ABSTRACT. We introduce a graph-grammar model based on edge-replacement, where both the rewriting and the embedding mechanisms are controlled by edge labels. The general power of this model is established—it turns out to have the complete power of recursive enumerability (in a sense to be made precise in the paper). In order to understand where this power originates, we identify three basic features of the embedding mechanism and examine how restrictions on these features affect the generative power. In particular, by imposing restrictions on all three features simultaneously, we obtain a graph-grammar model that was previously introduced by Kreowski and Habel.

Keywords: edge-rewriting, label-control, NLC, recursive enumerability.

CONTENTS

1. Introduction.
2. Edge-Label Controlled Graph Grammars.
3. Explicit ELC Grammars.
5. A Hierarchy of Language Classes.
6. Discussion.

1. Introduction

In recent years, node-label controlled (NLC) graph grammars have been intensively studied as a method for generating node-labeled graphs [5,6,7,8,11]. The key feature of NLC-grammars is that both the rewriting of a subgraph, and the embedding of a newly introduced subgraph are controlled by node labels. (This is in contrast to the algebraic approach [1], where rewriting and embedding are based on structural properties of graphs.) Node-labeled graphs, which are the subject of NLC-grammars, are fundamental objects with numerous applications in computer science and other areas. Within the realm of graphs, one also has edge-labeled graphs, which are equally fundamental. It is
natural to ask how the label-based control mechanism carries over to edge-labeled graphs. The aim of this paper is to initiate systematic research in this area.

The model of edge-label controlled (ELC) graph grammars we are presenting is clearly influenced by our experience with NLC-grammars. However, we should warn that making a proposal for ELC grammars involves much more than simply taking the dual of NLC grammars. This is because the node-edge duality of graphs is not perfect. In handling this we have been influenced by recent research of H.J. Kreowski and his co-workers [2,3].

This paper presents the fundamentals of our basic ELC model, where the rewritten element is an edge, and the way rewriting and embedding take place is controlled by edge-labels. The stress is on exposition and no proofs are given. All proofs of results are given in [10]. It turns out that our basic ELC model is "too powerful" in the sense that it generates all the class of recursively enumerable graph languages. In order to formally state and prove this result, we have to formalize the notion of a recursively enumerable graph language. We do this by introducing a simple algorithmic language for constructing graphs.

In analyzing ELC grammars more closely, one can distinguish three basic parameters used in the rewriting process. Roughly speaking, the first parameter determines whether neighboring edges can be deleted when an edge is rewritten. The second parameter determines whether the source and target nodes of a rewritten edge can "merge" during the rewriting process. The third parameter controls whether multiple copies of the source and target nodes of a rewritten edge can appear. We investigate the impact of these three parameters. It is interesting to notice that if we make all three parameters "restrictive", then we get the original edge-replacement systems proposed by Habel and Kreowski [2,3]. Other combinations of the parameters yield other classes of languages.

Terminology and notation: Throughout the paper, the term graph refers to a directed, edge-labeled, finite graph with at least one node (and with no self-loops). Multiple edges between the same pair of nodes are allowed. For a finite alphabet $\Delta$, the set of all graphs with labels chosen from $\Delta$ is denoted $G_\Delta$. Formally, such a graph is a tuple $(V, E, l, s, t)$, where $V$ is a non-empty finite set of nodes, $E$ is a finite set of edges, $l : E \rightarrow \Delta$ is a function assigning labels to edges, and $s, t : E \rightarrow V$ are functions assigning source and target nodes to edges. An edge labeled by $A$ is called an $A$-edge. We say that an edge is incident to its source and target nodes, and two edges which are incident to a common node are called adjacent. Similarly, two nodes which are incident to a common edge are called adjacent.

2. Edge-Label Controlled Graph Grammars

Definition. An edge-label controlled graph grammar (ELC grammar) is a 5-tuple $(\Sigma, \Delta, P, S, C)$, where

- $\Sigma$ is a finite set of edge labels.
- $\Delta$ is a proper subset of $\Sigma$, called the terminal labels; elements of $\Sigma - \Delta$ are nonterminals.
- $P$ is a finite set of productions; each production has the form
  \[ A := (\alpha, \alpha_{\text{source}}, \alpha_{\text{target}}) \]
  where $A$ is a nonterminal label, $\alpha \in G_\Sigma$, and $\alpha_{\text{source}}$ and $\alpha_{\text{target}}$ are non-empty subsets of the nodes of $\alpha$.
- $S$ is a nonterminal label called the start label.
- $C \subseteq (\Sigma \cup \{\text{ISOLATED}\}) \times \Sigma$ is the connection relation.