TWO-LEVEL GRAPH GRAMMARS

Wolfgang Hesse

Institut für Informatik der Technischen Universität München, Postfach 20 24 20, D-8 München 2

Abstract: Two-level graph grammars (2GG) combine the concepts of (one-level) graph grammars - as defined by PRATT - and of two-level (string) grammars - as introduced by v. WIJNGAARDEN for the formal definition of ALGOL 68. 2GG's settle both the inadequacies of one-level graph grammars and of two-level string grammars, the former resulting from the lack of parameters, the latter from the general lack of structure of string manipulation systems. As a field of application of 2GG's, the formal description of programming languages is focussed.

1 Introduction

Graph grammars have been introduced and successfully been used at various places in computer science and biology, wherever sets of more-dimensional objects are involved. Even for the formal description of (one-dimensional) programming languages graph grammars have proved useful, whether for the definition of the "abstract syntax" using a graph structure, or for the specification of control and data structures or for the formulation of semantical models. The application of graph models for the formal description of programming languages has been investigated by several authors (CULIK, PRATT, the VIENNA group etc.). PRATT used some sort of context-free graph grammars for the syntax and hierarchical graphs (H-graphs) for the formal definition of the semantics (/PRA 69/, /PRA 71/). PRATT's method provides for an adequate description of many control and data structures, his language descriptions, however, suffer from two drawbacks: The first is the context-free character of his grammars, which does not admit the formalisation of so-called "context conditions". The other is the lack of parameters in the H-graph model, which e.g. prohibits from directly modelling recursive functions with parameters by H-graphs.

W-grammars ("van WIJNGAARDEN grammars", "two-level string grammars") do away with both these difficulties. Parametrisation was the basic idea for the introduction of metanotions in W-grammars. The adequacy of W-grammars for the description of arbitrary context conditions has been confirmed theoretically and practically (/SIN 67/, /A6BR/, /A6BRR/). Even for the description of the semantics, W-grammars may be used very well, as is shown in /C-U 75/, /HES 76/ and /HES 77b/. Objections to W-grammars usually aim at the weak point, that they exclusively deal with strings, which forces one to represent all objects by strings, without regard to their natural structure.
Two-level graph grammars (2GG's) are the consequent answer to the objections to both one-level graph grammars and two-level string grammars. They provide metanotions acting as parameters and allowing the formalisation of the context conditions and of the semantics, as W-grammars do. On the other hand, they offer the facilities of graph grammars on both their two levels.

The definitions of one-level graph grammars in section 2 and of W-grammars in section 3 essentially follow PRATT and v. WIJNGAAARDEN et al. (/PRA 71/, /A68R/). Two-level graph grammars are defined in section 4 and some of their properties are considered in section 5. The W-grammar concept of "predicates" is extended to 2GG's in section 6. The use of 2GG's for complete formal language descriptions is outlined in section 7. Section 8 contains some remarks about the implementation of 2GG's.

2 One-level graph grammars

2.1 Basic sets

Let be given the following nonempty, finite, pairwise disjoint sets:
- N: a set of nodes,
- S: a set of nonterminal atoms,
- T: a set of terminal atoms,
- L: a set of edge labels containing the label "o".

V = S ⊔ T is called the set of atoms.

2.2 F-graphs

By means of an inductive definition, sets $\hat{F}_n(S,T,L)$ of F-graphs of degree n (n ≥ 1) over S, T and L are introduced.

Let $\hat{F}_0(S,T,L) = S \cup T$. An F-graph of degree n is a 4-tuple $F = (Q_F, C_F, E_F, N_F)$, where
- $Q_F \subseteq N$,
- $C_F : Q_F \rightarrow \bigcup_{j=1}^{n-1} \hat{F}_j(S,T,L)$ (the node value function of F),
- $E_F \subseteq Q_F \times L \times Q_F$ (the set of edges of F),
- $N_F \subseteq Q_F$ is a distinct node, the frame node of F, with the following properties:
  - there is exactly one edge $e_i \in E_F$ with $e_i = (N_F, o, K)$, $K \in Q_F$, called the entry edge of F (K is the entry node of F),
  - there is at most one edge $e_o \in E_F$ with $e_o = (K, o, N_F)$, $K \in Q_F$, called the exit edge of F (K is the exit node of F), if any.

Let $\hat{F}(S,T,L)$ denote the union $\bigcup_{j=1}^{\infty} \hat{F}_j(S,T,L)$ and $\hat{F}^+(S,T,L) = \hat{F}(S,T,L) \cup \hat{F}_0(S,T,L)$.

In the following, graph indices ("F" in $Q_F$, $C_F$ etc.) are omitted, if no confusion may arise.