SLAC-PUB-402
April 1968

A FORMAL PICTURE DESCRIPTION SCHEME AS A BASIS FOR PICTURE PROCESSING SYSTEMS*

by

Alan C. Shaw

Stanford Linear Accelerator Center
Stanford University, Stanford, California

(To be submitted to Information and Control)

*Work supported by U. S. Atomic Energy Commission, the SLAC IBM Graphics Study Project, and the National Science Foundation Grant GP-7615.

LIST OF SYMBOLS USED

$\mathcal{P}$, $\mathcal{G}$, $\mathcal{I}$, $\sum$, α , β , ϵ , λ , $\wedge$, $\vee$, $\cup$, $\subseteq$,

$|$, $\{\}$, $[]$, $<$, $\leq$, $\geq$, $>$, $=$, $\neq$, $\equiv$, $\not\equiv$, $::=$,

$:=$, $\longrightarrow$, $\odot$, $+$, $-$, $\times$, $*$, $\sim$, $/$, $\emptyset$.

Number of Pages: 65 (including figure pages)

Number of Tables: 0

Number of Figures: 17

Proposed Running Head: Formal Description of Pictures.

ABSTRACT

A formal picture description scheme to be used as the basis for picture processing systems is presented. The scheme is applicable to a large class of pictures including, but not restricted to, those containing line-like elements. The paper first presents a general linguistic model for picture processing in which the analysis and generation of pictures are defined as the derivation and execution, respectively, of descriptions. A particular realization of the descriptive component of the model including some of its formal properties is then given; a picture class is described in terms of its underlying graph structures by a grammar generating strings in a picture description language. A series of examples illustrate the capability and limitations of the description scheme. Some applications of implemented systems to the analysis and generation of pictures are discussed.

A FORMAL PICTURE DESCRIPTION SCHEME AS A BASIS FOR PICTURE PROCESSING SYSTEMS*

I. INTRODUCTION

Picture processing is defined here as the analysis and generation of pictures by computer, with or without human interaction; the area thus encompasses both computer graphics and digital pattern recognition. A distinction between "natural" and "artificial" pictures can be made analogous to that between natural and artificial languages. Formal methods of syntax and semantics are often employed to describe artificial languages and subsets of natural languages; moreover, the same mechanism can then be used to drive language analysis systems. A similar linguistic approach is suggested for pictures. This paper presents a formal linguistic description scheme which is applicable to a large class of pictures including, but not restricted to those containing line-like elements. The scheme is intended both as a language of discourse about pictures for humans and as the basis for computer picture processing systems.

The rationale for a linguistic approach to picture analysis has been persuasively presented by Kirsch (1964) and Narasimhan (1962, 1964, 1966). Narasimhan states:

> "... the aim of any adequate recognition procedure should be not merely to arrive at a "yes", "no", or "don't know" decision but to produce a structured description of the input picture. It is our contention that no model can hope to accomplish this in any satisfactory way unless it has built into it, in some sense, a generative grammar for the class of patterns it is set up to analyze and recognize." (Narasimhan 1964).

There have been several serious efforts at providing a linguistic model for picture processing. These include the work of Eden (1961, 1962), Narasimhan

*This research was supported in part by the Atomic Energy Commission, the SLAC-IBM Graphics Study Project, and the National Science Foundation grant GP-7615. The material in this paper appears as part of the author's Ph. D. thesis (Shaw 1968).

(1962, 1964, 1966), Clowes (1967a,b), and Anderson (1968). None of these, however, offer a formal descriptive mechanism upon which a reasonably large class of analysis and generation problems may be treated.

In order to understand the use of descriptions, a general linguistic picture-processing model is first discussed. The description scheme and some of its formal properties are then developed in detail. A series of examples illustrate its capability and limitations. Finally, some applications to analysis and generation problems are briefly mentioned.

II. A LINGUISTIC MODEL FOR PICTURE PROCESSING

The linguistic model consists of two parts:

1. A general model within which pictures may be described.
2. An approach to the analysis and generation of pictures based directly on descriptions.

A picture is conceived as consisting of a number of elementary or primitive patterns related to each other in some meaningful way. At this level, a primitive description scheme is postulated involving two components – structural and semantic. A primitive description $T(\alpha)$ of a picture α is defined:

$$T(\alpha) = (T_S(\alpha), T_V(\alpha)),$$

where $T_S(\alpha)$ specifies the primitive patterns contained in α and their relationship to one another, and $T_V(\alpha)$ gives the values or attributes of each primitive in α. For example, $T_S(\alpha)$ might describe the fact that α consists of two intersecting lines, whereas $T_V(\alpha)$ specifies the exact geometry of these lines. In this way, many pictures might have the same primitive structural description. Define $\mathcal{P}(S) = \{\alpha \mid \alpha \text{ is a picture } \wedge\ T_S(\alpha) = S\}$.

Consider a set of rules or grammar $\mathcal{G}$ generating a language $\mathcal{L}(\mathcal{G})$ whose "sentences" are primitive structural descriptions of a given class of pictures.

Then $\mathcal{G}$ is said to describe the picture class $\mathcal{P}_{\mathcal{G}} = \bigcup_{S \in \mathcal{L}(\mathcal{G})} \mathcal{P}(S)$. Let $\mathcal{S}$ be a given set of interpretation or semantic rules in 1-1 correspondence with $\mathcal{G}$. In addition to its primitive description, a picture $\alpha \in \mathcal{P}_{\mathcal{G}}$ is said to have a hierarchic description $H(\alpha) = (H_S(\alpha), H_V(\alpha))$, where the syntactic or structural component $H_S(\alpha)$ consists of the ordered set or sets of rules of $\mathcal{G}$ that may be used to generate $T_S(\alpha)$ – that is, $H_S(\alpha)$ is the linguistic structure or parse of $T_S(\alpha)$ according to $\mathcal{G}$ – and $H_V(\alpha)$ is the result of obeying the corresponding interpretation rule for each rule of $\mathcal{G}$ used in parsing $T_S(\alpha)$.

Example:

Let $\mathcal{G} = \{LC \rightarrow L,\ LC \rightarrow C,\ LC \rightarrow L \odot C,\ L \rightarrow \ell,\ C \rightarrow c\}$ and $\mathcal{S} = \{v_{LC} := v_L,\ v_{LC} := v_C,\ v_{LC} := \text{xsect}(v_L, v_C),\ v_L := v_\ell,\ v_C := v_c\}$, where ℓ and c are picture primitives naming the class of all line segments and circles respectively, $\odot$ denotes the geometric relationship of intersection, $\mathcal{G}$ is a phrase structure grammar (Chomsky 1959), v_i, $i \in \{LC, L, C, \ell, c\}$, represents the semantics or interpretation of the corresponding rule of $\mathcal{G}$, and xsect is a function which computes the intersection(s) of a line with a circle. Then

$$\mathcal{L}(\mathcal{G}) = \{\ell, c, \ell \odot c\} \text{ and } \mathcal{P}_{\mathcal{G}} = \mathcal{P}(\ell) \cup \mathcal{P}(c) \cup \mathcal{P}(\ell \odot c).$$

If $T_S(\alpha) = \ell \odot c$ for $\alpha \in \mathcal{P}_{\mathcal{G}}$, then $(H_S(\alpha), H_V(\alpha))$ could be represented by the simple tree of Fig. 1.

A complete description $D(\alpha)$ of a picture α is defined:

$$D(\alpha) = (T(\alpha), H(\alpha)) = ((T_S(\alpha), T_V(\alpha)), (H_S(\alpha), H_V(\alpha))),$$

where T and H are the primitive and hierarchic descriptions, and T_S, H_S and T_V, H_V are the structural and semantic components of each.

The general approach to picture processing can now be formulated:

1. The picture primitives are named and defined.
2. The picture class is described by a generative grammar $\mathcal{G}$ and associated semantics $\mathcal{S}$.

3. The analysis of a given picture α proceeds by parsing it according to $\mathcal{G}$ and $\mathcal{J}$ to obtain its description $D(\alpha)$; that is, $\mathcal{G}$ and $\mathcal{J}$ are used explicitly to direct the analysis.

Conversely, a picture α is generated by executing its description $D(\alpha)$.

It is important to note that the "meaning" of a picture is expressed by both its primitive and hierarchic descriptions. Thus, several grammars may be used to generate the same class of primitive descriptions, but the hierarchic descriptions, and hence the meaning, may be different for different grammars. Even more generally, the same picture class may be described by totally different primitive and hierarchic descriptions; the intended interpretation of the picture dictates its description.

The form of the various elements of the model are purposely left open at this point. It is expected that there does not exist one universal form that is useful for all applications. The next few sections discuss a particular realization of the description component within the above framework.

III. PICTURE PRIMITIVES AND CONNECTIVITY

Kirsch (1964) suggests that the elementary or primitive components of a picture be defined as those patterns "which are recognizable by suitable character recognition equipment." The definition is slightly different here. A picture primitive is any n-dimensional pattern ($n \geq 1$) with two distinguished points, a tail and a head. In general, a primitive will be a pattern that can be recognized or generated more conveniently as a unit than in terms of its subparts. Thus, what constitutes a primitive is primarily a matter of convenience and is dependent on the application and picture class. For example, in character recognition, the characters themselves may be primitives, or it may be more advantageous to consider line and curve segments as primitives and describe the characters in terms of these.

A primitive can be linked or concatenated to other primitives only at its tail and/or head. Because only two points of possible concatenations are defined, a primitive can be represented as a labeled directed edge of a graph, pointing from its tail to its head node (Fig. 2); this will be a frequent and useful abstraction. Note that generally there is no inherent direction associated with a primitive pattern per se; the use of directed edges to represent primitives is used to distinguish between the tail and head.

In many applications, the absence of a specific visible pattern in a particular area of a picture is a necessary part of its description. An example is a photograph of some high energy particle physics reactions; the apparent stopping of a particle track and the later appearance of several tracks emanating from the same vertex indicates the presence of an unseen neutral particle (Fig. 3). Blank (invisible) and "don't care" patterns connecting disjoint primitives are also extremely useful for describing simple geometric relations, such as those between adjacent characters of a word and adjacent words in text. When a relationship is to be described between disjoint primitives separated by other patterns, the separating patterns are defined as "don't care" primitives. Blank and "don't care" primitives are therefore allowed.

It is convenient to define one special primitive, the null point primitive λ having identical tail and head. λ consists only of its tail and head point and will be represented as a labeled node in a graph.

A primitive is treated as a member of a pattern class; the latter may be designated by a name, a tail and head specification, and a Boolean function on properties which its members satisfy. The structural description $T_s(\alpha)$ of a primitive α is defined simply as the name of the class to which it belongs; for example, if $T_s(\alpha)$ = line, then $\alpha \in \mathcal{P}$(line). The semantic description $T_v(\alpha)$ is given by the list: value (α) = (tail (α), head (α), $v_1, v_2, \ldots, v_n$), where tail (α) and head (α) are the coordinates of the tail and head of α, and $v_1, v_2, \ldots, v_n$

is an arbitrary list of attributes. The primitive description $T(\alpha)$ of a primitive $\alpha \in \mathcal{P}$(classname) is then $T(\alpha) = (T_s(\alpha), T_v(\alpha)) = (\text{classname}, \text{value}(\alpha))$. A particular instance α of the null point primitive has the description:

$$T(\alpha) = (\lambda, (\text{tail}(\alpha), \text{head}(\alpha))), \text{ where } \text{tail}(\alpha) = \text{head}(\alpha).$$

Example:

Let arc name the class of all two-dimensional pictures $\mathcal{P}$(arc), consisting of an arc of a circle subtending an angle of less than 180°, with tail at the clockwise extremity and head at the counterclockwise extremity of the arc. Then, a particular picture $\alpha \in \mathcal{P}$(arc) with radius r, tail (x_1, y_1), and head (x_2, y_2) might be described as

$$T(\alpha) = (\text{arc}, ((x_1, y_1), (x_2, y_2), r)).$$

Underlying the above definitions is the assumption that there exists one recognition or generation function for each primitive pattern class. On a successful recognition, the recognition function yields the description of the primitive; conversely, the description of a primitive is the input data to the generation function.

The primitive structure of a picture can be represented as a directed graph, where the edges are the abstracted primitives labeled by their class names, some nodes may be labeled λ, and the graph connectivity mirrors the tail/head concatenations of the primitives. A picture is said to be connected if, upon making each edge of its corresponding graph undirected, the resulting graph is connected. The following assumption is made: All pictures are connected.

That this is a reasonable assumption can be seen by considering the extreme case of a picture consisting of a number of disjoint, unrelated primitives. Here, the geometric relation (coordinates) of each primitive relative to the origin of the picture coordinate system is usually meaningful; the connectivity is obtained by linking the origin to each primitive by appropriate blank primitives; this is illustrated in Fig. 4 where t_i and h_i point to the tail and head of primitive i.

IV. THE PDL PICTURE DESCRIPTION LANGUAGE

PDL is a linear string language; a sentence S in PDL – expressed $S \epsilon PDL$ – provides the primitive structural description T_S of a picture by naming all its primitives (their class names) and their tail/head connectivity. The following syntax will generate any sentence $S \epsilon PDL$:

$$S \longrightarrow p \mid (S \emptyset\ S) \mid (\sim S) \mid SL \mid (/SL)$$
$$SL \longrightarrow S^{\ell} \mid (SL \emptyset\ SL) \mid (\sim SL) \mid (/SL)$$
$$\emptyset \longrightarrow + \mid \times \mid - \mid * \ ,$$

where p may be any primitive class name (including λ) and ℓ is any label designator. Any $S \epsilon PDL$ will also be called a PDL expression.

Example:

$$T_S(\alpha) = ((((fn + pred)^W + cond) * (\sim fn)) \times ((/(fn + pred)^W) + cond))$$

for the picture α.

Not only primitives, but all pictures have a tail and a head; concatenations among pictures can occur only at their tail and head positions. Consider the picture α consisting of two subpictures α_1 and α_2 such that $\alpha_1 \epsilon \mathcal{P}(S_1)$, $\alpha_2 \epsilon \mathcal{P}(S_2)$ and $T_S(\alpha) = (S_1 \emptyset\ S_2)$, $S_1, S_2 \epsilon PDL$. Then the tail and head of α according to $T_S(\alpha)$ is defined:

$$\text{tail}(\alpha) = \text{tail}(\alpha_1)$$
$$\text{head}(\alpha) = \text{head}(\alpha_2) .$$

In the same manner as primitives, more complex pictures can thus also be represented by a directed edge of a graph.

The meaning of the binary concatenation operators $\{+, -, \times, *\}$ is presented below by defining $\mathcal{P}((S_1 \emptyset S_2))$; it is assumed that $S_1, S_2 \epsilon PDL$, $\alpha_1 \epsilon \mathcal{P}(S_1)$, and

$\alpha_2 \in \mathcal{P}(S_2)$. The notation cat means "is concatenated onto":

$$\mathcal{P}((S_1 + S_2)) = \{\alpha_1, \alpha_2 \mid \text{head}(\alpha_1) \text{ cat } \text{tail}(\alpha_2)\}$$

$$\mathcal{P}((S_1 - S_2)) = \{\alpha_1, \alpha_2 \mid \text{head}(\alpha_1) \text{ cat } \text{head}(\alpha_2)\}$$

$$\mathcal{P}((S_1 \times S_2)) = \{\alpha_1, \alpha_2 \mid \text{tail}(\alpha_1) \text{ cat } \text{tail}(\alpha_2)\}$$

$$\mathcal{P}((S_1 * S_2)) = \{\alpha_1, \alpha_2 \mid (\text{tail}(\alpha_1) \text{ cat } \text{tail}(\alpha_2)) \wedge (\text{head}(\alpha_1) \text{ cat } \text{head}(\alpha_2))\}$$

The graphs of the resulting pictures are illustrated in Fig. 5; t and h indicate the tail and head of the resulting expression. Figure 6 uses these operators to describe a line drawing of an "A" and an "F"; typical members of each primitive class are shown with arrows pointing from the tail to the head positions.

