On the complexity of graph grammars

By GY. TURÁN

1. Introduction

Graph grammars generalize "usual" word grammars by considering graphs as basic objects instead of words. A derivation step consists of the replacement of a subgraph by another graph. The delicate part of the definition of graph grammars is the way the embedding of the new graph is specified (in the string case this problem does not appear).

This generalization appears to be quite natural and it has applications, too (Nagl [4]). However "nice" results of formal language theory characterizing classes of languages from different aspects (grammars, automata, algebraic and logical descriptions) does not seem to generalize for the case of graphs. (There are two possible explanations: either the right definitions are not found yet and one should not only try to generalize notions of string grammars, or the situation is indeed different.)

Another problematic aspect of graph grammars is the parsing of graph grammars (this is the topic we are going to discuss so we return to it later).

There are several theorems in graph theory describing a class of graphs as the class obtainable from a set of start graphs applying a finite set of operations (e.g. the theorem of Tutte on 3-connected graphs [7]). These theorems can actually be considered as positive results about graph grammars. Understanding the power of these operations could be of interest for graph theory as well. To return the problem of right definitions we remark that interesting operations of graph theory (e.g. Hajós' operations to generate non-k-colourable graphs [1]) quite often do not fit into the present framework of graph grammars.

As to our knowledge there are few results about the parsing of graph grammars. Grammars investigated are usually generalizations of context-free grammars, thus it is natural to try to generalize context-free parsing for the case of graphs. In the paper of Vigna—Ghezzi [8] it is shown that a certain parsing technique is exponential for their class of grammars and polynomial if further restrictions are imposed. Slisenko [6] gives a rather restricted class of grammars that can be parsed in polynomial time by a similar method (in fact he shows more: Hamiltonian cycles can be found in polynomial time when restricted to a context-free graph language). Results

of Janssens and Rozenberg [2] show that parsing their *node label controlled* (NLC) grammars is as hard as context-sensitive recognition.

One can ask the following questions about the parsing of graph grammars:
— what general parsing techniques exist?
— where is the borderline between "easy" and "hard" classes of grammars?

The theorem proved in this paper gives a step towards answering the second question. We introduce a natural restriction of NLC grammars by requiring graphs on the right-hand side of productions to consist of more than one vertex. Languages generated by these grammars are always in NP. We show that these grammars are strong enough to generate NP-complete languages. Thus no efficient parsing technique can be expected that is applicable for monotone NLC grammars.

2. Monotone node label controlled grammars

Following Janssens and Rozenberg [2] we define node label controlled (NLC) graph grammars as follows.

Definition. An NLC grammar $\mathcal{G}$ is a quintuple

$$(\Sigma, \Delta, G_0, \mathcal{P}, \mathcal{C})$$

where Σ is the (finite, nonempty) nonterminal alphabet; Δ is the (finite, nonempty) terminal alphabet (disjoint from Σ); G_0 is the start graph; $\mathcal{P}$ is the set of productions: a production P is a pair (a_i, G_i) where $a_i \in \Sigma$, G_i is a graph; $\mathcal{C}$ is the connection relation: $\mathcal{C} \subseteq (\Sigma \cup \Delta) \times (\Sigma \cup \Delta)$.

REMARK. Graphs considered are undirected, without loops and multiple edges. Edges are unlabeled, vertices are labeled with labels from $\Sigma \cup \Delta$.

Derivations and the language $L(\mathcal{G})$ generated by $\mathcal{G}$ are only described informally (see [2] for exact definitions). When applying a production $P=(a_i, G_i)$ to a graph G, we replace a vertex v_1 labeled with a_i by a graph isomorphic to G_i. If v_2 is a neighbour of v_1 in G labeled a_2 and v_3 is a vertex of G_i labeled a_3 then in the new graph G' v_2 and v_3 will be connected if and only if $(a_3, a_2) \in \mathcal{C}$ (see Fig. 1). Thus the embedding is controlled by the node labels only. $L(\mathcal{G})$ consists of graphs derivable from G_0 with all vertex labels belonging to Δ.

