Graph Grammars Applied to Metamodels and Flowcharts

Luka Fürst1, Marjan Mernik2, Viljan Mahnič1, Barrett R. Bryant3, Jeff Gray3

1University of Ljubljana, Slovenia; 2University of Maribor, Slovenia; 3University of Alabama at Birmingham, USA

Graph grammars (as defined by Rekers & Schürr)

Introduction
- Rekers & Schürr defined graph grammars as a generalization of context-sensitive string grammars.
- Primary use: Defining the syntax of visual languages.
- A graph grammar consists of rules (productions) for replacing subgraphs in an arbitrary graph.
- Every production takes the form \( I \rightarrow \{J, R\} \) where \( I \) and \( J \) are graphs (the left-hand side is called the context).

A production and its application

Example: a grammar of hydrocarbon graphs

Properties of the grammar construction process
- The output grammar consists of both terminal and nonterminal symbols.
- A nonterminal \( v_{1}, \ldots, v_{n} \) represents a vertex \( v \) that still has to be connected with \( 1 \) (or \( 0 \)) vertices \( v_{1}(1/0) \) vertices \( v_{2}(1/0) \), and \( \ldots \) and \( \{k\} \) vertices \( v_{n}(1/0) \). 
- A terminal \( v \) represents a vertex \( v \) for which all connection requirements are satisfied.
- For each nonterminal, the grammar contains a set of productions that (either immediately or gradually) convert it to a terminal.
- Every sentential form that the grammar can generate represents a (potentially) valid model graph.
- If a sentential form contains only terminals, then it is a valid model graph.
- Otherwise, the nonterminals in a sentential form determine how the sentential form can be completed to a valid model graph.

Converting a metamodel into an equivalent graph grammar

Sample metamodel and some of its models

Initial productions
For each class \( C \), add a production \( \lambda: C \rightarrow \{ (\text{required connections}) \} \) where \( \lambda \) is a sentential form composed of a nonterminal and nonterminal symbols.

Productions for unbounded multiplicities
- For each two-way unbounded multiplicity, also add a production \( \Delta Y := X \rightarrow Y \).

Complementary nonterminal form pairs
- In some cases, two nonterminal forms may mutually satisfy their connection requirements.
- Example: \( P \) needs a connection with one \( J \), and \( J \) needs a connection with one \( P \).

Applying \( p \) to a graph

Synthesis of cyclopropene

Expanding nonterminal forms
- Nonterminal form: A sentential form composed of a nonterminal vertex and all vertices connected to it.
- Expansion: \( v_{1}, v_{2}, \ldots, v_{n} := \{(\text{connections})\} \) if \( v_{n} \) has a connection with \( v_{n-1} \), \( v_{n-2} \), and so on.
- Every nonterminal form has to be eventually expanded.
- Expansion may create new nonterminal forms, but the process is guaranteed to terminate.

Selected productions that create new vertices

Selected productions that reuse existing vertices

Dealing with inheritance
- A simple solution: The terminal \( P \) is re-interpreted as a wildcard that can be replaced either with \( JP \) or with \( SP \).

Translating flowcharts into Java programs using graph grammar parsing

Introduction
- Problem statement: Convert a given flowchart into an equivalent Java program.
- Motivation: This is a simple example, but a similar approach could be used for parsing and translating real-world languages.

Implementation:
- Define a flowchart grammar and a translation scheme.
- Parse a given flowchart against this grammar using the Rekers-Schürr parser.
- Build a syntax graph from the derivation produced by the parser.
- Recursively execute the translation scheme on the syntax graph.

Sample input and output

Output:
```java
import java.util.Scanner;
public class Program {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a;
        int b;
        b = sc.nextInt();
        a = a - b;
        System.out.print(a);
    }
}
```

The simplified syntax graph for the sample input

Graph grammar with a translation scheme