Preface

This book is written out of a tradition that places special emphasis on the following three approaches to semantics:

– operational semantics,

– denotational semantics, and

– axiomatic semantics.

It is therefore beyond the scope of this introductory book to cover other approaches such as algebraic semantics, game semantics, and evolving algebras.

We strongly believe that semantics has an important role to play in the future development of software systems and domain-specific languages (and hence is not confined to the enormous task of specifying "real life" languages such as C++, Java or C#). We have therefore found the need for an introductory book that

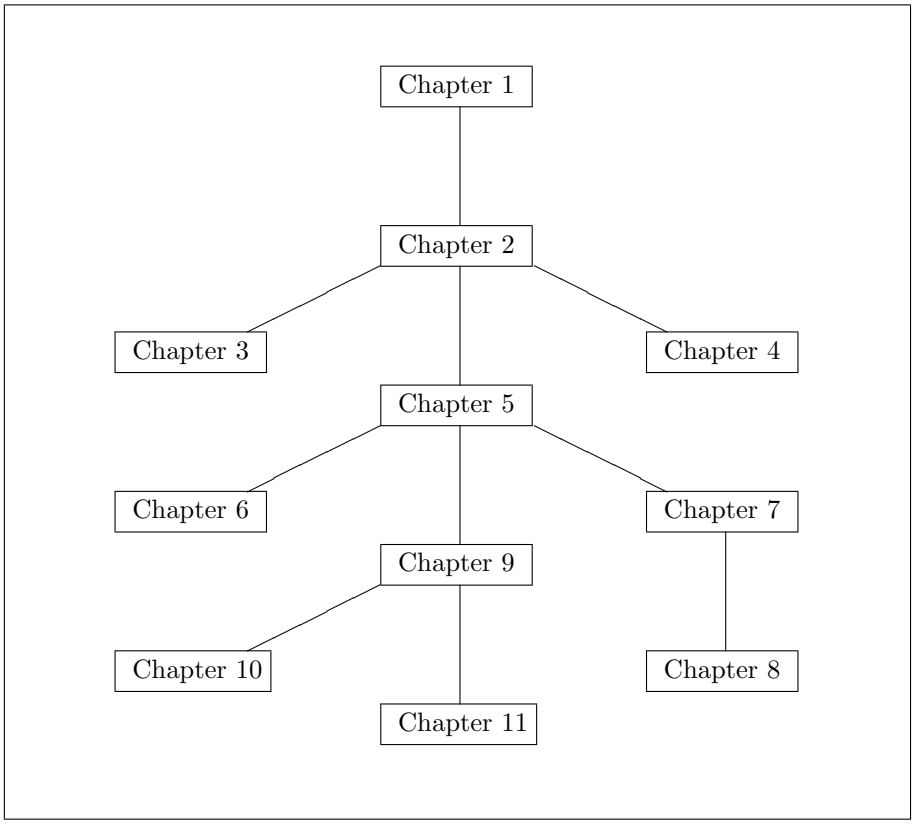
– presents the *fundamental ideas* behind these approaches,

– stresses their *relationship* by formulating and proving the relevant theorems, and

– illustrates the *applications* of semantics in computer science.

This is an ambitious goal for an introductory book, and to achieve it, the bulk of the technical development concentrates on a rather small core language of `while`-programs for which the three approaches are developed to roughly the same level of sophistication; this should enable students to get a better grasp of similarities and differences among the three approaches.

In our choice of applications, we have selected some of the historically important application areas as well as some of the more promising candidates for future applications:



– the use of semantics for validating prototype implementations of programming languages;
– the use of semantics for verifying program analyses that are part of more advanced implementations of programming languages;
– the use of semantics for verifying security analyses; and
– the use of semantics for verifying useful program properties, including information about execution time.

Clearly this only serves as an appetizer to the fascinating area of "Semantics with Applications"; some pointers for further reading are given in Chapter 11.

