Reprinted from O.D.Faugeras (ed.) Fundamentals in Computer Vision

Printed in Great Britain

SYNTACTIC AND STRUCTURAL METHODS I

Thomas C. Henderson
Department of Computer Science
The University of Utah
Salt Lake City, Utah 84112

The syntactic and structural methods developed for 2-D shape analysis derive from their more traditional counterparts in formal language theory. The notions of alphabet, string, grammar and language have been extended to account for the description of shapes in the plane. However, in applying syntactic techniques to the analysis of 2-D shapes, several major problems must be overcome which do not arise in classical language theory:

- On a theoretical level, concatenation is rarely the only relation possible between symbols; e.g., one piece of a shape can be above, below, left of or right of another piece of the shape.

- On the practical side, the quality of an actual image may require a preprocessing step, e.g., enhancement, filtering, etc., and for most structural approaches a pattern representation must be determined from the image before any syntax analysis can be performed.

The approaches described here attempt to provide solutions to these problems (see Gonzales & Thomason (1978) for an introduction to syntactic pattern recognition).

Some familiarity with formal language theory, e.g., Hopcroft & Ullmann (1969) is assumed, however, a brief review of the basic notions is in order. An alphabet, A, is a finite set of symbols or characters. A string, x, over A is defined as either the empty string, e, or as the concatenation of a non-empty symbol of A with a string s over A. The number of symbols in a string is denoted by $|x|$, and $|e| = 0$. The closure of A, called A^*, is the set of all sentences over A, including e. The positive closure of A, called A^+, is $A^* - \{e\}$. A language is a set of sentences over an alphabet, i.e., a subset of A^*. We define a^n as the empty string if $n = 0$, otherwise, a concatenated with a^{n-1}. A grammar for a language is a finite set of rules for generating exactly the set of

strings in the language, and is defined formally as a 4-tup G = (N,T,P,S), where N is a finite set of nonterminals, T is a finite of terminals, P is a finite set of productions, and S in N is the st symbol. Let $V = N \cup T$, $N \cap T = 0$, and P consists of rewrite rules of the form a -> b where a in $V^* N V^*$ and b in V^*, meaning that the string a may be rewrit as the string b. Let r and s be strings in V^* and a -> b be in P. ras => rbs is meant that rbs is derivable from ras by the application of single rewrite rule. The symbol $=>^*$ indicates the application of zero more rewrite rules, and the symbol $=>^+$ means one or more applications. for every production a -> b in P, a is in N and b is of the form cd, whe c is in T and d is in N, then the grammar is regular. If a is in N and is in V^+, then the grammar is context-free. If a = tcs and b = tds, whe t and s are in V^* and d is in V^+ and c is in N, then the grammar context-sensitive. An unrestricted grammar is one with any kind productions. The subset of T^* obtained from S by using a finite number productions is denoted by L(G) and is called the language generated by G

$$L(G) = \{x | x \text{ in } T^* \text{ and } S =>^* x\}.$$

A sentential form of a grammar G is a string over V derivable from S.

Once a grammar has been defined, some type of recogniti device is required. The application of such a recognizer is call parsing. The goal is to produce a description of the test string in ter of the syntactic model. String parsing techniques can also be used shape analysis, and such diverse methods as Earley's algorithm ar precedence parsers have been used. Finally, syntax-directed translatic has been used to create structural descriptions of shape (see Fu (1974)).

Syntactic pattern recognition analysis consists of three majc steps:

1. preprocessing-improving the quality of an image containing the shape, e.g., filtering, enhancement, etc.
2. pattern representation-segmenting the shape and assigning the segments to the parts in the structural model, and
3. syntax analysis-organizing the primitive shape according to the syntactic model.

The preprocessing step should make the primitive extraction easier an more reliable. The primitive parts produced by the segmentation step wil

in turn be used by the syntax analysis component. The syntax analysis step consists of determining if the set of primitives (or some subset) is syntactically well-formed. As a result of the syntax analysis, a parse tree is produced, and the parse tree can be used not only for recognition purposes, but also as a description of the shape.

A structural shape model describes the spatial decomposition of a shape, and consequently, must describe the primitive parts composing the shape. There are no established guidelines for choosing shape primitives; however, there are several desirable characteristics. Primitives should provide a compact description of the shape with little or no loss of information, and the extraction of shape primitives from an image should be relatively simple using existing non-syntactic techniques.

