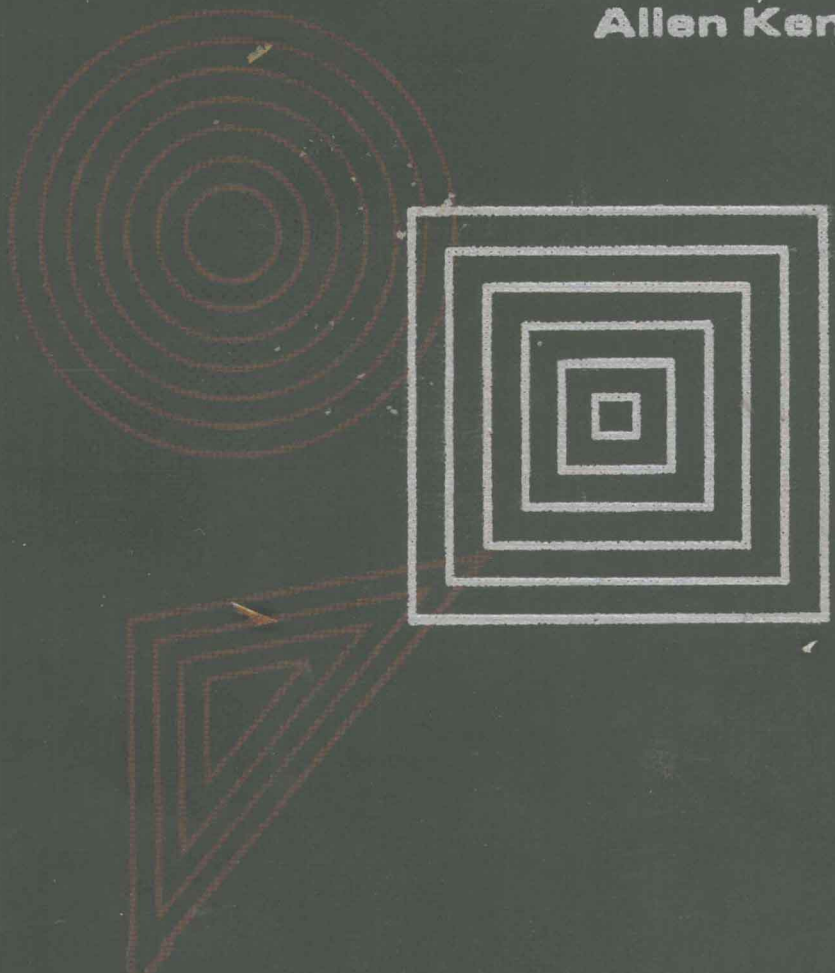
ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY

12

EXECUTIVE EDITORS

Jack Belzer
Albert G. Holzman
Allen Kent



ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY

EXECUTIVE EDITORS

Jack Belzer *Albert G. Holzman* *Allen Kent*

UNIVERSITY OF PITTSBURGH
PITTSBURGH, PENNSYLVANIA

VOLUME 12

Pattern Recognition to Reliability of Computer Systems

MARCEL DEKKER, INC. • NEW YORK and BASEL

MARCEL DEKKER, INC.
270 Madison Avenue, New York, New York 10016

LIBRARY OF CONGRESS CATALOG CARD NUMBER: 74-29436
ISBN: 0-8247-2262-0

Current printing (last digit):
10 9 8 7 6 5 4 3 2

PRINTED IN THE UNITED STATES OF AMERICA

ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY

VOLUME 12

INTERNATIONAL EDITORIAL ADVISORY BOARD

CONTRIBUTORS TO VOLUME 12

PAUL W. ABRAHAMS, Courant Institute of Mathematical Sciences, New York University, New York, New York: *PL/I*

G. M. BAKEN, Institute of Cybernetics, Ukrainian Academy of Sciences, Kiev, USSR: *Random Search*

RANAN BANERJI, Department of Computer Science, Temple University, Philadelphia, Pennsylvania: *Pattern Recognition: Structural Description Languages*