Definition. A *monotone* NCL *grammar* is an NLC grammar satisfying the following condition:

For every production $P=(a_i, G_i)$ the number of vertices of G_i is more than one.

This is the class of graph grammars we consider from the point of view of the complexity of languages generated.

3. The complexity of monotone NLC grammars

Proposition. If $\mathcal{G}$ is a monotone NLC grammar, than $L(\mathcal{G})$ is in NP (where, as usual, NP denotes the class of languages recognizable by nondeterministic Turing-machines in polynomial time).

Proof. As for every production $P=(a_i, G_i)$ G_i consists of more than one vertex, if a graph G has a derivation in $\mathcal{G}$ then the length of the derivation is at most $n-1$ where n is the number of vertices of G. Thus the derivation can be guessed and checked in polynomial time. □

Theorem. There exists a monotone NLC grammar $\mathcal{G}$ such that $L(\mathcal{G})$ is NP-complete.

Before turning to the proof we describe a property of graphs that will be used later on.

A graph $G=(V, E)$ has *cyclic bandwidth* $\leqq k$ if there exists a cyclic ordering $(v_1, \ldots, v_n)$ of the vertices s.t. if $(v_i, v_j)\in E$ then the cyclic distance of v_i and v_j is at most k. (The cyclic distance of v_i and v_j $(i<j)$ is $\min(j-i, n+i-j)$.)

The class of graphs with cyclic bandwidth $\leqq k$ is denoted by CB_k (here graphs are considered without vertex labels). The following result is mentioned in Johnson [3].

Theorem (Leung—Vornberger—Withoff). CB_2 is NP-complete. □

This result can be compared with the complexity of bandwidth (i.e. considering orders instead of cyclic orders): for any fixed k it can be decided in polynomial time whether the bandwidth of a graph is $\leqq k$ (Saxe [5]).

Now we give an informal description of our construction. By G_n we denote the graph on n vertices consisting of a cycle of length n and edges connecting vertices of distance 2 on the cycle. G_7 is shown on Fig. 2.

Every graph G on n vertices with cyclic bandwidth $\leqq 2$ is a subgraph of G_n. Thus G can be constructed by building G_n and then deleting the edges of G_n not belonging to G. G_n can be constructed by building a chain (shown on Fig. 3) and then closing the chain.

However, in order to be able to close the chain generated by an NLC grammar the edges connecting the "open" end of the chain with the "beginning" of the chain must always be present during the derivation and edges unnecessary after closing the chain must be forced to be deleted. These requirements can be fulfilled by defining the connection relation appropriately.

We remark that the grammar G used to prove the theorem is a rather large one, we did not try to make it as small as possible. Instead, we tried to make it easy to describe and analyze.

Proof of the theorem. First we describe the grammar $\mathcal{G}$ generating an NP-complete language.

The description of $\mathcal{G}$.

1) The nonterminal alphabet.

$$\Sigma = \{S, A_1, A_2, A_3, A_4, A_1', A_2', A_3', A_4'\} \cup \bigcup_{i=1}^{6} \mathcal{H}_i,$$

where

$$\mathcal{H}_i = \{B_i, C_i, D_i, E_i\} \cup \{C_i^{jk}, D_i^{jkm}, E_i^{jklm} : 0 \leqq j, k, l, m \leqq 1\}.$$

2) *The terminal alphabet.* $\Delta = \{x, y, z\}$.

3) *The start graph.* $G_0 = S$.

4) *The productions.* There are five groups of productions each playing different roles in the construction.

4.a) *Starting productions.* These productions can be applied at most once in every derivation and exactly one of them must be used in every derivation as a first step.

Consider the graph of Fig. 4 with 5 marked edges. Deleting all different subsets of these edges we get 32 graphs $H_1, \ldots, H_{32}$. The starting productions are of the form

$$S \Rightarrow H_i \quad \text{for} \quad i = 1, \ldots, 32.$$

4.b) *Chain-constructing productions.*

$$B_i \quad C_i \quad B_{i+1} \qquad \text{for} \quad i = 1, \ldots, 6.$$

Here and everywhere else in the construction addition and subtraction is meant cyclically, e.g. $6+1=1$. Using the productions belonging to this group a chain of arbitrary length can be generated. We use the following terminology: such a production *relabels the vertex labeled* B_i *by* C_i *and adds a new vertex labeled* B_{i+1}.

