SLAC-PUB-383
January 1968

STRING DESCRIPTIONS OF DATA FOR DISPLAY*

J. E. George and W. F. Miller
Computer Science Department and Stanford Linear Accelerator Center
Stanford University
Stanford, California

Abstract

A Picture Description Language (PDL) and a Picture Calculus have been developed for the formal description and manipulation of pictures. A display program which utilizes PDL string descriptions as the principal data structure has been developed. This display program permits the generation of drawings on a computer-controlled cathode ray tube and allows transformations according to the rules of the picture calculus. A description of the complete data structure for this display program is presented, and the type of transformations and manipulations possible are shown.

1. INTRODUCTION

A formal system is being developed for describing, manipulating, generating and recognizing pictures. This formal system is called the Picture Calculus[1] and consists of two elements. First, a Picture Description Language with both a syntax and semantics and, secondly, a representation which permits easy use of the language.[2]

2. PICTURE CALCULUS

2.1 SYNTAX AND SEMANTICS FOR THE PICTURE DESCRIPTION LANGUAGE

A sentence, S, in the language is defined by:

(1) $S \rightarrow p \mid (S \ \langle \text{BINARY OP} \rangle \ S) \mid (\langle \text{UNARY OP} \rangle \ S)$,

(2) p is a primitive class,

(3) $\langle \text{BINARY OP} \rangle \rightarrow + \mid \times \mid - \mid *$,

(4) $\langle \text{UNARY OP} \rangle \rightarrow \sim \mid \ulcorner \mid T(\omega)$.

2.1.1 The Primitives

A primitive class, p, is defined as any object with a head and a tail. The primitive class is given a name and a specification of its head and tail, and is defined further by a list of attributes. Each attribute may take on a set of values according to its definition including being unspecified. For example, the primitive class for ARC has the form:

ARC = (ARC, initial angle, final angle, curvature, subtended angle) .

An element p' of the class p has a value list containing specific values of the attributes in the attribute list of the class. For example, the primitive ARC' has the form:

ARC' = (ARC', 90, 45, - 2, 45)

which is of the general form,

value (p') = (Name, Tail, Head, Attribute 1, Attribute 2, ...) .

* Work supported in part by the National Science Foundation (Grant No. GP-7615) and in part by the U. S. Atomic Energy Commission

(9th Annual Symposium of the Society for Information Display, May 22-24, 1968)

Thus the attribute list contains information on how to generate the desired picture. It is also possible for an attribute to be a list, say of absolute or relative coordinates which give the explicit representation of a particular primitive.

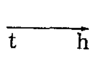
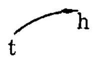
2.1.2 Operator Semantics

In all cases:

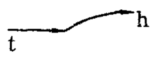
Tail (S1 <BINARY OP> S2) = Tail (S1)

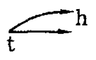
Head (S1 <BINARY OP> S2) = Head (S2)

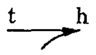
The binary concatenation operators specify how pictures are composed from more basic elements. These operators are defined by the following illustrations:

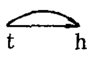
let S1 be  and S2 be 

then

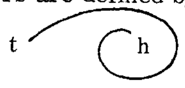
S1 + S2 =  (head to tail)

S1 × S2 =  (tail to tail)

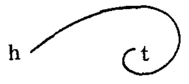
S1 - S2 =  (head to head)

S1 * S2 =  (tail to tail and head to head)

The unary operators are defined by:

let S be 

then

~S  (tail head reversal)

¬S ṫ ḣ (the blanking operator)

The third unary operator, T(ω), is a transformation operator which may transform primitives or pictures; the intent is to allow linear transformations such as rotation and translation; possibly non-linear transformations as well.

Thus the essential elements of the Picture Calculus are defined; however in this form, it is cumbersome to use due to the excessive use of parentheses. Two additional rules for interpreting a sentence in the language will correct this:

(1) Unary operators have precedence over binary operators,

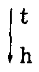
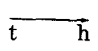
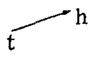
(2) A sentence is interpreted by a left-to-right parse with each operator having the smallest possible scope.

