SHAPE GRAMMAR COMPILERS

by

Thomas C. Henderson
Institut für Nachrichtentechnik
Deutsche Forschungs- und Versuchsanstalt
für Luft- und Raumfahrt
8031 Weßling, W.Germany

and

Larry Davis
Computer Science Department
The University of Texas
Austin, Texas USA

Shape Grammar Compilers

Introduction

Many syntactic shape models have been proposed (see Shaw 1969, Fu and Bhargava 1973, and Rosenfeld and Milgram 1972), but the choice of parsing algorithms for these models has received scant attention (an exception to this is Stockman 1977). However, if any practical result is expected from the synactic approach, it is necessary to establish the relation between the shape grammar and the parsing mechanism. In this paper we restrict our attention to bottom-up shape parsing, and we show how such a relation can be defined between a special class of shape grammars and a parsing mechanism called a hierarchical constraint process.

In choosing the parsing mechanism for a given shape grammar, the problem is much the same as that faced by a compiler writer trying to choose a recognizer for a string grammar. In the traditional bottom-up parsing approach (see Gries 1969 or Aho and Ullman 1973), a recognizer is implemented in a general way using tables. These tables are derived from the given grammar and describe relations between the vocabulary symbols of the grammar. A constructor is designed which, given a grammar, checks it for suitability and builds the necessary tables for the recognizer. That is, to implement the recognizer for a given grammar, the constructor is run with the grammar as data, and the output is merged with the recognizer. We will show that the Hierarchical Constraint Process (see Henderson 1979) is an extension of this approach from string grammars to shape grammars.

The general shape grammar scheme is shown in Figure 1a, while the traditional parsing arrangement is shown in Figure 1b. A Shape Grammar Compiler produces a shape parsing mechanism (or shape compiler) from some high level shape grammar description. The shape parsing mechanism performs the actual analysis of unknown shapes and outputs an organization imposed on the shape primitives in terms of the underlying grammar.

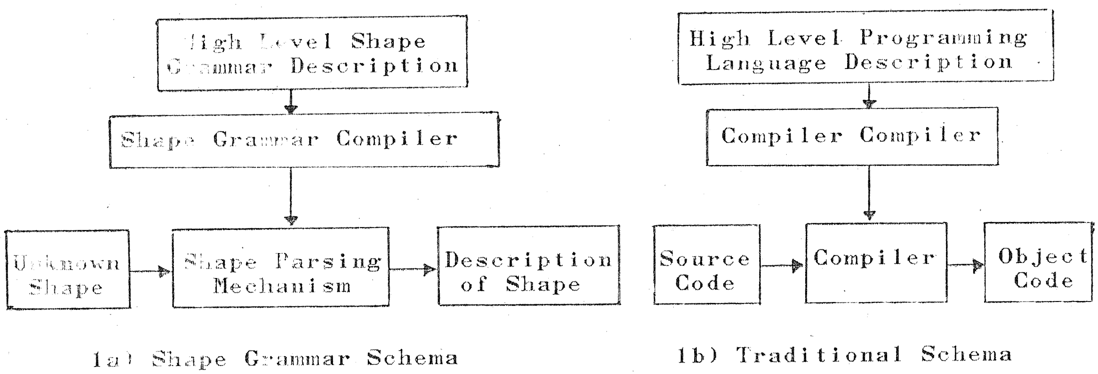

Figure 1. Shape Grammar Compilation

Most syntactic shape analysis methods proposed have dealt with the shape grammar (or model) at length, while the corresponding parsing algorithm has been chosen ad hoc and from a string grammar perspective, e.g., You and Fu 1977 use Earley's algorithm. The shape parsing mechanism has usually been constructed manually. Finally, in most formalisms it is a tedious process to produce the shape grammar for any interesting class of shapes. This provides an impetus for developing more suitable user-oriented languages for describing a shape grammar. Thus, one problem is the design of suitable shape grammar description languages, and the subsequent problem is the construction of correct and efficient compilers for such languages.

As an example of this approach to syntactic shape analysis, we propose a class of constraint grammars and a method for automatically deriving the shape parsing mechanism. In particular, we:

1) define a shape grammar formalism which accounts for most aspects of 2-d shape,

2) define a parsing mechanism which uses constraints between pieces of the shape, and

3) define an automatic method to compute constraint relations between the vocabulary symbols of the shape grammar.

This process can be viewed as a generalization of traditional precedence grammar techniques in that the precedence grammars involve constraints between string grammar symbols. With string grammars, bottom-up parsing involves scanning from left to right until the tail of the handle is found, then scanning right to left from the tail until the head of the handle is found. This works well enough for string grammars, but shape grammars pose the problem of complicated relations between the symbols, and these relations must be accounted for and taken advantage of by the shape parsing mechanism.

