Viability of a Parsing Algorithm for Context-sensitive Graph Grammars

Jeroen T. Vermeulen

August 26, 1996

Abstract

Graph Grammars describe formal languages similar to the textual ones commonly used to define computer languages, but their productions operate on graphs instead of on text. They are *context-sensitive* when productions rewrite patterns of symbols rather than single symbols. Parsing such grammars is inherently very hard because among other reasons, the input is not sequential in nature.

An algorithm for parsing a large class of context-sensitive graph grammars has been developed by Jan Rekers and Andy Schürr. This thesis describes a first implementation of this algorithm as well as several improvements, additional work, examples, theory of operation and performance characteristics. Future and existing optimizations are discussed.

1 Introduction

1.1 Graph Grammars

Although formal grammars are commonly used in computer science for the description of textual languages, relatively little progress has as yet been made towards understanding and implementing *visual* languages. A visual language defines a set of diagrams rather than a set of textual sentences. Commonly used visual languages include electrical diagrams, Petri nets, and ER diagrams; however they are only rarely defined in terms of formal grammars.

Interactive graphical editors are already available for some of these visual languages, eg. circuit-board design tools and flow-chart editors, but they are usually part of a dedicated environment and only allow syntactically correct diagrams to be generated by limiting the editing primitives to valid transformations on the underlying interpretation (or semantic value) of the diagram.

This procedure, known as *syntax-directed editing,* makes it easier on the environment but requires the user to perform a sequence of actions that is not necessarily related to his conception of the diagram he is creating. Other environments may allow more or less random editing, but provide no way of storing an unfinished diagram to disk because they don't know how to handle expressions that are not members of the formal language they implement.

It is believed that the general ability to parse visual languages after *free-form editing* with a generic graphical editor, which is accepted practice for textual languages such as C or TEX, could be used to improve user-friendliness of user interfaces and bring a host of applications that currently require expert knowledge closer to the end user.

Setting up a relational database, for instance, normally involves designing an entity-relationship architecture in ER notation, and manually converting that to a textual representation in the data-definition language included with the DBMS package. Given some kind of parser for ER diagrams this conversion