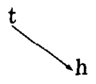
For example:

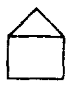
$$A + B + \sim C + D = (((A + B) + (\sim C)) + D)$$

2.2 GENERAL RESULTS USING THE PICTURE CALCULUS

One important feature of the Picture Calculus is that proofs about pictures are possible with this formalism.(2) Another feature is the separation of the operators and the primitive representation. A primary use is for the conversion between a string description and a graphic representation and the reverse. The former yields a picture generation system and the later a picture recognition system. For example,

let A =  B =  C = 

D = 

then,  $= A + B + \sim A * B * (C + D)$

Hence, instead of describing graphical concepts, a string representation may be used.

Currently the Picture Calculus has been used for recognition of particle physics pictures, generation of pictures, and for the description of the alphabet, text and flow charts.(2)

3. REPRESENTATION OF PRIMITIVES

The actual representation for primitives used in the display system differs somewhat from the general form presented. When implementing the display program, several problems developed:

(1) A need to save partially completed pictures,

(2) The normal requirement of finite storage,

(3) The attribute list must be explicit unless an interpreter system is provided for evaluating the attributes.

For practical reasons the primitive class concept was not used; instead only specific primitives are used. However, the system does provide for defining new primitives as an explicit operation or implicitly as the result of an assignment.

One general hierarchical form is used for both primitives and partially completed pictures when they are saved using the assignment function. This form is:

primitive = (name, basic, string, tail, head, image)

where

name is the primitive name,

basic is true if tail, head and image specifications are present,

string is the string expression if the primitive is a partial picture,

tail is the relative tail specification,

head is the relative head specification,

image are the relative coordinates of all the vectors which define the primitive.

Thus two types of primitives are defined; basic primitives have an explicit vector description and possibly a string description, whereas, non-basic primitives (temporary variables) have a string description. The general rule is to always retain the string description and to conditionally retain the explicit vector representation if sufficient storage is available. This explicit vector representation is required at some point in the display process since the graphic display device must be given explicit vector orders.

In the display system, pictures are constructed by forming a string description of the desired form. The picture is displayed by evaluating this string. This evaluation is performed in two distinct steps:

(1) All non-basic primitives must be replaced by their string description;

(2) The resultant string is evaluated by a left-to-right parse utilizing a push-down stack.

During the string expansion, infinite recursion is avoided by arbitrarily monitoring the length of the expanding string at each step.

During use of the display system, vectors are accumulated at a rapid rate and require considerable storage. The string descriptions proved to be a compact form which require considerably less storage, however, regenerating requires more computer time. Thus, the system utilizes storage if available, otherwise, it utilizes time.

The presence of the string in a partially completed picture offers the facility for other features important to a display system. This string may be modified by deletion, replacement or additional items may be concatenated onto it by operators. More importantly, it may be re-evaluated to reflect changes in its constituent parts.

For picture generation, one of the main strengths of the Picture Calculus rests in the transformational operator. Currently, the display system does not include this, however, it has proven useful enough to consider extensions such as the transformational operator and others.

4. EXTENSIONS

4.1 OPERATORS AND PRIMITIVES

The full generality of the primitive representation should be used for a larger system for two reasons. First, instead of many instances of a class being maintained, only the general class need be maintained. Further, one can then describe classes of pictures and illustrate particular members by selecting instances of the constituent primitives. Secondly, this results in a greater degree of generality in the manner in which the operators are applied. For example, consider the case of the "*" operator with one argument absolutely specified and the second argument a primitive class. The requirement of matching between the tail and the head of both arguments will result in an instance of the class of the second argument being selected. This is similar to the way a human draws a line between two points.

The transformational operator discussed so far is what is termed a static operator; i.e., it is applied once per occurrence in the evaluation. Another type which has proved useful in other systems(3) is the dynamic or continuous transformational operator. This is generally referenced to real time and used to illustrate motion which under proper use gives the effect of depth. This can be accommodated easily within the Picture

Calculus by allowing another argument τ; thus $T(\omega, \tau)$ would represent a transformation $T(\omega)$ applied at a time increment of τ. Thus, modeling of problems where motion is important is possible.

4.2 MULTIPLE TAIL HEAD PICTURES

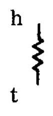
So far pictures have been restricted to having precisely one tail and one head. Although many pictures may be constructed in this manner, it is very inconvenient for some applications. For example, circuit schematics are very difficult because many elements have more than two terminals (transistors have 3, transformers have 4, etc.). A generalization of the representation and of the operators will solve this problem.

