Linear and Context-Free Graph Grammars

T. PAVLIDIS

Princeton University, Princeton, New Jersey*

ABSTRACT. Topological characterizations of sets of graphs which can be generated by context-free or linear grammars are given. It is shown, for example, that the set of all planar graphs cannot be generated by a context-free grammar while the set of all outerplanar graphs can

KEY WORDS AND PHRASES: graph grammars, graph connectivity, planar graphs, outerplanar graphs

CR CATEGORIES: 3.63, 5.32, 5.22

1. *Introduction*

The subject of graph grammars has received considerable attention recently because of picture processing and pattern recognition applications [1–11].

One basic difference between graph and string grammars is the many ways in which elements can be juxtaposed in graphs while there are only two for strings right and left). Therefore, the definition of such relations is crucial in the description of graph grammars and different models may result for different definitions. This paper attempts to give a characterization of the types of graph grammars which is simple enough to allow one to prove results about them and at the same time, general enough to encompass models of earlier investigators.

One generalization about graph grammars included in this model is to allow nonterminal symbols which are not simply branches or nodes.

Definition 1: An mth order (nonterminal) structure is an entity which is connected to the rest of the graph by m nodes. In particular a 2^d order structure will be called a *branch structure* and a 1st order structure a *node structure.*

One can think of such elements as polygons which are attached to each other through their vertices (Figure 1). Rewriting such a structure corresponds to replacing it by another, or a finite number of them in general, connected to the rest of the graph by exactly the same points and, if more than one replacement, not connected among themselves in any other way. Eventually a mth order structure must be replaced by a subgraph connected to the rest by m nodes. Such structures are closely related to the NAPE's proposed by Feder [10]. The main difference is that here we do not allow such entities to be terminals and because we restrict ourselves to the context-free case, we do not have to worry about identifiers of the attachment points.

The important defining feature of context-free string grammars is that there is

* Department of Electrical Engineering. This work was supported by NSF Grant GK-13622.

