Texts and Monographs in Computer Science

The AKM Series in
Theoretical Computer Science

A Subseries of Texts and Monographs in Computer Science

A Basis for Theoretical Computer Science
by M. A. Arbib, A. J. Kfoury, and R. N. Moll

A Programming Approach to Computability
by A. J. Kfoury, R. N. Moll, and M. A. Arbib

An Introduction to Formal Language Theory
by R. N. Moll, M. A. Arbib, and A. J. Kfoury

Algebraic Approaches to Program Semantics
by E. G. Manes and M. A. Arbib

Algebraic Approaches to Program Semantics

Ernest G. Manes
Michael A. Arbib

Springer-Verlag
New York Berlin Heidelberg
London Paris Tokyo

To
Bernadette and Prue

Preface

In the 1930s, mathematical logicians studied the notion of "effective computability" using such notions as recursive functions, λ-calculus, and Turing machines. The 1940s saw the construction of the first electronic computers, and the next 20 years saw the evolution of higher-level programming languages in which programs could be written in a convenient fashion independent (thanks to compilers and interpreters) of the architecture of any specific machine. The development of such languages led in turn to the general analysis of questions of *syntax*, structuring strings of symbols which could count as legal programs, and *semantics*, determining the "meaning" of a program, for example, as the function it computes in transforming input data to output results. An important approach to semantics, pioneered by Floyd, Hoare, and Wirth, is called *assertion* semantics: given a *specification* of which assertions (*preconditions*) on input data should guarantee that the results satisfy desired assertions (*postconditions*) on output data, one seeks a logical proof that the program satisfies its specification. An alternative approach, pioneered by Scott and Strachey, is called *denotational* semantics: it offers algebraic techniques for characterizing the denotation of (i.e., the function computed by) a program—the properties of the program can then be checked by direct comparison of the denotation with the specification.

This book is an introduction to denotational semantics. More specifically, we introduce the reader to two approaches to denotational semantics: the *order semantics* of Scott and Strachey and our own *partially additive semantics*. Moreover, we show how each approach may be applied both to the specification of the semantics of programs, including recursive programs, and to the specification of new data types from old. There has been a growing acceptance that *category theory*, a branch of abstract algebra, provides a perspicuous

general setting for all these topics, and for many other algebraic approaches to program semantics as well. Thus, an important aim of this book is to interweave the study of semantics with a completely self-contained introduction to a useful core of category theory, fully motivated by basic concepts of computer science.

Computer science seeks to provide a scientific basis for the study of information processing, algorithms, and the design and programming of computers. The past four decades have witnessed major advances in programming methodology, which allow immense programs to be designed with increasing speed and reduced error, and in the development of mathematical techniques to allow the rigorous specification of program, process, and machine. The present volume is one of a series, the AKM Series in Theoretical Computer Science, designed to make key mathematical developments in computer science readily accessible to undergraduate and beginning graduate students. The book is essentially self-contained: what little background is required may be found in the AKM volume *A Basis for Theoretical Computer Science.*

However, this book is more algebraic than other books in the AKM Series, and as such may prove somewhat heavier going—at least for American students, since the American curriculum in theoretical computer science, as distinct from the European curriculum, stresses combinatorial methods over algebraic methods.

The book is organized in three parts: Part 1 presents the denotational semantics of control, that is, the way in which the denotation of a program can be obtained from the denotation of the pieces from which it is composed. The approach is motivated by analysis of a fragment of Pascal, a functional programming fragment, and a consideration of nondeterministic semantics. Basic notions of category theory include those of product and coproduct. Chapter 3 presents the elements of partially additive semantics, including a denotational semantics of iteration and a new theory of guards ("test functions") which provides a bridge between denotational semantics and the assertion semantics presented in Chapter 4.

Part 2 extends the theory of Part 1 by showing how the Kleene sequence yields a denotation for the computation given by a recursive program. Chapter 6 then introduces *domains* as the setting for the order semantics of recursion, while Chapter 8 provides the partially ordered semantics of recursion. Chapter 7, on canonical fixed points, provides a unified setting for both approaches, as well as for the study of fixed points in metric spaces in Chapter 9.

Part 3 extends the theory to data types. The crucial tools are provided by the following notions from category theory, which are introduced in Chapters 10 and 11: functors, fixed points of functors, and co-continuous and continuous functors. We motivate these with a discussion of how a generalized Kleene sequence can provide the denotation of a recursive specification of a data type. In Chapter 12, we consider parametric specification of data types, analyzing arrays, stacks, queues, and our functional programming fragment

in the process. We devote Chapter 13 to the order semantics of data types. Finally, Chapter 14 gives a brief introduction to describing data types using operations and equations, and extends the earlier theory of functorial fixed points to include these ideas. As a result, the reader is not limited to any one algebraic approach to program semantics, but rather is given the tools to tailor the formal semantics to the need of different applications.

The book grew out of our research in partially additive semantics, which was in turn based on our general investigation of "category theory applied to computation and control." We thank the National Science Foundation for its support of this research. This volume represents an attempt to place the material in the perspective of other approaches to denotational semantics, and to render the common algebraic tools as accessible as possible. We thank our many colleagues in both America and Europe for all they taught us in the course of this research, and for their comments on an earlier draft of the book. It is with regret that we note that limitations of space make it impossible to address all the topics raised in this correspondence within the compass of an introductory text. Finally, we thank Gwyn Mitchell and Kathy Adamczyk for their typing of the draft of this manuscript; and Ms. Adamczyk for helping with research on the notes for Chapter 5.

Amherst, Massachusetts

Ernest G. Manes
Michael A. Arbib

Contents