HERBERT BARRY III, Department of Pharmacology, University of Pittsburgh School of Pharmacy, Pittsburgh, Pennsylvania: *Pharmacology*

RAYNOR DUNCOMBE, Department of Aerospace Engineering, University of Texas, Austin, Texas: *Punched Card Data Processing*

ROBERT M. FANO, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts: *Project MAC*

ROBERT J. FITZHUGH, Learning and Research Development Center, University of Pittsburgh, Pittsburgh, Pennsylvania and The Pacific Systems Group, Eugene, Oregon: *Planning Instruction and Monitoring Classroom Processes*

K. S. FU, School of Electrical Engineering, Purdue University, West Lafayette, Indiana: *Pattern Recognition: Discriminant and Syntactic Methods*

LYNWOOD A. JOHNSON, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia: *Production and Inventory Control*

OLIN JOHNSON, Department of Computer Science, University of Houston, Houston, Texas: *Petroleum Industry*

MANFRED W. KRACHT, Institute of Mathematics, University of Duesseldorf, Duesseldorf, Germany: *Probability*

ERWIN KREYSZIG, Department of Mathematics, University of Windsor, Windsor, Canada: *Probability*

ROBERT S. LEDLEY, Department of Physiology and Biophysics, Georgetown University Medical Center, Washington, D.C.: *Radiology in Medicine*

A. A. LETICHEVSKIY, Institute of Cybernetics, Ukrainian Academy of Sciences, Kiev, USSR: *Push-Down Automata*

DOUGLAS C. MONTGOMERY, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia: *Production and Inventory Control*

TONY MORELOCK, Pipeline Transmission, Texas Eastern Transmission, Houston, Texas: *Petroleum Industry*

DONALD PEACEMAN, Reservoir Modeling, Exxon Production Research, Houston, Texas: *Petroleum Industry*

DAVID POPE, Credit Card Center, Shell Oil Company, Tulsa, Oklahoma: *Petroleum Industry*

AZRIEL ROSENFELD, Computer Science Center, University of Maryland, College Park, Maryland: *Picture Processing*

V. A. SPOSITO, Department of Statistics, Iowa State University, Ames, Iowa: *Quadratic Programming*

STEPHEN SZYGENDA, Electrical Engineering Department, The University of Texas, Austin, Texas: *Reliability of Computer Systems*

MARGARET C. WANG, Learning and Research Development Center, University of Pittsburgh, Pittsburgh, Pennsylvania: *Planning Instruction and Monitoring Classroom Processes*

THEODORE J. WILLIAMS, Purdue Laboratory for Applied Industrial Control, Purdue University, West Lafayette, Indiana: *Process Control*

ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY

VOLUME 12

CONTENTS OF VOLUME 12

PATENTS OF COMPUTER PROGRAMS;

see Copyright and Computers

PATTERN RECOGNITION

Structural Description Languages

INTRODUCTION—A PARADIGM

There is a large risk involved in attempting a review of a field which is only 20 years old—especially if a strong consensus of opinion is not already developed around it. The field of pattern recognition is at what Kuhn [35] in his The Structure of Scientific Revolution would call the preparadigmatic stage: no uniform point of view of sufficient precision has been accepted from which problems can be stated, developed, and discussed.

As is characteristic at this stage for any field, the field of pattern recognition has been developed by the "breakthrough"-like appearance of a set of disconnected techniques: the Perceptron [1] of the late 1950s, the grammatical description [2, 3] of the early 1960s, and what appears to many as a new technique—that of "structural description" of the late 1960s and early this decade [4, 5]—have developed almost in isolation from one another.

Any attempt at a review of the field at this stage forces a mere recounting of all interesting results in the field and taking the risk of losing track of the central issues which history will eventually establish for the field. Contrarywise, any attempt to avoid this "piecemeal" approach forces the reviewer to identify "central" issues which may or may not be established as central issues by future insights and events.