The connectivity graph of a PDL expression will often be referred to; in this case, the notation tail (S) and head (S) is used to indicate the tail and head nodes of the graph of the PDL expression S. Thus S and each picture in $\mathcal{P}(S)$ has a tail and head position. Tail (S) and head (S) will generically refer to both the pictures and the graph unless specifically stated otherwise.

Because of the freedom allowed in specifying primitive classes, a PDL expression may be undefined for some primitives. For example, if $\mathcal{P}$(arc) is defined as in the example of Section III, and $\mathcal{P}(\ell)$ is the class of all line segments with tail and head at their endpoints, then the concatenation expressed by (ℓ * arc) can only have meaning for those members of $\mathcal{P}(\ell)$ and $\mathcal{P}$(arc) that are geometrically compatible; if $\mathcal{P}$(arc) is restricted so that any chord is less than m units in length, and $\mathcal{P}(\ell)$ is restricted to lines of length greater than 2 × m, then (ℓ * arc) is always undefined, i.e., $\mathcal{P}((\ell * \text{arc}))$ is empty. This is no problem theoretically since the connectivity graph is constructed by treating each primitive abstractly, regardless of whether the concatenations are geometrically possible. It would, however, lead to undefined results in generation and failure in analysis of pictures. This anomaly is ignored henceforth by allowing $\mathcal{P}(T_s)$ to define an empty set of pictures for some T_s.

The binary operators in conjunction with λ are sufficient to describe all possible tail/head concatenations between two pictures, i.e., they are locally complete; Fig. 7 enumerates and describes all possible non-degenerate local concatenations. The unary operators $\sim$ and / do not describe concatenations, but allow the tail and head to be moved. A notion of description equivalence is introduced in order to discuss the unary operators, labeled expressions, and some formal properties of PDL. For $S_1, S_2 \in$ PDL:

1. S_1 is <u>weakly equivalent</u> to S_2 ($S_1 \equiv_w S_2$) if there exists an isomorphism between the graphs of S_1 and S_2 such that corresponding edges have identical names.
2. S_1 is equivalent to S_2 ($S_1 \equiv S_2$) if
   a. $S_1 \equiv_w S_2$, and
   b. tail (S_1) = tail (S_2) and head (S_1) = head (S_2).

The unary $\sim$ operator acts as a tail/head reverser with the following properties:

1. $(\sim S_1) \equiv_w S_1$, $\quad S_1 \in$ PDL
2. tail (($\sim S_1$)) = head (S_1) and head (($\sim S_1$)) = tail (S_1) .

The purpose of PDL expressions with label designators, such as S^ℓ, is to allow cross-reference to that expression within a description; with the / operator, this enables the tail and head to be arbitrarily located. A PDL expression S^ℓ is equivalent to the value of the following function g:

$$
\begin{aligned}
g(S^\ell) = \ & \underline{\text{if}} \ \text{primitive}(S) \ \underline{\text{then}} \ S^\ell \\
& \underline{\text{else}} \\
& \underline{\text{if}} \ S = (S_1 \emptyset_b S_2), \ \emptyset_b \in \{+, \times, -, *\}, \\
& \qquad \underline{\text{then}} \ (g(S_1^\ell) \ \emptyset_b g(S_2^\ell)) \\
& \underline{\text{else}} \\
& \underline{\text{if}} \ S = (\emptyset_u S_1), \ \emptyset_u \in \{\sim, /\}, \\
& \qquad \underline{\text{then}} \ (\emptyset_u g(S_1^\ell)),
\end{aligned}
$$

where primitive (S) = true if (1) S is a primitive class name, or (2) $S = S_1^{\ell}$ where S_1 is a primitive class name, and false otherwise. Concatenated label designators are interpreted as single labels; thus $((a^i + b)^j + a^i) \equiv ((a^{ij} + b^j) + a^i)$.

Figure 8 illustrates the use of label designators and the / operator to describe (a) a picture whose connectivity is equivalent to that of the complete 4-node graph, and (b) a line drawing of a three-dimensional cube in 3-space; in the latter, the primitives are line segments in the X, Y, and Z directions, where the Z direction points into the paper. The explanation of the / operator assumes that any expression S^{ℓ} within a PDL expression has been recursively transformed by the above function g into an equivalent expression so that only primitives have label designators. Then it is required that each primitive within the scope of a / operator, i.e., each primitive that is part of some (/S) within the PDL expression, have a label designator (this is part of the PDL syntax given earlier) and be identical in name and label to one and only one primitive outside the scope of a /. The / is interpreted as a superposition or blanking operator. Each primitive within its scope is another instance of its identical outside primitive; the description of concatenations onto either one will refer to the same primitive. Thus / allows multiple descriptions of the same primitives and structures, effectively moving the tail or head to a more convenient place for further concatenations. A more formal definition of the meaning of / and label designators is given in Section V B.

It is now possible to state completely the rules for determining the tail and head of an expression $S \in$ PDL and of each $\alpha \in \mathcal{P}(S)$:

$$\begin{Bmatrix} \text{tail}\,(S) \\ \text{head}\,(S) \end{Bmatrix} = \underline{\text{if}} \quad \text{primitive}\,(S) \quad \underline{\text{then}} \quad \begin{Bmatrix} \text{tail}\,(S) \\ \text{head}\,(S) \end{Bmatrix}$$

$$\underline{\text{else}}$$

$$\underline{\text{if}} \quad S = (S_1 \, \emptyset \, S_2), \; \emptyset \in \{+, \times, -, *\},$$

$$\underline{\text{then}} \quad \begin{Bmatrix} \text{tail}\,(S_1) \\ \text{head}\,(S_2) \end{Bmatrix}$$

$$\underline{\text{else}}$$

$$\underline{\text{if}} \quad S = (\sim S_1) \quad \underline{\text{then}} \quad \begin{Bmatrix} \text{head}\,(S_1) \\ \text{tail}\,(S_1) \end{Bmatrix}$$

$$\underline{\text{else}}$$

$$\underline{\text{if}} \quad S = S_1^{\ell} \quad \underline{\text{then}} \quad \begin{Bmatrix} \text{tail}\,(g(S_1^{\ell})) \\ \text{head}\,(g(S_1^{\ell})) \end{Bmatrix}$$

$$\underline{\text{else}}$$

$$\underline{\text{if}} \quad S = (/S_1) \quad \underline{\text{then}} \quad \begin{Bmatrix} \text{tail}\,(S_1) \\ \text{head}\,(S_1) \end{Bmatrix}$$

where the function g is defined earlier.

The primitive semantic or value description $T_v(\alpha)$ of a picture α is a list of the descriptions $D(\beta)$ of those primitive pictures β contained in α which have their connectivity and class names described in $T_s(\alpha)$.

Example :

A picture α consisting of a straight line segment concatenated onto an end-point of an arc might have:

$$T_s(\alpha) = (\text{line} + \text{arc})$$

$$T_v(\alpha) = (\,(\text{line}, \,(\,(x_1, y_1), (x_2, y_2), m)\,),$$

$$(\text{arc}, \,(\,(x_2, y_2), (x_3, y_3), r))\,),$$

where m is the line slope and r is the radius of the arc-generating circle.

One more assumption is necessary in order to complete the PDL description scheme. It is assumed that all pictures have a well-defined origin from which a PDL description "starts"; that is, any PDL description S of a picture is interpreted as $(\lambda + S)$ where the tail and head of λ is the picture origin. The origin can be any convenient point in the picture and is usually determined by either the digitization or the generation mechanism. In analysis problems, this normally means that the first primitive concatenated onto the origin is a blank or "don't care" primitive whose recognition function is equivalent to a search strategy to find some interesting visible part of the picture.

V. PDL: FORMAL PROPERTIES AND BASIC THEOREMS

A. Algebraic Properties

The definition and interpretation of the PDL language can be viewed as a picture or graph algebra over the set of primitive structural description under the operations $+, -, \times, *, \sim$, and $/$. Elements ($S \in$ PDL) are considered equal if they are equivalent. A number of useful algebraic properties are given below; it is assumed that

$$S, S_1, S_2, S_3 \in \text{PDL}, \emptyset_b \in \{+, \times, -, *\}, \emptyset_{b-*} \in \{+, \times, -\}.$$

1. Associativity:

   Each of the binary concatenation operators is associative.

   (a) $((S_1 + S_2) + S_3) \equiv (S_1 + (S_2 + S_3))$

   (b) $((S_1 \times S_2) \times S_3) \equiv (S_1 \times (S_2 \times S_3))$

   (c) $((S_1 - S_2) - S_3) \equiv (S_1 - (S_2 - S_3))$

   (d) $((S_1 * S_2) * S_3) \equiv (S_1 * (S_2 * S_3))$

   This allows the elimination of parentheses from an expression whose operators are identical. Thus, $((S_1 + S_2) + S_3)$ can be put in the simpler form $(S_1 + S_2 + S_3)$, and $((S_1 - S_2) - S_3)$ in the form $(S_1 - S_2 - S_3)$.

2. Commutativity:

(a) $*$ is the only commutative binary operator.

$$(S_1 * S_2) \equiv (S_2 * S_1)$$

(b) $\times$ and $-$ are "weakly" commutative.

$$(S_1 \times S_2) \equiv_w (S_2 \times S_1)$$

$$(S_1 - S_2) \equiv_w (S_2 - S_1)$$

3. The $\sim$ Operator:

(a) $\sim$ acts much like complementation in a Boolean algebra:

$$(\sim(S_1 + S_2)) \equiv ((\sim S_2) + (\sim S_1))$$

$$(\sim(S_1 * S_2)) \equiv ((\sim S_2) * (\sim S_1))$$

(b) $\sim$ obeys a "de Morgan's law" with respect to $\times$ and $-$:

$$(\sim(S_1 \times S_2)) \equiv ((\sim S_2) - (\sim S_1))$$

$$(\sim(S_1 - S_2)) \equiv ((\sim S_2) \times (\sim S_1))$$

Note that $\sim$ reverses the order of the operands. The equivalences of (a) and (b) are useful for moving the $\sim$ within an expression.

(c) Involution:

$$(\sim(\sim S)) \equiv S$$

4. The / Operator

(a) $(/(/S)) \equiv (/S)$

(b) $(/(S_1 \emptyset_b S_2)) \equiv ((/S_1) \emptyset_b (/S_2))$

5. The Null Point Primitive λ:

(a) $(S \emptyset_b \lambda) \equiv (\lambda \emptyset_b S)$

(b) $(S \emptyset_{b-*} \lambda) \equiv S$ $\quad (S * \lambda) \not\equiv S$ since $(S * \lambda)$ implies head (S) = tail (S)

(c) $(\sim\lambda) \equiv \lambda$

(d) $(\lambda \emptyset_b \lambda) \equiv \lambda$

B. The Graph of a PDL Expression

By using some of the algebraic properties of the last section to move unary operators and label designators as far as possible within an expression, a standard form $f(S) \in PDL$ of an expression S can be obtained. $f(S)$ is defined:

$f(S) =$ if $(S = S_1 \vee S = (/S_1) \vee S = (\sim S_1) \vee S = (\sim(/S_1)))) \wedge$ primitive (S_1) then S

else

if $S = (S_1 \emptyset_b S_2)$, $\emptyset_b \in \{+, \times, -, *\}$, then $(f(S_1)\ \emptyset_b\ f(S_2))$

else

if $S = S_1^{\ell}$ then $f(g(S))$

else

if $S = (\sim(S_1 \emptyset S_2))$, $\emptyset \in \{+, *\}$, then $(f((\sim S_2))\ \emptyset\ f((\sim S_1)))$

else

if $S = (\sim(S_1 \times S_2))$ then $(f((\sim S_2)) - f((\sim S_1)))$

else

if $S = (\sim(S_1 - S_2))$ then $(f((\sim S_2)) \times f((\sim S_1)))$

else

if $S = (/(S_1 \emptyset S_2))$, $\emptyset \in \{+, \times, -, *\}$, then $(f((/S_1))\ \emptyset\ f((/S_2)))$

else

if $S = (\sim(\sim S_1))$ then $f(S_1)$

else

if $S = (\sim(/S_1)) \vee S = (/(\sim S_1))$ then $f((\sim f((/S_1))))$

else

if $S = (/(/S_1))$ then $f((/S_1))$

Example

$$((\sim(a^i + b)) + (/(\sim a^i)))$$

has the standard form:

$$(((\sim b) + (\sim a^i)) + (\sim(/a^i)))$$

The standard form f(S) of S has the properties:

1. $f(S) \equiv S$.
2. The operand of each / is a primitive class name.
3. The operand of each $\sim$ is either a primitive class name or / followed by a primitive class name.

The function definition is a case analysis of all possible forms of S as given by the PDL syntax.

A valid PDL expression (vPDL) is one whose standard form is such that if $(/p^{\ell})$ appears in it one or more times for some primitive p and label ℓ, then p^{ℓ} also appears once and only once outside the scope of a / .

The graph, and therefore the primitive connectivity, described by a vPDL S is defined by the following algorithm:

1. Transform S into standard form by applying the function f.
2. Replace each expression of the form $(/p^{\ell})$ by a new primitive $p^{\ell}_{/}$. This removes all / operators.
3. Generate the connectivity graph of the resulting expression.
4. Contract the tail and head nodes of each edge $p^{\ell}_{/}$ to the corresponding nodes of p^{ℓ}.
5. Eliminate all edges of the form $p^{\ell}_{/}$.

The above algorithm formally defines the meaning of labeled expressions and the / operator. A simple example is given in Fig. 9.

C. Basic Theorems

1. Connectivity Description

Each step in the formation of a graph of a vPDL can always be performed and has a unique result. This leads to:

THEOREM 1

Any vPDL describes a unique primitive connectivity.

This gives the assurance that one and only one primitive connectivity is represented by a vPDL.

2. Completeness

THEOREM 2

Any connected set of primitives can be effectively described by a vPDL.

Proof:

The proof is by induction on the number n of connected primitives. For n = 1, the vPDL is p, where p is the primitive class name. Suppose that any connected set of n primitives can be effectively described by a vPDL.

Consider (n + 1) connected primitives. Select n of these that are connected, say $p_1, p_2, \ldots, p_n$. By the induction hypothesis, their connectivity may be described by a vPDL:

$$S_n = S_n(p_1, p_2, \ldots, p_n)$$

(a) The first possibility is that the (n + 1)st primitive, p_{n+1}, is connected by only one of its nodes to a primitive in S_n. Then, there must exist at least one p_i, $1 \leq i \leq n$, whose tail or head, or both are connected to p_{n+1}. The following connectivities are possible:

(1) head (p_i) to head (p_{n+1})

(2) tail (p_i) to head (p_{n+1})

(3) head (p_i) to tail (p_{n+1})

(4) tail (p_i) to tail (p_{n+1})

Consider case (1):

Since $p_1, p_2, \ldots, p_n$ are connected, a "path", described by S_i, can be found from $\text{head}(S_n)$ to $\text{head}(p_i)$ such that:

$\text{tail}(S_i) = \text{head}(S_n)$ and $\text{head}(S_i) = \text{head}(p_i)$ (Fig. 10(a)).

