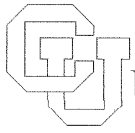
Graph Grammars with Node-Label Controlled Rewriting and Embedding

D. Janssens
G. Rozenberg

CU-CS-251-83



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

GRAPH GRAMMARS WITH NODE-LABEL CONTROLLED REWRITING AND EMBEDDING

by

D. Janssens and G. Rozenberg

CU-CS-251-83 July, 1983

*Institute of Applied Mathematics and Computer Sicence, University of Leiden, Wassenaarseweg 80, Leiden, The Netherlands

GRAPH GRAMMARS WITH NODE-LABEL CONTROLLED REWRITING AND EMBEDDING

by

D. Janssens
Institute of Applied Mathematics
and Computer Science,
University of Leiden,
Leiden, The Netherlands

G. Rozenberg
Institute of Applied Mathematics
and Computer Science,
University of Leiden,
Leiden, The Netherlands

INTRODUCTION

The theory of graph grammars is an area within theoretical computer science that is mathematically very interesting and challenging and at the same time it is an area with many fields of potential applications such as data bases, compiler techniques, syntax and semantics of programming languages, data flow analysis, concurrency, pattern recognition and developmental biology; this is well reflected in [CER], [N] and in the present volume. Although there exists quite a body of literature concerning the mathematical theory of graph grammars, it is not yet as well developed as the theory of string grammars; a partial explanation of this situation may be the objective fact that the theory of graph grammars seems to be more "difficult" than the corresponding theory of string grammars (after all, the structure of a graph is more involved than the structures of a string).

There certainly is a great need for fundamental research on graph grammars - in particular there is a need for a systematic framework for graph grammars within which various issues can be discussed and compared. A frequent situation within the existing theory is that various issues are investigated within various models, a situation that is often undesirable. NLC grammars were introduced (see [JR1] and [JR2]) as a first step towards such a systematic theory. Our approach is "bottom-up"; that is we would like to build up a "solid" theory of NLC grammars and languages, and once this is achieved (to some degree) we consider various extensions and variations of the basic model so that eventually a quite general (and mathematically solid) framework for graph grammars based on the NLC model will emerge. In particular we hope that in this way it will become more clear what are the central notions of the theory and which mathematical means are needed to investigate them.

In the last few years a number of basic and quite different issues were investigated within the theory of NLC grammars. In the present paper we will survey some of them trying to convey to the reader the "flavour" of the theory. Because of the

restrictions on the size of the paper our survey has a rather informal and sketchy character; however the interested reader can find the technical (formal) details in the references indicated. Also, the Ph.D. Thesis by D. Janssens (to appear) gives a rather complete picture of the theory of NLC grammars.

1. BASIC DEFINITIONS

We assume the reader to be familiar with the basic formal language theory and the basic graph theory. We recall now a number of basic mathematical notations.

For sets A, B, #A denotes the cardinality of A, $A \cap B$ denotes the intersection of A and B and A-B denotes the difference of A and B; $\emptyset$ denotes the empty set.

A node-labelled undirected graph is specified as a 4-tuple (V,E,Σ,φ) where V is the set of nodes, E is the set of edges, Σ is the set of labels and φ is the labelling function (of nodes by labels). Accordingly, for a node-labelled undirected graph M, V_M denotes its set of nodes, E_M its set of edges, Σ_M its set of labels and φ_M its labelling function. For an alphabet Δ, G_Δ denotes the family of all node-labelled undirected graphs H such that $\Sigma_M = \Delta$.

In the sequel a node-labelled undirected graph will be referred to simply as a graph.

If M is a graph and $\bar{M}$ is a full subgraph of M, then $M\backslash\bar{M}$ denotes the full subgraph of M spanned on the node set $V_M - V_{\bar{M}}$. For a graph M and subgraphs Q,R of M, $M(Q,R) = \{(u,v) \mid \{u,v\} \in E_M,\ u \in V_Q \text{ and } v \in V_R\}$.
diam(M) denotes the diameter of a graph M and und(M) denotes the unlabelled version of M, that is $\underline{und}(M) = (V_M,E_M)$.

A string grammar operates by rewriting strings into strings - a graph grammar operates by writing graphs into graphs. In the present paper we will consider sequential graph grammars in which in a direct derivation step one production is applied to one occurrence of its left-hand side.
A direct derivation step in a (sequential) graph grammmar consists of two stages:
(1) a subgraph (called a mother graph) of a graph (called a host graph) is replaced (rewritten) by applying a production and then (2) the graph (called the daughter graph) that has replaced the mother graph is embedded in the rest of the host graph (that is in the host graph with the mother graph removed). The first part of this process is completely analogous to the one performed in string grammars, the second part - the embedding - is intrinsic to graph grammars; strings can be viewed as graphs with a very uniform rigid stnucture so that the embedding process takes place "automatically" (one does not need a special mechanism to specify it).

In specifying a (new) model of a graph grammar one has to specify both its rewriting and its embedding part. In the basic model we will discuss in this paper we assume the following restrictions:

(1) on rewriting:

(1.1) our graph grammars will be node rewriting,

(1.2) the rewriting of a node depends on its label only;

(2) on embedding:

(2.1) we will assume that the embedding rule is global, i.e., the embedding mechanism is provided for the grammar and not for each of its productions individually,

(2.2) only nodes that are direct neighbours of the mother node are being considered for the connection with nodes of the daughter graph,

(2.3) whether a candidate pair of nodes (a node from the daughter graph and a node which is a direct neighbour of the mother graph) gets connected depends solely on the labels of these nodes.

Hence in this model both the rewriting and the embedding process are controlled by labels of the nodes involved.

Formally our model is defined as follows:

Definition 1.1. (i) A node-label controlled graph grammar (abbreviated NLC grammar) is a system $G = (\Sigma,\Delta,P,C,Z)$ where Σ is a finite nonempty set (called the total alphabet), Δ is a finite nonempty subset of Σ (called the terminal alphabet), P is a finite set of pairs of the form (d,D) with $d \in \Sigma$ and $D \in G_\Sigma$ (called the set of productions), C is a subset of $\Sigma \times \Sigma$ (called the connection relation) and Z is a graph over Σ (called the axiom).

(ii) Let $M, \bar{M} \in G_\Sigma$, $v \in V_M$ and $\bar{D} \in G_\Sigma$. Then M concretely directly derives $\bar{M}$ replacing v by $\bar{D}$ (denoted $M \underset{(v,D)}{\Longrightarrow} \bar{M}$) if

(1) $\bar{D}$ is a full subgraph of $\bar{M}$ and $V_{\bar{D}} \cap V_M = \emptyset$,

(2) there exists $(d,D) \in P$ such that $\varphi_M(v) = d$ and D is isomorphic to $\bar{D}$,

(3) $\bar{M}\backslash\bar{D} = M\backslash\{v\}$ and

(4) $\bar{M}(\bar{D},\bar{M}\backslash\bar{D}) = \{(x,y) \mid x \in V_{\bar{D}},\ y \in V_{\bar{M}\backslash\bar{D}},\ \{v,y\} \in E_M \text{ and } (\varphi_{\bar{D}}(x), \varphi_M(y)) \in C\}$.