Constraint Grammars

Grammatical models for shape analysis have been developed and investigated by many people. With some simple modifications these models can be integrated in a natural way with constraint analysis techniques. An extension of the geometrical grammars of Vamos 1973 will be used to model shapes.

A Stratified Context-Free Grammar, G, is a quadruple (T,N,P,S), where T is the set of terminal symbols, N is the set of non-terminal symbols, P is the set of productions, and S is the start symbol. Let $V = (N \cup T)$ be the set of vocabulary symbols. Associated with every symbol $v \in V$ is a level number, $ln(v):V \to \{0,1,...,n\}$, where $ln(S)=n$ and for every $v \in T$, $ln(v)=0$.

(1) T - corresponds to relatively large pieces of the shapes modeled by the grammar, e.g., straight-edge approximations to the boundary of the shape.

(2) N - consists of a set of symbols each of which has a level number from 1 to n associated with it. In any rule v := a (the rewrite part of a production), if ln(v) = k, then every symbol in the string a is at level k-1. Furthermore, for every $v \in V$

v = <name part> {attachment part} [semantic part], where

<name part> is a unique name by which the symbol v is known,

{attachment part} is a set of attachment points of the symbol,

[semantic part] is a set of predicates which describes certain aspects of the symbol.

(3) P - consists of productions of the form (v:=a,A,C,Ga,Gs), where

v:=a is the rewrite part that indicates the replacement of the symbol v by the group of symbols a, where $v \in N$ and $a = v1v2...vk$ such that $vi \in V$ and $ln(vi) = (ln(v)-1)$ for $i = 1,k$.

A is a set of applicability conditions on the syntactic arrangement of the vi, i = 1,k.

C is a set of semantic consistency conditions on the vi, i = 1,k, and consists of various predicates describing geometric and other properties of the vi.

Ga is a set of rules for generating the attachment part for v, the new symbol.

Gs is a set of rules for generating the semantic part of v, the new symbol.

Stratified grammars have been described in detail elsewhere (Henderson 1979). These grammars give rise to a large set of contextual constraints on the organization of a shape. It is these constraints which must be derived so that a constraint oriented parsing mechanism can be constructed.

We now discuss the procedures for deriving the constraints from the shape grammar. Two types of constraints, syntactic and semantic, are described. The semantic attributes of a vocabulary symbol are computed from the attributes of the symbols which produce it. Knuth (1967) gives a way to define semantics for context-free string languages using both inherited and synthesized attributes; however, the shape grammar formalism used here is closer to property grammars (Aho and Ullman 1973) in that properties are associated with the vocabulary symbols.

Consider a vocabulary symbol as representing a piece of the boundary of a shape. If a vocabulary symbol is part of a complete shape, then it is adjacent to pieces of the shape which can combine with it to produce a higher level vocabulary symbol. Suppose that the set of all possible neighbors of a vocabulary symbol, v, at any given attachment point is known, and that a silhouette boundary includes v. If at one of v's attachment points the attached symbol is not in the neighbor set of v, then the silhouette boundary cannot successfully produce the start symbol. This type of constraint is called a syntactic constraint. Without these constraints several levels of vocabulary symbols might be built before it can be determined that some vocabulary symbol lacks the appropriate context. The use of constraints, however, makes it possible to detect this much earlier.

The other type of constraint involves some (usually geometric) relation between the semantic features of two vocabulary symbols. E.g., let v = <v>(a,b)[s], and let w = <w>(c,d)[t]; then it may be the case that s is parallel to t in any parse producing the start symbol. These kinds of constraints are called semantic constraints. This makes it possible for high level information to be specified, e.g., the orientation of some high level part of the shape, and this information can be used to eliminate silhouette boundaries which are not consistent with this information.

Let G = (T,N,P,S), let v,w and x $\in$ V, let at(v) denote the attachment points of v, and let av $\in$ at(v). The syntactic constraints are defined by specifying for each symbol v a set of vocabulary symbols which may be attached to v. Note that, in general, the attachment points of the symbols must also be specified. To determine this set, we define

(1) (v,av) Ancestor (w,aw) iff there exists p in P such that the rewrite rule of p is v := ...w... and there exists aw in at(w) such that aw is identified with av in Ga of p. Then we say that v is an ancestor of w through attachment point av of v and aw of w, where av and aw represent the same physical location.

(2) (w,aw) Descendent (v,av) iff (v,av) Ancestor (w,aw).

