Notre Dame Journal of Formal Logic
Volume 31, Number 4, Fall 1990

Book Review

Melvin Fitting. *Computability Theory, Semantics, and Logic Programming*. Oxford University Press, 1987. 198 pages.

"We should be able to tell a computer what we want it to determine, and *it* should be able to figure out how to do it." This recurring fantasy is natural enough, given the frustrating and error-prone methods still being used to produce computer software. Through the years various new programming methodologies have been heralded as the answer. *Logic programming* has now been around for a decade and a half, and it is still going strong. The idea is certainly attractive (especially to logicians): specify what you would like the computer to do in first-order logic; the computation is then reduced to a matter of logical deduction which can be handled by an all-purpose automated deduction system. In practice, the inefficiency of existing automated deduction systems forces a compromise. The most popular compromise is PROLOG in its various dialects. In PROLOG the specification must be expressible as a finite set of *Horn clauses*, i.e. quantifier-free formulas of predicate calculus each of the form

$$A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow B$$

where $A_1, A_2, \ldots, A_n, B$ are atoms (i.e., atomic formulas). Logical deduction can then be accomplished by using *unification*, that is by finding substitutions for the variables that make the consequent atom of one clause identical with one of the conjuncts in the antecedent of another clause. Finally, in order that the system behave in a deterministic way, a specific order is specified in which the search for such substitutions is carried out. The question of to what extent this necessary compromise vitiates the promise of logic programming remains highly controversial.

The starting point of Fitting's striking book is the observation that the *elementary formal systems*, introduced by Raymond Smullyan almost three decades ago as a variant of Emil Post's *canonical systems* to provide an elegant foundation for recursive function theory (computability theory), are really logic programs. Thus, Fitting is able to provide an exposition of elementary computability theory in the context of an abstract theory of programming languages. Logicians will find that this treatment provides a particularly painless path to understanding some of the logical issues that have been of great concern to computer scientists. The *denotational semantics* of a programming language provides for each pro-

gram of the language a purely extensional description of what it is that the program computes (as opposed to the *operational semantics* which specifies the actual operations performed by the program). In Fitting's context, one predicate letter is taken to represent the "output" so the denotational semantics associates a definite relation with each program; this relation turns out to be simply the minimal "model" of the program. If an additional predicate letter is singled out as "input", then the program specifies an *operator* which maps relations to relations. These turn out to be *monotone operators* as studied by Tarski in the 1950s, and they have a least fixed point. Studying this least fixed point leads to a proof of Kleene's first recursion theorem.

As a textbook on computability theory, only the most basic results are obtained: the two recursion theorems, Rice's theorem, and Kleene's normal form theorem (which has an interesting interpretation in connection with programming practice). There is an unusually careful treatment of the mutual encodability of the various basic data structures. But mainly this book will issue once again, and with particular force, the siren call we logicians have been hearing: you too can be a computer scientist!

Martin Davis
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
New York, New York 10012