GRAPHICAL SYSTEMS AND TWO-DIMENSIONAL MATHEMATICAL EXPRESSIONS

by

James E. George
Computer Science Section
Colorado State University

ABSTRACT

The additional facilities needed to generate two-dimensional mathematical expression displays and their implications to a graphical language system are discussed and result in a linguistic based graphical system. An elementary mathematical expression display system is used to motivate and illustrate these additional facilities and is defined utilizing a linguistic based graphical meta system. The use of the elementary system is illustrated for the evaluation of expressions, the display of expressions on different display devices and the use of the entire elementary system by a text formatting program to produce linear and two-dimensional text.

I. INTRODUCTION

Early interest in the recognition and generation of two-dimensional mathematical expressions resulted from the disparity between normal mathematical notation and the linear notation required as input to most scientific compilers or produced by early symbolic systems.

MADCAP (Wells, 1961 and 1963) was an early compiler which accepted two-dimensional expressions from typewriter devices; internally the expression was kept as a two-dimensional array - a replica of the expression as it appeared on a printed page. This internal form was then analyzed and converted to a linear expression for use by the compiler. Klerer, May and Grossman (Klerer and Grossman, 1967; Klerer and May, 1965b) also developed a two-dimensional programming system which was typewriter oriented. In their system, arbitrary size symbols could be constructed from elementary strokes; the two-dimensional input is analyzed and converted to a linear representation using a character array for all possible typewriter positions. Although originally designed to recognize mathematical expressions, the basic techniques were extended to manipulate and format mathematical text. More recently, several systems have been utilizing a RAND Tablet with projection for input and output (Anderson, 1968; Bernstein and Williams, 1968); both of these have the two-dimensional recognition capability but are not necessarily tied to a compiler -- the input is analyzed and converted to an internal linear form which may be used by various sub-systems. This internal form is converted to a two-dimensional form for user communication.

Along with the interest in scientific compilers with a two-dimensional capability came an interest in two-dimensional output from symbolic manipulation by computer. Martin's Symbolic Mathematical Laboratory (Martin, 1967) is implemented in LISP and utilizes a display scope for user communication and a plotter for hard copy; two-dimensional output is provided for the scope (Krakauer, 1964). CHARYBDIS (Millen, 1968) is a

program to display the output from MATHLAB (Engelman, 1965) on typewriter like devices. In the previous symbolic manipulation systems, the input is linear and the output expressions have generally been given as two-dimensional expressions. A recent system (Blackwell and Anderson, 1969) has two-dimensional input and output and provides for manipulation of the expressions by user supplied rules.

Additionally, a new data type and two-dimensional concatenation operators have been added to SNOBOL4 (Gimpel, 1972) thus making SNOBOL4 more applicable to two-dimensional text processing.

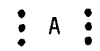
Thus, various systems have utilized special techniques to generate two-dimensional mathematical expressions. With the exception of the SNOBOL4 addition, the techniques used to generate the two-dimensional text displays have not been generalized and included in the host system for use by other applications.

II. TWO-DIMENSIONAL TEXT REQUIREMENTS

Utilizing a graphical system or language to derive a two-dimensional text display from a linear representation requires specific facilities from the graphical system. To illustrate these requirements, consider using a graphical system to implement a system to generate the two-dimensional display of a simple FORTRAN expression (i.e., those expressions containing simple variables and the mathematical operators; addition, subtraction, multiplication, division and exponentiation). The general facilities needed are:

1. Basic symbol and expression representation;
2. Ability to create arbitrary graphic symbols;
3. Ability to perform relative translations between symbols and expressions;
4. Ability to associate data with symbols and expressions;
5. Some method of analyzing the linear input string and generating a two-dimensional representation of it.

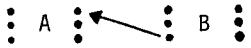
Within the various two-dimensional systems, the primitive representations have been similar and generally consist of alphanumerics with some associated data points which delimit the symbol or expression. For recognizing two-dimensional mathematical expressions, six data points for symbols have been sufficient (Anderson, 1968); although generation can be accomplished with less (Krakauer, 1964; Martin, 1967), it is sometimes necessary to generate virtual points.* An example of a basic symbol with the associated six data points is:



i.e. draw a rectangular box which encloses the symbol; the four corner points plus the mid-points on each side form the six data points.

The ability to create arbitrary graphic symbols is crucial even in this simple case; for example, the length of the division bar is dependent upon the size of the expressions forming the numerator and denominator. Also, variable size parentheses are desirable for devices with vector capabilities. For more sophisticated systems, the ability to create more complicated symbols such as summation, integration, etc. would be needed.

For the illustrative system, two-dimensional translation is required to perform exponentiation; viz.

A ** B 

These translations can be quite complex and involve computations upon the data points related to the symbols or expressions; e.g. the integration of an expression where the lower and upper limits are expressions involves translations among the expressions depending upon their size and the creation of an integral sign related to the size of the resultant expression.

In the SNOBOL4 example (Gimpel, 1972), both the symbol representation and translation become confused. Within this example, virtual points must be generated and the two concatenation operators along with blank primitives accomplish the relative translations, transparently.

Additionally, to generate aesthetic displays utilizing rules of reasonableness for

---

* For a good example of virtual points see the function IMAGE in the SNOBOL4 article (Gimpel, 1972).

parenthesization, intermediate data will have to be associated with intermediate expressions just as the six data points are. For example, keeping track of the last operator will allow the following;

(A + B) * C	becomes	(A + B) * C
A + (B * C)	becomes	A + B * C
A * (B * C)	becomes	A * B * C

Additionally, unary negation must be handled similarly to prevent consecutive operators from appearing in the output; viz.