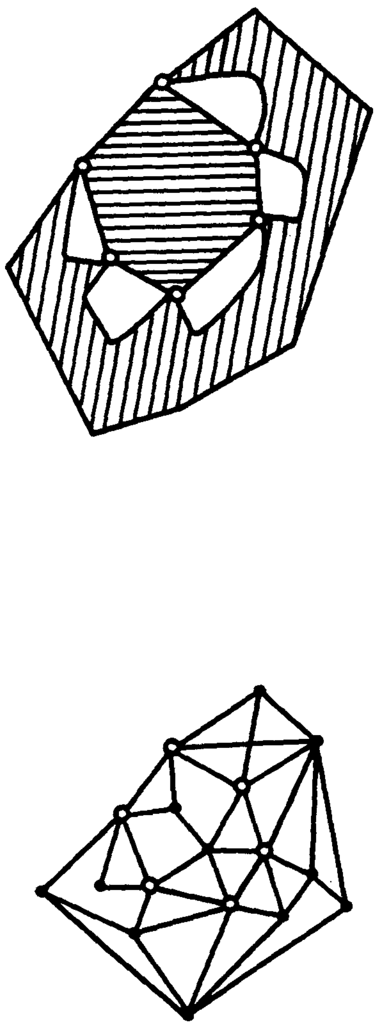


Fig. 1. Illustration of the juxtaposition of two high order structures

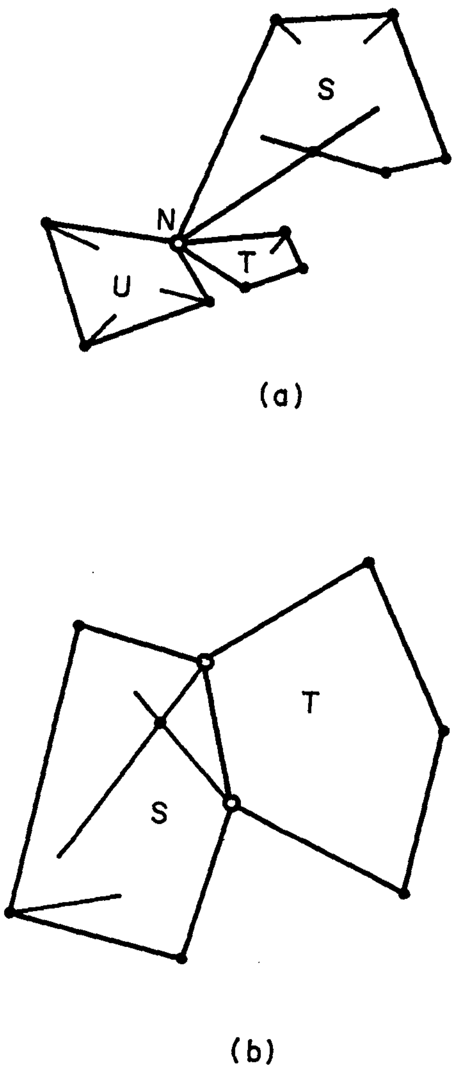


Fig. 2. Illustration of the symbols used in the examples: (a) $N(S + T + U)$, (b) S^*T

no "crosstalk" among the branches of the generating tree of any given string. A graph grammar which has this property should be called context-free without any additional qualifications if one wants to have as general a model as possible. The "no crosstalk" condition can be easily translated into the condition that the left hand side of any rewriting rule (RR) should contain only one nonterminal structure and that the replacement of a structure X should be embedded in the remaining part of the graph in such a way as to treat "uniformly" all the connecting nodes of X. However, the latter condition if taken in its full generality results in graphs which are very difficult to parse and for this reason we introduce an additional constraint which limits the number of nodes to which a nonterminal structure can be connected to.

Definition 2: An mth order context-free graph grammar (CFGG) is a quadruple (N, T, P, I), where

N is a set of nonterminal structures: nodes, branches, triangles, . . . polygons with m vertices.

T is a set of terminal elements: nodes and branches (for unlabeled graphs there will always be only a terminal node symbol and a terminal branch symbol).

P is a finite set of rewriting rules of the form $G \rightarrow H$, where G is a nonterminal structure and H a graph containing possibly both terminals and nonterminals.

H is connected to the rest of the graph through exactly the same nodes as G.

I is a set of initial graphs.

Definition 3: An mth order linear graph grammar (LGG) is an mth order CFGG with the additional constraint that the right hand sides of the rewriting rules contain at most one nonterminal structure.

Context-free web grammars [4, 5] are infinite order CFGG with the constraint that each nonterminal structure is eventually transformed into at least one node, in addition to the connecting nodes. Thus in some aspects they are more and in others less powerful than the present model. The classes of graphs generated by each model intersect but none contains the other.

In the sequences we will discuss only second order grammars. Before proceedings with the characterizations of sets of graphs generated by CFGG and LGG we present some simple examples.

2. *Examples of Context-Free and Linear Graph Grammars*

The following notations will be used in this section for simplicity:

M, N will denote nonterminal node structures.

$B, C, D, E \cdots$ will denote nonterminal branch structures.

Lower case symbols denote terminal elements.

$N(G + H + J + \cdots)$ denotes a number of graphs connected through a common node [Figure 2 (a)].

$G{*}H$ denotes two graphs connected by a pair of nodes [Figure 2 (b)].

Note that all rewriting rules of graph grammars are expressed formally through the graphs or structures involved (i.e. the formalism is two dimensional). Thus the rule

$$B \rightarrow B_1{*}B_2$$

should be interpreted as: Replace branch structure B (connected to the rest of the graph through nodes X and Y) by branch structures B_1 and B_2 (connected to the graph through the same nodes X and Y as B). No other connection exists between B_1 and B_2.

Similarly the rule

$$N \rightarrow N'(T + S)$$

should be interpreted as: Replace node structure N by a node structure N' and two other structures T and S connected to the rest of the graph by the same node as N.

When no ambiguity occurs we can use simple concatenation, e.g. $B'NB''$, to denote a (nonterminal) subgraph consisting of a branch structure B' with nodes X and Y connected to the node structure N through Y and a branch structure

B'' with nodes Y and Z connected to N through Y. The subgraph is connected to the rest of the graph through the nodes X and Z. The notation $B'nB''$ denotes a similar situation, only the node structure N has been replaced by a node n which in this case coincides with Y.