(iii) Let $M,\bar{M} \in G_\Sigma$. Then M concretely directly derives $\bar{M}$ (denoted $M \Longrightarrow \bar{M}$) if $M \underset{(v,\bar{D})}{\Longrightarrow} \bar{M}$ for some $v \in V_M$ and $\bar{D} \in G_\Sigma$.

(iv) Let $M,\bar{M} \in G_\Sigma$. Then M directly derives $\bar{M}$ (denoted $M \overset{\cdot}{\Rightarrow}_G \bar{M}$) if there exists a $\hat{M}$ such that $\hat{M}$ is isomorphic to $\bar{M}$ and $M \underset{G}{\Longrightarrow} \hat{M}$.

(v) The relation $\overset{*}{\Rightarrow}_G$ is the reflexive and transitive closure of $\overset{\cdot}{\Rightarrow}_G$ (if, for $M,\bar{M} \in G_\Sigma$, $M \overset{*}{\Rightarrow}_G \bar{M}$ then we say that M derives $\bar{M}$).

(vi) The exhaustive language of G (denoted $S(G)$) is the set $\{M \in G_\Sigma : Z \overset{*}{\Rightarrow}_G M\}$. □

By iterating the direct derivation step in our NLC grammar G one gets the set of (all) graphs that the grammar generates - this is its exhaustive language $S(G)$. Now based on those languages one can obtain - by using different "language squeezing" mechanisms - various other kinds of languages associated with G. There are also other important (especially the first one) kinds of languages one wants to associate with (define by) an NLC grammar.

Definition 1.2. Let $G = (\Sigma,\Delta,P,C,Z)$ be an NLC grammar.

(1) The language of G (denoted L(G)) is the set $\{M \in G_\Delta : Z \overset{*}{\underset{G}{\Rightarrow}} M\}$.

We use L(NLC) to denote the class of graph languages generated by NLC grammars.

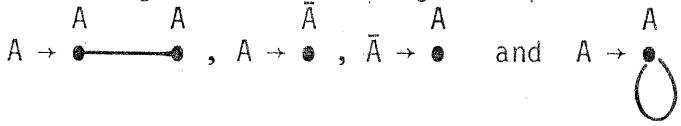
(2) The connected language of G (denoted $L_{con}(G)$) is the set $\{M \in G_\Delta : Z \overset{*}{\underset{G}{\Rightarrow}} M$ and M is connected$\}$. □

The structure of (graphs in) L(G) is provided by the language of unlabelled graphs obtained from graphs in L(G) by removing their node labels; this "structure language" is denoted by und(L(G)).

Example 1.1. Let $G = (\Sigma,\Delta,P,C,Z)$ be the NLC grammar with $\Sigma = \{A,\bar{A}\}$, $\Delta = \{A\}$, $Z =$ 

, the connection relation C given by the following directed graph

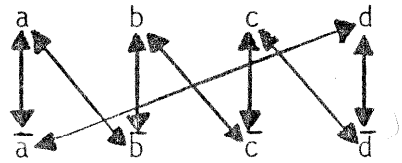
and P consisting of the following four productions:



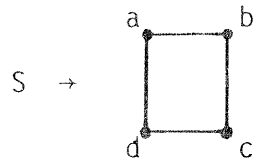
Then $L(G) = G_\Delta$. □

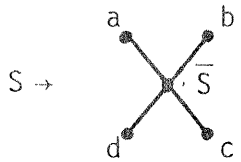
Example 1.2. Let $G = (\Sigma,\Delta,P,C,Z)$ be the NLC grammar with $\Sigma = \{a,b,c,d,\bar{a},\bar{b},\bar{c},\bar{d},S,\bar{S}\}$, $\Delta = \{a,b,c,d,\bar{a},\bar{b},\bar{c},\bar{d}\}$, $Z =$ 

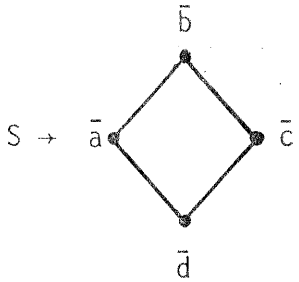
, the connection relation C given by the following directed graph

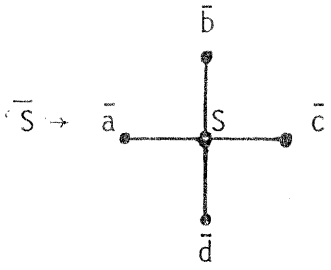


and P consists of the following four productions

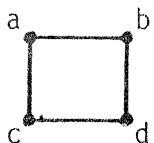


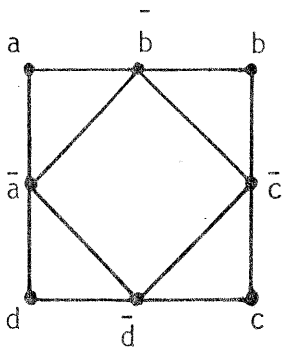




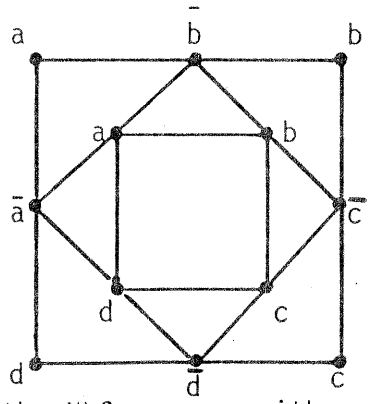


It is easy to see that L(G) consists of all graphs of the form:





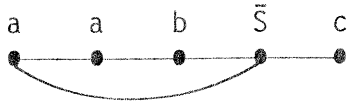
,



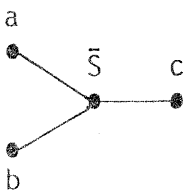
, ... □

Example 1.3. Let $G = (\Sigma,\Delta,P,C,Z)$ be the NLC grammar with $\Sigma = \{a,b,c,S,\bar{S}\}$, $\Delta = \{a,b,c\}$, $Z =$ S, $C = \{(a,a),(b,b),(c,c)\}$ and P consists of the following four productions