4.c) *Chain-closing productions.*

$$B_i \quad D_i \quad E_{i+1} \qquad \text{for} \quad i = 1, \ldots, 6.$$

The role of these productions is to close the chain generated by applying productions belonging to the previous group. Informally such a production relabels the vertex labeled B_i by D_i and adds a new vertex labeled E_{i+1} that becomes the last vertex of the chain.

4.d) *Edge-deleting productions.*

$$C_i \quad C_i^{jk} \quad y$$

$$D_i \quad D_i^{jkm} \quad y$$

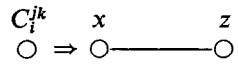
$$E_i \quad E_i^{jklm} \quad y \qquad \text{for} \quad i = 1, \ldots, 6, \quad 0 \leqq j, k, l, m \leqq 1.$$

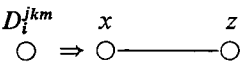
The role of these productions is to realize the deletion of edges. Vertex labeled C_i (resp. D_i, E_i) is relabeled C_i^{jk} (resp. D_i^{jkm}, E_i^{jklm}) and a new vertex labeled y is added. The binary vector (j, k, l, m) indicates the set of edges to be deleted (in G_n every vertex has degree 4).

$$A_i \quad A_i' \quad y \qquad \text{for} \quad i = 1, 2, 3, 4.$$

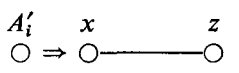
The role of these productions is to force the deletion of unnecessary edges if vertices labeled A_i "become terminal vertices too soon".

4.e) *Terminal productions.*





for $i = 1, \ldots, 6, \quad 0 \leqq j, k, l, m \leqq 1;$



for $i = 1, 2, 3, 4.$

5. *The connection relation.* As it is remarked already, the main regulating role in the derivation is played by the connection relation. Pairs belonging to the relation are divided into four groups.

5.a) *Pairs regulating the construction of the chain.*

$$(C_i, C_{i-1}),\ (C_i, C_{i-2}),\ (C_i, A_1),\ (C_i, A_2),$$

$$(B_i, C_{i-2}),\ (B_i, A_1),\ (B_i, A_2) \quad \text{for} \quad i = 1, \ldots, 6.$$

5.b) *Pairs regulating the closure of the chain.*

$$(D_i, C_{i-1}),\ (D_i, C_{i-2}),\ (D_i, A_1),\ (D_i, A_2)$$

$$(E_i, C_{i-2}),\ (E_i, A_1),\ (E_i, A_2) \quad \text{for} \quad i = 1, \ldots, 6.$$

5.c) *Pairs regulating the deletion of edges.*

$$(C_i^{jk}, N_{i+\delta}) \quad \text{for every} \quad N_{i+\delta} \in \mathscr{H}_{i+\delta}$$

if $(j = 1$ and $\delta = -2)$ or $(k = 1$ and $\delta = -1)$, or $(\delta = 1)$, or $(\delta = 2)$;

$$(D_i^{jkm}, N_{i+\delta}) \quad \text{for every} \quad N_{i+\delta} \in \mathscr{H}_{i+\delta}$$

if $(j = 1$ and $\delta = -2)$ or $(k = 1$ and $\delta = -1)$, or $(\delta = 1)$;

$$(E_i^{jklm}, N_{i+\delta}) \quad \text{for every} \quad N_{i+\delta} \in \mathscr{H}_{i+\delta}$$

if $(j = 1$ and $\delta = -2)$ or $(k = 1$ and $\delta = -1)$;