The form of S_i can be:

$$S_i = (S_{i1} + S_{i2} + \ldots + S_{in_i}),$$

where $S_{ij} = (\sim p_{ij})$ or $S_{ij} = p_{ij}$ and $p_{ij} \in \{p_i, p_2, \ldots, p_n\}$

$j = 1, 2, \ldots, n_i$.

(Parentheses are omitted in S_i since + is associative).

All the primitives in S_i which are labeled in S_n are now given the same label in S_i. Call the resulting expressions S_i^1. Label uniquely all the unlabeled primitives of S_i^1; attach the same labels to the corresponding primitives in S_n. Call the resulting expressions SL_i and SL_n, respectively. Then the following vPDL describes the connectivity of the $(n+1)$ primitives:

$$S_{n+1} = ((SL_n + (/SL_i)) - p_{n+1})$$

The remaining cases are handled by a similar construction.

(b) The only other possibility is that p_{n+1} is connected at both of its nodes to S_n. Therefore, there exist p_i and p_j, $1 \leq i, j \leq n$, such that:

(1) $\text{head}(p_i) = \text{head}(p_{n+1}) \wedge (\text{head}(p_j) = \text{tail}(p_{n+1})$
$\vee \text{tail}(p_j) = \text{tail}(p_{n+1}))$

or

(2) $\text{head}(p_i) = \text{tail}(p_{n+1}) \wedge (\text{head}(p_j) = \text{head}(p_{n+1})$
$\vee \text{tail}(p_j) = \text{head}(p_{n+1}))$

or

(3) $\text{tail}(p_i) = \text{head}(p_{n+1}) \wedge (\text{head}(p_j) = \text{tail}(p_{n+1})$
$\vee \text{tail}(p_j) = \text{tail}(p_{n+1}))$

or

(4) $\text{tail}(p_i) = \text{tail}(p_{n+1}) \wedge (\text{head}(p_j) = \text{head}(p_{n+1})$

$\vee \quad \text{tail}(p_j) = \text{head}(p_{n+1}))$.

Consider the case:

$$\text{head}(p_i) = \text{head}(p_{n+1}) \wedge \text{head}(p_j) = \text{tail}(p_{n+1}) .$$

As in (a) there is a path, described by S_i, from $\text{head}(S_n)$ to $\text{head}(p_i)$; similarly, there is a path that can be described by S_j from $\text{head}(p_j)$ to $\text{tail}(S_n)$. S_i and S_j satisfy:

$$\text{head}(S_i) = \text{head}(p_i), \ \text{tail}(S_i) = \text{head}(S_n)$$

$$\text{head}(S_j) = \text{tail}(S_n) , \ \text{tail}(S_j) = \text{head}(p_j)$$

(see Fig. 10(b)). The same labeling as in (a) is done except that any primitive common to S_i, S_j, and S_n is labeled in all three expressions. Call the resulting expressions SL_i, SL_j, and SL_n . Then the connectivity of the (n + 1) primitives is described by the vPDL:

$$S_{n+1} = (((/SL_j) + (SL_n + (/SL_i))) * p_{n+1}) .$$

The other cases are treated in a similar manner. Therefore, the case of (n + 1) connected primitives is proven.

Q. E. D.

Note that, in general, more than one vPDL can be obtained to describe the same connectivity; for example, in part (b) of the proof, a similar argument would yield the vPDL:

$$S'_{n+1} = ((SL_n + (/SL_i)) * ((\sim(/SL_j)) + p_{n+1})) .$$

Corollary (Linear Cipher):

Any directed graph can be described by a vPDL.

Theorem 2 proves the completeness of a PDL with respect to the primitive structural description of any connected set of primitives. The corollary further

suggests that graphs of various types may be represented and possibly manipulated within PDL.

3. Moving the Tail and Head

The path construction used in the proof of Theorem 2 can be employed to move the tail and/or head of a vPDL to any node(s) in the structure.

THEOREM 3

Given a vPDL S_1 describing a set of connected primitives whose corresponding graph has n nodes, it is possible to derive $n^2 - 1$ (and no more) other vPDL's, $S_2, S_3, \ldots, S_{n^2}$, such that

(1) $S_i \equiv_w S_j$

(2) $S_i \not\equiv S_j$

$i, j = 1, 2, \ldots, n^2$, $i \neq j$

(3) Each S_i, $i = 2, \ldots, n^2$, is equivalent to an expression having one of the forms:

(a) $((/S_{i1}) + (SL_1 + (/S_{i2})))$

(b) $((/S_{i1}) + SL_1)$

(c) $((SL_1 + (/S_{i2})))$,

where SL_1 is obtained from S_1 by giving the same labels to those primitives in S_1 that appear in S_{i1} and/or S_{i2}.

Proof:

Since there are n nodes in the graph, there are n^2 different ways of assigning the tail and head. Therefore, given S_1 with its tail and head, there are $n^2 - 1$ other assignments that can be made. Since the primitives are connected, a path can always be found from the desired tail to tail(S_1) and from head(S_1) to the desired head. Using the construction in the proof of Theorem 2, expressions of the form (a) (or (b) when the

new head = head (S_1), or (c) when the new tail = tail (S_1)), can always be derived. Properties (1) and (2) follow immediately.

Q. E. D.

Theorem 3 allows one to take the origin (tail) of a picture at any convenient place. It also assures access to any node in the graph when building up descriptions.

4. An Adequate and Independent Set of Operators

The question naturally arises whether label designators and the / operator are necessary or just convenient. Theorem 4 proves the inadequacy of the system without these features.

THEOREM 4:

The operator set $\{+, \times, *, -, \sim\}$ is not sufficient for the description of any connected set of primitives.

Proof:

Assume that / is not part of the PDL language. If $(S_1 \{\overset{+}{\times}\} S_2)$ is contained in a vPDL S, the nodes

$$\left\{\begin{array}{c} \text{head}\,(S_1)\,(=\ \text{tail}\,(S_2)\,) \\ \text{head}\,(S_1) \\ \text{tail}\,(S_2) \end{array}\right\}$$

are inaccessible within S since only tail (S_1) and head (S_2) can be used for further concatenations in S (by definition); furthermore, the inaccessible node has at most two edges meeting at it. Consider a picture whose connectivity is equivalent to that of the complete 4 node graph (Fig. 8(a)); let each edge have the name x. Then any description S of this connectivity must contain a subexpression equivalent to $(X_1 \emptyset_{b-*} X_2)$, where $X_1 = (\sim x)$ or x, $X_2 = (\sim x)$ or x,

and $\emptyset_{b-*} \in \{+, \times, -\}$. $*$ is not possible since $(X_1 * X_2)$ does not describe any subgraph of the graph; this also applies to $(X_1 * \lambda)$. Finally, if only expressions of the form $(X_1 \emptyset_{b-*} \lambda)$ appeared, the equivalence $(X_1 \emptyset_{b-*} \lambda) \equiv X_1$ could be applied to obtain the above form. But, each node must have 3 edges meeting at it. Since the expression $(X_1 \emptyset_{b-*} X_2)$ leaves one node inaccessible with at most two edges tied onto it, S cannot describe the picture.

Q. E. D.

However, there does exist an adequate and independent set of operators.

THEOREM 5

Any vPDL is equivalent to one that uses only the operator set $\{+, \sim, /\}$. Moreover, these operators are independent.

Proof:

The following equivalent expressions demonstrate the adequacy of $\{+, \sim, /\}$:

$$(S_1 * S_2) \equiv ((S_1^i + (\sim S_2)) + (/S_1^i))$$
$$(S_1 \times S_2) \equiv ((S_1^i + (/(\sim S_1^i))) + S_2)$$
$$(S_1 - S_2) \equiv ((S_1 + (\sim S_2^i)) + (/S_2^i)),$$

where i does not appear as a label in S_1 or S_2. $+$ is independent of $\sim$ and $/$, since it is the only concatenation operator. $\sim$ is independent since $+$ and $/$ cannot be used to describe the connectivity: $(a + (\sim b))$. $/$ is independent since $\sim$ and $+$ cannot alone describe the connectivity: $((a + (\sim b^i)) + (/b^i) + (\sim c))$.

Q. E. D.

The set $\{\times, -, *\}$, while unnecessary, is still very convenient, especially in the description of pictures with simple structure.

VI. HIERARCHIC DESCRIPTIONS

The set of rules or grammar $\mathcal{G}$ that describes (generates) the class of pictures $\mathcal{P}_{\mathcal{G}}$ will be a type 2 (context-free) phrase structure grammar (Chomsky 1959) with the following restrictions. Each rule or production is of the form:

$$S \longrightarrow \text{pdl}_1 \,|\, \text{pdl}_2 \,|\, \text{-----} \,|\, \text{pdl}_n, \qquad n \geq 1,$$

where S is a non-terminal symbol and pdl_i is any PDL expression with the addition that non-terminal symbols are allowable replacements for primitive class names. Sentences of $\mathcal{L}(\mathcal{G})$ will consist of PDL expressions; thus, the class of terminal symbols of $\mathcal{G}$ will be a subset of

$$\{+, \times, -, *, \sim, /, (,)\} \cup \left\{\begin{matrix}\text{primitive} \\ \text{class names}\end{matrix}\right\} \cup \left\{\begin{matrix}\text{label} \\ \text{designators}\end{matrix}\right\}.$$

Each grammar $\mathcal{G}$ will have one <u>distinguished</u> non-terminal symbol from which $\mathcal{L}(\mathcal{G})$ may be generated; the symbol on the left part of the first production of $\mathcal{G}$ will be the distinguished symbol. Any sentence $S \in \mathcal{L}(\mathcal{G})$ is assumed to have one parse; that is, $\mathcal{G}$ will be an unambiguous grammar.

The <u>hierarchic structural</u> description $H_s(\alpha)$ of a picture $\alpha \in \mathcal{P}_{\mathcal{G}}$ having primitive structural description $T_s(\alpha) \in \mathcal{L}(\mathcal{G})$ is defined as the parse of $T_s(\alpha)$ according to $\mathcal{G}$; $H_s(\alpha)$ is conveniently represented as a parenthesis-free tree. A simple example is given in Fig. 11.

The use of a formal grammar to describe picture classes has several advantages. Alternatives in a production allow the same name to be assigned to different structures that belong to the same pattern class. Large classes of similarly structured pictures can be concisely defined by recursive productions. For example, all tree structures with "branches" from primitive class b can be defined by the syntax:

$$\text{TREE} \longrightarrow b \,|\, (b + \text{TREE}) \,|\, (\text{TREE} \times \text{TREE})$$

Nodes or points in a picture may be named (and assigned properties by $\mathcal{J}$) by rules of the form:

$$\text{NODE} \longrightarrow \lambda .$$

The rationale behind the selection of context-free grammars rather than more complex ones is mainly one of simplicity; their form is simple, they can generate PDL descriptions for a large, useful, and interesting class of pictures, and there is a great deal of theoretical and practical knowledge on their use in the description and analysis of string languages (Ginsburg 1966, Feldman and Gries 1968).

Corresponding to each rule of $\mathcal{G}$ will be a semantic rule in $\mathcal{I}$. Two sets of semantic rules are postulated – a natural semantics $\mathcal{I}_n$ and an imposed semantics $\mathcal{I}_m$. The natural semantics $H_v(\alpha)$ of a picture α is a list containing the name, tail, and head of each non-terminal symbol (syntax rule) in $H_s(\alpha)$, where the tail and head of a non-terminal symbol is defined as the tail and head of the PDL expression generated by it. Any α_i, i = 1, 2, 3, in Fig. 11 would have:

$$H_v(\alpha_i) = (\ (P,(t,h)_p),\ (HOUSE,\ (t,h)_{HOUSE}),\ (TRIANGLE,\ (t,h)_{TRIANGLE})\ ),$$

where $(t,h)_k$ is the tail and head of k.

The purpose of an imposed semantics is (1) to take an action and assign a value or set of values to a non-terminal symbol upon successful application of its syntax rule during a parse, or (2) to augment the syntax by generation instructions and parameters for a picture generation. A mechanism to express elements of $\mathcal{I}_m$ has not been developed; the natural semantics only is used here.

The description scheme for pictures can now be summarized:

The class of pictures of interest is generated by a given grammar $\mathcal{G}$ such that

$$\mathcal{P}_{\mathcal{G}} = \bigcup_{T_s \in \mathcal{L}(\mathcal{G})} \mathcal{P}(T_s)\ ,$$

and

$$\mathcal{L}(\mathcal{G}) \subseteq \text{PDL}\ .$$

Then, the description of $D(\alpha)$ of any picture $\alpha \in \mathcal{P}_{\mathcal{G}}$ is

$$D(\alpha) = (\ (T_s(\alpha),\ T_v(\alpha)\ ),\ (H_s(\alpha),\ H_v(\alpha)\ )\ )\ ,$$

where $T_s(\alpha) \in \mathcal{L}(\mathcal{G})$,

$T_v(\alpha)$ is a list of the descriptions of all primitives of α,

$H_s(\alpha)$ is the parse of $T_s(\alpha)$ according to $\mathcal{G}$,

$H_v(\alpha)$ is the natural semantics of α.

VII. THE FORMAL DESCRIPTION OF SEVERAL PICTURE CLASSES

The examples of this section illustrate both the power and the limitations of the PDL system as a formal picture description scheme. Comparisons with the work of other researchers are made where appropriate.

Primitive classes are defined informally by a pictorial sample, a mnemonic name, and often a textual description, rather than by a detailed definition of their recognition or generation functions. The latter depend to a great extent on factors that are irrelevant at this point; these include the amount of noise in a particular picture, the hardware used for reading and displaying pictures, and the eventual purpose of the description.

A. Particle Physics

In high-energy particle physics, one of the most common methods for obtaining the characteristics of an elementary particle is to analyze the trajectory "trail" left by the particle and its byproducts in a detector chamber, such as a bubble or spark chamber (Shutt 1967). Several hundred-thousand photographs of these trails or tracks might be taken in a typical experiment. Because of the large numbers involved and the accuracy and quantity of computation required for each "interesting" photograph, machine processing of the pictures is desirable.

In addition to the particle tracks, the pictures usually contain some identifying information (in a "data box"), such as frame number, view, input beam characteristics, and date, and a set of "fiducials," which are marks on the chamber whose positions are precisely known. Fiducials allow the tracks to be reconstructed in real space.

Figure 12 gives the syntax for an abstracted particle physics picture. A negatively charged particle TM is assumed to enter a chamber containing positive particles P and under the influence of a magnetic field; TM enters from the left. The following types of reactions are provided for:

(a) Interaction with P:

$$\begin{aligned} TM + P &\longrightarrow TM + TP \\ &\longrightarrow TM + TP + TN \\ &\longrightarrow TN \end{aligned}$$

(b) Negative Particle Decay:

$$TM \longrightarrow TM + TN$$

(c) Neutral Particle Decay:

$$TN \longrightarrow TM + TP$$

(d) Positive Particle Decay:

$$TP \longrightarrow TP + TN$$

TP and TN represent positively charged and neutral particles, respectively. The notation used above is similar to the conventional physics notation. The products of the reactions can themselves undergo the same series of reactions; this can occur an indefinite number of times. The chamber has four fiducials ("X"s) and an identification box.

The descriptions ($\mathcal{L}(\mathcal{S})$) are ordered for left-to-right recognition in that the lower left-hand fiducial, FI, appears first and its center is then used as the tail for the descriptions of the rest of the fiducials, FID, the identification box, ID, and the particle tracks PT. The sketches of the primitives are only representative. For example, cm and cp are the names of curves with negative and positive curvature, respectively; dp is a short line segment of approximately unit slope. The blank and "don't care" primitives describe known and unknown distances between visible parts of the picture. The primitives eh and ev would be precisely

defined a priori since the fiducials are in fixed positions relative to each other. On the other hand, es, the starting primitive, would be defined as a search strategy to find the lower arm of the left-corner fiducial.

