to be used as a literal, it is underlined. Hence _{\} is a literal I rather than the operator separating alternatives. In particular the syntax of this meta-language is defined in terms of the language itself as follows:

\begin{verbatim}
literal: \{expression\} 
variable: _letter _{\{expression\} ...} 
primary: literal | variable | \{expression\} | expression 
unit: primary \{ ... \} 
expression: unit [[blank]... \{[blank]} ... unit] ... 
definition: variable \{[blank]... expression | an English phrase \} 
\end{verbatim}
APPENDIX B—PROCEDURE DEFINITIONS

In this appendix, the built-in SCROLL procedures are defined. The name of the procedure is given first, then the call indicating the number of arguments for the procedure and a brief description of the procedure and finally the definition itself.

1. ALIGN—$:A9(\&1 \&2 \&3)$, where \&1 is to be centered below \&2 and \&2 is to be centered above \&3 (used for summations and products).

\[
\begin{align*}
& A = D(\&3) \quad B = D(\&1) \quad C = E - 2 \quad D = F + 2 \quad \text{and} \quad E = \frac{D - A - C}{2}.
\end{align*}
\]

2. ARROW—$:A(\&1)$, where an arrow of length \&1 is to be drawn. Plotting terminates at the arrow’s tip; the arrow is horizontal pointing to the right if \&1 is positive and to the left if \&1 is negative.

\[
\begin{align*}
& A = \&1 \quad B = 0.5 \quad C = \frac{A}{2} \quad D = -B \quad E = -A \quad F = -A \quad \text{and} \quad G = F + 2.
\end{align*}
\]

3. ARROW1—$:A1(\&1)$, where an arrow of length \&1 is to be drawn. Plotting terminates at the arrow’s tip; the arrow is vertical pointing up if \&1 is positive and pointing down if \&1 is negative.

\[
\begin{align*}
& A = \&1 \quad B = 0.3 \quad C = \&1 \quad D = -B \quad E = A \quad F = -A \quad G = -A \quad \text{and} \quad H = \frac{D - A - C}{2}.
\end{align*}
\]

4. BEAD—$:B5(\&1)$, where \&1 is to be enclosed within a bead (oval).

\[
\begin{align*}
& A = D(\&1) \quad B = D + A \quad C = D + A \quad D = D + A \quad \text{and} \quad E = D(\&1).
\end{align*}
\]

5. BOX—$:B(\&1)$, where \&1 is to be plotted with a box (rectangle) around it.

\[
\begin{align*}
& A = D(\&1) \quad B = D + A \quad C = D + A \quad D = D + A \quad \text{and} \quad E = D(\&1).
\end{align*}
\]