$$(C_1^{jk}, A_3),\ (C_1^{jk}, A_3') \quad \text{if} \quad j = 1;$$

$$(C_1^{jk}, A_4),\ (C_1^{jk}, A_4') \quad \text{if} \quad k = 1;$$

$$(C_2^{jk}, A_4),\ (C_2^{jk}, A_4') \quad \text{if} \quad j = 1;$$

$$(D_i^{jkm}, A_1),\ (D_i^{jkm}, A_1') \quad \text{if} \quad m = 1;$$

$$(E_i^{jklm}, A_1),\ (E_i^{jklm}, A_1') \quad \text{if} \quad l = 1;$$

$$(E_i^{jklm}, A_2),\ (E_i^{jklm}, A_2') \quad \text{if} \quad m = 1.$$

5.d) *Additional pairs*

$$(N, x) \quad \text{for every} \quad N \in \bigcup_{i=1}^{6} \mathcal{H}_i;$$

$$(x, M) \quad \text{for every} \quad M \in \Sigma \cup \Delta;$$

$$(A_i', x) \quad \text{for} \quad i = 1, 2, 3, 4;$$

$$(A_i', A_j) \quad \text{for} \quad 1 \leqq i, j \leqq 4.$$

Let G be an arbitrary graph without vertex labels. Define a graph G^* with vertices labeled x, y, z as follows:

1) label the vertices of G with x,
2) join two different vertices to each vertex of G and label them y and z respectively.

(An example is shown on Fig. 5.)

The theorem will be proved if we prove the following claim.

Claim. $L(\mathcal{G}) = \{G^*: G \in \mathrm{CB}_2$ and G has $\geqq 8$ vertices$\}$.

First we show the $\supseteq$ part of the claim.

Let $G \in \mathrm{CB}_2$ be a graph on $\geqq 8$ vertices. We describe a derivation of G^*.

Take a suitable circular order $(v_1, \ldots, v_n)$ of the vertices of G having circular bandwidth $\leqq 2$. Consider vertices $v_1, \ldots, v_7$ and label them $A_1, \ldots, A_4, C_1, C_2, B_3$. Take the subgraph spanned by $v_1, \ldots, v_7$ and add edges

$$(v_5, v_1),\ (v_5, v_2),\ (v_6, v_1),\ (v_6, v_2),\ (v_7, v_1),\ (v_7, v_2),$$

$$(v_5, v_6),\ (v_5, v_7),\ (v_6, v_7)$$

if they are not present yet.

The derivation of G^*.

1) Apply a suitable starting production to obtain the labeled graph on vertices $v_1, \ldots, v_7$ described above.

2) Apply chain-constructing productions $n-8$ times. (The applicable production is always unique.)

3) Apply a chain-closing production. (The applicable production is unique.)

4) For each vertex v_k, $5 \leqq k \leqq n$ define the binary vectors $\mathbf{v}_k := (i_{-2}^{(k)}, i_{-1}^{(k)}, i_1^{(k)}, i_2^{(k)})$; $\mathbf{v}_k' := (i_{-2}^{(k)}, i_{-1}^{(k)}, i_2^{(k)})$; $\mathbf{v}_k'' := (i_{-2}^{(k)}, i_{-1}^{(k)})$ where $i_j^{(k)} = 1 \Leftrightarrow (v_k, v_{k+j}) \in E$. For $k = n, n-1, \ldots, 5$ apply

$E_i \quad E_i^{\mathbf{v}_k} \quad y$
$\circ \Rightarrow \circ\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!\circ$ if v_k is labeled E_i;

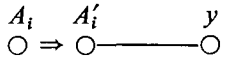
$D_i \quad D_i^{\mathbf{v}_k'} \quad y$
$\circ \Rightarrow \circ\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!\circ$ if v_k is labeled D_i;

$C_i \quad C_{i_1}^{\mathbf{v}_k''} \quad y$
$\circ \Rightarrow \circ\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!\circ$ if v_k is labeled C_i.

5) Apply terminal productions for each vertex labeled

$$E_i^{\mathbf{v}_k},\ D_i^{\mathbf{v}_k'} \quad \text{or} \quad C_i^{\mathbf{v}_k''}.$$

6) For $k=4, 3, 2, 1$ apply



7) Apply terminal productions for each vertex labeled A_i'.

After steps 1), 2), 3) we generated a labeled graph G' shown on Fig. 6, where $k \equiv n \pmod 6$. Vertices labeled A_1, A_2 are connected to every other vertex labeled C_i, D_i or E_i.