The use of λ for the vertices of interaction, P and N, illustrates the ability of PDL to deal meaningfully with points as well as edges. Physics pictures of this type are natural candidates for description by recursive syntaxes; the recursive definitions of TM, TP, and TN are based on charge conservation and allow for an indefinite number of well-formed reactions.

B. Kirsch's 45^{o} Right Triangles

Kirsch (1964) presents a two-dimensional context-sensitive grammar that generates all 45^{o} right triangles in a plane divided into unit squares; this is suggested as an illustration of the possible form of picture grammars. Unfortunately, it has not been possible to generalize his approach to more interesting pictures (Lipkin, Watt, and Kirsch 1966).

Figure 13 contains a syntax and examples of two-dimensional 45^{o} right triangles with the same point identifications or labels as that given by Kirsch. The primitives are defined as all translations over a two-dimensional grid of the samples shown. Each point in a triangle is assumed to appear as an "X" on one raster unit (square, grid point).

When a picture is represented as finite grid of points, the possible coordinates of the tail and head of any picture (including λ) are restricted to the grid point coordinates. The definitions of the binary operators as concatenations onto means that the expression $(h + \lambda)$ describes pictures where the coordinates of λ are identical to those of head (α), $\alpha \in \mathcal{P}(h)$ (the rightmost "X" in α); also if $\alpha \in \mathcal{P}((h+v))$, α is of the form $\begin{matrix} X \\ XX \end{matrix}$. These interpretations are used for digitized pictures.

The identification of the triangle points as interior (I), base (B), hypotenuse (H), right vertical leg (L), right angle (R), and the vertices bounded by the hypotenuse (V and W) is accomplished by the rules:

$$\left\{\begin{array}{c} I \\ B \\ H \\ L \\ R \\ V \\ W \end{array}\right\} \longrightarrow \lambda$$

In the examples, the subscript on each "X" indicates its label.

The right-triangle syntax will also generate expressions which do not describe any pictures; this is an example of the problem discussed in Section IV. DH and DI might not be the correct "length" for the $*$ concatenation; if this is the case, the class of pictures described by the particular T_s is empty.

C. Simple Block Letters and a Page of English Text

A block version of the upper case letters of the English alphabet is described in Fig. 14. Parentheses which are redundant because of the associativity of the operators are omitted. The PDL expressions for each letter were formed so that the tail and head is located uniformly throughout the alphabet on the "typographic" line; pictures containing groups of letters and other symbols can then be characterized by PDL easily. The description could be rewritten as a grammar taking advantage of some of the common structures in the letters; for example, $\rceil$ appears in P, R, S, and B.

The expressions in Fig. 14 can be compared to the descriptions used by Narasimhan (1966) for generating the upper case English alphabet. Narasimhan uses productions or rewriting rules of the form:

$$S(n_s) \longrightarrow S_1 \cdot S_2(n_{S_1S_2}; n_{S_1S}; n_{S_2S}) ,$$

where S_1 is a terminal symbol (primitive name) or non-terminal symbol (phrase name), S_2 is a terminal symbol, S is a non-terminal symbol (the defined phrase), $n_{S_1S_2}$ is a list of the nodes of concatenation between S_1 and S_2, n_{S_1S} and n_{S_2S} define the correspondence between the nodes of S_1 and S_2 and those of S, and n_S is a node list labeling the nodes of S. All nodes of possible concatenation must appear in the description; this is cumbersome for simple pictures such as the English alphabet, and might be unmanageable for more complex pictures. The system can only describe connected pictures and some other mechanism is required when dealing with pictures whose subparts are not connected. Because only two nodes of possible concatenation are defined in a PDL description, it is not necessary to explicitly number nodes and maintain node lists.

A page of text is broken into sentences, lines, words, and characters by the syntax of Fig. 15. Blank primitives establish the connectivity of words on a line (iws), characters within a word (ics), and lines (ils). Left, right, and bottom are left, right, and bottom of the page indicators. The PDL expressions of the last figure could be used for the letters generated by CHAR. This type of syntax could conceivably be the basis for analyzing and generating textual information.

D. Closed Boundaries of Figures

A description of the edge sequences comprising the boundary of a figure can be easily expressed by a PDL grammar.

Example:

$$\text{BOUNDARY} \longrightarrow (\text{CURVELIST} * \lambda)$$

$$\text{CURVELIST} \longrightarrow \text{CURVE} \mid (\text{CURVELIST} + \text{CURVE})$$

$$\text{CURVE} \longrightarrow c1 \mid c2 \mid \ldots\ldots\ldots \mid cn,$$

where $\{ci \mid i = 1, n\}$ is the set of edge or curve types that may appear in the figure.

Ledley et al. (1965) employ a standard BNF syntax to describe the curves of chromosome boundaries. Examples of their productions are:

$$\langle \text{arm} \rangle \quad ::= \quad B \ \langle \text{arm} \rangle \ | \ \langle \text{arm} \rangle \ B \ | \ A$$

$$\langle \text{side} \rangle \quad ::= \quad B \ \langle \text{side} \rangle \ | \ \langle \text{side} \rangle \ B \ | \ B \ | \ D$$

$$\langle \text{submedian chromosome} \rangle \quad ::= \quad \langle \text{arm pair} \rangle \langle \text{arm pair} \rangle$$

A, B, and D are basic curve types. An obvious PDL syntax for the same example is:

$$\langle \text{arm} \rangle \longrightarrow (B + \langle \text{arm} \rangle) \,|\, (\langle \text{arm} \rangle + B) \,|\, A$$

$$\langle \text{side} \rangle \longrightarrow (B + \langle \text{side} \rangle) \,|\, (\langle \text{side} \rangle + B) \,|\, B \,|\, D$$

$$\langle \text{submedian chromosome} \rangle \longrightarrow ((\langle \text{arm pair} \rangle + \langle \text{arm pair} \rangle) * \lambda)$$

E. Flow Charts

None of the previous examples require the use of label designators and the / operator. Flow charts provide a good illustration of a practical picture class for which the set $\{+, -, \times, *, \sim\}$ is not adequate. A notational convenience is introduced for the examples of this section: Consider a PDL expression with standard form,

$$S = S\left(p_1^{\ell_1}, p_2^{\ell_2}, \ldots, p_n^{\ell_n}, p_{n+1}, p_{n+2}, \ldots, p_m\right),$$

where $p_i^{\ell_i}$, $1 \leq i \leq n$ are the labeled primitives of the form and p_i, $n < i \leq m$ are the primitives without labels. Then

$$\underline{S}^{\ell} \equiv S\left(p_1^{\ell_1 \ell}, p_2^{\ell_2 \ell}, \ldots, p_n^{\ell_n \ell}, p_{n+1}, p_{n+2}, \ldots, p_m\right);$$

that is, the underbar on a label means that all primitives already labeled in the standard form of the expression, and only those, are given the additional label, e.g., $(a^i + b)^{\underline{j}} \equiv (a^{ij} + b)$. This eliminates many redundant labels that would otherwise appear in the standard form of the PDL descriptions generated by the flow chart syntaxes.

The example illustrates how an algorithmic programming language can be defined by the syntax of its flow charts in conjunction with the syntax of its strings; the latter describes the allowable strings in the language while the former denotes flow of control. Figure 16 (a) contains samples of the primitives. The line segments with arrow heads leading from enter, fn, and cond may be any sequence of concatenated segments thus allowing the head of these primitives to be placed anywhere in a picture relative to the tail. The box in fn is a functional box and pred represents a predicate or test. cond may be either the true or false branch of the predicate; the initial blank part at its tail carries it to one of the vertices of the diamond of pred. The primitives can be further described in PDL as concatenations of line segments and circles, cond could be given a true or a false label, and character strings could be defined within the boxes. Figure 16 (b) contains a partial syntax for a simple ALGOL-like language. ASSIGNMENT statements, BLOCKHEAD (e.g., begin ⟨declaration list⟩), BLOCKTAIL (e.g., end), AE (⟨arithmetic expression⟩), BE (⟨Boolean expression⟩), and VARIABLE are not defined further since this would add nothing essential to the example. The various statement types are similar to a subset of ALGOL 60 (Naur 1963). GO TO statements are not included; they cannot be translated, from their syntax alone, into a flow chart.

With the exception of ASSIGN, INIT, INC, and TEST, there is a one-to-one correspondence between the elements of the flow chart syntax of Fig. 16 (c) and the language syntax; each component of the language translates into a flow chart component. Examples of the flow chart elements are given in Fig. 16 (d); unlabeled hatched areas may contain any diagram generated by STMNT.

The use of different label designators for S, BLOCK, STMNT, and BASIC ensure unique labels for each generation of TEST in the STEPUNTIL statement

and (ASSIGN + TEST) in the WHILE statement. For example, S could generate: (STMNT + (STMNT + (STMNT + STMNT$^{\underline{S}}_{-}$)$^{\underline{S}}_{-}$)$^{\underline{S}}_{-}$) which is equivalent to: (STMNT + (STMNT$^{\underline{S}}_{-}$ + (STMNT$^{\underline{S\,S}}_{--}$ + STMNT$^{\underline{S\,S\,S}}_{---}$))).

Several automatic flow chart generating programs have been written. Some of these require the programmer to insert detailed flow charting instructions or comments in his source code (e.g., Knuth 1963). Sherman (1966) describes the control syntax of a source language by a series of descriptors, which are then used to produce a general flow charting program for the language; however, Sherman is unable to handle languages with recursively-defined elements. The methods presented in the last example could serve as the basis for a general flow-charting program which does not have the above restrictions. Sutherland (1966) has designed and implemented a system for graphically specifying programs (on a computer-controlled display) and executing them; he uses an unconventional set of primitive elements for the flow charts and the computations. Flow charts could be drawn, syntactically analyzed, and executed within the PDL system to provide a more conventional and natural system of this type; a suitable set of semantic rules $\mathcal{S}_m$ would have to be designed along with the interactive components of the system. Finally, it might be simpler for a compiler to deal with the derived flow chart rather than the source program for generating efficient code.

It should be noted that the above applications are only educated predictions by this writer, since the details of such PDL flow chart generation and analysis systems have not been worked out.

F. Some Description Limitations of the PDL System

PDL is not a description panacea; the previous examples suggest its range of application. Further experimentation is necessary in order to precisely delimit the class of pictures for which useful descriptions may be obtained. At this stage, nevertheless, it is possible to enumerate some of its limitations and possible extensions.

The class of PDL descriptions, $\mathcal{L}(\mathcal{G})$, that may be generated from a context free grammar $\mathcal{G}$ is theoretically limited (Chomsky 1959 , Ginsburg 1966). Consider the description of an arbitrary "staircase" of "X" 's on a grid (Fig. 17 (a)). If the notation $\sum_{i=1}^{n} a$ represents

$$\underbrace{a + a + \ldots + a}_{n\ a's},$$

and the primitives h and v are those of Fig. 13, then the set

$$\{(\sum_{i=1}^{\ell}[\sum_{j=1}^{m} h + \sum_{k=1}^{n} v])\,|\,\ell, m, n \geq 1\},$$

where "[" and "]" indicate expression grouping, contains all possible PDL staircase descriptions (without redundant parentheses) with constant horizontal and vertical distances. This set, however, cannot be generated by a context-free grammar since this would imply that

$$\{((ab)^m(cd)^n)^\ell \mid m, n, \ell \geq 1\}$$

is a context-free language.

Concatenation of picture elements is the only explicit relation in PDL. The use of blank primitives allows many simple geometric relations among disjoint picture elements to be expressed. There are a great many other relations that one might like to see directly expressible in a picture language. For example, $\mathcal{P}((S_1 + (b + S_2))$, where S_1 and S_2 describe picture components and b is a blank primitive, might be the class of pictures such that the elements of some subset of $\mathcal{P}(S_1)$ is contained within those of a subset of $\mathcal{P}(S_2)$; alternatively, one might want to say that some elements of $\mathcal{P}(S_1)$ overlap those of $\mathcal{P}(S_2)$ (Fig. 17 (b)). In either case, the intended relation is difficult to express generally; if $\mathcal{P}(S_1)$ and $\mathcal{P}(S_2)$ are severely restricted and b defined appropriately, then $(S_1 + (b + S_2))$ might be satisfactory, but the more complex relation is not obvious. A related difficulty is that of relations depending on magnifications, rotations, and other transformations of pictures. In Fig. 17 (b), it might be desired to group the small triangle with the small square;

if a wide range of sizes and rotations of thes elements are possible, then a PDL description reflecting this size grouping cannot be found. A possible solution to this problem is to allow the definition of arbitrarily complex relations as blank and don't care primitives; this has a natural appeal in that edges of a graph often denote relations. Use could also be made of the preservation of the topological relations among picture components under a large class of transformations. For example, if the primitive description of a picture α is:

$$T_S(\alpha) = S(p_1, p_2, \ldots, p_n)$$
$$T_V(\alpha) = (D(\beta_1), D(\beta_2), \ldots, D(\beta_n)),$$

where

$$\beta_i \in \mathcal{P}(p_i), \qquad i = 1, \ldots, n$$

and A represents a magnification or rotation transformation, then

$$T_S(A\alpha) = S(q_1, q_2, \ldots, q_n)$$
$$T_V(A\alpha) = (D(A\beta_1), D(A\beta_2), \ldots, D(A\beta_n)),$$

where

$$A\beta_i \in \mathcal{P}(q_i) .$$

Each primitive is restricted to only two points of possible concatenation. There are many cases where more than two concatenation points appear to be necessary. Some of these can be treated in a natural way by a judicious choice of the tail and head. In Fig. 17 (c), the circles c and line segments ℓ are primitives; c has both its tail and head at the center of the circle. The multiple concatenation of the lines onto the central circle can be expressed by adjoining blank primitives b to each end of a line segment; then a description is:

$$P \longrightarrow (c + ((L + c) \times (L + c) \times (L + c) \times (L + c)))$$
$$L \longrightarrow (b + \ell + b)$$

These suggestions are left for future work. The important points are:

1. A large, interesting, and useful class of pictures can be described in a simple and natural manner within the PDL system.
2. The system is capable of extension without destroying its basic simplicity.

VIII. PICTURE PROCESSING APPLICATIONS

The author's thesis (Shaw 1968) presents a general algorithm for parsing (analyzing) pictures based on this description scheme. A top-down goal-oriented picture parser accepts (1) a primitive recognizer or pattern recognition routine for each primitive class and (2) a grammar, and it uses the latter to direct the analysis of pictures; a successful analysis of a picture α yields its description $D(\alpha)$ as output. The analyzer has been implemented and applied to a digitized sample of spark chamber film. Two experimental interactive graphics systems have been developed which employ the PDL language to describe pictures (Noyelle 1967, and George 1967, George and Miller 1968). These systems allow a user at a CRT console equipped with a light pen to draw and manipulate pictures via PDL descriptions. Experience with these analysis and generation systems has demonstrated the potential usefulness of this approach to picture processing.

PDL is being used as the description notation for a picture calculus which is currently in development (Miller and Shaw 1967). This calculus is comprised of the picture description language PDL, formal rules for transforming and comparing pictures, data structures and control for generation of pictures, and the parsing and primitive recognizers needed for picture recognition.

IX. CONCLUSIONS

The description scheme has the following properties:

1. It is capable of describing, both to humans and to computers, a large and interesting class of pictures.
2. The notation is simple and natural.
3. Descriptions are complete. They contain both the syntax (structure) and semantics (meaning) of pictures in a form that allows a reasonable facsimile of a picture to be generated.
4. Descriptions may be used to drive picture processing systems.
5. Descriptions are (almost) independent of any digitized representation of a picture; that is, a description would generally remain invariant over changes in the number of levels of digitization of a picture and the coordinate system.
6. The language is applicable to pictures in n-dimensions, for $n \geq 1$.

