A syntax-directed approach to pattern recognition and description

by LARRY D. MENNINGA

Western Washington State College
Bellingham, Washington

INTRODUCTION

Pattern recognition has held the attention of researchers for quite some time. Early efforts were in optical character recognition and were intended to provide easier and more rapid communication from man to machine. In more recent years, this research has expanded to include the processing of pictorial information, such as that from high energy physics or medical research.

Until recently, most of the work and associated theory has treated pattern recognition as a classification or categorization problem. The methods used can be broadly characterized as follows: Each sample pattern is represented by an n-dimensional vector whose components are the values of the individual features, or properties, which have been selected for measurement. The classification is done by partitioning n-space, referred to as the feature space, into subspaces. Each subspace represents a class, and a sample pattern is considered to be a member of the class corresponding to the subspace of which the pattern vector is a member. The members of each class must be clustered in the feature space to achieve successful recognition. The use of weight vectors, proper selection of features, adaptive systems, and other techniques for improvement have been investigated. A survey of this work is given by Nagy.¹

The most significant shortcoming of the above method is that the result of such a recognition process yields only a classification, a class name or number. What is often desired is a structural description of the pattern or an analysis of the relationships existing between certain substructures within the pattern. This is especially true of more complex patterns. Of course, it is possible to divide the feature space into classes, with each one corresponding to a different structural description. However, complex patterns would necessitate an extremely large number of classes and would result in unmanageable problems.

Such classification methods are also essentially one-level. The features selected for measurement must be able to detect any structural relationships which are significant. As the patterns become more complex, the selection of a suitable set of features becomes more difficult.

Finally, the vector representation of a pattern is not the most natural for people. To make use of the potential for powerful recognition systems using interactive computing, the human being must be able to communicate with the computer using representations which he can grasp rapidly.

The linguistic approach

In recent years, research has been done using linguistic methods to overcome the failures of the classification methods. The linguistic methods are aimed specifically at producing structural descriptions of patterns. Miller and Shaw² give a survey of much of this work.

In the linguistic approach, the pattern, or picture, is considered to be a sentence in a language generated by a given grammar. This grammar is defined either explicitly or implicitly, and it is used to analyze each sample pattern. In such a grammar, the nonterminal symbols are phrases descriptive of a subpattern and the terminal symbols (called primitives) are the basic elements which are given a priori, and they are recognized by some method outside of the linguistic model.

A given sample pattern is then described, or recognized, by using the grammar to analyze it. A derivation tree gives a description of the sample in terms of substructures and the relationships which they satisfy—insofar as these are included in the grammar. Thus, in addition to being the desired result of an analysis, the description, as expressed in the rules for the grammar, can also direct the recognition process by defining the sequence of algorithms to be used.
A SYNTAX-DIRECTED SYSTEM

A system called PARSE, which uses the linguistic approach to pattern description, will be described here. PARSE is an acronym for Pattern Analysis and Recognition by Syntax Evaluation. It is a system in which the user must supply the metalanguage to be used to analyze and describe patterns.

Structure of grammar for PARSE

The formal syntax for a user-supplied grammar rule is shown in Figure 1. Each rule in the grammar is called a pattern rule and gives the definition of an object. A set of pattern rules is a pattern grammar. The left side of each rule begins with an identifier. This identifier is called an object type and is a nonterminal symbol in the vocabulary of the pattern grammar. In addition, the left side of each rule includes a list of labels. The right side of each rule consists of a list of object types, with associated labels, and predicates. There are two special object types, POINT and LINE, which are terminal symbols in the vocabulary. These are the primitives, that is, they are object types which have been defined a priori. They are the basic elements in the language (or in the patterns), and they are recognized outside of the PARSE system. The list of predicates in Figure 1 is not intended to be exhaustive, but rather to represent a typical set.

More than one pattern rule is allowed for any object type. This makes it possible to give alternate definitions for an object type. By using more than one pattern rule, it is possible to construct recursive definitions. Each object type must be defined by a pattern rule, or be a primitive, if it is to appear on the right side of a pattern rule.

As an example, consider the grammar rules given in Figure 2. In the example, the object type HOUSE is defined in terms of the object types TRIANGLE and RECTANGLE and the relationships are specified by the predicates above, parallel, perp, and skew. The labels are used to identify object types, and they may be used to specify a correspondence between substructures. The relationship of identity is given implicitly by repeating an object label more than once in a pattern rule. In the rule for HOUSE, the object label X is associated with both TRIANGLE and RECTANGLE to indicate that the same instance of the line X must be a part of each of these objects.