$S \rightarrow$ 

, $S \rightarrow$ 

,

$\bar{S} \rightarrow$ 

and $\bar{S} \rightarrow$ 

Then $L(G) = \{$ 

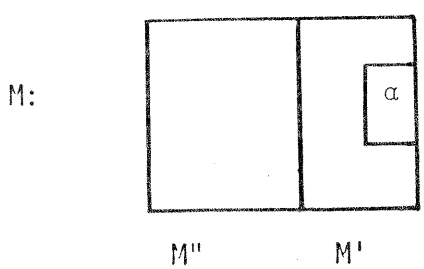
$: n \geq 1\}$. □

2. COMBINATORIAL PROPERTIES OF NLC LANGUAGES

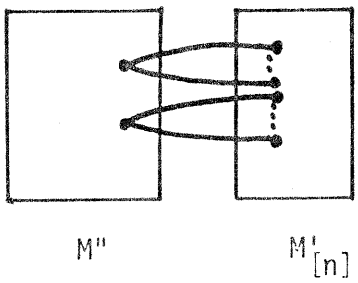
Some sets of graphs (graph languages) can be defined by NLC grammars and some cannot. An NLC grammar is a finite definition of a (possibly infinite) set of graphs. One can expect that the definability of a graph language by an NLC grammar should imply some specific properties of the graph language (properties that make it possible to define this language by an NLC grammar). Hence it is natural to look for combinatorial properties of NLC languages (that would distinguish them from other kinds of graph languages). The ultimate goal is to provide a combinatorial characterization of NLC graph languages, but such an aim seems to be too ambitious at the moment (even in the case of string languages we do not have yet a combinatorial characterization of context-free languages!). As a first step towards achieving this goal one looks into necessary conditions for a graph language to be an NLC language. A typical result in this direction would look as follows: "Let K be an NLC language. If $M \in K$ and the property P of M holds, then $L(M) \subseteq K$ where L(M) is a (preferably infinite) language of graphs that are "related" to M . If L(M) results from M by (iterating) local changes then such a result is traditionally referred to as a "pumping theorem" (see, e.g., classical pumping results for context-free string languages).

Pumping theorems are available also for NLC languages. The difficulty in obtaining such a result is (at least) two-fold: (1) NLC grammars are not "context-free" in nature, meaning that the order of applications of (the same set of) productions (to "the same" nodes) can considerably influence the final result, (2) it is not at all clear how to express mathematically the iteration of a subgraph in a graph (this is trivial in the case of strings which have very rigid structure). These difficulties were resolved in [JR1] where a pumping theorem for general NLC grammars is stated. Because of the restriction on the size of this paper we will not attempt to state formally the result (this requires the introduction of specific mathematical notions and notation to resolve point (2) stated above). Instead we will discuss the result in a rather informal way; the interested reader can find formal details in [JR1].

Roughly speaking, the "pumping theorem" for NLC languages states the following. If G is an NLC grammar then in each "big enough" graph M one can find three full subgraphs M', M" and α such that the pair (M',M") forms a partition of M, α is a subgraph of M', $2 \leq \#V_{M'} \leq C_G$ (for some positive integer constant C_G dependent on G only) and $\#V_{M'\setminus\alpha} \geq 1$:



where M' can be "pumped". This pumping goes as follows. One can find two relations $B_1, B_2 \subseteq V_{M'} \times V_{M'}$ such that for each $n \geq 2$ the graph $M^{(n)} \in L(G)$ where $M^{(n)}$ has the structure:

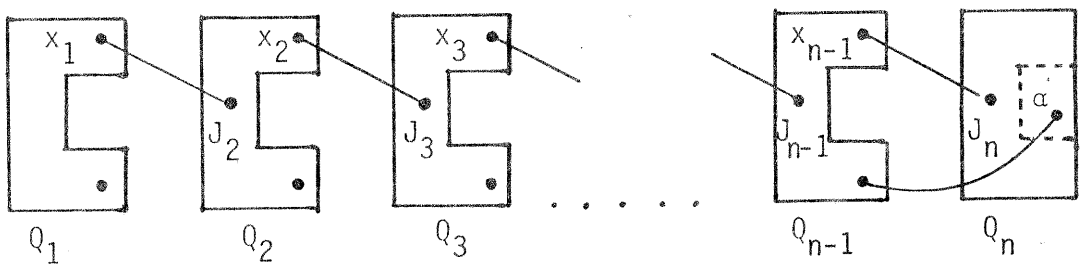


and

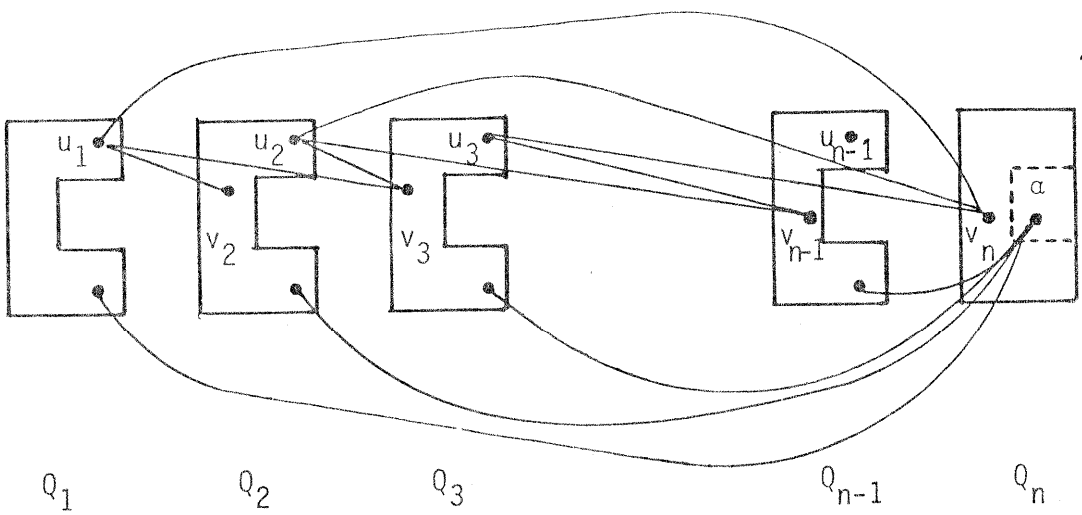
(1) there is a "simple algorithm" for establishing the edges between M" and $M'_{[n]}$, and

(2) $M'_{[n]}$ is constructed by forming (n-1) copies $Q_1,\ldots,Q_{n-1}$ of $M'\backslash\alpha$ and one copy Q_n of M' and connecting

(2.1) for each $i \in \{1,\ldots,n-1\}$ and each pair $(x,y) \in B_1$ the copy of x in Q_i with the copy of y in Q_{i+1}:



(2.2) for each $i \in \{1,\ldots,n-1\}$, each $j \in \{i+1,\ldots,n\}$ and each pair $(u,v) \in B_2$ the copy of u in Q_i with the copy of v in Q_j:

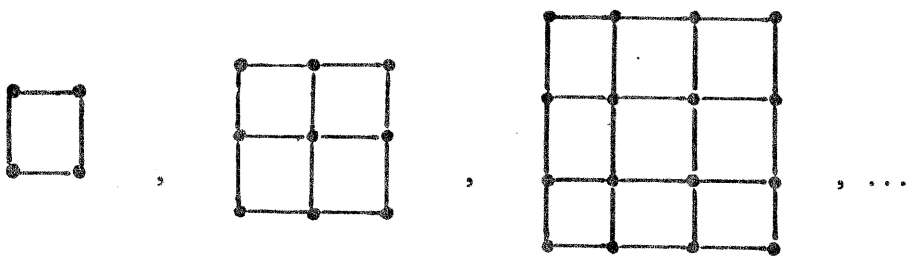


One of the main uses of results like the pumping theorem discussed above is to prove that specific graph languages are not NLC languages -a task which in general is rather difficult. We present now a corollary of the pumping theorem which then is used to show that a specific graph language is not an NLC language. We say that a graph language K is of bounded degree if there exists a positive integer d such that each node in each graph of K is of degree not exceeding d. Accordingly, an NLC grammar G is of bounded degree if L(G) is of bounded degree.