(3) (v,av) Neighbor (w,aw) iff

a) there exists p in P such that the rewrite rule of p is x := ...v...w..., and aw is specified as being joined to av in the applicability condition of p, or

b) there exists x in V with ax in at(x), and there exists y in V with ay in at(y) such that (x,ax) Ancestor (v,av) and (y,ay) Neighbor (x,ax) and (w,aw) Descendent (y,ay).

That is, vocabulary symbols are either directly specified as neighbors in a production, or they are neighbors indirectly by being at the end of higher level symbols which are neighbors.

Using matrix representations for these relations, the descendents and neighbors of a symbol at a particular attachment point can be computed (see Gries 1969 for an introduction to binary relations, their representation using matrices and their manipulation). Let s be the number of vocabulary symbols in V, and let the Boolean matrix Amn be the square matrix of order s whose (i,j)th entry is 1 iff symbol vi is in relation A to symbols vj through endpoint m of vi and endpoint n of vj (consider the endpoints of vocabulary symbols to be ordered). A relation (which is dependent on endpoints) is then fully specified by a total of k**2 matrices, where k is the number of endpoints per symbol. However, if the grammar is written so that endpoints are interchangable, then one matrix will define the relation, i.e., all k**2 matrices are the same.

The Ancestor relation, Amn, is the transitive closure of the matrix having a 1 in the (i,j)th position if the condition given in the definition is satisfied. The Descendent relation, Dmn, is just the transpose of Amn. Given Amn and Dmn, the neighbor relation, Nmn, is computed as:

$$Nmn := Nmn + \sum_{p} [Dmp * [\sum_{q} (Npq*Aqn)]],$$

where + is Boolean "or" and * is Boolean "and", and Npq is just the explicit neighbors given in the productions. This definition of Nmn follows directly from the definition of the relation. The first term on the right, Nmn, corresponds to part a) of the neighbor relation, i.e., this is the neighbor relation defined explicitly in the productions. Let DNAmn denote the second term, i.e.,

$$DNAmn(i,j) = \sum_{p} \{Dmp * \sum_{q} (Npq*Aqn)\}$$

where Rmn(i,j) denotes (i,m) R (j,n) for relation R. Let NApn denote the second term of DNAmn, i.e.,

$$NApn(i,j) = \sum_{q} (Npq*Aqn).$$

Finally, fix q and consider the computation of the (i,j)th entry of the qth summand:

$$= \sum_{K} \{Npq(i,k)*Aqn(k,j)\}.$$

Then each element of this sum is of the form:

(i,p) N (k,q) and (k,q) A (j,n)

which means that vocabulary symbol i at attachment point p is a neighbor of k at point q which is an ancestor of j at point n. thus, NApn(i,j) = 1 means that vocabulary symbol i neighbors an ancestor of j at attachment point p of i and n of j. Likewise, DNAmn(i,j) = 1 means that vocabulary symbol i at attachment point m is a descendent of some neighbor (called x in the definition of the neighbor relation) of some ancestor (called y in the definition) of j at point n. This is precisely what the definition calls for. This relation defines a set of neighbors for every symbol, and if a vocabulary symbol in silhouette boundary fails to have a neighbor from this set, then that silhouette boundary fails to produce the start symbol. This relation constitutes the syntactic constraints.

Semantic constraints can be generated in much the same way as syntactic constraints: by defining binary relations and computing their trasitive closure. For example, the axes of two symbols are parallel if a production states this explicitly or by transitivity through some third symbol. Such relations also allow for semantic features to be fixed (set to some constant), e.g., the orientation of one axis of a vocabulary symbol could be set to 45 degrees, and this certain information can be propagated to other vocabulary symbols. This can be done for example, by having global information available describing known orientations of the vocabulary symbols. In this way, it is possible to determine that certain boundary silhouettes cannot be parsed. The parallel relation was computed between all vocabulary symbols. Note that not every symbol is necessarily parallel to another symbol, moreover, some hypothesized symbols may require other hypotheses of the same symbol to exist. The parallel relation was computed using a binary-valued matrix, whose rows and columns correspond to the axes of the vocabulary symbols.

In general, a transitive relation is computed as:

P := (P(0)+~I)*P(0)!

where ~I is the complement of the Boolean identity matrix and P(0)! is the transitive closure of P(0), the explicit parallel relation. (An explicit relation is one found directly in the productions.) Computed this way a symbol is only parallel to itself if there must exist two distinct vocabulary symbols of the same name part in the boundary silhouette. Relations which are not transitive, e.g., perpendicular, require special procedures for their computation, but any transitive relation can be computed with the above form.