Although the user may specify the grammar which he wants to be used, this specification is subject to certain restrictions imposed by the syntax rules of Figure 1. In each pattern rule, the labels and predicates specify semantic information for the rule, and so these elements can be treated separately. Ignoring this semantic information, the “underlying” grammar can...
easily be seen to be a context-free phrase structure grammar, as defined by Chomsky, since only one nonterminal symbol is allowed to appear on the left side of a pattern rule. Note that the empty string cannot be generated by any of the allowable rules and that all rules are length preserving. For such a context-free grammar, there is a derivation for every sentence in the language which is generated by that grammar. Thus it is possible to define an algorithm to determine a description (structure) for a given pattern (string) in the language.

Semantics used by PARSE

Knowing the structure of a pattern is not enough. Semantics are also involved in the descriptions of patterns. The semantics give meaning (principally spatial relationships) to a particular structure. Semantic information is given primarily by the predicates and the object labels in the pattern rules. The meaning supplied by these elements of the language will be formalized in a manner similar to that suggested by Knuth.

The semantics will be supplied by the values of certain attributes, or properties, that the symbols of the language will have. The attributes selected can be divided into two areas: geometric properties and labels. The value of each attribute is assigned by evaluating a function. The same name will be used for an attribute and its corresponding function.

Each pattern rule has semantics associated with it. Thus it is necessary to have a function for each attribute of each symbol in the rule. The value assigned to an attribute of a given symbol depends on the values of some of the attributes of the other symbols in the rule and on some of the other attributes of the same symbol.

Consider the underlying grammar given by a set of pattern rules. Rules in the underlying grammar will be called syntax rules. Let G be such a context-free grammar, \( G = (V_T, V_N, P, S) \), where \( V_T = \{ \text{terminal symbols} \} \), \( V_N = \{ \text{nonterminal symbols} \} \), \( P = \{ \text{syntax rules} \} \), and S is the start symbol. For each symbol X in \( V_T \cup V_N \), there is a finite set of attributes or properties, \( A(X) \). Let \( Y_a \) be the set of values that can be assumed by a given attribute \( a \) in \( A(X) \).

If the \( r \)th syntax rule is

\[
X_r \rightarrow X_1 X_2 \ldots X_n
\]

where \( X_1, X_2, \ldots, X_n \in V_T \cup V_N \) for \( 1 \leq j \leq n \) then the semantics can be defined as follows: For each attribute \( \alpha \) of a symbol \( X_\alpha \), there is a function \( f_{\alpha} \) which maps the values of certain attributes of \( X_{\alpha_1}, X_{\alpha_2}, \ldots, X_{\alpha_n} \) into a value of \( \alpha \). That is, \( f_{\alpha}(X_{\alpha_1}, X_{\alpha_2}, \ldots, X_{\alpha_n}) \rightarrow \alpha_{\alpha} \), where \( \alpha_{\alpha} = \alpha_{\alpha}(j, \alpha) \) is an attribute of \( X_{\alpha} \) for \( 0 \leq k_i = K_i(j, \alpha) \leq n \) and \( 1 \leq i \leq t(j, \alpha) \).

Let \( A_g(X) = \{ \text{angle, length, } x_{\min}, x_{\max}, y_{\min}, y_{\max} \} \) for every symbol \( X \) which is not a primitive.

\[
A_g(\text{POINT}) = \{ x_{\min}, y_{\min} \}
\]

\[
A_g(\text{LINE}) = \{ \text{angle, length, } x_{\min}, x_{\max}, y_{\min}, y_{\max} \}.
\]

For each symbol \( X \), partition \( A(X) \) into two disjoint subsets \( A_g(X) \), the geometric attributes, and \( A_s(X) \), the attributes dealing with the labels. The sets of geometric attributes are given in Figure 3. The label attributes are handled in a similar fashion.