Theorem 2.1. Let G be an NLC grammar such that L(G) is infinite, L(G) is of bounded degree and each graph in L(G) is connected. Then there exists a positive integer constant C such that for each positive integer m there exists a graph M in L(G) such that $\#V_M > m$ and $\underline{diam}(M) \geq \frac{\#V_M}{C}$. □

For a graph language K, $\underline{und}(K) = \{\underline{str}(M) : M \in K\}$.

Corollary 2.1. Let K be a graph language such that und(K) consists of all the graphs (grids) of the form:



Then $K \notin L(NLC)$. □

3. DECISION PROBLEMS

NLC languages are defined using NLC grammars - in this way the "access" to an NLC language we have is through an NLC grammar defining it. In particular, in order to answer various questions concerning NLC languages we have to analyze NLC grammars defining them. In order to determine how "good" are NLC grammars as definitions of NLC languages one investigates various decision problems and in the first instance one tries to establish the boundary between algorithmic and non-algorithmic properties of NLC grammars. In this section we provide a number of results of this kind.

First of all we investigate a number of traditional (classical) decision problems concerning grammars as generative devices (of string or graph languages).

Theorem 3.1. The following problems are decidable for an arbitrary NLC grammar G:

(1) Is L(G) empty?

(2) Is L(G) infinite?

(3) Is M in L(G)?, where M is an arbitrary graph from G_Δ and Δ is the terminal alphabet of G. □

Theorem 3.2. The following problems are undecidable for arbitrary NLC grammars G_1 and G_2:

(1) $L(G_1) = L(G_2)$?

(2) $L(G_1) \cap L(G_2) = \emptyset$? □

The bulk of research on decision problems for NLC grammars concerns properties intrinsic to graph grammars and languages (as opposed to properties that can be stated for graph grammars as well as for string grammars; the two theorems above provide examples of such "common" properties).

A major technical result (underlying the proofs of many other results concerning undecidable properties of NLC grammars) is the following one. Its proof consists of a rather elaborate way of coding (an instance of) the Post Correspondence Problem into the language of an NLC grammar.

Theorem 3.3. It is undecidable whether or not the language of an arbitrary NLC grammar contains a discrete graph. □

The above result is essentially used in the proofs of, e.g., the following results

Theorem 3.4. It is undecidable whether or not the language of a given NLC grammar

(1) contains a planar graph,

(2) contains a connected graph,

(3) contains a hamiltonian graph. □

Given a graph $M = (V,E,\Sigma,\varphi)$ and two nonempty subsets Σ_1,Σ_2 of Σ, we say that Σ_1,Σ_2 are adjacent in M if E contains an edge $\{u,v\}$ such that $\varphi(u) \in \Sigma_1$ and $\varphi(v) \in \Sigma_2$; otherwise Σ_1,Σ_2 are not adjacent in M. We say that Σ_1,Σ_2 are connected in M if V contains nodes $v_1,v_2,\ldots,v_n$ for some $n \geq 2$ such that $\{v_i,v_{i+1}\} \in E$, $\varphi(v_1) \in \Sigma_1$ and $\varphi(v_n) \in \Sigma_2$. Given an NLC grammar G and two nonempty subsets A,B of its terminal alphabet, we say that A,B are adjacent (connected) in G if L(G) contains a graph M such that A,B are adjacent (connected) in G.

Theorem 3.5. It is undecidable whether or not L(G) contains a graph M such that A is not adjacent to B in M, where G is an arbitrary NLC grammar and A,B are arbitrary subsets of the terminal alphabet of G. □

The following two results put the above theorem in a better perspective.

Theorem 3.6. It is decidable whether or not A,B are adjacent in G, where G is an arbitrary NLC grammar and A,B are arbitrary subsets of the terminal alphabet of G. □

Theorem 3.7. It is undecidable whether or not A,B are connected in G, where G is an arbitrary NLC grammar and A,B are arbitrary subsets of the terminal alphabet of G. □

One of the important (also for practical reasons) properties of a graph grammar is the bounded degree. Surprisingly enough, it turns out that the following result holds.

Theorem 3.8. It is decidable whether or not an arbitrary NLC grammar G is of bounded degree. □

The above theorem should be contrasted with the following two results.

Theorem 3.9. The following problems are undecidable for an arbitrary NLC grammar G:

(1) $L_{con}(G)$ is of bounded degree?

(2) S(G) contains a graph M such that the family $\{\overline{M} : M \overset{*}{\underset{G}{\Rightarrow}} \overline{M}\}$ is of bounded degree. □

Theorem 3.10. It is decidable whether or not $L(G) - L_{con}(G)$ is of bounded degree where G is an arbitrary NLC grammar. □

4. CONTEXT-FREE NLC GRAMMARS

The connection relation is the "heart" of an NLC grammar - the embedding mechanism forms the difference between string and graph grammars (in the former it is not needed !). Hence a way to investigate properties intrinsic to graph grammars is to study their embedding mechanisms. In the case of NLC grammars this amounts to the study of connection relations. In particular one studies the classification of NLC grammars based on the properties of connection relations. An example of such a study is presented in this section.

Assume that G is an NLC grammar and M is a graph to be rewritten by G. Assume that M has two different nodes u and v both of which are labelled by a and both of which have a nonempty set of direct neighbours. Let $\pi = a \to \alpha$ be a production of G and let x be a node of α. The following may happen. If we rewrite u by π, then (a copy of) x will not be connected to any neighbour of u while when we rewrite v by π, then (a copy of) x will be connected to a neighbour of v. Such a situation may arise because the set of labels labelling the neighbours of u is not equal to the set of labels labelling the neighbours of v and hence a pair from the connection relation used to connect (a copy of) x to a neighbour of v is not applicable in establishing a connection between (a copy of) x and neighbours of u. This is an aspect of "context-sensitivity" of G that is not provided by its rewriting mechanism (productions) but by its embedding mechanism (the connection relation). In order to forbid this kind of context-sensitivity one considers context-free NLC grammars.

