Chapter 7.18
Deductive Semantics of RTPA

Yingxu Wang
University of Calgary, Canada

ABSTRACT

Deductive semantics is a novel software semantic theory that deduces the semantics of a program in a given programming language from a unique abstract semantic function to the concrete semantics embodied by the changes of status of a finite set of variables constituting the semantic environment of the program. There is a lack of a generic semantic function and its unified mathematical model in conventional semantics, which may be used to explain a comprehensive set of programming statements and computing behaviors. This article presents a complete paradigm of formal semantics that explains how deductive semantics is applied to specify the semantics of real-time process algebra (RTPA) and how RTPA challenges conventional formal semantic theories. Deductive semantics can be applied to define abstract and concrete semantics of programming languages, formal notation systems, and large-scale software systems, to facilitate software comprehension and recognition, to support tool development, to enable semantics-based software testing and verification, and to explore the semantic complexity of software systems. Deductive semantics may greatly simplify the description and analysis of the semantics of complicated software systems specified in formal notations and implemented in programming languages.

INTRODUCTION

Semantics in linguistics is a domain that studies the interpretation of words and sentences, and analysis of their meanings. Semantics deals with how the meaning of a sentence in a language is obtained, hence the sentence is comprehended. Studies on semantics explore mechanisms in the understanding of languages and their meanings on the basis of syntactic structures (Chomsky, 1956,

1957, 1959, 1962, 1965, 1982; Tarski, 1944).

Software semantics in computing and computational linguistics have been recognized as one of the key areas in the development of fundamental theories for computer science and software engineering (Bjoner, 2000; Gries, 1981; Hoare, 1969; McDermid, 1991; Slonneg & Kurts, 1995; Wang, 2006b, 2007c). The semantics of a programming language is the behavioral meaning that constitute what a syntactically correct instructional statement in the language is supposed to do during run time. The development of formal semantic theories of programming is one of the pinnacles of computing and software engineering (Gunter, 1992; Meyer, 1990; Louden, 1993; Bjoner, 2000; Pagan, 1981).

Definition 1. *The semantics of a program in a given programming language is the logical consequences of an execution of the program that results in the changes of values of a finite set of variables and/or the embodiment of computing behaviors in the underpinning computing environment.*

A number of formal semantics, such as the *operational* (Marcotty & Ledgard, 1986; Ollongren, 1974; Wegner, 1972; Wikstrom, 1987), *denotational* (Bjorner and Jones, 1982; Jones, 1980; Schmidt, 1988, 1994, 1996; Scott, 1982; Scott & Strachey, 1971), *axiomatic* (Dijktra, 1975, 1976; Gries, 1981; Hoare, 1969), and *algebraic* (Goguen, Thatcher, Wagner, & Wright, 1977; Gougen & Malcolm, 1996; Guttag & Horning, 1978), have been proposed in the last three decades for defining and interpreting the meanings of programs and programming languages. The classic software semantics are oriented on a certain set of software behaviors that are limited at the level of language statements rather than that of programs and software systems. There is a lack of a generic semantic function and its unified mathematical model in conventional semantics, which may be used to explain a comprehensive set of programming statements and computing behaviors. The mathematical models of the target machines and the semantic environments in conventional semantics seem to be inadequate to deal with the semantics of complex programming requirements, and to express some important instructions, complex control structures, and the real-time environments at run time. For supporting systematical and machine enabled semantic analysis and code generation in software engineering, the *deductive semantics* is developed that provides a systematic semantic analysis methodology.

Deduction is a reasoning process that discovers new knowledge or derives a specific conclusion based on generic premises such as abstract rules or principles (Wang, 2006b, 2007a, 2007c). The nature of semantics of a given programming language is its computational meanings or embodied behaviors expressed by an instruction in the language. Because the carriers of software semantics are a finite set of variables declared in a given program, program semantics can be reduced onto the changes of values of these variables over time. In order to provide a rigorous mathematical treatment of both the abstract and concrete semantics of software, a new type of formal semantics known as the deductive semantics is presented.

Definition 2. *Deductive semantics is a formal semantics that deduces the semantics of a program in a given programming language from a generic abstract semantic function to the concrete semantics, which are embodied onto the changes of status of a finite set of variables constituting the semantic environment of computing.*

This article presents a comprehensive theory of deductive semantics of software systems. The mathematical models of deductive semantics and the fundamental properties are described. The deductive models of semantics, semantic function, and semantic environment at various composing levels of programs are introduced. Properties of software semantics and relationships between