Furthermore an element in the RHS of a rule will be underlined if it is one connected to the rest of the graph. This notation will be omitted wherever it is indifferent which one is the connecting element.

Thus the symbols $*$ and $+$ as well as the other conventions are not part of the terminal alphabet but are used only as shorthand notations. This is a major difference from studies of graph grammars through linearization. The latter has usually been the case in the study of tree grammars [3, 8, 12, 13].

The following examples illustrate second order CFGG. (It is easy to verify that proposed grammar generates exactly the graphs in question):

Example 1: Two terminal series-parallel networds (TTSPN):

$$G_1 = \{B, b, n, P_1, nBn\},$$

$$P_1 = B \rightarrow BnB/B{*}B/b.$$

Example 2: Basic TTSPN [5]. These are networks where if there is a path of length greater than one between two points then there is no direct connection between them:

$$G_2 = \{B, C, b, n, P_2, nBn\},$$

$$P_2 = \begin{cases} B \rightarrow C/D/b, \\ C \rightarrow C{*}C/b, \\ D \rightarrow D{*}D/DnB/BnB. \end{cases}$$

Example 3: Connected trees:

$$G_3 = \{N, b, n, P_3, N\},$$

$$P_3 = N \rightarrow NbN/n.$$

Example 4: Connected binary trees:

$$G_4 = \{N, b, n, P_4, N\},$$

$$P_4 = N \rightarrow nbN/Nb\underline{n}bN/n.$$

Example 5: The homeomorphisms [14] of a given graph H. Let H' denote a graph identical to H having branch structures instead of branches:

$$G_5 = \{B, b, n, P_5, H'\},$$

$$P_5 = B \rightarrow BnB/b.$$

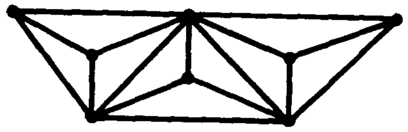
The following examples illustrate linear graph grammars.

Example 6: The trestlelike graphs shown in Figure 3 can be produced by

$$G_6 = \{B, b, n, P_6, K_4\},$$

$$P_6 = B \rightarrow K_4/b,$$

where K_4 is the complete graph with four nodes and just one nonterminal branch structure.



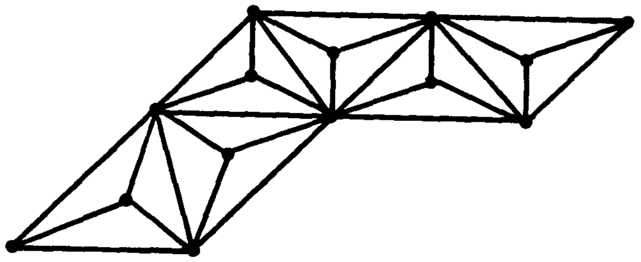


FIG. 3. Graphs generated by an LGG

Example 7: All cycle graphs with labeled branches having the property that all branches with the same label are contiguous can be produced by

$$
\begin{aligned}
G_7 &= \{B, C, D, E, b, c, n, P_7, S\}, \\
P_7 &= S \rightarrow {}_{(}\underline{b\text{–}n\text{–}B\text{–}n\text{–}b\text{–}n}_{)} / {}_{(}\underline{c\text{–}n\text{–}C\text{–}n\text{–}c\text{–}n}_{)}\,, \\
&\quad\; B \rightarrow b/bB/D, \\
&\quad\; C \rightarrow c/cC/E, \\
&\quad\; D \rightarrow c/cD, \\
&\quad\; E \rightarrow b/bE.
\end{aligned}
$$

G_7 generates graphs with only two kinds of labels. It is easy to generalize for graphs with more kinds of labels.

3. *Topological Characterization of Graphs Generated by Context-Free Graph Grammars*

The type of a string grammar depends primarily on the size of patterns which can occur in the strings it generates. Thus such grammars can be characterized by the memory of their recognition automata [15]. On the other hand graph grammars are characterized by topological as well as "size" constraints. We will present a number of such characterizations in this section.