Definition 4.1. Let $G = (\Sigma,\Delta,P,C,Z)$ be an NLC grammar. We say that G is a context-free NLC grammar (abbreviated CFNLC grammar) if for each $a \in \Sigma$, either $(\{a\} \times \Sigma) \cap C = \emptyset$ or $(\{a\} \times \Sigma) \cap C = \{a\} \times \Sigma$. Labels a satisfying the either clause above are called disconnecting labels and labels a satisfying the or clause above are called connecting labels. □

L(CFNLC) will denote the class of languages generated by CFNLC grammars.

Informally speaking two derivations in an NLC grammar G are called similar if they differ only by the order of applications of productions (the precise definition of similarity of derivations is given in [JR2]). Clearly, in general, two similar derivations may result in totally different graphs (the number of nodes is the only "invariant" of results of similar derivations !). However, the situation is different in the case of CFNLC grammars.

Theorem 4.1. If G is a CFNLC grammar and D_1, D_2 are two similar derivations in G then the graphs resulting from D_1 and D_2 are isomorphic.

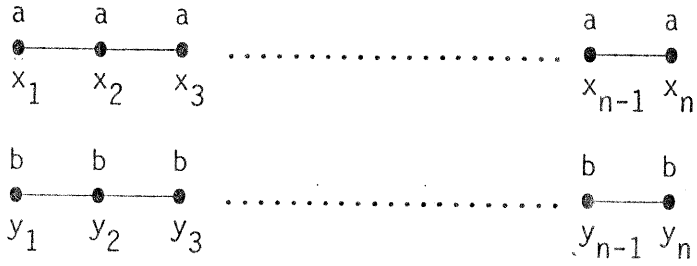
Note that the analogous result holds for the classical context-free (string) grammars which means that our context-free restriction on the connection relation of a CFNLC grammar captures an essential feature of the phenomenon of "context-freeness" in rewriting systems.

It turns out that the context-free restriction on connection relations of NLC

grammars yields a restriction on the resulting class of graph languages.

Theorem 4.2. $L(CFNLC) \subsetneq L(NLC)$. □

Actually one can prove that the language consisting of all the graphs in $G_{\{a,b\}}$ of the form



where $n \geq 1$, is in $L(NLC) - L(CFNLC)$.

The class of CFNLC grammars (and languages) has quite interesting properties especially when they are contrasted with the general class of NLC grammars (and languages). Firstly one gets a stronger version of the pumping theorem for NLC languages. Informally speaking, one can say much more about the structure of connections between pumped subgraphs (subgraphs $Q_1,\ldots,Q_n$ from the description of the pumping theorem in Section 2). The precise statement of the pumping theorem for CFNLC languages can be found in [JR6].

Secondly, many properties undecidable for the general class of NLC grammars turn out to be decidable for the class of CFNLC grammars. Examples of some of such properties are provided in the following result.

Theorem 4.3. The following problems are decidable for an arbitrary CFNLC grammar G.

(1). L(G) contains a discrete graph ?

(2). A,B are connected in G ? where A,B are arbitrary subsets of the terminal alphabet of G.

(3). $L_{con}(G)$ is of bounded degree ? □

5. GENERATING STRING LANGUAGES USING NLC GRAMMARS

As illustrated by Example 1.3, (NLC) graph grammars can generate languages of "string-like structures". A string grammar generates a string language ; moreover all "intermediate" sentential forms are strings. One can use a graph grammar to generate a string language (strings are special graphs), the difference will be that one allows "intermediate" sentential forms to be arbitrary graphs. Such an alternative of having a possibility of storing various "intermediate" information in a data structure more general than strings may turn out to be quite attractive from the practical point of view.

In this section we will consider NLC grammars as generators of string languages in the sense discussed above.

First of all, let us recall that we deal with node-labelled undirected graphs

which do not provide a unique representation of strings (that is why we say that the NLC grammar from Example 1.3 generates a language of "string-like structures" rather than a language of strings). This problem can be resolved very easily. Given an alphabet Δ we take a symbol $\not\in \Delta$ and then a string $a_1 \ldots a_n \in \Delta^+$, where $n \geq 1$, $a_1, \ldots, a_n \in \Delta$, will be represented as the graph 

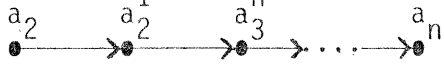
(the unique node labelled by ¢ gives the "orientation" in reading off this graph as a string). Then (to be sure that the label ¢ does not play any "unexpected" role in the generation process) we require that ¢ is a "reserved symbol" and in any NLC grammar G (used to generate a string language) $\{¢\} \times \Sigma \cup \Sigma \times \{¢\} \subseteq C$ where Σ is the alphabet of G and C is its connection relation. Let us denote (somewhat informally) by "STRINGS" the class of all graphs of the form described above.(assume that ¢ is a fixed unique symbol). Then for an NLC grammar G its string language is defined by $L_{string}(G) = L(G) \cap$ "STRINGS". Consequently we use $L_{string}(NLC)$ to denote the class of all string languages generated by NLC grammars and $L_{string}(CFNLC)$ to denote the class of all string languages generated by CFNLC graph grammars.

We have the following two basic results (we use $L(REG)$, $L(CS)$ and $L(REC)$ to denote the classes of regular, context-sensitive and recursive string languages).

Theorem 5.1. $L(CS) \subseteq L_{string}(NLC) \subseteq L(REC)$. □

Theorem 5.2 $L(REG) = L_{string}(CFNLC)$. □

The class of context-free string languages ($L(CF)$) is a very basic class of string languages and so a characterization of this class by NLC grammars (used as generators of string languages) seems to be a very natural research topic. We were not able to provide such a characterization by pointing out a subclass of NLC grammars generating exactly $L(CF)$ (in the way that CFNLC grammars generate $L(REG)$). However, it turns out that $L(CF)$ can be characterized by (modified) NLC grammars when one turns to the generation of directed (rather than undirected) graphs.

Clearly, each string $a_1 \ldots a_n$ where $n \geq 1$ and $a_1, \ldots, a_n$ are letters has a very natural representation as a directed graph: 

We use "DSTRINGS" to denote the class of all graphs of this form. We have to adjust now the notion of an NLC grammar in such a way that directed graphs are generated.