Once the primitives have been obtained, relations between the primitives are computed. For piecewise linear approximations, relations such as adjacency, collinearity and symmetry may be computed. The exact relations computed are determined by the form of the structural model. There are, essentially, two kinds of structural model: a single-level grammar or relational network, and a multi-level grammar. If relational networks are used, then shape analysis involves graph matching. Grammatical models require a parsing procedure. In this chapter, the application of string grammar techniques to shape analysis will be studied.

STRING GRAMMARS

Perhaps the greatest attraction of the syntactic method is the convenience with which a hierarchical description of a shape can be specified. For example, an airplane can be described as composed of a head section and a tail section. These in turn can be described in terms of the nose, wings, fuselage, etc. Thus, not only is a global description of the shape available, but also very local descriptions.

There are two main disadvantages in the use of the syntactic method for analyzing the boundaries of planar objects according to a string grammar:

1. Noise tends to complicate the process of computing the appropriate string structure, and

2. Context sensitive grammars are ordinarily necessary to analyze the complete boundary of a closed curve.

If piecewise linear approximations are first fit to the boundary of shape, then using these primitives as input for syntax analysis w suppress the noise in the boundary proportionately to the degree that linear segments fit the boundary. If the syntax analysis is restricted only a certain level, then context sensitive grammars are not necessary.

When the boundary of a shape is modeled by a string gramma the formal theory of automata can be used to parse strings of sha primitives, and a class of shapes can be characterized as a set strings. A major difficulty is the contruction of a simple gramm (preferably regular or context free) which generates only these string Another approach is to regard combinations of the lowest level symbol i.e., terminal symbols, as features. Then a shape can be characterized these features.

String grammars have been used to detect peaks in waveforms Horowitz (1977). Selected waveform features are extracted and compar with predetermined standards to monitor and classify a wide range phenomena. The technique is composed of two parts:

1. The waveform is fit with piecewise linear approximations and encoded as a string of symbols, and
2. The string is parsed to recognize peaks.

The symbols of the grammar correspond to positive, negative, and ze slope, and each linear segment of the piecewise linear approximation assigned one of these symbols according to its slope. The set of peaks described by a set of regular expressions, and a deterministic conte free grammar is derived which recognizes peaks (if any) in the waveform the non-terminal symbol corresponding to a peak is derived for some pa of the waveform.

PICTURE DESCRIPTION LANGUAGE

Major theoretical impetus was given to the area of syntacti shape analysis when Shaw (1969) designed the Picture Description Langua (PDL). PDL was intended not only for interpretation of pictures, but als for picture generation. In PDL, picture primitives are described as name objects, and objects have attributes. Two special attributes are the hea and the tail of an object, denoted head(a) and tail(a), respectively where a is an object. Six different concatenation operations are define

as follows:

1. a+b: Join head(a) to tail(b),
2. axb: Join tail(a) to tail(b),
3. a-b: Join head(a) to head(b),
4. a*b: Join tail(a) to tail(b) and Join head(a) to head(b),
5. a&b: Join tail(a) to head(b) and Join head(a) to tail(b), and
6. /a: Reverse the head and tail of a.

Objects can be defined using these six operators. Any arbitrary graph of connected primitives can be recognized or generated using these six operators. For describing string languages, only + is needed. A grammar can be defined to express a given class of pictures.

The PDL description of a simple house shape is given by:

T = {+,x,*,-,&,/,0,1,2,3}, where 0,1,2,3 represent the unit vector oriented at 0, 45, 90, and 135 degrees, respectively and whose heads are at the right, upper right, top, and upper left, respectively.

N = {<house>, <top>, <bottom>, <roof>, <right>, <down>},

```
P = {  <house>  ->  <top>*<bottom>,
       <top>    ->  <roof>*0,
       <roof>   ->  1+<right>,
       <right>  ->  /3,
       <bottom> ->  <down>+0+2,
       <down>   ->  /2 }, and
```

S = <house>.

Although a successful experiment was described concerning spark chamber events, there are several problems with PDL. It is a very unnatural representation for plane objects. Moreover, spatial relationships such as inside, to the right of, and above are difficult, if not impossible, to represent. Finally, PDL allows the description of self-intersecting patterns, e.g., (0+3)x1 with the symbols given above, and it is not easy to determine if a grammar allows such patterns.

A generalization due to Feder (1971) allows an arbitrary number of attachment points for syntactic objects. Such grammars are called plex grammars, and they can be used to describe general graphs.

Such grammars, however, are even more cumbersome to use than PDL, and set of productions may easily have unforeseen side effects.