Overview. As is illustrated in the dependency diagram, Chapters 1, 2, 5, 9, and 11 form the core of the book. Chapter 1 introduces the example language **While** of `while`-programs that is used throughout the book. In Chapter 2 we cover two approaches to *operational semantics*, the natural semantics of

G. Kahn and the structural operational semantics of G. D. Plotkin. Chapter 5 develops the *denotational semantics* of D. Scott and C. Strachey, including simple fixed point theory. Chapter 9 introduces *program verification* based on operational and denotational semantics and goes on to present the *axiomatic approach* due to C. A. R. Hoare. Finally, Chapter 11 contains suggestions for further reading. Chapters 2, 5, and 9 are devoted to the language **While** and cover specification as well as theory; there is quite a bit of attention to the proof techniques needed for proving the relevant theorems.

Chapters 3, 6, and 10 consider extensions of the approach by incorporating new descriptive techniques or new language constructs; in the interest of breadth of coverage, the emphasis is on specification rather than theory. To be specific, Chapter 3 considers extensions with abortion, non-determinism, parallelism, block constructs, dynamic and static procedures, and non-recursive and recursive procedures. In Chapter 6 we consider static procedures that may or may not be recursive and we show how to handle exceptions; that is, certain kinds of jumps. Finally, in Section 10.1 we consider non-recursive and recursive procedures and show how to deal with total correctness properties.

Chapters 4, 7, 8, and 10 cover the applications of operational, denotational, and axiomatic semantics to the language **While** as developed in Chapters 2, 5, and 9. In Chapter 4 we show how to prove the correctness of a simple compiler using the operational semantics. In Chapter 7 we show how to specify and prove the correctness of a program analysis for "Detection of Signs" using the denotational semantics. Furthermore, in Chapter 8 we specify and prove the correctness of a security analysis once more using the denotational semantics. Finally, in Section 10.2 we extend the axiomatic approach so as to obtain information about execution time.

Appendix A reviews the mathematical notation on which this book is based. It is mostly standard notation, but some may find our use of $\hookrightarrow$ and $\diamond$ non-standard. We use $D \hookrightarrow E$ for the set of *partial* functions from D to E; this is because we find that the $D \rightharpoonup E$ notation is too easily overlooked. Also, we use $R \diamond S$ for the composition of binary relations R and S. When dealing with axiomatic semantics we use formulae $\{\ P\ \}\ S\ \{\ Q\ \}$ for partial correctness assertions but $\{\ P\ \}\ S\ \{\ \Downarrow Q\ \}$ for total correctness assertions, hoping that the explicit occurrence of $\Downarrow$ (for termination) may prevent the student from confusing the two systems.

Appendix B contains some fairly detailed results for calculating the number of iterations of a functional before it stabilises and produces the least fixed point. This applies to the functionals arising in the program analyses developed in Chapters 7 and 8.

Notes for the instructor. The reader should preferably be acquainted with the BNF style of specifying the syntax of programming languages and should be familiar with most of the mathematical concepts surveyed in Appendix A.

We provide two kinds of exercises. One kind helps the student in understanding the definitions, results, and techniques used in the text. In particular, there are exercises that ask the student to prove auxiliary results needed for the main results but then the proof techniques will be minor variations of those already explained in the text. We have marked those exercises whose results are needed later by "Essential". The other kind of exercises are more challenging in that they extend the development, for example by relating it to other approaches. We use a star to mark the more difficult of these exercises. Exercises marked by two stars are rather lengthy and may require insight not otherwise presented in the book. It will not be necessary for students to attempt all the exercises, but we do recommend that they read them and try to understand what the exercises are about. For a list of misprints and supplementary material, please consult the webpage `http://www.imm.dtu.dk/~riis/SWA/swa.html`.

Acknowledgments. This book grew out of our previous book *Semantics with Applications: A Formal Introduction* [18] that was published by Wiley in 1992 and a note, *Semantics with Applications: Model-Based Program Analysis*, written in 1996. Over the years, we have obtained many comments from colleagues and students, and since we are constantly reminded that the material is still in demand, we have taken this opportunity to rework the book. This includes using shorter chapters and a different choice of security-related analyses. The present version has benefitted from the comments of Henning Makholm.

Kongens Lyngby, Denmark, January 2007 Hanne Riis Nielson

Flemming Nielson