A directed NLC grammar (abbreviated DNLC grammar) is a construct $G = (\Sigma, \Delta, P, C_{IN}, C_{OUT}, Z)$ such that both $(\Sigma, \Delta, P, C_{IN}, Z)$ and $(\Sigma, \Delta, P, C_{OUT}, Z)$ are NLC grammars. Intuitively speaking a DNLC grammar works in the same way as an NLC grammar except that now after a rewriting of a node is done, the embedding is performed in two steps (their relative order is not important): (1) all edges incoming from the neighbours of the mother node to the nodes in the daughter graph are established using C_{IN}, and (2) all edges outgoing from the nodes of the daughter graph to the direct neighbours of the mother node are established using C_{OUT}. Analogously to the use of NLC grammars we define its languages L(G) and its string language $L_{string}(G) = L(G) \cap$ "DSTRINGS"; the class of all string languages generated by directed CFNLC grammars is denoted by $L_{string}(CFDNLC)$. Then we get the following characterization of $L(CF)$.

Theorem 5.3. $L(CF) = L_{string}(CFDNLC)$. □

6. A GENERAL FRAMEWORK FOR GRAPH GRAMMARS

In the preceding sections the reader got acquainted with the several issues in the theory of NLC grammars. This theory is proposed as an initial step in the systematic build-up of the mathematical theory of graph grammars. The choice of the embedding mechanisms (the connection relation) used in NLC grammars is one of many possible choices and was dictated by its "naturalness" (elegance ?).

The aim of the present section is to put the theory of NLC grammars in the broader perspective of a general theory of graph grammars. We will consider a framework for the general theory of graph grammars in which embedding is restricted to direct neighbours of the mother graph. Our approach is related to that of [RM]. The following is the basic notion of such a general framework.

Definition 6.1. A graph grammar with neighbourhood controlled embedding (abbreviated NCE grammar) is a system $G = (\Sigma,\Delta,P,Z)$ where Σ is a finite nonempty set (called the total alphabet), Δ is a nonempty subset of Σ (called the terminal alphabet), P is a finite set of productions of the form (α,β,ψ) where α is a connected graph, β is a graph and ψ is a function from $V_\alpha \times V_\beta \times \Sigma$ into $\{0,1\}$; ψ is called the embedding function of the production and $Z \in G_\Sigma$ (called the axiom). □

Intuitively speaking, a direct derivation step in an NCE grammar is performed as follows. Let M be a graph. Let $\pi = (\alpha,\beta,\psi)$ be a production of P, let $\hat{\alpha}$ be a full subgraph of M such that $\hat{\alpha}$ is isomorphic to α (with h being an isomorphism from α into $\hat{\alpha}$) and let $\hat{\beta}$ be isomorphic to β (with g being an isomorphism from $\hat{\beta}$ into β) where $V_{\hat{\beta}} \cap V_{M\setminus\hat{\alpha}} = \emptyset$. Then the result of the application of π to $\hat{\alpha}$ (using h,g) is obtained by first removing $\hat{\alpha}$ from H, then replacing $\hat{\alpha}$ by $\hat{\beta}$ and finaly adding edges $\{n,v\}$ between every $n \in V_{\hat{\beta}}$ and every $v \in V - V_{\hat{\alpha}}$ such that

(1) there exists a node $m \in V_\alpha$ with $\{h(m),v\} \in E_M$, and

(2) $\psi(m,g(n),\varphi_M(v)) = 1$.

Note that the embedding function ψ explicitly specifies which nodes of $\hat{\beta}$ can be connected to nodes of $M\setminus\hat{\alpha}$ that are neighbours of nodes in $\hat{\alpha}$. Also ψ explicitly specifies nodes in $\hat{\alpha}$ the neighbours of which can be connected to nodes in $\hat{\beta}$. However, ψ cannot explicitly specify which neighbours of $\hat{\alpha}$ can be connected to nodes in $\hat{\beta}$ for the simple reason that, in general, the number of such neighbours cannot be a priori limited, while the specification of a NCE grammar must remain finite. Hence ψ is a function from $V_\alpha \times V_\beta \times \Sigma$; the only way we can specify which neighbours of $\hat{\alpha}$ can be connected to nodes of $\hat{\beta}$ is by specifying them by their labels.

The above given description of a (concrete) direct derivation step can be formalized giving rise to the direct derivation relation $\overset{}{\Rightarrow}_G$. Then the derivation relation $\overset{*}{\Rightarrow}_G$ is defined as the transitive and reflexive closure of $\overset{}{\Rightarrow}_G$.

Definition 6.2. Let $G = (\Sigma,\Delta,P,Z)$ be an NCE grammar. The language of G (denoted $L(G)$) is the set $\{M \in G_\Delta : Z \overset{*}{\Rightarrow}_G M\}$. □

Since NLC grammars are node rewriting grammars we will be interested in those NCE

grammars that rewrite single nodes.

Definition 6.3. A 1-NCE grammar is an NCE grammar (Σ,Δ,P,Z) such that each production in P is of the form (α,β,ψ) with $\#V_\alpha = 1$ and $E_\alpha = \emptyset$. □

If (α,β,ψ) is a production in a 1-NCE grammar then, clearly, ψ corresponds in a natural way to a function from $V_\beta \times \Sigma$ into $\{0,1\}$. Hence we assume that the productions of a 1-NCE grammar are given in the form (α,β,ψ) where ψ is a function from $V_\beta \times \Sigma$ into $\{0,1\}$. Thus ψ is a function of two arguments. Depending on the fact whether or not, for a given argument, ψ depends on this argument (or, in the case of the first argument V_β, whether or not ψ depends only on the label of the argument) we get the following "natural" subclasses of the class of 1-NLC grammars.

Definition 6.4. Let $G = (\Sigma,\Delta,P,Z)$ be a 1-NCE grammar. Then G is a (X,Y)- grammar for each $X \in \{0,1,2\}$ and $Y \in \{0,1\}$ that satisfy the following conditions:

(1) If there exists a production $(\alpha,\beta,\psi) \in P$, nodes $x,y \in V_\beta$ and a label $\ell \in \Sigma$ such that $\psi(x,\ell) \neq \psi(y,\ell)$, then $X \geq 1$.

(2) If there exists a production $(\alpha,\beta,\psi) \in P$, nodes $x,y \in V_\beta$ and a label $\ell \in \Sigma$ such that $\varphi_\beta(x) = \varphi_\beta(y)$ and $\psi(x,\ell) \neq \psi(y,\ell)$, then $X = 2$

(3) If there exists a production $(\alpha,\beta,\psi) \in P$, a node $x \in V_\beta$ and labels $\ell_1,\ell_2 \in \Sigma$ such that $\psi(x,\ell_1) \neq \psi(x,\ell_2)$ then $Y = 1$. □