TREE GRAMMARS

In using higher-dimensional grammars for describing structure patterns, one way to limit the complexity of the grammar is to use tree instead of arbitrary graphs. These grammars are called tree grammars, an their use in pattern recognition has been demonstrated in fingerprin classification by Moayer & Fu (1974), bubble chamber event analysis b Bhargava (1972) and LANDSAT data interpretation by Li & Fu (1976).

Our development of tree grammars follows that of Fu & Bhargav (1973). Let U be the free monoid generated by N^+, the set of positiv integers. Let + be the operator and 0 the identity element of U. Let d(a denote the depth of a in U, where d(0) = 0 and d(a+i) = d(a) + 1, for i i N^+. We say a < b iff there exists x in U such that a+x = b. A tre domain D is a finite subset of U satisfying: 1) if b in D and a < b, the a in D, and 2) if a+j in D and i < j in N^+, then a+i in D. The firs condition of a tree domain ensures a path to the root, while the secon condition ensures that if any son of a node is in the domain, then all sons less than that son are in the domain.

A stratified alphabet is a pair <T,r>, where T is a finite set of symbols and r : T -> N, where N = N^+ {0}.

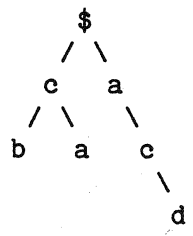
For x in T, r(x) is called the stratification of x. A tree is defined as a mapping from a tree domain to a set of symbols. A tree over <T,r> is a function a : D -> T such that D is a tree domain and r[a(x)] = max{i|x+i in D}. Let D(x) denote the domain of a tree, and let Tr(T) be the set of all trees over T.

A regular tree grammar over <T,r> is a system G = (V,r',P,S), where

1. <V,r'> is a finite stratified alphabet with T in V and r' restricted to T = r,
2. P is a finite set of production pairs of the form (a,b), where a and b are trees over <V,r'>, and
3. S is a finite subset of Tr(V), the set of trees over <V,r>.

We must now show how to represent patterns using trees. Let

the root of a tree be the special symbol $ which corresponds to a physical point. Just as in PDL, symbols in the grammar may correspond to geometric objects. Given a tree, the corresponding pattern can be drawn by starting at the root and drawing line segments according to the descendent of each node. The tail of the descendent is connected to the head of the current node until only nodes with no descendents remain. For example, to describe the same house as was shown in PDL, let a, b, c and d be the same as 0, 1, 2 and 3 in the PDL grammar. Then the house can be described as:



Fu & Bhargava give examples of grammars for directed triangles, chemical structures, electrical circuits, the English alphabet and bubble chamber events.

A major difficulty with using tree automata for pattern recognition is the problem of noisy or distorted data. Error-correcting tree automata have been proposed by Lu & Fu (1978) to deal with these problems; in particular, a distance between two trees can be defined in terms of the least number of error transformations (e.g., substitution or deletion) required to obtain one from the other. The similarity of trees can be defined in terms of the given tree metric.

ARRAY AND WEB GRAMMARS

Pictures are not very easily described as a concatenation of strings. For picture processing, the 2-D analog of the string is the array. Generalizations from phrase structure grammars to grammars capable of dealing with pictures have been proposed by several investigators (see Rosenfeld (1979)). The organization of a picture is in terms of an array, each of whose elements is an intensity in the picture. A picture can then be viewed as composed of subarrays which are themselves composed of subarrays. This gives rise to the following definition:

An array grammar, G, is a quadruple (N,T,P,S), where

- N is a finite set of non-terminals,
- T is a finite set of terminals,
- P is a finite set of pairs (a,b) called productions in which a and b are geometrically identical arrays; a production is applied to an array s by finding a and replacing it with b,
- S in N is the start symbol.

A special symbol, $, is used to allow the definition of geometricall identical arrays; this end marker symbol is sometimes added as the fift part of the grammar. The language of G is the set of arrays derivabl from the initial array (an infinite array composed of the symbol surrounded by an infinite number of $'s) and consisting of $'s an terminal symbols, where all non-$'s are connected.

An array grammar is monotonic if the productions do not allc for the creation of $. Monotonicity guarantees connectedness of th non-$'s if the non-$'s in all rewrite rules are connected. An arra grammar is called isotonic if for every production pair (a,b), a and b ar geometrically identical arrays. For us then, array grammars are b defintion isotonic. When replacing one string by another in a strin grammar, the result is a new string. However, when replacing an arbitrar subarray b, a problem is encountered in making the rows and columns of match the size of the array left vacant by a. One solution to this proble is to impose the isotonic property on the productions and use the specia symbol $ as filler.