I have chosen to take the second alternative. My attempt will be to correlate the different classes of techniques in the study of learning—and such a correlation is not possible without a uniform point of view.

Any uniform point of view one adopts, of course, sets up a contending paradigm, and in setting up such a paradigm I am—at least indirectly—making a value judgment as to what would be a good paradigm for discussing the field. This is only an incidental by-product of the major purpose of the review, to wit to compare certain pattern description techniques among themselves, and to indicate how this class of techniques is related to the rest of the field of pattern recognition.

The approach I intend to make the basis of my analysis was known very early to some psychologists and library scientists. Some work on semantic information processing [6] touched it tangentially, and recent research on scene analysis and robotics has brought it into the forefront with the name "structural description." My personal contacts with proponents of structural description and of grammatical descriptions, however, convince me that for the purposes of avoiding a lot of confusion I should, at least in this present review, generate a name for this approach. I shall call it the interpreted logical description.

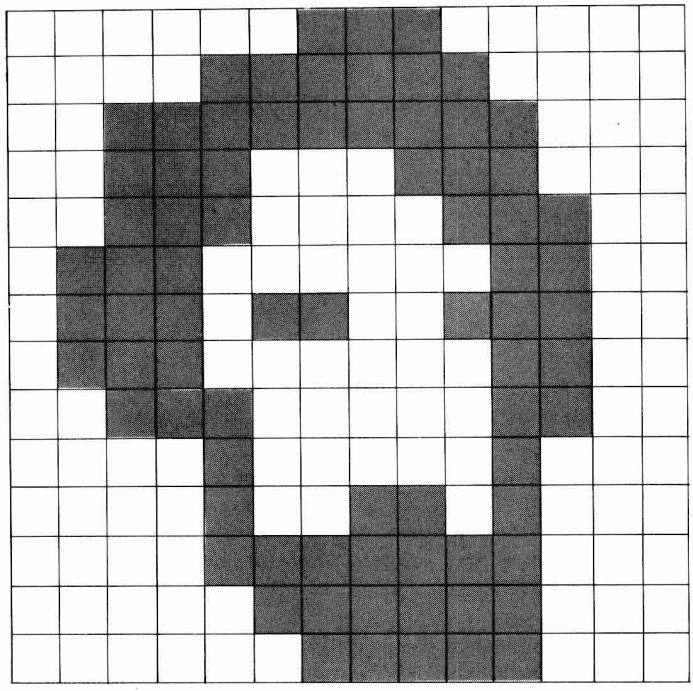


FIG. 1. A pattern to recognize.

BASIC DEFINITIONS

If we start with the agreement that the field of pattern recognition deals with the art and science of recognizing patterns, we still have to be specific as to what we mean by the words "pattern" and "recognition."

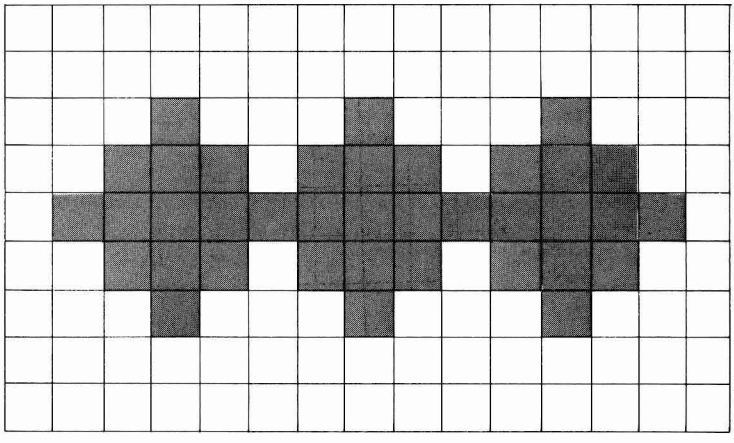


FIG. 2. A pattern with a pattern.