Current plans are to investigate the descriptive ability of PDL in other application areas, to continue the development of its formal properties as a part of the picture calculus, and to gain more experience on picture analysis and synthesis systems based on the scheme.

ACKNOWLEDGEMENT

The author is deeply indebted to his thesis advisor, Professor William F. Miller, for many useful discussions, for his constant encouragement and enthusiasm in this work, and for his constructive reading of this paper.

REFERENCES

Anderson, A (1968), Syntax-directed recognition of hand-printed two-dimensional mathematics. Ph.d. dissertation, Applied Mathematics, Harvard University.

Chomsky, N. (1959), On certain formal properties of grammars. Inf. Control 2, 137-167.

Clowes, M. (1967a), Perception, picture processing, and computers. Machine Intelligence 1, Collins and Michie (Eds.), Oliver and Boyd, London, 1967.

Clowes, M. (1967b), A generative picture grammar. Seminar Paper No. 6, Computing Research Section, Commonwealth Scientific and Industrial Research Organization, Melbourne, Australia.

Eden, M. (1961), On the formalization of handwriting. Am. Math. Soc. Appl. Math. Symp. 12, 83-88.

Eden M. (1962), Handwriting and pattern recognition. IRE Trans. Inform. Theory IT-8, 160-166.

Feldman, J. and Gries, D. (1968), Translator writing systems. Commun. Assoc. Comput. Mach. 11, 77-113.

George, J. (1967), Picture generation based on the picture calculus. GSG Memo 50, Computation Group, Stanford Linear Accelerator Center (internal report).

George, J. and Miller, W. (1968), String descriptions of data for display. SLAC-PUB-383, Stanford Linear Accelerator Center. Presented at 9th Annual Symposium of the Society for Information Display.

Ginsburg, S. (1966), The Mathematical Theory of Context-Free Languages. McGraw Hill, New York, 1966.

Kirsch, R. (1964), Computer interpretation of English text and picture patterns. IEEE Trans. Electronic Computers EC-13, 363-376.

Knuth, D. (1963), Computer-drawn flowcharts. Commun. Assoc. Comput. Mach. 6, 555-563.

Ledley, R. , Rotolo, L. , Golab, T., Jacobsen, J. , Ginsberg, M. , and Wilson, J. (1965), FIDAC: film input to digital automatic computer and associated syntax-directed pattern recognition programming system. Optical and Electro-Optical Information Processing, Tippet, J., Beckowitz, D., Clapp, L. , Koester, C. , and Vanderburgh, Jr. , A. (Eds.), M. I. T. Press, Cambridge, 1965.

Lipkin, L. , Watt, W. , and Kirsch, R. (1966), The analysis, synthesis, and description of biological images. Annals of the New York Academy of Sciences 128, 984-1012.

Miller, W. and Shaw, A. (1967), A picture calculus. SLAC-PUB-358, Stanford Linear Accelerator Center. Presented at the conference on Emerging Concepts in Computer Graphics, University of Illinois (to be published).

Narasimhan, R. (1962), A linguistic approach to pattern recognition. Digital Computer Laboratory Report No. 121, University of Illinois.

Narasimhan, R. (1964), Labeling schemata and syntactic description of pictures. Inf. Control 7, 151-179.

Narasimhan, R. (1966), Syntax-directed interpretation of classes of pictures. Commun. Assoc. Comput. Mach. 9, 166-173.

Naur, P. (Ed.) (1963), Revised report on the algorithmic language ALGOL 60. Commun. Assoc. Comput. Mach. 6, 1-17.

Noyelle, Y. (1967), Implementation on the PDP-1 of a subset of the picture calculus. Term project for CS260, Computer Science Department, Stanford University (internal report).

Shaw, A. (1968), The formal description and parsing of pictures. Computer Science Report No. 94, Computer Science Department, Stanford University (Ph. D. Thesis).

Sherman, P. (1966), FLOWTRACE, a computer program for flowcharting programs. Commun. Assoc. Comput. Mach. 9, 845-854.

Shutt, R. (Ed.) (1967), Bubble and Spark Chambers, Vol. II. Academic Press, New York, 1967.

Sutherland, W. (1966), On-line graphical specification of computer procedures. Technical Report No. 405. Lincoln Laboratory, M.I.T.

FIGURE TITLES

Fig. No.	Descriptive Legend
1.	Hierarchic description of a picture.
2.	Representation of a picture primitive.
3.	An invisible primitive.
4.	An extreme case of a connected picture.
	4(a) Labeled picture
	4(b) Corresponding graph
5.	Graphs of binary concatenations.
6.	Primitive structural descriptions of an "A" and an "F".
7.	Local completeness of $\{+, \times, -, *\}$.
8.	PDL descriptions with labels and / .
	8(a) The complete 4-node graph with directed edges
	8(b) A 3-dimensional cube
9.	The graph of a vPDL.
10.	Theorem 2
	10(a) head (p_{n+1}) cat head (p_i)
	10(b) head (p_{n+1}) cat head $(p_i) \wedge$ tail (p_{n+1}) cat head (p_j)
11.	Structure descriptions of a picture.
	11(a) $\mathcal{G}, \mathcal{L}(\mathcal{G})$, and primitives
	11(b) Examples and parse of a "P"
12.	Particle physics example.
	12(a) Sample picture
	12(b) Primitives
	12(c) Syntax $\mathcal{G}$

Fig. No.	Descriptive Legend
13.	Right-Angled 45^{o} triangle of Kirsch.
	13(a) Primitives
	13(b) $\mathcal{G}$: Right-Angled Triangle
	13(c) Examples
14.	Simple English block characters.
	14(a) Primitives
	14(b) Examples
	14(c) Primitive structural descriptions
15.	A page of English text
	15(a) Sample
	15(b) Primitives
	15(c) $\mathcal{G}$: Syntax for a page of text
16.	String and flow chart syntax for a small algorithmic language.
	16(a) Primitives
	16(b) Small language string syntax
	16(c) Small language flow chart syntax
	16(d) Examples
17.	Some description limitations.
	17(a) Staircase of "X"'s
	17(b) Complex relations among figures
	17(c) More than two concatenation points on a primitive

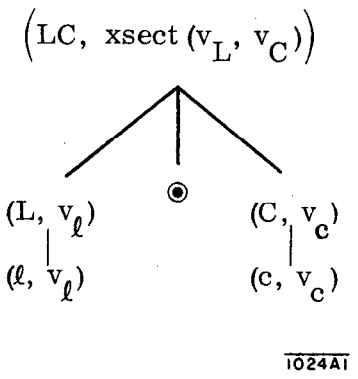


Fig. 1

Information and Control
Alan C. Shaw
Fig. 1

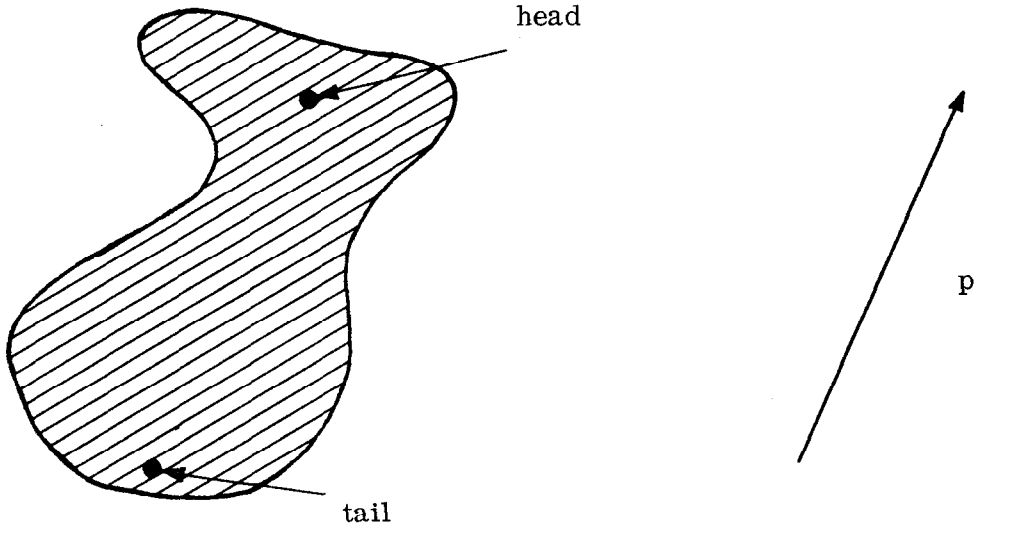


Fig. 2

Information and Control
Alan C. Shaw
Fig. 2

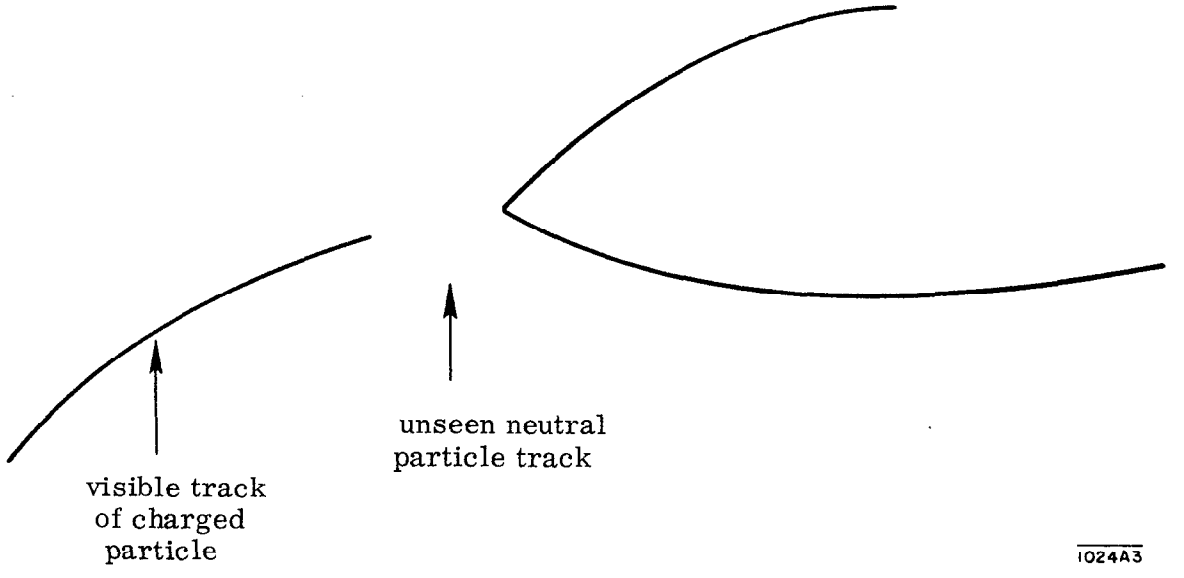


Fig. 3

Information and Control
Alan C. Shaw
Fig. 3

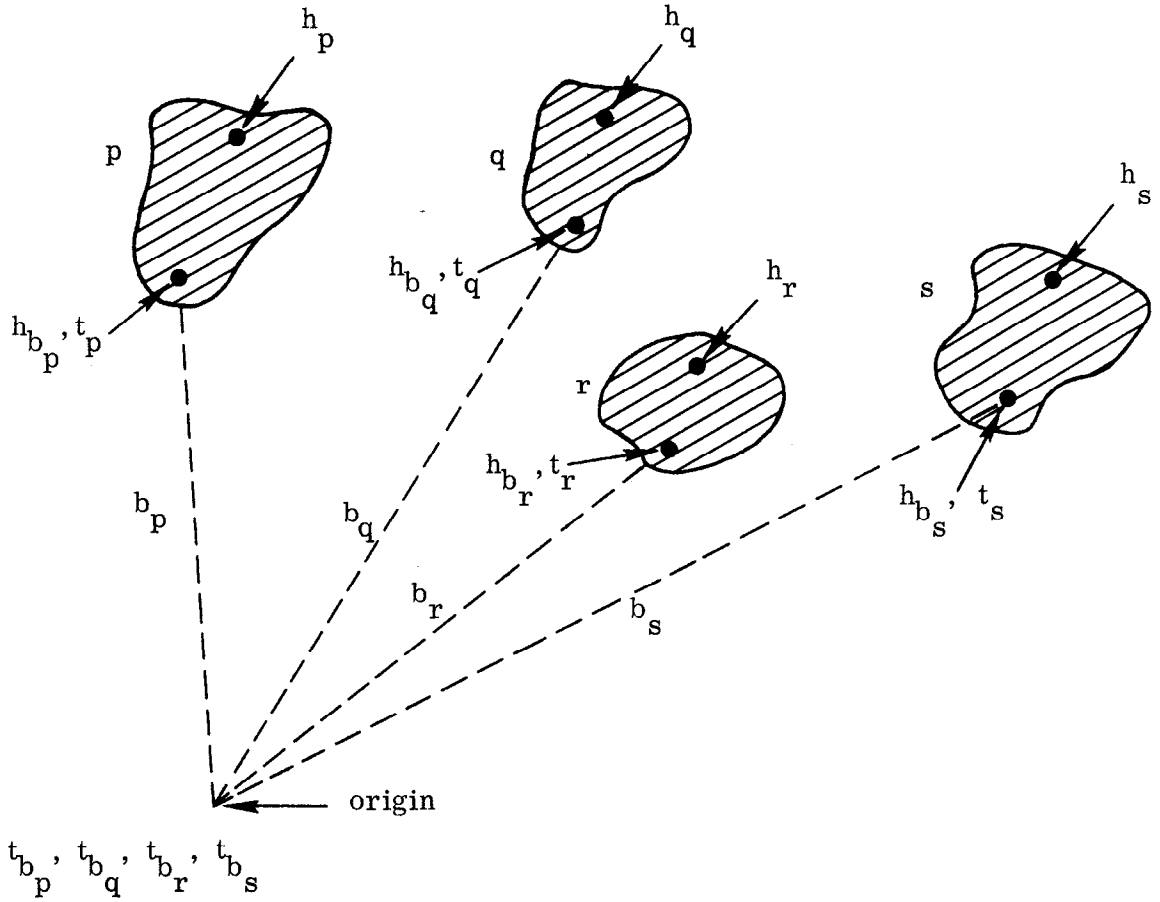
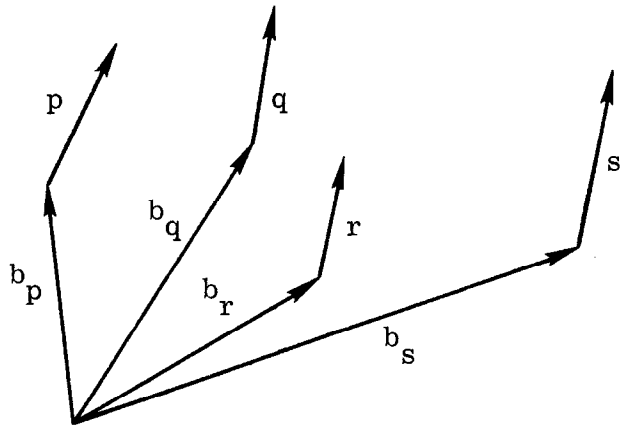


Fig. 4 (a)



1024A4

Fig. 4 (b)

Information and Control
Alan C. Shaw
Fig. 4 (a), (b)