Each attribute is assigned a value by a function of the other attributes. Let \( (x_1, y_1), (x_2, y_2) \) be a pair of points giving the cartesian coordinates of the primitive \( \text{LINE} \) and \( (x, y) \) be the coordinates for \( \text{POINT} \). Then the functions which give the semantic rules are defined as follows:

\[
\text{angle(LINE)} = \frac{\arctan[(y_1-y_2)/(x_1-x_2)]}{\pi/2} \text{ for } x_1 \neq x_2
\]

\[
\text{length(LINE)} = \sqrt{(x_1-x_2)^2 +(y_1-y_2)^2}
\]

\[
x_{\min}(\text{POINT}) = x
\]

\[
y_{\min}(\text{POINT}) = y
\]

\[
x_{\min}(\text{LINE}) = \text{minimum } \{x_1, x_2\}
\]

\[
y_{\min}(\text{LINE}) = \text{minimum } \{y_1, y_2\}
\]

\[
x_{\max}(\text{LINE}) = \text{maximum } \{x_1, x_2\}
\]

\[
y_{\max}(\text{LINE}) = \text{maximum } \{y_1, y_2\}
\]

For the \( r \)th rule \( X_r \rightarrow X_1 X_2 \ldots X_n \), the semantic rules for the attributes in \( A_g \) are:

\[
\text{angle}(X_\alpha) = \frac{\sum \text{angle}(X_i) \cdot \text{length}(X_i)}{\sum \text{length}(X_i)}
\]

\[
\text{length}(X_\alpha) = \sum \text{length}(X_i)
\]

\[
x_{\min}(X_\alpha) = \text{minimum } \{ x_{\min}(X_i) \ | \ i=1, 2, \ldots, n \}
\]

\[
x_{\max}(X_\alpha) = \text{maximum } \{ x_{\max}(X_i) \ | \ i=1, 2, \ldots, n \}
\]

\[
y_{\min}(X_\alpha) = \text{minimum } \{ y_{\min}(X_i) \ | \ i=1, 2, \ldots, n \}
\]

\[
y_{\max}(X_\alpha) = \text{maximum } \{ y_{\max}(X_i) \ | \ i=1, 2, \ldots, n \}
\]

A "meaning" is assigned to each sentence in the language by the semantics. A derivation of the sentence is carried out in the usual way, using the syntax rules. Starting with the terminal symbols, the attributes are evaluated for each symbol in the derivation. It is easy to see that the semantic rules define the attributes of a symbol \( X \) as a function of the attributes of those symbols which appear on the right side of a production.
defining $X$. Thus, by first evaluating the attributes of the terminal symbols, and working backward through the derivation, the attributes of each symbol can be defined. When all the attributes can be evaluated the semantic rules are said to be well-defined. The meaning of the sentence is the value of the attributes of the start symbol.

In addition, any symbol can be considered to have meaning, determined by the values of its attributes. The PARSE system allows restrictions to be placed upon the meaning of certain symbols or sub-derivations. These restrictions are used to limit membership in a given pattern language. Thus, if $P_G$ is a pattern grammar with underlying grammar $G$, then $L(P_G)$, the language generated by $P_G$, is just those sentences with structure generated by $G$ which satisfy the semantic restrictions imposed by the predicates and labels.

The specifications of the syntax for the pattern rules allow predicates to be used in two ways. When a predicate is used as an instance of (relator) it is a two argument predicate, with the first argument being the basic object preceding it, and the second argument the basic object that follows the predicate. When a predicate is used as an instance of the metalinguistic variable (predicate list), its arguments are the objects corresponding to the labels appearing as formal parameters in the rule. In both cases the predicate is evaluated using the geometric attributes of the object types which make up its arguments. The predicates are defined to allow sets of symbols as arguments because instances of (basic object) can include more than one symbol.

To illustrate how the semantic information is evaluated and used, consider the grammar rules given in Figure 2. For each symbol in the grammar, the set of geometric attributes is the same. Thus $A_x(\text{HOUSE}) = A_x(\text{TRIANGLE}) = A_x(\text{RECTANGLE}) = \{\text{angle, length}, x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}\}$. Figure 4 shows the derivation of a sample TRIANGLE and a table of attribute values using the given pattern rule. The underlying syntax rule for TRIANGLE is:

$$\text{TRIANGLE} \rightarrow \text{LINE}_1 \text{ LINE}_2 \text{ LINE}_3.$$  

### Using the grammar

#### Recognition

The analysis uses the techniques of syntax-directed compiling to recognize patterns. A "top down" analysis of each sample pattern is done to determine if it is a sentence in the language generated by the grammar. A sketch of the procedure to use is as follows: Consider the input pattern to be a set, $Q$, rather than a string. This set is assumed to be finite. Now, beginning with the start symbol, $S$, or global object type, generate a sentence, or pattern, by first replacing the start symbol by its definition as supplied by a production with that symbol as the left side: $S \rightarrow X_1X_2 \ldots X_n$.