In step 4) unnecessary edges are deleted from vertices labeled C_i^{jk}, D_i^{jkm}, E_i^{jklm} and pendant vertices labeled y are added. As $A_i \notin \mathscr{H}_i$ and $A_i' \notin \mathscr{H}_i$, all edges connecting A_1 and A_2 to vertices labeled C_i^{jk} disappear and only the necessary edges connecting A_1 and A_2 to vertices labeled D_i^{jkm}, E_i^{jklm} remain in the graph. In step 5) vertices labeled C_i^{jk}, D_i^{jkm}, E_i^{jklm} are relabeled x and their adjacencies are not changed. In step 6) vertices labeled A_i are relabeled A_i' and pendant vertices labeled y are added. No change is made in this step in the edges, as edges between vertices labeled A_i and vertices outside the set of vertices labeled A_i are already disposed of, and internal edges are chosen correctly by the choice of the starting production. Finally in step 7) vertices labeled A_i originally get label x and pendant vertices labeled z are added.

Now we turn to the $\subseteq$ part of the claim.

Let G be a graph belonging to $L(G)$. We use the "relabeling" terminology introduced at the description of the productions. By the *history* of a vertex v we mean the sequence of labels appearing on v. The histories possible are the following.

1) B_i, C_i, C_i^{jk}, x for some i, j, k;

2) C_i, C_i^{jk}, x for $i = 1, 2$ and some j, k;

3) B_i, D_i, D_i^{jkm}, x for some i, j, k, m;

4) E_i, E_i^{jklm}, x for some i, j, k, l, m;

5) A_i, A_i', x for some i;

6) y;

7) z.

(The exceptional case 2 refers to the vertices labeled C_1, C_2 of the graph generated by the starting production.)

The graph generated (not considering vertex labels) will always be a subgraph of G_n'' of Fig. 7 for some n.

(The nonterminal label B_i can be replaced by C_i or D_i, and a new nonterminal B_{i+1} will appear in the graph unless B_i is replaced by D_i. Thus the generation of new vertices labeled B_i, C_i, D_i or E_i must end with a chain-closing production. Edge-deleting and terminal productions can be applied to nonterminals already present in the graph, thus making progress in the histories of each vertex, but these productions do not introduce new edges as $(y, .)$ and $(z, .)$ is not in the connection relation.)

The last point we have to check is that forbidden edges connecting vertices labeled originally A_1, A_2 and vertices ever labeled C_i do actually disappear. This can be shown considering the histories (C_i, C_i^{jk}, x) and (A_s, A_s', x) $(s=1, 2)$. There are no pairs (C_i^{jk}, A_s) or (A_s', C_i) in the connection relation so the edge (C_i, A_s) disappears whichever of the two histories makes progress first. The same holds for the pair (A_2, D_i).

Thus the second half of the claim is proved.

Finally it is obvious that CB_2 can be reduced to $L(\mathcal{G})$ in polynomial time by forming graphs G^*. □