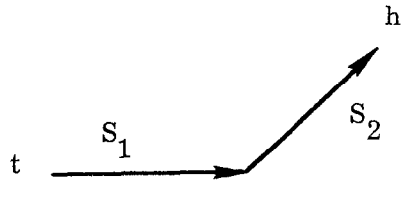
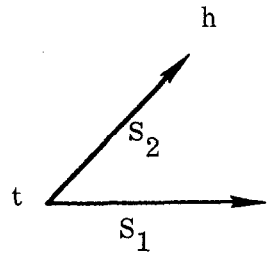
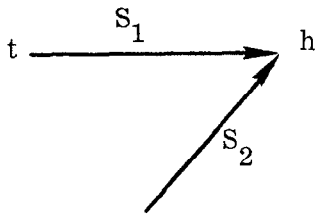
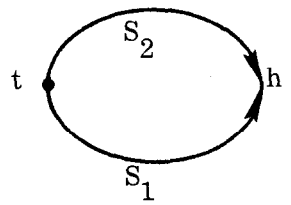
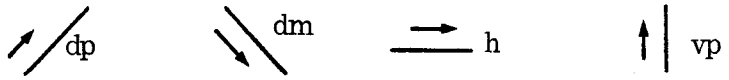
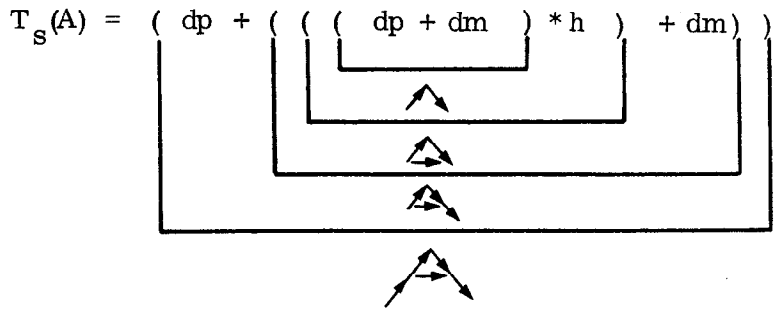
$T_s(\alpha)$	Graph
$(S_1 + S_2)$	
$(S_1 \times S_2)$	
$(S_1 - S_2)$	
$(S_1 * S_2)$	

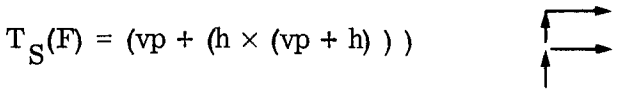
Fig. 5

Information and Control
Alan C. Shaw
Fig. 5



Primitive Classes

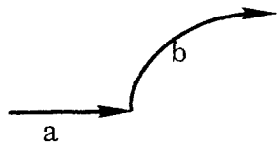
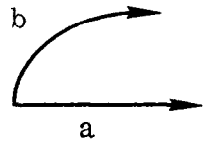
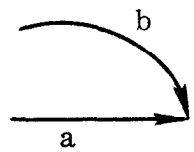
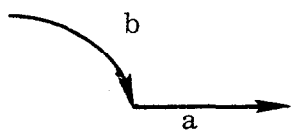
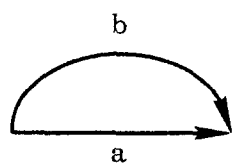
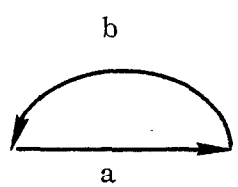




1024A6

Fig. 6

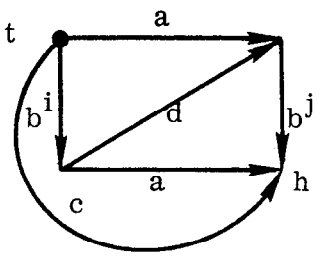
Information and Control
Alan C. Shaw
Fig. 6

Concatenation	Description
	(a + b)
	(a × b)
	(a - b)
	(b + a)
	(a * b)
	((a + b) * λ)

1024A7

Fig. 7

Information and Control
Alan C. Shaw
Fig. 7



$$T_s(\alpha) = (((b^i + a) * (((/b^i) + d) + (/b^j))) * ((a + b^j) * c))$$

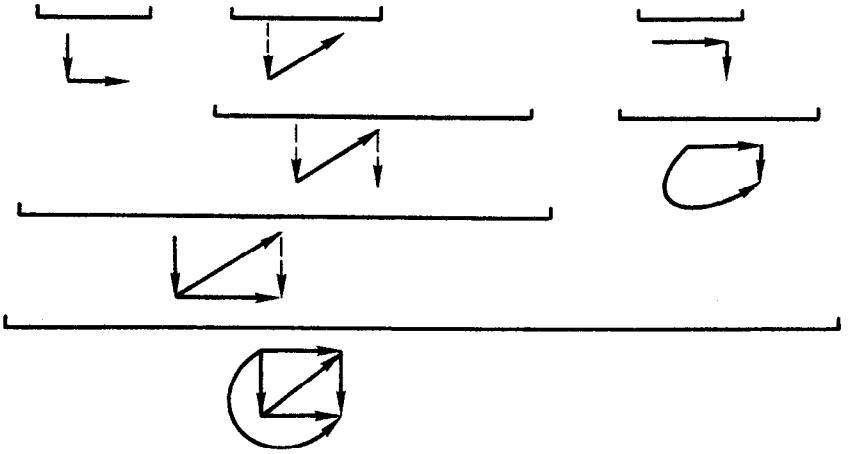
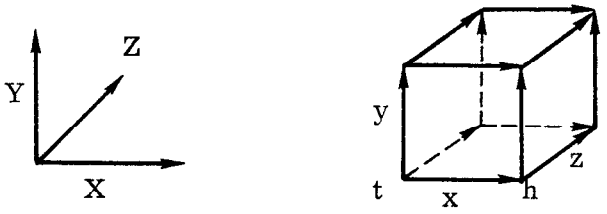


Fig. 8 (a)



$$T_s(\alpha) = (((x * ((y^i + x) + (\sim y^j)))$$
$$* ((((/y^i) + z) + ((x * ((\sim y) + (x^k + y)))$$
$$+ (\sim z))) + (\sim (/y^j))))$$
$$* ((z + (/x^k)) + (\sim z)))$$

1024A8

Fig. 8 (b)

Information and Control
Alan C. Shaw
Fig. 8 (a), (b)

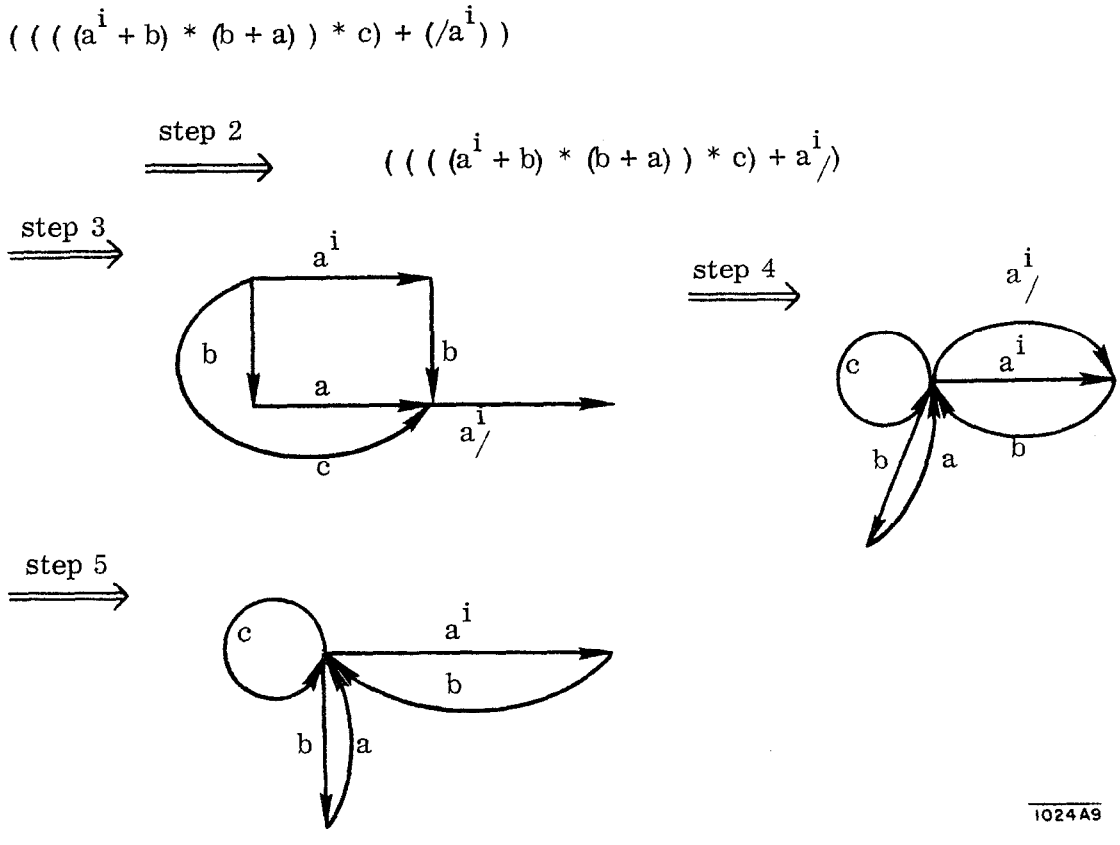


Fig. 9

Information and Control
Alan C. Shaw
Fig. 9

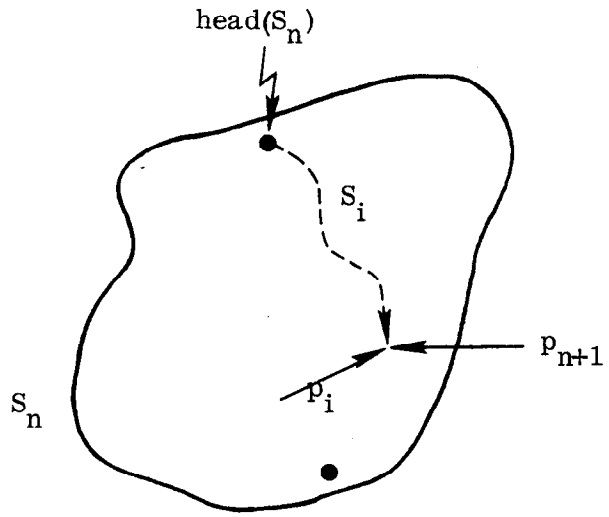


Fig. 10 (a)

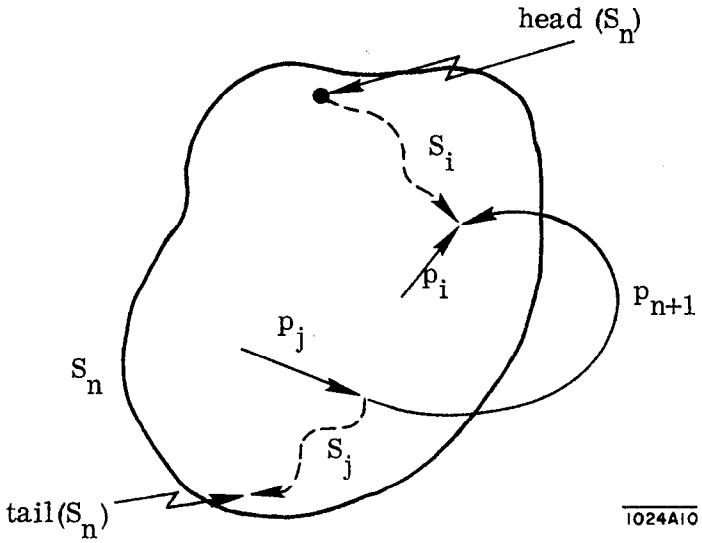


Fig. 10 (b)

Information and Control
Alan C. Shaw
Fig. 10 (a), (b)

$\mathcal{G}$:

$P \rightarrow A \mid HOUSE$

$A \rightarrow (dp + (TRIANGLE + dm))$

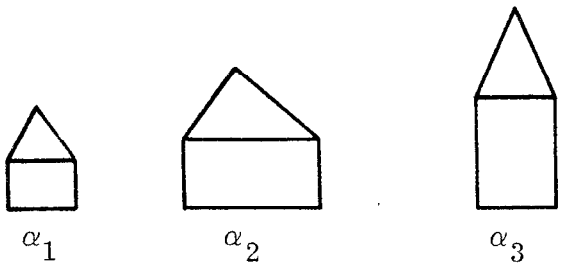
$HOUSE \rightarrow ((vm + (h + (\sim vm))) * TRIANGLE)$

$TRIANGLE \rightarrow ((dp + dm) * h)$

$\mathcal{L}(\mathcal{G}) = \{(dp + (((dp + dm) * h) + dm)),$

$((vm + (h + (\sim vm))) * ((dp + dm * h))\}$

Fig. 11 (a)



$T_s(\alpha_i) = ((vm + (h + (\sim vm))) * ((dp + dm) * h))$

$H_s(\alpha_i)$:

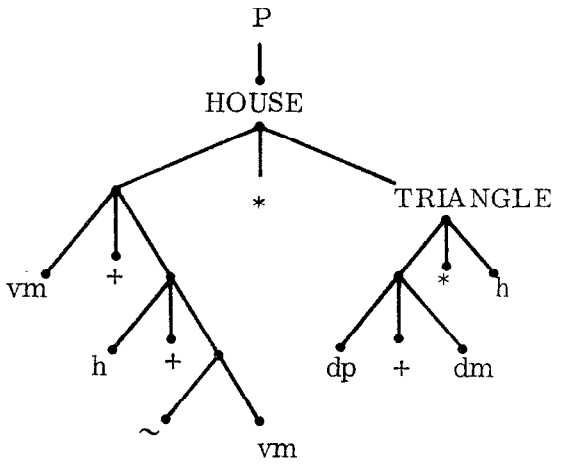


Fig. 11 (b)

1024A11

Information and Control
Alan C. Shaw
Fig. 11 (a), (b)

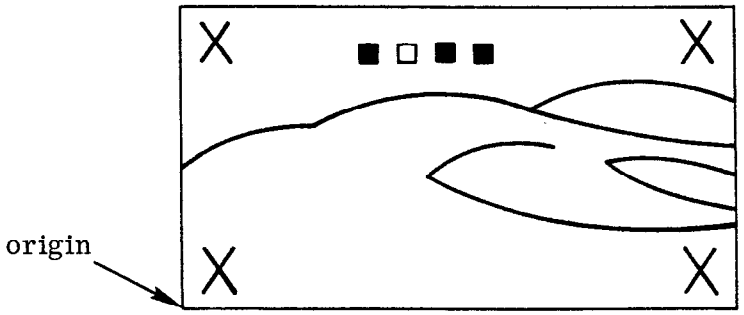


Fig. 12 (a)

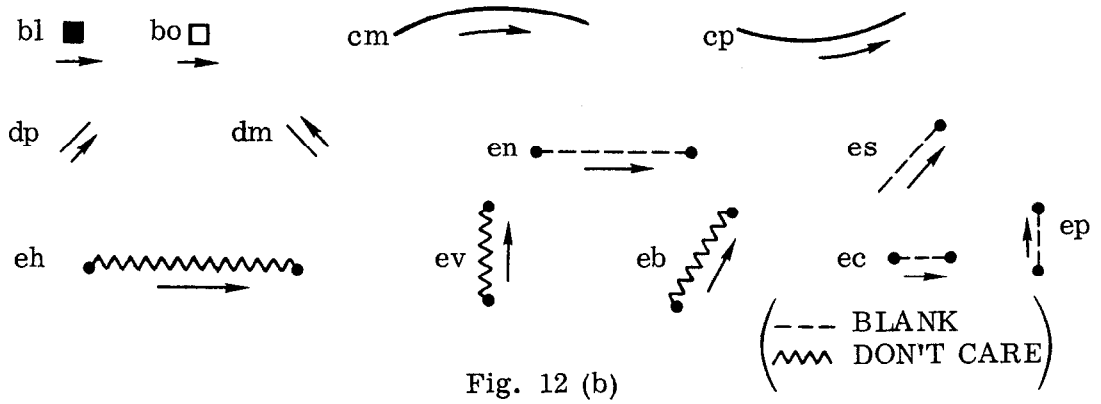


Fig. 12 (b)