A + -B	becomes	A + (-B)

Some method of analyzing the linear form to generate the two-dimensional form is needed. In the SNOBOL4 example, the linear form is converted to a prefix notation by a preprocessor, resulting in the input string being scanned twice. It is well known how to evaluate a mathematical expression and there are various general techniques for doing this; abstractly, the form of the expression may be described by a syntax and the meaning by the associated semantics. The syntax which describes how to evaluate an expression by separating or "parsing" the form to sub-parts and applying semantic rules to determine the meaning is precisely the syntax needed to generate the two-dimensional form (George, 1971a and 1972a).

Finally, a two-dimensional text system should be useable in and by a variety of applications and not just to generate a picture of the expression; it should also be device independent or easily device transferable. This is related to the early discovery in graphics that "It is only worthwhile to make drawings on the computer if you get something more out of the drawing than just a drawing." (Sutherland, 1963). Although it is sometimes easier to develop a display system interactively as a master system, it should be useable by other systems (including itself) as a sub-system with only minor modifications by another user.

III. IMPLICATIONS TO A GRAPHIC SYSTEM

Many graphical systems provide for the first four requirements of two-dimensional text generation as discussed in Section II; many also allow the fifth requirement to be satisfied entirely by an application but do not provide for it. However, there has been considerable research in the areas of parsing, general parsers, automatic syntactic analysis and automatic generation of parsers from a syntax (Feldman and Gries, 1968). The inclusion of some automatic syntax description and parser generation capability within a general graphical system would not restrict its present applications but would expand them; in addition to being able to easily experiment with two-dimensional mathematical expression displays, the addition will allow experimentation with linguistic based graphical languages.* But, if a new graphical language is being defined and implemented, these five requirements can easily be accomodated, however, the actual implementation techniques could differ (Morpargo and Sami, 1970).

IV. AN EXAMPLE

A graphical system with the facilities suggested in Section III and IV has been implemented as a meta system (George, 1971a and 1972a); the implementation is written in PL/I and consists of several pre-processors, each defined utilizing a translator writing system (George, 1971c). This meta system allows applications to be defined which may be partitioned into four components; a control component, a data structure with manipulations, a communication template and a graphic language; the graphic language is assumed to be a linear string language, sometimes called concatenation languages (Narasimhan, 1966) or picture description languages (Miller and Shaw, 1968; Rosenfeld, 1969a [Chap. 10] and 1969b; Shaw, 1968, 1969a, 1969b and 1970).

A system to construct, display and evaluate simple FORTRAN expressions has been defined utilizing the above meta system. Figure 1 is an accurate representation of a scope display (George, 1971a contains a photograph from which this figure is constructed). Figure 2 is an example of the display as presented on a printer (note the variable size brackets). Thus, the two dimensional representation can be displayed on various devices.

---

* For a survey of linguistic techniques see (Feder, 1966; Miller and Shaw, 1968; Rosenfeld, 1969a and b).

```
OPTIONS ASSIGN DEFINE DISPLAY EVAL PRINT
OUTPUT PRINTER INIT INPUT POP TERM DBG
OPERATORS + - / * ** NEG
DEFINED VARIABLES A B C D
EXPRESSION = ((A+B)**(A+C))/(D/(B**(C+D)))

         A+C
  [A+B]
 ------------
 [   D   ]
 [-------]
 [   C+D ]
 [ B     ]
 6.150912E+07
SELECT OPTION OR OPERATOR
```

FIG. 1 -- 2-D MATH EXAMPLE -- SCOPE OUTPUT.

```
((A+B)**(A+C))/(D/(B**(C+D)))


                A + C
--          --
|   A + B    |
--          --
-------------------
   --           --
   |    D        |
   |  -------    |
   |     C + D   |
   |  B          |
   --           --
```

FIG. 2 -- 2-D MATH EXAMPLE -- PRINTER OUTPUT.

Computer programs have been developed for formatting text (Berns, 1968 and 1969; Ehrman and Berns, 1971; Meadow, 1970; van Dam and Rice, 1970; ---- (TEXT 360), 1968). I had developed a personal text formatting system (George, 1971b); originally, for bibliography preparation but it was extended to produce reports. After implementing the expression display system, I added a new command in the text formatting program and utilized the two-dimensional system as a sub-system. Figure 3 is reconstructed from a sample output from this combined system (original in George, 1971a).

THIS IS A TEST OF THE VERSION WITH EQUATIONS

This test illustrates the inclusion of equations in the input in linear form being converted to a two-dimensional form.
For example the equation

((A+B)**(A+C))/(D/(B**(C+D)))

appears as:

```
             A + C
--       --
| A + B  |
--       --
---------------
  --        --
  |    D     |
  | -------  |
  |   C + D  |
  | B        |
  --        --
```

FIG. 3 -- TEXT FORMATTING WITH 2-D MATH

Picture languages have also been illustrated for flow charts (Shaw 1968 and 1969b), for line drawings (George, 1971a and 1972a; Shaw 1968 and 1969b) and various other classes of figures. Clearly, these could be added to a text formatting system along with a two-dimensional expression display system, resulting in a flexible text formatting system.

V. CONCLUDING REMARKS

The effects of two-dimensional text generation upon graphics have been presented, analyzed and illustrated. I believe that a merger of both

techniques will harm neither, but will help both. In particular, really acceptable text formatting by computer may be possible by this merger and the introduction of picture description languages.

VI. ACKNOWLEDGMENTS

I wish to thank Dr. William F. Miller for the intellectual encouragement which made this research possible; Dr. E. J. McCluskey, Dr. Cleve Moler and Dr. Alan Kay for their constructive criticisms; and especially Dr. Alan C. Shaw for his criticisms and suggestions.

This work was supported in part by the U.S. Atomic Energy Commission Contract AT(04-3)-515, the National Science Foundation Contract 2SFGJ687 and Colorado State University.