Definition 4: The degree of a node A in a graph is the number of other nodes connected directly (i.e. by at least a single branch) to A. The degree of a graph is the maximum of the degrees of its nodes. The degree of a rewriting rule is the degree of the graph in its right hand side (where the maximization is over all the nodes except the connecting nodes).

The first part of the definition is different than the standard definition of degree [14]. However, the two coincide for graphs where there is at most one branch between any pair of nodes. They differ only for multigraphs.

Definition 5: A rewriting rule will be called "terminal" if its RHS contains only terminal structures. Otherwise it will be called nonterminal.

THEOREM 1. *Let S be a family of graphs generated by a 2^d order CFGG. Then there exists a number k depending only on S such that every graph G belonging to S has at least one node with degree less or equal to k.*

PROOF. Let k' be the maximum among the degrees of terminal rules. Then any application of such a rule will produce some nodes with degrees less or equal to k' unless the RR is of the form $B \rightarrow b$ or $N \rightarrow n$. Let k'' be the maximum among the degrees of nonterminal rules. The application of such rules will produce some node structures with degree less or equal to k''. Then the application of the terminal rules $B \rightarrow b$ or $N \rightarrow n$ will result in nodes with degree less or equal to k''. The application of other rewirting rules will result in nodes with degree less or equal to k'. Rules of the form $B \rightarrow B*B$ produce only multiple branches among a pair of nodes and thus do not affect the value of the degree. Therefore there will be always some nodes with degree less or equal to $k = \max(k', k'')$. QED

It is easy to see that the only nodes with unbounded degree are the ones resulting from node structures which have been used as "connecting nodes" during the derivation of the graph.

Theorem 1 gives a necessary but not sufficient condition for the characterization of graphs generated by a 2^d order CFGG. Thus while it is true that the set of planar graphs satisfies this condition[1] we will show later that it cannot be generated by such a grammar. On the other hand it provides an easy test and one can at least be assured that sets of graphs which fail that condition cannot be generated by CFGG's.

COROLLARY 1.1. *The following sets of graphs cannot be generated by a 2^d order CFGG:*

(*a*) *all complete graphs;*

(*b*) *all m-connected graphs (for some finite m)*[2];

(*c*) *all nonseparable graphs.*

We proceed now with another characterization.

Definition 6: A k-reduction process for a graph is defined as follows:

Step (1) Replace by a single node any subgraphs with not more than k nodes which are connected to the rest of the graph by exactly one node.

Step (2) Replace by a single branch any subgraphs with not more than k nodes which are connected to the rest of the graph by exactly two nodes.

Step (3) If no reductions were made in the two previous steps or if the graph has

[1] This is so because every planar graph with more than four nodes has at least four nodes of degree not exceeding five [14].

[2] We will call a connected graph m-connected if m is the minimum number of nodes whose removal makes the graph disconnected, i.e. if its connectivity equals m. This is a slightly different definition than the standard one [14].

been reduced to one with not more than k nodes proceed to the next step. Else return to Step (1).

Step (4) If the reduced graph has not more than k nodes then the original graph is termed k-reducible. Else it is termed k-irreducible.

THEOREM 2. *A family of graphs S is generated by a 2^d order CFGG if and only if there exists a number k depending only on S such that every member of S is k-reducible.*

PROOF. Let k be the maximum number of nodes or node structures in the RHS of the RR's of a CFGG. We may assume without any loss of generality that this grammar contains all possible rules whose RHS has not more than k nodes or node structures (there will be only a finite number of them). The proof is now obvious by observing that step one of the k-reduction process is the inverse of a rule whose LHS is a node structure and step two the inverse of a rule whose LHS is a branch structure. QED

COROLLARY 2.1. *If a set of graphs is generated by a 2^d order CFGG then there exists a number k such that any member of the set with connectivity greater than 2 must have at most k nodes.*

PROOF. Let G be a 3-connected graph. Then except for single branches there is no subgraph which is connected to the rest of the graph by just a node or a pair of nodes. Thus Steps 1 and 2 of the k-reduction process cannot be applied unless G has no more than k nodes. If the latter is not the case then G is k-irreducible. QED

COROLLARY 2.2. *The set of all planar graphs cannot be generated by a 2^d order CFGG.*

PROOF. There exist arbitrarily large 3-connected plane graphs (e.g. wheels).