Let us look at Fig. 1—a crude copy of a celebrated digitization of a picture [7] barely recognizable as Lincoln. It is a pattern of black squares.

But Fig. 2 has a pattern, too: not just because it is an arrangement of black squares. We recognize a regularity. Do we recognize a pattern in a pattern? Did we recognize Lincoln or did we recognize the pattern to be Lincoln?

And all patterns are not arrangements. How about recognizing a schizophrenic pattern of behavior? Is the behavior recognized as schizophrenic or is the pattern recognized as schizophrenic?

I hope it is obvious that some fixing of terminology is needed, and all common usage of the same term cannot be allowed to have the same definition in a technical discourse.

In the terminology I am going to use, the word "pattern" will mean the pictures in Figs. 1 and 2 and the behavior in the third example—these are recognized in the sense that they have names, Lincoln, regular, and schizophrenic. When we recognize a pattern, we give it a name.

The naming determines a relation between patterns and names. With each pattern we can attach various names. By our definition we do not recognize the letter A, we recognize a given region of paper with a mark on it as having the possible name A. So the pattern is the region of paper, not the name A. The same pattern also has the name, "English letter," "upper case letter," "used part of a piece of paper," and so on. When we recognize it as an A we are essentially answering the question, "Which among the names A, B, C, ... etc. does this region of paper have?" Similarly, we recognize a bridge position as belonging to the class (or having the name) "biddable," or a certain person's behavior as having the name "healthy."

Our example patterns (pictures, bridge-hands, behavior) do not clearly define what a pattern "is." Let us leave that question unanswered for the present, using the mathematician's favorite trick of "abstraction," and say that we have a set of things called "patterns" and a set of things called "names" with a relation between them. This relational structure we shall call a recognition requirement.

The reason that an act of recognition is often considered an act of classification is that with each name we can associate the class of all patterns which have that name. When we attach a name to a pattern, we place it in this named class.

One part of the field of pattern recognition thus attempts to develop algorithms which answer questions regarding this relation. We want programs which, given a pattern, can be asked questions like "What are its various names?," "Which among the following names does it have?," or "Does it have this specific name?" For certain purposes, one can even input a name and instruct, "Construct patterns having this name."

It is not clear to me that the present-day terminology of the field makes a clear distinction between the first three. For the purposes of our discussion we shall call the first one "naming," the second one "partition selecting," and the third "verifying." We shall use recognition to mean any one of these activities. The fourth, that of "construction," we shall expand upon later. Another important phenomenon ("learning") will be discussed a little later.

The pattern of Fig. 2 can perhaps be given the name "regular" or whatever name pleases the reader—that is not the main issue about that picture. The important point to note is that when in common parlance we say that Fig. 2 "exhibits a pattern," we are not talking about its belonging to a named class of patterns (in the above technical sense) but that if we took it and either flipped it over about a

horizontal or vertical axis or shifted it cyclically 1/3rd or 2/3rd of its length, the picture would remain unchanged, i.e., the pattern (in our sense) is "invariant under transformation" [8].

Attributes, Descriptions, and Features

A number of important considerations will have to be taken up before we can place this last idea (that of invariance) in perspective. The first such consideration deals with the fact that one pattern can be distinguished from another only because there is some knowledge about the pattern by which this can be done. Let us take the point of view that this knowledge is in the form of the results of a set of measurements on the set of patterns. Using the example of the arrangement of spots on a two-dimensional grid, we may say that each cell in the grid defines a measurement: the result of it divides the set of all patterns into disjoint classes according to its "gray value." One could say alternatively that there is a function defined by the cell, whose values for a pattern act as possible names for patterns—i.e., define subclasses of the class of all patterns. The only difference between the names discussed previously (in connection with the requirement) and these new ones is that these new ones are easy to evaluate—i.e., obvious algorithms exist which, given a pattern, can find the values of these different functions (e.g., one for each cell in the above example). If the only names one were required to attach to patterns (i.e., by the requirement) were the names associated by these functions, the problems of pattern recognition would be simple indeed. Unfortunately, this is not so. Talking about pictures, like Figs. 1 and 2, for instance, one seldom is interested in attaching a name to the set of all pictures in which the upper left-hand cell is black!