Thus, we see that array grammars provide a reasonable notatic for describing pictures when pictures are thought of as arrays. On th other hand, defining more general relations between object parts an specifying attributes of these parts is not a straight-forward process. generalization of this notion will allow us to accomplish these things.

Web grammars generalize the notion of phrase structur grammars from strings to labeled graphs (see Pfaltz & Rosenfeld (1969)) Both the nodes and the arcs of the graph can be labeled, and the arcs ca be directed. A web grammar is still a quadruple, G=(T,N,P,S), but th vocabulary symbols now correspond to webs (labeled graphs). The star symbol, S, is a one-node graph. A production is a pair (a,b), where a an b are webs, and every production must now specify how b is attached to th

REFERENCES

Abe, N., Mizumoto, M., Toyoda, J. & Tanaka, K. (1973). Web Grammars and Several Graphs. J. Comp. and Sys. Sci., 7, 37-65.

Bhargava, B.K. (1972). Application of Tree Systems Approach to Classification of Bubble Chamber Photographs. TR-EE 72-30, Purdue U.

Brayer, J.M. & Fu, K.S. (1974). Some Problems of Web Grammars. TR-EE 74-19, Purdue U.

Brayer, J.M., Swain, P.H. & Fu, K.S. (1977). Modeling of Earth Resources Satellite Data. In Syntactic Pattern Recognition, Applications, ed. K.S. Fu, pp. 215-242, Berlin: Springer-Verlag.

Feder, J. (1971). Plex Languages. Inf. Sci., 3, 225-241.

Fu, K.S. & Bhargava, B.K. (1973). Tree Systems for Syntactic Pattern Recognition. IEEE T. on Computers, C-22, 1087-1099.

Fu, K.S. (1974). Syntactic Methods in Pattern Recognition. New York: Academic Press.

Gonzalez, R.C. & Thomason, M.G. (1978). Syntactic Pattern Recognition. Reading, Ma.: Addison-Wesley.

Hopcroft, J.E. & Ullman, J.D. (1969). Formal Languages and Their Relatio to Automata. Reading, Ma: Addison-Wesley.

Horowitz, S. (1977). Peak Recognition in Waveforms. In Syntactic Pattern Recognition, Applications, ed. K.S. Fu, pp. 1-31, Berlin: Springer-Verlag.

Li, R.Y. & Fu, K.S. (1976). Tree System Approach for LANDSAT Data Interpretation. Purdue-LARS Symp. on Mach. Proc.. of Remotely Sensed Data, West Lafayette, In.

Lu, S. & Fu, K.S. (1978). Error-Correcting Tree Automata for Syntactic Pattern Recognition. IEEE T. on Computers, C-27, no. 11, 1040-1053.

Moayer, B. & Fu, K.S. (1974). Syntactic Pattern Recognition of Fingerprints. TR-EE 74-36, Purdue U.

Pfaltz, J.L. & Rosenfeld, A. (1969). Web Grammars. Proc. 1st IJCAI, 609-619.

Rosenfeld, A. (1979). Picture Languages. New York: Academic Press.

Shaw, A.C. (1969). A Formal Picture Description Scheme as a Basis for Picture Processing Systems. Inf. and Con., 14, 9-52.

neighborhood of a. This specification is called an embedding. A normal embedding has been defined to be one that calls for the same number of nodes in a as in b. Other approaches give embeddings in terms of the neighbor labels or in terms of homomorphisms.

In describing an embedding, it has been found useful to use negative context as well as positive context. Positive context includes neighbor labels that can appear, while negative context rules out possible labels for neighbor nodes. This does not arise in string and array grammars since only a finite number of neighbors are possible, and therefore, negative context can be written as positive context. Since webs allow for an arbitrary number of connections between nodes, negative context becomes necessary.

Much of the hierarchy (in the Chomsky sense) of string grammars carries over to web grammars, and finite-state, context-free, context-sensitive and monotonic web grammars can be defined in a similar way. Brayer & Fu (1974) have shown context-sensitive and monotonic web grammars to be equivalent; they also provide a wide range of other results. For a discussion of inclusion relations between classes of web grammars, see Abe et al (1973).

Brayer et al (1977) have described an application of web grammars in pixel classification in LANDSAT images. Grammars are developed for clouds, shadows, downtown areas and highways, and these provide guidance in recognizing complexly structured land use classes in the imagery in a way that is not possible with spectral properties alone. It was shown that the use of spatial information was of some help in complex natural scenes, and that the amount of structure present allows a significant application of this syntactic technique to the land use problem.