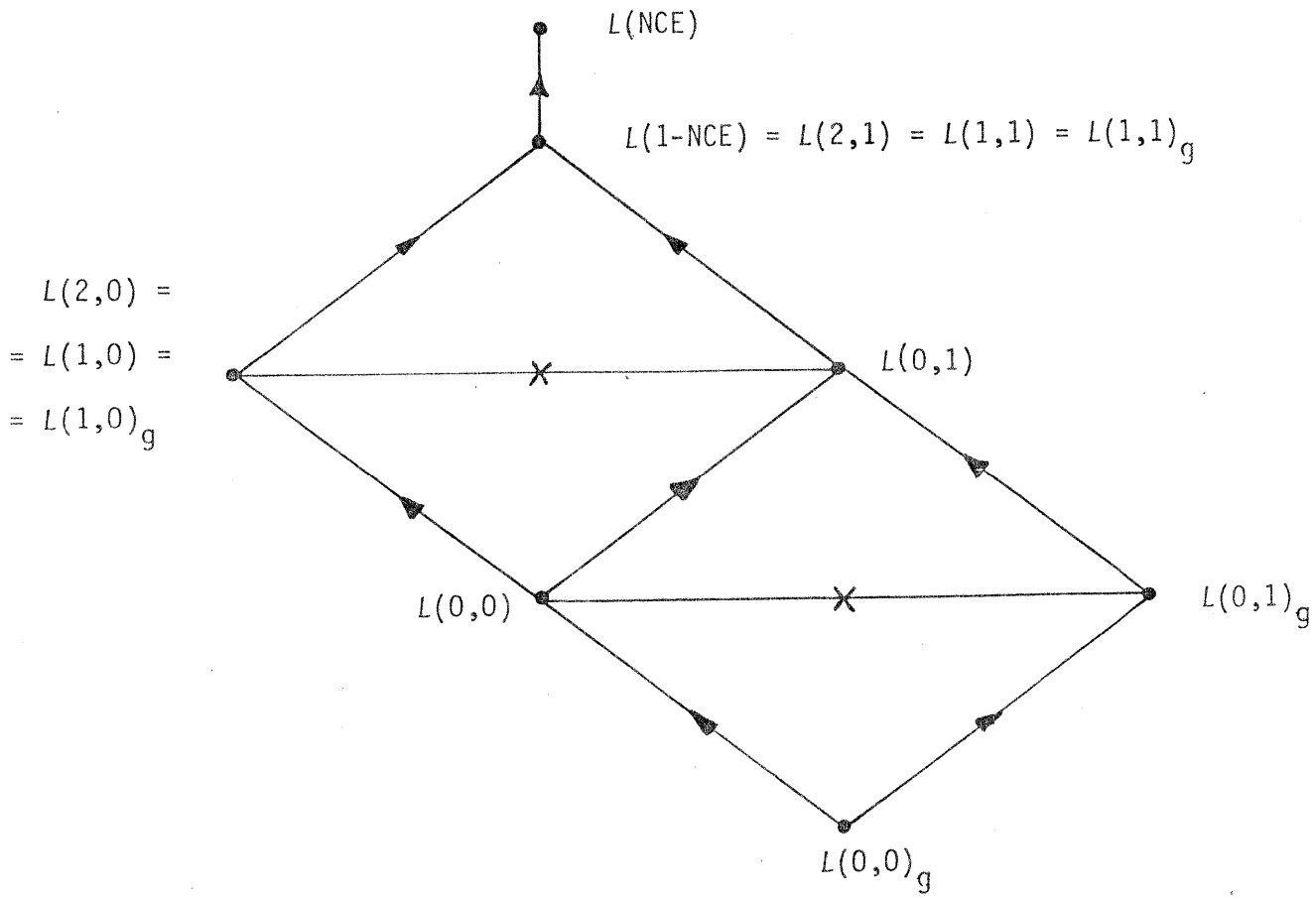
Thus, intuitively speaking, $X = 0$ implies that ψ is not dependent on the nodes of the daughter graph β and $X = 1$ implies that, although ψ can distinguish between different nodes of the daughter graph, ψ cannot distinguish two nodes of the daughter graph labelled in the same way. $Y = 0$ implies that ψ cannot distinguish between any two neighbours of the rewritten node (even if they have different labels). Observe that a "(2,1) grammar" is in this way a synonym for a "1-NCE grammar".

If all productions of a (X,Y)-grammar G use "the same" embedding function (this can easily be formalized) then we say that G is a global (X,Y)-grammar (denoted $(X,Y)_g$-grammar).

If T denotes a type of an NCE grammar (e.g. T = 1-NCE or T = (1,1) or $T = (1,0)_g$) then $L(T)$ denotes the class of all languages generated by type T grammars (to simplify the notations we will omit "redundant" parenthesis)

In order to understand better the role of various components of 1-NCE grammars in the graph generating process one investigates the relationship between classes of languages generated by various subclasses of the class of 1-NCE grammars. Here we have the following results.

Theorem 6.1. The following diagram holds:



where for classe A,B of graph languages $A \to B$ stands for "$A \subsetneq B$" and $A \not\to\not\leftarrow B$ stands for "$A - B \neq \emptyset$ and $B - A \neq \emptyset$". □

Now in order to understand the place of the class of NLC grammars in the general theory of graph grammars one should relate them to the classification of various subclasses as presented in the diagram of the statement of Theorem 6.1. It turns out that indeed L((CFNLC) and L(NLC) are already present in this diagram and moreover that (NLC) is the class of all 1-NCE languages. Hence, although arrived at in a different way, the class of NLC grammars (and languages) would have to be "discussed" in the systematic investigation of the theory of gramph grammars !!

Theorem 6.2. $L(2,1) = L(\text{NLC})$ and $L(2,0) = L(\text{CFNLC})$. □

7. DISCUSSION

In our paper we have provided a (rather sketchy) survey of a number of research areas within the theory of NLC grammars. We hope that it gives the reader an idea of a number of developments within this theory. Because of the restrictions on the size of this paper we could not cover quite many other issues that are being actively investigated within the theory of NLC grammars. We will outline some of them in this final section.

As we have indicated already, it is expected that the investigation of various issues within one (systematically built up) theory of graph grammars will lead to the development of mathematical techniques to deal with many problems of the theory. Although a number of such techniques were developed already, we feel that a lot remains to be done. A lack of mathematical techniques is often reflected in open technical problems. Here are some of such problems concerning NLC grammars - it is expected that solving them will lead to new techniques.

(1). We say that an NLC grammar is in a k-ary form, where $k \geq 1$, if all graphs at the right hand side of productions in G have no more than k nodes. Does there exist a positive integer k such that each NLC language can be generated by an NLC grammar in k-ary form? (Note that the corresponding question for "classical" classes of grammars, such as context-free or context-sensitive, gets a positive easy to prove answer!).

(2). We say that an NLC grammar is symmetric if its connection relation is a symmetric relation. Can every NLC language be generated by a symmetric NLC grammar?

(3). We say that an NLC grammar is functional if its connection relation is a function. Can every NLC language be generated by a functional NLC grammar?

(4). We have demonstrated that a number of existential questions concerning NLC grammars, such as "Does the language of an NLC grammar contain a planar (connected, discrete, hamiltonian,...) graph?", are undecidable. What about the decidability status of the universal questions, such as "Are all graphs in the language of an NLC grammar planar (connected, discrete, hamiltonian,...)?".

