WILLIAMS80

Williams, M.H.; A Formal Notation for Specifying Static Semantic Rules; Computer Languages, Volume 5, pp. 37-55.

This paper presents a formal notation for specifying static semantic rules and applies it to a simple block structured language.

The complete syntax of a programming language has two parts - ordinary syntax rules which describe the structure of the language, and static semantic rules (or syntax interpretation rules) which place restrictions on the occurrences of identifiers. Syntax rules are quite well understood and a notation exists which is generally accepted for their specification (BNF). However, no notation for static semantics has received wide acceptance.

There are two basic types of static semantic notations. Those of the first kind associate information, such as lists of identifiers, with the variables of the grammar. Those of the second category are syntax directed translation schemes (SDTS's) where actions are associated with the productions. These methods are considered less formal because the notation for specifying the actions is typically not formally defined. The author describes a SDTS notation in which this is formally defined.

The author's notation uses BNF to specify the basic syntax of a language. Each production may have actions appended to it, surrounded by brace brackets. The notation for these actions is quite powerful. The data structures available include strings and stacks, as well as ordinary integers. Conditional and loop actions are provided, and calls to run-time routines are available for dynamic checks.

Static semantic rules may be written with this notation in either a top-down or bottom-up fashion, often resulting in different placements of the checks. The author suggests that a bottom-up view is more suited to a compiler writer since it reflects the way a compiler works, while the user may find a top-down specification easier to read. The bottom-up approach results in a problem with forward references that a top-down approach doesn't have. However, this may be remedied either by keeping stacks of forward references, or by having a two pass compiler.

Finally, the author uses his notation to specify the syntax and static semantics of a simple block structured language. He concluded that the notation presented has the power to describe the complete syntax of any programming language.