ASHCROFT82
Ashcroft, E.A., Wadge, W.W.; Prescription for Semantics; ACM Transactions on Programming Languages and Systems 4, 2 (April 1982) pp. 283-294.

This paper presents a criticism of the roles which denotational and operational semantics have typically been given in the definition of programming languages, and suggests an alternative strategy for their use.

Semantics is "the study of the associations between programs and the mathematical objects which are their meaning". Denotational and operational semantics provide different methods of specifying the same meanings.

When specifying the semantics of a language operationally, one defines an abstract machine and the behavior which results when a program is run on that machine. The semantics of the program is given in terms of this behavior. This approach is termed bottom up, since it starts with the machine and works up to the language.

Specifying semantics denotationally involves defining functions which assign mathematical objects to programs and parts of programs so that the semantics of a module depends solely upon the semantics of its submodules. Hence, this technique is modular, where the operational approach is not. The key difference, however, is that denotational semantics specify what is to be computed, whereas operational semantics specify how it is computed.

The authors contend that the two techniques are complementary, but not symmetrical. Programming language features that are simple to define with one method are typically complex to define with the other. Thus, language designers must choose between language features that will be denotationally simple, and those that will be operationally simple. Most often, all features are chosen to be in one category or the other, resulting in either denotationally or operationally simple languages. Operationally straightforward languages are the most common.

Denotational semantics was originally intended as a design tool. However, it was put to use giving semantics for languages which already existed, and to a large extent has kept a descriptive role. Because of this descriptive use of the method with a large variety of existing languages, generality became a main objective. This is not considered advantageous since the power it affords gives the language designer the freedom to include undesirable features in the language. Furthermore, generality was achieved at the cost of simplicity.

A prescriptive use of denotational semantics is considered by the authors to be preferable, since it gives more freedom to both the designers and implementors of programming languages. Designers need not worry about describing how constructs will be implemented, and implementors are not constrained by any particular implementation.

Thus, the authors suggest that the roles of operational and denotational semantics be reversed. Languages should be prescribed using denotational methods, with operational concepts acting only as design aids. Operational techniques should then be developed to describe (i.e. implement) these languages.