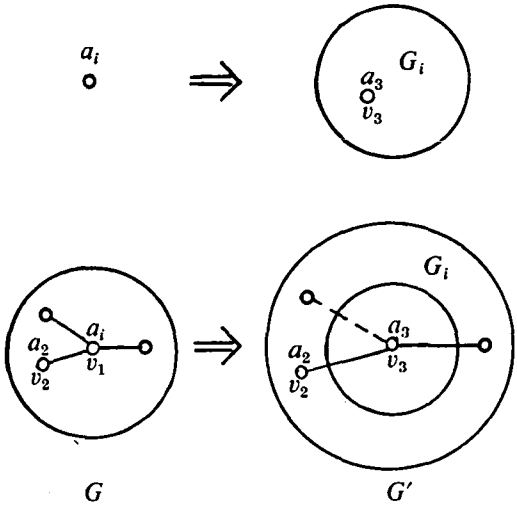


Fig. 1

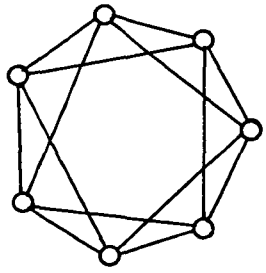


Fig. 2

Fig. 3

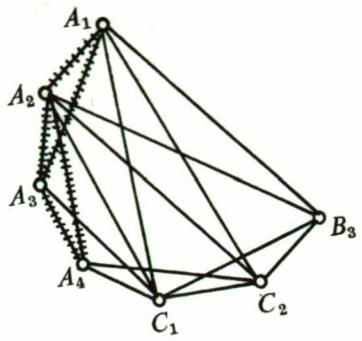


Fig. 4

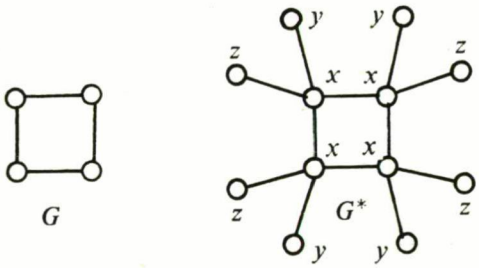


Fig. 5

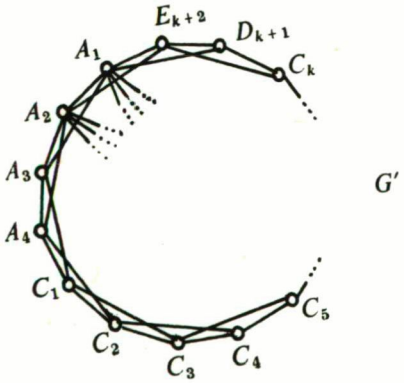


Fig. 6

G'

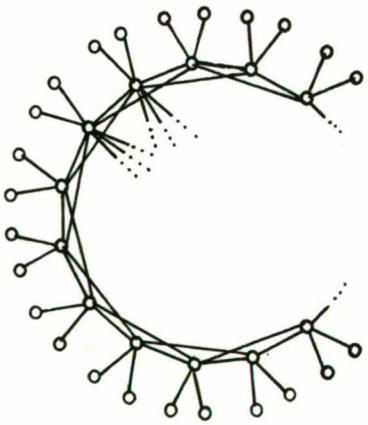


Fig. 7

Abstract

A problem in the theory of graph grammars is the following: for what classes of grammars can the languages generated be parsed in polynomial time? It is shown that a grammar belonging to a rather restricted class, the monotone node label controlled grammars can be strong enough to generate an NP-complete language.

RESEARCH GROUP ON THEORY OF AUTOMATA
HUNGARIAN ACADEMY OF SCIENCES
SOMOGYI U. 7
SZEGED, HUNGARY
H-6720

References

[1] HAJÓS, G., Über ein Konstruktion nicht *n*-färbbarer Graphen, *Wiss. Z. Martin-Luther Univ. Halle—Wittenberg Math.-Natur. Reihe*, v. 10, 1961, pp. 116—117.
[2] JANSSENS, D., G. ROZENBERG, Decision problems for node label controlled graph grammars, *J. Comput. System Sci.*, v. 22, 1981, pp. 147—177.
[3] JOHNSON, D. S., The NP-completeness column: an ongoing guide, *J. Algorithms*, v. 3, 1982, pp. 288—300.
[4] NAGL, M., Graph-Grammatiken, Theorie, Implementierung, Anwendungen, Vieweg, 1979.
[5] SAXE, J. B., Dynamic programming algorithms for recognizing small-bandwidth graphs in polynomial time, *SIAM J. Algebraic Discrete Methods*, v. 1, 1980, pp. 363—369.
[6] SLISENKO, A. O., Context-free grammars as a tool for describing polynomial-time subclasses of hard problems, *Inform. Process. Lett.*, v. 14, 1982, pp. 52—56.
[7] TUTTE, W. T., A theory of 3-connected graphs, *Indag. Math.*, v. 23, 1961, pp. 441—455.
[8] VIGNA, P. D., C. GHEZZI, Context-free graph grammars, *Inform. and Control.*, v. 37, 1978, pp. 207—233.

(Received Oct. 21, 1982)