PICTURE → (es + (FI + (FID × (ID × PT))))

FI → (dp + (dm × (dp × (λ - dm))))

FID → ((eh + X) + ((ev + X)) - (X + eh)))

ID → ((eb + B) + ((ec + B) + ((ec + B) + (ec + B))))

PT → (ep + TM)

X → ((dp × dm) × ((~ dp) × (λ - dm)))

B → bo | bl

TM → (cm + MD) | (cm + MP) | cm

MP → (P + ((TM × TP) × TN)) | (P + (TM × TP)) | (P + TN)

MD → (TM × TN) | TM

TP → (cp + PD) | cp

TN → (en + (N + (TM × TP)))

PD → (TP × TN) | TP

P → λ

N → λ

Fig. 12 (c)

Information and Control
Alan C. Shaw
Fig. 12 (a), (b), (c)

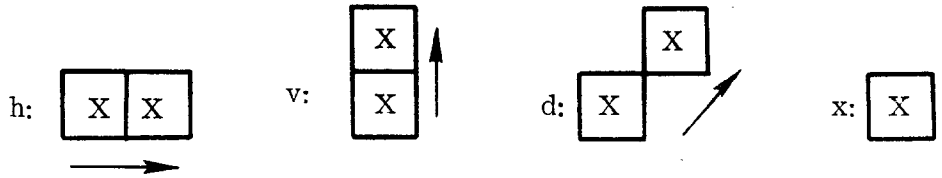


Fig. 13 (a)

RAT → (((V + h) + (IRAT + (v + W))) * DH)
IRAT → (((B + h) + (IRAT + (v + L))) * DI) | (x + R)
DH → ((d + H) + DH) | d
DI → ((d + I) + DI) | d
V → λ
W → λ
R → λ
H → λ
B → λ
L → λ
I → λ

Fig. 13 (b)

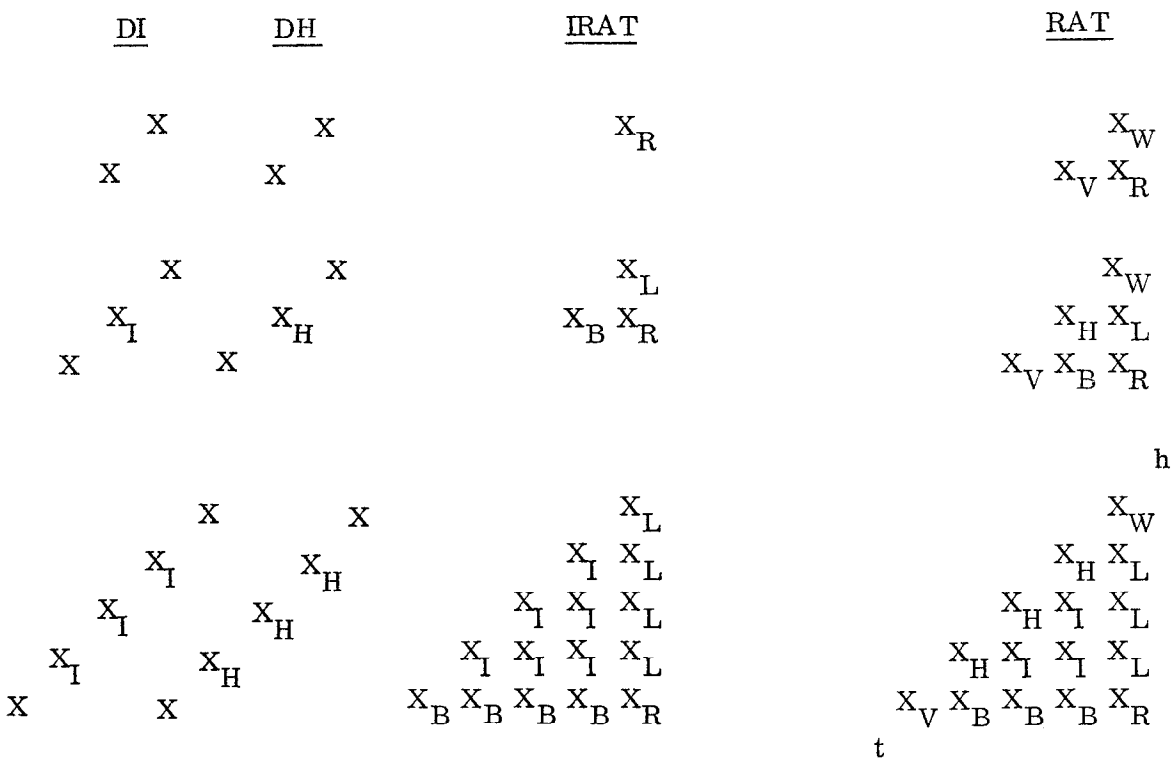


Fig. 13 (c)

Information and Control
Alan C. Shaw
Fig. 13 (a), (b), (c)

1024A13

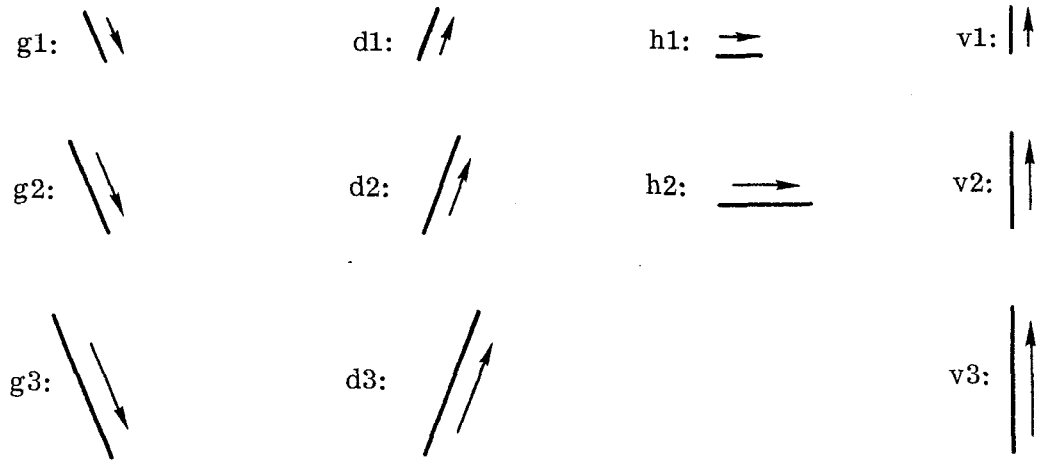


Fig. 14 (a)

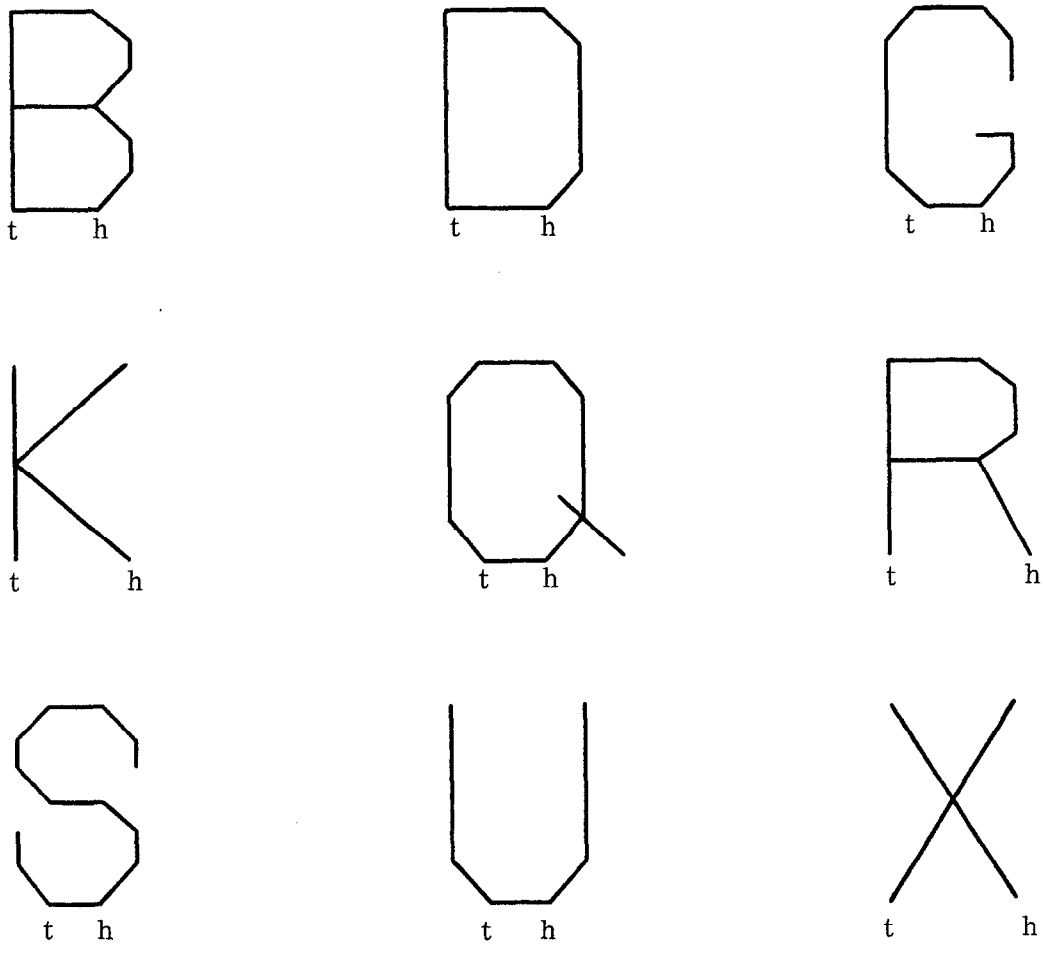


Fig. 14 (b)

1024A14

Information and Control
Alan C. Shaw
Fig. 14 (a), (b)

A → (d2 + ((d2 + g2) * h2) + g2)

B → ((v2 + ((v2 + h2 + g1 + (~ (d1 + v1))) * h2) + g1 + (~(d1 + v1))) * h2)

C → (((~g1) + v2 + d1 + h1 + g1 + (~v1)) × (h1 + ((d1 + v1) × λ)))

D → (h2 * (v3 + h2 + g1 + (~(d1 + v2))))

E → ((v2 + ((v2 + h2) × h1)) × h2)

F → ((v2 + ((v2 + h2) × h1)) × λ)

G → (((~ g1) + v2 + d1 + h1 + g1 + (~v1)) × (h1 + (((d1 + v1 - h1) × λ)))

H → (v2 + (v2 × (h2 + (v2 × (~v2)))))

I → (v3 × λ)

J → ((((~ g1) + v1)× h1) + ((d1 + v3) × λ))

K → (v2 + (v2 × d2 × g2))

L → (v3 × h2)

M → (v3 + g3 + d3 + (~ v3))

N → (v3 + g3 + (v3 × λ))

O → (h1 * ((~g1) + v2 + d1 + h1 + g1 + (~(d1 + v2))))

P → ((v2 + ((v2 + h2 + g1 + (~(d1 + v1))) * h2)) × λ)

Q → (h1 * ((~ g1) + v2 + d1 + h1 + g1 + (~(d1 + ((~ g1) × g1) + v2))))

R → (v2 + (h2 * (v2 + h2 + g1 + (~ (d1 + v1))) + g2)

S → ((((~ g1) + v1) × h1) + ((d1 + v1 + (~ (g1 + h1 + g1))
+ v1 + d1 + h1 + g1 + (~ v1)) × λ))

T → ((v3 + (h1 × (~h1))) × λ)

U → ((((~ g1) + v3) × h1) + ((d1 + v3) × λ))

V → ((~ g3) × d3 × λ)

W → (((~ g3) + d3 + g3) + (d3 × λ))

X → (d2 + ((~ g2) × d2 × g2))

Y → ((v2 + ((~ g2) × d2)) × λ)

Z → ((d3 - h2) × h2)

Fig. 14 (c)

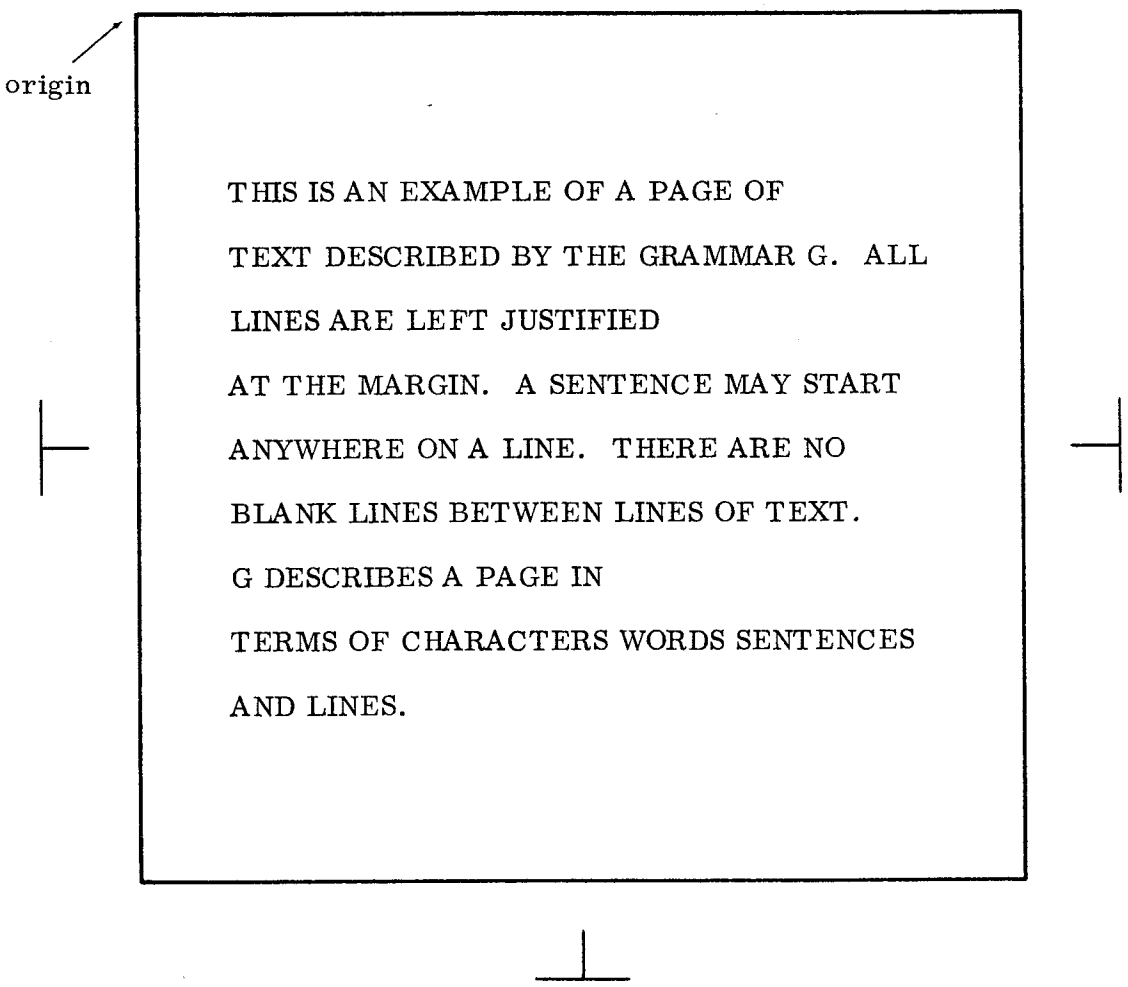

origin
THIS IS AN EXAMPLE OF A PAGE OF
TEXT DESCRIBED BY THE GRAMMAR G. ALL
LINES ARE LEFT JUSTIFIED
AT THE MARGIN. A SENTENCE MAY START
ANYWHERE ON A LINE. THERE ARE NO
BLANK LINES BETWEEN LINES OF TEXT.
G DESCRIBES A PAGE IN
TERMS OF CHARACTERS WORDS SENTENCES
AND LINES.