Hierarchical Constraint Processes

The discussion so far applies to string grammars as well as shape grammars. However, a a major problem faced by any syntactic analysis method, but not by string parsers, is the ambiguity of the underlying data. Usually no clear-cut decision can be made in associating the terminal symbols of a grammar with the shape primitives. Thus, the parsing mechanism must not only overcome the problem of parsing a complicated arrangement of symbols (i.e., concatenation is no longer the only relation between symbols), but must also disambiguate the interpretation of a given shape primitive.

The hierarchical constraint process (or HCP) has been proposed as a solution to this problem (Davis 1978, Henderson and Davis 1980 and Henderson 1979). Given a set of shape primitives (a set of linear segments approximating the boundary of some silhouette) and a stratified shape grammar, HCP performs the following actions:

(1) associate with each shape primitive a set of possible interpretations, i.e., terminal symbols,

(2) determine the initial network of hypotheses, that is, for each possible interpretation of each shape primitive, insert a node in the network; two nodes of the network are connected if their underlying shape primitives are physically adjacent,

(3) apply the procedures BUILD, CONSTRAIN and COMPACT to the network until the network is empty or the start symbol has been produced.

The shape primitives are generated by computing several piecewise linear approximations to the boundary of the shape. A modified split-and-merge algorithm (see Pavlidis 1973) fits straight edges to the boundary using the cornerity measure proposed by Freeman and Davis (1977) to choose break points.

The association of terminal symbols with shape primitives will (in the limit) be to hypothesize every terminal symbol for each primitive. However, methods for reducing the number of hypotheses include using a more global analysis to derive indications of appropriate scale, orientation, etc. from simple global properties, e.g., histogram selected features of the primitives themselves to infer properties of particular vocabulary symbols.

The network of hypotheses represents all possible sentential forms for the given shape primitives. Since our grammars define simple closed curves, every cycle in the network represents a distinct set of interpretations of the primitives and must be parsed. However, this is usually much too large a set to be parsed one after the other. The hierarchical constraint process computes a bottom-up parse of all the cycles in parallel. This is done by applying the constraints to the network and can be described by specifying three simple procedures and two sets which these procedures manipulate.

BUILD - Given level k of the network, BUILD uses the productions of the grammar to construct level k+1 nodes. Physically adjacent hypotheses are linked, and a record is kept of which nodes are used to construct each level k+1 node. All level k+1 nodes are put into the CONSTRAIN-SET, and all level k nodes are put into the COMPACT-SET (both of these sets are initially empty).

CONSTRAIN - While the CONSTRAIN-SET is not empty, CONSTRAIN examines each member of that set; if a node fails to satisfy the constraints, then its neighbors are put into the CONSTRAIN-SET, any nodes it helped produce and the nodes used to produce it are put into the COMPACT-SET, and it is deleted from the network.

COMPACT - While the COMPACT-SET is not empty, COMPACT examines each member of that set; given a node n from the COMPACT-SET, if one of the lower level nodes used to produce n has been deleted, or if n has not helped produce a node at the level above it (and that level has been built), then n's neighbors are put into the CONSTRAIN-SET, any nodes it helped produce and the nodes used to produce n are put into the COMPACT-SET, and n is deleted from the network.

This then is the shape parsing mechanism; the constraint propagation is based on the discrete relaxation techniques developed by Waltz (1975) and Rosenfeld et al (1976).

Discussion

In analyzing a class of shapes, we proceed as follows:

(1) define a stratified shape grammar for the class of shapes,

(2) derive the syntactic and semantic constraints between the vocabulary symbols of the grammar,

(3) apply the hierarchical constraint procedure to a set of shape primitives utilyzing the constraints to eliminate incorrect hypotheses. Successful experiments have been run for detecting airplane shapes (see Henderson 1979 for results). However, several problems have been encountered. The stratified shape grammar can have 30 to 40 productions, and a convenient means for defining a grammar has yet to be developed. Thus, at present, shape grammars present a major source of error and usually require much debugging. Furthermore, it may be more convenient to provide for attachment elements (instead of points) so that shape primitives might include polygons as well as line segments.

A major problem with the hierarchical constraint process is the initial size of the network of hypotheses. If there are 50 shape primitives and 10 terminal symbols in the grammar, then the initial network has 500 nodes. Each node has much information associated with it, and consequently a lack of storage may result. However, in such a case it is possible to define a partial network of hypotheses by choosing one hypothesis at a time for a given primitive (or primitives) and running HCP on the resulting network separately.

