WILLIAMS81

Williams, M.H.; Methods for Specifying Static Semantics; Computer Languages, Volume 6, pp. 1-17.

The specification of computer languages has three parts - syntax, static semantic, and semantic rules. Syntax rules simply state the structure of the language while semantic rules specify the effect or meaning of each of the syntactic constructs. Static semantic rules fall between these two. They involve relating different occurrences of the same identifier, and restricting these occurrences. Thus, they go beyond syntax which is concerned simply with the arrangement of tokens, but they are still used to determine the validity of programs in the language.

This paper describes the essential characteristics of a notation for specifying static semantics, such as allowing reference to the strings associated with nonterminals, and providing mechanisms for counting and creating lists. It then examines and characterizes several notations that have been proposed.

The author defines two categories between which all notations for static semantics can be divided. The first category he calls "notations with generalized nonterminals". A very simple mechanism which allows information to be associated with, and passed between nonterminals is the common characteristic among these schemes. In order to make static semantic checks, new nonterminals are usually added to the grammar. This can make the grammars harder to read because it disguises the syntax. Also, it is typically not clear whether information associated with a nonterminal is meant to be passed up or down in the parse tree. The earlier notations fall into this category, for example canonic systems and two-level grammars, which are described in some detail.

Notations in the second category avoid the two basic problems of the earlier methods. In this category, actions which perform static semantic checks are appended to the productions of the grammar and the basic structure of the grammar need not be altered. Confusion over the direction of propagation of information in the parse tree is eliminated by stating it explicitly. Three schemes in this category are described - attribute grammars, production systems, and an extended BNF.

It is concluded that an ideal notation would be one from the second category in which actions are appended to the productions, and would be based on BNF because of its general acceptance. An interface with some notation for semantic specifications should also be provided.