Let us assume, then, that an individual pattern is represented to the pattern recognizing device by specifying the results of a set of measurements on the pattern. These measurements are a part of the preprocessing of the patterns to convert them into an input to the recognizing device. The device which makes these measurements we shall consider fixed and call the "front end" or the "preprocessor," and hence these measurements cannot be modified by the mechanisms of the recognizing device. We shall call these measurements the "basic attributes." The results of the measurements, which we shall call "values," need not be numerical. We shall be perfectly amenable to calling "color" a basic attribute, for instance, yielding a value, "red."

Since all that is known to the device about the pattern is the set of values for these basic attributes, it is clear that for a computer to recognize a named class, the name must be a function of the values of the basic attributes. Another way to state the same thing is to say that the named class is obtainable from the set-theoretic combination (the description) of the classes defined by the values of the basic attributes.

This idea (of describing classes by means of set-theoretic combinations of pattern classes named by values of different attributes) gives us certain insights into what pattern classes can be named correctly by a recognition algorithm and what classes cannot be so named. Consider, for instance, the rather trivial example given in Fig. 3. The space of patterns and the pattern class to be recognized are shown in Fig. 3(a). If the basic attributes can place the patterns only in the two single-hatched classes shown in Fig. 3(b) and their complements, then the

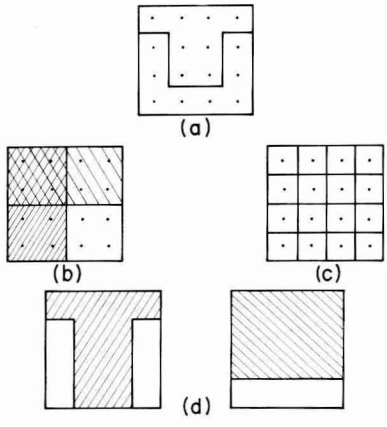


FIG. 3. (a) A class to be described. (b) Useless basic measurements. (c) Inefficient basic measurements. (d) Efficient basic measurements.

class of Fig. 3(a) cannot be described at all. On the other hand, another set of basic attributes, i.e., which separate out the classes shown in Fig. 3(c), can distinguish any two distinct patterns and thus can describe the class of Fig. 3(a).

In a typical pattern recognition requirement one does not know what pattern classes have to be named and so one cannot make an initial judgment as to what basic attributes would make the required class describable. The best one can do is at least to be careful that any two individual patterns which are distinguishable should remain distinguishable in terms of the basic attributes—so no information is lost.

This retention of information, unfortunately, does not solve another problem--the one involving the size and complexity of the description. Any attempt at writing a description of the class in Fig. 3(a) in terms of the attributes shown in Fig. 3(c) would show that the resulting function is complicated. On the other hand, if the basic attributes isolated the classes shown in Fig. 3(d), a simple conjunction would describe the class. However, the use of these latter measurements takes away the ability of describing many other classes, e.g., the one shown double hatched in Fig. 3(b).

This example yields an explication of a truism in the field, i.e., that the efficiency (and even the ultimate capability) of a pattern recognition program depends on the basic measurements used.

Our discussion above indicated that the basic measurements should be chosen so that they can distinguish all distinguishable patterns [the properties in Figs. 3(b) and 3(d) cannot, the one in Fig. 3(c) can]. However, it is also clear that the set of attributes which have such strength may not yield an acceptably short description of all named classes in the requirement.