Montanari [4] has described context sensitive grammars with node rewriting rules to generate the class of planar graphs, nonseparable graphs, etc. The above results indicate that the addition of branch rewriting rules is not enough to change the type of the grammars required to generate them.

The k-reduction process described above is in essence a parsing procedure for graphs. One possible source of trouble in its application can occur if the RHS's of some of the rewriting rules of a CFGG contain cut nodes. For example, a first order CFGG may have an RR of the form:

$$N \rightarrow \underline{N} G_1 n G_2,$$

where G_1 and G_2 contain nonterminal structures. It may happen during the k-reduction procedure that a subgraph corresponding to G_2 is "collapsed" onto the node n. Then the k-reduction process will be one corresponding to a grammar with rewriting rules

$$N \rightarrow \underline{N} G_1 N,$$

$$N \rightarrow \underline{N} G_2 .$$

This will not cause any problems in cases similar to the one occuring in the proof of Theorem 2 where one checks whether a family of graphs can be produced by *some* grammar where the RHS's of the RR's contain not more than k nodes. There the number of steps required for the k-reduction process will not be greater than the number of nodes of the graph Z. However, if one wants to check whether a graph is produced by a specific grammar it will be necessary to apply repeatedly the k-

reduction procedure with different starting cut nodes. Then the number of steps of the k-reduction process will be bounded by Z^2 rather than Z. More precise estimates of the computational complexity of this process should be the subject of future research. These observations may be summarized as follows.

THEOREM 3. *If a set of graphs S is generated by a 2^d order CFGG where the RHS's of the rewriting rules do not contain cut sets consisting of one or two nodes then any graph $G \in S$ can be parsed by a k-reduction procedure in not more than Z steps, where Z is the number of nodes of G. The same limit is true in checking whether a set of graphs is generated by some 2^d order CFGG where the RHS's of the rewriting rules have no more than k nodes.*

There is an interesting subset of planar graphs which can be generated by a 2^d order CFGG, the outerplanar graphs. These are defined as graphs which can be embedded in the plane so that all their nodes lie on the same face [14]. It can be shown easily that such graphs are always 2-connected [14, p. 107].

THEOREM 4. *The following grammar generates all outerplanar graphs and only those:*

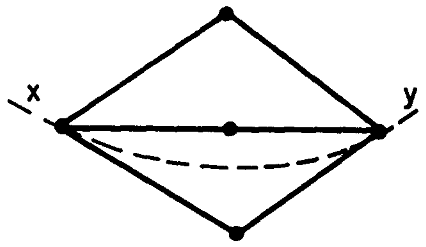
$$H = \{B, b, N, n, P_h, N\},$$
$$P_h = B \rightarrow BNB/b*BNB/b$$

$$\begin{tikzcd} & & N \arrow[dl, dash] \arrow[dr, dash] & & \\ N \rightarrow & B \arrow[d, dash] & & B \arrow[d, dash] & /NBN/n. \\ & N \arrow[r, dash] & B \arrow[r, dash] & N & \end{tikzcd}$$

PROOF. Outerplanar graphs are characterized by not containing homeomorphisms of K_4 and $K_{2,3}$ except $\{K_4\text{-one branch}\}$ [14, p. 107] (Figure 3 shows these graphs). It can be readily verified that none of these graphs can be produced by H and neither can their homeomorphisms. It remains to verify that they cannot be formed as subgraphs by the application of the rules. This can be excluded because the application of a rule leads to a 2-separable graph. K_4 is 3-connected and therefore cannot be part of both components. $K_{2,3}$ can only if divided by a line xy (Figure 4). This can be excluded because the only way to produce "parallel" structures is through rule $B \rightarrow b*BNB$. Thus H cannot generate any non-outerplanar graphs.