1. If $X_1, X_2, \ldots, X_n$ are all terminal symbols, then map the set $\{X_1, X_2, \ldots, X_n\}$ into the input set, $Q$. Evaluate the semantics for the production using the values of the attributes of the images of the $X_i$'s. If the semantic restrictions are met, then $Q$ is an instance of the object type $S$, and it is a member of the language generated by $S$. If the semantic restrictions are not satisfied a new mapping must be tried. This continues until either all the possible mappings are tried or the semantics are satisfied.

2. If the $X_i$'s are not all terminal symbols, then choose the first nonterminal symbol, say $X_j$. Apply the first production for $X_j$ (if there is more than one production): $X_j \rightarrow X_{j1}X_{j2} \ldots X_{jm}$. Now proceed as in step (1) above, mapping $\{X_{j1}, X_{j2}, \ldots, X_{jm}\}$ into $Q$.

3. Continue, by repeating steps (1) and (2) until all terminals are reached. If at any point in the procedure all the mappings have been tried and the semantic restrictions have not been met, then go back to the previous performance of step (2) and apply the next alternate definition. If all the alternate definitions for a given nonterminal symbol have been tried for a particular performance of step (2), then go to the step (2)
two levels back. If all the alternate definitions for the start symbol have been tried and the semantics are still not satisfied, then the input set, $Q$, is not in the language generated by $S$.

**Description**

Since the primary motivation for the linguistic approach to pattern recognition is to produce a description of the patterns processed, the result of an analysis must allow this possibility. The result of a parse, as presented in the above, will be a yes or no answer, as to whether the input pattern is an instance of the object type which is the start symbol of the pattern grammar being used.

In addition, the derivation of the sentence should be kept, so that the user can have this derivation printed out. This will then be a description of the pattern in terms of the subobjects of which it is composed and the spatial relationships which they satisfy, as specified in the pattern rules.

This is all very good and useful, but it does not give any information about patterns which are not sentences in the pattern language. A description of such a pattern, in terms of object types within the grammar and the spatial predicates, should also be produced.

The object types, used as nonterminal symbols in the pattern grammar, form a natural hierarchy. The start symbol is the highest level object type in that hierarchy. The object types are ordered by assigning $X$ a lower level than $Y$ if the first production defining $X$ occurs previous to the first production defining $Y$. Because the object types in the right side of a pattern rule must all be defined, this will result in having $X$ lower level than $Y$ if there is a production $Y \rightarrow sXt$ and no production $X \rightarrow uYe$, where $s$, $t$, $u$, and $v$ are strings of symbols (possibly empty). Also, $X$ is lower level than $Y$ if $Z$ is lower level than $Y$ and $Z \rightarrow sXt$ occurs before any production of the form $X \rightarrow uYe$.

In cases in which the input pattern is not a sentence in the language, the desired result is a description of the input in terms of the highest level object types. This description will not be unique, in general, but should be minimal in some sense.

Minimization will be achieved as follows: First the highest level object will be found such that the input pattern is a candidate for inclusion in the language generated starting with that object type. In other words, the highest level object type, such that the input includes an instance of it, is found first. Each input line, used to compose the object found, is marked. Now an instance of the highest level object type in the hierarchy, which is composed of the maximum number of un-marked lines, is sought. If none is found with at least one unmarked line, the object type which is the second highest in the hierarchy must be tried. If an instance is found, the unmarked lines are marked, and another instance of that type is sought. This process continues until all lines are marked or all object types have been tried unsuccessfully.

A list of the instances of object types found in this manner is constructed. The predicates defined within the PARSE system are evaluated using all pairwise combinations of objects found as arguments. Those predicates which are true are added to the list of object types, and this then becomes the description of the input pattern. Figure 5 is an example of such a description.

**COMPUTER IMPLEMENTATION**

**Computer and language used**

PARSE has been programmed on the Burroughs B5500 computer system at the University of Washington Computer Center. Interaction is provided by using a remote teletype as an input and output device. Consideration was given to using a list processing language for the implementation, since much of the data is best handled by using linked-list data structures. However, for reasons of availability, as well as ease of programming, Burroughs Extended ALGOL was used.
The PARSE program can be divided into three major functional areas:

1. Preprocessor for pattern grammar rules.
2. Building of the data structure.
3. Analysis of the data according to the pattern grammar.