We consider the NLC model to be the basic, initial model in the systematic build up of a general theory of graph grammars. There are several ways of extending this model in order to get a more general theory.

(i) First of all one can extend the rewriting mechanism. This can be achieved in several ways.

(i.1) The rewriting of a node can be made "context-sensitive" by providing application conditions for a production; e.g., a node u labelled by b can be rewritten by a production $b \to \beta$ only if u has a direct neighbour labelled by c. This type of a context-sensitive extension of the notion of an NLC grammar is considered in [JR2].

(i.2) Rewriting units may be more general than nodes only. For example, one can rewrite the so called "generalized handles" (see [GJTR]). The so extended model, called GH grammars, turns out to be very useful in the study of basic issues

concerning concurrent processes. In particular it is demonstrated in [GJTR] that a very close connection can be established between the theory of GH grammars and the theory of Petri Nets.

(i.3) Modes of rewriting other than the sequential one are also considered. In [GJTR] where the theory of GH grammars is investigated a concurrent mode of rewriting is considered. Such a mode of rewriting is also considered in [EJKR] where the theory of NLC grammars is "tested" against various issues concerning concurrency and canonical derivations as developed in the "Berlin school of graph grammars" (see, e.g., [EK] and [K]). The parallel mode of rewriting, as used in the theory of L-systems (see, e.g., [RS]), is investigated in [JRV1] and [JRV2]. Here the comparison with the Culik-Lindenmayer model of graph grammars based on parallel rewriting (see, e.g., [CL]) is done and various new variants of both models are considered.

(ii) One can extend the rewriting mechanism. At least two extensions seem to be natural.

(ii.1) A "context-sensitive" extension: e.g. an edge between a node u labelled by b in the daughter graph and a node v labelled by c in the direct neighbourhood of the mother node is established only if (b,c) is in the connection relation and u has a direct neighbour labelled by a.

(ii.2) A connection relation may be used to connect (with the nodes of a daughter graph) nodes that are connected to, but are not necessarily the direct neighbours of, the mother node.

The work on (ii.1) and (ii.2) is at progress.

There are many problem areas that must be investigated before we can get a really mature theory of graph grammars based on the NLC model. Among those problem areas are:

(1) Parsing and complexity considerations.

(2) Relationship to other models such as Nagl model (see, e.g., [N]) and Schneider-Ehrig-Kreowski model (see, e.g., [E]).

(3) Distinguishing and studying the "central" subclasses of the class of NLC grammars (the class of CFNLC grammars is an example of such a central subclass).

(4) Considering classes of languages obtained from NLC grammars by "squeezing mechanisms" other than the intersection with the terminal alphabet; taking only "connected graphs" or graphs of degree not exceeding k, where k is a fixed positive integer, are two examples of such squeezing mechanisms.

(5) Extending the basic notion of a graph and considering the generation of directed, edge labelled, ... graphs.

8. BIBLIOGRAPHICAL COMMENTS

NLC grammars were introduced in [JR1] and [JR2]. Section 2 is based on [JR1]. All results from Section 3 are from [3] except for Theorem 3.8 which is from [JR1]. CFNLC grammars were introduced in [JR2] where Theorem 4.1 and 4.2 were proved. Theorem 4.3 and the stronger version of the pumping theorem are from [JR6]. Section 5 is based on [JR2] and [JR4]; Theorems 5.1 and 5.2 are from [JR2] and Theorem 5.3 is from [JR4]. Section 6 is based on [JR5].

ACKNOWLEDGEMENTS

The second author gratefully acknowledges the financial support of NSF grant MCS 79-038038.

REFERENCES

[CER] Claus, V., Ehrig, H. and Rozenberg, G. (Eds.), Graph grammars and their application to computer science and biology, Lecture Notes in Computer Science, v. 73, 1979.

[CL] Culik II, K. and Lindenmayer, A., Parallel graph generating and graph recurrence systems for multicellular development, International Journal of General Systems, v. 3, 53-66, 1976.

[E] Ehrig, H., Introduction to the algebraic theory of graph grammars (a survey), in CER .

[EJKR] Ehrig, H., Janssens, D., Kreowski, H.-J. and Rozenberg, G., Concurrency of node-label controlled graph transformations, University of Antwerp, U.I.A., Technical Report 82-38, 1982.

[EK] Ehrig, H. and Kreowski, H.-J., Parallelism of manipulations in multidimensional information structures, Lecture Notes in Computer Science, v. 45, 284-293, 1976.

[GJTR] Genrich, H., Janssens, D., Thiagarajan, P.S. and Rozenberg, G., Generalized handle grammars and their relation to Petri Nets, Institut für Informationssystemforschung, GMD Bonn, Technical Report 82-06, 1982.

[JR1] Janssens, D. and Rozenberg, G., On the structure of node-label controlled graph languages, Information Sciences, v. 20, 191-216, 1980.

[JR2] Janssens, D. and Rozenberg, G., Restrictions, extensions and variations of NLC grammars, Information Sciences, v. 20, 217-244, 1980.

[JR3] Janssens, D. and Rozenberg, G., Decision problems for node-label controlled graph grammars, Journal of Computer and System Sciences, v. 22, 144-177, 1981.

[JR4] Janssens, D. and Rozenberg, G., A characterization of context-free string languages by directed node-label controlled graph grammars, Acta Informatica, v. 16, 63-85, 1981.

[JR5] Janssens, D. and Rozenberg, G., Graph grammars with neighbourhood controlled embedding, Theoretical Computer Science, v. 21, 55-74, 1982.

[JR6] Janssens, D. and Rozenberg, G., Context-free NLC grammars, University of Leiden, Institute of Applied Mathematics and Computer Science, Technical Report, 1983.

[JR7] Janssens, D. and Rozenberg, G., Bounded degree is decidable for NLC grammars, Institute of Applied Mathematics and Computer Science, University of Leiden, Technical Report, 1983.

[JRV1] Janssens, D., Rozenberg, G. and Verraedt, R., On sequential and parallel node-rewriting graph grammars, part 1, Computer Graphics and Image Processing, v. 18, 279-301, 1982.

[JRV2] Janssens, D., Rozenberg, G. and Verraedt, R., On sequential and parallel node-rewriting graph grammars, part 2, Computer Graphics and Image Processing, to appear.

[K] Kreowski, H.-J., Manipulationen van graphmanipulationen, Ph.D. Thesis, Technical University of Berlin, Computer Science Department, 1977.

[N] Nagl, M., Graph-Grammatiken, Vieweg und Sohn, Braunschweig-Wiesbaden, 1979.

[RM] Rosenfeld, A. and Milgram, D., Web automata and web grammars, Machine Intelligence, v. 7, 307-324, 1972.

[RS] Rozenberg, G. and Salomaa, A., The mathematical theory of L systems, Academic Press, London-New York, 1981.