To show that H can produce all outerplanar graphs we proceed as follows. Let G be such a graph and let T', T'', $\cdots$ be subgraphs of it which are trees and they are connected to the rest of the graph by their root. Since H contains G_3 the inverse application of the corresponding production rules will result in "collapsing" the trees to their roots.

Let G' be a subgraph "between" two cut nodes P and Q which does not contain any cut nodes. Thus there will be a branch connecting P and Q. G' is also outerplanar. Removing the branch $\overline{PQ}$ results in more cut nodes in the remaining branches of the graph (if this were not the case it would mean the existence of three independent paths between P and Q which contradicts outerplanarity). One can proceed till he finds subgraphs which consist only of chains of nodes and branches. If the path between two nodes involves more than one node these can be "collapsed" into one by the inverse application of the rule $B \rightarrow BNB$. Such a path now together with the original branch between the two nodes can be collapsed into a single branch by the



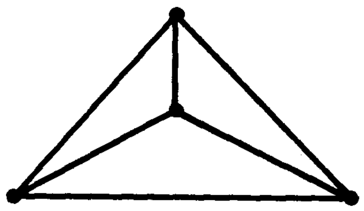


FIG. 4. The graphs K_4 and $K_{2,3}$

inverse application of the rule $B \rightarrow b{*}BNB$. One can thus proceed backwards till the whole graph is reduced to a triangle which is the RHS of an RR. QED

As a final remark we mention that it is possible for a 2d order CFGG to produce strings. This could be done by branch rewriting rules of the form

$$B_i \rightarrow B_{i_1} nB_{i_2} n \cdots nB_{i_k},$$

where B_i, B_{i_1}, B_{i_2}, $\cdots$, B_{i_k} are terminal or nonterminal branch labels. A similar formalism can be obtained by using node rewriting rules. It is obvious that in either case a CFGG reduces to a context-free string grammar.

4. Topological Characterization of Graphs Generated by Linear Graph Grammars

It turns out that the class of graphs generated by LGG's is rather limited. We will give a characterization for them below but first we need a definition.

Definition 7: An interior node of an RR is a node (or node structure) in the graph of the RHS which is not connected directly to any node (or node structure) which is not part of the graph in the RHS of the rule.

THEOREM 5. *A set of graphs S is generated by a 2d order LGG if and only if there exists a number k such that any cut node of the graph or any pair of adjacent cut nodes separate the graph into components such that not more than two of them have more than k nodes.*

PROOF. Let k be the maximum of nodes or node structures in the RHS of the RR's of the grammar. Suppose now that a graph has been generated which contains a nonterminal node structure M. Such a graph will have in general more than k nodes but it will not contain any other nonterminals because of Definition 3. Let us denote this graph by G'. There are two types of RR's which can be applied: $N \rightarrow Ng$ or $N \rightarrow n\hat{G}$ where g denotes a graph without any nonterminals (and hence with at most k nodes) and $\hat{G}$ a graph with exactly one terminal (node or branch structure) different than M. The last type of RR must be applied at least once. Let m be the resulting node from M. It is obvious that m will be a cut node separating the graph into subgraphs G', G'' (which is $\hat{G}$ plus any graphs generated from its nonterminal structure) and $g_1, g_2, \cdots, g_n$. Since the latter have at most k nodes there will be at most two subgraphs with more than k nodes, G' and G''.

A similar argument can be repeated for the case when the nonterminal structure in G' is a branch structure. Thus all nonterminals result in cut nodes (or pairs of adjacent cut nodes). Suppose now that m is an interior cut node on the RHS of an RR. The graph in the RHS of the RR will be connected to the rest of the graph by two cut sets resulting from nonterminal structures. Figure 5 illustrates this situation. The graph inside the dotted lines is the RHS of the RR. Then it is obvious that $G_1' \cup G_1$ and $G_2'' \cup G_2$ will the only components which can have more than k nodes. A similar argument holds for an interior pair of adjacent cut nodes. Thus if a graph is generated by an LGG it has the properties mentioned in the statement of the theorem.