It might be questioned whether stratification of the grammar is necessary. Without stratification, constraints cannot be applied usefully since a symbol will continually be rebuilt even though it fails to satisfy contextual constraints at some later point. The following example shows this; let part of the production set be:

g <- a + b

h <- c + d

i <- e + f

j <- g + c

k <- d + i

l <- h + f.

Figure 2 shows how h will be continually rebuilt and deleted (dashed lines indicate application of a production, solid lines indicate neighbors).

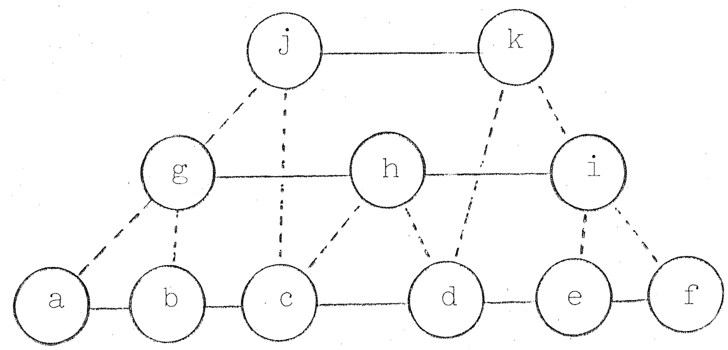

Figure 2 - Necessity of Stratification

Stratification ensures that a hypothesis will be made only once, and if it is incorrect, then it will be permanently deleted.

Thus, we have shown one possible approach to constructing a theory of shape parsing. While drawing heavily on the traditional theory of parsing, the ambiguity of the underlying data requires a different parsing strategy. We have shown how to implement a bottom-up constraint-driven parsing mechanism and how it relates to traditional string parser theory.

References

1. Aho,S. and J.D. Ullman (1973) The Theory of Parsing, Translation and Compiling, Vol. 2, Prentice Hall, New Jersy.

2. Davis, L. (1978) "Hierarchical Relaxation for Shape Analysis," Pattern Recognition and Image Processing Conf., Chicago, Il., June, pp. 275-279. Davis 81

3. Freeman, H. and L. Davis (1977) "A Corner-Finding Algorithm for Chain Coded Curves," IEEE Trans. on Computers, Vol. C-26, March, pp.297-303. Free 77

4. Fu, K.S. and B.K. Bhargava (1973) "Tree Systems for Syntactic Pattern Recognition," IEEE Trans. on Computers, Vol. C-22, December, pp. 1087-1099. Fu 73

5. Gries, D. (1969) Compiler Construction for Digital Computers, John Wiley, New York. Gries 69

6. Henderson, T. and L. Davis (1980) "Hierarchical Models and Analysis of Shape," BPRA 1980 Conf. on Pattern Recognition, Oxford, England, p.79. ? Henderson 81

7. Henderson, T. (1979) "Hierarchical Constraint Processes for Shape Analysis," Ph.D. Dissertation, U. of Texas, December. Davis 79

8. Knuth, D. (1968) "Semantics of Context-Free Languages," Math. Systems Theory, Vol. 2, No. 2, pp.127-145. Knuth 68

9. Pavlidis, T. and S. Horowitz (1973) "Piecewise Approximation of Plane Curves," Pattern Recognition, pp. 346-405.

10. Rosenfeld, A. and D. Milgram (1972) "Web Automata and Web Grammars," in Machine Intelligence (eds. B. Meltzer and D. Michie),7, Edinburgh U. Press, pp. 307-324. Rose 72

11. Rosenfeld, A., R.A. Hummel and S. Zucker (1976) "Scene Labeling by Rose 76a

Relaxation Operations," IEEE Trans. on SMC, Vol. SMC-6, pp. 420-433.

12. Shaw, A.C. (1969) "A Formal Picture Description Scheme as a Basis for Picture Processing Systems," Information and Control, Vol. 14, pp. 9-52.

13. Stockman, G. (1977) "A Problem Reduction Approach to the Linguistic Analysis of Waveforms," U. of Maryland, TR-538, May.

14. Vamos, T. and Z. Vassy (1973) "Industrial Pattern Recognition Experiment - A Syntax Aided Approach," IJCPR, Washington, pp. 4. 445-452.

15. Waltz, D. (1975) "Understanding Line Drawings of Scenes with Shadows," in The Psychology of Computer Vision, (ed. P. Winston), McGraw-Hill, pp. 19-91.

16. You, K. and K.S. Fu (1977) "Syntactic Shape Recognition," in Image Understanding and Information Extraction, Summary Report of Research, March, pp. 72-83.