One way of obviating this dilemma is to start with measurements which can distinguish all patterns (a set of fine-structure attributes) as suggested, then attaching names to certain classes with complex descriptions [like the classes shown in Fig. 3(d)] and using these names (which we shall call "features") to simplify the description of the classes defined by the requirement [like the class

in Fig. 3(a)]. Such simplification is obtained, however, at the expense of the need to store the description of the sets in Fig. 3(d) in terms of the sets in Fig. 3(c). This expense can be justified only if the sets in Fig. 3(d) can be used to simplify descriptions of more than one named class in the requirement. One may say that straight lines form good features in the character recognition requirement because many characters can be described in terms of straight lines.

Growth and Predicates of Higher Arity

One cannot always devise measurements which yield simple descriptions to all the named classes in the requirement. However, one often can simplify many of the descriptions by using features. Therefore, the recognition algorithms should be capable of using the features as well as the basic measurements. Recognition systems which exhibit this capability will be said to be capable of growth. The reason for the name lies in the fact that if we think of the basic measurements and the features as the ingredients of a "language of description," then as we introduce the features by definition, more and more named classes can have simple, uncomplicated descriptions.

We have so far restricted ourselves to values of basic attributes as the building blocks of pattern recognition languages. This is inadequate for dealing with the kinds of descriptions we would need for the picture in Fig. 2, where there is the concept of transformation involved. Transformations are binary relations and all we have in our language so far are unary functions. Before we get to that problem, let us turn to a simpler example than the one in Fig. 2 and consider the picture in Fig. 4—two rather innocuous grids with lines drawn on them with a 45° slope. Let us once more think of each cell on the grid as defining a basic attribute. Let us name these attributes A, B, C, ..., P reading from left to right and top to bottom. The pattern of Fig. 4(A) would be represented by the conjunction of statements (C(u) = 1) & (F(u) = 1) & (I(u) = 1) & (A(u) = 0) & (B(u) = 0) & ... P(u) = 0 while Fig. 4(B) would be represented by the conjunction (H(u) = 1) & (K(u) = 1) & (N(u) = 1) & (A(u) = 0) & ... P(u) = 0.

As can be seen, the set of all straight lines at 45° inclination yields a somewhat repulsive Boolean expression with many disjuncts. What is missing here is the information that somehow A is to the left of B and above E—all information about the spatial arrangements has been lost. It appears on the surface, then, that these facts can only be given to the computer by invoking some binary relations like "above" and "left of" in the set of basic inputs, or a coordinate system with its associated arithmetic has to be invoked in the front end preprocessor for the pattern recognizing device. This does not have to be so. Let us recall that the measurements A, B, C, ... above were defined by cells, each of which has a

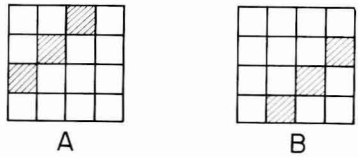


FIG. 4. Two patterns with straight lines.

position defined on it, i.e., has x and y coordinates. Each cell α is denoted by its two coordinates $x(\alpha)$ and $y(\alpha)$. Each measurement, then, can be named in terms of the cell which defines it. To allow this, we shall denote the measurement defined by cell α as m_α. Using this notation, one can write the definition of the class of pictures having a 45° straight line (of any length) through its origin as follows:

$$u \in \text{straight line} \equiv (\forall \alpha)(m_\alpha(u) = 1 \rightarrow x(\alpha) = y(\alpha))$$

It appears, then, that we should extend the mode of representing a pattern, allowing the preprocessor at the front end to allow patterns to act as names of measurements so that these measurement names have measurements defined on them. The values of measurements can also be patterns. As we saw above, with this extension to the representation mode quite complex descriptions can be simplified merely by using the equality sign.