Conversely, suppose that there exists a set of graphs with these properties for some number k. Consider now a linear grammar which contains all the allowable rules whose RHS contains not more than k nodes. Then we can write a generating tree for any given graph by considering all its cut nodes (and pairs of adjacent cut nodes) as nonterminals at some stage. QED

COROLLARY 5.1. *The set of all connected trees cannot be generated by an LGG.*

Note that although the theorem was proved for 2^d^ order LGG's, it is obvious that the last result will be valid for arbitrary order.

COROLLARY 5.2. *The set of all strings over a finite alphabet can be generated by an LGG.*

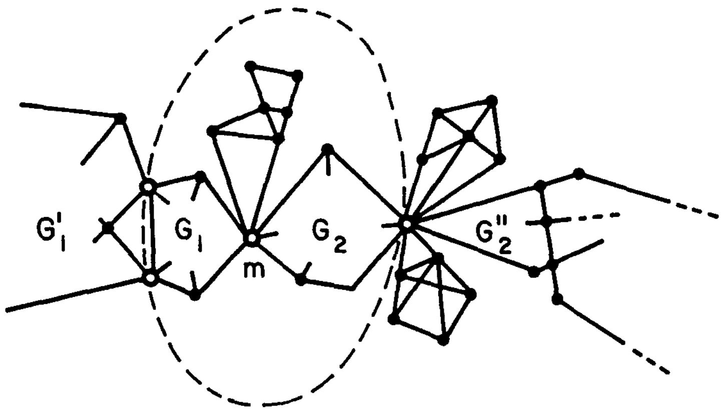


FIG. 5. Components of a graph generated by an LGG

This result, although trivial in itself, illustrates one major difference between string and graph grammars which we mentioned in the beginning of this section. It is always possible to embed a context free string language into a regular set, while this is not the case with a set of graphs generated by a CFGG.

5. *Conclusions*

We have shown that graphs which are generated by CFGG's or LGG's are subject to restrictions regarding the interconnections between different parts of them. Such restrictions are of course nonexistent for string grammars. Consider for example a graph representing a rectangular two dimensional grid as shown in Figure 6. The set of all graphs representing grids of arbitrarily large size cannot be generated by a 2^d order CFGG. Indeed it is easy to verify that for any k, a graph of this type with more than $k + 4$ nodes is not k-reducible. *A fortriori* no such grammar can generate labeled graphs of this form and thus any kind of "pictures." An important class of questions in the literature of computational geometry is whether a context-free or linear grammar can label a given graph in a certain way [16–19]. This is an "easier" task than the one mentioned above and therefore one might expect some simple grammars to produce pictures of interest. Results of this type, proving either the possibility or the impossibility of a given task, have been rather difficult to come by [16–19]. Probably one of the reasons for this is that such proofs must take into account topological constraints rather than "counting" constraints (e.g. "pumping" lemmas) as it is this case for string languages [15].

It is also interesting to notice that we do not have as yet a counterpart of regular sets. The class of graphs generated by 2^d order LGG's is a proper subset of the class generated by 2^d order CFGG's (compare Corollary 5.1 to Example 3) but still is wider than the regular sets as seen by Example 7.

Such constraints seem particularly important only when one wants to consider arbitrarily large graphs. For certain pattern recognition applications it is possible to use preprocessing to transform the original pictures into graphs of reduced size and complexity [9]. Then one can find simple graph grammars generating many interesting classes of patterns. However, in this case one is faced with the question of whether the grammatical approach results in a better algorithm than an ad hoc graph analysis. For example, in some cases cycles in a graph correspond to "holes" in the original picture [9]. Thus deciding whether a picture is simply connected is under

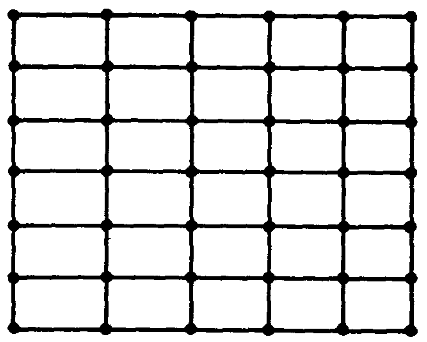


FIG. 6. The graph of a rectangular grid

certain conditions equivalent to deciding whether the corresponding graph is 1-connected, i.e. it is a tree. In the linguistic approach this would mean to test whether the graph in question can be parsed according to the CFGG G_3 (described in Section 2). The classical approach requires finding a spanning tree and then checking whether there are any branches left [14]. It seems that the latter approach results in algorithms which are at least as simple as any parsing scheme

Finally, we might mention that all the results of the previous sections can be extended readily to directed graphs.