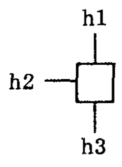
Consider the tail and head in the primitive representation to be a list of tails and heads. Further, let the operators apply as previously, except that they continue until the shortest list used by that operator is exhausted or while matches are allowable. e.g.,

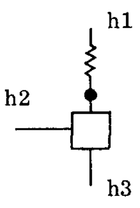
t1——h1

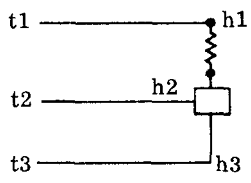
let A = t2——h2

t3——h3

B =


C =


then C + B =


and A - (C + B) =


Two additional operators will be required for this generalized system:

(1) An operator for reordering tails and heads,

(2) An operator for deleting or inserting tails and heads.

The deletion and insertion can be performed by the operators and the normal definitions of:

Head (S1 <OP> S2) = Head (S2)

Tail (S1 <OP> S2) = Tail (S1)

In the example, an object with no tail was used. Thus, an object in this generalized system is something with a tail list and a head list either or both of which may be null.

4.3 TOPOLOGICAL CONCEPTS

Topological concepts such as inside, adjacent and above are considerably more difficult. The basic limitation of the Picture Calculus is that it defines the syntactic structure of a picture; the Picture Description Language does not formally include topological concepts. These concepts can be viewed as "the meaning of a picture." To include these concepts requires the implementation of a specialized recognition system in conjunction with a drawing system. Although possible, it is currently viewed as too restrictive a case for current implementation.

This illustrates that the Picture Calculus is sufficient for describing syntactical structure and possibly simple topological concepts which are included in the primitives; it is not very convenient for describing complex topological concepts without the inclusion of concept recognizers.

4.4 STRING OPERATIONS

With a string description for pictures several natural ideas are suggested. The first are canonical forms and the possibility for a specification of similarity of pictures. Preliminary work on a canonical "tree" form[4] indicates promise for similarity comparisons. The current problem is the recognition of meaningless blank operations within the string. Intuitively, one would desire two pictures to be similar if they have the same tail and head and the same visible structure.

The second desirable item is the conditional replacement of sub-strings. A powerful operation would be to allow a test for a list of sub-strings and if all are present then to substitute another list for those occurrences. This is the type of operation which proved useful in FLIP.[5] This is easily accomplished within the present display program since it was implemented in PL/1 primarily utilizing string operations.

4.5 COMBINED GENERATION RECOGNITION SYSTEM

Currently, the recognition system and the generation system are distinct and separate systems. The recognition system requires a syntax for the class of pictures to be recognized and a set of recognizers for the base primitives. The consolidation of these two systems to provide an interactive pattern recognition system seems attractive. The basic idea is to utilize the generation system to help in specifying the syntax for a class of pictures by generating example members. For this to be successful, a grammar inference heuristic is essential and work is progressing in this area.(6)

5. CONCLUSION

The Picture Calculus has proven convenient and powerful for generating, describing, manipulating and recognizing a large class of pictures. Further, desirable extensions have resulted in very little basic surface modification to the Picture Calculus, thus indicating its usefulness.

References

(1) W. F. Miller and Alan C. Shaw, "A Picture Calculus," SLAC-PUB-358, Stanford Linear Accelerator Center, Stanford University, Stanford, California (presented at the conference on "Emerging Concepts in Computer Graphics," University of Illinois, Urbana, Illinois, November 5-8, 1967).

(2) Alan C. Shaw, "The Formal Description and Parsing of Pictures," Ph.D. Thesis, Stanford University, Stanford, California (in press).

(3) I. E. Sutherland, "Computer Graphics," Datamation, (May 12, 1966) pp. 22-27.

(4) I. Carlbom, "Algorithms for Transforming PDL Expressions into Standard Form and Into a Primitive Connection Matrix," CGTM 38, Computation Group, Stanford Linear Accelerator Center, Stanford University, Stanford, California.

(5) Warren Teitelman, "Pilot: A Step Toward Man-Computer Symbiosis," M.I.T. Project MAC, MAC-TR-32, (September 1966) pp. 45-63.

(6) Jerome Feldman, "First Thoughts on Grammatical Inference," Stanford Artificial Intelligence Memo No. 55, Stanford University, Stanford, California (August 11, 1967).