Fig. 15 (a)

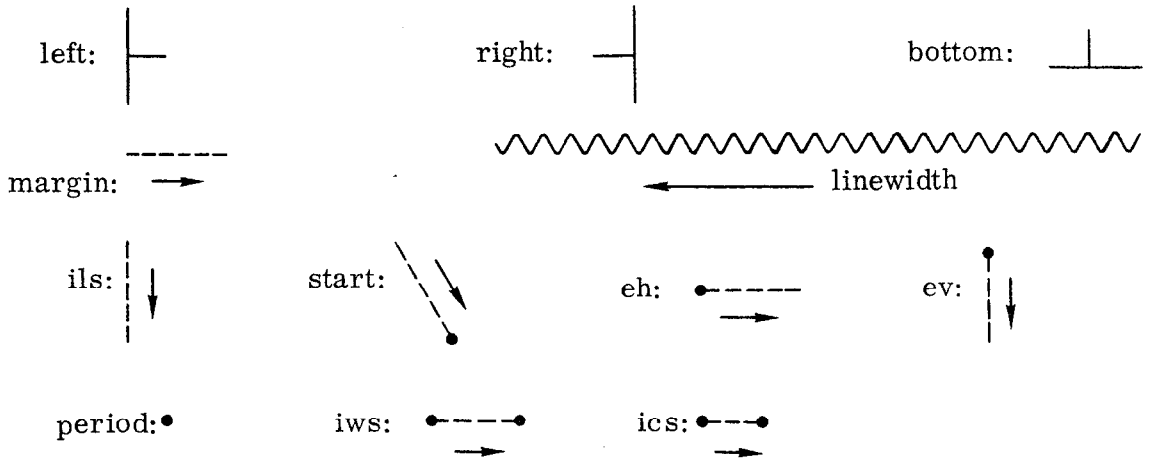


Fig. 15 (b)

PAGE → (start + (S + EOP))

S → SENT | (S + SENT)

SENT → (BEGINSENT + (L + (ics + period)))

BEGINSENT → iws | (EOL + ((linewidth + left) + (ils + margin))) | λ

L → LINE | (L × ((ils + margin) + LINE))

LINE → WORD | (LINE + (iws + WORD))

WORD → CHAR | (WORD + (ics + CHAR))

CHAR → A | B | C . . . . . | Y | Z

EOP → (ev + bottom)

EOL → (eh + right)

Fig. 15 (c)

1024A15

Information and Control
Alan C. Shaw
Fig. 15 (b), (c)

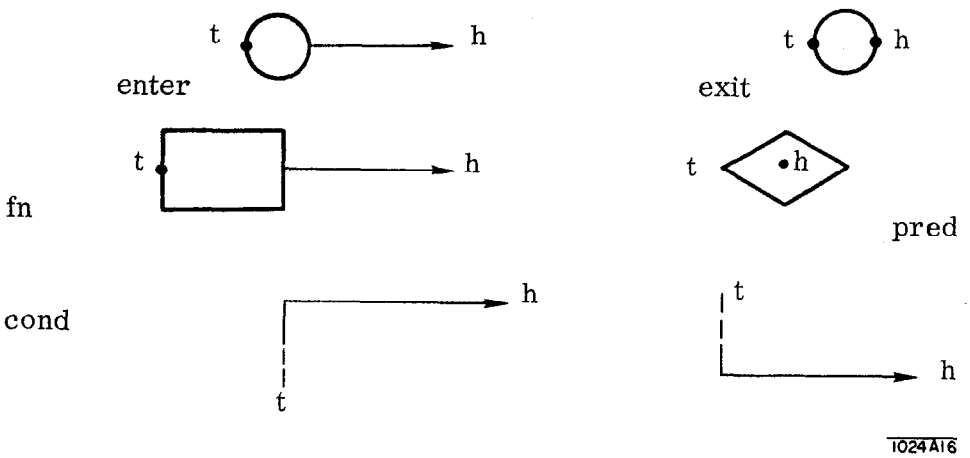


Fig. 16 (a)

Information and Control
Alan C. Shaw
Fig. 16 (a)

PROGRAM	→	BLOCK
BLOCK	→	BLOCKHEAD;STATEMENTLIST BLOCKTAIL
STATEMENTLIST	→	STATEMENT | STATEMENT; STATEMENTLIST
STATEMENT	→	BASIC | CONDITIONAL
BASIC	→	ASSIGNMENT | FOR | BLOCK
CONDITIONAL	→	IFTHEN | IFTHENELSE
FOR	→	STEPUNTIL | WHILE
STEPUNTIL	→	for VARIABLE := AE step AE until AE do STATEMENT
WHILE	→	for VARIABLE := AE while BE do STATEMENT
IFTHEN	→	if BE then BASIC
IFTHENELSE	→	if BE then BASIC else STATEMENT

Fig. 16 (b)

PROGRAM	→	BLOCK
BLOCK	→	(entry + (S + exit))
S	→	STMNT | (STMNT + $S^{\underline{S}}$)
STMNT	→	BASIC | CNDTNL
BASIC	→	ASSIGN | FOR | $BLOCK^{\underline{b}}$
CNDTNL	→	IFTHEN | IFTHENELSE
ASSIGN	→	fn
FOR	→	STEPUNTIL | WHILE
STEPUNTIL	→	(INIT + ((((TEST ^{su} + cond) + $STMNT^{\underline{su}}$) * (~ INC)) × ((/TEST ^{su}) + cond)))
WHILE	→	((((ASSIGN + TEST) ^{w} + cond) * (~ $STMNT^{\underline{w}}$)) × ((/(ASSIGN + TEST) ^{w}) + cond))
IFTHEN	→	(pred + ((cond + $BASIC^{\underline{i}}$) * cond))
IFTHENELSE	→	(pred + ((cond + $BASIC^{\underline{it1}}$) * (cond + $STMNT^{\underline{it2}}$)))
INIT	→	fn
INC	→	fn
TEST	→	pred

Fig. 16 (c)

Information and Control
Alan C. Shaw
Fig. 16 (b), (c)

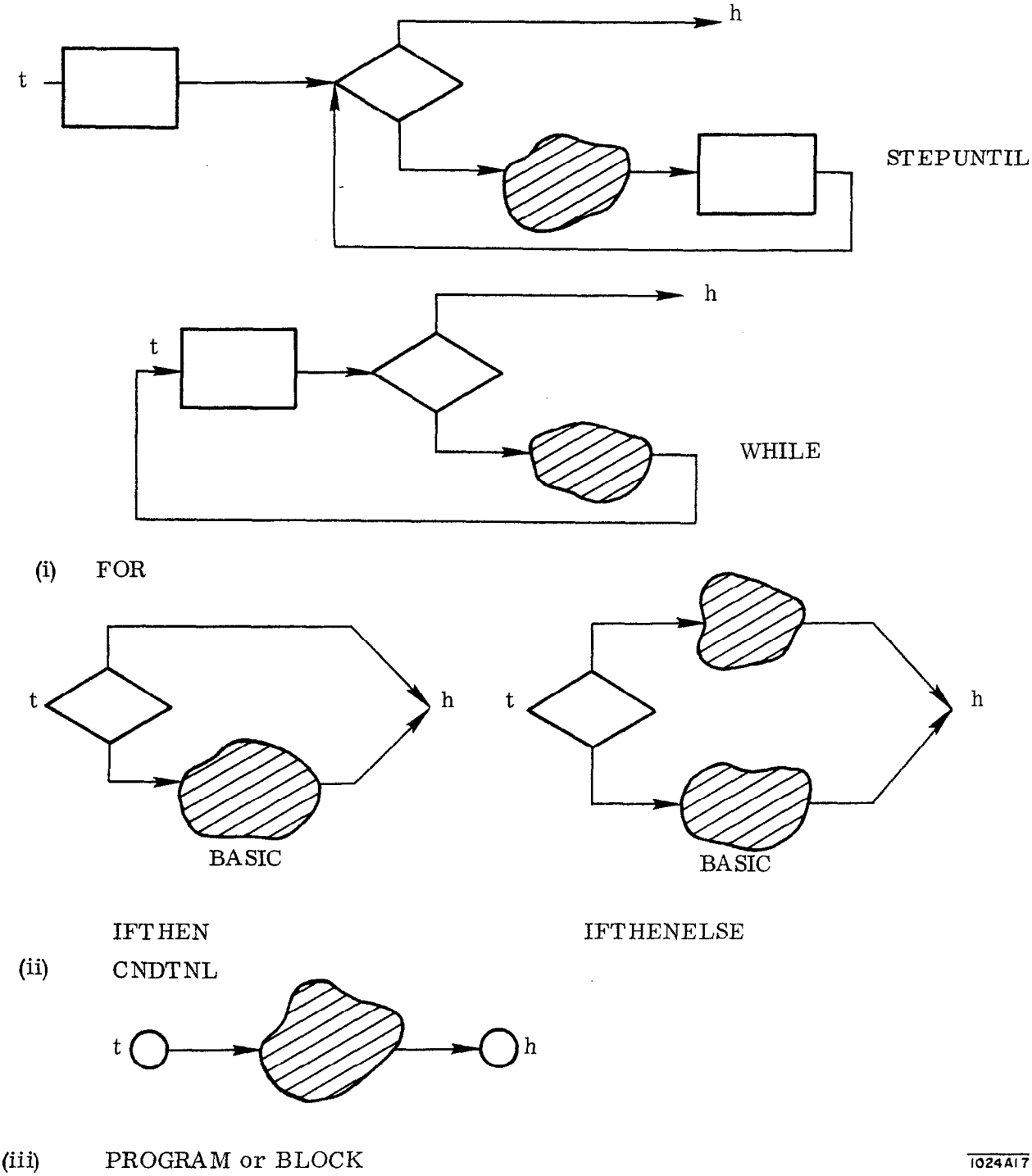
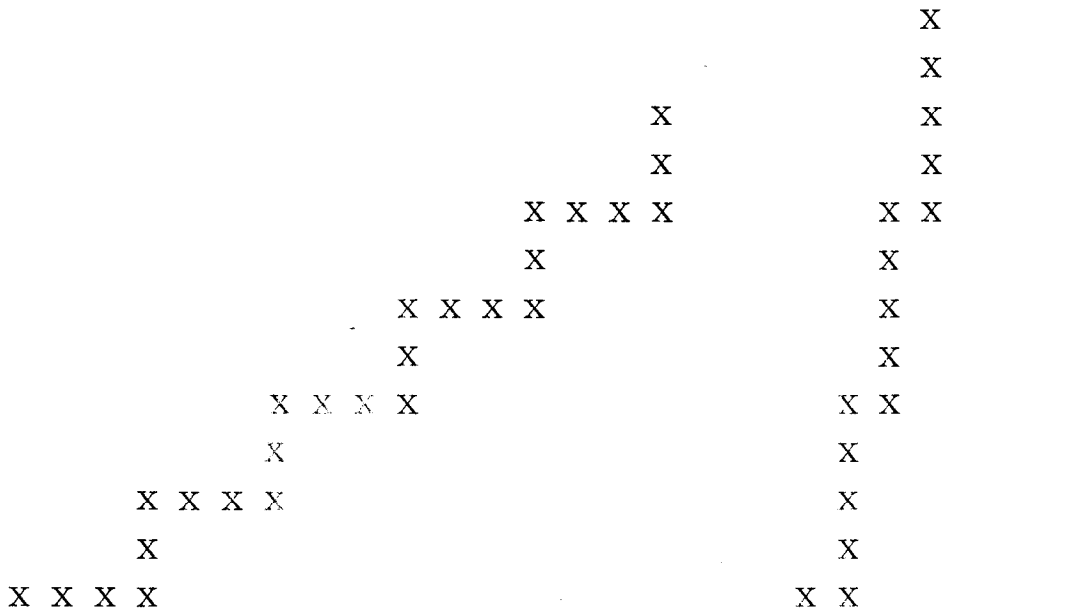


Fig. 16 (d)

Information and Control
Alan C. Shaw
Fig. 16 (d)



$$\left(\sum_{i=1}^{5}\left[\sum_{j=1}^{3} h + \sum_{k=1}^{2} v\right]\right) \qquad \left(\sum_{i=1}^{3}\left[\sum_{j=1}^{1} h + \sum_{k=1}^{4} v\right]\right)$$

Fig. 17 (a)

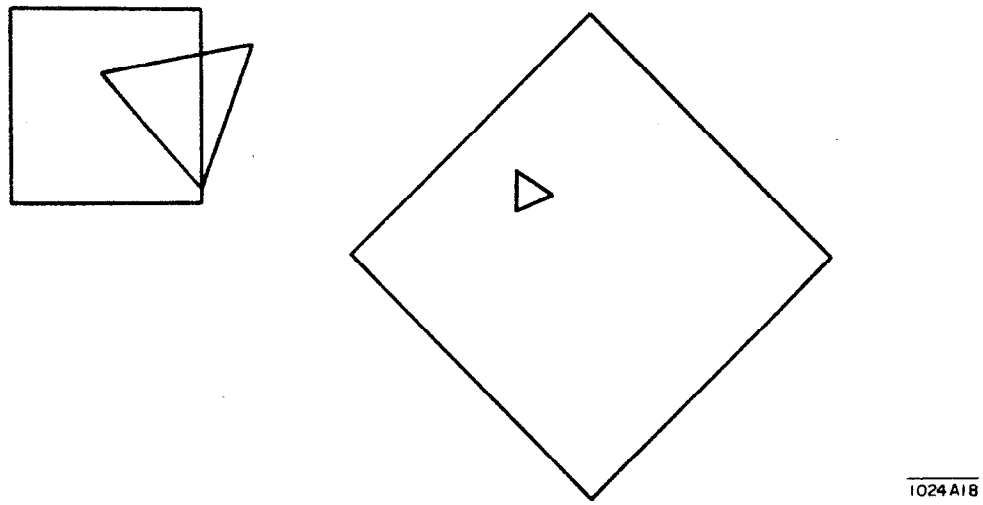


Fig. 17 (b)

Information and Control
Alan C. Shaw
Fig. 17 (a), (b)

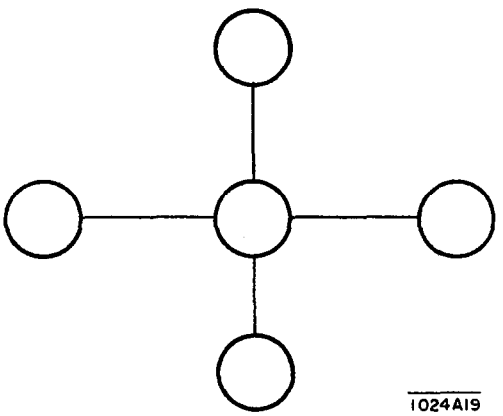


Fig. 17 (c)

Information and Control
Alan C. Shaw
Fig. 17 (c)