REFERENCES

1. MILLER, W. F., AND SHAW, A. C. Linguistic methods in picture processing—a survey. *Proc. FJCC, Vol. 33* (1968), pp. 279–290.
2. PFALTZ, J. L., AND ROSENFELD, A. Web grammars. *Proc. Joint Intern. Conf. on Artificial Intelligence*, Washington, D. C., May 1969.
3. SHAW, A. C. Parsing of graph-representable pictures. *J.ACM 17* (July 1970), 453–481.
4. MONTANARI, U. G. Separable graphs, planar graphs and web grammars. *Inform. Cont. 16* (May 1970), 243–267.
5. PFALTZ, J. L. Web grammars and picture description. Tech. Rep. 70–138, Comp. Sci. Center, U. of Maryland, Sept. 1970.
6. FU, KING-SUN, AND SWAIN, P. H. On syntactic pattern recognition. In *Software Engineering, Vol. 2* (J. Tou, Ed.), Academic Press, New York, 1971, pp. 155–182.
7. ROSENFELD, A., AND STRONG, J. P. A grammar for maps. In *Software Engineering, Vol. 2* (J. Tou, Ed.), Academic Press, New York, 1971, pp. 227–239.
8. SHAW, A. C. Picture graphs, grammars and parsing. Tech. Rep. No. 71-89, Dep. of Comp. Sci., Cornell U., Jan. 1971 (presented at the International Conference on Frontiers of Pattern Recognition, Honolulu, Hawaii, Jan. 1971).
9. PAVLIDIS, T. Structural pattern recognition: Primitives and juxtaposition relations. Tech. Report No. 89, Comp. Sci. Lab., Princeton U., Jan. 1971 (presented at the International Conference on Frontiers of Pattern Recognition, Honolulu, Hawaii, Jan. 1971).
10. FEDER, J. Plex languages. *Inform. Sci. 3* (1971), 225–241.
11. ROSENFELD, A. Isotonic grammars, parallel grammars and picture grammars. In *Machine Intelligence, Vol. 6, 1971.* Edinburgh U. Press, Edinburgh, 1971, pp. 281–294.
12. ROUNDS, W. C. Context-free grammars on trees. *Conf. Rec. ACM Symp. on Theory of Computing.* ACM, New York, 1969, pp. 143–148.
13. THATCHER, J. W. There's a lot more to finite automata theory than you would have thought. *Proc. 4th Annual Princeton Conference*, March 1970, pp. 263–276.
14. HARARY, F. *Graph Theory.* Addison-Wesley, Reading, Mass., 1969.
15. HOPCROFT, J. E., AND ULLMAN, J. D. *Formal Languages and Their Relation to Automata.* Addison-Wesley, Reading, Mass., 1969.
16. BLUM, M., AND HEWITT, C. Automata on a two-dimensional tape. *Proc. 8th IEEE Conference on Switching and Automata Theory*, 1968, pp. 155–160.
17. MINSKY, M., AND PAPERT, S. *Perceptrons.* M.I.T. Press, Cambridge, Mass., 1969.
18. MYLOPOULOS, J. On the definition and recognition of patterns in discrete spaces. Tech. Rep. No. 84, Comp. Sci. Lab., Princeton U., Aug. 1970.
19. MILGRAM, D. L., AND ROSENFELD, A. Array automata and array grammars. Tech. Rep. No. 70–141, Comp. Sci. Center, U. of Maryland, Nov. 1970.

RECEIVED MARCH 1971; REVISED JULY 1971