The pattern rules are input from the teletype according to the syntax of Figure 1. The pattern rules are checked for syntactical correctness by a top-down analysis. Each pattern rule is made into a doubly-linked list structure. An example of such a structure is shown in Figure 6.

Each node in the list contains two pointers. The forward pointer indicates the next node in the definition, while the backward pointer specifies the node to go back to in case backtracking is necessary during the analysis of a pattern. Each node has a flag to indicate whether it is a predicate or an object type, the name of the predicate or object type, and a list of integers specifying the labels used by that object type or predicate.

The data structure

The pattern is also represented by using a linked-list structure. This representation arises naturally from the hierarchical treatment of the data during analysis. Each node represents an instance of an object type and consists of several fields. These are a label, a type designation, a pointer to the next node of the same type object, and pointers to several lists. There is a subobject list made up of objects which compose the given object and also a superobject list which contains those objects of which it is a part. There is also an attribute list which contains the values of the geometric attributes the object has.

Before any processing of the pattern is done, the data structure consists of only lines and points, the primitives. During the analysis, if a given object type is being sought, the data structure is examined to see if an instance of that type object is present. If not, the rule defining that object is invoked. Each node of the definition is to be satisfied by searching the data or invoking a definition if the node specifies an object type, or by evaluating the predicate if the node specifies one.

As each object type is found it is inserted into the data structure. Thus, once a given instance of an object has been found, by satisfying the pattern rule defining it, the work done in finding it will not need to be repeated even though it may not be used at that point in the analysis.

CONCLUSION

PARSE is most similar to the system described by Evans.7 Evans' grammars are somewhat more restrictive in their specification of semantics. The Picture Description Language of Shaw8,9,10 can be modeled in PARSE by associating a label for head and a label for tail with each object type. Similarly, the system of Ledley11,12 is less powerful than PARSE. The only spatial relationship that his system allows is concatenation, which can be handled with the labels alone in PARSE.

While some simple patterns have been processed using PARSE,13 it has not been tested on any complex pictures, and thus performance figures are not available. In order to handle any automatic picture processing, a device for recognizing the primitives would be a necessary addition to the system.

The PARSE system produces a description of an input pattern in terms of a meta-language supplied by the user. Although "natural" is a subjective judgment, the description must be termed natural, in that it is symbolic, using a familiar vocabulary with its usual meaning. The PARSE system allows interaction between man and machine. Success at recognizing any instance of a pattern defined by the grammar is guaranteed by the restriction that the grammar be context-free. Further, a description of any picture will always
be produced, although not a unique one, or necessarily
the same description that a human being would give.

REFERENCES

1 G NAGY
State-of-the-art in pattern recognition
2 W F MILLER A C SHAW
Linguistic methods in picture processing—A survey
Proceedings AFIPS 1968 Fall Joint Computer Conference
Thompson Book Co Washington 1968
3 N CHOMSKY
Formal properties of grammars
In Handbook of Mathematical Psychology Vol II
Luce Bush Galanter Eds Wiley New York 1963
4 J E HOPCROFT J D ULLMAN
Formal languages and their relation to automata
Addison Wesley Menlo Park California 1969
5 D E KNUTH
Semantics of context free languages
6 T E CHEATHAM JR K SATTLEY
Syntax-directed compiling
Proceedings of the AFIPS Spring Joint Computer
Conference Spartan Books Inc Washington 1964
7 T G EVANS
A grammar controlled pattern analyzer
Proceedings of the IFIP Congress Edinburgh 1968
8 A C SHAW
The formal description and parsing of pictures
Report No 84 Stanford Linear Accelerator Center
Stanford California 1968
9 A C SHAW
A formal picture description scheme as a basis for picture
processing systems
Information and Control 14 9–52 January 1969
10 A C SHAW
Parsing of graph representable pictures
JACM 17-3 453–81 July 1970
11 R S LEDLEY ET AL
FIDAC Film input to digital automatic computer and
associated syntax directed pattern programming system
In Optical and Electro Optical Information Processing
12 R S LEDLEY
High speed automatic analysis of biomedical pictures
Science 146 October 1964
13 L D MENNINGA
A syntax-directed approach to the recognition and
description of visual images
Tech Rep TR 70–10–06 University of Washington
Seattle 1970