The reader will notice that the above description mode is still inadequate for describing Fig. 4. To obtain greater descriptive strength, one has to have more complex relations definable in the language. Fortunately, this does not need any further enrichment of the capability of the preprocessor or the introduction into the language of any relations other than equality. If the integers themselves are defined by their attributes (say by showing its binary expansion), then arithmetic operations can be described by Boolean algebra also. For example, a "half-adder" would be described

$$(x + y = z) \equiv ((z = 1) \equiv x \neq y)$$

Most recent work, as we shall see later, has concentrated on enriching the preprocessor. Instead of describing complex relations as sentences in the language, basic relations (like "sum of" or "above") are extracted by the preprocess just as the basic measurements were.

What has gone above should not in any way be construed to imply that the design of the basic preprocessor is merely a matter of deciding what they should be. The article Picture Processing in this Encyclopedia discusses the various problems that occur in preprocessing two-dimensional pictures and the techniques that have been used to solve them.

However, one has to remember that careful preprocessing does not obviate the need for introducing new features into the language if they make recognition tasks simpler. Since the basic preprocesses are built in, these features also have to be described in terms of the predicates defined by the basic preprocesses, and the algorithms will have to be capable of using features by definition, i.e., have the capability of growth. Also, one still needs to design the preprocesses quite carefully so fine structure information is not lost.

Let us consider, for instance, a case where one describes the arrangement of geometric figures in two dimensions. One may well be tempted to define basic binary relations between parts of the picture, like "above" and "smaller than." Unfortunately, new needs may be dictated by the requirement. Let us assume that we need the same name for the pictures in Fig. 5. A possible description would be, "If the smaller figure is at the top, then the left edges are flush—else the right edges are flush." The relation of having flush edges cannot be described in terms of the basic relations introduced so far. On the other hand, if the preprocess described attributes of the parts like positions of the boundaries, this would preserve sufficient information. All the binary relations, including "flush edges,"

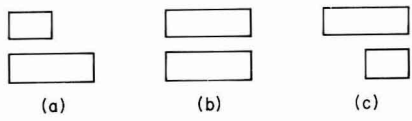


FIG. 5. Pictures with relational descriptions.

could be introduced into the language by growth. And, it is to be pointed out that this latter "information preservation" is intuitively more obvious in terms of the unary predicates defined by the measurements on measurement names.

An Overview

Summarizing, then, we start with a set of patterns, a set of names, and the relation between them, forming a recognition requirement. A set of patterns having the same name is called a named class in the requirement (see above in the section entitled Basic Definitions).

We have a set of predicates on the set of patterns. We started with unary predicates defined by unary functions (measurements) on the set of patterns which defined the basic attributes. We noted that it may not always be possible that a Boolean combination of the values of the basic attributes be the description of a named class in the requirement. Since one often does not know what the descriptions of the named classes are, the least one can do is to devise a set of basic measurements such that each distinct unit set of patterns has a distinct description. This makes it possible to describe all named classes in the requirement. However, to have short descriptions (and short descriptions simplify both storage and processing), one may have to develop intermediate features, defined in terms of the basic predicates, which can then be used to simplify descriptions.

A further method (perhaps as much in keeping with realism) for the simplification of description lies in considering the basic attributes and values themselves to be named by patterns with their own basic attributes and values. These allow the expression of relationships between attributes and values so that parts of a pattern can be defined and talked about. Often, instead of taking this route, one uses basic relations as part of the preprocess, in addition to basic attributes. These basic predicates (defined by basic measurements, measurements on names of measurements, basic relations, etc.) and the allowed connectives constitute the language—in cases enriched by growth—which is used to construct expressions that describe named classes. One can have algorithms which, given a pattern and a set of descriptions of named classes, attach names to the pattern, choose one name for it among a given set of names, or answer questions about its having a specific name.

In most of the discussions above we have used the normal Boolean connectives of propositional calculus as the connectives of the language. However, we have had one example (the description of "straight line") where a universal quantifier was used. It will be noticed that the variable α quantified there ranged over the finite set of measurements defined on the pattern and hence the universal quantifier was basically a shorthand for a long conjunction. Most use of quantifiers, explicit or